ANNEXE 3 ; LES NORMES DEDEVELOPPEMENT DE LA TMA

Région
Provence-Alpes-Côte d'Azur

Direction des Systèmes d'Information
Service Applications et Données

Tierce Maintenance Applicative
Normes de développement .NET
-
Annexe au C.C.A.T.P.

Sommaire

1 INTRODUCTION

Ce document a pour objet de décrire les normes de développement définies par la Région Provence-Alpes-Côte d'Azur, pour le développement de ses application .net.

La plate forme technique d'utilisation de l'environnement .NET est décomposée en deux versions :

- Une première version dite Framework 2.0, pour laquelle deux normes de développement ont été définies :
  - Norme 1.00 : Visual Studio 2005 – Dataset
  - Norme 1.01 : Visual Studio 2005 – Objet
- Une deuxième version dite Framework 4.0 adoptée en octobre 2010, avec une norme de développement 2.00

A chaque norme est associée une liste de documents.

	Norme 1.00	Norme 1.01	Norme 2.00
Architecture de développement	**Document 1 – 1.00**	**Document 1 – 1.01**	**Document 1 – 2.00**
Règles et recommandations	**Document 2 – 1.00**	Document 2 – 1.00	**Document 2 – 2.00**
C# - Règles de nommage et recommandations	**Document 3 – 1.00**	**Document 3 – 1.01**	**Document 3 – 2.00**
Accès aux bases de données Oracle	**Document 4 – 1.00**	Document 4 – 1.00	Document 4 – 1.00
Socle technique	**Document 5 – 1.00**	**Document 5 – 1.01**	
Ergonomie de l'interface client	**Document 6 – 1.00**	**Document 6 – 1.01**	**Document 6 – 2.00**
Mise en production des applications	**Document 7 – 1.00**	Document 7 – 1.00	Document 7 – 1.00
Authentification AD	**Document 8 – 1.00**	Document 8 – 1.00	Document 8 – 1.00
Règles et recommandations TFS			**Document 9 – 2.00**
Guide de gestion de la configuration			**Document 10 – 2.00**
Accessibilité en ASP.Net – Document de référence			**Document 11 – 2.00**
Guide d'implémentation de l'accessibilité en Framework .Net			**Document 12 – 2.00**

2 DOCUMENT 1 – 1.00 - ARCHITECTURE DE DEVELOPPEMENT

2.1 Introduction

Le présent document constitue l'architecture technique et la configuration de développement utilisées par la Région PACA et les prestataires de service développant pour la Région PACA, dans le cadre de leurs projets utilisant le langage C# au travers de l'outil de développement Visual Studio .NET

Les points abordés dans ce document concernent :

- L'architecture technique de développement,
- Le framework,
- Les modes de déploiement des composants serveurs,
- L'arborescence des répertoires

2.2 Architecture de développement

2.2.1 Choix de l'architecture

Plusieurs modèles d'architecture de développement s'offraient à la Région PACA. Elles étaient au nombre de trois :

- Non distribuée
- Distribuée en .net remoting
- Distribuée en web services

Même si ce modèle est plus couteux et plus complexe en développement, la Région PACA a choisi l'architecture distribuée en .net remoting pour les 3 raisons suivantes :

- Performance : possibilité de répartir les charges sur des serveurs différents en cas de problème de performance, tout en ayant des clients les moins lourds possibles.
- Compatibilité entre les composants métier clients riches et client web
- Installation sur le poste de l'utilisateur des applications « Client riche » du framework seulement. Pas d'installation de client Oracle, par exemple.

Nota bene : la solution distribuée en web services n'a pas été choisie car les applications de la Région PACA ne devraient a priori pas être déployées sur des serveurs présents sur des réseaux différents.

Schéma pour un client riche

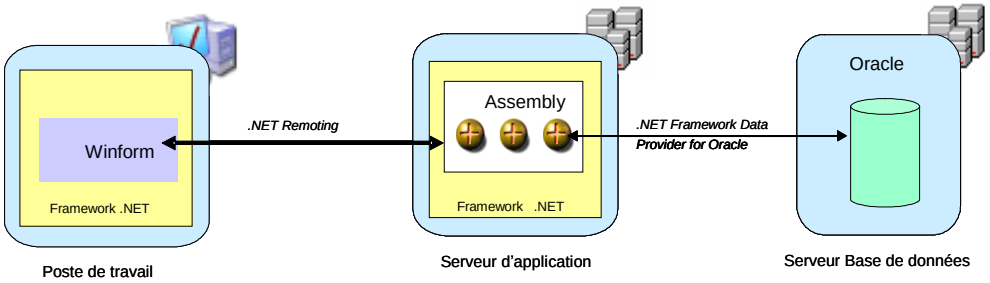
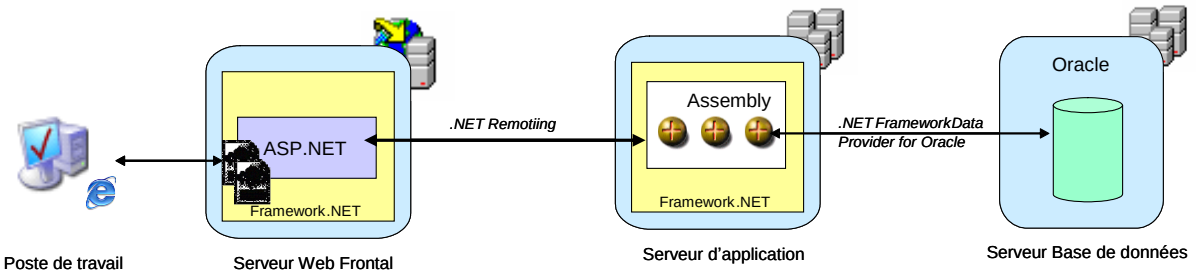
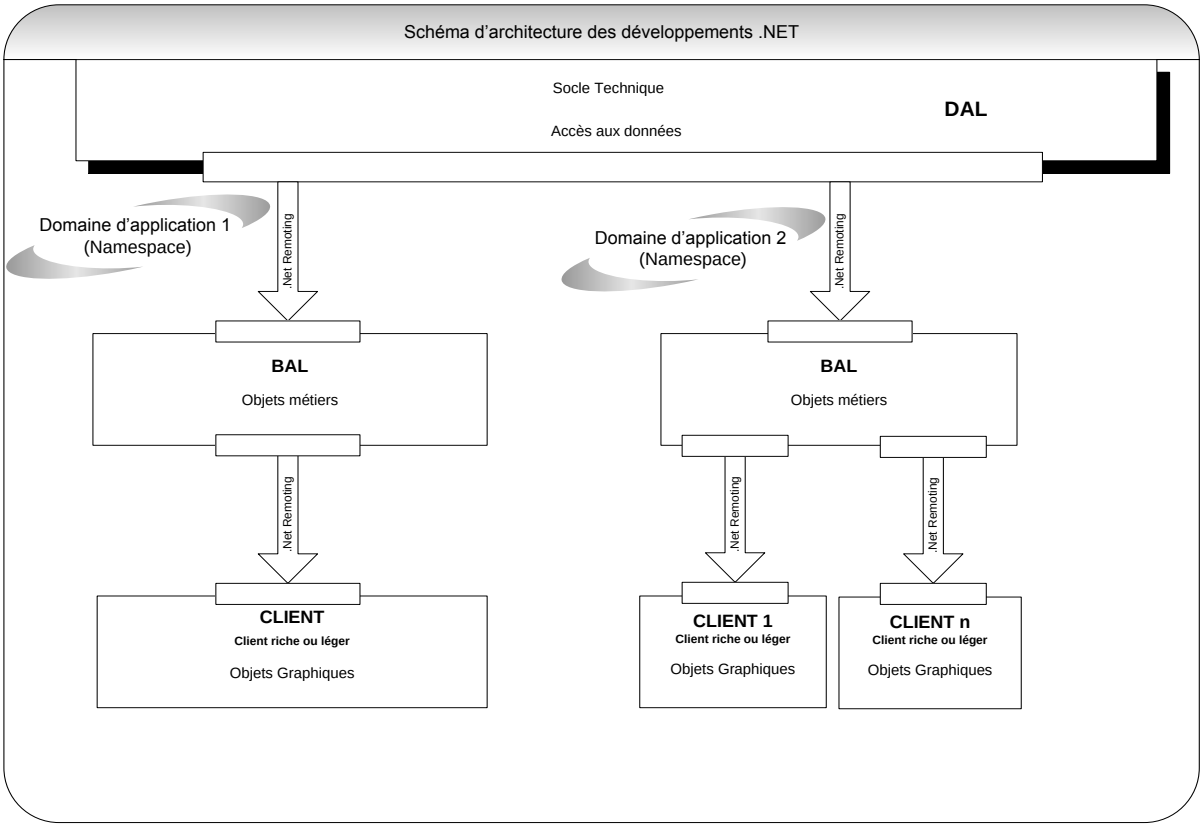


Schéma pour un client léger



2.2.2 Schéma d'architecture global



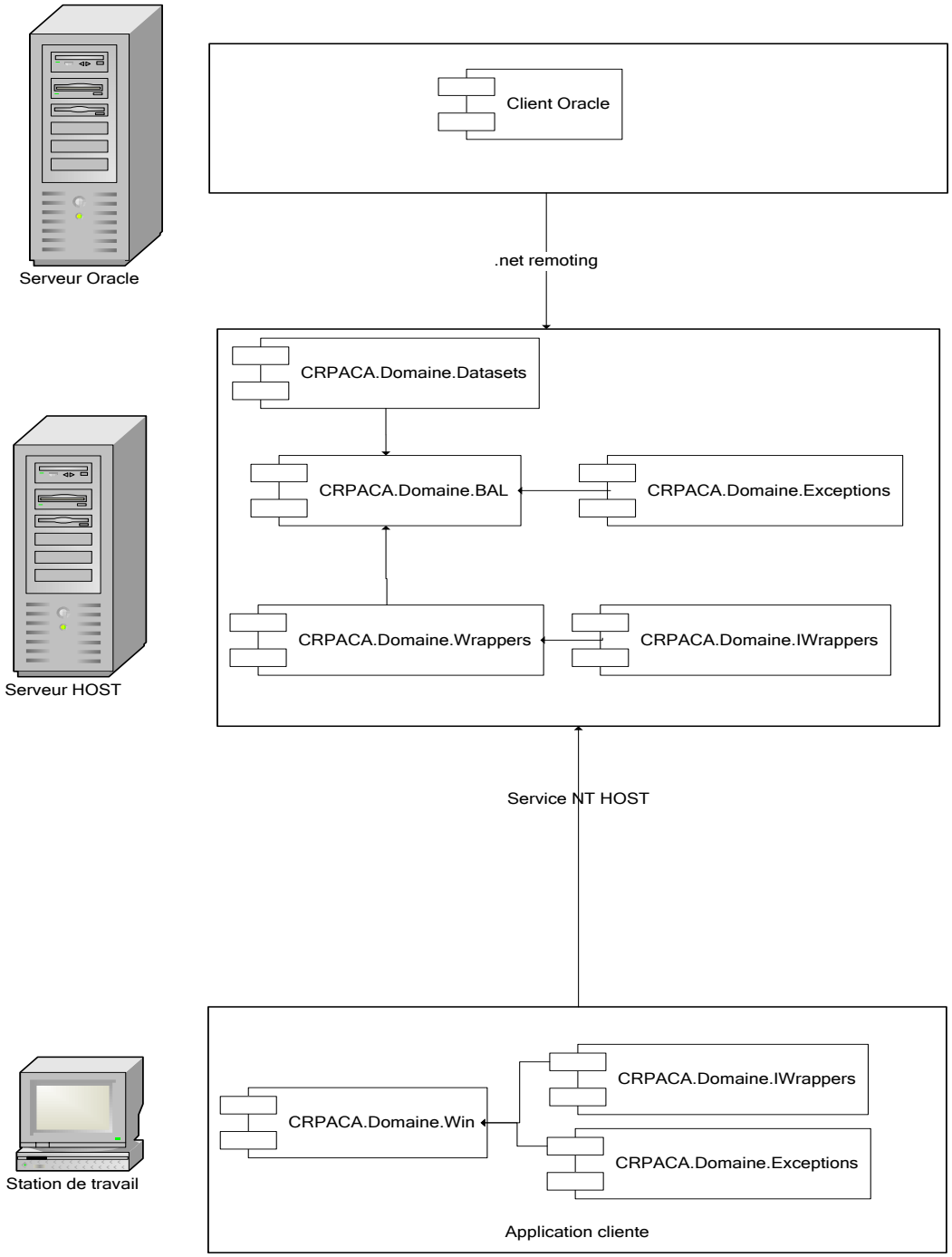
L'architecture technique employée est une architecture 3-tiers à base de client riche .NET :

- Un serveur de données hébergeant des bases Oracle,
- Un serveur applicatif en charge de l'hébergement des composants exécutant la logique métier,
- Des postes clients consommateurs des services exposés par le serveur applicatif.

Dans la suite du document, le serveur applicatif sera appelé Host, les postes de travail seront appelés Client(s).

2.2.3 Schéma d'architecture détaillé & vue des composants

2.2.3.1 Schéma d'architecture détaillé



2.2.3.2 Serveur de données

Le serveur de données expose une base de données Oracle dédiée à l'application.

Il n'y a aucun déploiement de composants .NET sur ce serveur, ni framework, ni composants applicatif.

2.2.3.3 Serveur applicatif Host

Le serveur applicatif exécute sous forme d'un service NT une application Host, qui expose les différents composants applicatifs de l'application aux clients.

Ces composants sont exposés via la technologie .NET Remoting, au moyen d'un fichier de configuration spécifique.

En pratique le serveur héberge les assemblies suivantes du projet :

- CRPACA.Domaine.Datasets : Cette assembly contient l'ensemble des schéma XSD des datasets typés
- CRPACA. Domaine.BAL : « Business Access Layer ». Cette assembly contient l'ensemble des objets métiers (classe .NET custom) et expose les différents cas d'utilisation de l'application sous forme de classes hébergeant la logique applicative et orchestrant les objets en fonction des besoins.
- CRPACA.Domaine.Wrappers : « Wrappers ». Les wrappers sont des classes adaptatrices, qui encapsulent les méthodes des classes BAL et les exposent aux clients par la technologie .NET Remoting. Concrètement, chaque classe héritent de MarshalByRefObject (classe du framework assurant le passage par référence d'un objet du serveur au client, notamment utile pour implémenter .NET Remoting) et implémente l'interface IWrappers associée.

De façon transverse, on trouve également les assemblies

- CRPACA.Domaine.Exceptions : contient l'ensemble des exceptions applicatives dédiées à l'application. Ces exceptions sont utilisées pour gérer les erreurs fonctionnelles de l'application, en réponse à une action dans un Use Case (par exemple erreur « CFA déjà utilisé dans un projet » => ne peut pas être supprimé). Chaque exception doit être sérializable afin d'être transférable vers le client.
- CRPACA.Domaine.IWrappers : interfaces définissant les contrats que doivent implémenter les Wrappers.

2.2.3.4 Postes clients

Les clients consomment les services Wrappers exposés par le Host. Aucune logique applicative n'est donc réellement implémentée côté client puisque les règles de gestion sont à la charge des composants BAL implémentant les UseCases côté serveur.

La partie cliente est donc composée des assemblies :

- CRPACA.Domaine.Exceptions : idem que côté serveur. Ceci permet au client de connaître les exceptions émises côté serveur et de pouvoir les traiter en conséquence.
- CRPACA.Domaine.IWrappers : idem que côté serveur. L'objectif est de pouvoir fournir au client non pas une implémentation des Wrappers, qu'il n'utilisera pas puisqu'il les consomme à distance, mais une simple vue sur les interfaces. Ceci permet de faire évoluer les règles métiers avec beaucoup moins de contraintes côté serveur (à condition que les Wrappers continuent de respecter la même interface).
- CRPACA.Domaine.Win : exécutable Winforms (client riche) présentant les IHM à l'utilisateur et invoquant, suivant les actions de l'utilisateur, les composants applicatifs du Host par .NET Remoting.

2.3 Framework, langage préconisé et outil de reporting

Actuellement, la version de l'outil de développement Visual Studio .NET est Visual Studio 2003 avec le framework v1.1.4322 et le langage C#.

Dès le passage à Visual Studio 2005, le framework utilisé sera v2.0.50727.

L'outil de reporting préconisé est « Reporting Services » version SQL 2000, puis SQL 2005 depuis décembre 2006.

2.4 Mode de déploiement des composants serveurs

Les composants serveurs sont invoqués via la technologie .NET Remoting au travers du réseau. Le composant Host, qui embarque les différents composants serveurs, est exposé sous forme de Services Windows sur un serveur. Il est donc nécessaire de réaliser ce Service Windows, puis de l'installer sur la machine.

2.4.1 Réalisation d'un service Windows

La réalisation du service se fait sous Visual Studio .NET. Un tutorial très simple est disponible à l'URL suivante : http://www.codeguru.com/Csharp/Csharp/cs_network/windowsservices/article.php/c6027/

La procédure à suivre consiste à :

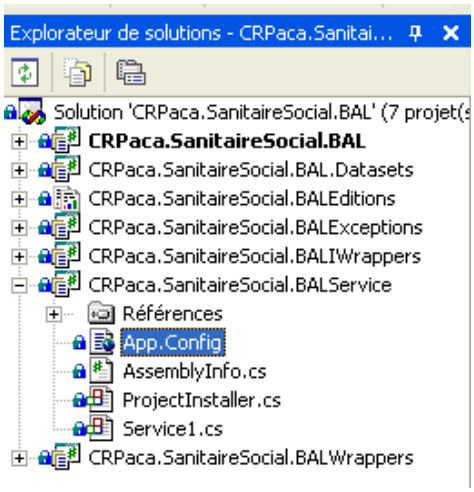
- Dans la solution, ajouter un nouveau projet C# de type Service Windows.
- Importer une nouvelle référence sur le projet Wrappers, qui embarque l'ensemble des services,
- Ajouter le fichier de configuration de l'application, qui comporte les paramètres applicatifs techniques (répertoires pour les éditions etc…) et les paramètres liés à l'exposition de service sous .NET Remoting
- Dans le fichier <nom_du_projet>.cs (fichier généré automatiquement à la création du projet et qui constitue le point d'entrée du service), il faut implémenter l'événement On_Start, correspondant au démarrage du service par le serveur : la seule chose nécessaire à faire est de lancer la configuration du Remoting, en s'appuyant sur le fichier de configuration, de la façon suivante :

```
/// <summary>
/// Démarrage du service.
/// </summary>
protected override void OnStart(string[] args)
{
    //chargement de la configuration pour le REMOTING
    RemotingConfiguration.Configure(System.Reflection.Assembly.GetExecutingAssembly().Location+".config");
}
```

2.4.2 Ajout d'un Installer au service

Une fois ceci fait, le service est prêt à être exécuté. Cependant, il est nécessaire de l'installer via l'utilitaire du framework InstallUtil.exe. Pour le rendre installable, il faut rajouter un Installer au projet :

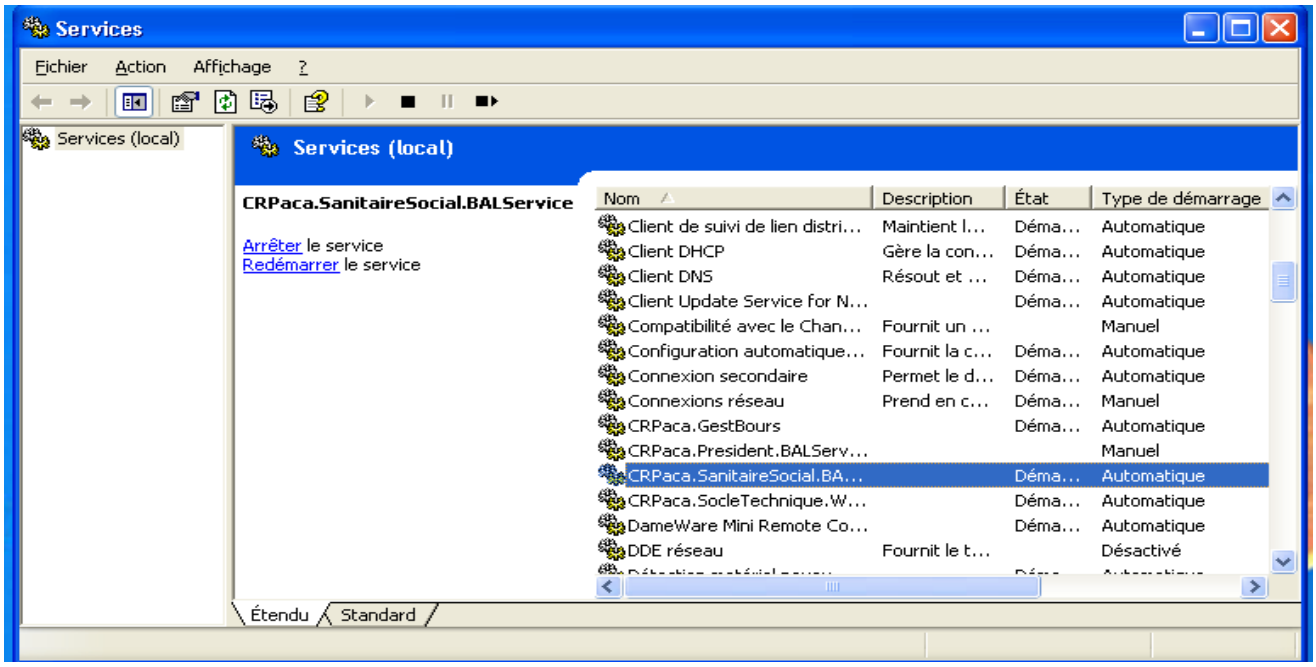
- Dans la vue en mode Design du fichier <nom_du_projet>.cs, cliquer bouton droit et sélectionner « Ajouter le programme d'installation » : un fichier ProjectInstaller.cs est généré
- Dans ce fichier paramétrer le composant ServiceProcessInstaller1 (Account = LocalSystem)
- Paramétrer le composant ServiceInstaller1 (Service Name = <nom du service> cad le nom qui sera affiché sous la console de gestion des services Windows de la machine / StartType = Automatic : le service est lancé dès le démarrage de la machine).



2.4.3 Installation du service sur la machine

Pour installer le service, compiler le projet puis :

- Se placer sur un Command Prompt .NET
- Se positionner sur le répertoire de génération de l'exécutable du projet
- Saisir Installutil.exe <nom_du_projet>.exe pour installer le service



Le service est installé. Pour le désinstaller, saisir Installutil <nom_du_projet>.exe /u.

Son démarrage n'est pas automatique à l'installation : il est donc nécessaire de le lancer la première fois.

2.5 Règles et recommandations d'arborescence

Ces règles et recommandations visent à améliorer la lisibilité des projets utilisant l'outil de développement Visual Studio .NET et Visual Source Control et permettre la mise en œuvre commune d'une charte accessible et utilisée par l'ensemble d'une équipe de développement.

Visual Source Safe permet de partager les sources en cours de développement entre tous les développeurs. Ce produit propose de stocker dans une base de donnée sur un serveur, les solutions et les projets créés via Visual Studio .net et permet ainsi le partage des codes sources parmi toute l'équipe. Cette organisation nécessite une arborescence de fichiers cohérente et stable afin de faciliter, entre autre, le déploiement des applications.

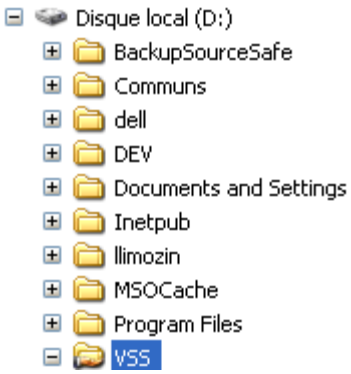
Cette organisation générale de l'espace de travail se scinde en 3 parties :

- Organisation du répertoire de contrôle de code source,
- Organisation du répertoire de développement,
- Organisation des répertoires de déploiement.

2.5.1 Organisation du répertoire de contrôle de code source

2.5.1.1 Répertoire de contrôle de code source

La base de données Visual Source Safe doit être stockée dans un répertoire partagé « VSS » sur une machine réseau CR-PACA. Ce répertoire sera accessible par les autres plates formes de développement via un lecteur réseau.

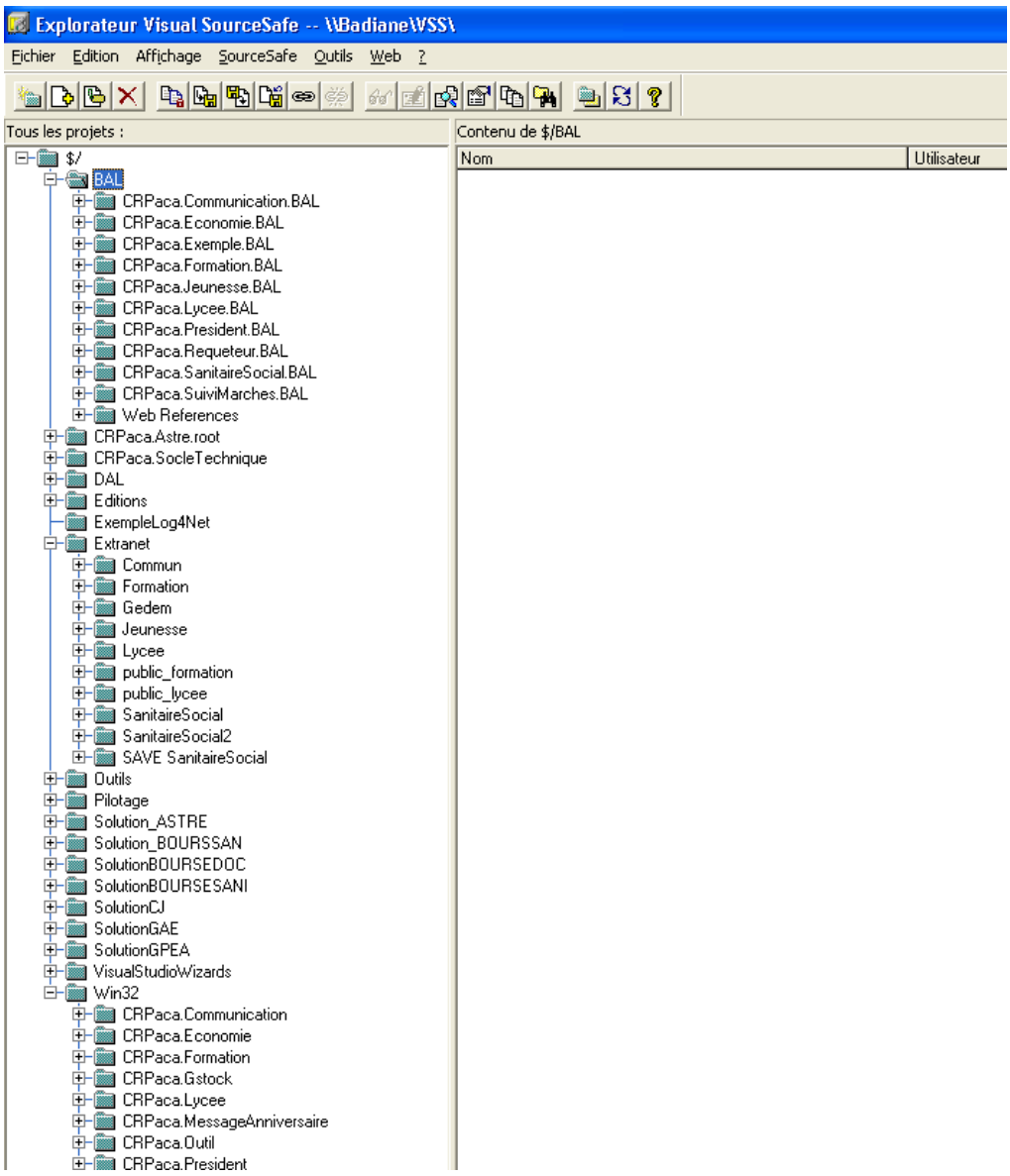


Le répertoire VSS contient la base de données de tous les développements de CR-PACA :

Nom	Taille	Type	Date de modification
data		Dossier de fichiers	29/07/2004 17:27
temp		Dossier de fichiers	30/07/2004 10:25
users		Dossier de fichiers	29/07/2004 17:24
srcsafe.ini	2 Ko	Paramètres de confi...	26/05/2004 11:22
users.txt	1 Ko	Document texte	29/07/2004 17:24

Chaque développeur ayant créé un lecteur réseau qui pointe sur VSS peut se connecter à la base de donnée Visual Source Safe « CRPaca » (uniquement s'il a l'autorisation de se connecter avec Visual Source Safe) à partir du fichier « srcsafe.ini » pour initialiser sa solution de développement pour ainsi récupérer et/ou sauver le code qu'il produit.

2.5.1.2 Organisation l'arborescence dans Visual Source Safe



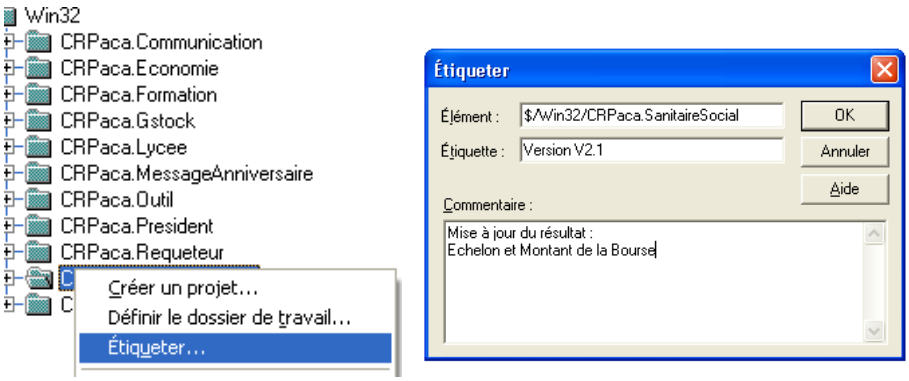
On distingue principalement les solutions :

- CRPaca.BAL qui comprend l'ensemble des composants métier d'accès aux données, pour chaque domaine d'application.
- CRPaca.SocleTechnique qui comprend les projets CRPaca.SocleTechnique.Dal, composants de la couche d'accès aux données, CRPaca.SocleTechnique.UI composants de la couche présentation, et CRPaca.SOcleTechnique.Util composants factorisables pouvant être utiliser par l'ensemble des applications développées.
- CRPaca.Extranet qui comprend l'ensemble des applications Web par domaine d'application.
- CRPaca.Commun placé dans le répertoire logique Visual Source Safe « extranet » qui comprend le projet Commun réunissant l'ensemble des ressources nécessaires à la charte graphique des sites web, les scripts, ainsi que les images pour la mise en page.
- CRPaca.Win qui comprend l'ensemble des applications clients riches par domaine d'application.

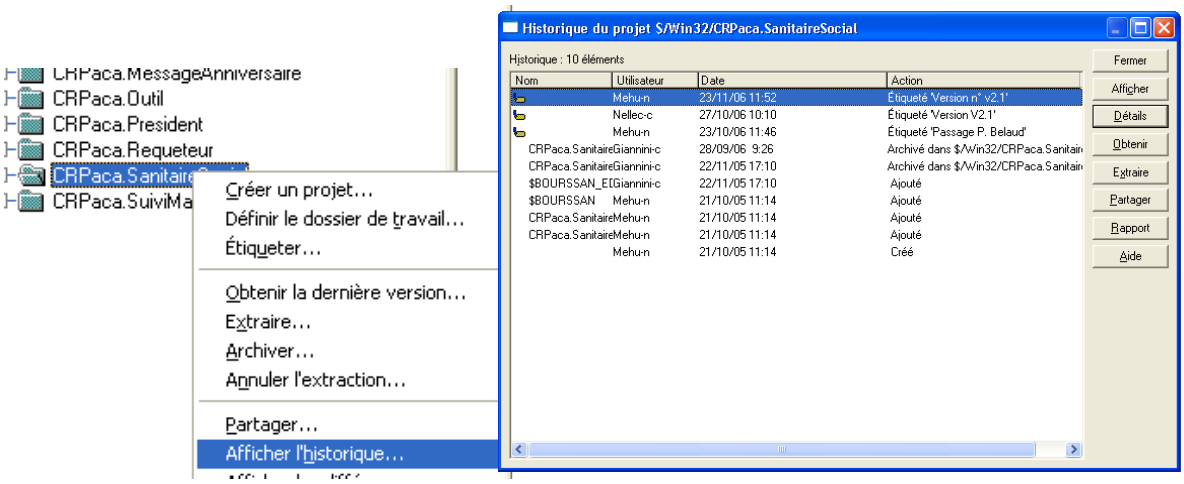
2.5.1.3 Etiquetage des versions dans Visual Source Safe

Lorsque des modifications importantes sont apportées à une application, il faut préciser la version et préciser les modifications apportées en étiquetant le projet.

Pour ce faire, dans Source Safe, il faut :



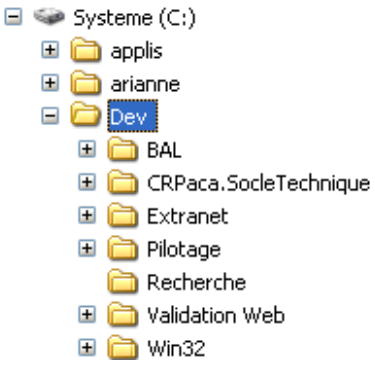
Pour visualiser les différentes versions :



2.5.2 Organisation du répertoire de développement

2.5.2.1 Généralités sur la plateforme de développement

L'arborescence des répertoires de développement doit être le même sur chaque plate forme de développement (poste du développeur).



2.5.2.2 Applications Client riche (Windows forms)

Les projets Clients riches sont situés dans le répertoire

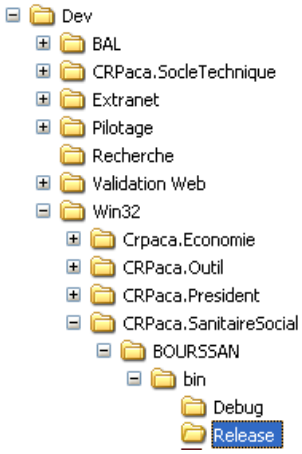
C:/Dev/Win32/

Avec un sous répertoire par domaine d'application.

2.5.2.3 Librairies de classe d'accès aux données pour les applications Client Riche

Les libraires de classes d'accès aux données pour les applications Clients riches se situent dans le répertoire

C:/dev/Win32/CRPaca.« Domaine »/«Application»/bin



2.5.2.4 Applications Web

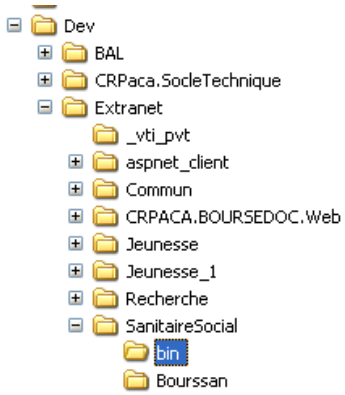
Les projets Web sont situés dans le répertoire

C:/Dev/Extranet/ correspondant au répertoire de référence du serveur IIS de chaque machine.

2.5.2.5 Librairies de classe d'accès aux données pour les applications Web

Les libraires de classes d'accès aux données pour les applications Web se situent dans le répertoire

C:/dev/Extranet/ « Domaine »/bin



2.5.3 Organisation du répertoire de déploiement en production

2.5.3.1 Applications Clients riches

Le serveur de production se nomme [redacted]. Les programmes source doivent être stockés sur :

\\[redacted]\applis\progs\Specif\.« Domaine »\«Application»\ Client

2.5.3.2 Applications Web

Le serveur de production se nomme [redacted]. Les programmes source doivent être sur le disque D:\

Les applications web seront copiées dans le répertoire :

« D:\appli\ wwwroot\ »

3 DOCUMENT 1 – 1.01 - ARCHITECTURE DE DEVELOPPEMENT

3.1 Introduction

Le présent document constitue l'architecture technique et la configuration de développement utilisées par la Région PACA et les prestataires de service développant pour la Région PACA, dans le cadre de leurs projets utilisant le langage C# au travers de l'outil de développement Visual Studio .NET

Les points abordés dans ce document concernent :

- L'architecture technique de développement,
- Le framework,
- Les modes de déploiement des composants serveurs,
- L'arborescence des répertoires

3.2 Architecture de développement

3.2.1 Choix de l'architecture

Plusieurs modèles d'architecture de développement s'offraient à la Région PACA. Elles étaient au nombre de trois :

- Non distribuée
- Distribuée en .net remoting
- Distribuée en web services

Même si ce modèle est plus couteux et plus complexe en développement, la Région PACA a choisi l'architecture distribuée en .net remoting pour les 3 raisons suivantes :

- Performance : possibilité de répartir les charges sur des serveurs différents en cas de problème de performance, tout en ayant des clients les moins lourds possibles.
- Compatibilité entre les composants métier clients riches et client web
- Installation sur le poste de l'utilisateur des applications « Client riche » du framework seulement. Pas d'installation de client Oracle, par exemple.

Nota bene : la solution distribuée en web services n'a pas été choisie car les applications de la Région PACA ne devraient a priori pas être déployées sur des serveurs présents sur des réseaux différents.

Schéma pour un client riche

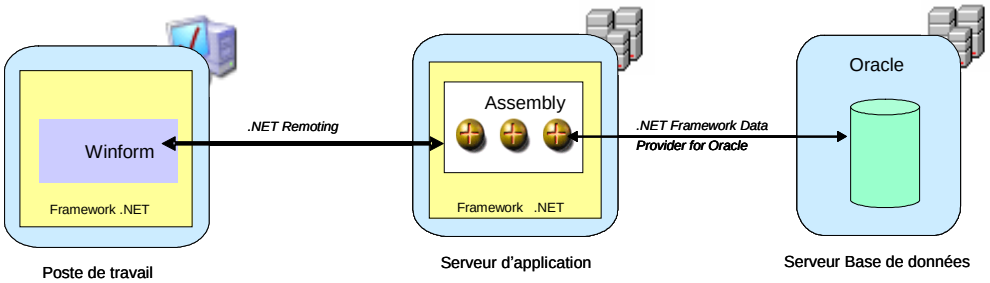
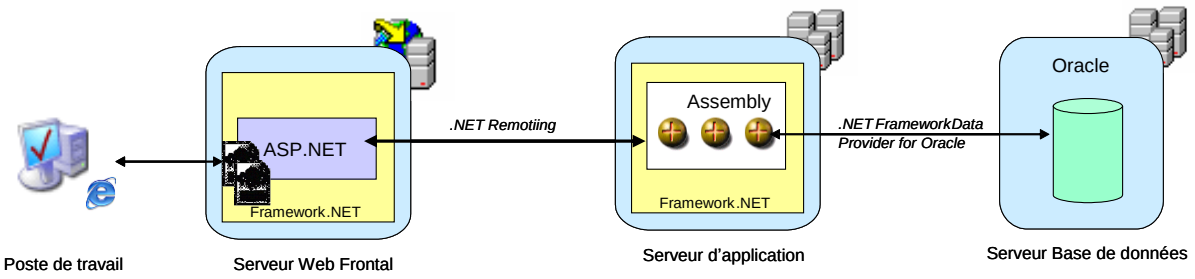
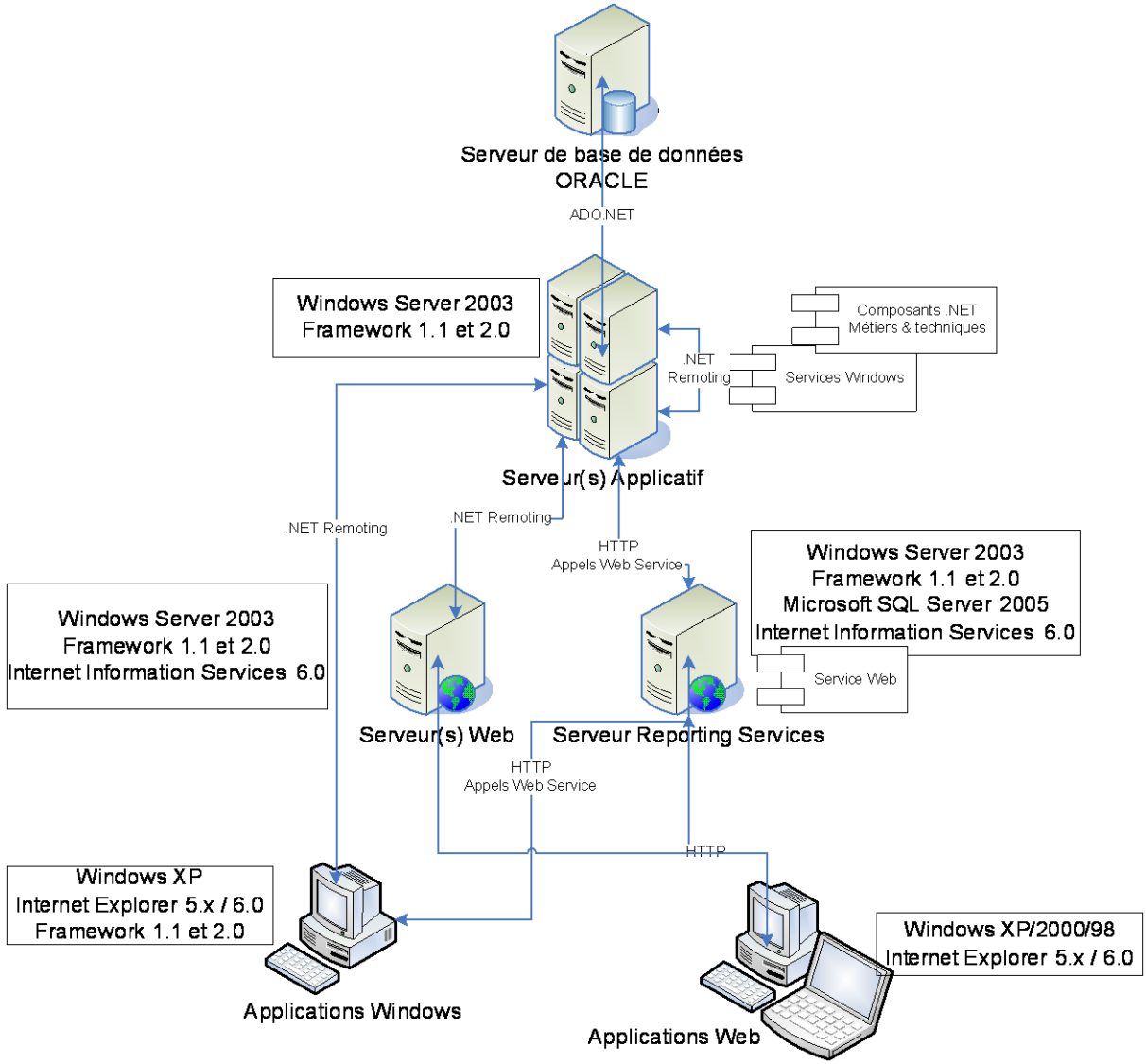


Schéma pour un client léger



Le schéma ci-dessous représente de manière détaillée l'architecture technique du panel d'applications du Conseil Régional PACA :



Ce schéma précise les points fondamentaux de l'architecture suivants :

Le serveur de base de données ORACLE est accédé au travers du socle technique appelé DAL dans le découpage logique (et au travers d'ADO.NET).

Ce composant technique d'accès aux données est « remoté » donc accessible à distance si nécessaire.

Le serveur Microsoft Reporting Services est accessible aussi bien au travers d'une interface Web qu'au travers d'un service Web. Dans les deux cas, l'appel est effectué au travers du protocole HTTP.

Tous les composants « métiers » hébergé sur un serveur applicatif sont « remotés » afin d'être accessible à distance.

Dans le cadre des applications Web :

l'accès aux applications s'effectue à l'aide d'un poste disposant simplement d'un navigateur web dernière génération sans contrainte particulière.

la couche de présentation précisée plus haut est prise en charge par le serveur Web

le serveur Web s'appuie sur les composants « métiers » se trouvant éventuellement sur un autre serveur et y accède au travers de .NET Remoting

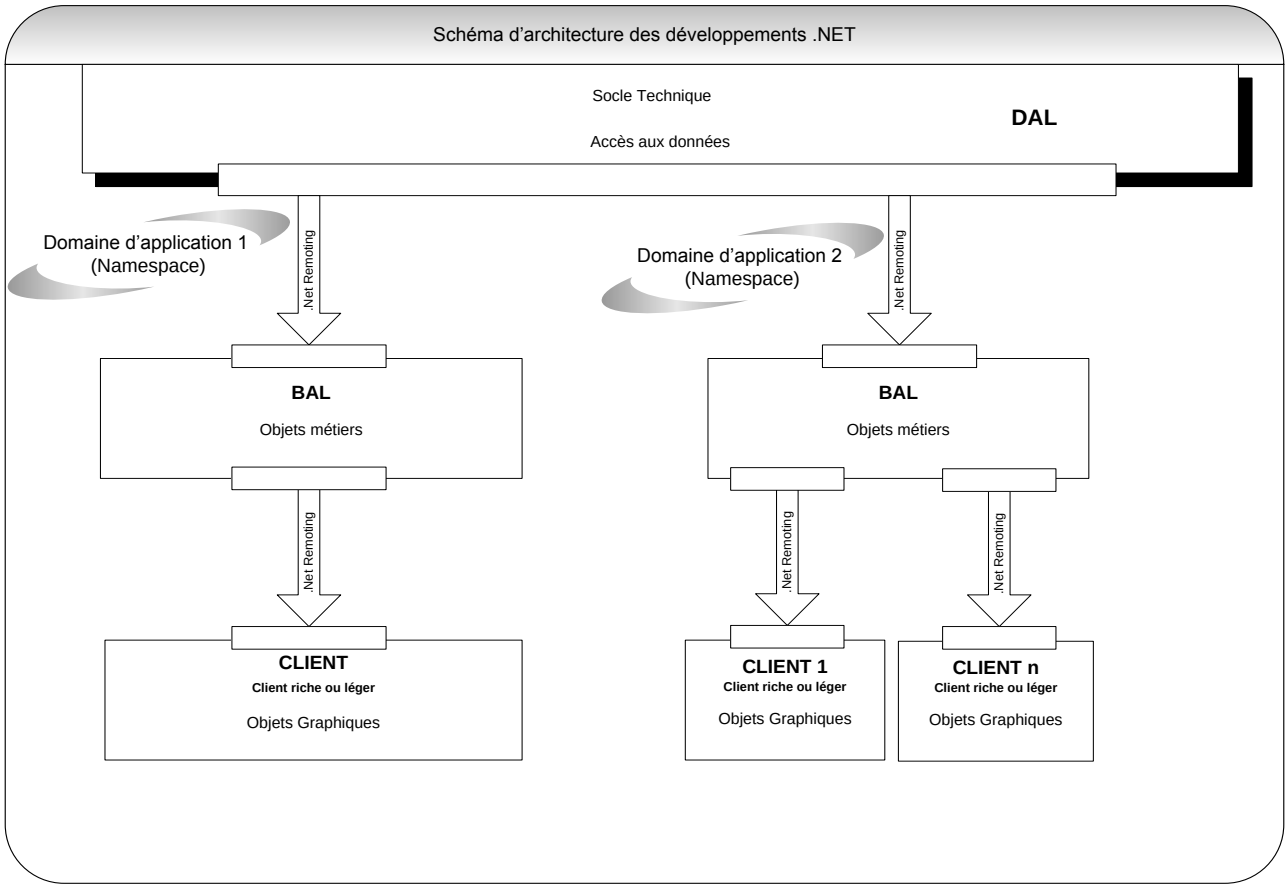
Dans le cadre des applications Windows :

La couche de présentation est prise en charge par une application Windows localement déployée sur le poste.

L'application locale appelle les composants « métiers » distants au travers de .NET Remoting.

Les appels à des éditions s'effectuent à l'aide du service Web de Microsoft Reporting Services.

3.2.2 Schéma d'architecture global



L'architecture technique employée est une architecture 3-tiers à base de client riche .NET :

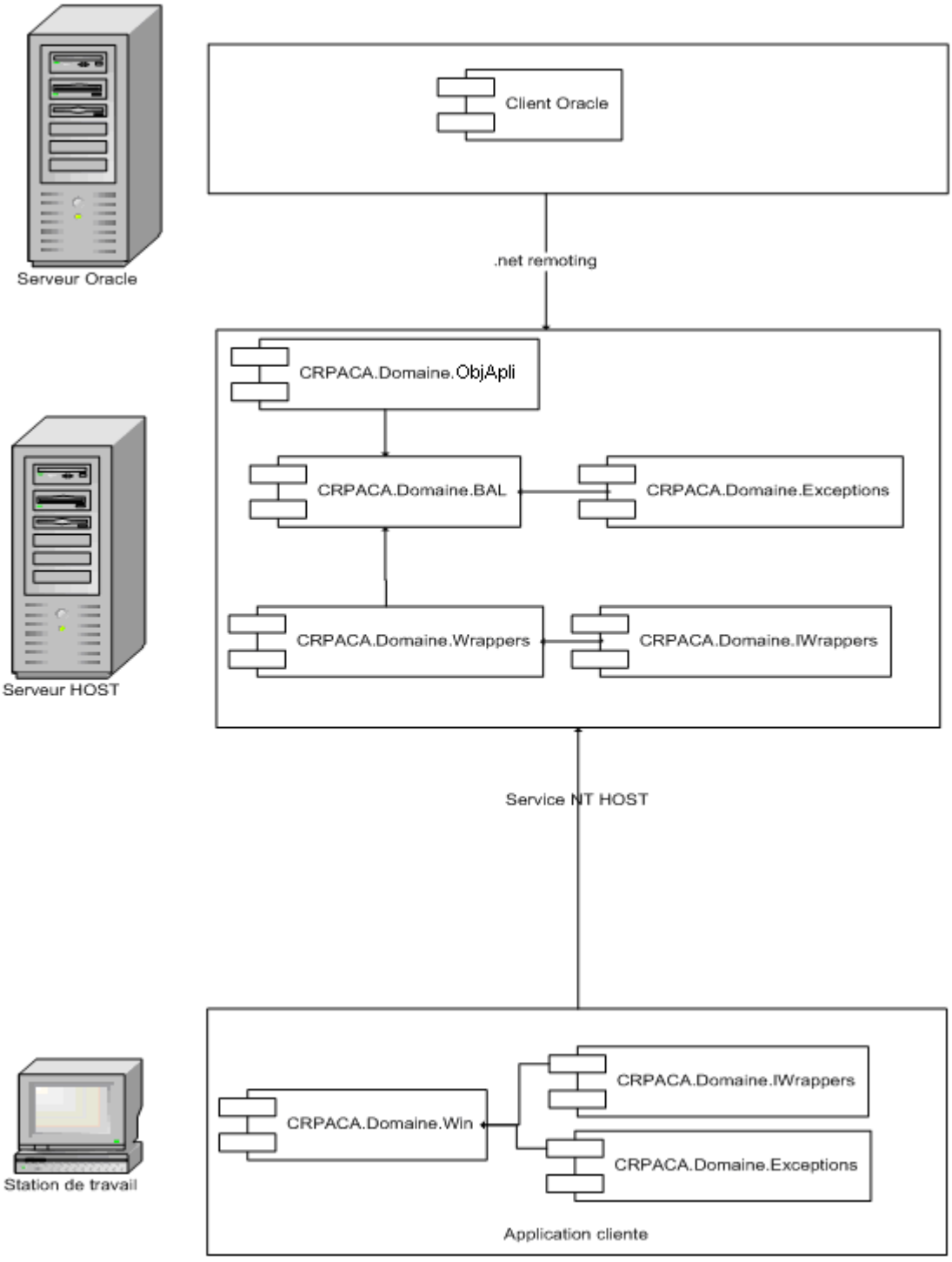
- Un serveur de données hébergeant des bases Oracle,

- Un serveur applicatif en charge de l'hébergement des composants exécutant la logique métier,
- Des postes clients consommateurs des services exposés par le serveur applicatif.

Dans la suite du document, le serveur applicatif sera appelé Host, les postes de travail seront appelés Client(s).

3.2.3 Schéma d'architecture détaillé & vue des composants

3.2.3.1 Schéma d'architecture détaillé



3.2.3.2 Serveur de données

Le serveur de données expose une base de données Oracle dédiée à l'application.

Il n'y a aucun déploiement de composants .NET sur ce serveur, ni framework, ni composants applicatif.

3.2.3.3 Serveur applicatif Host

Le serveur applicatif exécute sous forme d'un service NT une application Host, qui expose les différents composants applicatifs de l'application aux clients.

Ces composants sont exposés via la technologie .NET Remoting, au moyen d'un fichier de configuration spécifique.

En pratique le serveur héberge les assemblies suivantes du projet :

- CRPACA.Domaine.ObjApli : Cette assembly contient l'ensemble des objets métier de l'application
- CRPACA. Domaine.BAL : « Business Access Layer ». Cette assembly contient l'ensemble des objets métiers (classe .NET custom) et expose les différents cas d'utilisation de l'application sous forme de classes hébergeant la logique applicative et orchestrant les objets en fonction des besoins.
- CRPACA.Domaine.Wrappers : « Wrappers ». Les wrappers sont des classes adaptatrices, qui encapsulent les méthodes des classes BAL et les exposent aux clients par la technologie .NET Remoting. Concrètement, chaque classe hérite de MarshalByRefObject (classe du framework assurant le passage par référence d'un objet du serveur au client, notamment utile pour implémenter .NET Remoting) et implémente l'interface IWrappers associée.

De façon transverse, on trouve également les assemblies

- CRPACA.Domaine.Exceptions : contient l'ensemble des exceptions applicatives dédiées à l'application. Ces exceptions sont utilisées pour gérer les erreurs fonctionnelles de l'application, en réponse à une action dans un Use Case (par exemple erreur « CFA déjà utilisé dans un projet » => ne peut pas être supprimé). Chaque exception doit être sérializable afin d'être transférable vers le client.
- CRPACA.Domaine.IWrappers : interfaces définissant les contrats que doivent implémenter les Wrappers.

3.2.3.4 Postes clients

Les clients consomment les services Wrappers exposés par le Host. Aucune logique applicative n'est donc réellement implémentée côté client puisque les règles de gestion sont à la charge des composants BAL implémentant les UseCases côté serveur.

La partie cliente est donc composée des assemblies :

- CRPACA.Domaine.Exceptions : idem que côté serveur. Ceci permet au client de connaître les exceptions émises côté serveur et de pouvoir les traiter en conséquence.
- CRPACA.Domaine.IWrappers : idem que côté serveur. L'objectif est de pouvoir fournir au client non pas une implémentation des Wrappers, qu'il n'utilisera pas puisqu'il les consomme à distance, mais une simple vue sur les interfaces. Ceci permet de faire évoluer les règles métiers avec beaucoup moins de contraintes côté serveur (à condition que les Wrappers continuent de respecter la même interface).
- CRPACA.Domaine.Win : exécutable Winforms (client riche) présentant les IHM à l'utilisateur et invoquant, suivant les actions de l'utilisateur, les composants applicatifs du Host par .NET Remoting.

3.3 framework, langage préconisé et outil de reporting

Actuellement, la version de l'outil de développement Visual Studio .NET est Visual Studio 2005 avec le framework v2.0.50727 et le langage C#.

L'outil de reporting préconisé est « Reporting Services » via SQL SERVER 2005.

3.4 Mode de déploiement des composants serveurs

Les composants serveurs sont invoqués via la technologie .NET Remoting au travers du réseau. Le composant Host, qui embarque les différents composants serveurs, est exposé sous forme de Services Windows sur un serveur. Il est donc nécessaire de réaliser ce Service Windows, puis de l'installer sur la machine.

3.4.1 Réalisation d'un service Windows

La réalisation du service se fait sous Visual Studio .NET. Un tutorial très simple est disponible à l'URL suivante : http://www.codeguru.com/Csharp/Csharp/cs_network/windowsservices/article.php/c6027/

La procédure à suivre consiste à :

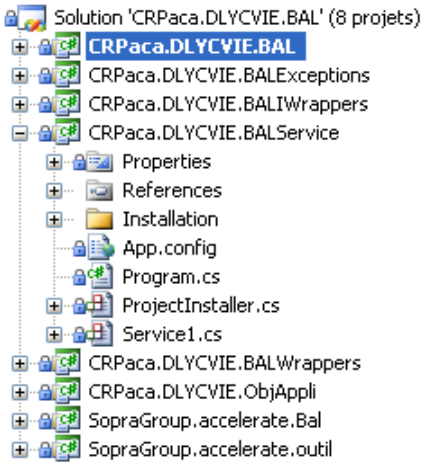
- Dans la solution, ajouter un nouveau projet C# de type Service Windows.
- Importer une nouvelle référence sur le projet Wrappers, qui embarque l'ensemble des services,
- Ajouter le fichier de configuration de l'application, qui comporte les paramètres applicatifs techniques (répertoires pour les éditions etc…) et les paramètres liés à l'exposition de service sous .NET Remoting
- Dans le fichier <nom_du_projet>.cs (fichier généré automatiquement à la création du projet et qui constitue le point d'entrée du service), il faut implémenter l'événement On_Start, correspondant au démarrage du service par le serveur : la seule chose nécessaire à faire est de lancer la configuration du Remoting, en s'appuyant sur le fichier de configuration, de la façon suivante :

```
/// <summary>
/// Démarrage du service.
/// </summary>
protected override void OnStart(string[] args)
{
    //chargement de la configuration pour le REMOTING
    RemotingConfiguration.Configure(System.Reflection.Assembly.GetExecutingAssembly().Location+".config");
}
```

3.4.2 Ajout d'un Installer au service

Une fois ceci fait, le service est prêt à être exécuté. Cependant, il est nécessaire de l'installer via l'utilitaire du framework InstallUtil.exe. Pour le rendre installable, il faut rajouter un Installer au projet :

- Dans la vue en mode Design du fichier <nom_du_projet>.cs, cliquer bouton droit et sélectionner « Ajouter le programme d'installation » : un fichier ProjectInstaller.cs est généré
- Dans ce fichier paramétrer le composant ServiceProcessInstaller1 (Account = LocalSystem)
- Paramétrer le composant ServiceInstaller1 (Service Name = <nom du service> cad le nom qui sera affiché sous la console de gestion des services Windows de la machine / StartType = Automatic : le service est lancé dès le démarrage de la machine).



3.4.3 Installation des services sur la machine

Les paragraphes qui suivent décrivent la procédure complète d'installation des services SocleTechnique et BAL.

3.4.3.1 Règles, contraintes et précautions

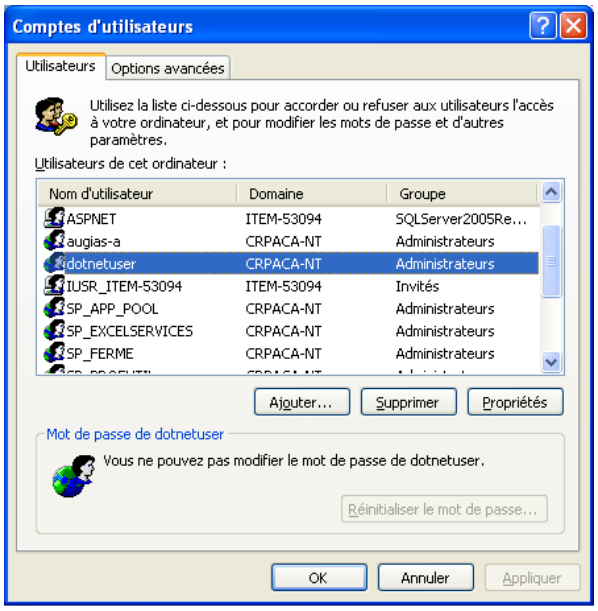
Vérifier que l'utilisateur dotnetuser@cr-paca dispose de privilèges suffisants pour écrire dans les journaux d'évènements CRPACA.BAL et CRPACA.ST.

La configuration de l'utilisation dotnetuser dépend de la machine sur laquelle les services BAL ou Socle Technique seront exécutés.

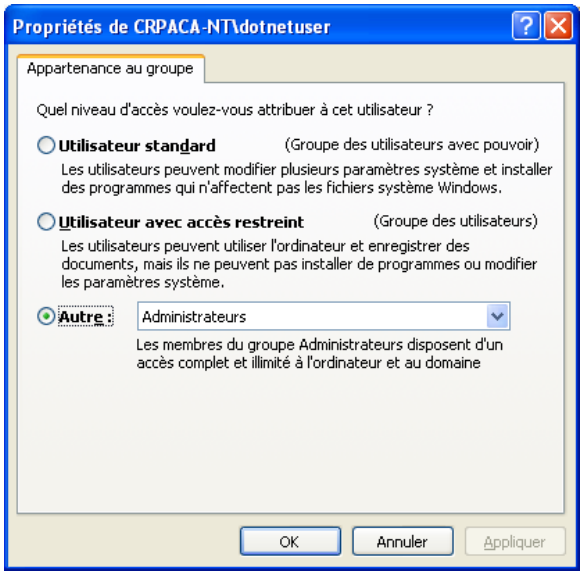
a) Serveurs de recette et de production : demander à l'équipe système.

b) Machine locale : ouvrir Panneau de configuration -> Utilisateurs et élever les droits de dotnetuser@cr-paca.fr.

Pour cela cliquer sur :



Double cliquer sur dotnetuser, cela ouvre une fenêtre de propriétés, choisissez « Administrateurs » dans la liste déroulante « Autre », cliquer sur Ok pour valider.



Comme annoncé ci-dessus, cette manipulation ne concerne que les machines de développement.

3.4.3.2 Installation du service Socle Technique

Pour installer le Socle Technique, il faut se placer dans le répertoire contenant l'exécutable du service et créer un fichier .bat (batch file).

Editer ce fichier BAT avec les lignes suivantes :

%windir%\Microsoft.NET\Framework\v2.0.50727\installutil.exe CRPaca.SocleTechnique.WrapperService.exe.

Entrer ensuite les informations relatives aux identifiants de service (ex : dotnetuser@cr-paca.fr).

Une fois cette saisie achevée le service est maintenant disponible dans le gestionnaire de services.

3.4.3.3 Installation d'un service BAL

Pour installer un BAL, il faut se placer dans le répertoire contenant l'exécutable du service et créer un fichier .bat (batch file).

Editer ce fichier BAT avec les lignes suivantes :

%windir%\Microsoft.NET\Framework\v2.0.50727\installutil.exe [nom de l'exe du service host].exe

Exemple pour le service host de GPOSTOS nommé CRPaca.TOS.BALService.exe :

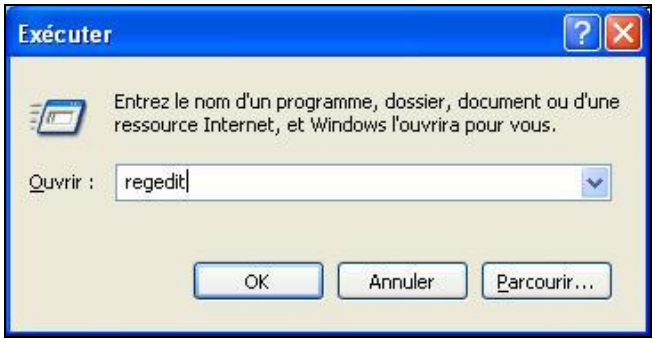
c:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\installutil CRPaca.TOS.BALService.exe

Entrer ensuite les informations relatives aux identifiants de service (ex : dotnetuser@cr-paca.fr ou Système Local dans l'onglet Connexion de la fenêtre Propriétés du service).

Une fois cette saisie achevée le service est maintenant disponible dans le gestionnaire de services.

3.4.3.4 Installation des clés registre d'un service BAL

Pour terminer l'installation de services BAL, il faut entrer dans la base de registre Windows :



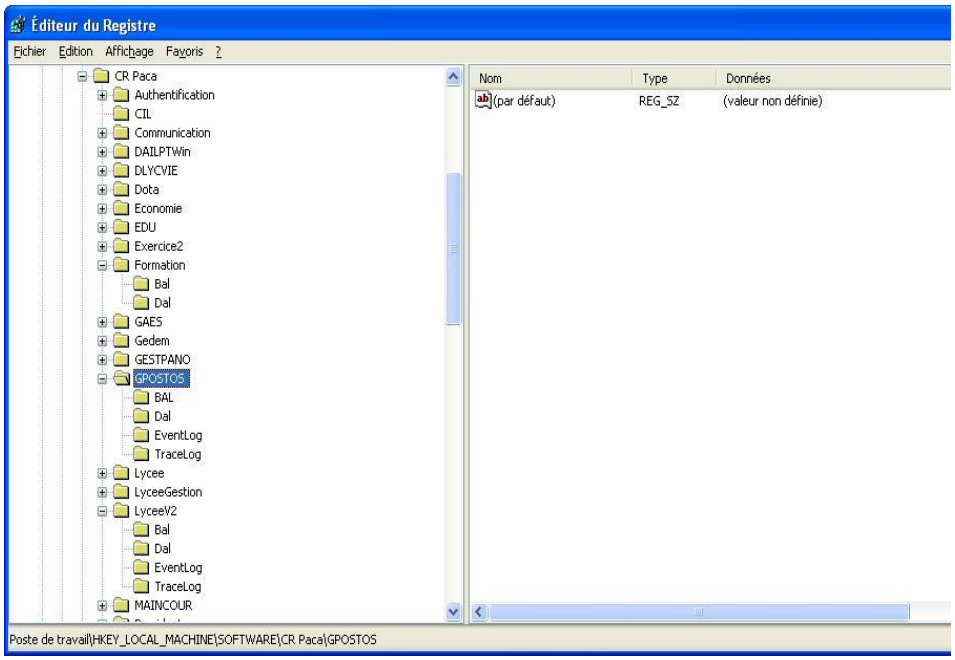
Une fois l'arborescence dévoilée, il faut se placer dans la clé :

HKEY_LOCAL_MACHINE/SOFTWARE/CR Paca/

Cette clé regroupe en sous-clés les entrées registres nécessaires à l'exécution des services distants.

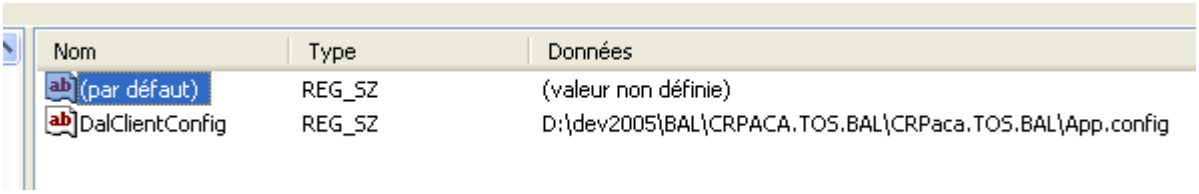
Pour le module GPOSTOS, la clé est *GPOSTOS* et contient 4 sous-clés :

- BAL
- DAL
- EventLog
- TraceLog



La clé BAL correspond au chemin du fichier de configuration du BAL et la clé DAL à la chaine de connexion à la base de données. L'ajout des clés BAL et DAL est obligatoire pour le fonctionnement du service.

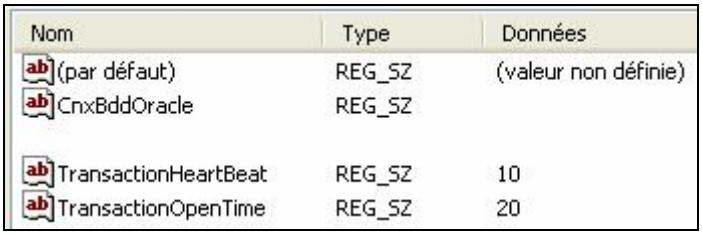
Dans la clé BAL se retrouve 1 valeur de chaînes : DalClientConfig.



La valeur correspond au chemin du fichier de configuration app.config du BAL (assembly métier) :

D:\Dev2005\bal\CRPaca.TOS.BAL\CRPaca.TOS.BAL\App.config sur l'exemple.

Dans la clé DAL se trouve plusieurs valeurs de chaînes :



Les valeurs de chaînes sont CnxBddOracle, TransactionHeartBeat, TransactionOpenTime.

CnxBddOracle est la chaîne de connexion à la DB Oracle (qui n'est pas affichée dans l'exemple pour des raisons évidentes). Son format est de la forme :

User ID=<*Utilisateur*>; Password=<*MotDePasse*>; Data Source=<*Base*>; validate connection=true.

Chaque registre de BAL possède sa propre chaîne de connexion à sa DB.

Les valeurs de chaînes TransactionHeartBeat et TransactionOpenTime sont définies à 10 et 20 par défaut. Ces valeurs de chaînes sont invariantes suivant les registres de BAL.

EventLog et TraceLog contiennent diverses chaînes obsolètes, l'ajout de ces chaînes est facultatif.

3.4.3.5 Installation des sources d'évènements du service Socle Technique

Il existe une solution archivée Visual Source Safe dans $Dev2005/BAL/CreationJournal qui permet de créer une source pour le journal d'évènements du Socle Technique (CRPACA.ST).

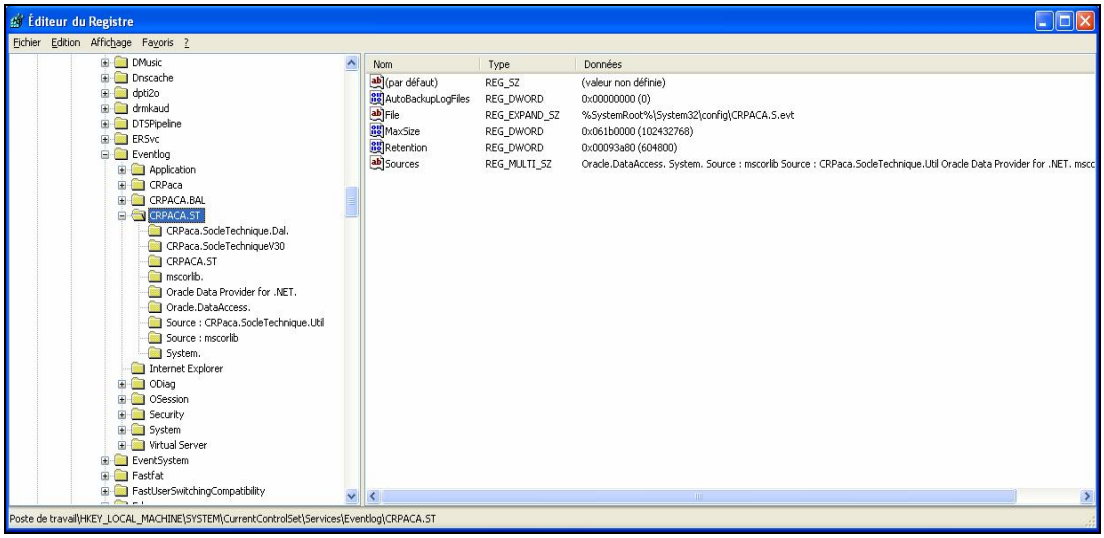
Ouvrez cette solution avec Visual Studio 2005 et utiliser le code suivant dans le point d'entrée `static void Main(string[] args)` pour générer l'application console :

```
            //Ajout CRPaca.SocleTechniqueV30 en tant que source du journal
            CRPACA.ST
            if (EventLog.SourceExists("CRPaca.SocleTechniqueV30"))
            {
                //do nothing
            }
            else
            {

EventLog.CreateEventSource("CRPaca.SocleTechniqueV30","CRPACA.ST");
            }

            //Ajout Socle Technique en tant que source du journal CRPACA.ST
            if (EventLog.SourceExists("Socle Technique"))
            {
            }
            else
            {
                EventLog.CreateEventSource("Socle Technique", "CRPACA.ST");
            }
```

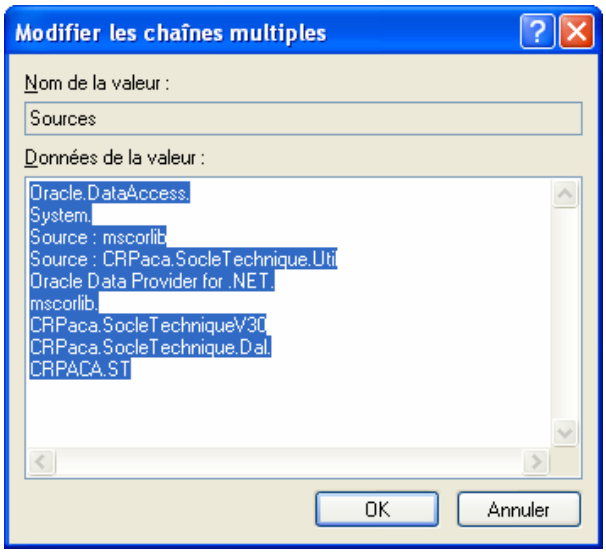
Compiler puis exécuter le ConsoleApplication2.exe sur la machine hôte du Socle Technique. Le registre système de la machine hôte connaît maintenant les sources utilisées par le Socle Technique V30.

Pour vérifier que ces sources ont bien été créées, il faut se placer sur la clé registre suivante :

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Eventlog\CRPACA.ST



Double-cliquer sur la chaîne de valeur Sources pour faire apparaître les sources enregistrées du journal Socle Technique CRPACA.ST :



3.4.3.6 Installation des sources d'évènements du service BAL

Il existe une solution archivée Visual Source Safe dans $Dev2005/BAL/CreationJournal qui permet de créer une source pour le journal d'évènements des BAL (CRPACA.BAL).

Ouvrez cette solution avec Visual Studio 2005 et utiliser le code suivant dans le point d'entrée `static void Main(string[] args)` pour générer l'application console :

```
//Ajout de la source "BAL TOS v20" du journal CRPACA.BAL dans le registre système
            if (EventLog.SourceExists("BAL TOS v20"))
            {
                //do nothing
            }
            else
            {
                EventLog.CreateEventSource("BAL TOS v20", "CRPACA.BAL");
            }
```

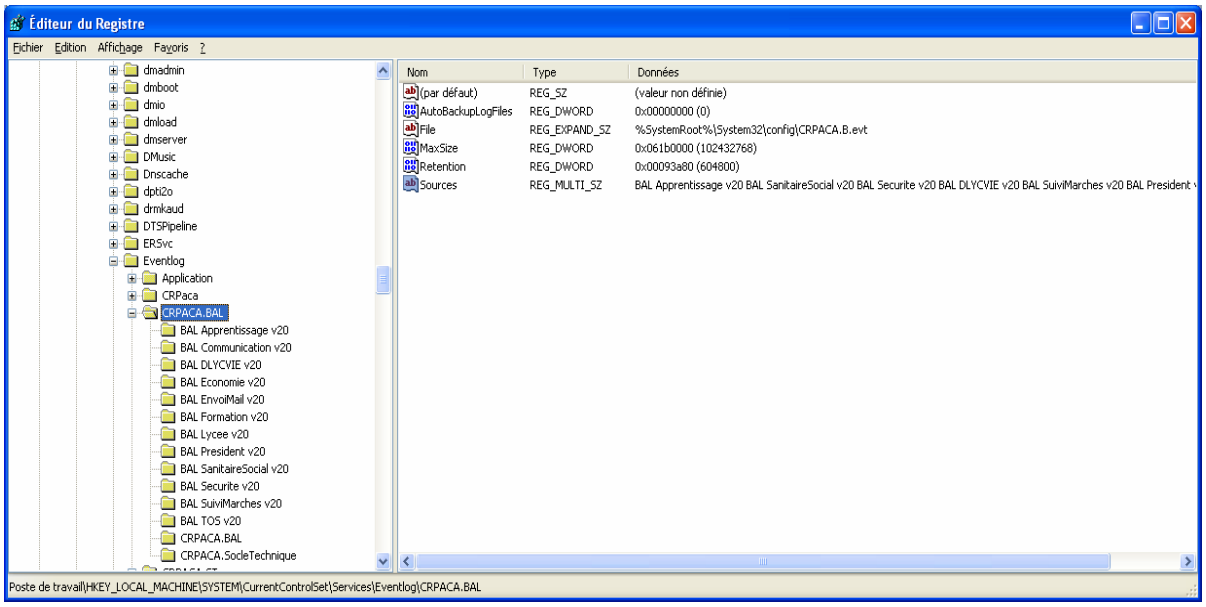
Le nom de la source doit correspondre à la valeur de la clé du fichier app.config du service BAL (exemple pour GPOSTOS ici) :

```
<add key="ModuleBAL" value="BAL TOS v20"/>
```

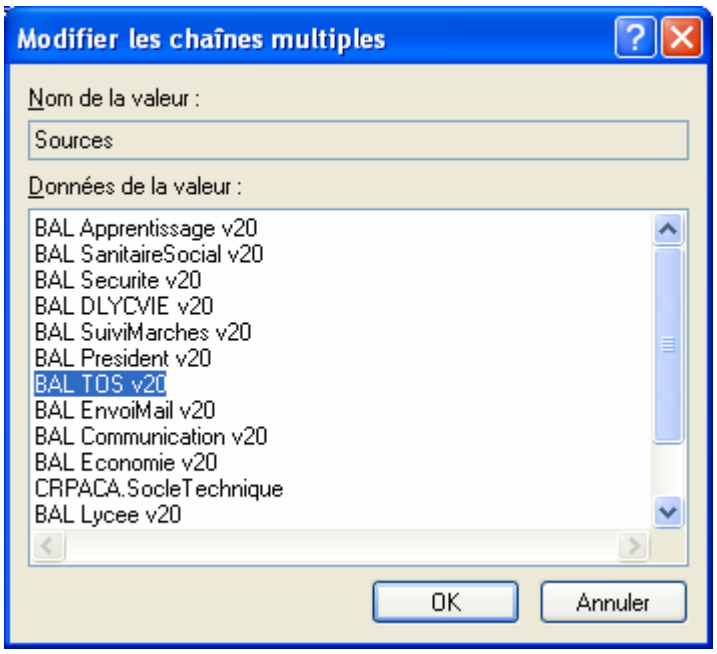
Une fois sauvegardé et compilé, utiliser le ConsoleApplication2.exe sur votre machine cible qui supportera le BAL. Les sources d'évènements pour votre BAL sont maintenant installées et votre service pourra démarrer.

Pour vérifier que les sources d'évènements du BAL sont crées il faut aller dans la clé de registre :

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Eventlog\CRPACA.BAL

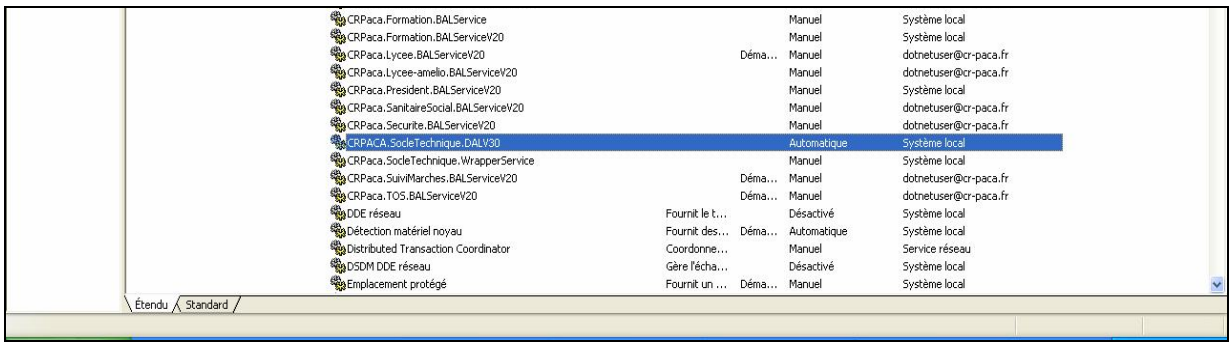


Double-cliquer sur la valeur "Sources" : celle s'ouvre et vous montre toutes les sources d'évènements du journal CRPACA.BAL.



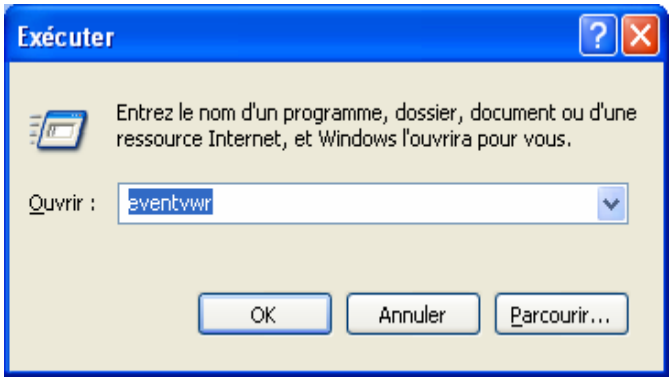
3.4.3.7 Procédure de vérification de l'installation du Socle Technique

Allez dans le gestionnaire des services :

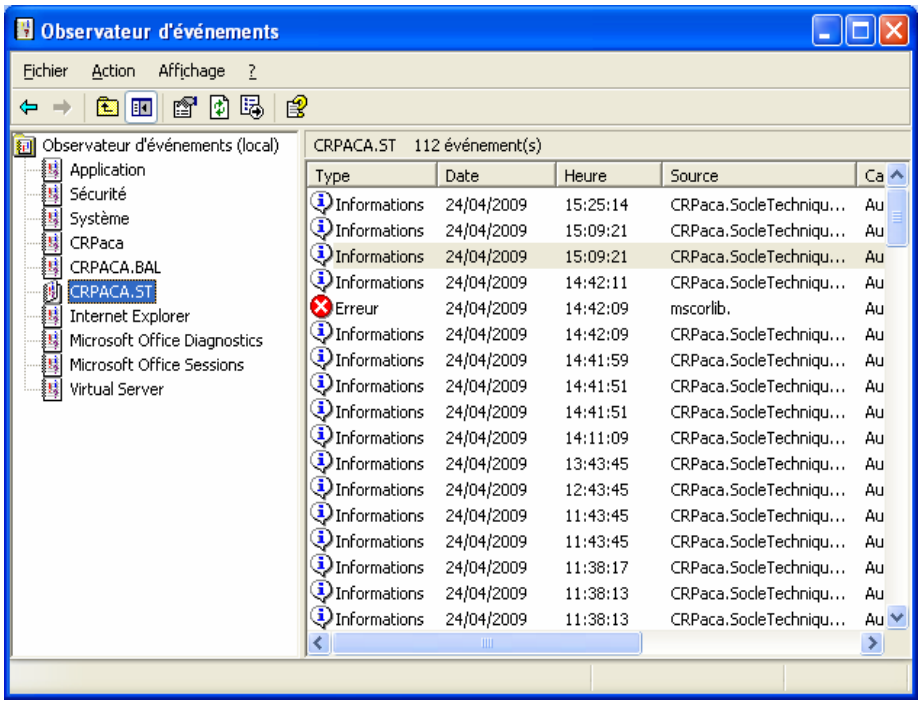


Vérifier que votre service est bien installé (ici CRPACA.SocleTechnique.DALV30), session : Système local.

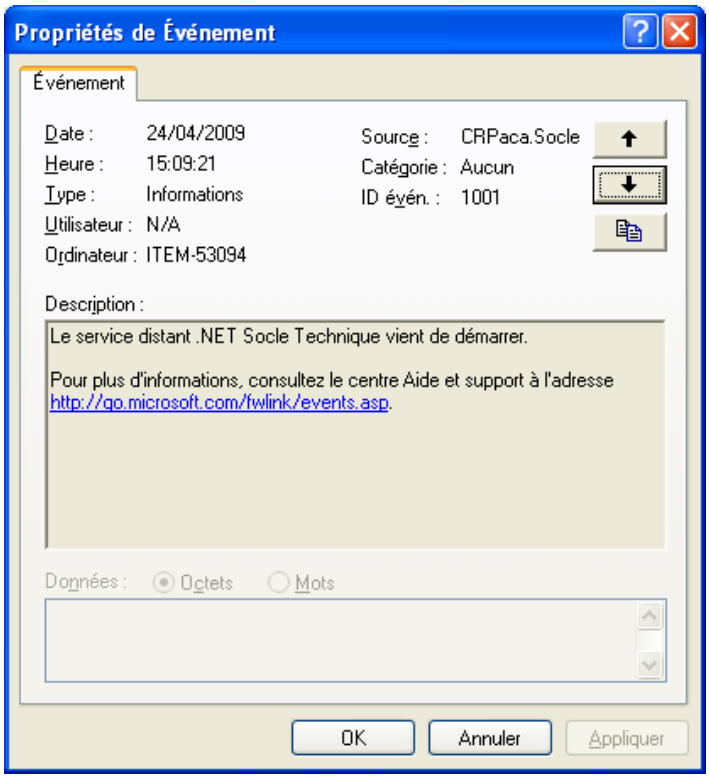
Démarrer le service, allez dans le journal d'évènements :



Sélectionnez le journal CRPACA.ST :

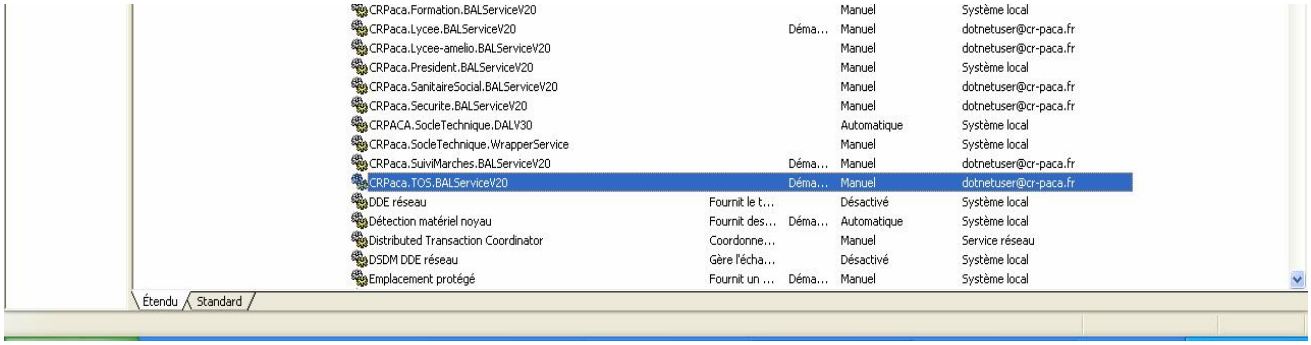


Un évènement apparaît dans la liste de droite, double cliquez dessus, il devrait contenir les informations suivantes :

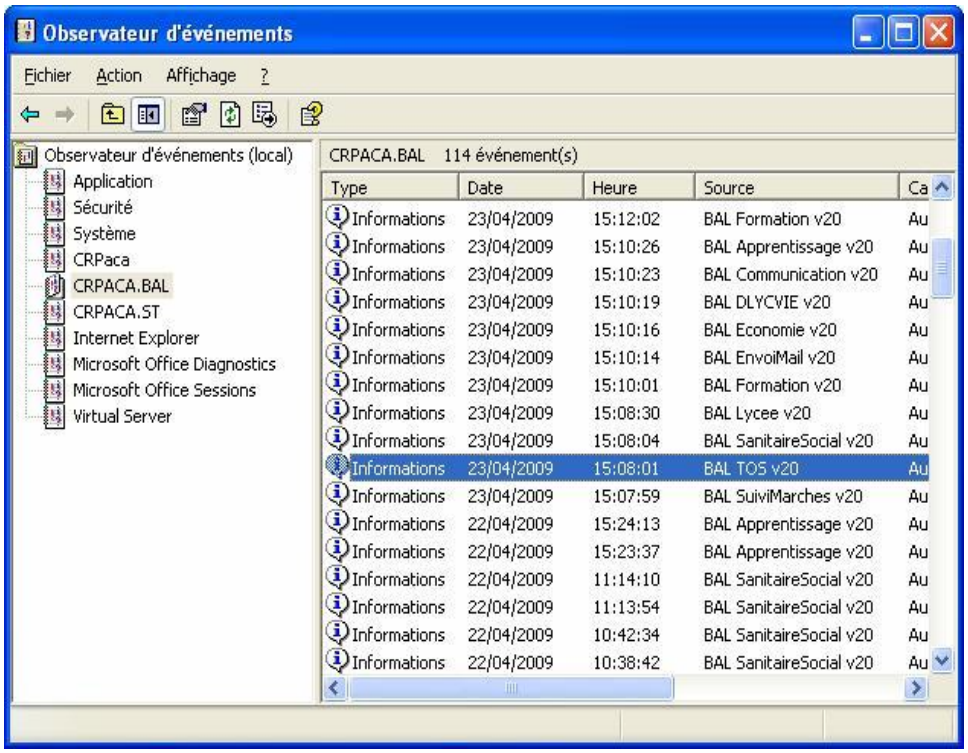


3.4.3.8 Procédure de vérification de l'installation du BAL

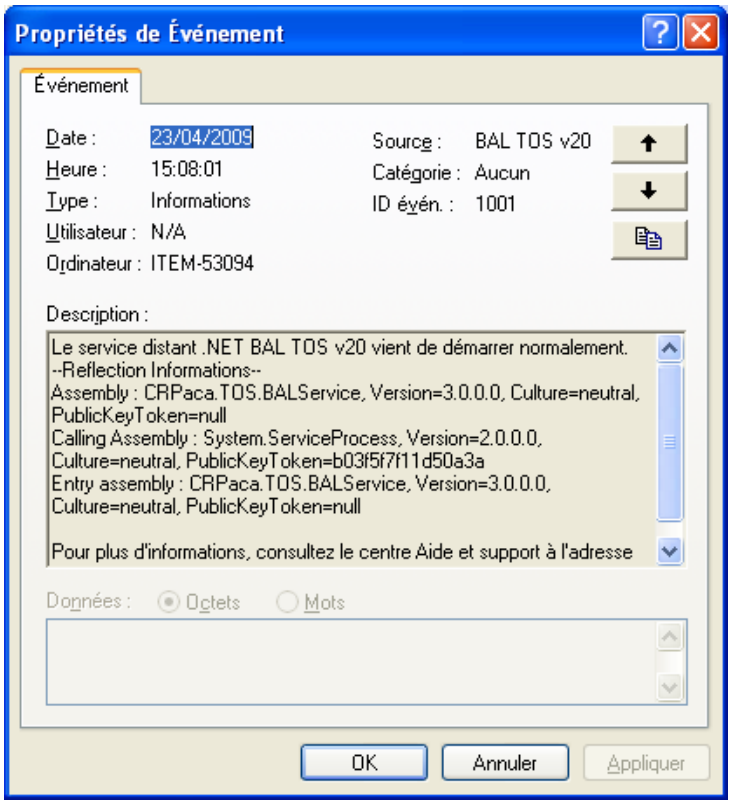
De manière analogue à la procédure pour le Socle Technique, aller dans le gestionnaire de services :



Démarrer sur votre service BAL récemment installé (ici CRPACA.TOS.BALServicev20) puis aller dans le journal d'évènements :



Double-cliquer sur l'évènement de source BAL TOS v20, il devrait obtenir les informations suivantes :



3.4.4 Procédures de désinstallation

Le fichier uninstall.bat fourni avec le service permet de désinstaller le service. Double-cliquer dessus pour désinstaller le service.

Si celui-ci n'est pas fourni il vous faut suivre la procédure ci-dessous.

Pour désinstaller un BAL ou un Socle Technique, il faut se placer dans le répertoire contenant l'exécutable du service et créer un fichier .bat (batch file) :

%windir%\Microsoft.NET\Framework\v2.0.50727\installutil.exe -u [chemin et nom de l'exe du service host].exe

Exemple pour le service host de GPOSTOS nommé CRPaca.TOS.BALService.exe :

c:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\installutil –u ./CRPaca.TOS.BALService.exe

La désinstallation d'un service doit se produire dans son répertoire d'installation sinon le service ne se désinstallera pas.

Procédure de vérification de désinstallation

Vous pouvez voir qu'un service a été désinstallé en regardant la liste des services du gestionnaire après votre tentative de désinstallation (rafraîchir la liste avec F5). Le nom du service est libéré du gestionnaire et peut être réutilisé lors d'une installation ultérieure.

3.5 Règles et recommandations d'arborescence

Ces règles et recommandations visent à améliorer la lisibilité des projets utilisant l'outil de développement Visual Studio .NET et Visual Source Control et permettre la mise en œuvre commune d'une charte accessible et utilisée par l'ensemble d'une équipe de développement.

Visual Source Safe permet de partager les sources en cours de développement entre tous les développeurs. Ce produit propose de stocker dans une base de données sur un serveur, les solutions et les projets créés via Visual Studio .net et permet ainsi le partage des codes sources parmi toute l'équipe. Cette organisation nécessite une arborescence de fichiers cohérente et stable afin de faciliter, entre autre, le déploiement des applications.

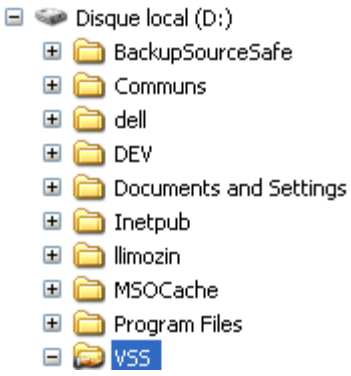
Cette organisation générale de l'espace de travail se scinde en 3 parties :

- Organisation du répertoire de contrôle de code source,
- Organisation du répertoire de développement,
- Organisation des répertoires de déploiement.

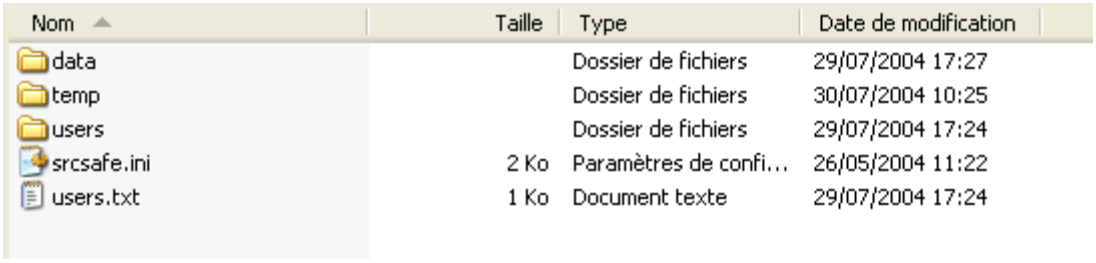
3.5.1 Organisation du répertoire de contrôle de code source

3.5.1.1 Répertoire de contrôle de code source

La base de données Visual Source Safe doit être stockée dans un répertoire partagé « VSS » sur une machine réseau CR-PACA. Ce répertoire sera accessible par les autres plates formes de développement via un lecteur réseau.



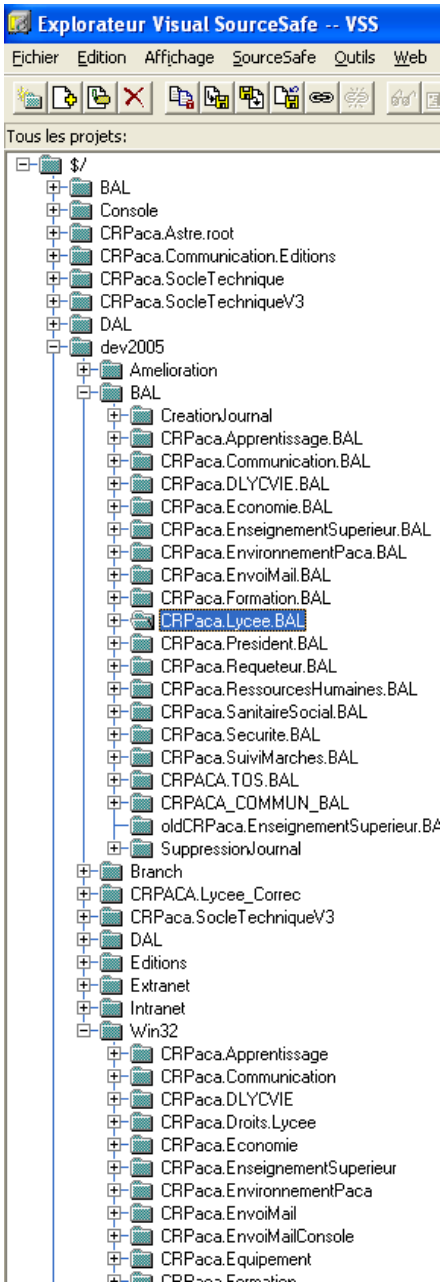
Le répertoire VSS contient la base de données de tous les développements de CR-PACA :



Nom	Taille	Type	Date de modification
data		Dossier de fichiers	29/07/2004 17:27
temp		Dossier de fichiers	30/07/2004 10:25
users		Dossier de fichiers	29/07/2004 17:24
srcsafe.ini	2 Ko	Paramètres de confi...	26/05/2004 11:22
users.txt	1 Ko	Document texte	29/07/2004 17:24

Chaque développeur ayant créé un lecteur réseau qui pointe sur VSS peut se connecter à la base de donnée Visual Source Safe « CRPaca » (uniquement s'il a l'autorisation de se connecter avec Visual Source Safe) à partir du fichier « srcsafe.ini » pour initialiser sa solution de développement pour ainsi récuperer et/ou sauver le code qu'il produit.

3.5.1.2 Organisation l'arborescence dans Visual Source Safe



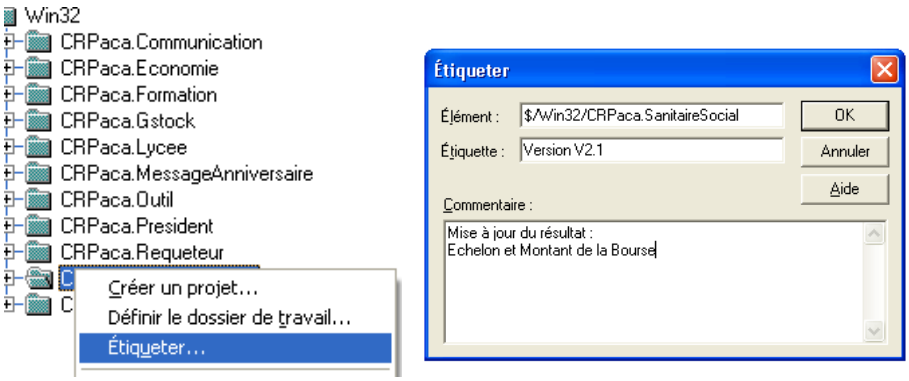
Le dossier contenant tous les projets est le répertoire DEV2005. On y distingue principalement les solutions :

- CRPaca.BAL qui comprend l'ensemble des composants métier d'accès aux données, pour chaque domaine d'application.
- CRPaca.SocleTechniqueV3 qui comprend les projets CRPaca.SocleTechnique.Dal, composants de la couche d'accès aux données, CRPaca.SocleTechnique.UI composants de la couche présentation, et CRPaca.SOcleTechnique.Util composants factorisables pouvant être utilisés par l'ensemble des applications développées.
- Extranet qui comprend l'ensemble des applications Web par domaine d'application.
- Win32 qui comprend l'ensemble des applications clients riches par domaine d'application.
- Editions qui comprend l'ensemble des éditions « reporting services ».

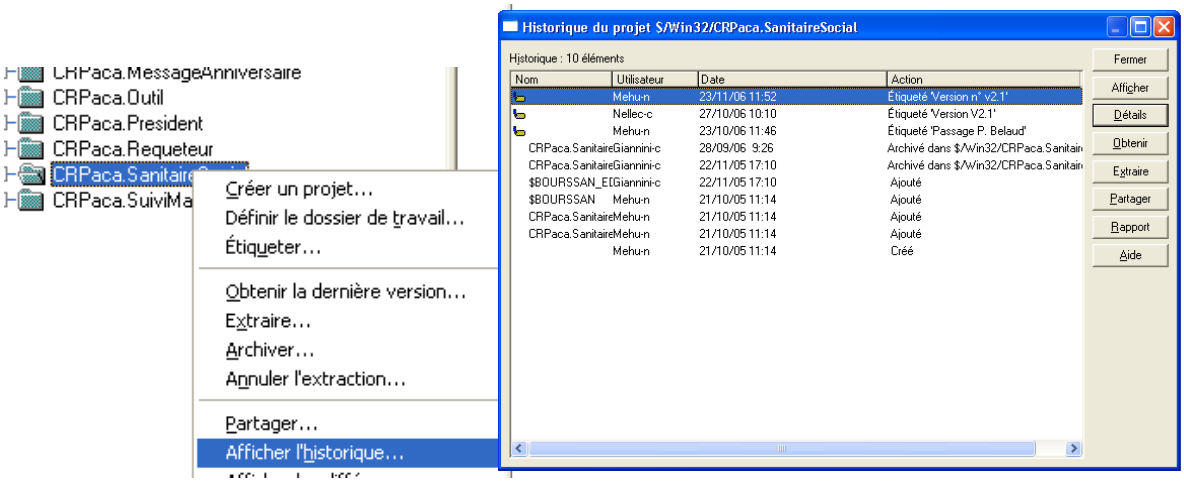
3.5.1.3 Etiquetage des versions dans Visual Source Safe

Lorsque des modifications importantes sont apportées à une application, il faut préciser la version et préciser les modifications apportées en étiquetant le projet.

Pour ce faire, dans Source Safe, il faut :



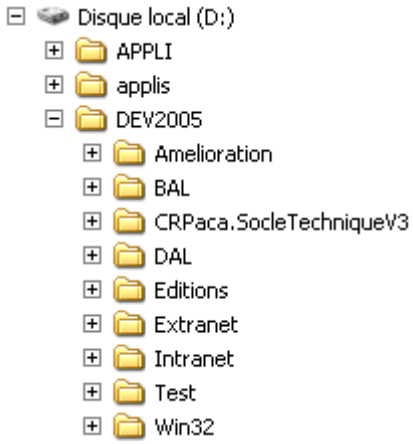
Pour visualiser les différentes versions :



3.5.2 Organisation du répertoire de développement

3.5.2.1 Généralités sur la plateforme de développement

L'arborescence des répertoires de développement doit être le même sur chaque plate forme de développement (poste du développeur).



3.5.2.2 Applications Client riche (Windows forms)

Les projets Clients riches sont situés dans le répertoire

D:/Dev2005/Win32/

Avec un sous répertoire par domaine d'application.

3.5.2.3 Librairies de classe d'accès aux données pour les applications Client Riche

Les librairies de classes d'accès aux données pour les applications Clients riches se situent dans le répertoire

D:/Dev2005/Win32/CRPaca.« Domaine »/«Application»/bin

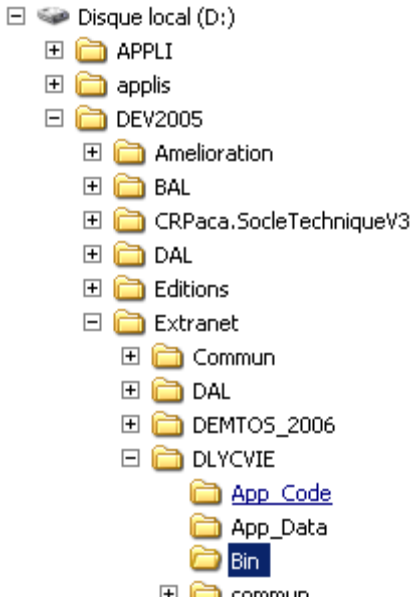
3.5.2.4 Applications Web

Les projets Web sont situés dans le répertoire

D:/Dev2005/Extranet/ correspondant au répertoire de référence du serveur IIS de chaque machine.

3.5.2.5 Librairies de classe d'accès aux données pour les applications Web

Les librairies de classes d'accès aux données pour les applications Web se situent dans le répertoire D:/Dev2005//Extranet/ « Domaine »/bin



3.5.3 Organisation du répertoire de déploiement en production

3.5.3.1 Applications Clients riches

Le serveur de production se nomme [redacted]. Les programmes source doivent être stockés sur :

\\[redacted]\applis\progs\Specif\.« Domaine »\«Application»\ Client

3.5.3.2 Applications Web

Le serveur de production se nomme [redacted]. Les programmes source doivent être sur le disque D:\

Les applications web seront copiées dans le répertoire :

« D:\appli\ wwwroot\ »

4 DOCUMENT 1 – 2.00 - ARCHITECTURE DE DEVELOPPEMENT .NET 4.0

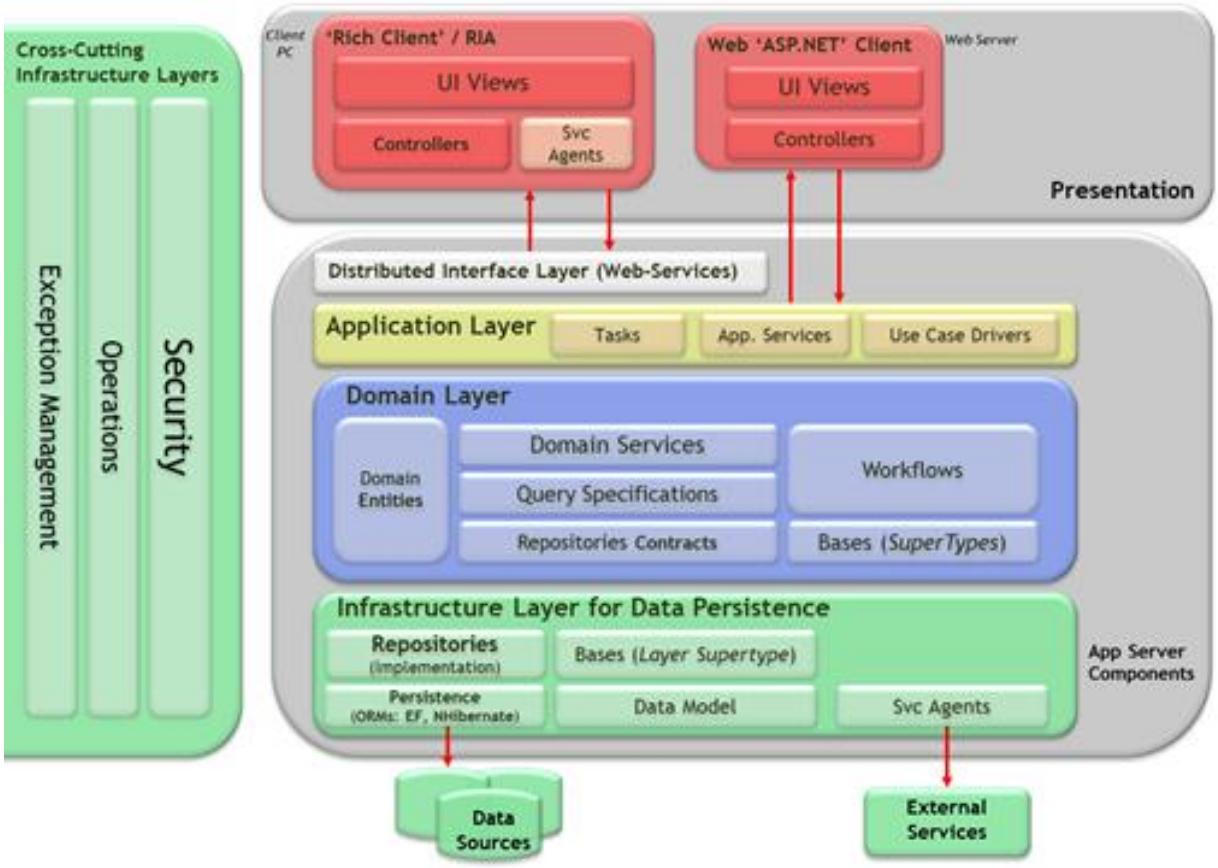
4.1 Introduction

Ce document a pour objectif de présenter la nouvelle architecture recommandée pour la Région Provence Alpes Côte d'Azur (PACA).

La première partie présentera le découpage architectural préconisé. La deuxième partie reviendra plus en détail sur les éléments technique de WCF. La troisième partie présentera en détail les composants de l'interface utilisateurs : ASP .NET 4 et WPF 4. Dans la troisième partie l'accent sera mis sur WPF, ASP .NET étant plus ancien et généralement mieux maîtrisé.

4.2 Architecture préconisée

Vue d'ensemble des couches d'architecture

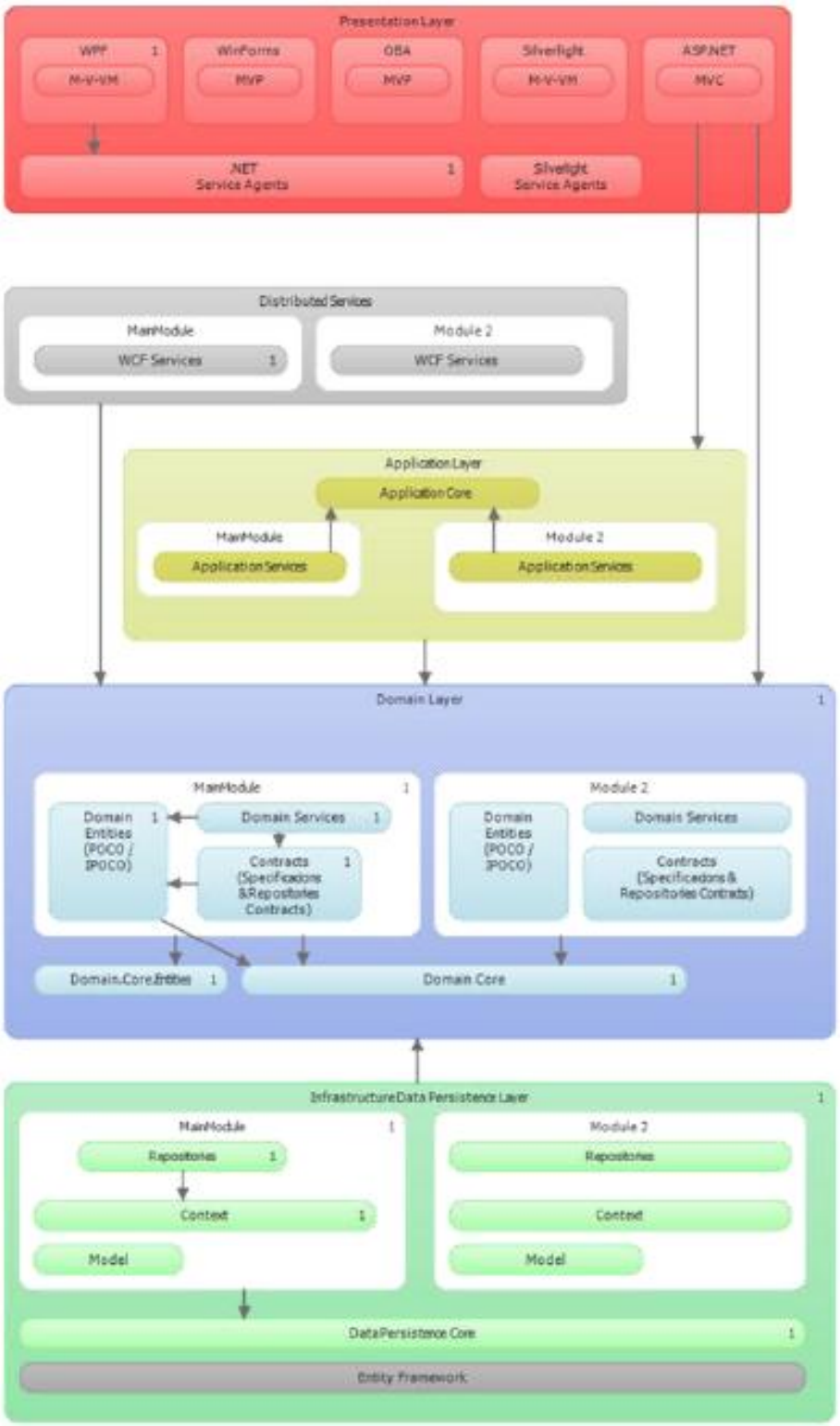


L'architecture préconisée s'appuie sur le Domain Driven Developement (DDD), il s'agit de définir des objets fonctionnels. Cette approche permettra de créer plus aisément un Framework métier et une « boîte à outils » d'interface utilisateur plus métier que technique.

Cette architecture ciblera 2 types d'applications :

- Des applications de type client riche ('Rich Client' dans le schéma)
- Des applications web pures, ASP .NET. Ces applications devront être conformes aux normes d'accessibilités et seront ouvertes sur l'intranet et l'extranet.

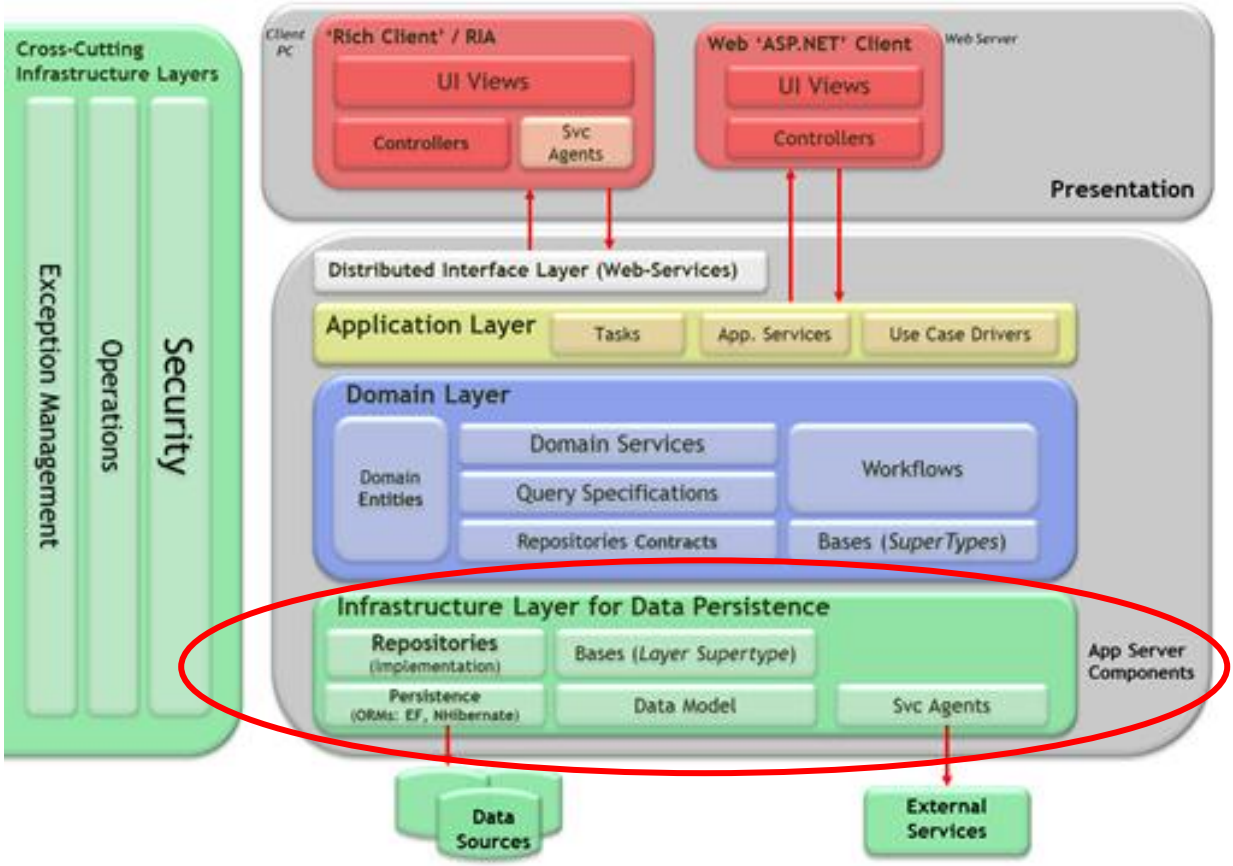
Le schéma d'architecture ci-dessus conduit au schéma ci-dessous une fois modélisé dans Visual Studio 2010. Le schéma ci-dessous reflète la structure à adopter dans les projets conformément à l'architecture préconisée.



4.3 Présentation des couches de l'architecture

4.3.1 Couche d'Infrastructure de Persistance de Données (Infrastructure Data Persistence Layer)

Avant de présenter les divers composants le diagramme ci-dessous présente le positionnement de cette couche dans l'architecture.



4.3.1.1 Sous couche Persistence

La technologie de persistance retenue est Entity Framework 4. Il s'agit de la brique technique de .NET 4 pour le mapping O/RM. Cette brique technologique permet de travailler sur un modèle objet. Les appels à la base de données étant générés par l'Entity Framework (EF). EF sera encapsulé au travers d'un Repository afin de pouvoir découpler cette couche et la remplacer le cas échéant (ex : NHibernate, Devart). L'utilisation d'un Repository permet aussi de travailler avec une PI (Persistence Ignorance) ce qui est une bonne pratique.

4.3.1.2 Sous-couche Repository (Repository Pattern)

Il s'agit ici de l'implémentation concrète des Repository définit dans la couche de Domaine. On ne définit pas les interfaces dans la couche de données, afin d'inverser la dépendance. Cela découple l'implémentation Technologique de l'implémentation du domaine. Si par la suite on décide d'utiliser d'autre implémentation pour l'accès aux données il ne sera pas nécessaire de re-factoriser le code après la migration puisque tout sera contenu dans le domaine. Il s'agit du design pattern « Separated Interface » qui consiste à définir une interface dans un package séparé de celui de son implémentation. Cette solution permet d'avoir une approche de type Plugin, puisque l'import des interfaces n'amène pas l'implémentation concrète. La ré-implémentation en sera d'autant plus aisée. L'avantage de l'approche plugin est que l'on peut décider par code ou par configuration de l'implémentation concrète à utiliser.

Le pattern Repository permet de travailler sur le modèle objet en mémoire, la persistance ne se faisant qu'à la fin. Il s'agit de l'implémentation tirée du livre PoEAA de Martin Fowler et du livre DDD de Eric Evans, pour le cas particulier du Repository sur le domaine.

Toutes interactions avec la base de données doit passer par le Repository, la base n'a pour seule et unique vocation que de stocker des objets.

4.3.1.3 Modèle de données (Data Model)

Le modèle de données est toujours présent lorsque l'on travaille avec des outils de mapping O/RM. Il s'agit de la partie créant la relation entre les entités du modèle objet et les éléments dans la base de données.

4.3.1.4 Sous couche Bases

Il s'agit de l'implémentation du pattern Supertype Layer. Ce pattern est un pattern de base il consiste à regrouper dans un type de base les éléments commun des types en dérivant.

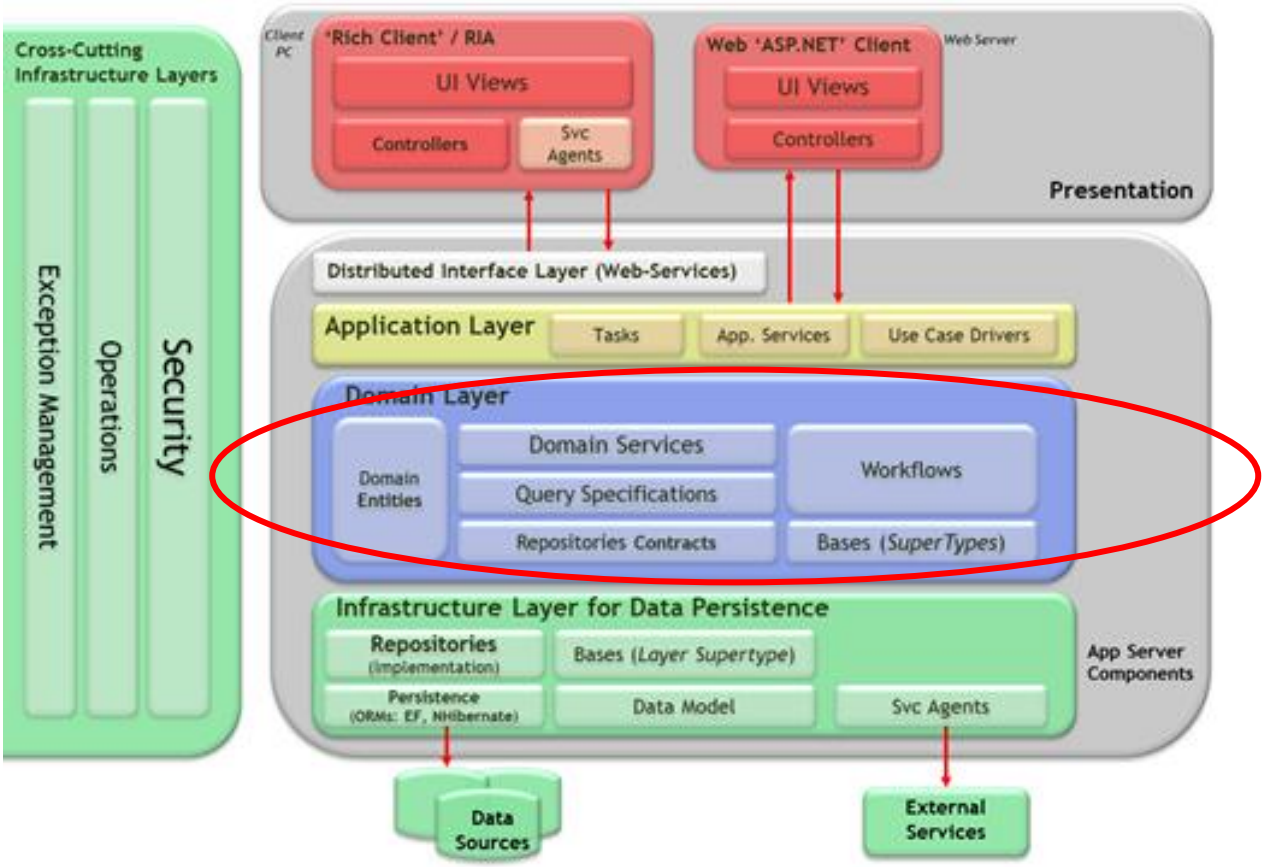
Par exemple tous les objets Repository ont des comportements communs, on va donc créer des classes de bases les implémentant. Il est à ce stade préférable de travailler sur des objets concrets (on évitera donc à priori les classes abstraites).

4.3.1.5 Agents de services distribués externes

Cette brique de l'architecture permet de considérer des services externes à l'application comme des sources de données. Dans le contexte de la région on pourra y prévoir les appels aux services TCP/IP existants. On encapsulera ces services dans un Repository afin par la suite de les faire évoluer sans impact sur les applications.

4.3.2 Couche de Domaine (Domain Layer)

Avant de présenter les divers composants, le diagramme ci-dessous présente le positionnement de cette couche dans l'architecture.



4.3.2.1 Entités du domaine

Les entités du domaine sont les objets qui proviennent du langage métier. Par exemple dans cette brique on trouvera : lycée, facture, demande, dossier …

Ce sont les objets qui sont persistés en base de données après de légères transformations (faites dans la couche Repository qui communique avec l'O/RM).

La définition clef d'une entité est qu'elle a une identité. Cette identité peut être matérialisée par un identifiant par exemple le numéro de lycée.

Les entités respectent le principe d'ignorance de la persistance (PI), c'est-à-dire que l'on ne codera que des POCO (Plain Old CLR Object). Nous utiliserons aussi les principes de IPOCO (Interface POCO), c'est-à-dire que l'on utilisera des interfaces pour décrire nos objets en utilisant des interfaces agnostiques de l'outil de persistance.

Aujourd'hui c'est une des critiques à l'encontre d'Entity Framework, qui oblige les entités à implémenter ses interfaces (http://blogs.msdn.com/b/adonet/archive/2009/05/11/sneak-preview-persistence-ignorance-and-poco-in-entity-framework-4-0.aspx). De ce fait, les objets du domaine sont liés à la logique de persistance. Comme l'architecture devra supporter des bases Oracle, et donc utiliser un autre système de persistance nous cherchons à découpler l'ensemble avec IPOCO. Tout en gardant la possibilité d'utiliser Entity Framework par la suite.

4.3.2.2 Pattern Value-Object

La caractéristique clé qui définit un Value Object, c'est qu'il n'a pas d'identité. C'est peut-être un peu simpliste, mais l'intention d'un Value Object est de représenter quelque chose par ses seuls attributs. Deux Value Object peuvent avoir des attributs identiques, auquel cas ils sont identiques. Ils n'ont toutefois pas d'autre indication de leur similarité que par la vertu de leurs attributs.

Un autre aspect commun aux Value Object, c'est qu'ils sont immutables, une fois créés, ils ne peuvent pas être modifiés ou adaptés. Vous pouvez en créer un nouveau, et comme ils n'ont pas d'identité, c'est la même chose que de changer l'objet d'origine. Exemples d'objets : Money, Adresse, ProductCode.

Voici un exemple pour illustrer le propos, prenons l'exemple du numéro de téléphone. Sur cet objet on peut vouloir en extraire l'indicatif. Si le numéro de téléphone est simplement représenté par un attribut de type String dans un objet, on devra implémenter la logique d'extraction de l'indicatif dans plusieurs objets. En utilisant un Value Object on implémentera simplement un constructeur prenant un String en paramètre, et une fonction retournant l'indicatif. Comme l'objet est construit on valide au moment du constructeur la validité de l'objet. Cet objet étant immutable on est toujours sûr d'en avoir un créé valide. Un objet "immutable" est un objet dont les propriétés sont définies à la création de l'objet puis ne peuvent plus changer durant la vie de l'objet.

Une des raisons de la création de ce type d'objet (c'est d'ailleurs aussi un design pattern) : la gestion de la mémoire. En effet, vu que les données sont créées à la création de l'objet, il suffit d'allouer l'ensemble de la mémoire en une fois et de ne plus y toucher jusqu'à la destruction de l'objet.

Un autre avantage est le multithreading. En effet, si l'objet n'évolue pas au cours de sa vie, tous les threads peuvent y accéder quand ils veulent sans avoir besoin de se synchroniser. Cela permet une programmation plus simple et un programme plus rapide.

Exemple de code :

```
public class PhoneNumber
{
string _phone ;
        public PhoneNumber ( string phone)
{
```

```
//application de vérifications de validité
this._phone = phone ;
}

public GetPhoneCode()
{
// exemple pour extraire l'indicatif de téléphone
return  _phone.Substring(…) ;
}
}
```

Règle d'architecture : on implémentera un objet de type Value Object à chaque fois que cela sera possible.

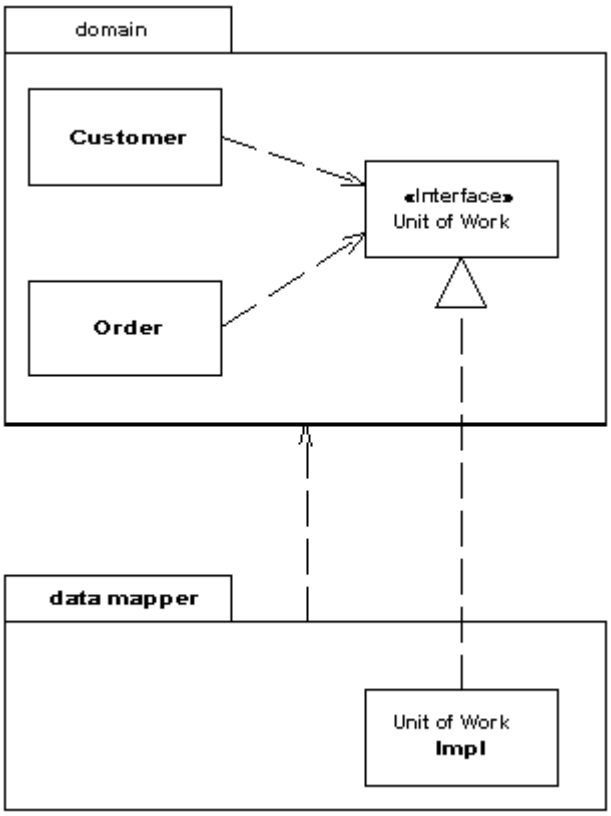
4.3.2.3 Pattern Agreggate

L'agrégation est plus une bonne pratique qu'un pur pattern l'idée derrière cela est de faire en sorte de centraliser les points de manipulation de certains objets. L'agrégation permet de limiter le nombre de relations des objets en encapsulant la manipulation de certains objets dans d'autres. L'exemple le plus simple est la facture avec les lignes de produits. Comme les lignes de produits n'existent pas sans facture, elles sont crées dans la facture.

Règle d'architecture : on utilisera l'agrégation pour gérer la relation d'objets entre eux.

4.3.2.4 Contrats et Interfaces de la couche de domaine

L'implémentation des interfaces se fera en respectant le pattern Separated Interfaces. La définition de ce pattern est de : « définir une interface dans un package mais l'implémenter dans un autre. Ainsi un client qui a besoin de cette interface ne la reçoit pas avec son implémentation. » Ce pattern est décrit par le schéma ci-dessous, il est mis en œuvre dans les Repositories entre autre.



Règle d'architecture : les interfaces ne doivent pas être déclarées dans le même package que celui où elles sont implémentées.

4.3.2.5 Sous-couche des services du modèles de domaine

Il ne s'agit pas ici d'une approche de type services distribués (comme par exemple avec des web services) mais plutôt d'un ensemble d'objets permettant de faire communiquer la couche application tout en définissant une couche d'abstraction. Cette couche de services permet d'exposer un ensemble d'opérations et de gérer les réponses de l'application à chaque opération.

Règle d'architecture : les opérations de l'application seront définies dans une couche de service du domaine.

4.3.2.6 Pattern Unit of Work

On met ici en œuvre ce pattern pour son aspect Persistence Ignorance. En effet, il permet de découpler l'application du Framework de persistance.

Explications

Le pattern Unit of Work permet de découpler les objets du domaine et logique du Framework de persistance. On va donc travailler sur des objets POCO (ne s'appuyant que sur des objets du Framework .NET).

Ce point est important en effet si lors du codage des tests unitaires nous devions avoir des références au Framework de persistance, cela indiquerait que nous ne pouvons pas le remplacer sans devoir réécrire les tests. Hors il n'y a aucune raison que des tests métiers doivent prendre en considération les aspects technique de la persistance.

4.3.2.7 Pattern Specification

Ce pattern est simple mais très puissant. Il permet de :

- marquer et identifier les règles métiers,
- les centraliser,
- les réutiliser,
- communiquer entre développeurs et fonctionnels sur ces règles métiers

La difficulté dans l'implémentation de règle métier est d'éviter de créer une dette technique :

- règles perdues et dissimulées dans le code,
- implémentation de règles dupliquées,
- règles difficilement modifiables.

Ainsi, idéalement un expert fonctionnel aimerait :

- définir une règle métier,
- composer plusieurs règles métiers,
- avoir l'état exact d'implémentation des règles métiers tel que développé dans le code source (aux valeurs près !),
- activer/désactiver une règle rapidement, souvent sans redémarrage des serveurs d'application.

Pour faciliter la maintenance d'un tel modèle, la solution suivante doit être mise en place : l'utilisation du Pattern Specification.

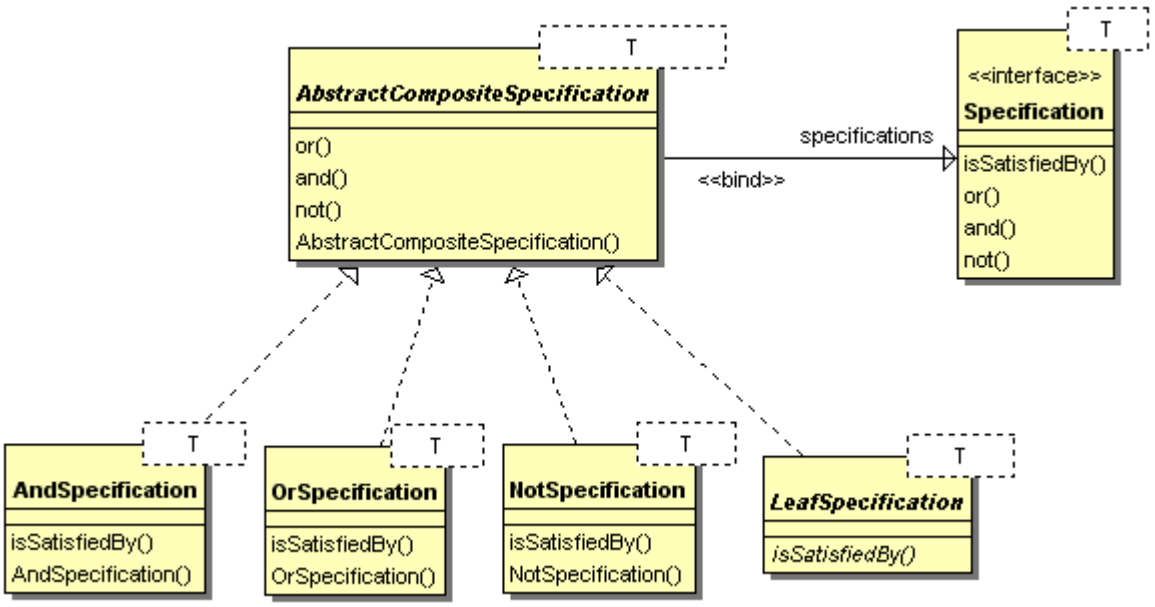
L'implémentation d'une règle métier suit le contrat d'utilisation suivant :

```
public interface ISpecification<TEntity>
   where TEntity : class,new()
{
   Expression<Func<TEntity, bool>> SatisfiedBy();
}
```

Implémenté dans la classe abstraite suivante :

```
public abstract class Specification<TEntity>
        : ISpecification<TEntity>
        where TEntity : class,new()
    {
        public abstract Expression<Func<TEntity, bool>> SatisfiedBy();
     public static Specification<TEntity> operator &(Specification<TEntity> leftSideSpecification,
Specification<TEntity> rightSideSpecification)
        {
            return new AndSpecification<TEntity>(leftSideSpecification, rightSideSpecification);
        }
        public static Specification<TEntity> operator |(Specification<TEntity> leftSideSpecification,
Specification<TEntity> rightSideSpecification)
        {
            return new OrSpecification<TEntity>(leftSideSpecification, rightSideSpecification);
        }
        public static Specification<TEntity> operator !(Specification<TEntity> specification)
        {
            return new NotSpecification<TEntity>(specification);
        }
        public static bool operator false(Specification<TEntity> specification)
        {
            return false;
        }
        public static bool operator true(Specification<TEntity> specification)
        {
            return true;
        }
        #endregion
    }
```

Une méthode Satisfiedby détermine si la règle métier est respectée. Les trois autres méthodes « or, and et not » permettent de combiner les règles métiers entre elles.



Il y a trois classes utilitaires permettant d'implémenter les opérateurs :

- AndSpecification
- OrSpecification
- NotSpecification

Par exemple la classe AndSpecification :

Dans un premier temps on définit une classe de base pour gérer les membres gauche et droits des opérations logiques.

```
public abstract class CompositeSpecification<TEntity>
        : Specification<TEntity>
        where TEntity : class,new()
    {
   public abstract ISpecification<TEntity> LeftSideSpecification { get; }
   public abstract ISpecification<TEntity> RightSideSpecification { get; }
    }
```

Maintenant nous allons implémenter le And :

```
    public class AndSpecification<T>
       : CompositeSpecification<T>
       where T : class,new()
    {
        private ISpecification<T> _RightSideSpecification = null;
        private ISpecification<T> _LeftSideSpecification = null;

        public AndSpecification(ISpecification<T> leftSide, ISpecification<T> rightSide)
        {
            if (leftSide == (ISpecification<T>)null)
                throw new ArgumentNullException("leftSide");

            if (rightSide == (ISpecification<T>)null)
                throw new ArgumentNullException("rightSide");

            this._LeftSideSpecification = leftSide;
            this._RightSideSpecification = rightSide;
        }
        public override ISpecification<T> LeftSideSpecification
        {
            get { return _LeftSideSpecification; }
        }
        public override ISpecification<T> RightSideSpecification
        {
            get { return _RightSideSpecification; }
        }
        public override Expression<Func<T, bool>> SatisfiedBy()
        {
            Expression<Func<T, bool>> left = _LeftSideSpecification.SatisfiedBy();
            Expression<Func<T, bool>> right = _RightSideSpecification.SatisfiedBy();

            return (left.And(right));

        }
}
```

Exemple d'implémentation pour la vérification de la validité d'un compte bancaire

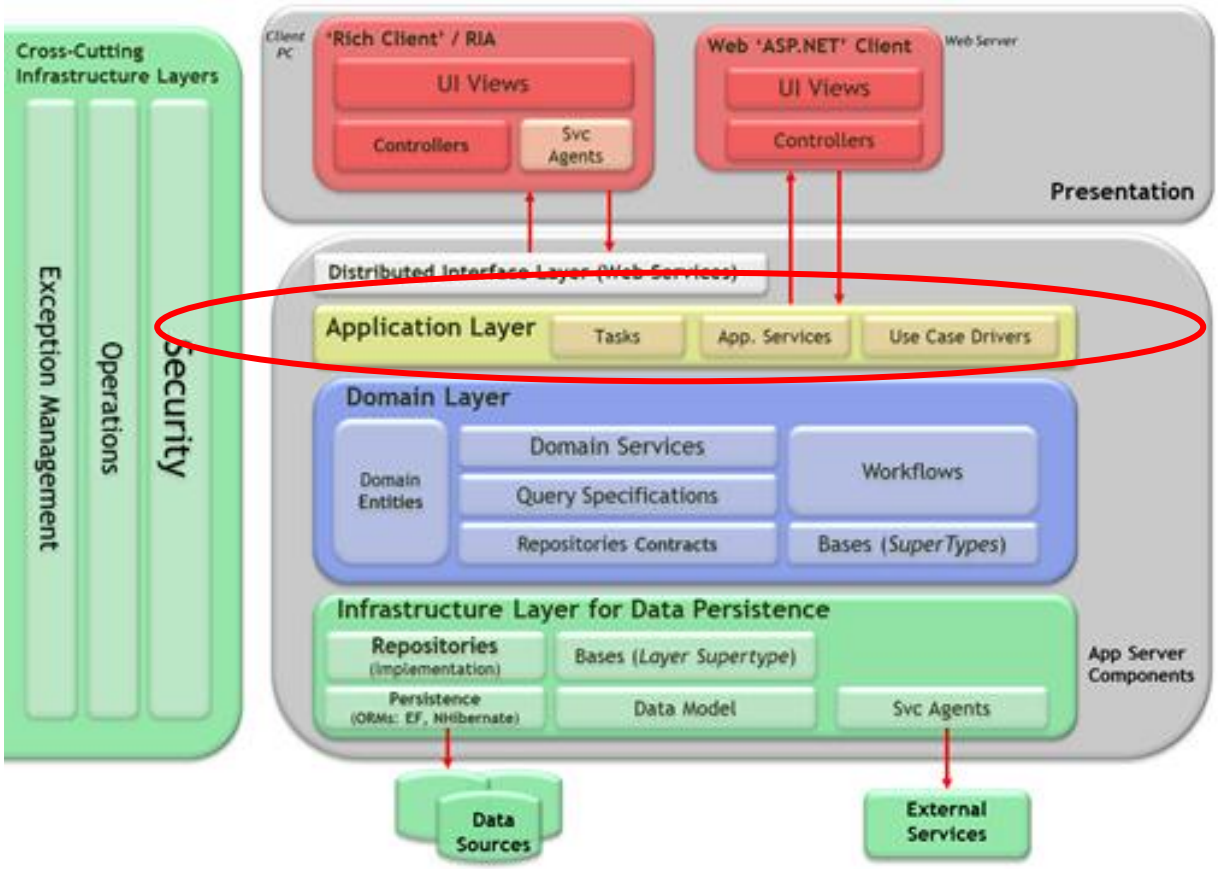
```
public class BankAccountNumberSpecification
       :Specification<BankAccount>
    {

        string _BankAccountNumber = default(String);

        public BankAccountNumberSpecification(string bankAccountNumber)
        {
            if (string.IsNullOrEmpty(bankAccountNumber)
                ||
                string.IsNullOrWhiteSpace(bankAccountNumber))
            {
                throw new ArgumentNullException("bankAccountNumber");
            }
```

```
            _BankAccountNumber = bankAccountNumber;
        }
        public override System.Linq.Expressions.Expression<Func<BankAccount, bool>>
    SatisfiedBy()
        {
            return ba => ba.BankAccountNumber == _BankAccountNumber;
        }
    }
```

Bien que ce pattern soit relativement simple il nécessite de la rigueur dans le suivi de la documentation.

4.3.3 Couche Application (Application Layer)

Avant de présenter les divers composants le diagramme ci-dessous présente le positionnement de cette couche dans l'architecture.



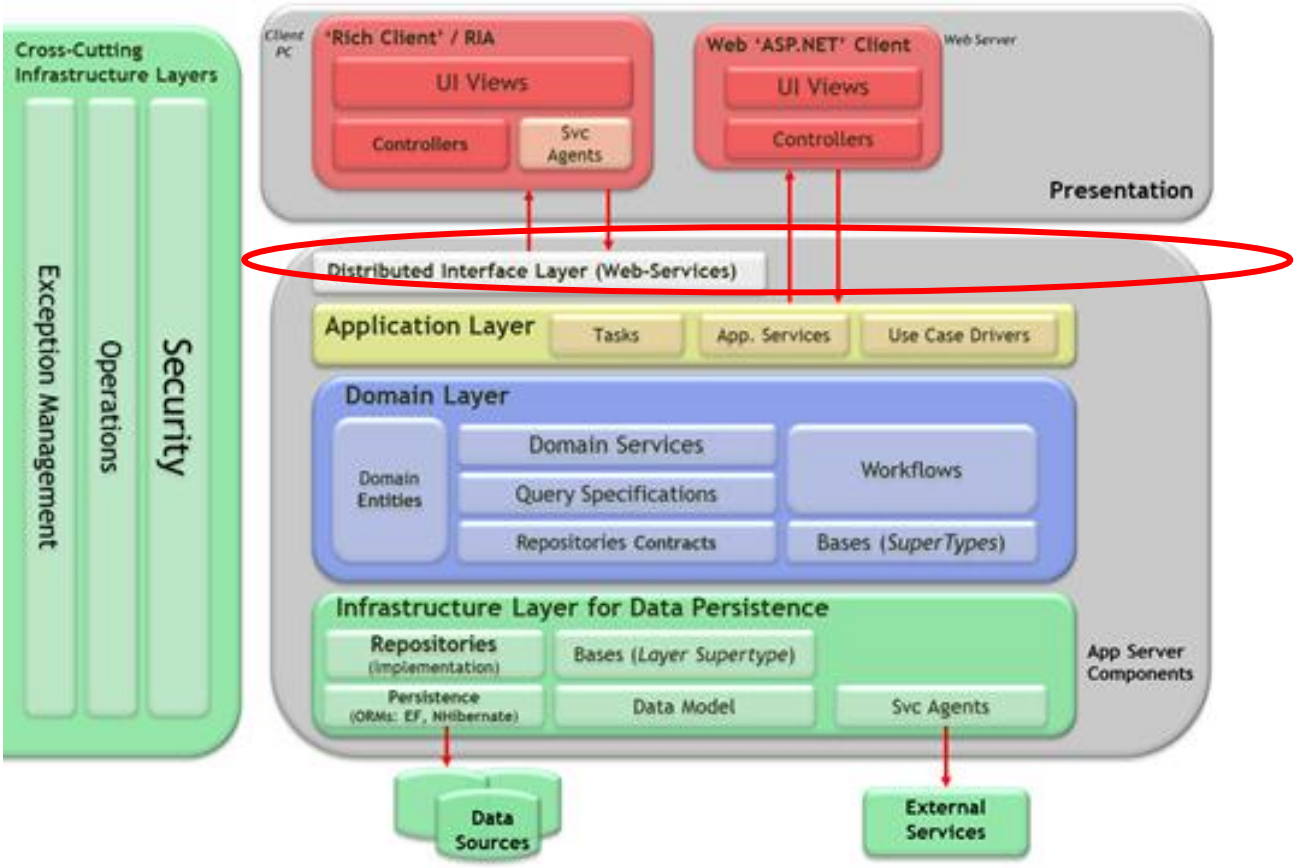
4.3.3.1 Composants de la couche Application

Cette couche est très fine. Elle sert principalement à permettre des abstractions entre le métier et la partie présentation. La nécessité de mettre en place des workflow sera adressée par workflow Foundation. Les services distribués appellent cette couche pour accéder aux couches inférieures.

Dans cette couche on trouvera la mise en œuvre du Framework UNITY, Framework qui permet les inversions de dépendances (Pattern IoC).

4.3.3.2 Couche de Services Distribués (Distributed Interface Layer)

Avant de présenter les divers composants le diagramme ci-dessous présente le positionnement de cette couche dans l'architecture.

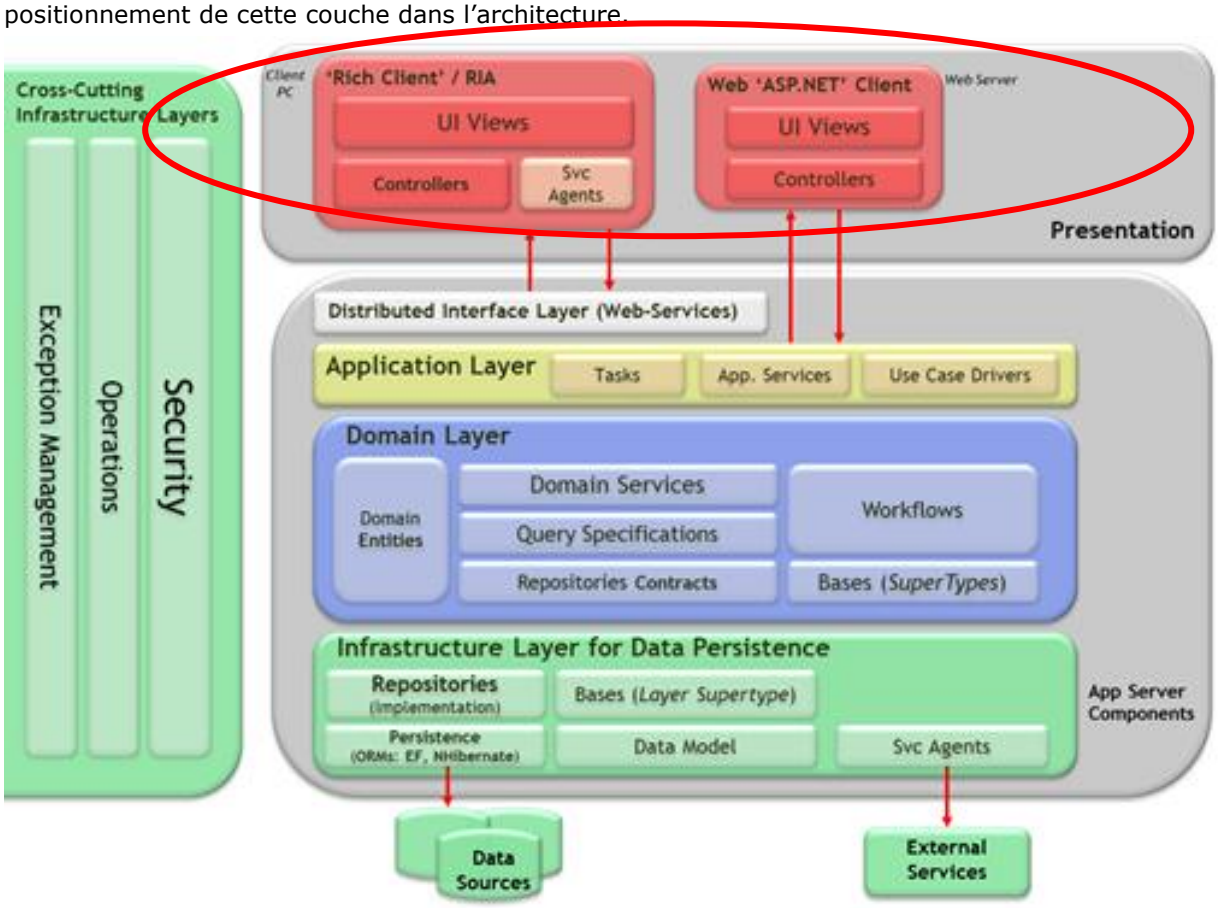


4.3.3.3 Implémentation de la couche de service distribué

Cette couche s'appuie sur WCF pour exposer les services. Actuellement les services sont exposés au travers de .Net Remoting, nous conserverons ce type d'exposition. Cependant les possibilités de WCF nous permettront si besoin d'exposer cela au travers de web services, si le besoin apparaît par la suite.

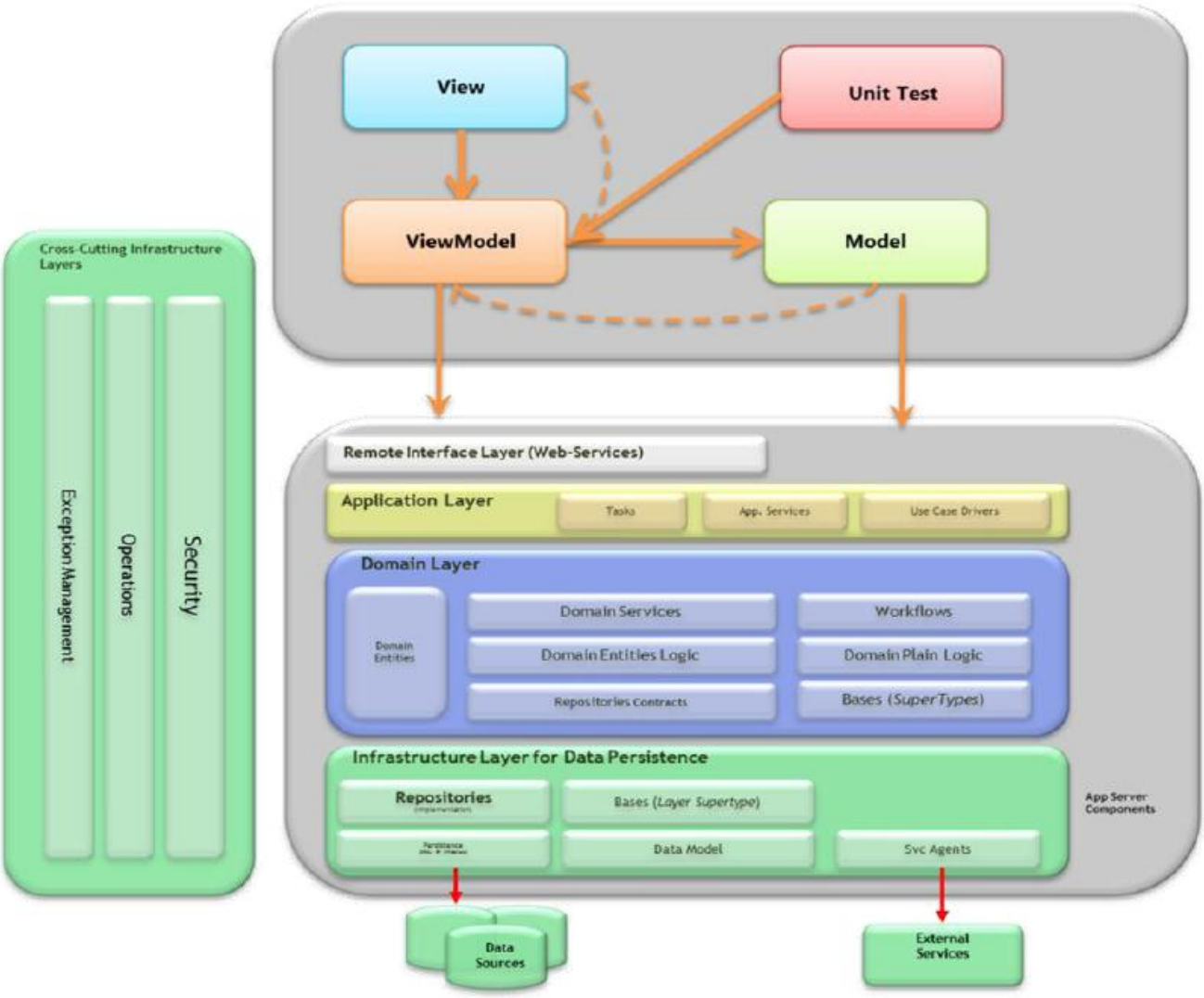
4.3.4 Couche de Présentation (Presentation Layer)

Avant de présenter les divers composants le diagramme ci-dessous présente le positionnement de cette couche dans l'architecture.



4.3.4.1 Pattern MVVM pour les applications WPF

Le design pattern MVVM est un dérivé des patterns MVC et MVP. Présentation de la mise en œuvre de MVVM dans l'architecture :



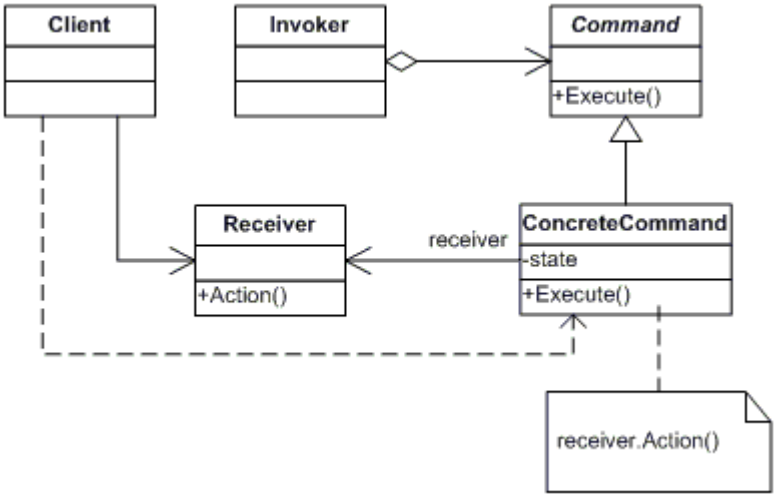
Après avoir présenté la vue d'ensemble nous allons détailler les étapes de la mise en œuvre du pattern MVVM. Nous allons commencer par présenter le pattern Commande utilisé dans MVVM, nous ferons un rappel du pattern Observer, enfin nous présenterons l'implémentation de cette couche selon MVVM et avec WPF 4.

4.3.4.1.1 Pattern Commande

Le design pattern de commande apporte une interface d'abstraction dans la gestion des opérations, cela permet à un client de faire exécuter une opération sans connaître ni le type ni l'implémentation. On utilise le pattern de commande pour :

- Pour paramétrer des objets en fonction de leurs actions
- L'objet commande a une durée de vie différente de la commande qui l'appelle
- L'objet commande peut gérer un état et avoir une action de rollback
- L'objet commande peut gérer un historique et permettre ainsi une récupération d'état en cas de crash de l'application
- L'objet commande permet de créer un système se basant sur des éléments de bases et appelés par des éléments de plus hauts niveaux.

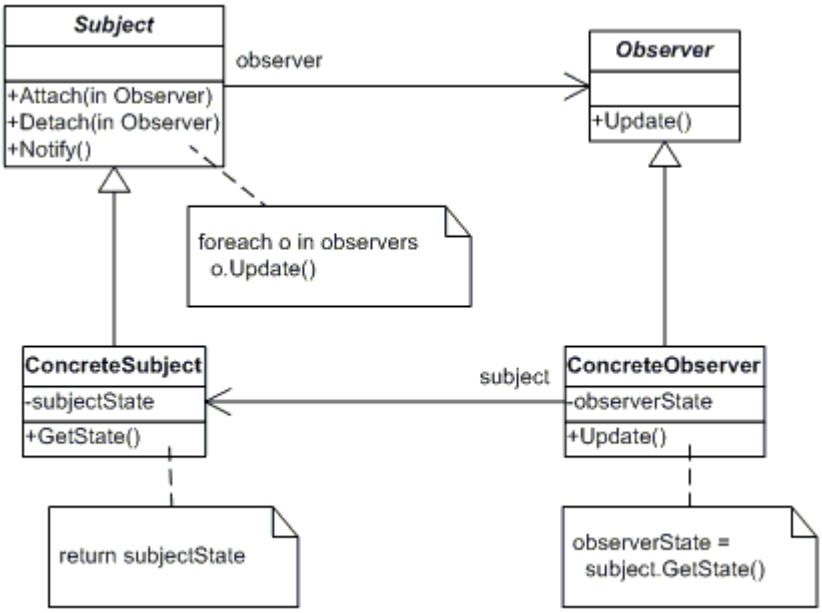
Le schéma UML suivant décrit l'implémentation d'un pattern de commande



4.3.4.1.2 Design pattern Observer

Ce design pattern permet à un objet de s'abonner aux événements d'un autre objet. Ce design pattern est utile pour informer les interfaces graphiques des événements tout en les laissant décider de leur volonté de s'abonner.

Ce design pattern est décrit par le schéma UML ci-dessous



4.3.4.1.3 Implémentation du pattern MVVM avec WPF 4

MVVM est donc un mix entre MVC et MVP. Ce qui est la vue est en fait un mélange de XAML et de C#. Le modèle lui représente les données disponibles et la partie VueModèle prépare le modèle pour le rendre utilisable par la vue.

La VueModèle est une abstraction qui agit comme une méta-vue (un modèle de vue), qui permet ainsi de mieux maîtriser la compartimentation entre les vues.

Avant d'introduire le concept de VueModèle, voici un schéma représentant l'implémentation de cela :

NB : on utilise une ObservableCollection pour ses capacités à communiquer avec WPF ce qui n'est pas le cas d'une List traditionnelle.

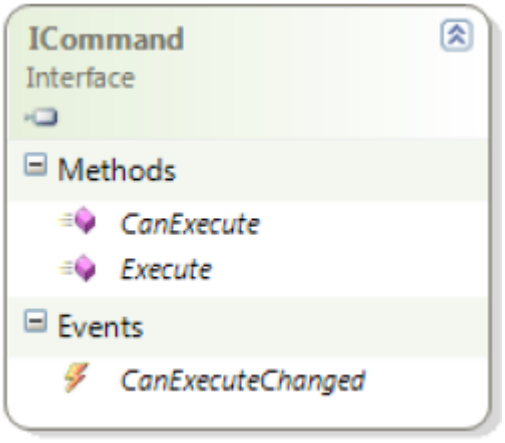
La VueModèle n'a aucune connaissance des vues l'utilisant, elle ne connaît que le modèle dont elle détient une référence. Bien sûr la VueModèle expose un pattern de Command pour permettre aux vues de travailler avec.

4.3.4.1.4 Utilisation de Commande avec MVVM WPF

WPF implémente le design pattern de Commande comme mécanisme de programmation d'entrée d'événements. Les commandes permettent de découpler l'origine de l'action avec l'avantage d'autoriser des sources multiples. Une commande n'est ni plus ni plus qu'une propriété d'un objet à déclencher un événement.

De plus le pattern de commande nous permet de gérer un affichage contextuel d'une commande en utilisant la fonction CanExecute(), fonction que l'on peut lier à la vue. Par exemple, on n'affichera pas de possibilité d'annuler si une commande de suppression n'a pas été exécutée.

Si dessous, nous présentons l'interface ICommand qui doit être utilisé dans le cadre du pattern Command.



Cette interface est constituée de deux méthodes et d'un événement.

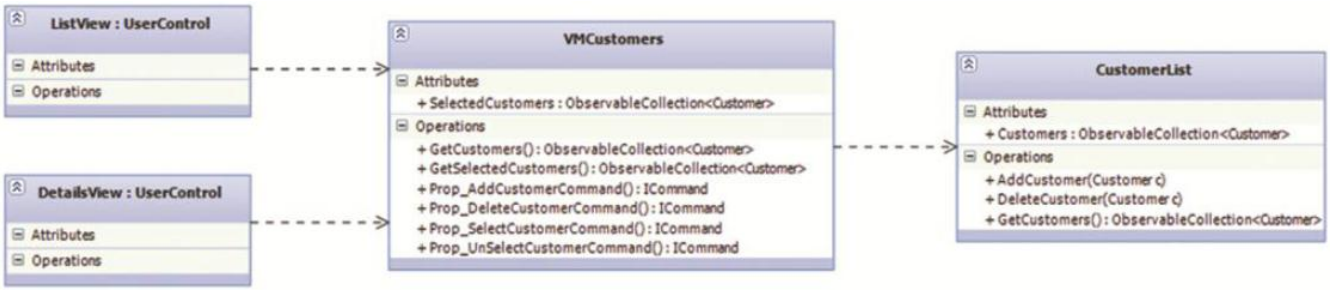
- Execute () : contient la logique de l'action à effectuer.
- CanExecute() : Permet de déterminer si l'état de la commande permet de l'exécuter ou non
- CanExecuteChanged() : se déclenche si quelque chose affecte ou non la capacité d'exécution de la commande.

Exemple d'implémentation de l'interface :

```
public class SaveCommand : ICommand
{
        private CustomerViewModel _view ;
        public SaveCommand (CustomerViewModel view)
{
_view = view ;
}

public bool CanExecute (object parameter)
{
return true ;
}

public event EventHandler  CanExecuteChanged ;

public void Execute(object parameter)
{
// Implémenter ici la logique de sauvegarde d'un client
}
}
```

Pour les événements provenant des vues, WPF fournit une classe spéciale appelée Routedcommand qui ne sait pas comment effectuer la tâche qu'elle représente. Quand on lui demande si elle peut fonctionner (CanExecute) et qu'on lui ordonne de s'exécuter (Execute), Elle délègue la responsabilité à d'autres. Les RoutedCommand traversent l'arbre visuel WPF donnant la possibilité pour chaque élément de l'interface utilisateur d'exécuter la commande qui fait le travail. De plus, tous les contrôles qui peuvent être utilisés "désactive" automatiquement la RoutedCommand lorsque vous ne pouvez pas exécuter d'autres actions.

Cependant, le design MVVM nous permet de d'introduire une classe légèrement différente : DelegateCommand, elle implémente ICommand et permet de se lier sur un délégué. Cette possibilité nous conduit à l'implémentation suivante :



Exemple d'implémentation :

```
public class VMCustomerList :ObservableObject
{
private ICommand  _editCommand ;
 }
Et ci-dessous le code XAML  qui utilise ce code
<UserControl>
…
<StackPanel>
…
<Button Content= « Button » Command= « {Binding EditCommand, Mode=OneWay} »
CommandParameter=  « {Binding SelectItem, ElementName=listBox>
…
</StackPanel>
…
</UserControl>
```

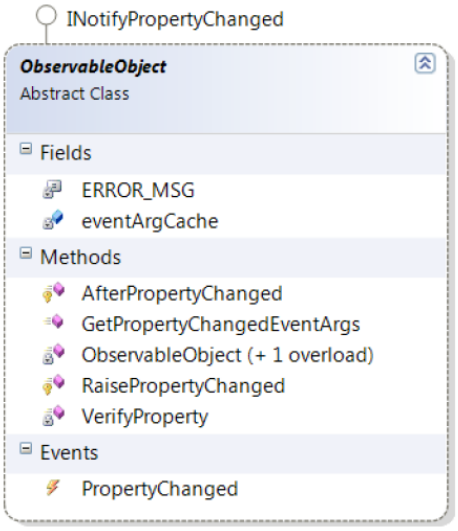
4.3.4.1.5 Utilisation de INotifyPropertyChanged avec MVVM WPF

Il existe en WPF une interface qui s'appelle INotifyPropertyChanged, qui peut s'implémenter pour notifier à l'utilisateur qu'un objet a été modifié, et que par conséquent l'interface doit actualiser ses données. Tout ce mécanisme de souscription réalise l'enchainement des données de WPF de façon automatique. Quand nous voulons rendre la collection d'un objet nous utilisons la collection observable (ObservableCollection). Mais quand nous devons passer du Model à la View, nous passons par le ModelView, un seul objet qui nous permet d'utiliser cette interface.

Cette interface définit un seul évènement, qui s'appelle PropertyChanged qui se lance pour informer du changement de propriété. C'est la responsabilité de chaque classe du modèle de lancer cet évènement quand il est opportun :

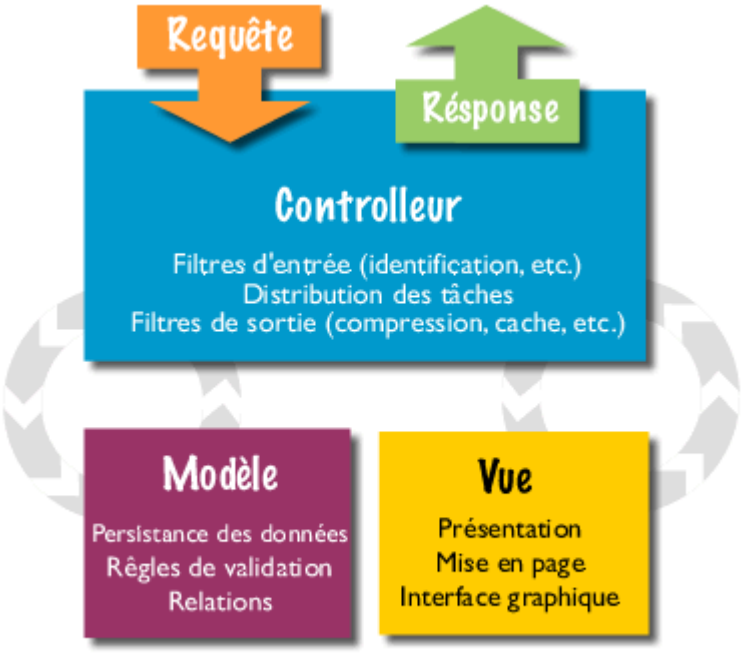
```
Public class A : INotifyPropertyChanged
{
Private string_name ;
//Évènement définit pour l'interface
Public Event PropertyChangedEventHandler PropertyChanged ;
// Lancement de l'évènement « PropertyChanged »
Private void NotifyPropertyChanged(string Info)
{
Var handler = this.propertyChanged;
If (handler != null)
{Handler (this, new PropertyChangedEventArgs(info));}
// Propriété qui informe du changement
Public string Name
{
Get { return _name ; }
Set
{If (_name != value)
{_name = value ;
NotifyPropertyChanged("Name");
}}}
```

Ce code est lourd à réaliser pour des classes avec beaucoup de propriétés et il est facile de commettre des erreurs. C'est pourquoi il est important de créer une petite classe qui nous permet d'éviter de répéter le code, nous vous recommandons de faire de la façon suivante :



4.3.4.2 Pattern MVC pour les applications ASP .NET (MVC 2)

Le Model-View-Controller (MVC) est un modèle de conception logicielle très répandu et fort utile. Créé dans les années 80 par Xerox PARC pour Smalltalk-80, il est aujourd'hui fortement recommandé dans les applications à forte IHM.



Diagrame de fonction du modèle MVC

4.3.4.2.1 Principe - Un modèle à trois couches

Le MVC est un modèle de conception qui repose sur la volonté de séparer les données, les traitements et la présentation. Ainsi l'application se retrouve segmentée en trois composants essentiels :

- Le modèle
- La vue
- Le contrôleur

Chacun de ces trois composants a un rôle bien défini.

Le modèle représente les données et les règles métiers. C'est dans ce composant que s'effectuent les traitements liés au cœur du métier. Les données peuvent être liées à une base de données, des services Web, … Il est important de noter que les données sont indépendantes de la présentation. En d'autres termes, le modèle ne réalise aucune mise en forme. Ces données pourront être affichées par plusieurs vues. Du coup le code du modèle est factorisé : il est écrit une seule et unique fois puis réutilisé par chaque vue.

La vue correspond à l'IHM. Elle présente les données et interagit avec l'utilisateur. Dans le cadre des applications Web, il s'agit d'une interface HTML, mais n'importe quel composant graphique peut jouer ce rôle.

Le contrôleur, quant à lui, se charge d'intercepter les requêtes de l'utilisateur, d'appeler le modèle puis de rediriger vers la vue adéquate. Il ne doit faire aucun traitement. Il ne fait que de l'interception et de la redirection.

4.3.4.2.2 Cinématique

- L'utilisateur émet une requête
- Le contrôleur intercepte la requête de l'utilisateur
- Le contrôleur détermine quelle partie du modèle est concernée et quelle vue y est associée
- Le modèle traite les interactions avec les données, applique les règles métier et renvoie les données au contrôleur
- Le contrôleur sélectionne la vue et lui renseigne les données
- La vue présente les données à l'utilisateur

4.3.4.2.3 MVC model 2

Le MVC très pratique, peut se révéler lourd à mettre en place. Ceci à cause de la multitude de contrôleur à implémenter. Afin de simplifier la réalisation d'un tel modèle, une nouvelle version a été introduite : le MVC2. C'est exactement le même modèle de conception à la différence qu'il n'y a plus qu'un seul contrôleur qui se charge de rediriger la requête vers le bon traitement.

4.3.4.2.4 Avantages

A l'époque des applications Web, il n'est pas rare que le développeur soit tenté de mettre du code de traitement dans les composants de présentation. Certains composants facilitent même ce genre de développement!! Le MVC impose cette séparation. Comme précisé plus haut, plusieurs vues peuvent utiliser le même modèle. Ce qui représente un gain en coût de développement important.

Le modèle est totalement indépendant de la vue. Si l'application a besoin d'un nouveau mode d'accès, le modèle restera inchangé. Il suffit juste de changer la partie IHM.

Le modèle étant totalement autonome, il peut être modifié beaucoup plus facilement. En effet si le mode de persistance des données change ou bien si des règles métier évoluent, il suffit de modifier seulement le modèle. La vue n'a pas besoin d'être modifiée dans ce cas.

Deux équipes peuvent travailler en parallèle. Une équipe d'infographistes peut travailler sur les vues et en même temps une équipe de développeurs peut travailler sur le modèle et le contrôleur. Cet aspect nécessite tout de même une bonne communication entre les deux entités.

Les trois couches doivent être réellement indépendantes et ne doivent communiquer que par des interfaces. Dans ce cas l'application sera très modulaire et n'importe quelle couche pourra être inter changée sans conséquence pour les autres.

Le contrôleur permet une très grande souplesse dans l'application. C'est lui qui assemble différentes parties du modèle avec une vue à partir d'une requête. S'il est maitrisé à la perfection, la factorisation des vues est envisageable. L'architecte peut alors s'amuser à en surprendre plus d'un développeur!!

4.3.4.2.5 Inconvénients

Le MVC se révèle trop complexe pour de petites applications. Le temps accordé à l'architecture peut ne pas être rentable pour le projet.

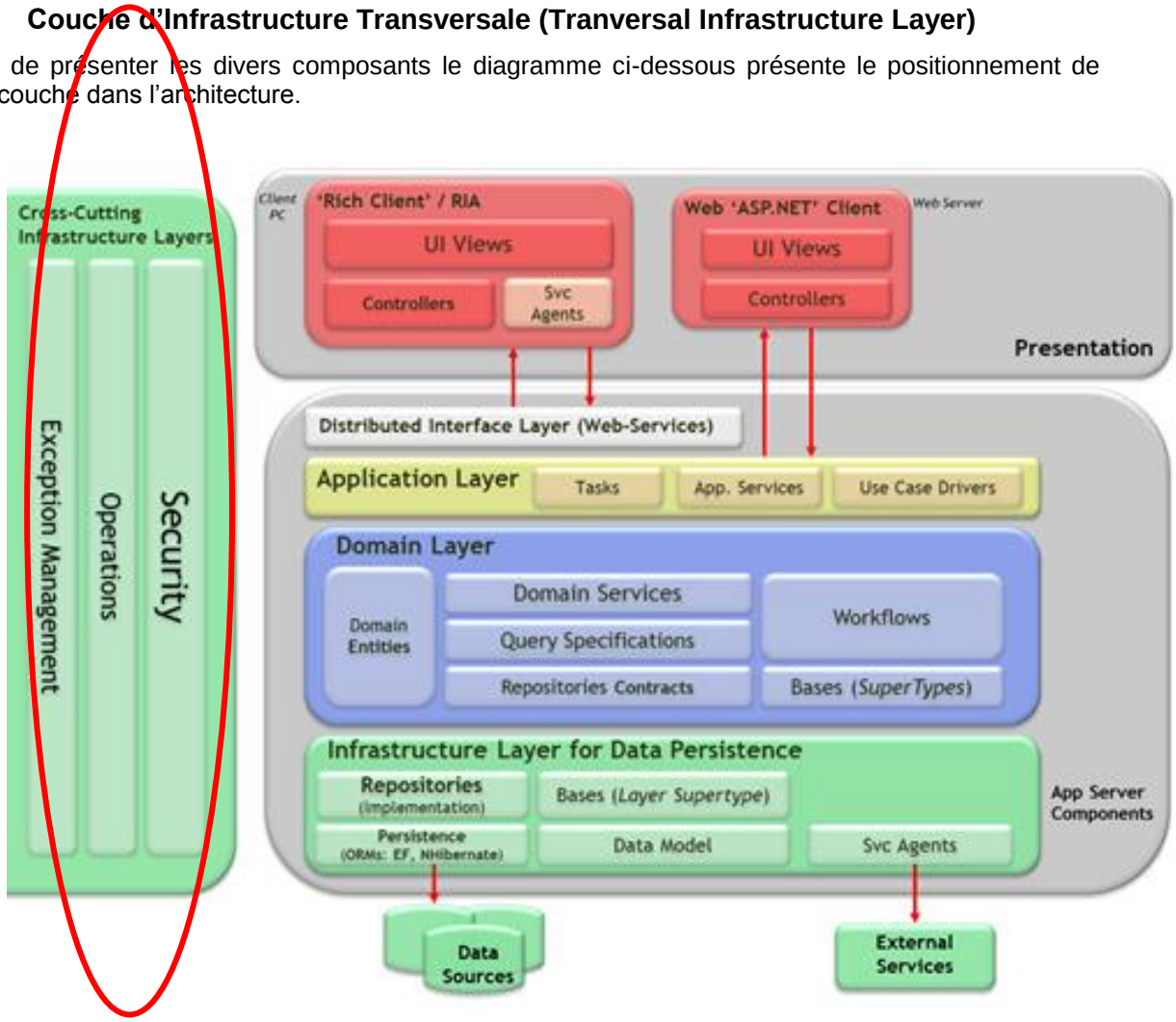
Même si le code est factorisé, le nombre de microcomposant n'en est pas moins augmenté. C'est le prix à payer pour la séparation des 3 couches. Et toutes les personnes qui font de la gestion de configuration comprendront que le nombre important de fichiers représente une charge non négligeable dans un projet.

4.3.4.2.6 Conclusion

Le MVC favorise le développement et la maintenance du code. Sur de gros projets et/ou avec de grandes équipes de développements, l'application d'un tel modèle de conception se révèle très performant. Il existe aujourd'hui des Frameworks très avancés qui se basent sur le MVC ou le MVC2. L'utilisation de ces Frameworks facilite sa mise en place et cadre sa réalisation.

4.3.5 Couche d'Infrastructure Transversale (Tranversal Infrastructure Layer)

Avant de présenter les divers composants le diagramme ci-dessous présente le positionnement de cette couche dans l'architecture.



4.3.5.1 Aspects Transversaux

Les aspects suivants sont les plus communs à considérer comme faisant partie des couches d'infrastructure transversale.

4.3.5.1.1 Sécurité Authentification / Autorisation

Uniquement en prenant en compte la sécurité des architectures et le développement d'applications, nous pourrions écrire un guide complet d'architecture, en effet il existe d'innombrables concepts de niveaux de sécurité (code sécurisé et failles de sécurité, communications sécurisées, chiffrement …). Dans ce guide nous ne couvrirons pas toutes les possibilités de la gestion de la sécurité dans les applications.

Cependant, il existe deux points fondamentaux de la sécurité d'une application qui doit être couvert. L'identité et l'authentification pour identifier les usagers de l'application et le concept d'autorisation pour la gestion des accès à l'application.

Enfin, pour l'authentification et la gestion des autorisations, au niveau architecture et technologie, il existe plusieurs possibilités. Depuis les types d'authentifications manuels Login / Password, en passant par l'authentification de domaine basée sur l'Active Directory, LDAPs, Certificats X.509, etc. ainsi que les méthodes et technologies pour implémenter l'autorisation (Orientation des rôles, Roless .NET Windows, Roles .NET Custom, Roles-Membership, Authorization Manager, WIF, etc.).

4.3.5.1.1.1 Authentification

La définition d'une authentification correcte est fondamentale pour la sécurité de l'application. Si elle n'est pas réalisée correctement, l'application peut être vulnérable aux attaques de « spoofing », attaques de dictionnaires, fermeture de session et d'autres types d'attaques.

- Identifier les frontières de confiance et authentifier les usages et appels qui transitent par les frontières de confiance. Il faut pouvoir authentifier aussi bien les appels provenant du client que du serveur (authentification mutuelle).
- Si on utilise l'authentification Utilisateur/Mot de passe, les mots de passe doivent être « Fort », c'est-à-dire remplir les conditions minimales de complexité (alphanumérique, longueur minimum, inclusion de chiffres, etc.)
- Ne pas transmettre de mots de passes en clair sur le réseau et ne pas stocker les mots de passe en clair en base de données. Il faut garder les mots de passe cryptés.

Pour prendre en charge l'ensemble de ces contraintes nous utiliserons des mots de passes cryptés en base de données par un système de hachage. C'est-à-dire que lors de la perte d'un mot de passe nous générerons un nouveau mot de passe temporaire. Ce mot de passe aura une durée de validité de 24h. Il sera envoyé sur l'adresse mail de l'utilisateur. Ce dernier devra donc en avoir fournit une obligatoire au moment de son inscription.

Les appels faisant transiter les informations de login / password seront mise dans un appel SSL afin de préserver la confidentialité des données.

4.3.5.1.1.2 Autorisations

Définir une bonne gestion des autorisations est fondamental pour la sécurité d'une application. Car même si la gestion des authentifications est correcte, elle ne sert à rien si la gestion des autorisations est défaillante. Cela peut conduire à rendre vulnérable votre application à l'élévation de privilège et aux manipulations de données non autorisées.

Identifier les frontières de confiance, les autorisations des.

Protéger les recours aux autorisations des appels basés sur l'identité, les groupes, les rôles et aussi, idéalement, les Claims. Minimiser le nombre de rôles quand cela est possible.

4.3.5.1.2 « Permission et Rôles » versus « Rôles »

Si la gestion des autorisations de l'application est complexe, il faut considérer l'utilisation d'une granularité plus fine qu'uniquement groupes/rôles. C'est-à-dire, utiliser les permissions requises pour accéder à une ressource. Sont assignées aux rôles de l'application des permissions et des utilisateurs. L'utilisation des permissions est alors plus puissante.

4.3.5.1.3 Cache

Le Cache peut augmenter drastiquement le rendement et le degré de réponse d'une application. Il faut simplement connaitre de façon précise comment utiliser ou non le Cache. Il faut aussi avoir à l'esprit qu'une mauvaise utilisation du Cache peut dégrader le rendement et la stabilité d'une application.

Il faut utiliser le cache pour optimiser la recherche de données, éviter les communications à distance et en général éviter les processus dupliqués. Quand on implémente le cache il faut définir quand nous devons sauvegarder les données en cache et quand nous devons éliminer les données calculées.

Il est préférable de pré-charger les données utilisées fréquemment et le faire de façon asynchrone ou utiliser le processus « Batch », qui évitent les délais pour le client.

- Emplacement du Cache : Il est fondamental de choisir correctement l'emplacement du Cache. Si l'application est déployée dans une ferme de serveurs web (Web-Farm), il faut éviter d'utiliser le Cache local à chaque nœud de la ferme (comme les sessions ASP.NET dans l'espace de mémoire des processus des services web), puisque nous ne pourrons pas balancer ces cluster-software. Il faut donc, considérer l'utilisation de cache distribué et synchronisé entre les différents serveurs de façon automatique.

- Cache en format structuré : Il faut utiliser un format structuré quand on utilise le cache. Par exemple au lieu de mettre en cache simplement des données textes simples, il vaut mieux mettre en cache des objets sérialisables qui en même temps agissent comme des entités.
- Ne pas mettre en cache des données volatiles et ne jamais mettre en cache des données sensibles sans qu'elles soient cryptées.
- Pour les opérations d'une certaine durée et la concaténation de sous opérations critiques, il ne faut pas dépendre de l'existence de données dans le cache, car elles pourraient être supprimées. Il faut implémenter un mécanisme de gestion de manque de cache, par exemple en rechargeant un élément dans sa forme originale.
- Il faut faire particulièrement attention quand on accède au cache depuis des Threads multiples. Dans ce cas, il faut être sûr que tous les accès au cache sont « Thread Safe » de façon à maintenir la consistance, en utilisant les mécanismes de synchronisation.

4.3.5.1.4 Gestion de configuration

La définition d'un mécanisme de configuration approprié est importante pour la sécurité et la flexibilité des applications. Si elle n'est pas réalisée correctement nos applications pourraient être vulnérables aux attaques et pourraient devenir une charge de travail inutile aux administrateurs des applications.

- Il faut considérer avec attention que les caractéristiques doivent pouvoir être configurables de façon externe. Vérifier que réellement il y a une nécessité de pour chaque option soit configurable. Une complexité excessive de la configuration peut amener à un système très complexe à administrer et à maintenir. Ce qui peut provoquer des disfonctionnements et des trous de sécurité
- Il faut décider si la configuration se gère de façon centralisée ou si elle s'applique aux utilisateurs au moment de lancer l'application (par exemple basée sur une politique Active Directory). Il faut considérer comment se restreint l'accès à l'information de configuration et utiliser des processus s'exécutants avec les mêmes privilèges possibles prenant en compte un service correctement configuré.
- Il faut chiffrer l'information sensible dans le magasin de configuration. Par exemple, les parties chiffrées à l'intérieur d'un fichier .Config.
- Il faut catégoriser les éléments de configurations regroupées en différentes sections logiques dépendant de l'usage de chaque configuration.
- Il faut catégoriser aussi les éléments de configuration en sections logiques si l'application possède plusieurs niveaux physiques (Tiers).Si le serveur d'application s'exécute dans une « Web-Farm », il faut définir quelles parties de la configuration sont partagées et lesquelles sont spécifiques pour chaque machine au sein de laquelle l'application s'exécute. Il faut définir un magasin approprié pour chaque section.
- Il faut fournir une interface utilisateur pour les administrateurs à travers laquelle ils peuvent éditer les informations de configuration (Applications du type Snap-in sur la MMC, de l'édition de configuration, par exemple).

4.3.5.1.5 Gestion des Exceptions

Définir une bonne stratégie de gestion des exceptions est important pour la sécurité et la stabilité d'une application. Si elle n'est pas réalisée correctement, il peut être difficile de diagnostiquer et résoudre les problèmes dans une application. De plus cela peut rendre l'application vulnérable aux attaques comme Dos (Denial of Service) et monter des informations sensibles provenant d'exceptions internes.

Une bonne approche est de définir un mécanisme centralisé de gestion des exceptions et prévoir de fournir des points d'accès au système de gestion des exceptions (comme les évènements WMI) pour supporter la monitorisation au niveau de l'entreprise comme « Microsoft System Center ».

- Il faut définir une stratégie appropriée à la propagation des exceptions qui enveloppe ou remplace les exceptions (erreurs internes), ou ajoute des informations Par exemple, permettre que les exceptions montent dans les couches supérieures jusqu'à arriver aux couches frontières (comme les services web ou la couche de présentation Web ASP.NET), où les exceptions sont enregistrées (logs) et transformées si nécessaire avant de passer la couche suivante (normalement avant d'arriver à la couche de présentation ou l'interface graphique utilisateur).
- Il faut inclure un identifiant de contexte de manière à ce que les exceptions connectées puissent être associées le long des différentes couches et que l'on puisse identifier la cause des erreurs ou des échecs. Il faut aussi s'assurer que les exceptions non gérées (unhandled exceptions) sont prises en compte.
- Ne pas faire de « Catch() » des exceptions/erreurs internes à moins de les gérer ou que l'on veut ajouter plus d'information.
- Ne jamais utiliser les exceptions pour contrôler le flux de l'application.
- Définir une stratégie appropriée de registre (logging) et notifier les erreurs critiques, qui montrent suffisamment d'informations sur le problème pour permettre aux administrateurs de l'application de recréer le scénario. Il ne faut pas pour autant révéler des informations confidentielles à l'utilisateur final (ni dans les messages qui arrivent à l'utilisateur, ni dans les Logs).

4.3.5.1.6 Logging

La définition d'une bonne stratégie de Logging et d'instrumentalisation est importante pour la sécurité et le diagnostic des applications. Si ce n'est pas réalisé correctement, l'application peut être vulnérable à des menaces de répudiation, où les utilisateurs nient leurs actions et les fichiers de Log peuvent être requis pour des procédures légales qui peuvent prouver leurs actions. Il faut auditer et enregistrer l'activité de l'application dans les différentes couches aux points clés qui peuvent aider à détecter des activités suspectes et fournir rapidement des indications sur des attaques sérieuses. Les audits sont mieux considérés s'ils sont produit au moment précis de l'accès aux ressources et par le même algorithme qui accède à la ressource.

- Il faut définir un système centralisé de Registre/Logging qui capture les évènements les plus critiques. Éviter de faire un registre par défaut trop granulaire (qui générerai trop de volume), mais il faut prévoir de pouvoir changer la configuration au moment de l'exécution et alors générer une information plus détaillée.
- Il faut créer une politique de sécurité de gestion des registres/logs. Il ne faut pas garder d'informations sensibles d'accès non autorisé, dans les fichiers de Log. Il faut définir comment vont accéder et passer les données de registre et d'audit de façon sécurisé entre les différentes couches.
- Il faut permettre différent types de traces (Trace listeners), de façon qu'elles puissent être extensibles à d'autres types de fichiers ou de registres, et modifiables pendant l'exécution.

4.3.5.1.7 Instrumentation

L'instrumentation peut s'implémenter au moyen de compteurs de rendement et d'évènements qui fournissent aux administrateurs les informations sur les états, le rendement et la santé d'une application.

4.3.5.1.8 Gestion des Etats

La gestion des états (sessions, etc.) est en relation avec la persistance des données qui représentent les états d'un composant, opération ou passage dans un processus. Les données d'état peuvent persister sous différents formats et sous de multiples formes. La définition du mécanisme de gestion des états peut affecter le rendement de l'application, la maintenance de petits volumes d'information d'état peut affecter négativement le rendement et la faculté de l'application à pouvoir fonctionner correctement. Doivent seulement persister les données qui nécessitent réellement de persister et il faut connaitre toutes les possibilités de la gestion des états.

- Il faut maintenir la gestion des états aussi propre que possible, conserver la quantité minimale d'information pour maintenir un état.
- Il faut s'assurer que les données des états sont sérialisables si elles doivent être conservées ou réparties entre différents processus et frontières du réseau.
- Conserver les états dans l'espace de mémoire d'un processus est la technique qui offre le meilleur rendement, mais seulement si les états ne doivent survivre au processus ou au redémarrage du serveur. On peut conserver les états sur un disque local ou une base de données si l'on veut disposer des données après la fin du processus ou après le redémarrage du serveur.
- Au niveau de la technologie à utiliser pour fournir les états entre différents serveurs, les deux plus puissantes sont :
    1. Utiliser le système de Cache/Etat que supporte le Web-Farm et synchroniser automatiquement des données cache entre les différents serveurs.
    2. Utiliser un magasin central basé sur une base de données. Cette option baisse le rendement pour obtenir les données persistantes physiquement.

On retiendra la gestion des sessions en base de données dans le contexte des applications PACA.

4.3.5.1.9 Validation

La définition du système de validation des données d'entrées est fondamentale pour l'utilisation et la stabilité de l'application. Si elle n'est pas réalisée correctement, on peut retrouver dans notre application des données inconsistantes, des violations de règles de gestion et une expérience dégradée pour l'utilisateur. De plus, cela peut ouvrir des trous de sécurité comme les attaques « Cross-Site Scripting), attaques d'injection SQL, etc.

Malheureusement il n'existe pas de définition standard qui permet de différencier les données d'entrées valides des données d'entrées pernicieuses. De plus, la façon dont l'application utilise les données d'entrées influence complètement les risques associés à l'exploitation de la vulnérabilité.

- « Toutes les données d'entrées peuvent être pernicieuses, tant que le contraire n'est pas démontré ».
- Pour valider les données d'entrées il faut prendre en compte la longueur permise, le format, le type de données et le rang permis.
- La liste des données permises contre liste des données bloquées : Si cela est possible, il faut définir un système de validation pour permettre la création d'une liste qui définit spécifiquement les données d'entrée acceptables, au lieu de définir une liste des données d'entrées qui ne sont pas acceptables ou qui peuvent compromettre le système. Il est plus facile d'ouvrir à postériori le rang de portée d'une liste de valeurs permises que de diminuer une liste de données bloquées.
- La validation en client serveur : Il ne faut pas uniquement confiée la validation des données d'entrées côté client. Au lieu de cela, il faut faire une validation cliente pour offrir une réponse rapide à l'utilisateur et améliorer ressenti. Mais il faut aussi implémenter une validation côté serveur pour vérifier les données d'entrées incorrectes ou les entrées pernicieuses qui ont sautés la validation de la couche client.

- Il faut centraliser l'approche de validation en composants séparés, si la logique peut être réutilisée, ou considérer l'utilisation de bibliothèques tierces. De cette façon, les mécanismes de validation s'appliquent de façon consistante et homogène pour toute l'application.
- Il faut s'assurer que les données de l'utilisateur soient restreintes, repoussées et/ou nettoyée.

4.3.5.2 Implémentation en .Net des éléments transversaux

4.3.5.2.1 Implémentation du logging

Il existe des librairies à caractère général, comme la « Enterprise Library de Microsoft Pattern & Practices » qui sont très utiles pour implémenter un Login complexe, avec différentes possibilités.

Quelques unes des implémentations intéressantes sont :

- Microsoft Enterprise Library\Logging Building Block
- NLog
- Log4net

Pour nos applications nous utiliserons Enterprise Library et le Logging Building Block

4.3.5.2.2 Implémentation de la validation

Il existe des librairies à caractère général, comme la « Enterprise Library de Microsoft Pattern & Practices » qui sont très utiles pour implémenter un système réutilisable de validation des données d'entrées («Microsoft Patterns & practices Enterprise Library Validation Block).

4.4 WCF 4

.NET 4 vient avec de nouvelles fonctionnalités et de nombreuses améliorations concernant WCF (Windows Communication Foundation). Ces améliorations sont principalement en direction des développeurs afin de faciliter leurs tâches. L'une d'entre elle est notamment l'amélioration du travail avec WF (Workflow Foundation), en facilitant la création de « service sous forme de workflow ».

Ce document présentera WCF 4 et les façons de l'utiliser.

4.4.1 Qu'y a-t-il de nouveau dans WCF 4

Les nouveaux éléments fournis par WCF 4 sont listés dans le tableau de la figure 1.

Figure 1: WCF 4 Nouvelles fonctionnalités

Fonctionnalités	Description
Simplification de la configuration	Simplification de la configuration de WCF, notamment par le support de configuration par défauts de : endpoints, binding et behavior. Ces changements permettent entre autre d'avoir des services sans configuration, ce qui simplifie grandement la tâche du développeur.
Découverte de service	Le nouveau Framework supporte deux comportements de découverte de service ad hoc et manage. Cette fonctionnalité est conforme au standard WS-Discovery.
Routage de service	Possibilité de créer des tables de routage pour les appels de services.
Améliorations REST	Simplification des développements de service REST.
Workflow Services	Meilleure intégration de WF et WCF pour fabriquer des workflows exposés sous forme de services.

Maintenant que nous avons couvert les nouvelles fonctionnalités nous allons nous focaliser sur les éléments de WCF 4 utilisés dans l'architecture de la région PACA.

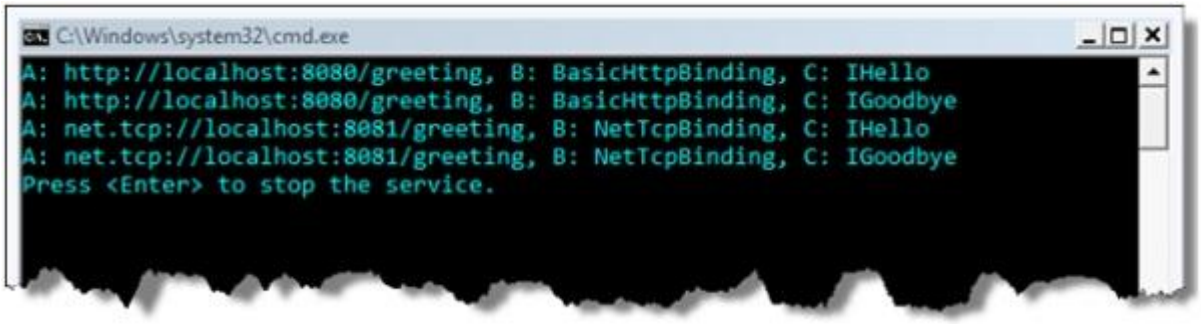
4.4.1.1 Default Endpoints

Avec WCF 4, on peut maintenant utiliser ServiceHost pour héberger un service sans avoir besoin de configurer une application complète. Quand on utilise ServiceHost, on va avoir besoin de spécifier une ou plusieurs adresses de base utilisable. L'exemple suivant montre comment héberger un service appelé GreetingService dans une application console sans pour pourtant qu'il y ait un fichier de configuration :

```
[ServiceContract]
public interface IHello
{
   [OperationContract]
   void SayHello(string name);
}
[ServiceContract]
public interface IGoodbye
{
   [OperationContract]
   void SayGoodbye(string name);
}
public class GreetingService : IHello, IGoodbye // le service implémente les deux contrats
{
   public void SayHello(string name)
   {
      Console.WriteLine("Hello {0}", name);
   }
   public void SayGoodbye(string name)
   {
      Console.WriteLine("Goodbye {0}", name);
   }
}
class Program
{
   static void Main(string[] args)
   {
      // Le Host est configuré avec deux adresses de base une en HTTP une en TCP
      ServiceHost host = new ServiceHost(typeof(GreetingService),
         new Uri("http:///localhost:8080/greeting"),
         new Uri("net.tcp://localhost:8081/greeting"));
      host.Open();
      foreach (ServiceEndpoint se in host.Description.Endpoints)
         Console.WriteLine("A: {0}, B: {1}, C: {2}",
            se.Address, se.Binding.Name, se.Contract.Name);
      Console.WriteLine("Press <Enter> to stop the service.");
      Console.ReadLine();
      host.Close();
   }
}
```

Cet exemple configure le ServiceHost avec deux adresses de base une en HTTP l'autre en TCP. Quand on exécute ce programme on voit les 4 endpoints affichés sur la console comme illustré sur la figure 2. Il y a deux endpoints pour l'implémentation HTTP et 2 endpoints pour le TCP, finalement une par contrat. Cette prise en charge est fournie par ServiceHost instance.

Figure 2: Default endpoints affichés dans la console



A noter que WCF choisi d'utiliser le BasicHttpBinding pour le endpoint par défaut HTTP et le NetTcpBinding pour le endpoint par défaut TCP. On montre ci dessous comment modifier ces comportements par défaut.

Il faut se rappeler que ce comportement par défaut n'agit que tant que le service n'a pas eu de configuration d'endpoint. Si dans l'application console on configure l'exposition de l'un de ces endpoints, on ne voit plus alors de endpoint par défaut. Pour illustrer cela on ajoute une simple ligne de code qui appelle AddServiceEndpoint après la construction de l'instance de ServiceHost :

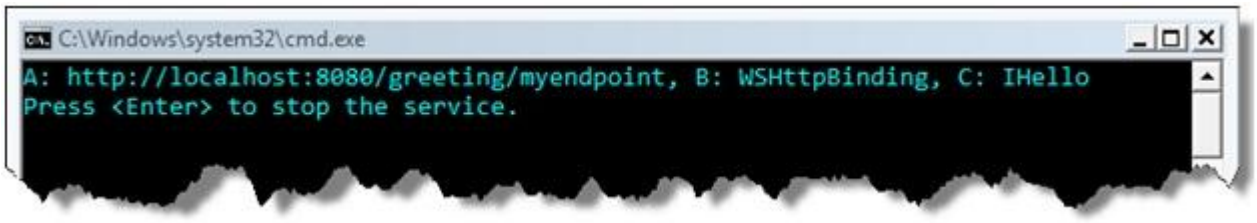
…

```
ServiceHost host = new ServiceHost(typeof(GreetingService),
    new Uri("http://localhost:8080/greeting"),
    new Uri("net.tcp://localhost:8081/greeting"));
host.AddServiceEndpoint(typeof(IHello), new WSHttpBinding(), "myendpoint");
```

…

Si on lance l'application console avec cette ligne de code, on note qu'un seul endpoint apparaît désormais.

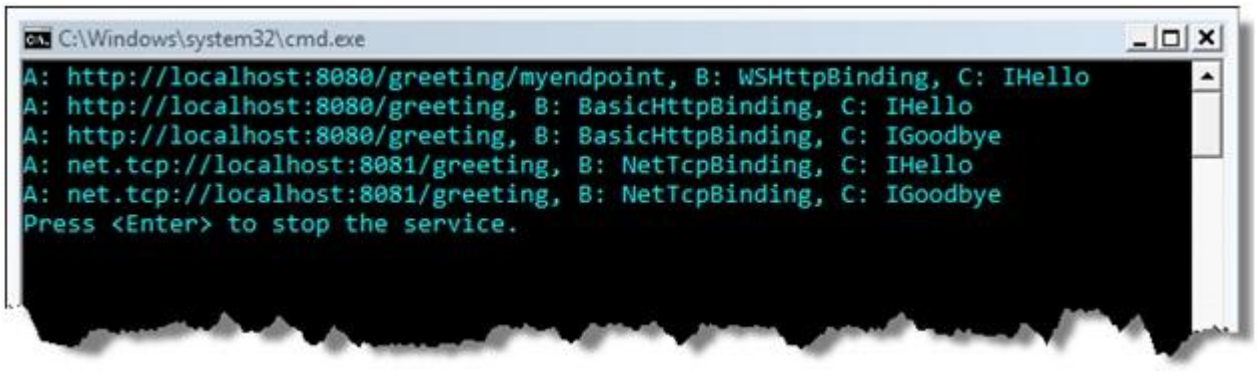
Figure 3: Sortie Console après que l'on ait configure le endpoints



On peut toujours faire ajouter après coup les endpoints par défaut simplement en demandant explicitement leur ajout. On utilisera pour cela AddDefaultEndpoints. Le code suivant illustre cela :

```
ServiceHost host = new ServiceHost(typeof(GreetingService),
    new Uri("http://localhost:8080/greeting"),
    new Uri("net.tcp://localhost:8081/greeting"));
host.AddServiceEndpoint(typeof(IHello), new WSHttpBinding(), "myendpoint");
host.AddDefaultEndpoints();
```

Figure 4: Sortie de la Console après l'ajout de AddDefaultEndpoints manuellement



Nous allons maintenant présenter la mise en œuvre des bindings par défaut.

4.4.1.2 Relation entre les protocoles par défaut

La réponse à cette question est simple WCF défini un protocole par défaut pour chaque protocole de transport. Le protocole par défaut est renseigné dans le fichier machine.config, cette description ressemble a :

```
<system.serviceModel>

  <protocolMapping>

    <add scheme="http" binding="basicHttpBinding" bindingConfiguration="" />

    <add scheme="net.tcp" binding="netTcpBinding" bindingConfiguration=""/>

    <add scheme="net.pipe" binding="netNamedPipeBinding" bindingConfiguration=""/>

    <add scheme="net.msmq" binding="netMsmqBinding" bindingConfiguration=""/>

  </protocolMapping>

  …
```

On peut surcharger cette définition de deux façons :

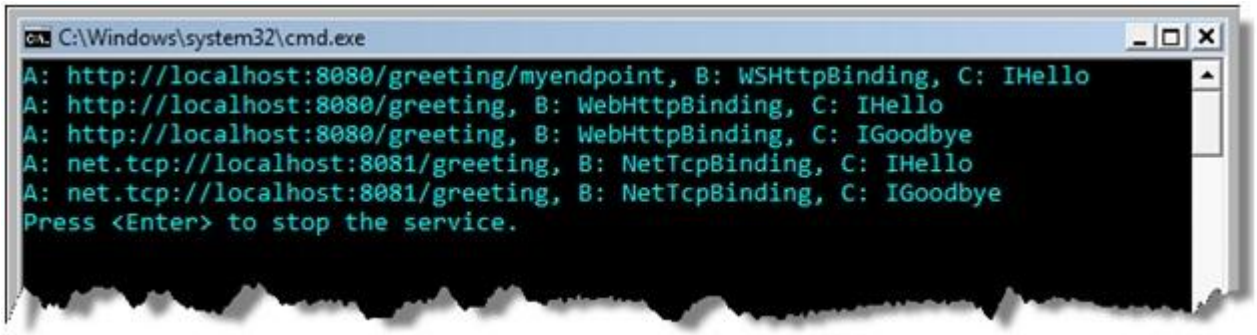
- Soit en réécrivant le machine.config cela impactera la configuration du serveur
- Soit en redéfinissant la section dans le fichier de configuration en redéfinissant la section dans le fichier web.config par exemple.

Par exemple, si l'on souhaite avoir comme binding par défaut un système RESTful, cela pour avoir du sens de redéfinir les bindings par défauts sur WebHttpBinding. L'exemple suivant illustre cela :

```
<configuration>

 <system.serviceModel>

  <protocolMapping>

    <add scheme="http" binding="webHttpBinding"/>

  </protocolMapping>

 </system.serviceModel>

</configuration>
```

Maintenant si on relance l'application console du début on verra apparaître la nouvelle configuration.

Figure 5: Sortie de la Console après la surcharge du mapping HTTP



4.4.1.3 Default Binding Configurations

Chaque binding WCF vient avec une configuration par défaut qui est utilisée tant qu'elle n'est pas surcharge dans l'application.

En WCF 3.x, on fait cela en définissant une configuration de binding nommée que l'on peut ensuite appliquer en utilisant l'attribut bindingConfiguration. Voici un exemple de définition :

```
<configuration>
  <system.serviceModel>
    <bindings>
      <basicHttpBinding>
        <binding name="BasicWithMtom" messageEncoding="Mtom"/>
      </basicHttpBinding>
    </bindings>
    <services>
      <service name="GreetingService">
        <endpoint address="mtom" binding="basicHttpBinding"
                  bindingConfiguration="BasicWithMtom"
                  contract="IHello"/>
      </service>
    </services>
  </system.serviceModel>
</configuration>
```

Dans l'exemple ci dessus, la configuration de binding « BasicWithMtom » surcharge les valeur par défaut de la configuration de binding « BasicHttpBinding » en changeant l'encodage du message en MTOM. Quoi qu'il en soit cette configuration ne prend effet que lorsqu'on l'applique au endpoint en utilisant l'attribut « bindingConfiguration » – c'est une étape souvent oubliée qui explique certains problèmes de configuration.

Avec WCF 4, on peut maintenant définir des bindings de configuration par défaut simplement en omettant de préciser le binding. Dans ce cas WCF va utiliser la configuration par défaut.

Par exemple, si nous ajoutons le fichier de configuration suivant à l'application console ci-dessus, les deux endpoints http vont prendre la configuration par défaut décrite ci-dessous qui active MTOM :

```
<configuration>
  <system.serviceModel>
    <bindings>
      <basicHttpBinding>
        <binding messageEncoding="Mtom"/> <!-- il n'y a pas d'attribut name-->
      </basicHttpBinding>
    </bindings>
  </system.serviceModel>
</configuration>
```

Bien sûr, on peut aussi surcharger les bindings par défaut dans le machine.config, ou au niveau application.

Cette fonctionnalité permet donc de définir une fois pour toute les comportements des services, les développeurs n'ayant plus besoin de s'en préoccuper.

Après avoir vue les endpoints et les bindings par défaut nous allons voir les behaviors par défaut.

4.4.1.4 Configuration des behaviors par défaut

WCF 4 rend possible la définition de configuration de behaviors par défaut pour les servies et les endpoints, cela permet de simplifier les choses quand on veut partager un comportement (behavior) standard à travers tout les services et les endpoints fonctionnant sur un serveur (cela fonctionne aussi pour une solution Visual Studio).

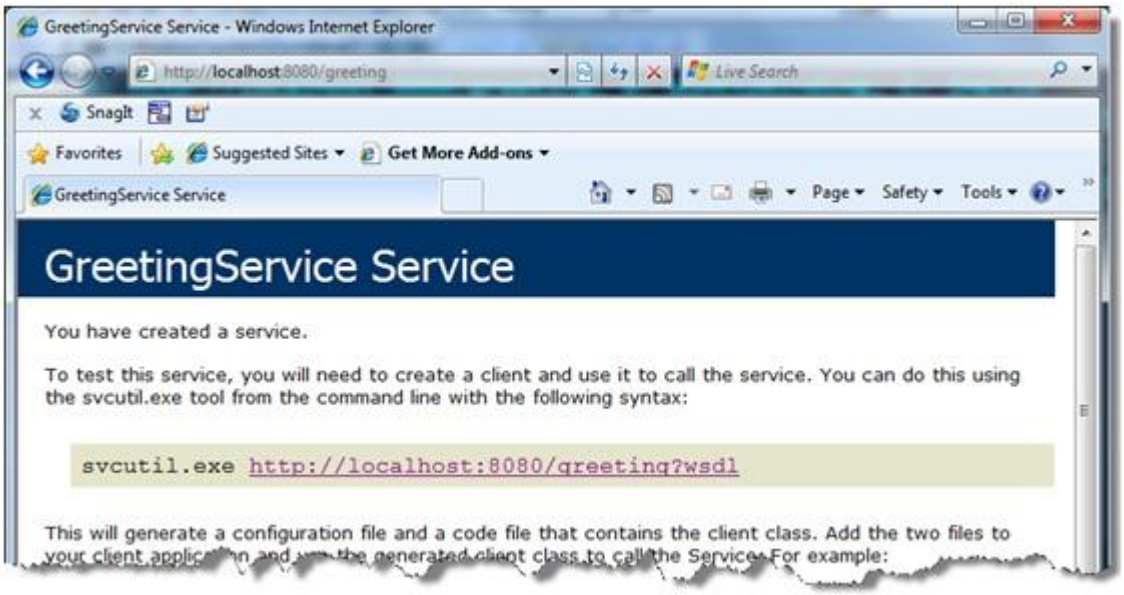
En WCF 3.x, on doit définir une configuration de behavior nommée, que l'on appliqué ensuite explicitement aux services et endpoints à travers l'attribut « behaviorConfiguration ». Avec WCF 4, on peut définir une configuration par défaut pour les behavior en omettant le nom dans la définition de la configuration. Si on ajoute des behaviors par défaut au fichier machine.config, ils s'appliqueront à tous les services et tous les endpoints hébergés sur la machine. Si on les ajoute au fichier app.config, ils ne prendront effet que dans le périmètre de l'application les hébergeant.

Voici un exemple :

```
<configuration>
  <system.serviceModel>
    <behaviors>
      <serviceBehaviors>
        <behavior> <!-- il n'y a pas d'attribut name -->
          <serviceMetadata httpGetEnabled="true"/>
        </behavior>
      </serviceBehaviors>
    </behaviors>
  </system.serviceModel>
</configuration>
```

Cet exemple active les métadonnées sur le service pour n'importe quel service qui n'aurait pas de behavior explicite définit dans sa configuration. Si nous ajoutons ce behavior dans le fichier app.config de l'application présentée dans ce document, et que l'on exécute l'application, on peut naviguer vers la page d'aide du web service et le WSDL du web service.

Figure 6: Naviguer vers les métadonnées du service actives par le behavior



Une autre fonctionnalité du WCF 4 est que l'on peut maintenant utiliser l'héritage entre les behavior. Si une application utilise un nom de behavior qu'un de ceux définit dans le machine.config, le behavior de l'application et celui du machine.config vont être fusionnés. Cela ajoutera donc les behaviors du machine.config à ceux de l'application.

4.4.1.5 Standard Endpoints

WCF 4 a une nouvelle fonctionnalité appelée « standard endpoints ». Il s'agit en fait d'endpoints préconfigurés disponible dans le framework WCF 4 que l'on peut utiliser. Il s'agit de endpoints standard que l'on n'est pas sensé modifier.

La Figure 7 montre les standards endpoint livrés avec WCF 4. Il s'agit des endpoints les plus souvent mis en œuvre dans les modèles de communication. Par exemple, lorsque l'on publie un endpoint MEX, on doit toujours spécifier IMetadataExchange pour le contrat de service et l'on choisit généralement HTTP. Au lieu de devoir recréer cela à chaque fois manuellement, WCF fournit un standard endpoint pour les échanges de métadonnées, appelé « mexEndpoint » qui est plus simple à utiliser.

Figure 7: Standard endpoints en WCF 4

Nom des Standard endpoint	Description
mexEndpoint	Définit un endpoint de type MEX configuré avec le contrat de service, mexHttpBinding étant le binding par défaut et une adresse vide.
dynamicEndpoint	Permet de définir un service WCF utilisant Discovery avec une application cliente. Quand on utilise cet endpoint on n'a pas besoin de spécifier d'adresse. L'adresse est obtenu par le client lorsqu'il envoi la requête au service pour obtenir un service avec un endpoint correspondant au contrat demandé, ensuite il s'y connecte automatiquement. Par défaut la requête de découverte est envoyée en multicast UDP, mais on peut spécifier un discovery binding différent et modifier aussi les critères de recherche.
discoveryEndpoint	Permet de définir un service WCF utilisant Discovery avec une application cliente. L'utilisateur doit spécifier le binding et l'adresse quand il utilise ce standard endpoint.
udpDiscoveryEndpoint	Permet de définir un service WCF utilisant Discovery avec une application cliente. Il utilise un binding multicast UDP, il dérive du DiscoveryEndpoint.
announcementEndpoint	Définit un endpoint préconfiguré pour la fonctionnalité d'annonce de discovery. l'utilisateur doit spécifier une adresse et le binding quand il utilise cela.
udpAnnouncementEndpoint	Définit un endpoint préconfiguré pour la fonctionnalité d'annonce de discovery en utilisant le binding UDP et une adresse multicast. Cet endpoint dérive d'announcementEndpoint.
workflowControlEndpoint	Defines a standard endpoint for controlling the execution of workflow instances (create, run, suspend, terminate, etc).
webHttpEndpoint	Defines a standard endpoint configured with the WebHttpBinding and the WebHttpBehavior. Use to expose REST services.
webScriptEndpoint	Defines a standard endpoint configured with the WebHttpBinding and the WebScriptEnablingBehavior. Use to expose Ajax services.

On peut utiliser n'importe lequel de ces endpoints simplement en le nommant dans notre configuration. L'élément <endpoint> a maintenant un attribut « kind » que l'on utilise pour spécifier le nom d'un endpoint par défaut. Par exemple, voici comment utiliser le endpoint MEX avec le service Greetings :

```
<configuration>
  <system.serviceModel>
    <services>
      <service name="GreetingService">
        <endpoint kind="basicHttpBinding" contract="IHello"/>
        <endpoint kind="mexEndpoint" address="mex" />
      </service>
    </services>
  </system.serviceModel>
</configuration>
```

En utilisant les standard endpoints on économise certain éléments de configuration (i.e., avec le mexEndpoint on ne spécifie ni le binding ni le contrat), il y a cependant des moments où l'on peut avoir besoin de spécifier certains éléments.

Quand on doit le faire, on utilise la section <standardEndpoints> et on surcharge la configuration du standard. On va référencer la nouvelle configuration en utilisant l'attribut « endpointConfiguration » lors de la définition du nouveau <endpoint> :

```
<configuration>
  <system.serviceModel>
    <services>
      <service name="GreetingService">
        <endpoint binding="basicHttpBinding" contract="IHello"/>
        <endpoint kind="udpDiscoveryEndpoint" endpointConfiguration="D11"/>
      </service>
    </services>
    <standardEndpoints>
      <udpDiscoveryEndpoint>
        <standardEndpoint name="D11" discoveryVersion="WSDiscovery11"/>
      </udpDiscoveryEndpoint>
    </standardEndpoints>
    <behaviors>
      <serviceBehaviors>
        <behavior>
```

```
        <serviceDiscovery/>
        <serviceMetadata httpGetEnabled="true"/>
      </behavior>
    </serviceBehaviors>
  </behaviors>
 </system.serviceModel>
</configuration>
```

Cet exemple change la version par défaut du protocole WS-Discovery pour le standard endpoint nommé « udpDiscoveryEndpoint ».

4.4.1.6 Simplifier l'hébergement IIS/ASP.NET

L'hébergement dans IIS/ASP.NET s'est fortement simplifié avec WCF 4.

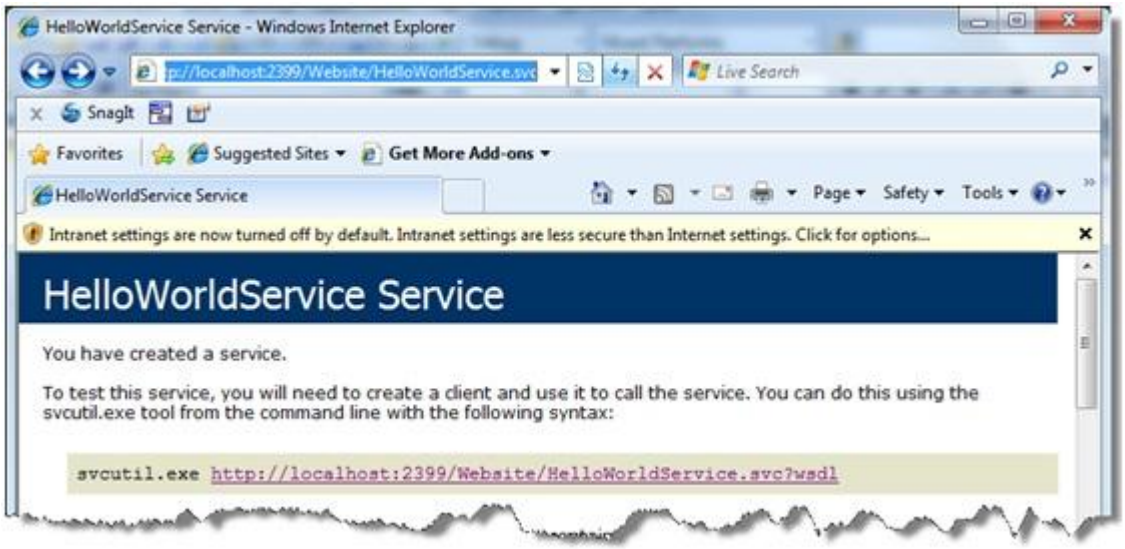
Voici une définition de service WCF :

```
<!-- HelloWorld.svc -->
<%@ ServiceHost Language="C#" Debug="true" Service="HelloWorldService
   CodeBehind="~/App_Code/HelloWorldService.cs" %>
[ServiceContract]
public class HelloWorldService
{
   [OperationContract]
   public string HelloWorld()
   {
      return "hello, world";
   }
}
```

Voici la forme la plus simple de définition d'un service WCF, car nous n'utilisons pas d'interface pour définir le contrat du service et tout est contenu dans un seul fichier (ceci est un exemple, il ne s'agit pas d'une bonne pratique ce n'est donc pas recommandé).

Avec les nouvelles fonctionnalités de WCF décrite ci-dessus, vous pouvez maintenant parcourir le service sans aucune configuration supplémentaire. La logique d'activation des services WCF va utiliser ServiceHost et le configurer avec un endpoint HTTP par défaut, cela de façon transparente pour le développeur. Et si un behavior par défaut activant les métadonnées de service a été ajouté au fichier machine.config, on doit pouvoir accéder à la page d'aide du service et son affichage de WSDL, comme dans la Figure 8.

Figure 8: HelloWorldService page d'aide



Si le service de métadonnées n'a pas été activé au niveau de la machine, on l'activera par l'intermédiaire du fichier web.config en ajoutant un behavior par défaut :

```
...
<system.serviceModel>
  <behaviors>
    <serviceBehaviors>
      <behavior> <!-- il n'y a pas d'attribut name -->
        <serviceMetadata httpGetEnabled="true"/>
      </behavior>
    </serviceBehaviors>
  </behaviors>
</system.serviceModel>
...
```

On peut bien sur changer plusieurs autres aspects du service. Si le service implémente plus d'un contrat de service l'instance de ServiceHost en résultant aura un endpoint HTTP par contrat.

Toutes ces simplifications ont pour objectif de faciliter la création de service WCF pour le développeur ASP .NET.

4.4.1.7 File-less Activation

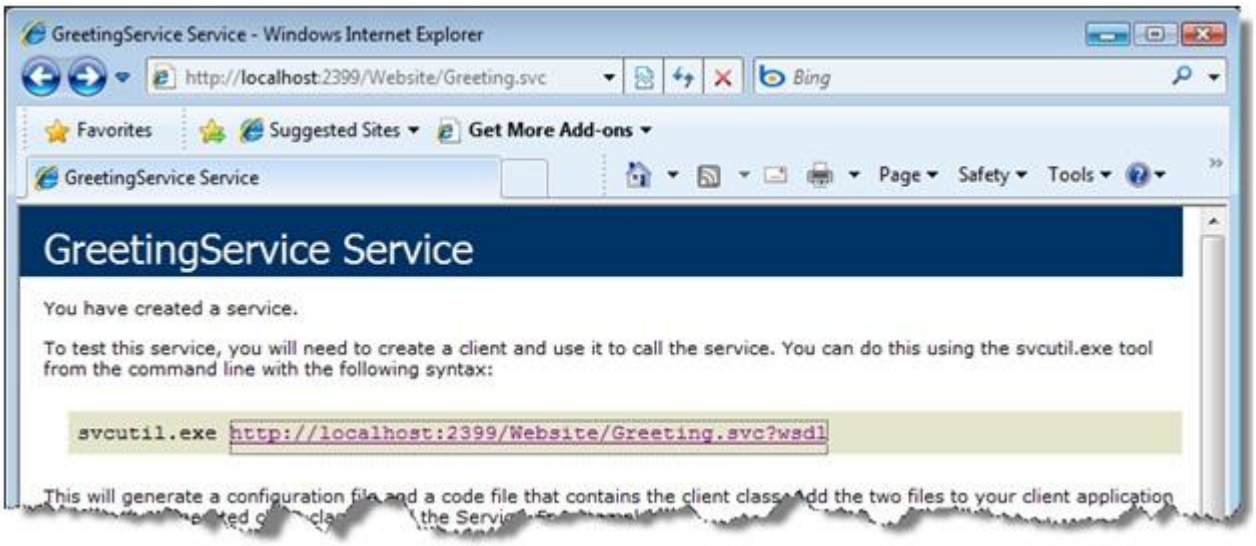
Même si les fichiers .svc facilitent l'exposition de service WCF, une approche plus simple serait de définir un endpoint virtuel directement dans le fichier web.config. On pourrait ainsi supprimer les fichiers .svc.

En WCF 4, on peut définir un endpoint virtuel qui est lié à un service définit dans le fichier web.config. Cela permet l'activation de service sans avoir besoin de maintenir un fichier .svc physique (a.k.a. "file-less activation"). L'exemple suivant montre comment configurer l'activation de cet endpoint :

```
<configuration>
  <system.serviceModel>
    <serviceHostingEnvironment>
      <serviceActivations>
        <add relativeAddress="Greeting.svc" service="GreetingService"/>
      </serviceActivations>
    </serviceHostingEnvironment>
  </system.serviceModel>
</configuration>
```

On peut maintenant invoquer le service Greeting en utilisant un chemin relatif vers « Greeting.svc » (relative à l'adresse de l'application Web). Pour démontrer cela on va créer une application web, on l'appellera « WebSite », on l'assignera à un pool d'application « ASP.NET v4.0 », et on déploiera dedans le fichier web.config décrit ci-dessus. Maintenant on peut simplement naviguer vers le site http://localhost:[port de l'application ici 2399]/WebSite/Greeting.svc sans avoir de fichier .svc physique.

Figure 9: File-less activation exemple



4.4.2 Conclusion WCF

Il ne s'agit que d'une description succincte de WCF. Il existe d'autres aspects de WCF comme les workflows, les files d'attentes, le transfert de Byte. Actuellement, compte tenu des typologies d'application existantes et des patterns de développement mis en œuvre la présentation ci dessus couvre le périmètre de l'existant.

5 DOCUMENT 2 – 1.00 - REGLES ET RECOMMANDATIONS

5.1 Introduction

Le présent document constitue les « Règles et les Recommandations d'implémentation » que devront utiliser les développeurs de la Région PACA et les prestataires de service développant pour la Région PACA, dans le cadre de leurs projets de développement sous Visual Studio .NET

Ces Règles et Recommandations visent à améliorer la lisibilité et permettent la mise en œuvre commune d'une charte accessible et utilisée par l'ensemble d'une équipe de développement.

Les règles sont basées sur le guideline de Microsoft.

5.2 Accès aux données

5.2.1 Principe

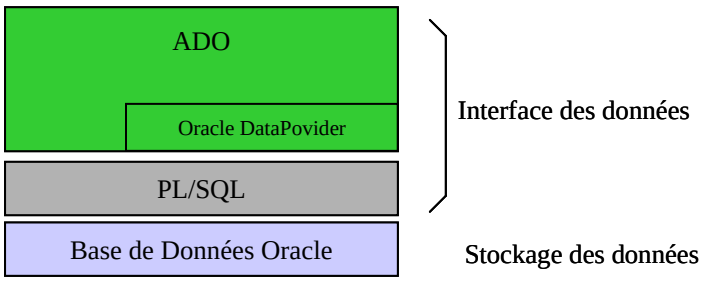
L'environnement de développement .NET propose en standard un ensemble complet de routines permettant d'accéder à des données.

Cette technologie nommée ADO sera utilisée systématiquement sur un développement.

5.2.2 Mécanismes d'accès aux données

Sauf contre indication, les mécanismes d'accès aux données mis en place sur un développement reposent sur :

un développement par couches,
des procédures stockées pour la logique d'accès aux données.



La mise en place de ces principes permet de simplifier la mise au point et la maintenance d'une solution informatique.

5.3 Implémentation des membres d'une classe

Ce paragraphe propose des préconisations d'utilisation.

5.3.1 Utilisation des propriétés

Déterminez si une propriété ou une méthode est le plus approprié pour votre besoin. Vous pouvez consulter le paragraphe, Propriétés vs Méthodes pour vous aider.

Choisissez un nom basé sur les recommandations du nommage des propriétés.

Quand vous implémenter l'accesseur set, n'écrasez l'ancienne valeur qu'à la fin de la méthode ceci pour éviter de perdre cette valeur si une exception se produit dans la méthode.

5.3.1.1 Etat d'une propriété

Les propriétés d'une classe doivent pouvoir être accédées dans n'importe quel ordre. Il ne doit pas y avoir de dépendance d'état entre plusieurs propriétés. Tout accès à une propriété doit laisser l'état de l'objet cohérent.

Si une fonctionnalité dépend de la valeur de plusieurs propriétés, cette fonctionnalité n'est active que quand l'objet est dans un état correct.

Par exemple, une TextBox a deux propriétés reliées : DataSource et DataField. DataSource spécifie le nom de la table et DataField le nom de la colonne. Quand les 2 propriétés sont spécifiées, le control peut charger automatiquement la valeur de la colonne. L'ordre d'affectation des propriétés n'a pas d'importance.

Exemple de codage de ces propriétés :

```
public class TextBox
{
   string dataSource;
   string dataField;
   bool active;
   public string DataSource
   {
      get
      {
         return dataSource;
      }
      set
      {
         if (value != dataSource)
         {
            // Update active state.
            SetActive(value != null && dataField != null);
            dataSource = value;
         }
      }
   }

   public string DataField
   {
      get
      {
         return dataField;
      }
      set
      {
         if (value != dataField)
         {
            // Update active state.
            SetActive(dataSource != null && dataField != null);
            dataField = value;
         }
      }
   }
   void SetActive(Boolean value)
   {
      if (value != active)
      {
         if (value)
         {
            Activate();
            Text = dataBase.Value(dataField);
         }
         else
         {
            Deactivate();
            Text = "";
         }
         // Set active only if successful.
         active = value;
      }
   }
   void Activate()
   {
      // Open database.
   }
```

```
   void Deactivate()
   {
      // Close database.
   }
}
```

Dans l'exemple précèdent, l'expression suivante détermine si l'objet est dans un état correct et si la liaison de données peut-être activée.

```
value != null && dataField != null
```

5.3.1.2 Propriétés vs Méthodes

En général, les méthodes représentent des actions et les propriétés des données. Utilisez les règles suivantes pour vous aider à choisir entre ces deux options.

Utilisez une propriété quand vous avez a faire à une donnée logique.

```
public string Name
{
   get
   {
      return name;
   }
   set
   {
      name = value;
}
```

Utilisez une méthode quand :

- L'opération est une conversion, tel que Object.ToString.
- L'opération est très consommatrice de ressources et vous souhaitez informé l'utilisateur qu'il est nécessaire de stocker l'information plutôt que de la récupérer à chaque fois.
- Obtenir la valeur engendre des effets de bord (Modification de l'état de l'objet…)
- Appeler cette fonctionnalité plusieurs fois ne génère pas obligatoirement le même résultat.
- L'ordre est important. Rappelez vous que l'ordre d'appel des propriétés d'un objet ne doit pas avoir d'importance.
- La fonction est statique mais retourne une valeur qui peut changer.
- La fonction renvoie un tableau. Les propriétés renvoyant un tableau peuvent être source d'erreurs.

Habituellement, il est nécessaire de renvoyer une copie du tableau de façon à ce que l'utilisateur ne puisse pas en modifier le contenu interne. Ceci, avec le fait que l'utilisateur peut légitimement croire que cette propriété est indexée, génère du code non performant. Dans l'exemple suivant, chaque appel de la propriété Methods crée une copie du tableau. Le résultat génère 2n + 1 copies du tableau à chaque boucle.

```
Type type = // Get a type.
for (int i = 0; i < type.Methods.Length; i++)
{
   if (type.Methods[i].Name.Equals ("text"))
   {
      // Perform some operation.
   }
}
```

L'exemple suivant illustre l'utilisation correcte des propriétés et des méthodes.

```
class Connection
{
// Les 3 membres suivants doivent être des propriétés
// car ils peuvent être appelés dans n'importe quel ordre.
string DNSName {get{};set{};}
string UserName {get{};set{};}
string Password {get{};set{};}

// Le membre suivant doit être une méthode car l'ordre
// d'exécution est important.
// Cette méthode ne peut pas être appelée tant que les propriétés
// n'ont pas été affectées.
bool Execute ();
}
```

5.3.1.3 Propriétés lecture seule et écriture seule

Vous devez utiliser des propriétés en lecture seule {get} quand l'utilisateur ne doit pas changer la valeur.

N'utilisez jamais de propriétés en écriture seule {set}

5.3.1.4 Utilisation des propriétés indexées

Note : Une propriété indexée est aussi appelée un indexeur.

Utilisez les règles suivantes pour les propriétés indexées :

- Utilisez une propriété indexée quand la donnée est un tableau.
- N'utilisez que des entiers ou des chaînes comme index. Si vous devez utiliser d'autres types de valeur, utilisez une méthode.
- N'utilisez que des indexeurs à une dimension sinon utilisez une méthode.
- N'utilisez qu'une propriété indexée par classe et définissez la comme propriété indexée par défaut.
- N'utilisez pas de propriété indexée qui ne sont pas par définit par comme par défaut (C# ne le permet pas)
- Nommez la propriété indexée Item. Utilisez cette règle, a moins qu'il existe un nom beaucoup plus adapté tel que la propriété Chars de la classe String. En C#, les indexeurs sont toujours nommés Item.

5.3.2 Utilisation des événements

Utilisez void comme valeur de retour pour un handler d'événement. Le nom du handler doit être suffixé par Handler.

```
public delegate void MouseEventHandler(object sender, MouseEventArgs e);
```

Utilisez une classe de données fortement typée comme valeur de l'événement, tel que les coordonnées de la souris. Une classe de donnée d'événement doit hériter de System.EventArgs tel que montré dans l'exemple suivant :

```
public class MouseEvent: EventArgs {}
```

Utilisez une méthode protégée (protected) pour la méthode qui va déclencher l'événement. Le but de cette méthode est de fournir une possibilité à une classe dériver d'intercepter le moment ou l'événement va être déclenché.

Le nom de cette méthode prend la forme de OnEventName où EventName est le nom de l'événement qui va être envoyé.

Voici un exemple d'implémentation :

```
public delegate void ButtonClickHandler(object sender, ClickEventArgs e) ;
public class ClickEventArgs : System.EventArgs { /* votre implémentation */}
public class Button
{
   ButtonClickHandler onClickHandler;

   protected virtual void OnClick(ClickEventArgs e)
   {
      // Call the delegate if non-null.
      if (onClickHandler != null)
            onClickHandler(this, e);
   }
}
```

La classe dérivée peut choisir de ne pas appeler l'implémentation de la classe de base et ne pas générer l'événement. Pour cette raison, n'incluez jamais de traitement particulier dans la méthode de la classe de base qui pourrait ne pas être exécuté.

Vous devez prendre en compte qu'un handler d'événement peut contenir du code. Dans tous les cas, l'objet doit rester dans un état cohérent après que l'évènement est été envoyé. Utilisez des blocs try/finally au niveau du code qui génère l'événement. Puisque le développeur qui intercepte l'événement peut appeler d'autres fonctions de l'objet lorsqu'il traite celui-ci, vous ne pouvez pas faire des estimations sur l'état d'un objet entre le moment où l'événement sa été émis et la suite du code. Par exemple, ne stockez pas la valeur d'une propriété avant l'émission de l'événement pour la réutiliser après, relisez là dans tous les cas.

```
public class Button
{
   protected void DoClick()
   {
      int a = this.MaPropriete;
      try
      {
         // Call event handler.
         OnClick();
      }
      finally
      {
      }
      // N'utilisez pas la valeur de a, rechargez la avant de l'utiliser
      a = this.MaPropriete;
      // Utilisation de a
   }
}
```

Utilisez ou étendez la classe System.ComponentModel.CancelEventArgs pour permettre à un développeur de contrôler l'événement que vous émettez. Cette classe permet au développeur qu'il souhaite canceler l'opération à votre charge de tester la valeur retournée. Par exemple, le contrôle TreeView émet un événement quand l'utilisateur souhaite modifier un label. Le code suivant montre comment vous pouvez empêcher cette opération.

```
public class Form1: Form
{
   TreeView treeView1 = new TreeView();

   void treeView1_BeforeLabelEdit(object source,
      NodeLabelEditEventArgs e)
   {
      e.CancelEdit = true;
   }
}
```

Notez que dans ce cas, aucune erreur n'est générée. Le label est considéré comme en lecture seule.

Cette méthode n'est pas appropriée dans le cas où le développeur souhaite canceller l'opération et retourner une erreur. Dans ce cas, vous devez émettre une erreur à l'intérieur de votre handler.

5.3.3 Utilisation des méthodes

Choisissez un nom en respectant les règles de nommage des méthodes.

Par défaut, les méthodes ne sont pas virtuelles. Ne rendez une méthode virtuelle que si cela est réellement nécessaire.

5.3.3.1 Surcharge des méthodes

La surcharge d'une méthode survient quand une classe contient plusieurs méthodes de même nom ayant des signatures différentes (paramètres).

Cette section contient quelques règles pour utiliser la surcharge de méthode.

Utilisez la surcharge de méthode pour fournir différentes méthodes qui effectue sémantiquement la même chose. La surcharge permet de simuler la notion de paramètre par défaut.

Utilisez les paramètres par défaut correctement. Dans un groupe de méthodes surchargés, la méthode la plus complexe doit utiliser des noms de paramètres qui indique le changement par rapport aux valeurs par défaut des méthodes plus simple. Par exemple, dans le code suivant, la 1ére méthode présume que la recherche ne dépend pas de la casse. La seconde méthode utilise le nom ignoreCase plutôt que caseSensitive pour indiquer quelle est la différence avec la valeur par défaut.

```
// Method #1: ignoreCase = false.
MethodInfo Type.GetMethod(String name);
// Method #2: Indicates how the default behavior of method #1 is being // changed.
 MethodInfo Type.GetMethod (String name, Boolean ignoreCase);
```

Respectez un ordre et un nommage des paramètres cohérent entre les différentes méthodes surchargées. Il est courant de fournir un ensemble de méthodes surchargées avec un nombre croissant de paramètres. Dans l'exemple suivant, la définition des paramètres conserve un ordre et des noms cohérents.

```
public class SampleClass
{
   readonly string defaultForA = "default value for a";
   readonly int defaultForB = "42";
   readonly double defaultForC = "68.90";

   public void Execute()
   {
      Execute(defaultForA, defaultForB, defaultForC);
   }

   public void Execute (string a)
   {
      Execute(a, defaultForB, defaultForC);
   }

   public void Execute (string a, int b)
   {
      Execute (a, b, defaultForC);
   }

   public void Execute (string a, int b, double c)
   {
      Console.WriteLine(a);
      Console.WriteLine(b);
      Console.WriteLine(c);
      Console.WriteLine();
   }
}
```

Si vous devez rendre les méthodes virtuelles, ne définissez que la méthode la plus complexe comme virtuelle.

```
public int IndexOf(string s)
{
   return IndexOf (s, 0);
}
public int IndexOf(string s, int startIndex)
{
   return IndexOf(s, startIndex, myString.Length - startIndex );
}
public virtual int IndexOf(string s, int startIndex, int count)
{
   return myString.IndexOf(s, startIndex, count);
}
```

5.3.3.2 Méthode avec un nombre variable de paramètres

Vous pouvez définir une méthode avec un nombre variable de paramètres en utilisant le mot-clé params. Par exemple, utilisez le code suivant plutôt que d'utiliser un ensemble de méthodes surchargées.

```
void Format(string formatString, params object [] args)
```

Pour des raisons de performance, vous pouvez définir un petit sous-ensemble de méthodes surchargées. Vous devez faire cela uniquement si chaque méthode implémente un code spécifique permettant d'optimiser le code et ne se contente pas d'appeler la méthode la plus complexe.

```
void Format(string formatString, object arg1)
void Format(string formatString, object arg1, object arg2)
void Format(string formatString, params object [] args)
```

5.3.4 Utilisation des constructeurs

Utilisez les règles suivantes pour la définition des constructeurs.

Fournissez un constructeur privé par défaut s'il y a uniquement des méthodes et propriétés statiques dans cette classe. Un constructeur privé empêchera la classe d'être instanciée.

```
public sealed class Environment
{
   // Private constructor prevents the class from being created.
   private Environment()
   {
      // Code for the constructor goes here.
   }
}
```

Réduisez au maximum les traitements contenus dans le constructeur. Un constructeur ne devrait pas faire plus que de stocker la valeur des paramètres.

Fournissez un constructeur pour chaque classe. Si vous ne définissez pas un constructeur, C# ajoutera implicitement un constructeur par défaut. Si la classe est abstraite, ce sera un constructeur protected.

Fournissez un constructeur protected par défaut qui pourra être utilisé par les classes dérivées.

Utilisez des paramètres dans les constructeurs pour simplifier l'initialisation des propriétés

De la même manière que les méthodes surchargées, utilisées un ordre et des noms cohérents pour les paramètres.

5.3.5 Utilisation des champs

Ne définissez jamais des champs public ou protected. Si des champs doivent être accédés en dehors de votre classe, définissez des propriétés. Ceci permet une évolution plus facile de votre classe en conservant une compatibilité binaire au niveau des interfaces.

Exposez un champ à vos classes dérivées en utilisant une propriété protected. Une classe dérivée ne doit jamais accéder directement au champ de sa classe de base.

```
public class Control: Component
{
   private int handle;
   protected int Handle
   {
      get
      {
         return handle;
      }
   }
}
```

Utilisez le mot-clé const pour déclarer des constantes. Le compilateur utilisera directement la valeur de la constance plutôt qu'une référence au champ.

Utilisez des champs en static readonly pour des instances d'objets prédéfinis. Utilisez le Pascal Casing dans ce cas car les champs sont public. N'oubliez pas que vous pouvez utiliser un constructeur statique pour initialiser de tels champs.

```
public struct Color
{
   public static readonly Color Red = new Color(0x0000FF);
   public static readonly Color Green = new Color(0x00FF00);
   public static readonly Color Blue = new Color(0xFF0000);
   public static readonly Color Black = new Color(0x000000);
   public static readonly Color White = new Color(0xFFFFFF);
}
```

N'utilisez pas les majuscules pour un nommé un champ.

N'utilisez pas la notation hongroise pour les champs, un nom correct décrit la sémantique pas le type.

Par convention, préfixez les noms des champs par un _ pour éviter que la distinction entre le champ et la propriété ne se fasse que sur la casse.

5.3.6 Utilisation des paramètres

Vérifiez systématiquement la validité des arguments pour les méthodes public ou protected et pour les propriétés d'écriture (set). Emettez des exceptions explicites pour le développeur. Utilisez les exceptions prédéfinis du framework (ArgumentNullException ou ArgumentOutOfRangeException) ou créez en des nouvelles à partir de la classe System.ArgumentException.

Exemple de validation de paramètre :

```
class SampleClass
{
   public int Count
   {
      get
      {
         return count;
      }

      set
      {
            // Check for valid parameter.
            if (count < 0 || count >= MaxValue)
               throw new ArgumentOutOfRangeException(
                  Sys.GetString("InvalidArgument","value",count.ToString()));
      }
   }

   public void Select(int start, int end)
   {
      // Check for valid parameter.
      if (start < 0)
         throw new ArgumentException(
             Sys.GetString("InvalidArgument","start",start.ToString()));
      // Check for valid parameter.
      if (end < start)
         throw new ArgumentException(
               Sys.GetString("InvalidArgument","end",end.ToString()));
   }
}
```

Faites bien la distinction entre le passage de paramètre par valeur ou par référence. Passer un paramètre par valeur copie la valeur qui est passée et n'a pas d'effet sur la valeur initiale. L'exemple suivant passe un paramètre par valeur.

```
public void Add(object value){}
```

Passer un paramètre par référence passé un lien sur la valeur. Toute modification sur le paramètre a un impact sur la valeur initiale. L'exemple suivant montre un passage par référence.

```
public static int Exchange(ref int location, int value){}
```

Passer un paramètre en sortie (out) correspond à un passage par référence sans que la valeur initiale soit initialisé. Si vous omettez ce mot-clé, le compilateur vous empêchera de passer une référence sur une valeur qui n'aura pas été initialisée.

```
[DllImport("Kernel32.dll"]
public static extern bool QueryPerformanceCounter(out long value)
```

5.3.7 Utilisation des classes de bases

Une classe peut-être abstract ou sealed. Une classe abstract nécessite d'implémenter une classe dérivée. Une classe sealed ne lui permet pas d'être dérivée.

5.3.7.1 Classes de base vs Interfaces

Un type interface permet de spécifier un protocole potentiellement supporté par plusieurs classes. Utilisez des classes de bases plutôt que des interfaces dans la mesure du possible. Dans une perspective de versionnage, les classes sont plus flexibles que les interfaces. Avec une classe, vous pouvez faire évoluer votre classe en ajoutant une méthode. Tant que la classe n'est pas abstraite (abstract), toutes les classes dérivées peuvent continuer à fonctionner.

Parce que les interfaces ne supportent pas l'héritage de l'implémentation, ce principe ne peut pas s'appliquer aux interfaces. Ajouter une méthode à une interface correspond à l'ajout d'une méthode abstraite sur une classe. Toutes les classes qui implémentent cette interface ne fonctionneront plus.

Les interfaces sont appropriées dans les cas suivants :

- Plusieurs classes sans relation doivent implémenter un protocole.
- Ces classes ont déjà une classe de base.
- L'agrégation n'est pas pratique ou approprié.

Dans tous les autres cas, l'héritage est plus adapté.

5.3.7.2 Utilisation des énumérations

Les règles suivantes s'appliquent lors de l'utilisation des énumérations :

N'utilisez pas le suffixe Enum sur les types énumérations.

Utilisez une énumération (enum) pour des paramètres, propriétés ou valeur de retour fortement typé.

Utilisez un enum plutôt que des constantes statiques.

N'utilisez pas d'enum pour un ensemble de valeurs non fixes (comme un numéro de version par exemple).

Utilisez l'attribut System.FlagsAttribute sur un enum uniquement si il sera nécessaire d'effectuer des ou (or) binaires sur les valeurs. Dans ce cas, utilisez des puissances de 2 pour pouvoir mixer plusieurs valeurs facilement comme sur l'exemple suivant :

```
[Flags()]
public enum WatcherChangeTypes
{
   Created = 1,
   Deleted = 2,
   Changed = 4,
   Renamed = 8,
   All = Created | Deleted | Changed | Renamed
};
```

Note : Le ou binaire peut s'utiliser directement lors de la déclaration de l'énumération.

Ne partez pas du principe qu'une valeur d'énumération est comprise dans une plage de valeurs. Il est possible de caster n'importe qu'elle valeur entière en un enum même si cette valeur ne correspond à rien. Pour éviter ce type d'erreur, vous devez vous assurer que la valeur est correcte. L'exemple suivant vous montre comment faire :

```
public void SetColor (Color color)
{
   if (!Enum.IsDefined (typeof(Color), color)
      throw new ArgumentOutOfRangeException();
}
```

5.3.7.3 Utilisation des délégués

Les délégués s'utilisent principalement pour les 2 utilisations suivantes :

- Notifications d'événements. Utilisez les délégués pour définir les événements pour plus d'informations sur la déclaration des événements, regardez le paragraphe sur l'utilisation des événements.

- Fonction de rappel (callback). Les fonctions de retour sont passées comme argument de méthode pour permettre au développeur d'appeler directement une fonction de multiples fois. Passer une méthode de comparaison à un algorithme de tri est une utilisation classique des méthodes de rappel. Nommez toujours vos méthodes de rappel avec le suffixe Callback.

5.3.7.4 Utilisation des types imbriqués

Un type imbriqué est un type définit à l'intérieur d'un autre. Les types imbriqués sont intéressants pour encapsuler les détails d'implémentation d'un type, tel qu'un énumérateur sur une collection, parce qu'ils peuvent avoir accès aux informations privées.

Les types imbriqués public doivent être utilisés très rarement.

N'utilisez pas les types imbriqués dans les cas suivants :

- Le type doit être instancié par du code client. Si un type a un constructeur public, c'est qu'il ne doit sûrement pas être imbriqué. Un type imbriqué ne doit pas être utilisé en dehors de son type « propriétaire » sans qu'il y soit en relation avec celui-ci.
- Des références existent sur le type dans le code client.

5.3.7.5 Utilisation des événements

Toute portion de code susceptible d'émettre un événement doit mettre à disposition une méthode permettant de tester son exécution sans émettre d'événement. Par exemple, pour éviter un FileNotFoundException vous devez appeler File.Exists.

Cette règle n'est pas toujours possible, mais le but est que dans le cadre d'une exécution normale aucune exception ne doit être générée.

Utilisez le suffixe Exception pour nommer vos classes exceptions.

Essayez d'utiliser au maximum les exceptions standards du Framework.

Si vous créez de nouvelles exceptions, celles-ci ne doivent pas être propagées entre couches. Si vous devez propager une exception, encapsulez-la en utilisant la propriété InnerException. Pour le découpage en couche d'une application, voyez la section correspondante. L'exception que vous avez créé doit faire partie du namespace de la couche concernée.

N'incluez des informations supplémentaires dans votre exception que si cela a un intérêt pour un traitement particulier. En règle générale, il n'est pas nécessaire d'inclure des informations supplémentaires.

N'intégrez pas des informations sensibles dans le texte de vos messages. Des informations tel qu'un chemin d'accès sont considérées comme une info sensible.

N'utilisez pas des exceptions pour des erreurs normales ou attendues. Comme son nom l'indique, une exception ne doit arriver que pour prévenir que quelque chose d'exceptionnel vient d'arriver. Par exemple, un client qui n'existe pas ne doit pas être traité avec un exception.

Vous devez utilisez la valeur null pour gérer la plupart des cas d'anomalies. Par exemple, un File.Open doit renvoyer null si le fichier n'existe pas mais génère une exception si le fichier est verrouillé. N'utilisez jamais d'autre valeur de retour, si vous devez le faire, utilisez l'exception.

Concevez les classes de sorte que dans le cadre d'une utilisation normale aucune exception ne doit être générée.

Assurez vous de conserver un état cohérent lorsque vous émettez une exception. Par exemple, une Hashtable.Insert génère une exception, on est en droit de penser que l'insertion n'a pas été faite.

5.3.7.6 Exceptions standards

Voici une liste des exceptions fournit par le Framework.

+--System.Object
+--System.Exception
+--System.SystemException
+--System.ArgumentException
+--System.ArgumentNullException
+--System.ArgumentOutOfRangeException
+--System.DuplicateWaitObjectException
+--System.ArithmeticException
+--System.DivideByZeroException
+--System.OverflowException
+--System.NotFiniteNumberException
+--System.ArrayTypeMismatchException
+--System.ExecutionEngineException
+--System.FormatException
+--System.IndexOutOfRangeException
+--System.InvalidCastException
+--System.InvalidOperationException
+--System.ObjectDisposedException
+--System.InvalidProgramException
+--System.IO.IOException
+--System.IO.DirectoryNotFoundException
+--System.IO.EndOfStreamException
+--System.IO.FileLoadException
+--System.IO.FileNotFoundException
+--System.IO.PathTooLongException
+--System.NotImplementedException
+--System.NotSupportedException
+--System.NullReferenceException
+--System.OutOfMemoryException
+--System.RankException
+--System.Security.SecurityException
+--System.Security.VerificationException
+--System.StackOverflowException
+--System.Threading.SynchronizationLockException
+--System.Threading.ThreadAbortException
+--System.Threading.ThreadStateException
+--System.TypeInitializationException
+--System.UnauthorizedAccessException

5.3.7.7 Préconisation sur la remontée des Exceptions

Ne plus utiliser des levées d'exceptions comme des sorties de méthode de BAL.

Exemple : dans plusieurs méthodes BAL, on retrouve le code suivant pour le retour de procédures dans un datatable tbl :

```
[signature méthode]
1.[initialisation DAL]
        2.[declaration des parameters]
                If(tbl.Rows.count == 0 )
                        Throw new BALException(…) ;
3.[instructions du cas else mais non contenues dans le branchement else]
```

Le code ci-dessus doit être remplacé par :

```
1.[signature méthode]
2.[initialisation DAL]
3.[declaration des parameters]
        If(tbl.Rows.count == 0 )  {
                //traitement du IF
        }
        Else  {
                //traitement du else
        }
```

Le BAL fournit un résultat de sortie contenant des données pertinentes ou « neutres ».

Exemple : si la signature de la méthode demande un type DSSuiviWeb ou DataTable en sortie, la méthode Bal doit fournir un type DSSuiviWeb ou DataTable en sortie (dans le return).

La prise en compte de la vacuité ou non de ces types doit être délégué au client du BAL concerné (comme tout autre test sur la sortie retournée du BAL).

De même pour les méthodes BAL retournant un type string / int, une méthode BAL doit retourner impérativement un type string ou int.

Plusieurs méthodes du BAL utilisent à tord la méthode A, c'est a dire que leur sortie est soit une exception soit une donnée pertinente

```
If( tbl.Rows.Count == 0)
        Throw BALException(…) ;
Fin du if

[…]
        Return tbl.Rows[0][0].ToString()  ;
```

Il faudrait dorénavant utiliser ce type de branchement If :

```
If(tbl.Rows.Count ==0)
        Return Constantes.ValeurNeutre;
```

```
Else
        Return tbl.Rows[0][0].ToString()  ;
```

Constantes.ValeurNeutre ici est de type string et modélise un résultat de retour vide (c'est à dire la procédure stockée s'est bien déroulée mais ramène un résultat vide).

Par la suite, c'est au client de prendre en compte cette valeur et d'effectuer les traitements en accord avec la valeur retournée du string.

5.3.7.8 Utilisation des tableaux

Il n'est pas toujours facile de décider d'utiliser un tableau ou une collection. En général, vous devez utilisez une collection si vous devez effectuer des mises telles que Add, Remove ou Insert.

Ne retournez jamais l'instance interne d'un tableau car ceci permet au code appelant de modifier son contenu. L'exemple suivant montre comment le tableau badChars peut être changé en utilisant la propriété Path même si cette propriété n'implémente pas un accès en écriture.

```
public class ExampleClass
{
   public sealed class Path
   {
      private Path(){}
      private static char[] badChars = {'\"', '<', '>'};
      public static char[] GetInvalidPathChars()
      {
         return badChars;
      }
   }
   public static void Main()
   {
      // Le code suivant modifie le contenu de badChars
      Path.GetInvalidPathChars()[0] = 'A';
   }
}
```

Pour corriger ce problème, vous devez renvoyer une copie du tableau en utilisant la méthode clone.

```
public static char[] GetInvalidPathChars()
{
   return (char[])badChars.Clone();
}
```

Note : L'utilisation de readonly sur le tableau ne résoudrait pas le problème. Car il ne s'appliquerait qu'au tableau (qui ne pourrait pas évoluer) mais à son contenu (qui pourrait être modifié).

5.3.7.9 Retourner des tableaux vides

Les propriétés renvoyant des tableaux ou des chaînes ne doivent jamais renvoyer un null. La valeur nulle peut être difficile à traiter pour un développeur. Par exemple, un utilisateur peut considérer que le code suivant marchera dans tous les cas :

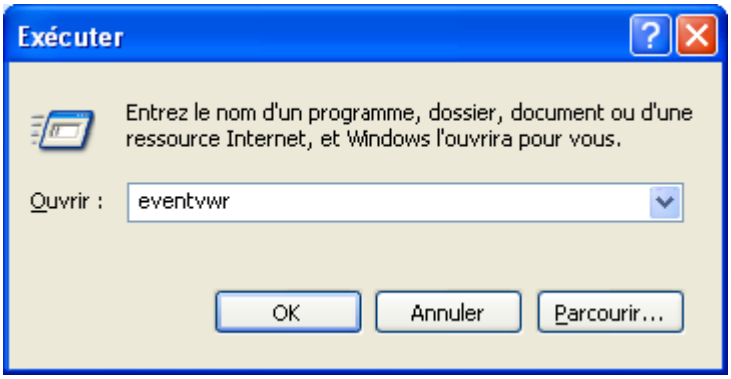
```
public void DoSomething()
{
   string s = SomeOtherFunc();
   if (s.Length > 0)
   {
      // Do something else.
   }
}
```

La règle veut qu'un null, une chaîne vide ou un tableau vide soit traité de la même manière. Renvoyez toujours un tableau vide plutôt qu'un null.

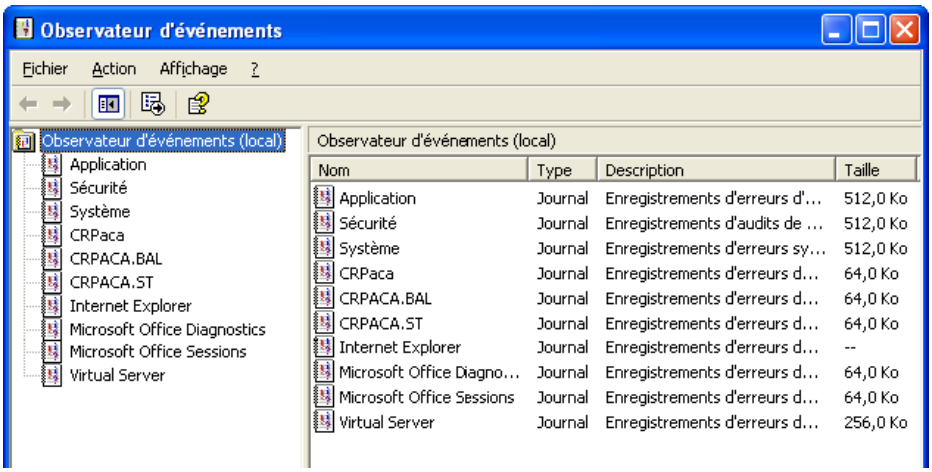
5.3.7.10 Journaux d'événements

5.3.7.10.1 Accès à un journal d'évènement

Dans une invite « Exécuter », taper eventvwr.



5.3.7.10.2 Configuration d'un journal d'évènement



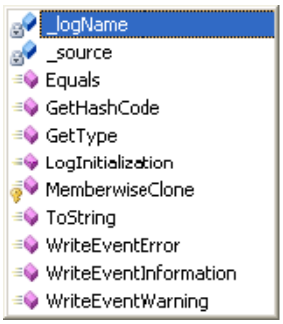
Le journal d'évènement a un nom d'affichage (displayname), un nom de fichier (logname), un emplacement fichier, une taille exprimée en kilobytes (Kb) et un comportement en cas de débordement (OverflowAction).

Ces propriétés peuvent être soit modifiées dans le code applicatif en faisant appel au namespace System.Diagnostics ; soit modifiées dans le panneau propriété d'un journal dans l'observateur d'évènements.

5.3.7.10.3 Gestion du journal d'évènements dans l'applicatif

Au niveau applicatif, par exemple, la solution SocleTechnique.Util, se trouve une classe DALEvents (fichier DALEvents.cs) qui permet de créer un journal et des évènements pour la couche applicative DAL.

Cette classe expose les membres suivants :



Propriétés

_logName : nom du journal dans lequel seront inscrits les évènements

_source : source de l'évènement inscrit dans le journal, la source doit être associée dans le code au journal.

Méthodes

LogInitialization() : 3 surcharges, voir descriptif dans l'IDE.

WriteEventError() : permet d'inscrire dans le journal un évènement Erreur.

WriteEventInformation() : permet d'inscrire dans le journal un évènement Information.

WriteEventWarning() : permet d'inscrire dans le journal un évènement Warning.

Au niveau applicatif, la création d'un journal s'effectue lors du démarrage du service : si le journal n'existe pas, il est crée avec 1 ou 2 sources constantes définies dans les fichiers de configuration de l'application.

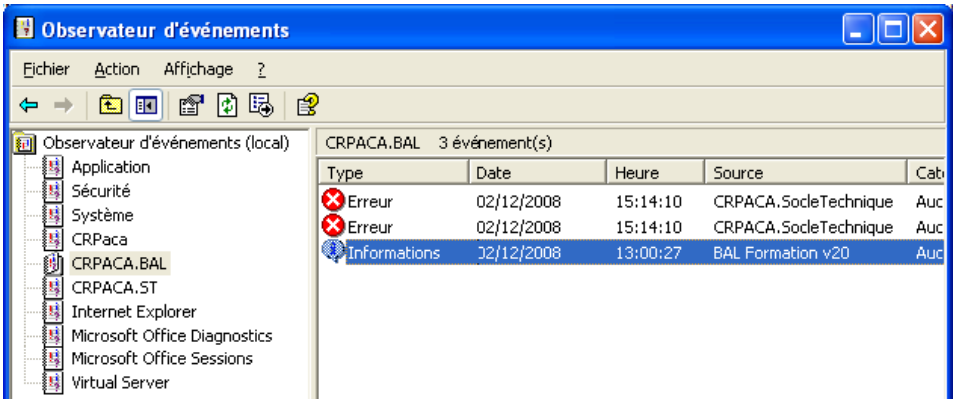
Ainsi, pour la couche SocleTechnique, le journal a un nom et une source à sa création :

Le nom du journal : CRPACA.ST

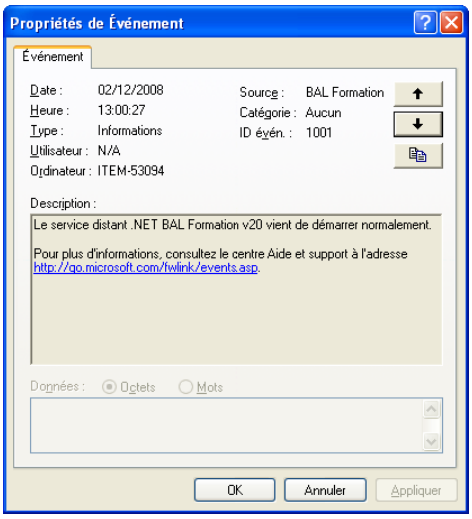
La source du journal à la création : Socle Technique

5.3.7.10.4 Gestion des évènements dans l'applicatif

L'observateur d'évènements de Windows permet d'afficher les évènements contenus dans un journal:



De même qu'un journal d'évènements, les évènements ont des propriétés visibles dans un panneau de propriétés :



De manière analogue aux journaux d'évènements, les propriétés des évènements peuvent être éditées au niveau applicatif.

Date : date d'inscription dans le journal.

Heure : heure d'inscription dans le journal.

Type : Type d'évènement (information/erreur/avertissement).

Utilisateur : utilisateur applicatif lors de l'inscription de l'évènement.

Ordinateur : nom réseau de la machine sur laquelle est survenu l'évènement.

Source : application ou composant applicatif où s'est produit l'évènement.

Catégorie : information facultative (ne sera pas déterminante pour les applications CRPACA), peut être configurée ultérieurement.

ID Evènement : information facultative (ne sera pas déterminante pour les applications CRPACA), peut être configurée ultérieurement.

La création d'un évènement se fait par l'appel de l'une des surcharges du constructeur de la classe DALEvents : pour le cas du SocleTechnique, le code ci-dessous instancie un évènement vide dans le journal avec la source « DAL ».

```
DALEvents DalEvent = new DALEvents("DAL");
```

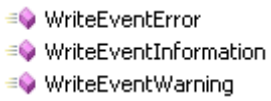
On peut aussi préciser une autre source pour l'évènement instancié :

```
DALEvents DalEvent = new DALEvents("System.Text");
```

A noter que le nom du journal est constant et ne peut être changé.

Il reste alors à préciser le contenu et inscrire cet évènement dans le journal.

Ces opérations sont effectuées avec la classe DALEvents (DALEvents.cs). Le développeur peut créer 3 types d'évènement différents (information/erreur/avertissement) et préciser le contenu de ces évènements avec l'utilisation des méthodes :



Pour le cas du SocleTechnique, l'inscription d'un évènement dans le journal se traduit en code :

```
DalEvent.WriteEventError("Contenu informatif de l'évènement");
```

Une fois ce code traité, un évènement est crée dans le journal.

5.3.7.10.5 Intégration dans l'applicatif métier

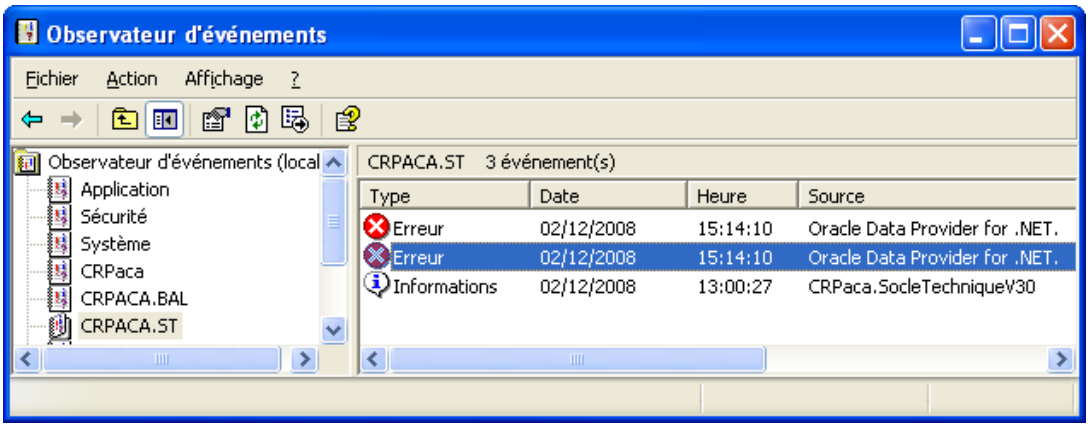
De manière générale, la création d'un évènement dans le journal BAL ou DAL intervient lorsqu'une exception est catchée au niveau du code dans n'importe quelle couche.

Avec l'architecture actuelle, Client (Riche ou Extranet) <-> BAL <-> DAL, une exception levée dans une couche supérieure est propagée aux couches inférieures de l'infrastructure.

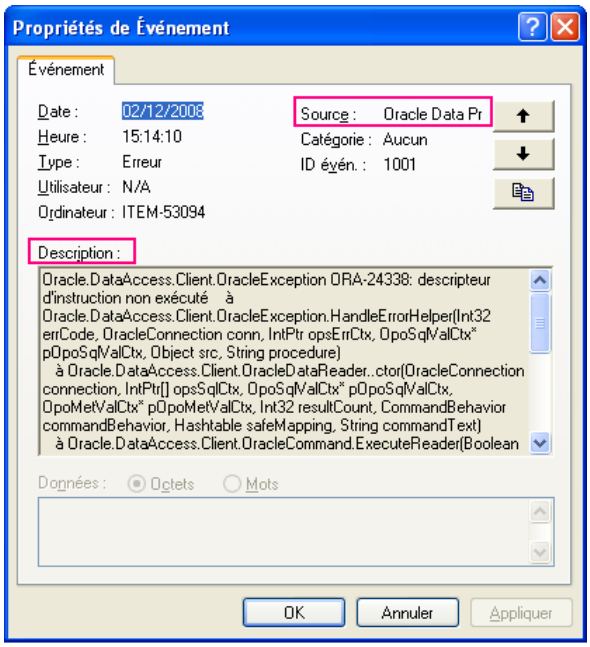
La création d'évènement suit donc cette propagation mais précisera la source applicative de l'exception encourue pour ne pas faire d'amalgame entre levée d'exception et inscription d'évènements.

Création de l'évènement dans le journal DAL – Code métier DAL

```
catch (Exception ex2)
{
string strExceptionDAL2 = "Source applicative : " +
ex2.Source.ToString()+"." + " \nType Exception : " + ex2.ToString();
DALEvents DalEvent = new DALEvents("DAL");
DalEvent.WriteEventError(strExceptionDAL2);
throw new DALException(ex2.Message, -22)
}
```



Cet évènement relate toutes les informations nécessaires au traitement de l'exception :



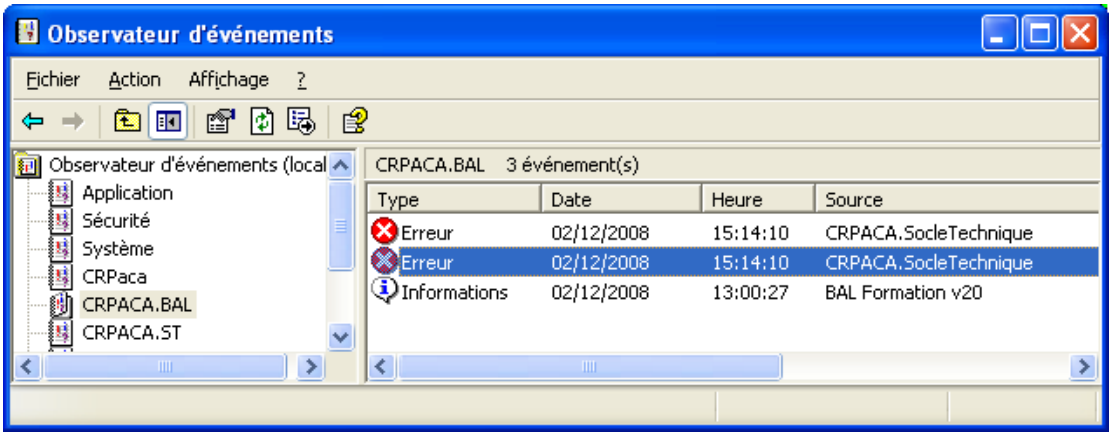
L'applicatif, ayant notifié l'évènement de cette exception dans le journal DAL, va alors propager cette exception à la couche inférieure (BAL) avec un « throw » en code :

```
throw new DALException(ex2.Message, -22);
```

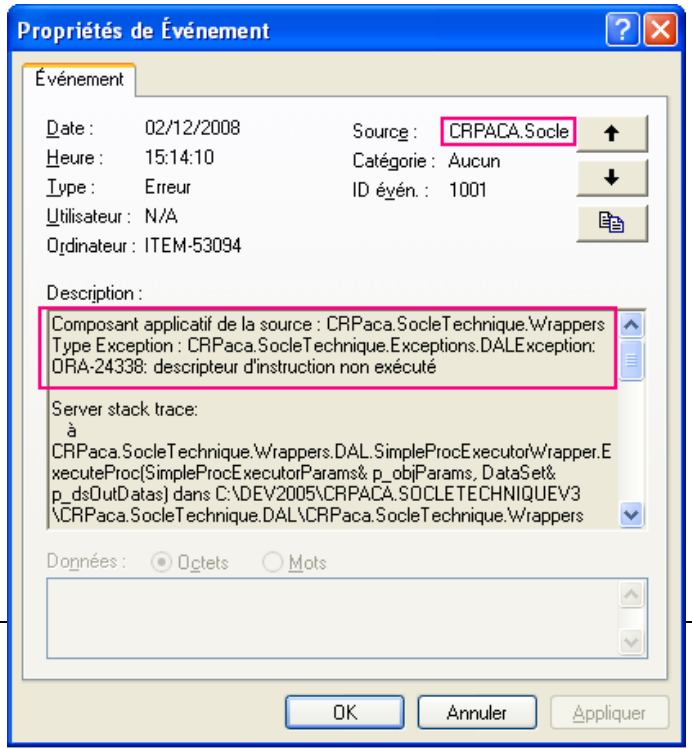
De même, l'applicatif BAL va gérer cette exception en créant un évènement de source CRPACA.SocleTechnique.Wrappers dans le journal BAL :

Création de l'évènement dans le journal BAL – Code métier BAL

```
catch (DALException dalE)
{
string strExceptionMessage = "Composant applicatif de la source : " + dalE.Source.ToString() + " \nType Exception : " + dalE.ToString();
BALEvents BALEvent = new BALEvents("DAL");
BALEvent.WriteEventError(strExceptionMessage);
throw new BALException(dalE.Message, dalE.ErrorCode);
}
```



De même, cet évènement relate de toutes les informations de l'exception relevée :



6 2 – 2.00 - REGLES ET RECOMMANDATIONS

6.1 Introduction

Le présent document constitue les « Règles et les Recommandations d'implémentation » que devront utiliser les développeurs de la Région PACA et les prestataires de service développant pour la Région PACA, dans le cadre de leurs projets de développement sous Visual Studio .NET

Ces Règles et Recommandations visent à améliorer la lisibilité et permettent la mise en œuvre commune d'une charte accessible et utilisée par l'ensemble d'une équipe de développement.

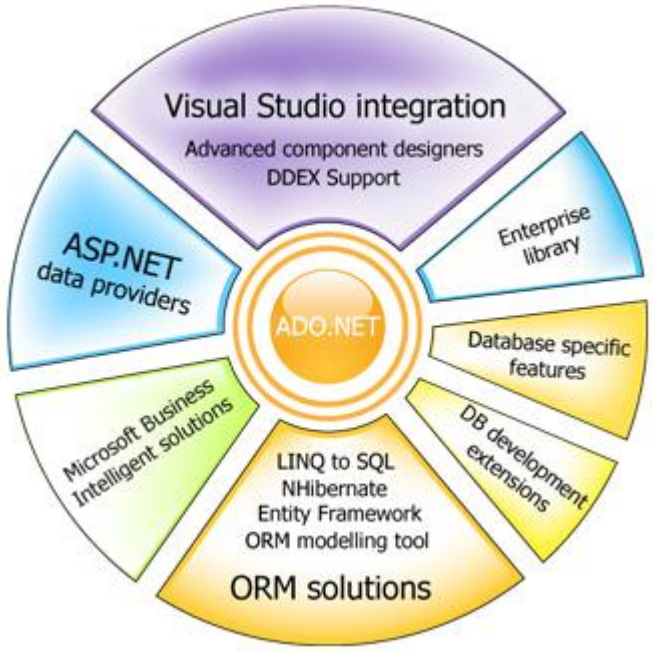
Les règles sont basées sur le guideline de Microsoft.

6.2 Accès aux données

6.2.1 Principe

L'accès aux données est une notion qui va radicalement changer avec la migration de Framework. En version 2.0 l'accès se faisait au moyen d'ADO.NET et de l'ODP fourni par oracle.

La solution qui sera choisie devra couvrir l'ensemble des fonctionnalités fournies dans le contexte actuel.



Nous avons eu plusieurs possibilités :

- Nhibernate avec ODP.NET
- Entity Framework
- ADO, ODP.NET (ancienne méthode)

Il est à noter que la dernière solution ne repose pas sur une logique de mapping O/R.

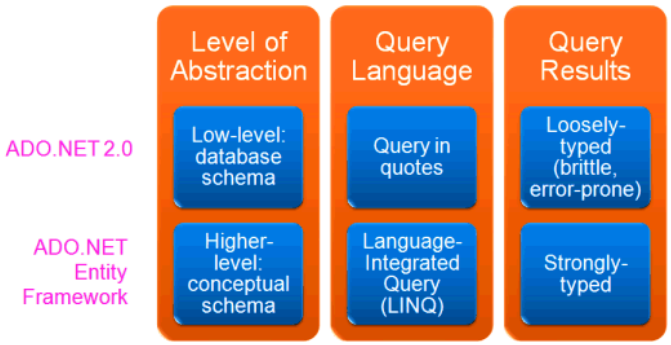
6.2.2 Choix d'un provider d'accès aux données

La préconisation Microsoft est d'utiliser un modèle O/R ainsi que le provider fourni par devArt dans un premier temps (sortie courant 2010 de la technologie Entity fournie par Oracle). Les raisons d'un choix orienté mapping O/R sont multiples :

Style de programmation orienté objet pour tous les SGBDs (moins de code)

Abstraction du SGBD

Facilité de migration : possibilité de changer de SGBD simplement en modifiant une variable de configuration



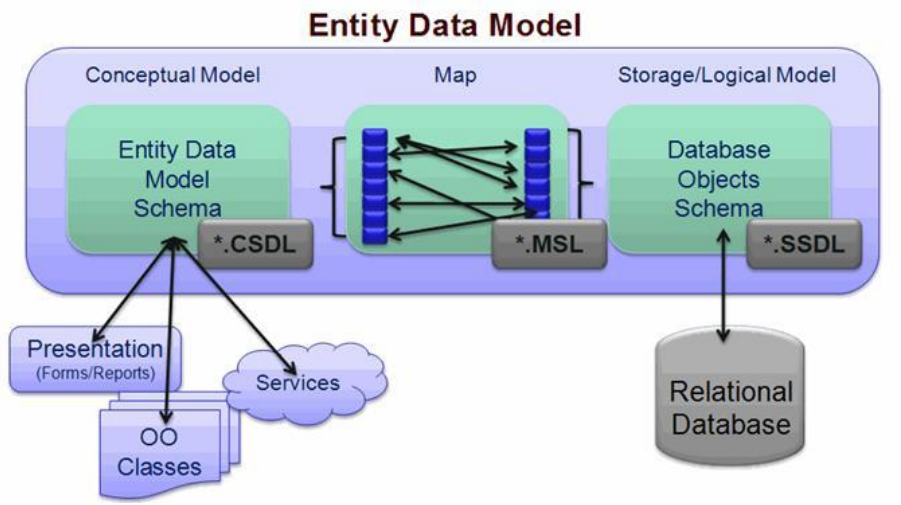
6.2.3 Mécanismes d'accès aux données

Sauf contre indication, les mécanismes d'accès aux données mis en place reposent sur :

un développement par couches,

une logique de sélection fournie par le provider Entity

des procédures stockées pour l'insertion et la mise à jour aux données



La mise en place de ces principes permet de simplifier la mise au point et la maintenance d'une solution informatique.

6.3 Implémentation des membres d'une classe

Ce paragraphe propose des préconisations d'utilisation.

6.3.1 Génération des entités métiers

6.3.1.1 Les entités POCO

L'utilisation de POCO signifiant Plain Old CLR Object avec le Framework Entity permet de créer des entités (classes métier), sans se soucier de la persistance des données contenues dans les objets qu'elles permettront de créer, tout en gardant les avantages liés à l'utilisation du Framework Entity. Autrement dit, nous travaillons avec des objets sans se soucier de la manière dont ils sont persistés.

6.3.1.2 Génération des entités

Dans le contexte de la nouvelle architecture d'accès aux données les entités sont générées à l'aide d'un fichier template « .tt ». Celui-ci se base sur le modèle généré à partir d'Oracle (fichier « .edmx »). La procédure de génération des entités consiste à sauvegarder le fichier « types.tt » dans son arborescence, on retrouvera les diverses entités générées dans l'éventualité d'une modification.

6.3.1.3 Gestion des modifications : ChangeTracker

Chacun des objets comporte une gestion de modification (ChangeTracker) qui permettra de déclencher les opérations nécessaires à la validation par exemple. Cette fonctionnalité permet de s'affranchir de la validation par code qui pouvait être fastidieuse avec l'ancienne méthode.

```
IEntityChangeTracker _changeTracker = null;


// Specify the IEntityChangeTracker to use for tracking changes.
void IEntityWithChangeTracker.SetChangeTracker(IEntityChangeTracker changeTracker)
{
   _changeTracker = changeTracker;
   // Every time the change tracker is set, we must also set all the
   // complex type change trackers.
   if (_extendedInfo != null)  {
      _extendedInfo.SetComplexChangeTracker("ExtendedInfo", _changeTracker);
   }
}
```

6.3.2 Utilisation des propriétés

Déterminez si une propriété ou une méthode est la plus appropriée pour votre besoin. Vous pouvez consulter le paragraphe, Propriétés vs Méthodes pour vous aider. Choisissez un nom basé sur les recommandations du nommage des propriétés.

Quand vous implémentez l'accesseur set, n'écrasez l'ancienne valeur qu'à la fin de la méthode ceci pour éviter de perdre cette valeur si une exception se produit dans la méthode.

6.3.2.1 Etat d'une propriété

Les propriétés d'une classe doivent pouvoir être accédées dans n'importe quel ordre. Il ne doit pas y avoir de dépendance d'état entre plusieurs propriétés. Tout accès à une propriété doit laisser l'état de l'objet cohérent.

Si une fonctionnalité dépend de la valeur de plusieurs propriétés, cette fonctionnalité n'est active que quand l'objet est dans un état correct.

Par exemple, une TextBox a deux propriétés reliées : DataSource et DataField. DataSource spécifie le nom de la table et DataField le nom de la colonne. Quand les 2 propriétés sont spécifiées, le control peut charger automatiquement la valeur de la colonne. L'ordre d'affectation des propriétés n'a pas d'importance.

Exemple de codage de ces propriétés :

```
public class TextBox
{
   string dataSource;
   string dataField;
   bool active;
   public string DataSource
   {
      get
      {
         return dataSource;
      }
      set
      {
         if (value != dataSource)
         {
            // Update active state.
            SetActive(value != null && dataField != null);
            dataSource = value;
         }
      }
   }

   public string DataField
   {
      get
      {
         return dataField;
      }
      set
      {
         if (value != dataField)
         {
            // Update active state.
            SetActive(dataSource != null && dataField != null);
            dataField = value;
         }
      }
   }
   void SetActive(Boolean value)
   {
      if (value != active)
      {
         if (value)
         {
            Activate();
            Text = dataBase.Value(dataField);
         }
         else
         {
            Deactivate();
            Text = "";
         }
         // Set active only if successful.
         active = value;
      }
   }
   void Activate()
   {
      // Open database.
   }

   void Deactivate()
   {
      // Close database.
   }
}
```

Dans l'exemple précédent, l'expression suivante détermine si l'objet est dans un état correct et si la liaison de données peut-être activée.

```
value != null && dataField != null
```

6.3.2.2 Propriétés vs Méthodes

En général, les méthodes représentent des actions et les propriétés des données. Utilisez les règles suivantes pour vous aider à choisir entre ces deux options.

Utilisez une propriété quand vous avez à faire à une donnée logique.

```
public string Name
{
   get
   {
      return name;
   }
   set
   {
      name = value;
}
```

Utilisez une méthode quand :

- L'opération est une conversion, tel que Object.ToString.
- L'opération est très consommatrice de ressources et vous souhaitez informer l'utilisateur qu'il est nécessaire de stocker l'information plutôt que de la récupérer à chaque fois.
- Obtenir la valeur engendre des effets de bord (Modification de l'état de l'objet…)
- Appeler cette fonctionnalité plusieurs fois ne génère pas obligatoirement le même résultat.
- L'ordre est important. Rappelez-vous que l'ordre d'appel des propriétés d'un objet ne doit pas avoir d'importance.
- La fonction est statique mais retourne une valeur qui peut changer.
- La fonction renvoie un tableau. Les propriétés renvoyant un tableau peuvent être source d'erreurs.

Habituellement, il est nécessaire de renvoyer une copie du tableau de façon à ce que l'utilisateur ne puisse pas en modifier le contenu interne. Ceci, avec le fait que l'utilisateur peut légitimement croire que cette propriété est indexée, génère du code non performant. Dans l'exemple suivant, chaque appel de la propriété Methods crée une copie du tableau. Le résultat génère 2n + 1 copies du tableau à chaque boucle.

```
Type type = // Get a type.
for (int i = 0; i < type.Methods.Length; i++)
{
   if (type.Methods[i].Name.Equals ("text"))
   {
      // Perform some operation.
   }
}
```

L'exemple suivant illustre l'utilisation correcte des propriétés et des méthodes.

```
class Connection
{
// Les 3 membres suivants doivent être des propriétés
// car ils peuvent être appelés dans n'importe quel ordre.
string DNSName {get{};set{};}
string UserName {get{};set{};}
string Password {get{};set{};}

// Le membre suivant doit être une méthode car l'ordre
// d'exécution est important.
// Cette méthode ne peut pas être appelée tant que les propriétés
// n'ont pas été affectées.
bool Execute ();
}
```

6.3.2.3 Propriétés lecture seule et écriture seule

Vous devez utiliser des propriétés en lecture seule {get} quand l'utilisateur ne doit pas changer la valeur.

N'utilisez jamais de propriétés en écriture seule {set}

6.3.2.4 Utilisation des propriétés indexées

Note : Une propriété indexée est aussi appelée un indexeur.

Utilisez les règles suivantes pour les propriétés indexées :

- Utilisez une propriété indexée quand la donnée est un tableau.
- N'utilisez que des entiers ou des chaînes comme index. Si vous devez utiliser d'autres types de valeur, utilisez une méthode.
- N'utilisez que des indexeurs à une dimension sinon utilisez une méthode.
- N'utilisez qu'une propriété indexée par classe et définissez la comme propriété indexée par défaut.
- N'utilisez pas de propriété indexée qui ne sont pas par définit par comme par défaut (C# ne le permet pas)
- Nommez la propriété indexée Item. Utilisez cette règle, à moins qu'il existe un nom beaucoup plus adapté tel que la propriété Chars de la classe String. En C#, les indexeurs sont toujours nommés Item.

6.3.3 Utilisation des événements

Utilisez void comme valeur de retour pour un handler d'événement. Le nom du handler doit être suffixé par Handler.

```
public delegate void MouseEventHandler(object sender, MouseEventArgs e);
```

Utilisez une classe de données fortement typée comme valeur de l'événement, tel que les coordonnées de la souris. Une classe de donnée d'événement doit hériter de System.EventArgs tel que montré dans l'exemple suivant :

```
public class MouseEvent: EventArgs {}
```

Utilisez une méthode protégée (protected) pour la méthode qui va déclencher l'événement. Le but de cette méthode est de fournir une possibilité à une classe dérivé d'intercepter le moment ou l'événement va être déclenché.

Le nom de cette méthode prend la forme de OnEventName où EventName est le nom de l'événement qui va être envoyé.

Voici un exemple d'implémentation :

```
public delegate void ButtonClickHandler(object sender, ClickEventArgs e) ;
public class ClickEventArgs : System.EventArgs { /* votre implémentation */}
public class Button
{
   ButtonClickHandler onClickHandler;

   protected virtual void OnClick(ClickEventArgs e)
   {
      // Call the delegate if non-null.
      if (onClickHandler != null)
            onClickHandler(this, e);
   }
}
```

La classe dérivée peut choisir de ne pas appeler l'implémentation de la classe de base et ne pas générer l'événement. Pour cette raison, n'incluez jamais de traitement particulier dans la méthode de la classe de base qui pourrait ne pas être exécuté.

Vous devez prendre en compte qu'un handler d'événement peut contenir du code. Dans tous les cas, l'objet doit rester dans un état cohérent après que l'évènement est été envoyé. Utilisez des blocs try/finally au niveau du code qui génère l'événement. Puisque le développeur qui intercepte l'événement peut appeler d'autres fonctions de l'objet lorsqu'il traite celui-ci, vous ne pouvez pas faire des estimations sur l'état d'un objet entre le moment où l'événement a été émis et la suite du code. Par exemple, ne stockez pas la valeur d'une propriété avant l'émission de l'événement pour la réutiliser après, relisez là dans tous les cas.

```
public class Button
{
   protected void DoClick()
   {
      int a = this.MaPropriete;
      try
      {
         // Call event handler.
         OnClick();
      }
      finally
      {
      }
      // N'utilisez pas la valeur de a, rechargez la avant de l'utiliser
      a = this.MaPropriete;
      // Utilisation de a
   }
}
```

Utilisez ou étendez la classe System.ComponentModel.CancelEventArgs pour permettre à un développeur de contrôler l'événement que vous émettez. Cette classe permet au développeur qu'il souhaite canceler l'opération à votre charge de tester la valeur retournée. Par exemple, le contrôle TreeView émet un événement quand l'utilisateur souhaite modifier un label. Le code suivant montre comment vous pouvez empêcher cette opération.

```
public class Form1: Form
{
   TreeView treeView1 = new TreeView();

   void treeView1_BeforeLabelEdit(object source,
      NodeLabelEditEventArgs e)
   {
      e.CancelEdit = true;
   }
}
```

Notez que dans ce cas, aucune erreur n'est générée. Le label est considéré comme en lecture seule.

Cette méthode n'est pas appropriée dans le cas où le développeur souhaite canceller l'opération et retourner une erreur. Dans ce cas, vous devez émettre une erreur à l'intérieur de votre handler.

6.3.4 Utilisation des méthodes

Choisissez un nom en respectant les règles de nommage des méthodes.

Par défaut, les méthodes ne sont pas virtuelles. Ne rendez une méthode virtuelle que si cela est réellement nécessaire.

6.3.4.1 Surcharge des méthodes

La surcharge d'une méthode survient quand une classe contient plusieurs méthodes de même nom ayant des signatures différentes (paramètres).

Cette section contient quelques règles pour utiliser la surcharge de méthode.

Utilisez la surcharge de méthode pour fournir différentes méthodes qui effectue sémantiquement la même chose. La surcharge permet de simuler la notion de paramètre par défaut.

Utilisez les paramètres par défaut correctement. Dans un groupe de méthodes surchargées, la méthode la plus complexe doit utiliser des noms de paramètres qui indiquent le changement par rapport aux valeurs par défaut des méthodes plus simple. Par exemple, dans le code suivant, la 1ére méthode présume que la recherche ne dépend pas de la casse. La seconde méthode utilise le nom ignoreCase plutôt que caseSensitive pour indiquer quelle est la différence avec la valeur par défaut.

```
// Method #1: ignoreCase = false.
MethodInfo Type.GetMethod(String name);
// Method #2: Indicates how the default behavior of method #1 is being // changed.
 MethodInfo Type.GetMethod (String name, Boolean ignoreCase);
```

Respectez un ordre et un nommage des paramètres cohérent entre les différentes méthodes surchargées. Il est courant de fournir un ensemble de méthodes surchargées avec un nombre croissant de paramètres. Dans l'exemple suivant, la définition des paramètres conserve un ordre et des noms cohérents.

```
public class SampleClass
{
   readonly string defaultForA = "default value for a";
   readonly int defaultForB = "42";
   readonly double defaultForC = "68.90";

   public void Execute()
   {
      Execute(defaultForA, defaultForB, defaultForC);
   }

   public void Execute (string a)
   {
      Execute(a, defaultForB, defaultForC);
   }

   public void Execute (string a, int b)
   {
      Execute (a, b, defaultForC);
   }

   public void Execute (string a, int b, double c)
   {
      Console.WriteLine(a);
      Console.WriteLine(b);
      Console.WriteLine(c);
      Console.WriteLine();
   }
}
```

Si vous devez rendre les méthodes virtuelles, ne définissez que la méthode la plus complexe comme virtuelle.

```
public int IndexOf(string s)
{
   return IndexOf (s, 0);
}
public int IndexOf(string s, int startIndex)
{
   return IndexOf(s, startIndex, myString.Length - startIndex );
}
public virtual int IndexOf(string s, int startIndex, int count)
{
   return myString.IndexOf(s, startIndex, count);
}
```

6.3.4.2 Méthode avec un nombre variable de paramètres

Vous pouvez définir une méthode avec un nombre variable de paramètres en utilisant le mot-clé params. Par exemple, utilisez le code suivant plutôt que d'utiliser un ensemble de méthodes surchargées.

```
void Format(string formatString, params object [] args)
```

Pour des raisons de performance, vous pouvez définir un petit sous-ensemble de méthodes surchargées. Vous devez faire cela uniquement si chaque méthode implémente un code spécifique permettant d'optimiser le code et ne se contente pas d'appeler la méthode la plus complexe.

```
void Format(string formatString, object arg1)
void Format(string formatString, object arg1, object arg2)
void Format(string formatString, params object [] args)
```

6.3.5 Utilisation des constructeurs

Utilisez les règles suivantes pour la définition des constructeurs.

Fournissez un constructeur privé par défaut s'il y a uniquement des méthodes et propriétés statiques dans cette classe. Un constructeur privé empêchera la classe d'être instanciée.

```
public sealed class Environment
{
   // Private constructor prevents the class from being created.
   private Environment()
   {
      // Code for the constructor goes here.
   }
}
```

Réduisez au maximum les traitements contenus dans le constructeur. Un constructeur ne devrait pas faire plus que de stocker la valeur des paramètres.

Fournissez un constructeur pour chaque classe. Si vous ne définissez pas un constructeur, C# ajoutera implicitement un constructeur par défaut. Si la classe est abstraite, ce sera un constructeur protected.

Fournissez un constructeur protected par défaut qui pourra être utilisé par les classes dérivées.

Utilisez des paramètres dans les constructeurs pour simplifier l'initialisation des propriétés

De la même manière que les méthodes surchargées, utilisées un ordre et des noms cohérents pour les paramètres.

6.3.6 Utilisation des champs

Ne définissez jamais des champs public ou protected. Si des champs doivent être accédés en dehors de votre classe, définissez des propriétés. Ceci permet une évolution plus facile de votre classe en conservant une compatibilité binaire au niveau des interfaces.

Exposez un champ à vos classes dérivées en utilisant une propriété protected. Une classe dérivée ne doit jamais accéder directement au champ de sa classe de base.

```
public class Control: Component
{
   private int handle;
   protected int Handle
   {
      get
      {
         return handle;
      }
   }
}
```

Utilisez le mot-clé const pour déclarer des constantes. Le compilateur utilisera directement la valeur de la constance plutôt qu'une référence au champ.

Utilisez des champs en static readonly pour des instances d'objets prédéfinis. Utilisez le Pascal Casing dans ce cas car les champs sont public. N'oubliez pas que vous pouvez utiliser un constructeur statique pour initialiser de tels champs.

```
public struct Color
{
   public static readonly Color Red = new Color(0x0000FF);
   public static readonly Color Green = new Color(0x00FF00);
   public static readonly Color Blue = new Color(0xFF0000);
```

```
   public static readonly Color Black = new Color(0x000000);
   public static readonly Color White = new Color(0xFFFFFF);
}
```

N'utilisez pas les majuscules pour un nommé un champ.

N'utilisez pas la notation hongroise pour les champs, un nom correct décrit la sémantique pas le type.

Par convention, préfixez les noms des champs par un _ pour éviter que la distinction entre le champ et la propriété ne se fasse que sur la casse.

6.3.7 Utilisation des paramètres

Vérifiez systématiquement la validité des arguments pour les méthodes « public » ou « protected » et pour les propriétés d'écriture (set). Emettez des exceptions explicites pour le développeur. Utilisez les exceptions prédéfinis du framework (ArgumentNullException ou ArgumentOutOfRangeException) ou créez en des nouvelles à partir de la classe System.ArgumentException.

Exemple de validation de paramètre :

```
class SampleClass
{
   public int Count
   {
      get
      {
         return count;
      }

      set
      {
            // Check for valid parameter.
            if (count < 0 || count >= MaxValue)
               throw new ArgumentOutOfRangeException(
                  Sys.GetString("InvalidArgument","value",count.ToString()));
      }
   }

   public void Select(int start, int end)
   {
      // Check for valid parameter.
      if (start < 0)
         throw new ArgumentException(
             Sys.GetString("InvalidArgument","start",start.ToString()));
      // Check for valid parameter.
      if (end < start)
         throw new ArgumentException(
               Sys.GetString("InvalidArgument","end",end.ToString()));
   }
}
```

Faites bien la distinction entre le passage de paramètre par valeur ou par référence. Passer un paramètre par valeur copie la valeur qui est passée et n'a pas d'effet sur la valeur initiale. L'exemple suivant passe un paramètre par valeur.

```
public void Add(object value){}
```

Passer un paramètre par référence passé un lien sur la valeur. Toute modification sur le paramètre a un impact sur la valeur initiale. L'exemple suivant montre un passage par référence.

```
public static int Exchange(ref int location, int value){}
```

Passer un paramètre en sortie (out) correspond à un passage par référence sans que la valeur initiale soit initialisée. Si vous omettez ce mot-clé, le compilateur vous empêchera de passer une référence sur une valeur qui n'aura pas été initialisée.

```
[DllImport("Kernel32.dll"]
```

```
public static extern bool QueryPerformanceCounter(out long value)
```

6.3.8 Utilisation des classes de bases

Une classe peut-être abstract ou sealed. Une classe abstract nécessite d'implémenter une classe dérivée. Une classe sealed ne lui permet pas d'être dérivée.

6.3.8.1 Classes de base vs Interfaces

Un type interface permet de spécifier un protocole potentiellement supporté par plusieurs classes. Utilisez des classes de bases plutôt que des interfaces dans la mesure du possible. Dans une perspective de versionnage, les classes sont plus flexibles que les interfaces. Avec une classe, vous pouvez faire évoluer votre classe en ajoutant une méthode. Tant que la classe n'est pas abstraite (abstract), toutes les classes dérivées peuvent continuer à fonctionner.

Parce que les interfaces ne supportent pas l'héritage de l'implémentation, ce principe ne peut pas s'appliquer aux interfaces. Ajouter une méthode à une interface correspond à l'ajout d'une méthode abstraite sur une classe. Toutes les classes qui implémentent cette interface ne fonctionneront plus.

Les interfaces sont appropriées dans les cas suivants :

- Plusieurs classes sans relation doivent implémenter un protocole.
- Ces classes ont déjà une classe de base.
- L'agrégation n'est pas pratique ou approprié.

Dans tous les autres cas, l'héritage est plus adapté.

6.3.8.2 Utilisation des énumérations

Les règles suivantes s'appliquent lors de l'utilisation des énumérations :

N'utilisez pas le suffixe Enum sur les types énumérations.

Utilisez une énumération (enum) pour des paramètres, propriétés ou valeur de retour fortement typé.

Utilisez un enum plutôt que des constantes statiques.

N'utilisez pas d'enum pour un ensemble de valeurs non fixes (comme un numéro de version par exemple).

Utilisez l'attribut System.FlagsAttribute sur un enum uniquement si il sera nécessaire d'effectuer des ou (or) binaires sur les valeurs. Dans ce cas, utilisez des puissances de 2 pour pouvoir mixer plusieurs valeurs facilement comme sur l'exemple suivant :

```
[Flags()]
public enum WatcherChangeTypes
{
   Created = 1,
   Deleted = 2,
   Changed = 4,
   Renamed = 8,
   All = Created | Deleted | Changed | Renamed
};
```

Note : Le ou binaire peut s'utiliser directement lors de la déclaration de l'énumération.

Ne partez pas du principe qu'une valeur d'énumération est comprise dans une plage de valeurs. Il est possible de caster n'importe qu'elle valeur entière en un enum même si cette valeur ne correspond à rien. Pour éviter ce type d'erreur, vous devez vous assurer que la valeur est correcte. L'exemple suivant vous montre comment faire :

```
public void SetColor (Color color)
{
   if (!Enum.IsDefined (typeof(Color), color)
      throw new ArgumentOutOfRangeException();
}
```

6.3.8.3 Utilisation des délégués

Les délégués s'utilisent principalement pour les 2 utilisations suivantes :

- Notifications d'événements. Utilisez les délégués pour définir les événements pour plus d'informations sur la déclaration des événements, regardez le paragraphe sur l'utilisation des événements.
- Fonction de rappel (callback). Les fonctions de retour sont passées comme argument de méthode pour permettre au développeur d'appeler directement une fonction de multiples fois. Passer une méthode de comparaison à un algorithme de tri est une utilisation classique des méthodes de rappel. Nommez toujours vos méthodes de rappel avec le suffixe Callback.

6.3.8.4 Utilisation des types imbriqués

Un type imbriqué est un type définit à l'intérieur d'un autre. Les types imbriqués sont intéressants pour encapsuler les détails d'implémentation d'un type, tel qu'un énumérateur sur une collection, parce qu'ils peuvent avoir accès aux informations privées.

Les types imbriqués public doivent être utilisés très rarement.

N'utilisez pas les types imbriqués dans les cas suivants :

- Le type doit être instancié par du code client. Si un type a un constructeur public, c'est qu'il ne doit sûrement pas être imbriqué. Un type imbriqué ne doit pas être utilisé en dehors de son type « propriétaire » sans qu'il y soit en relation avec celui-ci.
- Des références existent sur le type dans le code client.

6.3.8.5 Utilisation des événements

Toute portion de code susceptible d'émettre un événement doit mettre à disposition une méthode permettant de tester son exécution sans émettre d'événement. Par exemple, pour éviter un FileNotFoundException vous devez appeler File.Exists.

Cette règle n'est pas toujours possible, mais le but est que dans le cadre d'une exécution normale aucune exception ne doit être générée.

Utilisez le suffixe Exception pour nommer vos classes exceptions.

Essayez d'utiliser au maximum les Exceptions standards du Framework.

Si vous créez de nouvelles exceptions, celles-ci ne doivent pas être propagées entre couches. Si vous devez propager une exception, encapsulez-la en utilisant la propriété InnerException. Pour le découpage en couche d'une application, voyez la section correspondante. L'exception que vous avez créée doit faire partie du namespace de la couche concernée.

N'incluez des informations supplémentaires dans votre exception que si cela a un intérêt pour un traitement particulier. En règle générale, il n'est pas nécessaire d'inclure des informations supplémentaires.

N'intégrez pas des informations sensibles dans le texte de vos messages. Des informations telles qu'un chemin d'accès sont considérées comme une info sensible.

N'utilisez pas des exceptions pour des erreurs normales ou attendues. Comme son nom l'indique, une exception ne doit arriver que pour prévenir que quelque chose d'exceptionnel vient d'arriver. Par exemple, un client qui n'existe pas ne doit pas être traité avec une exception.

Vous devez utilisez la valeur null pour gérer la plupart des cas d'anomalies. Par exemple, un File.Open doit renvoyer null si le fichier n'existe pas mais génère une exception si le fichier est verrouillé. N'utilisez jamais d'autre valeur de retour, si vous devez le faire, utilisez l'exception.

Concevez les classes de sorte que dans le cadre d'une utilisation normale aucune exception ne doit être générée.

Assurez vous de conserver un état cohérent lorsque vous émettez une exception. Par exemple, une Hashtable.Insert génère une exception, on est en droit de penser que l'insertion n'a pas été faite.

6.3.8.6 Exceptions standards

Voici une liste des exceptions les plus utilisées fournies par le Framework :

+--System.Object
  +--System.Exception
    +--System.SystemException
      +--System.ArgumentException
  +--System.ArgumentNullException
  +--System.ArgumentOutOfRangeException
  +--System.DuplicateWaitObjectException
    +--System.ArithmeticException
  +--System.DivideByZeroException
  +--System.OverflowException
  +--System.NotFiniteNumberException
    +--System.ArrayTypeMismatchException
    +--System.ExecutionEngineException
    +--System.FormatException
    +--System.IndexOutOfRangeException
    +--System.InvalidCastException
    +--System.InvalidOperationException
  +--System.ObjectDisposedException
    +--System.InvalidProgramException
    +--System.IO.IOException
  +--System.IO.DirectoryNotFoundException
  +--System.IO.EndOfStreamException
  +--System.IO.FileLoadException
  +--System.IO.FileNotFoundException
  +--System.IO.PathTooLongException
    +--System.NotImplementedException
    +--System.NotSupportedException
    +--System.NullReferenceException
    +--System.OutOfMemoryException
    +--System.RankException
    +--System.Security.SecurityException

+--System.Security.VerificationException

+--System.StackOverflowException

+--System.Threading.SynchronizationLockException

+--System.Threading.ThreadAbortException

+--System.Threading.ThreadStateException

+--System.TypeInitializationException

+--System.UnauthorizedAccessException

6.3.8.7 Utilisation des tableaux

Il n'est pas toujours facile de décider d'utiliser un tableau ou une collection. En général, vous devez utilisez une collection si vous devez effectuer des mises telles que Add, Remove ou Insert.

Ne retournez jamais l'instance interne d'un tableau car ceci permet au code appelant de modifier son contenu. L'exemple suivant montre comment le tableau badChars peut être changé en utilisant la propriété Path même si cette propriété n'implémente pas un accès en écriture.

```
public class ExampleClass
{
   public sealed class Path
   {
      private Path(){}
      private static char[] badChars = {'\"', '<', '>'};
      public static char[] GetInvalidPathChars()
      {
         return badChars;
      }
   }
   public static void Main()
   {
      // Le code suivant modifie le contenu de badChars
      Path.GetInvalidPathChars()[0] = 'A';
   }
}
```

Pour corriger ce problème, vous devez renvoyer une copie du tableau en utilisant la méthode clone.

```
public static char[] GetInvalidPathChars()
{
   return (char[])badChars.Clone();
}
```

Note : L'utilisation de readonly sur le tableau ne résoudrait pas le problème. Car il ne s'appliquerait qu'au tableau (qui ne pourrait pas évoluer) mais à son contenu (qui pourrait être modifié).

6.3.8.8 Retourner des tableaux vides

Les propriétés renvoyant des tableaux ou des chaînes ne doivent jamais renvoyer un null. La valeur nulle peut être difficile à traiter pour un développeur. Par exemple, un utilisateur peut considérer que le code suivant marchera dans tous les cas :

```
public void DoSomething()
{
   string s = SomeOtherFunc();
   if (s.Length > 0)
   {
      // Do something else.
   }
}
```

La règle veut qu'un null, une chaîne vide ou un tableau vide soit traité de la même manière. Renvoyez toujours un tableau vide plutôt qu'un null.

6.3.9 Logique de réservation

6.3.9.1 Principe de base

Concernant la logique de persistance pour les clients riches il est nécessaire de procéder selon un mécanisme de réservation de données en base.

En effet un utilisateur qui arrivera sur un écran lui présentant une entité métier se verra octroyer un droit de réservation dans oracle. C'est lui seul qui pourra faire une action de sauvegarde sur la donnée et cela tant qu'il n'aura pas quitté son écran.

6.3.9.2 Gestion des transactions

Chaque objet métier ou écran possédera un numéro de transaction généré par .Net lors de l'interaction avec la base oracle. Cet identifiant sera conservé dans le client .Net. A chaque appel d'une procédure stockée d'enregistrement on va se servir de cet identifiant comme clé identifiant la transaction vers oracle.

6.3.9.3 Implémentation dans oracle

6.3.9.3.1 Sélection

Une procédure stockée de sélection devra ressembler à cela :

```
SELECT Champ1, Champ2, Champ3
FROM MA_TABLE  FOR UPDATE NOWAIT ;
```

6.3.9.3.2 Autre procédures

La syntaxe concernant l'Update, le Delete ne varie pas dans Oracle c'est essentiellement dans le code .Net que les différences seront visibles.

6.3.9.4 Implémentation .Net

L'implémentation repose globalement sur les principes ci-dessous :

Lorsqu'on instancie la classe servant à se connecter à oracle via ODP on créé le GUID et on enlève le Commit automatique qui perdrait les informations de transaction.

```
//On initialise le GUID de la classe
if (m_uniqueKey==System.Guid.Empty)
m_uniqueKey=System.Guid.NewGuid();

//et on spécifie AutoCommit à false
base.AutoCommit=false;
```

Lorsqu'on effectue la transaction dans oracle on ne fermera pas la connexion qui persistera contrairement à l'ancienne méthode :

```
try
{
…
```

```
            cmd.ExecuteNonQuery();
        }
        catch (Exception ex)
        {
            throw;
        }
        finally
        {
            //on ne ferme plus la connexion à la base de données
            //afin de ne pas perdre la transaction en cours
            //CloseDataBase();
        }
```

Lorsqu'on voudra effectivement faire une sauvegarde on procédera à un commit qui aura pour effet de libérer le numéro de transaction dans .Net et de libérer l'enregistrement dans Oracle.

6.3.9.5 Remarque

Un enregistrement est réservé dans Oracle donc quelque soit l'outil (SQLDeveloppeur, .Net, Oracle Forms, PL/SQL) l'enregistrement ne pourra pas être modifié.

7 DOCUMENT 3 – 1.00 – C# - REGLES DE NOMMAGE ET RECOMMANDATIONS

7.1 Introduction

Le présent document constitue les « Règles de nommages et les Recommandations » que devront utiliser les développeurs de la Région PACA et les prestataires de service développant pour la Région PACA, dans le cadre de leurs projets utilisant le langage C# au travers de l'outil de développement Visual Studio .NET

Ces Règles et Recommandations visent à améliorer la lisibilité et permettent la mise en œuvre commune d'une charte accessible et utilisée par l'ensemble d'une équipe de développement.

Les points abordés dans ce document concernent :

- Les commentaires,
- les règles de nommage des classes, variables, fonctions, structures…,
- les règles de nommage des objets des ihm.

Des particularités seront précisées dans le cas du code généré automatiquement par l'outil de développement (utilisation des Wizard).

L'objet de ce document n'est pas de former un développeur au langage C# et à l'outil Visual Studio .Net. Les connaissances minimales sont considérées acquises avant la lecture et assimilation de ces Règles.

Les règles de nommage sont basées sur le guideline de Microsoft.

7.2 règles de nommage

7.2.1 Principes généraux

Le nom des objets et des méthodes sont le plus descriptifs possible.

Ils ne doivent pas dépasser 35 caractères.

Pas d'underscore (sauf pour préfixer), pas de tiret, pas de caractères accentués.

Ne pas ajouter de mots qui n'apportent pas de valeur ajoutée. Exemple Employe.NomEmploye doit devenir Employe.Nom

Les abréviations peuvent être utilisées si ces dernières sont connues de tous ou facilement identifiables. Par exemple, Client.ID est acceptable à la place de Client.Identifiant ou Client.Identite.

7.2.2 Capitalisation

Utilisez les conventions suivantes pour la capitalisation des identifiants.

7.2.2.1 Pascal Casing

La 1ère lettre de chaque mot est en majuscule, les autres en minuscules.

```
Ex : BackColor
```

7.2.2.2 Camel Casing

La 1ère lettre du 1er mot est en minuscule et les 1ères lettres des mots suivants sont en majuscules.

```
Ex : backColor
```

7.2.2.3 Majuscule

Toutes les lettres sont en majuscules. Utilisez cette convention pour les identifiants à deux ou trois lettres.

```
Ex : IO
```

La table suivante résume l'utilisation des conventions d'écriture pour les types et les identifiants.

Identifiant	Casse	Exemple
Class	Pascal	AppDomain
Enum type	Pascal	ErrorLevel
Enum values	Camel	fatalError
Event	Pascal	ValueChange
Exception class	Pascal	WebException Note : Se termine toujours par Exception.
Interface	Pascal	IDisposable Note : Commence toujours par la lettre I.
Method	Pascal	ToString
Namespace	Pascal	System.Drawing
Parameter	Camel	typeName
Property	Pascal	BackColor
Read-only Static field	Camel	redValue
Protected instance field	Camel	redValue Note : Rarement utilisée. Une propriété est préférable.
Public instance field	Camel	redValue Note : Rare

7.2.3 Casse

Pour éviter des confusions et garantir l'interopérabilité entre les langages, utilisez ces règles concernant l'utilisation de la sensibilité de la casse, n'employez pas des identifiants dont le nom est dépendent de la casse. Ne distinguez pas des identifiants uniquement par leurs casses.

```
Ex : arguments et Arguments
```

Bien que csharp soit sensible à la casse, cela reste une source d'erreur et n'est pas compatible avec l'utilisation d'un autre langage non sensible à la casse tel que VB.

7.2.4 Abréviations

N'utilisez pas des abréviations ou des contractions de terme comme tout ou partie d'un identifiant.

```
Ex : Utilisez GetWindow au lieu de GetWin
```

Quand cela est approprié, utilisez des acronymes pour remplacer des termes ou des phrases longues.

```
Ex : UI pour User Interface
```

Quand vous utilisez des acronymes, utilisez le Pascal Casing. S'il fait moins de 4 caractères, utilisez les majuscules.

Voici une liste de quelques abréviations couramment utilisées.

Abréviations	Commentaire
CRPaca	
UI	User Interface
IO	Input/Output
DAL	Data Access Layer (Note : Couche d'accès aux données)

7.2.5 Convention de nommage

7.2.5.1 Nommage des namespaces

La convention de nommages des namespaces peut se résumer ainsi :

CRPaca.[.Application][.Domaine]

```
Ex : CRPaca.SocleTechnique.Util
```

CRPaca correspond au nom de la société qui produit le namespace à aujourd'hui il n'y a pas d'ambiguïté.

Utilisez le Pascal Casing pour nommer les namespaces et le point comme séparateur.

Utilisez le pluriel si cela est sémantiquement approprié. Par exemple, System.Collections plutôt que System.Collection. Cette règle ne s'applique pas pour les abréviations.

Evitez les noms de namespace identiques à ceux du framework car il y aura conflit lors de l'utilisation d'intellisence dans Visual Studio.

N'utilisez pas le même nom pour un namespace et une classe.

Il n'y a pas de lien entre le namespace et une assembly mais dans la mesure du possible, on essayera de conserver une cohérence.

Nom de domaine prédéfini :

Domaine	Commentaire
Util	Utilitaire
Communs	Framework commun
Intranet	Outils collaboratifs

7.2.5.2 Nommage des classes

Les règles suivantes s'appliquent pour le nommage des classes :

- Utilisez le Pascal Casing
- Utilisez les abréviations avec modération
- N'utilisez pas de préfixe tel que C ou Cls pour nommer une classe
- N'utilisez pas le _ dans le nom de la classe (cela est valable pour tous les identifiants)
- De temps en temps, il est nécessaire de fournir un nom de classe qui commence par la lettre I, bien que la classe ne soit pas une interface. Ceci est possible aussi longtemps que I est la première lettre d'un mot entier qui est une partie du nom de la classe. Par exemple, le nom IdentityStore de classe est approprié
- Le cas échéant, employez un mot composé pour nommer une classe dérivée. La deuxième partie du nom de la classe dérivée devrait être le nom de la classe de base. Par exemple, ApplicationException est un nom approprié pour une classe dérivée d'une classe appelée Exception, parce qu'ApplicationException est du genre Exception.
- Cette règle n'est pas exclusive et doit être employée raisonnablement. Par exemple, Bouton est un nom approprié pour une classe dérivée de Control. Bien qu'un bouton soit dérivé de Control, l'utilisation de Control dans le nom de la classe rallongerait celui-ci inutilement.

Remarque :

Ces règles concernent uniquement les classes développées par le CR PACA ou les prestataires de service chargés des développements. L'outil de développement Visual Studio, au travers de ses assistants (Wizard), génère automatiquement (Notamment pour les aspects IHM) des classes dont les règles de nommage reposent alors sur celles de l'outil.

7.2.5.3 Nommage des interfaces

Les règles suivantes s'appliquent pour le nommage des classes :

- Utilisez des noms ou des phrases condensés, ou des adjectifs qui décrivent une fonctionnalité. Par exemple, IPersistable utilise un adjectif, IContainer un nom
- Utilisez le Pascal Casing
- Utilisez les abréviations avec modération
- Préfixez le nom de l'interface avec le lettre I
- N'utilisez pas le _ dans le nom des interfaces.

```
Ex : IContainer
```

7.2.5.4 Nommage des attributs

Un attribut permet d'intégrer des métas donnés dans un programme. Un attribut est d'abord une classe et doit se conformer aux règles de nommages d'une classe.

Un nom d'attribut sera TOUJOURS suffixé par le nom Attribute

```
Ex : ObsoleteAttribute
```

7.2.5.5 Nommage des énumérations

Les énumérations (Enum) héritent de la classe Enum. A ce titre, elles doivent être déclarées comme une classe.

Les règles suivantes s'appliquent aux énumérations :

- Utilisez le Pascal Casing pour les types Enum et leurs valeurs.
- Préfixer le nom par « Et » (Enumerate Type)
- Utilisez les abréviations avec modération
- N'utilisez pas de suffixe de type Enum
- Quand cela est possible, si l'enum concerne une propriété ou une classe reprenait son nom dans le nom de l'enum en lui ajoutant le suffixe Type.

```
Ex : EtBorderType
```

Utilisez toujours l'attribut FlagsAttribute dans le cas d'une énumération binaire.

```
Ex :
[Flags]
enum EtBorderType
{
  Single = 1,
  Double = 2
}
```

7.2.5.6 Nommage des structures

Les règles suivantes s'appliquent aux structures :

- Utilisez le Pascal Casing pour les types Struct.
- Préfixer le nom par « St » (Structure Type)
- Utilisez les abréviations avec modération
- N'utilisez pas de suffixe de type Struct
- Pour les variables de la structure se reporter au paragraphe Nommage des variables
- Préférer l'utilisation de classe à la place des structures

L'exemple suivant décrit une structure (StExemple) constituée de 4 éléments :

```
  //------------------------------------------------------------------------
  /// <summary>
  /// Exemple de structure
  /// </summary>
  //------------------------------------------------------------------------
  public Struct StExemple
  {
    public  int    iValeur,
    public  bool   bValeur,
    public  string strValeur,
    public  float  fValeur,
  };
```

7.2.5.7 Nommage des champs statiques

Les règles suivantes s'appliquent pour le nommage des champs statiques :

- Utilisez le Camel Casing
- Il est recommandé d'utiliser des propriétés statiques à la place de champs statiques

7.2.5.8 Nommage des variables

7.2.5.8.1 Principe général

Les règles suivantes s'appliquent pour le nommage des paramètres :

- Utilisez le Camel Casing
- Utilisez des noms décrivant précisément le paramètre. Ceci est très utile lors de l'utilisation d'Intellisence dans Visual Studio.

Les règles de nommage des variables sont les suivantes :

Préfixe précisant le type de la donnée	Commentaire
i	données entières (int, long, signée, non signée …)
b	données booléennes
f	données réelles
str	données de type chaîne de caractères
ds	données de type DataSet
obj	données objet

A titre d'exemple :

```
int      iValeur;
bool     bValeur;
float    fValeur;
string   strValeur;
DataSet  dsValeur;
TraceLog objTraceLog;
```

7.2.5.8.2 Objets Ihm

Les règles de nommage concernant les objets de type IHM reposent sur le principe suivant :

- Utilisation d'un préfixe spécifique, caractérisant le type d'objet IHM utilisé
- Utilisez le Camel Casing

Sur l'exemple d'un objet de type TextBox qui permet de saisir un type d'organisme, on obtient :

A titre d'exemple :

```
protected System.Web.UI.WebControls.TextBox tbTypeOrganisme;
```

L'ordre d'énumération des types d'objet IHM correspond à celui de la boîte à outils utilisée dans l'environnement de développement de formulaire ASP.Net.

Type d'objet	Préfixe à placer avant le nom de l'élément
Label	lbl
TextBox	tb
Button	btn
LinkButton	lnkbtn
ImageButton	imgbtn
RadioButton	rdobtn
CheckBox	chkbx
HyperLink	hyplnk
DropDownList	ddl
DataGrid	dg
DataList	dl
DateTimePicker	dtp

Pour les autres composants, la règle d'élaboration du préfixe est la suivante :

Préfixe en minuscule

Première lettre de chaque mot composé (Ex : **tb** pour **TextBox**) ou composition de 2 à 3 caractères maximum de chaque mot composé (Ex : **chkbx** pour **CheckBox**)

7.2.5.8.3 Les variables membres

Les variables membres d'une classe répondent à la même syntaxe que celle décrite au paragraphe précédent sauf qu'elles sont préfixées par « ***m***_ ».

A titre d'exemple :

```
//------------------------------------------------------------------------------
/// <summary>
/// Nom complet du fichier de trace (ex: "c:\Nom projet\log\Trace.log")
/// </summary>
//------------------------------------------------------------------------------
private string m_strFileName;
```

7.2.5.8.4 Les constantes

Les constantes répondent à la même syntaxe que celle décrite au paragraphe précédent sauf qu'elles sont préfixées par « ***c***_ ».

A titre d'exemple :

```
const int    c_iValeur   = 14;
const string c_strValeur = "Chaine";
```

7.2.5.9 Nommage des paramètres

Cf. Nommage des variables

7.2.5.10 Nommage des méthodes

Les règles suivantes s'appliquent pour le nommage des méthodes :

- Utilisez des verbes à l'infinitif pour décrire la fonctionnalité.
- Utilisez le Pascal Casing
- Utilisez une propriété si la méthode se contente de retourner une valeur sans paramètres d'entrée.
- Vous pouvez utiliser le franglais en utilisant les verbes du type : Get, Set, Update, Read, Delete, Commit … si cela a un intérêt et aide à la compréhension tout en limitant la longueur des noms.

7.2.5.11 Nommage des propriétés

Les règles suivantes s'appliquent pour le nommage des propriétés :

- Utilisez le Pascal Casing
- Essayez de nommer la propriété avec le même nom que le type renvoyé. Par exemple, si vous nommez une propriété Couleur, son type devra être nommé Couleur.

```
Exemple :
public class SampleClass
{
   public Color BackColor
   {
   }
}
```

L'exemple suivant montre une propriété avec le même nom que son type

```
public enum Color
{
   // Insert code for Enum here.
}
public class Control
{
   public Color Color
   {
      get {// Insert code here.}
      set {// Insert code here.}
   }
}
```

7.2.5.12 Nommage des événements

Utilisez les conventions de nommage suivantes :

Utilisez le Pascal Casing

Utilisez le suffixe EventHandler pour le nom des événements.

Spécifiez deux paramètres nommés sender et e. Sender représente l'objet qui a envoyé l'événement.

Le paramètre sender est toujours du type object même si il est possible d'utiliser un type plus spécifique.

L'état associé à l'événement est encapsulé dans une classe d'une classe EventArgs nommée e. Utilisez une classe appropriée pour e dérivée de EventArgs.

Cette classe devra être suffixée par EventArgs. Utilisez des verbes verbes au participe passé pour nommer des événements.

Ex : Clôturé, fermé, sélectionné ou du franglais Clicked, Dropped…

En anglais, il est aisé de préciser le moment de l'événement dans le verbe car cela ne nécessite pas de caractères accentués. Par exemple, utilisez des verbes comme Closing pour indiquer que l'événement est en train de se produire ou Closed pour indiquer qu'il est terminé.

N'utilisez pas des préfixes dans le nom de l'événement. Par exemple, utilisez Close au lieu de OnClose.

En général, vous devez fournir une méthode virtuelle protégée nommé OnXxx sur les types générant l'événement. Cette méthode ne doit contenir qu'un paramètre événement e car le sender est toujours l'instance en cours.

Exemple de nommage d'un événement.

```
public delegate void MouseEventHandler(object sender, MouseEventArgs e);
```

Exemple de nommage d'un argument d'événement

```
public class MouseEventArgs : EventArgs
{
   int x;
   int y;
   public MouseEventArgs(int x, int y)
      { this.x = x; this.y = y; }
   public int X { get { return x; } }
   public int Y { get { return y; } }
}
```

7.2.6 Fichiers

7.2.6.1 Fichiers C#

Principe général

Sauf **contrainte**, un fichier C# contient une et une seule classe.

Il porte le même nom que la classe qu'il contient. A titre d'exemple :

```
La classe UtilTraceLog est codée dans le fichier UtilTraceLog.cs
```

Utilise le Pascal Casing

Règle de nommage des composants d'un client riche

Type d'objet	Préfixe à placer avant le nom de l'élément
Boite de dialogue	Dlg
Fenêtre (*)	Wnd
Contrôle	Ctrl

(*) A l'exception de l'application GPEA qui utilise **Frm**

7.2.6.2 Fichiers xml

Les fichiers XML utilisent le Pascal Casing pour le nom des éléments et le Camel Casing pour les attributs.

7.3 Commentaires à appliquer aux objets d'un fichier C#

7.3.1 Préambule

L'objet principal d'un commentaire est d'ajouter au sein du code toutes informations permettant de rendre plus lisible ce dernier et de préciser éventuellement des limitations ou pré requis.

Il existe deux types de commentaires :

- Commentaires libres (généralement placés au milieu du code),
- Commentaires imposés (Entête de fichier, de classe, de fonctions,…).

7.3.2 Syntaxe

Pour mémoire, la syntaxe pour écrire un commentaire en C# est la suivante :

```
// Commentaire sur une ligne
```

ou

```
/*
Commentaire
Sur plusieurs lignes
*/
```

Par convention, on utilisera la première méthode pour les commentaires et on utilisera la seconde pour supprimer temporairement une partie du code en phase de développement.

A titre indicatif, l'outil de développement met à disposition dans la barre des commandes une fonction qui place ou supprime une partie de code en commentaire.

7.3.3 Outil de génération de documentation technique

Les commentaires imposés (Espace de noms, classes, méthodes et variables membres ou tout autre objet exporté) doivent utiliser des tags Xml permettant ainsi la génération automatique de documentation technique type « JavaDoc ».

Dans ce cas de figure, les commentaires sont :

- encadrés par des tags,
- commence par un triple /

Exemple :

```
    //------------------------------------------------------------------------------
    /// <summary>
    /// Commentaire généré dans la documentation technique
    /// </summary>
    //------------------------------------------------------------------------------
```

Pour plus d'information sur les tags Xml, le lecteur se référera à la documentation du MSDN de MicroSoft.

7.3.4 Commentaires

Un fichier source C# contient le code relatif à une classe. Ce fichier, généralement généré automatiquement à l'aide de l'assistant de l'outil, a la structure suivante :

```
using xxx
namespace yyy
{
  class zzz
  {
    Liste des méthodes et variables membres de la classe
  }
}
```

7.3.4.1 Déclarations

La zone de déclaration permet de spécifier les librairies avec lesquelles se lie la classe. Exemple :

```
using System;
```

Dans le cas d'utilisation de librairies externes plutôt que celles livrées en standard dans l'environnement de développement, le développeur commentera l'utilité de cette librairie et éventuellement les pré-requis associés. Exemple :

```
using Oracle.DataAccess.Client; // Nécessite d'avoir installé la librairie Oracle DataProvider
for .NET
using Oracle.DataAccess.Types;  // Nécessite d'avoir installé la librairie Oracle DataProvider
for .NET
```

7.3.4.2 L'espace de nom

L'espace de nom permet de définir la déclaration et portée d'utilisation d'une librairie. Cette définition est généralement commune à une librairie et donc un à ensemble de classes.

Le cartouche de commentaires donné ci-dessous met en place les tags Xml.

Il permet de suivre :

- la version du composant,
- ses évolutions
- un descriptif général de ses fonctionnalités.

```
//-----------------------------------------------------------------------------
/// <summary>
/// Espace de nom       : CRPaca.SocleTechnique.Util
/// Logiciel            : Socle Technique CR Paca
/// Version du composant: 1.00
/// Modification:
///
/// Description:
/// Utilitaires du logiciel:
///   Accès à une clé de la base de registre
///   Trace log
///   Envoi d'eMail
///   Paramètrage applicatif au travers d'un fichier Xml
///   Le log d'évènement dans l'OS
/// </summary>
//-----------------------------------------------------------------------------
namespace CRPaca.SocleTechnique.Util
{
  ...
}
```

7.3.4.3 La classe

La classe permet de définir l'ensemble des méthodes, variables… relatives à l'objet codé.

Le cartouche de commentaires donné ci-dessous met en place les tags Xml. Il permet de préciser :

- le nom,
- un exemple d'utilisation de la classe,
- les remarques ou complément associés à l'objet.

```
  //-----------------------------------------------------------------------------
  /// <summary>
  /// Classe permettant la gestion des traces
  /// </summary>
  /// <example>Exemple d'utilisation
  /// <code>
  /// void MyFunction(void)
  /// {
  ///   UtilTraceLog  traceLog = UtilTraceLog.OpenLog("Nom Application", "Trace");
  ///   traceLog.WriteLine("{0} {1} {2}", "Coucou", 10, 12.18);
```

```
  ///   DataSet ds;
  ///   traceLog.TraceTable(ds.Tables["TABLE"]);
  ///   ...
  /// }
  /// </code>
  /// </example>
  /// <remarks>
  /// Le paramètrage des traces s'effectue en base de registre [HKEY_LOCAL_MACHINE\SOFTWARE\CR
Paca\Nom projet\TraceLog] dans les clés:
  ///   RepTraceLog   : Répertoire de la machine contenant l'ensemble des fichiers log (ex
"C:\log")
  ///   IsTraceActive : Flag permettant d'activé (="1") ou non (="0") les traces
  ///   IsDebugActive : Flag permettant d'activé (="1") ou non (="0") les traces dans la
console de debug de Visual Studio. Cette option a sens qu'en mode de compilation Debug et si
IsTraceActive="1"
  /// </remarks>
  /// <remarks>
  /// Exemple d'utilisation
  /// <code>
  ///   UtilTraceLog  traceLog = UtilTraceLog.OpenLog("Nom Application", "Trace");
  ///   traceLog.WriteLine("{0} {1} {2}", "Coucou", 10, 12.18);
  ///   DataSet ds;
  ///   traceLog.TraceTable(ds.Tables["TABLE"]);
  /// </code>
  /// </remarks>
  //---------------------------------------------------------------------------
  public class UtilTraceLog
  {
    ...
  }
```

7.3.4.4 Les méthodes d'une classe

Les méthodes contiennent le code élémentaire d'une fonction d'une classe.

Le cartouche de commentaires donné ci-dessous met en place les tags Xml.

Il permet de préciser :

- la fonctionnalité réalisée par la méthode,
- l'ensemble des paramètres d'entrée et de sortie.

```
    //---------------------------------------------------------------------------
    /// <summary>
    /// Retourne le libellé de l'erreur spécifiée
    /// </summary>
    /// <param name="numErr">Numéro d'erreur</param>
    /// <returns>Libellé de l'erreur</returns>
    //---------------------------------------------------------------------------
    public string GetError(Et_err numErr)
    {
      // Code
    }
```

7.3.4.5 Les objets exportés d'une classe

Les objets exportés d'une classe sont :

- Les variables membres,
- les structures,
- les énumérations,

Le cartouche de commentaires donné ci-dessous met en place les tags Xml. Il permet de préciser la nature de l'objet.

```
    //---------------------------------------------------------------------------
    /// <summary>
    /// Nom complet du fichier de trace (ex: "c:\Nom projet\log\Trace.log")
    /// </summary>
    //---------------------------------------------------------------------------
    private string m_strFileName;
```

8 DOCUMENT 3 – 1.01 – C# - REGLES DE NOMMAGE ET RECOMMANDATIONS

8.1 Introduction

Le présent document constitue les « Règles de nommages et les Recommandations » que devront utiliser les développeurs de la Région PACA et les prestataires de service développant pour la Région PACA, dans le cadre de leurs projets utilisant le langage C# au travers de l'outil de développement Visual Studio .NET

Ces Règles et Recommandations visent à améliorer la lisibilité et permettent la mise en œuvre commune d'une charte accessible et utilisée par l'ensemble d'une équipe de développement.

Les points abordés dans ce document concernent :

- Les commentaires,
- les règles de nommage des classes, variables, fonctions, structures…,
- les règles de nommage des objets des ihm.

Des particularités seront précisées dans le cas du code généré automatiquement par l'outil de développement (utilisation des Wizard).

L'objet de ce document n'est pas de former un développeur au langage C# et à l'outil Visual Studio .Net. Les connaissances minimales sont considérées acquises avant la lecture et assimilation de ces Règles.

Les règles de nommage sont basées sur le guideline de Microsoft.

8.2 règles de nommage

8.2.1 Principes généraux

Le nom des objets et des méthodes sont le plus descriptifs possible.

Ils ne doivent pas dépasser 35 caractères.

Pas d'underscore (sauf pour préfixer), pas de tiret, pas de caractères accentués.

Ne pas ajouter de mots qui n'apportent pas de valeur ajoutée. Exemple Employe.NomEmploye doit devenir Employe.Nom

Les abréviations peuvent être utilisées si ces dernières sont connues de tous ou facilement identifiables. Par exemple, Client.ID est acceptable à la place de Client.Identifiant ou Client.Identite.

8.2.2 Capitalisation

Utilisez les conventions suivantes pour la capitalisation des identifiants.

8.2.2.1 Pascal Casing

La 1ère lettre de chaque mot est en majuscule, les autres en minuscules.

```
Ex : BackColor
```

8.2.2.2 Camel Casing

La 1ère lettre du 1er mot est en minuscule et les 1ères lettres des mots suivants sont en majuscules.

```
Ex : backColor
```

8.2.2.3 Majuscule

Toutes les lettres sont en majuscules. Utilisez cette convention pour les identifiants à deux ou trois lettres.

```
Ex : IO
```

La table suivante résume l'utilisation des conventions d'écriture pour les types et les identifiants.

Identifiant	Casse	Exemple
Class	Pascal	AppDomain
Enum type	Pascal	ErrorLevel
Enum values	Camel	fatalError
Event	Pascal	ValueChange
Exception class	Pascal	WebException Note : Se termine toujours par Exception.
Interface	Pascal	IDisposable Note : Commence toujours par la lettre I.
Method	Pascal	ToString
Namespace	Pascal	System.Drawing
Parameter	Camel	typeName
Property	Pascal	BackColor
Read-only Static field	Camel	redValue
Protected instance field	Camel	redValue Note : Rarement utilisée. Une propriété est préférable.
Public instance field	Camel	redValue Note : Rare

8.2.3 Casse

Pour éviter des confusions et garantir l'interopérabilité entre les langages, utilisez ces règles concernant l'utilisation de la sensibilité de la casse :

- N'employez pas des identifiants dont le nom est dépendent de la casse. Ne distinguez pas des identifiants uniquement par leurs casses.

```
Ex : arguments et Arguments
```

Bien que csharp soit sensible à la casse, cela reste une source d'erreur et n'est pas compatible avec l'utilisation d'un autre langage non sensible à la casse tel que VB.

8.2.4 Abréviations

N'utilisez pas des abréviations ou des contractions de terme comme tout ou partie d'un identifiant.

```
Ex : Utilisez GetWindow au lieu de GetWin
```

Quand cela est approprié, utilisez des acronymes pour remplacer des termes ou des phrases longues.

```
Ex : UI pour User Interface
```

Quand vous utilisez des acronymes, utilisez le Pascal Casing. S'il fait moins de 4 caractères, utilisez les majuscules.

Voici une liste de quelques abréviations couramment utilisées.

Abréviations	Commentaire
CRPaca	
UI	User Interface
IO	Input/Output
DAL	Data Access Layer (Note : Couche d'accès aux données)

8.2.5 Convention de nommage

8.2.5.1 Nommage des namespaces

La convention de nommages des namespaces peut se résumer ainsi :

CRPaca.[.Application][.Domaine]

```
Ex : CRPaca.SocleTechnique.Util
```

CRPaca correspond au nom de la société qui produit le namespace à aujourd'hui il n'y a pas d'ambiguïté.

Utilisez le Pascal Casing pour nommer les namespaces et le point comme séparateur.

Utilisez le pluriel si cela est sémantiquement approprié. Par exemple, System.Collections plutôt que System.Collection. Cette règle ne s'applique pas pour les abréviations.

Evitez les noms de namespace identiques à ceux du framework car il y aura conflit lors de l'utilisation d'intellisence dans Visual Studio.

N'utilisez pas le même nom pour un namespace et une classe.

Il n'y a pas de lien entre le namespace et une assembly mais dans la mesure du possible, on essayera de conserver une cohérence.

Nom de domaine prédéfini :

Domaine	Commentaire
Util	Utilitaire
Communs	Framework commun
Intranet	Outils collaboratifs

8.2.5.2 Nommage des classes

Les règles suivantes s'appliquent pour le nommage des classes :

- Utilisez le Pascal Casing
- Utilisez les abréviations avec modération
- N'utilisez pas de préfixe tel que C ou Cls pour nommer une classe
- N'utilisez pas le _ dans le nom de la classe (cela est valable pour tous les identifiants)
- De temps en temps, il est nécessaire de fournir un nom de classe qui commence par la lettre I, bien que la classe ne soit pas une interface. Ceci est possible aussi longtemps que I est la première lettre d'un mot entier qui est une partie du nom de la classe. Par exemple, le nom IdentityStore de classe est approprié
- Le cas échéant, employez un mot composé pour nommer une classe dérivée. La deuxième partie du nom de la classe dérivée devrait être le nom de la classe de base. Par exemple, ApplicationException est un nom approprié pour une classe dérivée d'une classe appelée Exception, parce qu'ApplicationException est du genre Exception.
- Cette règle n'est pas exclusive et doit être employée raisonnablement. Par exemple, Bouton est un nom approprié pour une classe dérivée de Control. Bien qu'un bouton soit dérivé de Control, l'utilisation de Control dans le nom de la classe rallongerait celui-ci inutilement.

Remarque :

Ces règles concernent uniquement les classes développées par le CR PACA ou les prestataires de service chargés des développements. L'outil de développement Visual Studio, au travers de ses assistants (Wizard), génère automatiquement (Notamment pour les aspects IHM) des classes dont les règles de nommage reposent alors sur celles de l'outil.

8.2.5.3 Nommage des interfaces

Les règles suivantes s'appliquent pour le nommage des classes :

- Utilisez des noms ou des phrases condensés, ou des adjectifs qui décrivent une fonctionnalité. Par exemple, IPersistable utilise un adjectif, IContainer un nom
- Utilisez le Pascal Casing
- Utilisez les abréviations avec modération
- Préfixez le nom de l'interface avec la lettre I
- N'utilisez pas le _ dans le nom des interfaces.

```
Ex : IContainer
```

8.2.5.4 Nommage des attributs

Un attribut permet d'intégrer des métas donnés dans un programme. Un attribut est d'abord une classe et doit se conformer aux règles de nommages d'une classe.

Un nom d'attribut sera TOUJOURS suffixé par le nom Attribute

```
Ex : ObsoleteAttribute
```

8.2.5.5 Nommage des énumérations

Les énumérations (Enum) héritent de la classe Enum. A ce titre, elles doivent être déclarées comme une classe.

Les règles suivantes s'appliquent aux énumérations :

- Utilisez le Pascal Casing pour les types Enum et leurs valeurs.
- Préfixer le nom par « Et » (Enumerate Type)
- Utilisez les abréviations avec modération
- N'utilisez pas de suffixe de type Enum
- Quand cela est possible, si l'enum concerne une propriété ou une classe reprenait son nom dans le nom de l'enum en lui ajoutant le suffixe Type.

```
Ex : EtBorderType
```

Utilisez toujours l'attribut FlagsAttribute dans le cas d'une énumération binaire.

```
Ex :
[Flags]
enum EtBorderType
{
  Single = 1,
  Double = 2
}
```

8.2.5.6 Nommage des structures

Les règles suivantes s'appliquent aux structures :

- Utilisez le Pascal Casing pour les types Struct.
- Préfixer le nom par « St » (Structure Type)
- Utilisez les abréviations avec modération
- N'utilisez pas de suffixe de type Struct
- Pour les variables de la structure se reporter au paragraphe Nommage des variables
- Préférer l'utilisation de classe à la place des structures

L'exemple suivant décrit une structure (StExemple) constituée de 4 éléments :

```
  //------------------------------------------------------------------------
  /// <summary>
  /// Exemple de structure
  /// </summary>
  //------------------------------------------------------------------------
  public Struct StExemple
  {
    public  int    iValeur,
    public  bool   bValeur,
    public  string strValeur,
    public  float  fValeur,
  };
```

8.2.5.7 Nommage des champs statiques

Les règles suivantes s'appliquent pour le nommage des champs statiques :

- Utilisez le Camel Casing
- Il est recommandé d'utiliser des propriétés statiques à la place de champs statiques

8.2.5.8 Nommage des variables

8.2.5.8.1 Principe général

Les règles suivantes s'appliquent pour le nommage des paramètres :

- Utilisez le Camel Casing
- Utilisez des noms décrivant précisément le paramètre. Ceci est très utile lors de l'utilisation d'Intellisence dans Visual Studio.

Les règles de nommage des variables sont les suivantes :

Préfixe précisant le type de la donnée	Commentaire
i	données entières (int, long, signée, non signée …)
b	données booléennes
f	données réelles
str	données de type chaîne de caractères
ds	données de type DataSet
obj	données objet

A titre d'exemple :

```
int      iValeur;
bool     bValeur;
float    fValeur;
string   strValeur;
DataSet  dsValeur;
TraceLog objTraceLog;
```

8.2.5.8.2 Objets Ihm

Les règles de nommage concernant les objets de type IHM reposent sur le principe suivant :

- Utilisation d'un préfixe spécifique, caractérisant le type d'objet IHM utilisé
- Utilisez le Camel Casing

Sur l'exemple d'un objet de type TextBox qui permet de saisir un type d'organisme, on obtient :

A titre d'exemple :

```
protected System.Web.UI.WebControls.TextBox tbTypeOrganisme;
```

L'ordre d'énumération des types d'objet IHM correspond à celui de la boîte à outils utilisée dans l'environnement de développement de formulaire ASP.Net.

Type d'objet	Préfixe à placer avant le nom de l'élément
Label	lbl
TextBox	tb
Button	btn
LinkButton	lnkbtn
ImageButton	imgbtn
RadioButton	rdobtn
CheckBox	chkbx
HyperLink	hyplnk
DropDownList	ddl
DataGrid	dg
DataList	dl
DateTimePicker	dtp
ObjectDataSource	ods
Gridview	gv
ListBox	lb
RequieredFiledValidator	rfv

Pour les autres composants, la règle d'élaboration du préfixe est la suivante :

Préfixe en minuscule

Première lettre de chaque mot composé (Ex : **tb** pour **TextBox**) ou composition de 2 à 3 caractères maximum de chaque mot composé (Ex : **chkbx** pour **CheckBox**)

8.2.5.8.3 Les variables membres

Les variables membres d'une classe répondent à la même syntaxe que celle décrite au paragraphe précédent sauf qu'elles sont préfixées par « ***m***_ ».

A titre d'exemple :

```
//---------------------------------------------------------------------------
/// <summary>
/// Nom complet du fichier de trace (ex: "c:\Nom projet\log\Trace.log")
/// </summary>
//---------------------------------------------------------------------------
private string m_strFileName;
```

8.2.5.8.4 Les constantes

Les constantes répondent à la même syntaxe que celle décrite au paragraphe précédent sauf qu'elles sont préfixées par « ***c***_ ».

A titre d'exemple :

```
const int    c_iValeur   = 14;
const string c_strValeur = "Chaine";
```

8.2.5.9 Nommage des paramètres

Cf. Nommage des variables

8.2.5.10 Nommage des méthodes

Les règles suivantes s'appliquent pour le nommage des méthodes :

- Utilisez des verbes à l'infinitif pour décrire la fonctionnalité.
- Utilisez le Pascal Casing
- Utilisez une propriété si la méthode se contente de retourner une valeur sans paramètres d'entrée.
- Vous pouvez utiliser le franglais en utilisant les verbes du type : Get, Set, Update, Read, Delete, Commit … si cela a un intérêt et aide à la compréhension tout en limitant la longueur des noms.

8.2.5.11 Nommage des propriétés

Les règles suivantes s'appliquent pour le nommage des propriétés :

- Utilisez le Pascal Casing
- Essayez de nommer la propriété avec le même nom que le type renvoyé. Par exemple, si vous nommez une propriété Couleur, son type devra être nommé Couleur.

```
Exemple :
public class SampleClass
{
   public Color BackColor
   {
   }
}
```

L'exemple suivant montre une propriété avec le même nom que son type

```
public enum Color
{
   // Insert code for Enum here.
}
public class Control
{
   public Color Color
   {
      get {// Insert code here.}
      set {// Insert code here.}
   }
}
```

8.2.5.12 Nommage des événements

Utilisez les conventions de nommage suivantes :

Utilisez le Pascal Casing

Utilisez le suffixe EventHandler pour le nom des événements.

Spécifiez deux paramètres nommés sender et e. Sender représente l'objet qui a envoyé l'événement.

Le paramètre sender est toujours du type object même si il est possible d'utiliser un type plus spécifique.

L'état associé à l'événement est encapsulé dans une classe d'une classe EventArgs nommée e. Utilisez une classe appropriée pour e dérivée de EventArgs.

Cette classe devra être suffixée par EventArgs.

Utilisez des verbes au participe passé pour nommer des événements.

Ex : Clôturé, fermé, sélectionné ou du franglais Clicked, Dropped…

En anglais, il est aisé de préciser le moment de l'événement dans le verbe car cela ne nécessite pas de caractères accentués. Par exemple, utilisez des verbes comme Closing pour indiquer que l'événement est en train de se produire ou Closed pour indiquer qu'il est terminé.

N'utilisez pas des préfixes dans le nom de l'événement. Par exemple, utilisez Close au lieu de OnClose.

En général, vous devez fournir une méthode virtuelle protégée nommé OnXxx sur les types générant l'événement. Cette méthode ne doit contenir qu'un paramètre événement e car le sender est toujours l'instance en cours.

Exemple de nommage d'un événement.

```
public delegate void MouseEventHandler(object sender, MouseEventArgs e);
```

Exemple de nommage d'un argument d'événement

```
public class MouseEventArgs : EventArgs
{
   int x;
   int y;
   public MouseEventArgs(int x, int y)
      { this.x = x; this.y = y; }
   public int X { get { return x; } }
   public int Y { get { return y; } }
}
```

8.2.6 Fichiers

8.2.6.1 Fichiers C#

Principe général

Sauf **contrainte**, un fichier C# contient une et une seule classe.

Il porte le même nom que la classe qu'il contient. A titre d'exemple :

```
La classe UtilTraceLog est codée dans le fichier UtilTraceLog.cs
```

Utilise le Pascal Casing

Règle de nommage des composants d'un client riche

Type d'objet	Préfixe à placer avant le nom de l'élément
Boite de dialogue	Dlg
Fenêtre (*)	Win
Contrôle	Ctrl
BindingSource	bs
GroupBox	gb
TabControl / TabPage	tac / tap
DataGridView	dgv

(*) Ou **Wnd** dans les anciennes applications voire pour l'application GPEA **Frm**

8.2.6.2 Fichiers xml

Les fichiers XML utilisent le Pascal Casing pour le nom des éléments et le Camel Casing pour les attributs.

8.3 Commentaires à appliquer aux objets d'un fichier C#

8.3.1 Préambule

L'objet principal d'un commentaire est d'ajouter au sein du code toutes informations permettant de rendre plus lisible ce dernier et de préciser éventuellement des limitations ou pré requis.

Il existe deux types de commentaires :

- Commentaires libres (généralement placés au milieu du code),
- Commentaires imposés (Entête de fichier, de classe, de fonctions,…).

8.3.2 Syntaxe

Pour mémoire, la syntaxe pour écrire un commentaire en C# est la suivante :

```
// Commentaire sur une ligne
```

Ou

```
/*
Commentaire
Sur plusieurs lignes
*/
```

Par convention, on utilisera la première méthode pour les commentaires et on utilisera la seconde pour supprimer temporairement une partie du code en phase de développement.

A titre indicatif, l'outil de développement met à disposition dans la barre des commandes une fonction qui place ou supprime une partie de code en commentaire.

8.3.3 Outil de génération de documentation technique

Les commentaires imposés (Espace de noms, classes, méthodes et variables membres ou tout autre objet exporté) doivent utiliser des tags Xml permettant ainsi la génération automatique de documentation technique type « JavaDoc ».

Dans ce cas de figure, les commentaires sont :

- encadrés par des tags,
- commence par un triple /

Exemple :

```
    //------------------------------------------------------------------------
    /// <summary>
    /// Commentaire généré dans la documentation technique
    /// </summary>
    //------------------------------------------------------------------------
```

Pour plus d'information sur les tags Xml, le lecteur se référera à la documentation du MSDN de Microsoft.

8.3.4 Commentaires

Un fichier source C# contient le code relatif à une classe. Ce fichier, généralement généré automatiquement à l'aide de l'assistant de l'outil, a la structure suivante :

```
using xxx
namespace yyy
{
```

```
  class zzz
  {
    Liste des méthodes et variables membres de la classe
  }
}
```

8.3.4.1 Déclarations

La zone de déclaration permet de spécifier les librairies avec lesquelles se lie la classe. Exemple :

```
using System;
```

Dans le cas d'utilisation de librairies externes plutôt que celles livrées en standard dans l'environnement de développement, le développeur commentera l'utilité de cette librairie et éventuellement les pré-requis associés. Exemple :

```
using Oracle.DataAccess.Client; // Nécessite d'avoir installé la librairie Oracle DataProvider
for .NET
using Oracle.DataAccess.Types;  // Nécessite d'avoir installé la librairie Oracle DataProvider
for .NET
```

8.3.4.2 L'espace de nom

L'espace de nom permet de définir la déclaration et portée d'utilisation d'une librairie. Cette définition est généralement commune à une librairie et donc un à ensemble de classes.

Le cartouche de commentaires donné ci-dessous met en place les tags Xml.

Il permet de suivre :

- la version du composant,
- ses évolutions
- un descriptif général de ses fonctionnalités.

```
//-----------------------------------------------------------------------------
/// <summary>
/// Espace de nom       : CRPaca.SocleTechnique.Util
/// Logiciel            : Socle Technique CR Paca
/// Version du composant: 1.00
/// Modification:
///
/// Description:
/// Utilitaires du logiciel:
///   Accès à une clé de la base de registre
///   Trace log
///   Envoi d'eMail
///   Paramètrage applicatif au travers d'un fichier Xml
///   Le log d'évènement dans l'OS
/// </summary>
//-----------------------------------------------------------------------------
namespace CRPaca.SocleTechnique.Util
{
  ...
}
```

8.3.4.3 La classe

La classe permet de définir l'ensemble des méthodes, variables… relatives à l'objet codé.

Le cartouche de commentaires donné ci-dessous met en place les tags Xml.

Il permet de préciser :

- le nom,
- un exemple d'utilisation de la classe,
- les remarques ou complément associés à l'objet.

```
  //---------------------------------------------------------------------------
  /// <summary>
  /// Classe permettant la gestion des traces
  /// </summary>
  /// <example>Exemple d'utilisation
  /// <code>
  /// void MyFunction(void)
  /// {
  ///   UtilTraceLog  traceLog = UtilTraceLog.OpenLog("Nom Application", "Trace");
  ///   traceLog.WriteLine("{0} {1} {2}", "Coucou", 10, 12.18);
  ///   DataSet ds;
  ///   traceLog.TraceTable(ds.Tables["TABLE"]);
  ///   ...
  /// }
  /// </code>
  /// </example>
  /// <remarks>
  /// Le paramètrage des traces s'effectue en base de registre [HKEY_LOCAL_MACHINE\SOFTWARE\CR
Paca\Nom projet\TraceLog] dans les clés:
  ///   RepTraceLog   : Répertoire de la machine contenant l'ensemble des fichiers log (ex
"C:\log")
  ///   IsTraceActive : Flag permettant d'activé (="1") ou non (="0") les traces
  ///   IsDebugActive : Flag permettant d'activé (="1") ou non (="0") les traces dans la
console de debug de Visual Studio. Cette option a sens qu'en mode de compilation Debug et si
IsTraceActive="1"
  /// </remarks>
  /// <remarks>
  /// Exemple d'utilisation
  /// <code>
  ///   UtilTraceLog  traceLog = UtilTraceLog.OpenLog("Nom Application", "Trace");
  ///   traceLog.WriteLine("{0} {1} {2}", "Coucou", 10, 12.18);
  ///   DataSet ds;
  ///   traceLog.TraceTable(ds.Tables["TABLE"]);
  /// </code>
  /// </remarks>
  //---------------------------------------------------------------------------
  public class UtilTraceLog
  {
    ...
  }
```

8.3.4.4 Les méthodes d'une classe

Les méthodes contiennent le code élémentaire d'une fonction d'une classe.

Le cartouche de commentaires donné ci-dessous met en place les tags Xml.

Il permet de préciser :

- la fonctionnalité réalisée par la méthode,
- l'ensemble des paramètres d'entrée et de sortie.

```
    //---------------------------------------------------------------------------
    /// <summary>
    /// Retourne le libellé de l'erreur spécifiée
    /// </summary>
    /// <param name="numErr">Numéro d'erreur</param>
    /// <returns>Libellé de l'erreur</returns>
    //---------------------------------------------------------------------------
    public string GetError(Et_err numErr)
    {
      // Code
    }
```

8.3.4.5 Les objets exportés d'une classe

Les objets exportés d'une classe sont :

- Les variables membres,
- les structures,
- les énumérations,

Le cartouche de commentaires donné ci-dessous met en place les tags Xml. Il permet de préciser la nature de l'objet.

```
//---------------------------------------------------------------------------
/// <summary>
/// Nom complet du fichier de trace (ex: "c:\Nom projet\log\Trace.log")
/// </summary>
//---------------------------------------------------------------------------
private string m_strFileName;
```

8.3.4.6 Ressources

Ce sont les ressources localisables telles que les messages d'erreur et le texte de menu.

- Il est préférable de donner des identificateurs descriptifs plutôt que courts, d'être concis autant que possible mais veillez à préserver la lisibilité.
- Ne pas utiliser de mots clés spécifiques aux langages de programmation du Common Language Runtime (CLR).
- Utiliser uniquement des caractères alphanumériques et des traits de soulignement dans les noms des ressources.
- Utiliser le point (« . ») comme séparateur pour imbriquer des identificateurs dans une hiérarchie claire.

  Menus.FileMenu.Close.Text et Menus.FileMenu.Close.Color respectent cette règle.
- Utilisez la convention d'affectation de noms suivante pour les ressources de messages d'exception. L'identificateur de ressource doit représente le nom de type d'exception suivi d'un point puis d'un identificateur synthétique de l'exception.

  ArgumentException.BadEnumValue respecte cette règle.

8.3.4.7 Types génériques

Les génériques constituent une grande nouveauté du .NET Framework version 2.0.Les règles suivantes permettent de sélectionner des noms corrects pour les paramètres de type générique.

- Attribuez aux paramètres de type générique des noms descriptifs, sauf si une seule lettre est suffisamment explicative et si l'ajout d'un nom n'apporte rien.

  IDictionary<TKey, TValue> est un exemple d'interface qui respecte cette règle.
- Envisagez d'utiliser la lettre T comme nom de paramètre de type pour les types possédant un paramètre de type composé d'une seule lettre.
- Ajoutez la lettre T comme préfixe du paramètre de type descriptif.
- Envisagez de signaler les contraintes placées sur un paramètre de type dans le nom de paramètre. Par exemple, un paramètre limité à ISession peut être appelé TSession.

9 DOCUMENT 3 – 2.00 – C# - REGLES DE NOMMAGE ET RECOMMANDATIONS

9.1 Introduction

Le présent document constitue les « Règles de nommages et les Recommandations » que devront utiliser les développeurs de la Région PACA et les prestataires de service développant pour la Région PACA, dans le cadre de leurs projets utilisant le langage C# au travers de l'outil de développement Visual Studio .NET

Ces Règles et Recommandations visent à améliorer la lisibilité et permettent la mise en œuvre commune d'une charte accessible et utilisée par l'ensemble d'une équipe de développement.

Les points abordés dans ce document concernent :

- Les commentaires,
- les règles de nommage des classes, variables, fonctions, structures…,
- les règles de nommage des objets des ihm.

Des particularités seront précisées dans le cas du code généré automatiquement par l'outil de développement (utilisation des Wizard).

L'objet de ce document n'est pas de former un développeur au langage C# et à l'outil Visual Studio .Net. Les connaissances minimales sont considérées acquises avant la lecture et assimilation de ces Règles.

Les règles de nommage sont basées sur le guideline de Microsoft.

9.2 règles de nommage

9.2.1 Principes généraux

Le nom des objets et des méthodes sont le plus descriptifs possible.

Ils ne doivent pas dépasser 35 caractères.

Pas d'underscore (sauf pour préfixer), pas de tiret, pas de caractères accentués.

Ne pas ajouter de mots qui n'apportent pas de valeur ajoutée. Exemple Employe.NomEmploye doit devenir Employe.Nom

Les abréviations peuvent être utilisées si ces dernières sont connues de tous ou facilement identifiables. Par exemple, Client.ID est acceptable à la place de Client.Identifiant ou Client.Identite.

9.2.2 Capitalisation

Utilisez les conventions suivantes pour la capitalisation des identifiants.

9.2.2.1 Pascal Casing

La 1ère lettre de chaque mot est en majuscule, les autres en minuscules.

```
Ex : BackColor
```

9.2.2.2 Camel Casing

La 1ère lettre du 1er mot est en minuscule et les 1ères lettres des mots suivants sont en majuscules.

```
Ex : backColor
```

9.2.2.3 Majuscule

Toutes les lettres sont en majuscules. Utilisez cette convention pour les identifiants à deux ou trois lettres.

```
Ex : IO
```

La table suivante résume l'utilisation des conventions d'écriture pour les types et les identifiants.

Identifiant	Casse	Exemple
Class	Pascal	AppDomain
Enum type	Pascal	ErrorLevel
Enum values	Camel	fatalError
Event	Pascal	ValueChange
Exception class	Pascal	WebException Note : Se termine toujours par Exception.
Interface	Pascal	IDisposable Note : Commence toujours par la lettre I.
Method	Pascal	ToString
Namespace	Pascal	System.Drawing
Parameter	Camel	typeName
Property	Pascal	BackColor
Read-only Static field	Camel	redValue
Protected instance field	Camel	redValue Note : Rarement utilisée. Une propriété est préférable.
Public instance field	Camel	redValue Note : Rare

9.2.3 Casse

Pour éviter des confusions et garantir l'interopérabilité entre les langages, utilisez ces règles concernant l'utilisation de la sensibilité de la casse :

- N'employez pas des identifiants dont le nom est dépendent de la casse. Ne distinguez pas des identifiants uniquement par leurs casses.

```
Ex : arguments et Arguments
```

Bien que csharp soit sensible à la casse, cela reste une source d'erreur et n'est pas compatible avec l'utilisation d'un autre langage non sensible à la casse tel que VB.

9.2.4 Abréviations

N'utilisez pas des abréviations ou des contractions de terme comme tout ou partie d'un identifiant.

```
Ex : Utilisez GetWindow au lieu de GetWin
```

Quand cela est approprié, utilisez des acronymes pour remplacer des termes ou des phrases longues.

```
Ex : UI pour User Interface
```

Quand vous utilisez des acronymes, utilisez le Pascal Casing. S'il fait moins de 4 caractères, utilisez les majuscules.

Voici une liste de quelques abréviations couramment utilisées.

Abréviations	Commentaire
CRPaca	
UI	User Interface
IO	Input/Output
DAL	Data Access Layer (Note : Couche d'accès aux données)

9.2.5 Convention de nommage

9.2.5.1 Nommage des namespaces

La convention de nommages des namespaces peut se résumer ainsi :

CRPaca.[.Application][.Domaine]

```
Ex : CRPaca.SocleTechnique.Util
```

CRPaca correspond au nom de la société qui produit le namespace à aujourd'hui il n'y a pas d'ambiguïté.

Utilisez le Pascal Casing pour nommer les namespaces et le point comme séparateur.

Utilisez le pluriel si cela est sémantiquement approprié. Par exemple, System.Collections plutôt que System.Collection. Cette règle ne s'applique pas pour les abréviations.

Evitez les noms de namespace identiques à ceux du framework car il y aura conflit lors de l'utilisation d'intellisence dans Visual Studio.

N'utilisez pas le même nom pour un namespace et une classe.

Il n'y a pas de lien entre le namespace et une assembly mais dans la mesure du possible, on essayera de conserver une cohérence.

Nom de domaine prédéfini :

Domaine	Commentaire
Util	Utilitaire
Communs	Framework commun
Intranet	Outils collaboratifs

9.2.5.2 Nommage des classes

Les règles suivantes s'appliquent pour le nommage des classes :

- Utilisez le Pascal Casing
- Utilisez les abréviations avec modération
- N'utilisez pas de préfixe tel que C ou Cls pour nommer une classe
- N'utilisez pas le _ dans le nom de la classe (cela est valable pour tous les identifiants)
- De temps en temps, il est nécessaire de fournir un nom de classe qui commence par la lettre I, bien que la classe ne soit pas une interface. Ceci est possible aussi longtemps que I est la première lettre d'un mot entier qui est une partie du nom de la classe. Par exemple, le nom IdentityStore de classe est approprié
- Le cas échéant, employez un mot composé pour nommer une classe dérivée. La deuxième partie du nom de la classe dérivée devrait être le nom de la classe de base. Par exemple, ApplicationException est un nom approprié pour une classe dérivée d'une classe appelée Exception, parce qu'ApplicationException est du genre Exception.
- Cette règle n'est pas exclusive et doit être employée raisonnablement. Par exemple, Bouton est un nom approprié pour une classe dérivée de Control. Bien qu'un bouton soit dérivé de Control, l'utilisation de Control dans le nom de la classe rallongerait celui-ci inutilement.

Remarque :

Ces règles concernent uniquement les classes développées par le CR PACA ou les prestataires de service chargés des développements. L'outil de développement Visual Studio, au travers de ses assistants (Wizard), génère automatiquement (Notamment pour les aspects IHM) des classes dont les règles de nommage reposent alors sur celles de l'outil.

9.2.5.3 Nommage des interfaces

Les règles suivantes s'appliquent pour le nommage des classes :

- Utilisez des noms ou des phrases condensés, ou des adjectifs qui décrivent une fonctionnalité. Par exemple, IPersistable utilise un adjectif, IContainer un nom
- Utilisez le Pascal Casing
- Utilisez les abréviations avec modération
- Préfixez le nom de l'interface avec la lettre I
- N'utilisez pas le _ dans le nom des interfaces.

```
Ex : IContainer
```

9.2.5.4 Nommage des attributs

Un attribut permet d'intégrer des métas donnés dans un programme. Un attribut est d'abord une classe et doit se conformer aux règles de nommages d'une classe.

Un nom d'attribut sera TOUJOURS suffixé par le nom Attribute

```
Ex : ObsoleteAttribute
```

9.2.5.5 Nommage des énumérations

Les énumérations (Enum) héritent de la classe Enum. A ce titre, elles doivent être déclarées comme une classe.

Les règles suivantes s'appliquent aux énumérations :

- Utilisez le Pascal Casing pour les types Enum et leurs valeurs.
- Préfixer le nom par « Et » (Enumerate Type)
- Utilisez les abréviations avec modération
- N'utilisez pas de suffixe de type Enum
- Quand cela est possible, si l'enum concerne une propriété ou une classe reprenait son nom dans le nom de l'enum en lui ajoutant le suffixe Type.

```
Ex : EtBorderType
```

Utilisez toujours l'attribut FlagsAttribute dans le cas d'une énumération binaire.

```
Ex :
[Flags]
enum EtBorderType
{
  Single = 1,
  Double = 2
}
```

9.2.5.6 Nommage des structures

Les règles suivantes s'appliquent aux structures :

- Utilisez le Pascal Casing pour les types Struct.
- Préfixer le nom par « St » (Structure Type)
- Utilisez les abréviations avec modération
- N'utilisez pas de suffixe de type Struct
- Pour les variables de la structure se reporter au paragraphe Nommage des variables
- Préférer l'utilisation de classe à la place des structures

L'exemple suivant décrit une structure (StExemple) constituée de 4 éléments :

```
  //------------------------------------------------------------------------
  /// <summary>
  /// Exemple de structure
  /// </summary>
  //------------------------------------------------------------------------
  public Struct StExemple
  {
    public  int    iValeur,
    public  bool   bValeur,
    public  string strValeur,
    public  float  fValeur,
  };
```

9.2.5.7 Nommage des champs statiques

Les règles suivantes s'appliquent pour le nommage des champs statiques :

- Utilisez le Camel Casing
- Il est recommandé d'utiliser des propriétés statiques à la place de champs statiques

9.2.5.8 Nommage des variables

9.2.5.8.1 Principe général

Les règles suivantes s'appliquent pour le nommage des paramètres :

- Utilisez le Camel Casing
- Utilisez des noms décrivant précisément le paramètre. Ceci est très utile lors de l'utilisation d'Intellisence dans Visual Studio.

Les règles de nommage des variables sont les suivantes :

Préfixe précisant le type de la donnée	Commentaire
i	données entières (int, long, signée, non signée …)
b	données booléennes
f	données réelles
str	données de type chaîne de caractères
ds	données de type DataSet
obj	données objet

A titre d'exemple :

```
int      iValeur;
bool     bValeur;
float    fValeur;
string   strValeur;
DataSet  dsValeur;
TraceLog objTraceLog;
```

9.2.5.8.2 Objets Ihm

Les règles de nommage concernant les objets de type IHM reposent sur le principe suivant :

- Utilisation d'un préfixe spécifique, caractérisant le type d'objet IHM utilisé
- Utilisez le Camel Casing

Sur l'exemple d'un objet de type TextBox qui permet de saisir un type d'organisme, on obtient :

A titre d'exemple :

```
protected System.Web.UI.WebControls.TextBox tbTypeOrganisme;
```

L'ordre d'énumération des types d'objet IHM correspond à celui de la boîte à outils utilisée dans l'environnement de développement de formulaire ASP.Net.

Type d'objet	Préfixe à placer avant le nom de l'élément
Label	lbl
TextBox	tb
Button	btn
LinkButton	lnkbtn
ImageButton	imgbtn
RadioButton	rdobtn
CheckBox	chkbx
HyperLink	hyplnk
DropDownList	ddl
DataGrid	dg
DataList	dl
DateTimePicker	dtp
ObjectDataSource	ods
Gridview	gv
ListBox	lb
RequieredFiledValidator	Rfv

Pour les autres composants, la règle d'élaboration du préfixe est la suivante :

Préfixe en minuscule

Première lettre de chaque mot composé (Ex : **tb** pour **TextBox**) ou composition de 2 à 3 caractères maximum de chaque mot composé (Ex : **chkbx** pour **CheckBox**)

9.2.5.8.3 Les variables membres

Les variables membres d'une classe répondent à la même syntaxe que celle décrite au paragraphe précédent sauf qu'elles sont préfixées par « ***m***_ ».

A titre d'exemple :

```
    //---------------------------------------------------------------------------
    /// <summary>
    /// Nom complet du fichier de trace (ex: "c:\Nom projet\log\Trace.log")
    /// </summary>
    //---------------------------------------------------------------------------
    private string m_strFileName;
```

9.2.5.8.4 Les constantes

Les constantes répondent à la même syntaxe que celle décrite au paragraphe précédent sauf qu'elles sont préfixées par « ***c***_ ».

A titre d'exemple :

```
    const int    c_iValeur   = 14;
    const string c_strValeur = "Chaine";
```

9.2.5.9 Nommage des paramètres

Cf. Nommage des variables

9.2.5.10 Nommage des méthodes

Les règles suivantes s'appliquent pour le nommage des méthodes :

- Utilisez des verbes à l'infinitif pour décrire la fonctionnalité.
- Utilisez le Pascal Casing
- Utilisez une propriété si la méthode se contente de retourner une valeur sans paramètres d'entrée.
- Vous pouvez utiliser le franglais en utilisant les verbes du type : Get, Set, Update, Read, Delete, Commit … si cela a un intérêt et aide à la compréhension tout en limitant la longueur des noms.

9.2.5.11 Nommage des propriétés

Les règles suivantes s'appliquent pour le nommage des propriétés :

- Utilisez le Pascal Casing
- Essayez de nommer la propriété avec le même nom que le type renvoyé. Par exemple, si vous nommez une propriété Couleur, son type devra être nommé Couleur.

```
Exemple :
public class SampleClass
{
   public Color BackColor
   {
   }
}
```

L'exemple suivant montre une propriété avec le même nom que son type

```
public enum Color
{
   // Insert code for Enum here.
}
public class Control
{
   public Color Color
   {
      get {// Insert code here.}
      set {// Insert code here.}
   }
}
```

9.2.5.12 Nommage des événements

Utilisez les conventions de nommage suivantes :

Utilisez le Pascal Casing

Utilisez le suffixe EventHandler pour le nom des événements.

Spécifiez deux paramètres nommés sender et e. Sender représente l'objet qui a envoyé l'événement.

Le paramètre sender est toujours du type object même si il est possible d'utiliser un type plus spécifique.

L'état associé à l'événement est encapsulé dans une classe d'une classe EventArgs nommée e. Utilisez une classe appropriée pour e dérivée de EventArgs.

Cette classe devra être suffixée par EventArgs.

Utilisez des verbes au participe passé pour nommer des événements.

Ex : Clôturé, fermé, sélectionné ou du franglais Clicked, Dropped…

En anglais, il est aisé de préciser le moment de l'événement dans le verbe car cela ne nécessite pas de caractères accentués. Par exemple, utilisez des verbes comme Closing pour indiquer que l'événement est en train de se produire ou Closed pour indiquer qu'il est terminé.

N'utilisez pas des préfixes dans le nom de l'événement. Par exemple, utilisez Close au lieu de OnClose.

En général, vous devez fournir une méthode virtuelle protégée nommé OnXxx sur les types générant l'événement. Cette méthode ne doit contenir qu'un paramètre événement e car le sender est toujours l'instance en cours.

Exemple de nommage d'un événement.

```
public delegate void MouseEventHandler(object sender, MouseEventArgs e);
```

Exemple de nommage d'un argument d'événement

```
public class MouseEventArgs : EventArgs
{
   int x;
   int y;
   public MouseEventArgs(int x, int y)
      { this.x = x; this.y = y; }
   public int X { get { return x; } }
   public int Y { get { return y; } }
}
```

9.2.5.13 Nommage des nouvelles classes et composants WPF

Type d'objet	Règle sur le nom de l'élément
Vues	**Suffixé** de « View »
Vue-Model	Préfixé de « VM »
Règles de validation	**Suffixé** de « ValidationRule »
Converter	**Suffixé** de « Converter »

9.2.6 Fichiers

9.2.6.1 Fichiers C#

Principe général

Sauf **contrainte**, un fichier C# contient une et une seule classe.

Il porte le même nom que la classe qu'il contient. A titre d'exemple :

```
La classe UtilTraceLog est codée dans le fichier UtilTraceLog.cs
```

Utilise le Pascal Casing

Règle de nommage des composants d'un client riche

En WPF, la plupart des composants, s'ils ont une fonction purement graphique, n'ont pas besoin de posséder d'identifiant (x :Name). Si malgré tout il est nécessaire d'en affecter un (comme par exemple pour un Datagrid ou un Textbox) alors les règles de nommage préconisées seront les suivantes :

Type d'objet	Préfixe à placer avant le nom de l'élément
Boite de dialogue	Dlg
GroupBox	gb
TabControl / TabPage	tac / tap
DataGrid	dg
Canva	cnv
Expander	expd
TextBlock	text
TextBox	tb
Grid	grd

9.2.6.2 Fichiers xml

Les fichiers XML utilisent le Pascal Casing pour le nom des éléments et le Camel Casing pour les attributs.

9.3 Commentaires à appliquer aux objets d'un fichier C#

9.3.1 Préambule

L'objet principal d'un commentaire est d'ajouter au sein du code toutes informations permettant de rendre plus lisible ce dernier et de préciser éventuellement des limitations ou pré requis.

Il existe deux types de commentaires :

- Commentaires libres (généralement placés au milieu du code),
- Commentaires imposés (Entête de fichier, de classe, de fonctions,…).

9.3.2 Syntaxe

Pour mémoire, la syntaxe pour écrire un commentaire en C# est la suivante :

```
// Commentaire sur une ligne
```

Ou

```
/*
Commentaire
Sur plusieurs lignes
*/
```

Par convention, on utilisera la première méthode pour les commentaires et on utilisera la seconde pour supprimer temporairement une partie du code en phase de développement.

A titre indicatif, l'outil de développement met à disposition dans la barre des commandes une fonction qui place ou supprime une partie de code en commentaire.

9.3.3 Outil de génération de documentation technique

Les commentaires imposés (Espace de noms, classes, méthodes et variables membres ou tout autre objet exporté) doivent utiliser des tags Xml permettant ainsi la génération automatique de documentation technique type « JavaDoc ».

Dans ce cas de figure, les commentaires sont :

- encadrés par des tags,
- commence par un triple /

Exemple :

```
    //---------------------------------------------------------------------------
    /// <summary>
    /// Commentaire généré dans la documentation technique
    /// </summary>
    //---------------------------------------------------------------------------
```

Pour plus d'information sur les tags Xml, le lecteur se référera à la documentation du MSDN de MicroSoft.

9.3.4 Commentaires

Un fichier source C# contient le code relatif à une classe. Ce fichier, généralement généré automatiquement à l'aide de l'assistant de l'outil, a la structure suivante :

```
using xxx
namespace yyy
{
  class zzz
  {
    Liste des méthodes et variables membres de la classe
  }
}
```

9.3.4.1 Déclarations

La zone de déclaration permet de spécifier les librairies avec lesquelles se lie la classe. Exemple :

```
using System;
```

Dans le cas d'utilisation de librairies externes à celles livrées en standard dans l'environnement de développement, le développeur commentera l'utilité de cette librairie et éventuellement les pré-requis associés. Exemple :

```
using Oracle.DataAccess.Client; // Nécessite d'avoir installé la librairie Oracle DataProvider
for .NET
using Oracle.DataAccess.Types;  // Nécessite d'avoir installé la librairie Oracle DataProvider
for .NET
```

9.3.4.2 L'espace de nom

L'espace de nom permet de définir la déclaration et portée d'utilisation d'une librairie. Cette définition est généralement commune à une librairie et donc un à ensemble de classes.

Le cartouche de commentaires donné ci-dessous met en place les tags Xml.

Il permet de suivre :

- la version du composant,
- ses évolutions
- un descriptif général de ses fonctionnalités.

```
//---------------------------------------------------------------------------
/// <summary>
/// Espace de nom       : CRPaca.SocleTechnique.Util
/// Logiciel            : Socle Technique CR Paca
/// Version du composant: 1.00
/// Modification:
///
/// Description:
/// Utilitaires du logiciel:
///   Accès à une clé de la base de registre
///   Trace log
///   Envoi d'eMail
///   Paramètrage applicatif au travers d'un fichier Xml
///   Le log d'évènement dans l'OS
/// </summary>
//---------------------------------------------------------------------------
namespace CRPaca.SocleTechnique.Util
{
```

```
  ...
}
```

9.3.4.3 La classe

La classe permet de définir l'ensemble des méthodes, variables… relatives à l'objet codé.

Le cartouche de commentaires donné ci-dessous met en place les tags Xml.

Il permet de préciser :

- le nom,
- un exemple d'utilisation de la classe,
- les remarques ou complément associés à l'objet.

```
  //---------------------------------------------------------------------------
  /// <summary>
  /// Classe permettant la gestion des traces
  /// </summary>
  /// <example>Exemple d'utilisation
  /// <code>
  /// void MyFunction(void)
  /// {
  ///   UtilTraceLog  traceLog = UtilTraceLog.OpenLog("Nom Application", "Trace");
  ///   traceLog.WriteLine("{0} {1} {2}", "Coucou", 10, 12.18);
  ///   DataSet ds;
  ///   traceLog.TraceTable(ds.Tables["TABLE"]);
  ///   ...
  /// }
  /// </code>
  /// </example>
  /// <remarks>
  /// Le paramètrage des traces s'effectue en base de registre [HKEY_LOCAL_MACHINE\SOFTWARE\CR
Paca\Nom projet\TraceLog] dans les clés:
  ///   RepTraceLog   : Répertoire de la machine contenant l'ensemble des fichiers log (ex
"C:\log")
  ///   IsTraceActive : Flag permettant d'activé (="1") ou non (="0") les traces
  ///   IsDebugActive : Flag permettant d'activé (="1") ou non (="0") les traces dans la
console de debug de Visual Studio. Cette option a sens qu'en mode de compilation Debug et si
IsTraceActive="1"
  /// </remarks>
  /// <remarks>
  /// Exemple d'utilisation
  /// <code>
  ///   UtilTraceLog  traceLog = UtilTraceLog.OpenLog("Nom Application", "Trace");
  ///   traceLog.WriteLine("{0} {1} {2}", "Coucou", 10, 12.18);
  ///   DataSet ds;
  ///   traceLog.TraceTable(ds.Tables["TABLE"]);
  /// </code>
  /// </remarks>
  //---------------------------------------------------------------------------
  public class UtilTraceLog
  {
    ...
  }
```

9.3.4.4 Les méthodes d'une classe

Les méthodes contiennent le code élémentaire d'une fonction d'une classe.

Le cartouche de commentaires donné ci-dessous met en place les tags Xml.

Il permet de préciser :

- la fonctionnalité réalisée par la méthode,
- l'ensemble des paramètres d'entrée et de sortie.

```
    //---------------------------------------------------------------------------
    /// <summary>
    /// Retourne le libellé de l'erreur spécifiée
    /// </summary>
```

```
    /// <param name="numErr">Numéro d'erreur</param>
    /// <returns>Libellé de l'erreur</returns>
    //---------------------------------------------------------------------------
    public string GetError(Et_err numErr)
    {
      // Code
    }
```

9.3.4.5 Les objets exportés d'une classe

Les objets exportés d'une classe sont :

- Les variables membres,
- les structures,
- les énumérations,

...

Le cartouche de commentaires présentée ci-dessous met en place les tags Xml. Il permet de préciser la nature de l'objet.

```
    //---------------------------------------------------------------------------
    /// <summary>
    /// Nom complet du fichier de trace (ex: "c:\Nom projet\log\Trace.log")
    /// </summary>
    //---------------------------------------------------------------------------
    private string m_strFileName;
```

9.3.4.6 Ressources

Ce sont les ressources localisables telles que les messages d'erreur et le texte de menu.

- Il est préférable de donner des identificateurs descriptifs plutôt que courts, d'être concis autant que possible mais veillez à préserver la lisibilité.
- Ne pas utiliser de mots clés spécifiques aux langages de programmation du Common Language Runtime (CLR).
- Utiliser uniquement des caractères alphanumériques et des traits de soulignement dans les noms des ressources.
- Utiliser le point (« . ») comme séparateur pour imbriquer des identificateurs dans une hiérarchie claire.

  Menus.FileMenu.Close.Text et Menus.FileMenu.Close.Color respectent cette règle.
- Utilisez la convention d'affectation de noms suivante pour les ressources de messages d'exception. L'identificateur de ressource doit représente le nom de type d'exception suivi d'un point puis d'un identificateur synthétique de l'exception.

  ArgumentException.BadEnumValue respecte cette règle.

9.3.4.7 Types génériques

Les génériques constituent une grande nouveauté du .NET Framework version 2.0.Les règles suivantes permettent de sélectionner des noms corrects pour les paramètres de type générique.

- Attribuez aux paramètres de type générique des noms descriptifs, sauf si une seule lettre est suffisamment explicative et si l'ajout d'un nom n'apporte rien.

  IDictionary<TKey, TValue> est un exemple d'interface qui respecte cette règle.
- Envisagez d'utiliser la lettre T comme nom de paramètre de type pour les types possédant un paramètre de type composé d'une seule lettre.
- Ajoutez la lettre T comme préfixe du paramètre de type descriptif.
- Envisagez de signaler les contraintes placées sur un paramètre de type dans le nom de paramètre. Par exemple, un paramètre limité à ISession peut être appelé TSession.

10 DOCUMENT 4 – 1.00 – ACCES AUX BASES DE DONNEES ORACLE

10.1 Introduction

Le présent document constitue les Règles et Recommandations de développement en PL/SQL ainsi que les normes à appliquer sur les objets Oracle que devront utiliser les développeurs de la Région PACA et les prestataires de service développant pour la Région PACA, dans le cadre de tous leurs projets.

Ces Règles et Recommandations visent à améliorer la lisibilité et permettent la mise en œuvre commune d'une charte accessible.

Au final, ce référentiel propose un certain nombre de conventions qui permettent de :

garantir une homogénéité de programmation si le développement intègre plusieurs programmeurs,

faciliter la phase d'intégration,

faciliter la phase de maintenance par l'homogénéité du code.

Les points abordés dans ce document concernent :

- La définition du modèle physique de données,
- Les normes à appliquer dans les packages PL/SQL et les procédures stockées,
- Les normes à appliquer concernant les tables, les références de clé étrangères, les index, les séquences,
- La définition des propriétaires des bases de données ORACLE, et leur utilisation
- Les spécificités du requêteur
- Les erreurs possibles

10.2 Le modèle physique de données

Le modèle physique de données est fait à l'aide de Visio for Enterprise Architect 10.0.5110

Le script de création des tables, index et références des clés étrangères sont également générés avec Visio.

10.3 Les règles syntaxiques générales

10.3.1 Les identifiants

Les règles syntaxiques générales qui s'appliquent sont les suivantes :

- Un nom d'identifiant commence nécessairement par une lettre.
- La longueur de l'identifiant ne peut dépasser 30 caractères.
- Les caractères qui peuvent être utilisés sont :
  - les lettres,
  - les chiffres,
  - les caractères $ (dollar) et _ (underscore).

Il faut tirer partie de ces règles relativement souples pour attribuer des noms explicites aux objets et sous-programmes. Les conventions de nommages sont précisées dans le paragraphe 4.

10.3.2 Majuscule et minuscule

En PL/SQL il n'y a pas de différenciation entre minuscules et majuscules.

Pour l'écriture des programmes, on prendra la convention suivante :

- les majuscules seront utilisées pour :
- les mots réservés du langage (PROCEDURE, FUNCTION, DECLARE, BEGIN, END, FOR, LOOP, IF...THEN...ELSE, EXCEPTION, WHEN, ...)

10.3.3 Les commentaires

Pl/SQL permet de mettre un commentaire soit ligne par ligne soit en bloc :

```
-- indique le début d'un commentaire qui se termine à la fin de la ligne
/* indique le début d'un commentaire multi-lignes
*/ indique la fin d'un commentaire multi-lignes
```

10.4 Convention de nommage

10.4.1 Déclaration des tables

10.4.1.1 Préambule

Etape 1 : s'assurer de l'existence du mot dans le glossaire

https://websat.cr-paca.fr/

User / Password: Identifiants AD

Etape 2 :

Si l'abrégé du mot existe alors utiliser celui-là, sinon demander la création de cet abrégé à la cellule _DSI_Production_Systemes et plus particulièrement A. Gon.

Exemple de nommage des colonnes:

Nature	**Abrégé**	**Exemple:**
Code	CODE	CODE_FONC_SERV Code fonction de service
Type	TYPE	TYPE_MONT_SUBV Type montant subventionné.
Date	DATE	DATE_FACT_EXPD Date expédition de la facture
Heure	HRE	HRE_RIGR_OFFR Heure rigueur offre
Top	TOP	TOP_EDIT_BORD_TRANS Bordereau facture édité O/N
Montants	MONT / MT	MT_TVA_LIGN_MAND Montant TVA ligne mandat MONT_GLOB_MARC Montant global marché
Numéro	NUM	NUM_OBJT Numéro Objectif NO_CRTF_PAIM Numéro certificat paiement
Coefficient	COEF	PCHARC_COEF_POND Coefficient de pondération
Libellé	LIBL	LIBL_RECA_ART Libellé récapitulatif Article
Quantité	QTTE	QTTE_IND Quantité indicateur
Taux	TX	TX_TVA_LIGN_FACT Taux de TVA pour la ligne facture
Années	ANEE	ANEE_ENGA Année engagement
Maximum	MAX	MAX_BORD_MAND Maximum bordereau de mandat
Nom	NOM	NOM_SIGN Nom signataire

Etape 3 :

Faire valider le script par la cellule _DSI_Production_Systemes.

Etape 4 :

Faire la modification de structure sur la base de développement.

10.4.1.2 Format de données

Code domaine	Format	Commentaire
Code	Varchar2(4)	Dans le cas où le code peut avoir une liste de valeur de type code-libellé il faut impérativement définir un domaine particulier. NB : Ce domaine sera un sous-ensemble de la table des tables code-libellé (CG_REF_CODES).
Types	Varchar2(4)	
Dates	Date	Format DD/MM/YYYY
Heures	Date	Format HH24 :MI
Top	Varchar2(1)	
Montant	Number(12,2)	Cas d'un montant en euros
Identifiant court	Number(7)	
Identifiant long	Number(10)	
Coefficient	Number(5,2)	
Libellé Court	Varchar2(40)	
Libellé long	Varchar2(80)	
Raison Sociale	Varchar2(128)	NB : compte tenu des possibilités de Forms, la raison sociale ne sera plus décomposée en quatre zones de 32 caractères mais sera une zone multiligne de 4 lignes sur 32 caractères de long
Quantité	Number(11)	
Taux	Number(5,2)	
Année	Number(4)	
Nom	Varchar2(32)	
Observation	Varchar2(300)	
Commentaire	Varchar2(300)	
Objets	Varchar2(300)	
Intitulé	Varchar2(128)	

10.4.1.3 Commentaires

Toute table, vue ou colonne devra avoir sa définition fonctionnelle mémorisée en clair dans le dictionnaire Oracle sur la base de production à minima.

10.4.1.4 Exemple de création d'une table

Toutes les colonnes seront préfixées du "short name" de la table sauf exception, les colonnes faisant références à une table mère (en gras dans le script ci-dessous) qui ne seront pas préfixées par le "short name" de la table mais garderont leur nom d'origine.

Exemple : **Table** CONVENTION **Short name** CONV **colonne** CONV_num_conv (numéro de la convention)

```
CREATE TABLE CONVENTION
        (CONV_NUM_CONV  NUMBER(6) NOT NULL
        ,CONV_DTCRE  DATE NOT NULL
        ,CONV_DTMOD  DATE NOT NULL
        ,CONV_USMOD  VARCHAR2(32) NOT NULL
        ,CONV_USCRE  VARCHAR2(32)
        ,CONV_DATE_EDIT  DATE
```

```
,CONV_DATE_DEB  DATE
,CONV_DATE_FIN  DATE
,CONV_DATE_RTOU_SAC  DATE
,CONV_DATE_RTOU_ORGA  DATE
,CONV_MONT  NUMBER(12,2)
,CONV_MONT_REVIS  NUMBER(12,2)
,TSESS_CODE  VARCHAR2(4) NOT NULL
,SESS_NUM  NUMBER(7) NOT NULL
,ORGA_CODE_SERV_INSTR  VARCHAR2(2) NOT NULL
,ORGA_ANEE_CONV  NUMBER(4) NOT NULL
,ORGA_NUM_SEQ_ORGA  NUMBER(3,0) NOT NULL
,ORGA_CODE_CONV  VARCHAR2(1) NOT NULL
,TFRM_NUM_SEQ  NUMBER(5)
)
```

10.4.2 Déclaration des clefs étrangères

Toute référence vers une table mère doit forcement donner lieu à la création d'une clé étrangère dans la table fille. Les clefs doivent être suffixées de « _**FK** ».

Norme: **Nom court de la table de référence _ nom court de la table référencée _FK**

Exemple : table CONVENTION

```
ALTER TABLE CONVENTION
    ADD CONSTRAINT  CONV_ORGA_FK FOREIGN KEY
                (ORGA_ANEE_CONV
                ,ORGA_NUM_SEQ_ORGA
                ,ORGA_CODE_CONV
                ,ORGA_CODE_SERV_INSTR) REFERENCES ORGA
                                (ORGA_ANEE_CONV
                                ,ORGA_NUM_SEQ_ORGA
                                ,ORGA_CODE_CONV
                                ,ORGA_CODE_SERV_INSTR)
    ADD CONSTRAINT CONV_TFRM_FK FOREIGN KEY
                (TFRM_NUM_SEQ) REFERENCES TYPE_FRMT
                                (TFRM_NUM_SEQ)
    ADD CONSTRAINT  CONV_SESS_FK FOREIGN KEY
                (TSESS_CODE
                ,SESS_NUM) REFERENCES SESS
                                (TSESS_CODE
                                ,SESS_NUM).
```

10.4.3 Déclaration des index

Chaque clef étrangère doit posséder un index. Les index doivent être suffixés de **« FK_I** ».

Exemple: Table Convention, 3 clefs étrangères, trois index.

1 / CREATE INDEX **CONV_SESS_FK_I** ON CONVENTION
(TSESS_CODE
,SESS_NUM)

2/ CREATE INDEX **CONV_ORGA_FK_I** ON CONVENTION
(ORGA_CODE_SERV_INSTR
,ORGA_ANEE_CONV
,ORGA_NUM_SEQ_ORGA
,ORGA_CODE_CONV)

3/ CREATE INDEX **CONV_TFRM_FK_I** ON CONVENTION
(TFRM_NUM_SEQ)

10.4.4 Déclaration des séquences

Les séquences auront le nom suivant:

SEQ_*nom de l'objet.*

Le synonyme associé répondra à la règle suivante :

SEQ_*nom_court objet_nom_court_appli*

10.4.5 Déclaration des packages

Le nom d'un package est défini de la manière suivante :

<Préfixe>_<Appli>_<Module>_[<Numéro_d'ordre>]

où

- <Préfixe> = PCKG
- < Appli> = Exemple : Nom court de l'application
- <Module> = Module du module
- <Numéro_d'ordre> = 001 , 002 ... [Optionnel]

Exemple : PCKG_DLYCVIE_UTILISATEUR

10.4.6 Déclaration des procédures stockées

Leur nom commence toujours par PROC_ suivi du nom : PROC_*Nom*. En outre le nommage des procédures stockées doit respecter une certaine norme suivant le type d'opérations effectuées.

La règle de nommage est la suivante :

<Préfixe>_<Ordre_SQL>_<Nom>_[<Numéro_d'ordre>]

où

<Préfixe> = PROC

<Ordre_SQL> :

Ordre SQL	<Ordre_SQL>
SELECT	sel
UPDATE	upd
DELETE	del
INSERT	ins
<DIVERS>	div

<Nom> : description du traitement effectué (doit faire référence à la table principale traitée par la procédure)

<Numéro_d'ordre> : 001, 002 ... [Optionnel]

Exemple : PROC_SEL_IDENTIFICATION_AD

10.4.7 Déclaration des fonctions

Le nom d'une fonction est défini de la manière suivante :

<Préfixe>_<Type_Retour>_<Nom>_[<Numéro_d'ordre>]

où

- <Préfixe> = FUNC
- <Type_retour> :

Type PL/SQL	<Type_Retour>
BOOLEAN	b
CHAR	c
DATE	d
NUMBER(X)	n
VARCHAR2(X)	v

- <Nom> : description du traitement effectué (doit faire référence à la table principale traitée par la fonction)
- <Numéro_d'ordre> = 001, 002 ... [Optionnel]

Exemple : FUNC_N_CALCUL_TAUX : N représente un nombre.

10.4.8 Déclaration des paramètres

Le nom d'un paramètre est défini de la manière suivante :

<Préfixe>_<Type>_<Mode>_<Nom>

où

<Préfixe> = p

<Type> =

Type PL/SQL	<Type >
BOOLEAN	b
CHAR	c
DATE	d
NUMBER(X)	n
VARCHAR2(X)	V
ROWTYPE	R
TABLE	t

<Mode> =

Parameter mode	<Mode>
IN	i
IN OUT	io
OUT	o

(Rappelons que tous les paramètres d'une fonction sont en mode IN).

- <Nom> = description du contenu

L'ordre des paramètres dans la déclaration d'une procédure sera le suivant :

- 1 Tous les paramètres IN.
- 2 Tous les paramètres IN OUT.
- 3 Tous les paramètres OUT.

Exemple de déclaration de procédure :

```
PROCEDURE proc_upd_client002 (p_n_i_id_cli   IN     NUMBER(2),
                              p_b_io_test    IN OUT BOOLEAN);
```

10.4.9 Déclaration des variables

Le nom d'une variable est défini de la manière suivante :

<Préfixe>_<Type>_<Nom>

où

<Préfixe> = VL pour les variables locales, VG pour les variables globales

<Type> =

Type PL/SQL	<Type >
BOOLEAN	b
CHAR	c
DATE	d
NUMBER(X)	n
VARCHAR2(X)	V
ROWTYPE	R
TABLE	t

<Nom> = description du contenu

Pour les variables faisant référence au dictionnaire de données, on privilégiera une déclaration de la forme :

nom_variable nom_table.nom_colonne%TYPE

ou

nom_variable nom_table%ROWTYPE

Les variables possédant une valeur initiale seront initialisées de préférence au moment de la déclaration.

```
VL_n_num  TAB1.num%type := 0 ;
VL_b_test BOOLEAN       := true ;
```

10.4.10 Déclarations des curseurs

Le nom d'un curseur est défini de la manière suivante :

<Préfixe>_<Nom>

où

<Préfixe> = C

<Nom> : description du traitement effectué (doit faire référence à la table principale traitée par la procédure)

10.4.11 Déclaration des exceptions

Le nom d'une exception est défini de la manière suivante :

<Préfixe>_<Nom>

où

<Préfixe> = E

<Nom> = description de l'erreur correspondante

Pour expliciter d'avantage l'erreur correspondante, un commentaire devra accompagner la déclaration de chaque exception.

Exemple :

```
-- exception levée lorsque le client n'existe pas
e_client_inexistant    exception ;
```

10.5 La gestion des erreurs

10.5.1 Les blocks d'exception

Chaque procédure devra comporter un block de traitement des exceptions qui va gérer toutes les erreurs utilisateurs (exceptions levées par RAISE ou erreurs déclenchées par RAISE_APPLICATION_ERROR).

Pour les erreurs plus génériques, des exceptions prédéfinies existent (ex : NO_DATA_FOUND, TOO_MANY_ROWS, VALUE_ERROR, ZERO_DIVIDE,...). Il faudra les gérer lorsque c'est nécessaire.

Tous les traitements d'exceptions devront se terminer par un cas WHEN OTHERS qui s'exécutera lorsque se produira une erreur autre que celles prévues explicitement. La réponse à ce cas pourra être :

exécution d'un ROLLBACK (si le traitement risque de rendre la base de données incohérente),

envoi d'un message d'information vers l'utilisateur (affiché à l'écran pour les traitements transactionnels, inséré dans une table pour les batchs).

Le message d'information devra nécessairement fournir le code et le libellé de l'erreur survenue. Ces éléments peuvent être obtenus par appels aux fonctions suivantes :

SQLCODE retourne un nombre qui correspond au code de la dernière exception qui a été levée,

SQLERRM retourne le libellé correspondant au code d'erreur passé en paramètre (donc SQLERRM(SQLCODE) renvoie le libellé de la dernière exception levée).

10.5.2 La gestion des erreurs de l'utilisateur

La gestion des erreurs de l'utilisateur peut se faire de deux manières :

en déclenchant une erreur utilisateur avec RAISE_APPLICATION_ERROR,

en levant des exceptions par RAISE.

10.5.2.1 La procédure RAISE_APPLICATION_ERROR

10.5.2.1.1 Description

La procédure RAISE_APPLICATION_ERROR du package DBMS_STANDARD permet de déclencher des erreurs définies par l'utilisateur lui-même :

Syntaxe :

raise_application_error (numéro_d'erreur, message_d'erreur)

Règles :

le numéro d'erreur doit être compris entre -20.000 et -20.999

le message d'erreur doit être explicite puisque c'est lui qui sera affiché

Afin de mieux localiser la cause de l'erreur, on concaténera le nom de la procédure au message d'erreur.

Exemple :

```
cv_nom_proc CONSTANT VARCHAR2(33) := 'sp_cn_600_init_batch : ' ;
...
IF(vn_num_client IS NULL) THEN
raise_application_error (-20245,
cv_nom_proc || 'Le n° de client doit être renseigné') ;
END IF;
```

Une erreur qui a été levée par la procédure RAISE_APPLICATION_ERROR peut être traitée dans un block EXCEPTION au même titre que les erreurs prédéfinies ORACLE.

La fonction SQLCODE renvoie le numéro de l'erreur. (ex : -20245)

La fonction SQLERRM renvoie la concaténation du numéro et du message de l'erreur (ex : 'ORA-20245:Le n° de client doit être renseigné')

L'expression suivante permet d'extraire le message d'erreur de SQLERRM :

```
SUBSTR(SQLERRM,INSTR(SQLERRM,':')+2)
```

10.5.2.1.2 Propagation des erreurs

On peut également propager une erreur levée par RAISE_APPLICATION_ERROR en refaisant appel à cette procédure dans le block EXCEPTION.

Dans ce cas, il faudra distinguer les erreurs utilisateur des erreurs système ORACLE.

Un block EXCEPTION pourra alors se présenter ainsi :

```
EXCEPTION
WHEN OTHERS THEN
-- Si c'est une erreur système, il faut lui concaténer le nom de la
-- procédure
IF (SQLCODE >  -20000) OR (SQLCODE < -20999) THEN
RAISE_APPLICATION_ERROR(-20999,vc_nom_proc || SQLERRM);
-- Si c'est une erreur utilisateur, le nom de la proc est déjà
-- concaténé au message
ELSE
RAISE_APPLICATION_ERROR(SQLCODE,
SUBSTR(SQLERRM,INSTR(SQLERRM,':')+2));
END IF;
END;
```

10.5.2.2 La levée d'exceptions

Pour chaque erreur spécifique qui peut survenir lors de l'exécution d'une procédure, le développeur peut déclarer une exception à laquelle correspondra un traitement à la fin du programme (dans la partie EXCEPTION).

Dans ce cas, toutes les procédures devront retourner un code d'erreur qui indiquera si le traitement s'est bien déroulé ou non.

Pour cela, on ajoutera à toutes les procédures un paramètre avec les caractéristiques suivantes :

nom : p_n_niverr

type : NUMBER(1)

mode : IN OUT de type

valeurs possibles:

valeur	signification
0	OK : le traitement s'est terminé avec succès
+1	une exception utilisateur a été levée
< 0	une exception système a été levée (= SQLCODE)

10.5.3 Préconisation pour la gestion des exceptions

De manière générale, il est préférable de se limiter aux exceptions générées par le système.

10.6 Structure d'un programme en PL/SQL (Procédure ou Fonction)

10.6.1 L'en-tête

Chaque procédure ou fonction comportera un en-tête de la forme suivante :

```
-- ------------------------------------------------
-- Nom : PROC_ALLER_AOF.SQL
-- ------------------------------------------------
-- Création
-- Auteur : Olivier REYRE (SOPRA)
-- Date : 15/02/2001
-- ------------------------------------------------
-- Modification(s)
-- Auteur : Christophe RUBINI (Sopra Group)
-- Date    : 25/02/2003
-- Description : Suppression des codes en dur sur les types
--                       de formation, dispositif, etc .....
-- ------------------------------------------------
-- Description :
--    Procédure Aller après l'Appel d'Offre sur Extranet
--    Récupération des projets saisis sur Extranet
-- ------------------------------------------------
-- Paramètres : Aucun
-- ------------------------------------------------
```

Dans la rubrique Modification, ne seront citées que les modifications majeures réalisées : Ces commentaires sont empilés, c'est-à-dire que les modifications les plus récentes se trouvent en tête de rubrique.

10.6.2 Séparations des différentes parties

Pour améliorer la lisibilité des programmes, on séparera par un titre en commentaire les différentes parties :

- en-tête
- déclaration des types
- déclaration des constantes
- déclaration des variables
- déclaration des curseurs
- déclaration des exceptions
- corps du programme
- traitement des exceptions

Pour les parties déclaratives, on adoptera une présentation tabulaire comme ci-dessous :

```
vn_num    NUMBER(1)  := 0;
vb_test   BOOLEAN    := true;
```

10.6.3 Documentation des programmes

Pour faciliter la maintenance ultérieure des programmes, ils devront comporter des commentaires pertinents en quantité suffisante. Il ne faut pas hésiter à insérer un long commentaire pour expliquer certains traitements complexes.

10.6.4 Indentation du code

Indenter le code de deux espaces à chaque nouveau bloc.

L'indentation unitaire sera faite au moyen de la touche Espace (2 x Espaces), ceci afin d'avoir une présentation homogène des programmes.

Veuillez trouver ci-dessous quelques exemples.

10.6.4.1 Requêtes

```
Select  X  (2 espaces entre le select et le X )
From   Y
Where  Z
   And  A = A
   And  B = B
```

10.6.4.2 Conditions

```
IF XXXX THEN
  yyyy  (2 espaces de décalage avec le IF )
ELSE
  zzzzz
END IF ;
```

10.6.5 Lisibilité

Afin d'améliorer la lisibilité des programmes, on adoptera une indentation claire des ordres SQL :

- une seule colonne par ligne dans la clause SELECT
- une seule table par ligne dans la clause FROM
- une seule condition par ligne dans la clause WHERE
- une seule colonne par ligne dans la clause ORDER BY

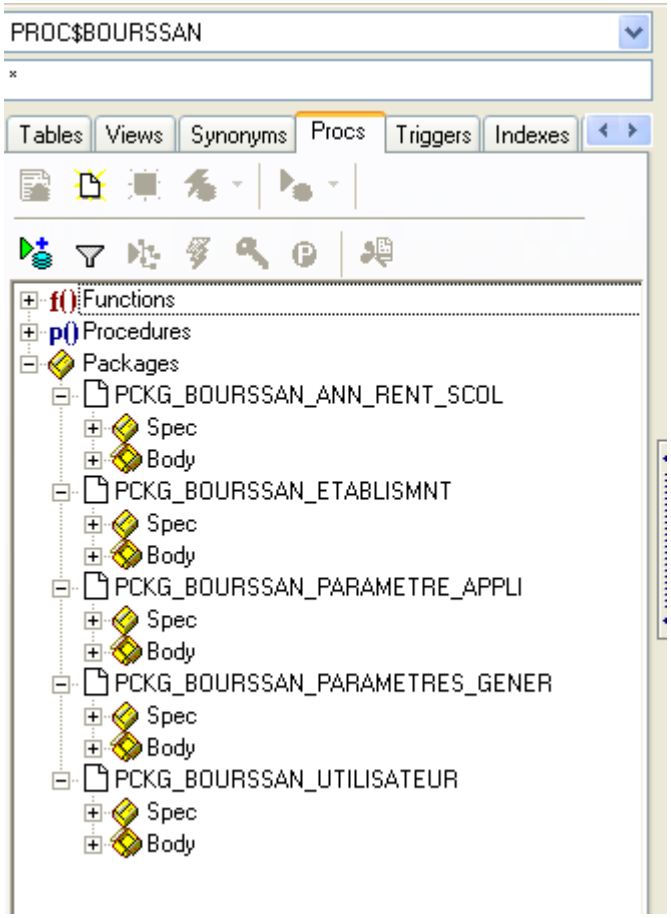
10.6.6 Alias sur sélection de données

Dans toutes les procédures de sélection il est nécessaire d'attribuer un alias à chacun des champs récupérer.

```
Ex :    SELECT  NUM_PROJET AS NUMERO_PROJET
        FROM  PROJET;
```

10.6.7 Les Packages : Partie Spec et Body

Un PACKAGE se divise en deux parties : la partie Spec et la partie Body.



10.6.7.1 La partie Spec

La partie Spec du PACKAGE contient la liste des procédures du package avec leurs paramètres associés.

Exemple de Spec avec 3 procédures dont une procédure avec 2 paramètres :

```
 CREATE OR REPLACE PACKAGE PCKG_PILOTAGE_REF AS
/******************************************************************************
   NAME:       PCKG_PILOTAGE_REF
   PURPOSE:
   REVISIONS:
   Ver        Date        Author           Description
   ---------  ----------  ---------------  ------------------------------------
   1.0        29/09/2005             1. Created this package.
******************************************************************************/
/* Types locaux au package*/
TYPE refcur IS REF CURSOR;

PROCEDURE PROC_SEL_SESSION
(
 CUR_o_data OUT PCKG_PILOTAGE_REF.refcur
);


PROCEDURE PROC_SEL_EXERCICE
(
 CUR_o_data OUT PCKG_PILOTAGE_.refcur
);


PROCEDURE PROC_SEL_COMMUNE_NOM_DEPT
(
   pv i Cod Dept IN VARCHAR2,
   pv_i_Lib_Commune IN VARCHAR2,
   CUR_o_data OUT PCKG_ PILOTAGE_REF.refcur
) ;
END PCKG_REF;
/
```

Le paramètre « refcur » est obligatoire lorsque la requête ramène un résultat (Select).

10.6.7.2 La partie Body

La partie Body du PACKAGE contient le code des procédures du package.

Exemple de Body avec 3 procédures dont une procédure avec 2 paramètres :

```
CREATE OR REPLACE PACKAGE BODY PCKG_PILOTAGE_REF AS
/******************************************************************************
   NAME:       PCKG_PILOTAGE_REF
   PURPOSE:

   REVISIONS:
   Ver        Date        Author           Description
   ---------  ----------  ---------------  ------------------------------------
   1.0        29/09/2005             1. Created this package.
******************************************************************************/
-------------------------------------------------------------
-- Nom : PROC_SEL_SESSION
-- -----------------------------------------------------------
-- Création : Evolution CF DEMANDE GEDEM N 4444
-- -------------------------------
-- Auteur : rampon-c
-- Date : 30/09/2010
-- -----------------------------------------------------------
-- Description :
-- --------------------
-- selection les differents éléments d'une session de pilotage
-- -----------------------------------------------------------
-- Paramètres :    CUR o data Curseur de sortie
-- -----------------------------------------------------------
PROCEDURE PROC_SEL_SESSION
(
 CUR_o_data OUT PCKG_PILOTAGE_REF.refcur
)
IS
BEGIN
OPEN CUR_o_data FOR
       SELECT SESS_CODE AS SESSION_CODE
       ,SESS_LIBL AS SESSION_LIBELLE,
       ,SESS_DATE AS SESSION_DATE
       ,SESS_TYPE AS SESSION_TYPE
       FROM sess_pilotage
       ORDER BY dt_session DESC;
EXCEPTION
  WHEN  NO_DATA_FOUND THEN
    RETURN;
 WHEN OTHERS THEN
    RAISE;
END PROC_SEL_SESSION;
```

```
-------------------------------------------------------------
-- Nom : PROC_SEL_EXERCICE
-- -----------------------------------------------------------
-- Création : Evolution CF DEMANDE GEDEM N 4445
-- -------------------------------
-- Auteur : rampon-c
-- Date : 30/09/2010
-- -----------------------------------------------------------
-- Description :
-- --------------------
-- selection les differents exercice dudgetaire
-- -----------------------------------------------------------
-- Paramètres :    CUR_o_data Curseur de sortie
-- -----------------------------------------------------------
PROCEDURE PROC_SEL_EXERCICE
(
 CUR_o_data OUT PCKG_PILOTAGE_.refcur
)
IS
BEGIN
OPEN CUR_o_data FOR
       SELECT num_exbudg AS NUMERO_EXERCICE_BUDG
       FROM exer pilotage  AS EXERCIE_PILOTAGE
     ORDER BY num_exbudg;
EXCEPTION
  WHEN  NO DATA_FOUND THEN
    RETURN;
 WHEN OTHERS THEN
    RAISE;
END PROC_SEL_EXERCICE;


-------------------------------------------------------------
-- Nom : PROC_SEL_COMMUNE_NOM_DEPT
-- -----------------------------------------------------------
-- Création : Correctif CF DEMANDE GEDEM N 4446
-- -------------------------------
-- Auteur : rampon-c
-- Date : 30/09/2010
-- -----------------------------------------------------------
-- Description :
-- --------------------
-- selectionne des informations sur une commune
-- -----------------------------------------------------------
-- Paramètres :    CUR_o_data Curseur de sortie
--                 pv_i_Cod_Dept : Code du département sur 2 caractères
--                 pv_i_Lib_Commune : libelle de la commune
-- -----------------------------------------------------------
PROCEDURE PROC_SEL_COMMUNE_NOM_DEPT
(
   pv i Cod Dept IN VARCHAR2,
   pv_i_Lib_Commune IN VARCHAR2,
   CUR_o_data OUT PCKG_ PILOTAGE_REF.refcur
)
IS
BEGIN
OPEN CUR_o_data FOR
       SELECT comm cod insee AS CODE INSEE
            , comm_libl_commune AS CODE_COMMUNE
            , comm_cod_dept AS CODE_DEPARTEMENT
       FROM comm pilotage
       where cod dept = pv i Cod Dept
       and  UPPER(lib_commune) like '%'||UPPER(pv_i_Lib_Commune)||'%'
       ORDER BY comm_cod_insee;
EXCEPTION
  WHEN  NO DATA_FOUND THEN
    RETURN;
 WHEN OTHERS THEN
    RAISE;
END PROC_SEL_COMMUNE_NOM_DEPT ;
```

10.7 Les propriétaires, comptes, synonymes, rôles & procédures à lancer

10.7.1 Propriétaires, comptes, synonymes et rôles

Chaque application contient :

- Un propriétaire d'objets (Tables, vues, séquences)
- Un propriétaire des packages
- Un compte administrateur de l'application
- Un compte utilisateur
- Trois rôles pour les accès aux objets
- Des synonymes publics pour chaque objet
- Des packages contenant les requêtes

Pour chaque objet (table, vue, séquence), il faut :

- Faire un GRANT Select …… To ROLE_APPLI_S
- Faire un GRANT Select, Insert, Update, Delete …To ROLE_APPLI_SIUD
- Faire un GRANT Select, Insert, Update, Delete …To PROC$APPLI : pour pouvoir utiliser un objet dans une procédure stockée Oracle, il faut avoir les droits en direct sur l'objet (et non à travers le rôle)
- Faire un GRANT Select, Insert, Update, Delete …To GENUSER_APPLI.

Remarque : le droit DROP PUBLIC SYNONYM a été rajouté aux users **PROC$...**

Les procédures stockées dialoguant entre 2 bases doivent être écrites sous le user **INTERFACE**.

10.7.2 Procédures à lancer

Pour exemple, voir les procédures du projet Pilotage sous O:\DSI\Projets\Pilotage et Paramétrage d'Astre\Expression des besoins 2005.

10.7.2.1 Droits SELECT sur les tables, vues et séquences.

S'exécute sous le propriétaire GEN$*APPLI.*

Donne le droit SELECT au rôle ROLE_*APPLICATION*_S.

Ex.:
```
grant select on <TABLE> to <ROLE>
/
grant select on aide to role_pilotage_s
/
grant select on ap to role_pilotage_s
/
```

(cf. procédure **Infocentre – Base (1) Grant Select.sql**)

A exécuter si un compte utilisateur particulier a été créé (autre que les GENUSER_Application et PROC$).

10.7.2.2 Droits Select, Insert, Update, Delete sur les tables et vues et Select sur les séquences.

S'exécute sous GEN$*APPLI.*

Donne les droits SELECT, INSERT, UPDATE, DELETE au rôle ROLE_APPLICATION_SIUD.

Ex : grant select, insert, update, delete on <TABLE> to <ROLE>
/
grant select, insert, update, delete on aide to role_pilotage_siud
/
grant select, insert, update, delete on ap to role_pilotage_siud
/

(cf. procédure **Infocentre – Base (2) Grant SIUD.sql**)

A exécuter pour le compte administrateur.

10.7.2.3 Synonymes publics sur les tables, vues et séquences.

S'exécute sous GEN$*APPLI.*

Crée des synonymes publics sur chaque objet (table, vue, séquence) pour que les autres comptes puissent accéder à ces objets.

Le nom du synonyme doit être composé du nom court de la table suivi de *_APPLI.*

Ex : create public synonym <SYNONYM> for <TABLE>

create public synonym aide_pilotage for aide
/
create public synonym ap_pilotage for ap
/
create public synonym apreel_pilotage for ap_reel
/

(cf. procédure **Infocentre – Base (3) Synonymes.sql**)

10.7.2.4 Droits en S, I, U, D pour le propriétaire des Packages.

S'exécute sous GEN$*APPLI.*

Donner les droits SELECT, INSERT, UPDATE et DELETE au user PROC$*APPLI* pour que les objets soient accessibles dans les packages.

Ex : grant select, insert, update, delete on <TABLE> to <proc$APPLI>

grant select, insert, update, delete on aide to proc$pilotage
/
grant select, insert, update, delete on ap to proc$pilotage
/
grant select, insert, update, delete on ap_reel to proc$pilotage
/

(cf. procédure **Infocentre – Base (4) Grant SIUD PROC$.sql**)

10.7.2.5 Droits d'exécution des Packages au rôle ROLE_PILOTAGE_PCKG

S'exécute sous PROC$*APPLI.*

Donne les droits d'EXECUTION de chaque package Pckg*_APPLI_Mnémonique* qui ont le rôle ROLE_PILOTAGE_PCKG.

Ex : grant execute on <NOM PACKAGE> to <role_APPLI_pckg>
/
grant execute on pckg_pilotage_sit to role_pilotage_pckg
/
grant execute on pckg_pilotage_dos to role_pilotage_pckg
/

(cf. procédure **Infocentre – Base (5) Grant execute proc.sql**).

L'utilisateur de l'application **GENUSER** doit aussi posséder un droit d'exécution sur le package
grant execute on GENUSER_APPLI

ATTENTION :
Pour les packages de PROC$ENV qui sont communs à toutes les applications (Communes, types de voie, etc.), **le droit d'exécution du package doit être Attribué directement au user PROC$*APPLI* :**

```
grant execute on <NOM PACKAGE de PROC$ENV> to proc$APPLI
/

grant execute on pckg_env_commptt to proc$bourssan
/
grant execute on pckg_env_tvoie to proc$bourssan
/
```

10.7.2.6 Création des synonymes publics pour chaque package

S'exécute sous PROC$*APPLI*
Permet d'exécuter des procédures stockées sans avoir à préciser le propriétaire des packages.

Par exemple, sous Reporting Services, pour exécuter la procédure proc_session du package Pckg_pilotage_ref, il suffit d'écrire : **Pckg_pilotage_ref.proc_session.**

Ex :
```
create public synonym <NOM PACKAGE> for <proc$APPLI.NOM PACKAGE>
/
create public synonym pckg_pilotage_dos for proc$pilotage.pckg_pilotage_dos
/
create public synonym pckg_pilotage_ses for proc$pilotage.pckg_pilotage_ses
/
create public synonym pckg_pilotage_tiers for proc$pilotage.pckg_pilotage_tiers
/
```
(cf. procédure **Infocentre – Base (6) Synonyme proc.sql**)

10.8 Database Link

10.8.1 Cas d'utilisation d'un DBLINK

Un dblink permet de se connecter depuis une base de données vers une autre base distante. Pour ce faire, il est possible de le configurer de plusieurs manières.

10.8.1.1 Public / Privé

- Soit le dblink est public auquel cas tous les utilisateurs identifiés peuvent l'utiliser.
- Soit le dblink est privé et dans ce cas seul l'utilisateur qui a créé le dblink peut en faire usage.

10.8.1.2 Utilisateur de connexion

- Soit l'utilisateur de connexion est spécifié, dans ce cas tout utilisateur identifié peut emprunter les identifiant du dblink pour effectuer des opérations dans la base cible.
- Soit l'utilisateur de connexion n'est pas renseigné et dans ce cas il faut que cet utilisateur existe dans les bases cible et source avec le même mot de passe.

Généralement seules les 2 configurations suivantes sont utilisées (principalement pour des raisons de sécurité) :

- Un dblink public et des utilisateurs de connexion non spécifiés
- Un dblink privé et l'utilisateur de connexion spécifié (exemple avec interface GEDELIB)

10.8.2 Droits utilisateurs

Dans chacun des cas les droits utilisateurs de SIUD ou exécution de package doivent respecter les mêmes règles que celles précisées plus haut dans le document.

10.9 Le requêteur

Les droits nécessaires

Si une application utilise le requêteur, il faut donner les droits d'exécution du package du requêteur :

O:\DSI\Projets\Requeteur\Developpements\Base de données\Requeteur – Base (5) –Grant execute proc.sql

Et donner les droits d'accès aux tables de l'application au rôle `role_reqtr_siud` **:**

O:\DSI\Projets\Requeteur\Developpements\Base de données\Requeteur – Base (2) –Grant SIUD.sql

10.10 Quand lancer ces procédures

10.10.1 A chaque création de table, vue ou séquence.

Lorsqu'une nouvelle table (ou vue ou séquence) est créée, il faut donner les 'GRANT' aux rôles et créer le synonyme.

Pour cela, lancer les 4 procédures :

- Infocentre – Base (1) Grant Select.sql
- Infocentre – Base (2) Grant SIUD.sql
- Infocentre – Base (3) Synonymes.sql
- Infocentre – Base (4) Grant SIUD PROC$.sql

Cela n'est pas nécessaire pour une modification de table.

10.10.2 A chaque création de package.

Lorsqu'un nouveau Package est créé, il faut donner les 'GRANT' au rôle et créer le synonyme.

Pour cela, lancer les 2 procédures :

- Infocentre – Base (5) Grant execute proc.sql
- Infocentre – Base (6) Synonyme proc.sql

Cela n'est pas nécessaire pour une modification de package (Rajout d'une procédure ou modification du code d'une procédure).

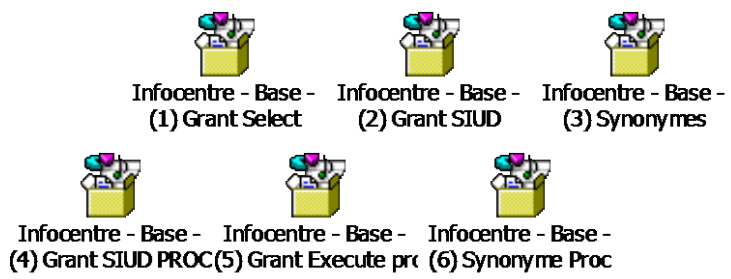
Les scripts cités sont récupérables en annexe du document.

10.11 Tableau récapitulatif

COMPTES, ROLES, SYNONYMES	NOM	DESCRIPTION
Un Compte PROPRIETAIRE D'OBJETS	**GEN$*APPLI***	Contient les objets (tables, vues, séquences) de l'application
Un Compte PROPRIETAIRE des PACKAGES PROCEDURES et FONCTIONS	**PROC$*APPLI***	Contient les packages de l'application A les droits Select, Insert, Update, Delete sur les objets
Un Compte UTILISATEUR	**GENUSER_*APPLI***	C'est le compte utilisateur de connexion à l'application A le rôle ROLE_APPLI_PCKG (droit d'exécuter les packages)
Un Compte ADMINISTRATEUR	***APPLI*_ADMIN**	Permet d'administrer l'application A le rôle ROLE_*APPLI*_SIUD
TROIS ROLES	**ROLE_*APPLI*_S** **ROLE_*APPLI*_SIUD** **ROLE_*APPLI*_PCKG**	Droit de Select sur les objets Droit de Select, Insert, Update, Delete sur les objets Droit d'exécution des procédures stockées
SYNONYMES PUBLICS	***Nom_Court_Objet_APPLI***	Ces synonymes seront vus par tous les comptes de la base
PACKAGES	**PCKG_*APPLI*_Mnémonique**	Contient les requêtes codées en Pl/Sql

10.12 Annexe : Scripts Infocentre

Les fichiers ci-dessous contiennent les scripts grants et synonymes généralement nécessaires lors de la création de packages et tables :



11 DOCUMENT 5 – 1.00 – SOCLE TECHNIQUE

11.1 Introduction

11.1.1 Objectifs du document

Le présent document a pour objectif de préciser l'ensemble des fonctions du Socle Technique, utilisé par l'ensemble des développeurs pour l'accès aux données.

11.1.2 Domaine d'application

Tous les développements pour la Région PACA fait en C# ou ASP avec le FRAMEWORK .NET 1.1

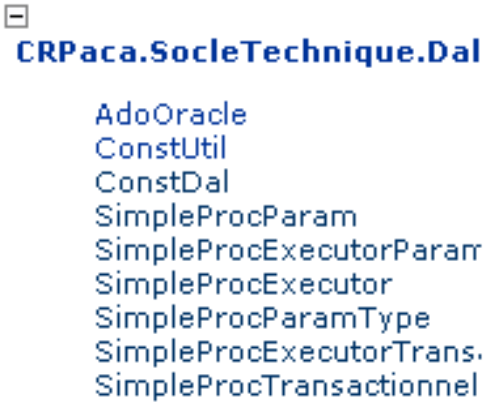
11.2 Contenu détaille du socle technique

CRPaca.SocleTechnique Solution

Projet
CRPaca.SocleTechnique.Dal
CRPaca.SocleTechnique.Exceptions
CRPaca.SocleTechnique.IWrappers
CRPaca.SocleTechnique.UI
CRPaca.SocleTechnique.Util
CRPaca.SocleTechnique.Wrappers
CRPaca.SocleTechnique.WrapperService
ServerRemotingTesteur

11.2.1 CRPaca.SocleTechnique.Dal

Ce projet contient l'ensemble des méthodes servant à l'accès à la base de données.



11.2.1.1 AdoOracle

Cette classe offre les fonctions de base de gestion de la base de données: Connexion Déconnexion Gestion des transactions.

11.2.1.2 ConsUtil

Cette classe détermine les fonctions d'accès à la base de registre, définit le répertoire de base ou se trouve les informations de registre :

11.2.1.3 ConstDal

Cette classe définit les noms des constantes de base permettant l'accès à la base de registre

11.2.1.4 SimpleProcParam

Cette classe représente un paramètre d'entrée ou de sortie pour l'appel d'une procédure stockée. C'est une "structure" de type: nom, valeur, type, taille Cette classe est censé être utilisée uniquement en interne du DAL

11.2.1.5 SimpleProcExecutorParams

Cette classe permet de définir tout les paramètres pour exécuter une procédure stockée: nom de la procédures, paramètres d'entrées et de sorties, etc.

11.2.1.6 SimpleProcExecutor

Cette classe exécute de procédure stockée sur Oracle. Elle utilise la classe SimpleProcExecutorParams afin de paramétrer l'appel d'une procédure. Cette classe distribue 2 méthode , permettant l'appel de procédures. Elle ne peux exécuter qu'une procédure à la fois Elle ne peux utiliser qu'un curseur de sortie à la fois Elle utilise les clés registre pour obtenir la chaine de connexion sur la base de donnée En cas d'erreurs elle déclenche des Exceptions (voir classes d'Exceptions du DAL)

11.2.1.7 SimpleProcParamType

SimpleProcParam représente un paramètre d'entrée ou de sortie pour l'appel d'une procédure stockée. C'est une "structure" de type: nom, valeur, type, taille. Cette classe est censé être utilisée uniquement en interne du DAL

11.2.1.8 SimpleProcExecutorTransactionnel

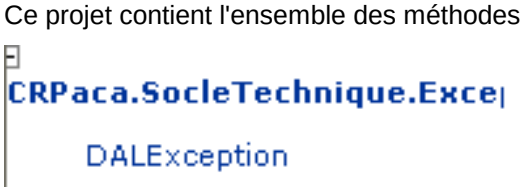
SimpleProcExecutorParams afin de paramétrer l'appel d'une procédure. Cette classe distribue 2 méthode, permettant l'appel de procédures. Elle ne peux exécuter qu'une procédure à la fois Elle ne peut utiliser qu'un curseur de sortie à la fois Elle utilise les clés registre pour obtenir la chaine de connexion sur la base de donnée En cas d'erreurs elle déclenche des Exceptions (voir classes d'Exceptions du DAL)

11.2.1.9 SimpleProcTransactionnel

Cette classe représente une collection d'objets SimpleProcExecutorTransactionnel

11.2.2 CRPaca.SocleTechnique.Exceptions

Ce projet contient l'ensemble des méthodes permettant la gestion des exceptions.



11.2.2.1 DALException

Cette classe reçoit un message, un code erreur, SerializationInfo info et StreamingContext context en paramètre.

11.2.3 CRPaca.SocleTechnique.IWrappers

Ce projet regroupe l'ensemble des interfaces et permet d'exécuter de manière distante une procédure stockée.

11.2.3.1 ISimpleProcExecutor

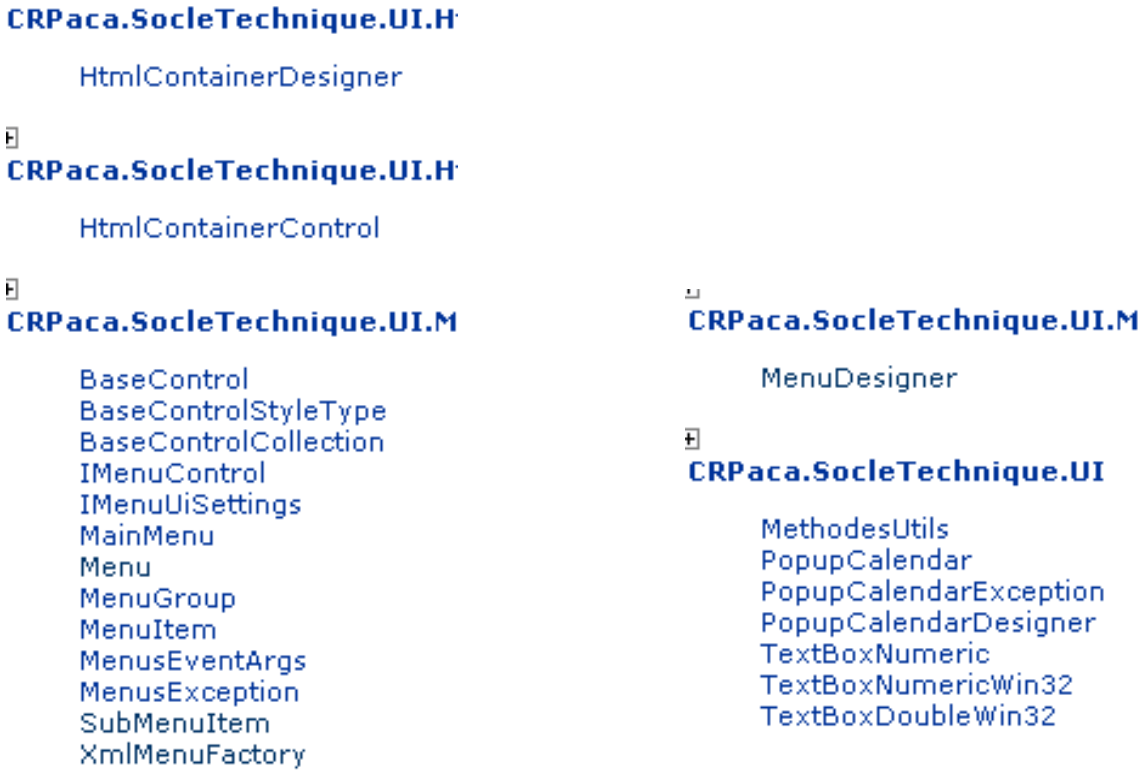
Cette interface donne accès à la Partie DAL du SocleTechnique Elle permet d'exécuter de manière distante une procédure stockée.

11.2.3.2 ISimpleProcExecutorTransactionnel

Cette interface donne accès à la Partie DAL transactionnelle du SocleTechnique Elle permet d'exécuter de manière distante une procédure stockée.

11.2.4 CRPaca.SocleTechnique.UI

Ce projet regroupe l'ensemble des customs control WEB et Win32 pouvant être utilisé sur l'ensemble des projets .net.



11.2.4.1 HtmlContainerDesigner

Permet d'écrire à la volée un contenu HTML.

11.2.4.2 HtmlContainerControl

Génère ce contrôle dans le paramètre de sortie spécifié.

11.2.4.3 BaseControl

Classe permettant l'accès aux propriétés d'un élément du Menu XML généré

11.2.4.4 BaseControlStyleType

Permet de définir couleur de texte et classe CSS utilisé

11.2.4.5 BaseControlCollection

Collection de contrôles présents dans un Menu XML

11.2.4.6 IMenuControl

Interface IMenuControl, met à disposition onItemClick et getUniqueID qui permettent le click et l'identification unique d'un élément du menu

11.2.4.7 IMenuUiSettings

Inteface IMenuUiSettings. Expose les méthodes qui obtiennent les différents paramètres graphiques du menu (couleur, css..)

11.2.4.8 MainMenu

Classe qui construit de manière dynamique les éléments du Menu

11.2.4.9 Menu

Menu est un contrôle web permettant de générer un menu html vertical ou horizontal. Ce menu est composé de tables et de liens html, ainsi que de javascript. Il est personnalisable à partir de fichiers css, pour permettre des "rollovers" sur les items du menu. La "hiérarchie" du menu se présente sous la forme suivante : 1 MainMenu qui contient 1 ou plusieurs MenuGroup qui contiennent 0 ou plusieurs MenuItem qui contiennent 0 ou plusieurs SubMenuItem.

11.2.4.10 MenuGroup

Définit sur un élément du menu le linkButton nécessaire à la création du lien HTML

11.2.4.11 MenuItem

Définit les événements pouvant être associés à un élément du Menu (onMouseClick, onMouseOver...)

11.2.4.12 MenusEventArgs

Définit une classe MenusEventArgs qui hérite EventArgs. Permet d'ajouter au menu des événements particulier non présents dans la classe EventArgs

11.2.4.13 MenusException

MenusException hérite de la classe system Exception. Permet de surcharger oui de définir de nouveaux types d'exception propres à l'utilisation du menu.

11.2.4.14 SubMenuItem

SubMenuItem hérite de la classe MenuItem. Permet de placer un menu enfant dans le menu parent.

11.2.4.15 XmlMenuFactory

Cette classe est utilisée pour créer les instances des MainMenu, MenuGroup et MenuItem à partir d'un fichier de configuration xml

11.2.4.16 MenuDesigner

Construit une page Web de type Menu HTML. Utilise l'ensemble des classes décrites pour ça.

11.2.4.17 MethodesUtils

Classe qui rassemble l'ensemble des méthodes utilitaires pouvant être utilisée coté WEB et coté client riche. (exemple, contrôle de saisie...)

11.2.4.18 PopUpCalendar

Ce contrôle Web permet d'obtenir un champ de type TextBox et un bouton permettant de faire apparaitre un calendrier jacascript. L'ensemble devient un composant Dot Net.

11.2.4.19 PopUpCalendarExceptions

Classe qui hérite de la classe Exception. Permet de déclencher des exceptions propres au calendrier comme un mauvais format de date par exemple.

11.2.4.20 PopUpCalendarDesigner

Cette classe va générer le contenu HTML nécessaire à l'affichage du contrôle dans une page Web

11.2.4.21 TextBoxNumeric

Contrôle TextBox n'acceptant que des données numériques. Ce contrôle gère côté client via un javascript la saisie de chiffres. Optimiser pour l'insertion dans la ToolBox de VS .net Préfixe contrôle : crpaca Nom du contrôle : TextBoxNumeric

11.2.4.22 TextBoxNumericWin32

Contrôle TextBox pour client riche n'acceptant que des données numériques entières.

11.2.4.23 TextBoxDoubleWin32

Contrôle TextBox pour client riche n'acceptant que des données numériques entières.

11.2.5 CRPaca.SocleTechnique.Util

Ce projet regroupe les différentes classe utilitaire permettant l'accès aux bases de registre (entre autres)

11.2.5.1 ConstUtil

Cette classe définit les noms des constantes de base permettant l'accès à la base de registre.

11.2.5.2 ConstUtilTraceLog

Cette classe définit les noms des constantes de la base de registre permettant l'utilisation des outils de trace log

11.2.5.3 ConstUtilSmtp

Cette classe définit les noms des constantes de la base de registre permettant l'utilisation du serveur de messagerie.

11.2.5.4 ConstUtilXml

Cette classe définit le nom des constantes utiles aux outils XML.

11.2.5.5 ConstUtilEventLog

Cette classe définit le nom des constantes utiles aux outils des traces log.

11.2.5.6 PerfCounterMana

PerfCounterMana. Permet d'utiliser un compteur de performance dans les applications. Initialise lance et incrémente un compteur

11.2.5.7 UtilConfig

Cette classe permet la lecture d'informations dans la base de registre

11.2.5.8 UtilEventLog

Cette classe met à disposition les outils nécessaires à la création de « trace » et de « log » de l'application.

11.2.5.9 UtilSmtp

Cette classe permet l'envoi d'Email

11.2.5.10 UtilTraceLog

Cette classe permet la gestion des traces

11.2.5.11 UtilXml

Cette classe permet la gestion des erreurs à remonter au niveau applicatif. Le paramétrage des erreurs est réalisé au travers d'un fichier de configuration au format XML.

11.2.5.12 UtilCfgXml

Cette classe permet la lecture d'informations dans la base de registre.

11.2.5.13 UtilRegistre

Cette classe met à disposition les outils nécessaires à la lecture d'information dans la base de registre. Par exemple, la chaine de connexion.

11.2.6 CRPaca.SocleTechnique.Wrapper

Partie « Wrapper » du service. hérite de l'interface IWrapper. Permet d'implémenter les méthodes public du socle technique

11.2.6.1 SimpleProcExecutorTransactionnelWrapper

.Cette classe est le "proxy" entre l'interface ISimpleProcExecutorTransactionnel et la classe d'exécution de procédures stockées SimpleProcExecutorTransactionnel Elle implémente les méthodes proposée par l'interface ISimpleProcExecutorTransactionnel.

11.2.6.2 SimpleProcExecutorWrapper

Cette classe est le "proxy" entre l'interface ISimpleProcExecutor et la classe d'exécution de procédures stockées SimpleProcExecutor Elle implémente les méthodes proposée par l'interface ISimpleProcExecutor.

11.2.7 CRPaca.SocleTechnique.WrapperService

Ce projet contient le service socle technique.

11.2.7.1 ProjectInstaller

Projet Windows Installer. Permet si on le souhaite de créer un assistant d'installation du socle technique.

11.2.7.2 CRPacaWrapperService

Contient l'exécutable du socle technique installé comme service sous le serveur Windows.

11.2.8 ServerRemotingTesteur

Ce programme en mode texte est la pour tester le bon fonctionnement de la partie serveur sans installer de service.

12 DOCUMENT 5 – 1.01 – SOCLE TECHNIQUE

12.1 Introduction

12.1.1 Objectifs du document

Le présent document a pour objectif de préciser l'ensemble des fonctions du Socle Technique, utilisé par les développeurs pour l'accès aux données.

12.1.2 Domaine d'application

Tous les développements pour la Région PACA fait en C# ou ASP avec le FRAMEWORK .NET 2.0

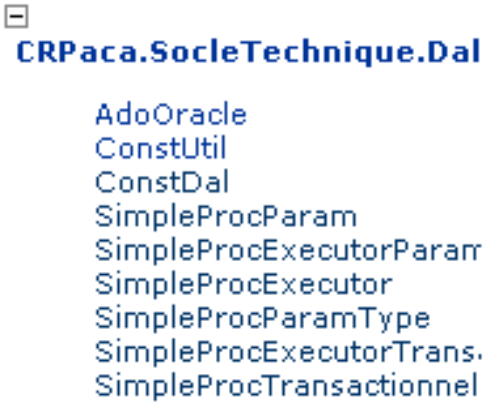
12.2 Contenu détaillé du socle technique

CRPaca.SocleTechnique Solution

Projet
CRPaca.SocleTechnique.Dal
CRPaca.SocleTechnique.Exceptions
CRPaca.SocleTechnique.IWrappers
CRPaca.SocleTechnique.UI
CRPaca.SocleTechnique.Util
CRPaca.SocleTechnique.Wrappers
CRPaca.SocleTechnique.WrapperService
ServerRemotingTesteur

12.2.1 CRPaca.SocleTechnique.Dal

Ce projet contient l'ensemble des méthodes servant à l'accès à la base de données.



12.2.1.1 AdoOracle

Cette classe offre les fonctions de base de gestion de la base de données: Connexion Déconnexion Gestion des transactions.

12.2.1.2 ConsUtil

Cette classe détermine les fonctions d'accès à la base de registre, définit le répertoire de base ou se trouve les informations de registre :

12.2.1.3 ConstDal

Cette classe définit les noms des constantes de base permettant l'accès à la base de registre.

12.2.1.4 SimpleProcParam

Cette classe représente un paramètre d'entrée ou de sortie pour l'appel d'une procédure stockée. C'est une "structure" de type: nom, valeur, type, taille Cette classe est censé être utilisée uniquement en interne du DAL.

12.2.1.5 SimpleProcExecutorParams

Cette classe permet de définir tous les paramètres pour exécuter une procédure stockée: nom de la procédure, paramètres d'entrées et de sorties, etc.

12.2.1.6 SimpleProcExecutor

Cette classe exécute de procédure stockée sur Oracle. Elle utilise la classe SimpleProcExecutorParams afin de paramétrer l'appel d'une procédure. Cette classe distribue 2 méthodes permettant l'appel de procédures. Elle ne peut exécuter qu'une procédure à la fois Elle ne peut utiliser qu'un curseur de sortie à la fois. Elle utilise les clés registre pour obtenir la chaine de connexion sur la base de donnée En cas d'erreurs elle déclenche des Exceptions (voir classes d'Exceptions du DAL).

12.2.1.7 SimpleProcParamType

SimpleProcParam représente un paramètre d'entrée ou de sortie pour l'appel d'une procédure stockée. C'est une "structure" de type: nom, valeur, type, taille. Cette classe est censée être utilisée uniquement en interne du DAL.

12.2.1.8 SimpleProcExecutorTransactionnel

SimpleProcExecutorParams afin de paramétrer l'appel d'une procédure. Cette classe distribue 2 méthodes permettant l'appel de procédures. Elle ne peut exécuter qu'une procédure à la fois. Elle ne peut utiliser qu'un curseur de sortie à la fois. Elle utilise les clés registre pour obtenir la chaine de connexion sur la base de donnée En cas d'erreurs elle déclenche des Exceptions (voir classes d'Exceptions du DAL).

12.2.1.9 SimpleProcTransactionnel

Cette classe représente une collection d'objets SimpleProcExecutorTransactionnel.

12.2.2 CRPaca.SocleTechnique.Exceptions

Ce projet contient l'ensemble des méthodes permettant la gestion des exceptions.

12.2.2.1 DALException

Cette classe reçoit un message, un code erreur, SerializationInfo info et StreamingContext context en paramètre.

12.2.3 CRPaca.SocleTechnique.IWrappers

Ce projet regroupe l'ensemble des interfaces et permet d'exécuter de manière distante une procédure stockée.

12.2.3.1 ISimpleProcExecutor

Cette interface donne accès à la Partie DAL du SocleTechnique Elle permet d'exécuter de manière distante une procédure stockée.

12.2.3.2 ISimpleProcExecutorTransactionnel

Cette interface donne accès à la Partie DAL transactionnelle du SocleTechnique Elle permet d'exécuter de manière distante une procédure stockée.

12.2.3.3 CRPaca.SocleTechnique.UI

Ce projet regroupe l'ensemble des customs control WEB et Win32 pouvant être utilisé sur l'ensemble des projets .net.

12.2.3.4 HtmlContainerDesigner

Permet d'écrire à la volée un contenu HTML.

12.2.3.5 HtmlContainerControl

Génère ce contrôle dans le paramètre de sortie spécifié.

12.2.3.6 MethodesUtils

Classe qui rassemble l'ensemble des méthodes utilitaires pouvant être utilisée coté WEB et coté client riche. (Exemple, contrôle de saisie...)

12.2.3.7 PopUpCalendar

Ce contrôle Web permet d'obtenir un champ de type TextBox et un bouton permettant de faire apparaitre un calendrier jacascript. L'ensemble devient un composant Dot Net. **Ce composant devrait à terme être abandonné au profit du calendar XHTML que l'on peut retrouver dans la solution extranet Commun.**

12.2.3.8 PopUpCalendarExceptions

Classe qui hérite de la classe Exception. Permet de déclencher des exceptions propres au calendrier comme un mauvais format de date par exemple.

12.2.3.9 PopUpCalendarDesigner

Cette classe va générer le contenu HTML nécessaire à l'affichage du contrôle dans une page Web

12.2.3.10 TextBoxNumeric

Contrôle TextBox n'acceptant que des données numériques. Ce contrôle gère côté client via un javascript la saisie de chiffres. Optimiser pour l'insertion dans la ToolBox de VS .net Préfixe contrôle : crpaca Nom du contrôle : TextBoxNumeric

12.2.3.11 TextBoxNumericWin32

Contrôle TextBox pour client riche n'acceptant que des données numériques entières.

12.2.3.12 TextBoxDoubleWin32

Contrôle TextBox pour client riche n'acceptant que des données numériques entières.

12.2.4 CRPaca.SocleTechnique.Util

Ce projet regroupe les différentes classes utilitaires permettant l'accès aux bases de registre (entre autres)

12.2.4.1 ConstUtil

Cette classe définit les noms des constantes de base permettant l'accès à la base de registre.

12.2.4.2 ConstUtilTraceLog

Cette classe définit les noms des constantes de la base de registre permettant l'utilisation des outils de trace log

12.2.4.3 ConstUtilSmtp

Cette classe définit les noms des constantes de la base de registre permettant l'utilisation du serveur de messagerie.

12.2.4.4 ConstUtilXml

Cette classe définit le nom des constantes utiles aux outils XML.

12.2.4.5 ConstUtilEventLog

Cette classe définit le nom des constantes utiles aux outils des traces log.

12.2.4.6 PerfCounterMana

PerfCounterMana. Permet d'utiliser un compteur de performance dans les applications. Initialise lance et incrémente un compteur

12.2.4.7 UtilConfig

Cette classe permet la lecture d'informations dans la base de registre

12.2.4.8 UtilEventLog

Cette classe met à disposition les outils nécessaires à la création de « trace » et de « log » de l'application.

12.2.4.9 UtilSmtp

Cette classe permet l'envoi d'Email

12.2.4.10 UtilTraceLog

Cette classe permet la gestion des traces

12.2.4.11 UtilXml

Cette classe permet la gestion des erreurs à remonter au niveau applicatif. Le paramétrage des erreurs est réalisé au travers d'un fichier de configuration au format XML.

12.2.4.12 UtilCfgXml

Cette classe permet la lecture d'informations dans la base de registre.

12.2.4.13 UtilRegistre

Cette classe met à disposition les outils nécessaires à la lecture d'information dans la base de registre. Par exemple, la chaine de connexion.

12.2.5 CRPaca.SocleTechnique.Wrapper

Partie « Wrapper » du service. Hérite de l'interface IWrapper. Permet d'implémenter les « méthodes public » du socle technique

12.2.5.1 SimpleProcExecutorTransactionnelWrapper

Cette classe est le "proxy" entre l'interface ISimpleProcExecutorTransactionnel et la classe d'exécution de procédures stockées SimpleProcExecutorTransactionnel Elle implémente les méthodes proposée par l'interface ISimpleProcExecutorTransactionnel.

12.2.5.2 SimpleProcExecutorWrapper

Cette classe est le "proxy" entre l'interface ISimpleProcExecutor et la classe d'exécution de procédures stockées SimpleProcExecutor Elle implémente les méthodes proposée par l'interface ISimpleProcExecutor.

12.2.6 CRPaca.SocleTechnique.WrapperService

Ce projet contient le service socle technique.

12.2.6.1 ProjectInstaller

Projet Windows Installer. Permet si on le souhaite de créer un assistant d'installation du socle technique.

12.2.6.2 CRPacaWrapperService

Contient l'exécutable du socle technique installé comme service sous le serveur Windows.

12.2.7 ServerRemotingTesteur

Ce programme en mode texte est là pour tester le bon fonctionnement de la partie serveur sans installer de service.

12.2.8 Composants dépréciés

Les composants que l'on va lister ci-dessous sont utilisés pour les menu d'un site web basé sur l'ancienne ergonomie et le Framework 1.1. Ce sont des composants qui ne sont plus préconisés pour un développement en 2.0. En effet le Framework possède nativement ses propres composants liés à un SiteMap. Ces derniers sont intégrés dans la master page de chacune des solutions.

12.2.8.1 BaseControl

Classe permettant l'accès aux propriétés d'un élément du Menu XML généré

12.2.8.2 BaseControlStyleType

Permet de définir couleur de texte et classe CSS utilisé

12.2.8.3 BaseControlCollection

Collection de contrôles présents dans un Menu XML

12.2.8.4 IMenuControl

Interface IMenuControl, met à disposition onItemClick et getUniqueID qui permettent le click et l'identification unique d'un élément du menu

12.2.8.5 IMenuUiSettings

Inteface IMenuUiSettings. Expose les méthodes qui obtiennent les différents paramètres graphiques du menu (couleur, css..)

12.2.8.6 MainMenu

Classe qui construit de manière dynamique les éléments du Menu

12.2.8.7 Menu

Menu est un contrôle web permettant de générer un menu html vertical ou horizontal. Ce menu est composé de tables et de liens html, ainsi que de javascript. Il est personnalisable à partir de fichiers css, pour permettre des "rollovers" sur les items du menu. La "hiérarchie" du menu se présente sous la forme suivante : 1 MainMenu qui contient 1 ou plusieurs MenuGroup qui contiennent 0 ou plusieurs MenuItem qui contiennent 0 ou plusieurs SubMenuItem.

12.2.8.8 MenuGroup

Définit sur un élément du menu le linkButton nécessaire à la création du lien HTML

12.2.8.9 MenuItem

Définit les événements pouvant être associés à un élément du Menu (onMouseClick, onMouseOver...)

12.2.8.10 MenusEventArgs

Définit une classe MenusEventArgs qui hérite EventArgs. Permet d'ajouter au menu des événements particulier non présents dans la classe EventArgs

12.2.8.11 MenusException

MenusException hérite de la classe system Exception. Permet de surcharger oui de définir de nouveaux types d'exception propres à l'utilisation du menu.

12.2.8.12 SubMenuItem

SubMenuItem hérite de la classe MenuItem. Permet de placer un menu enfant dans le menu parent.

12.2.8.13 XmlMenuFactory

Cette classe est utilisée pour créer les instances des MainMenu, MenuGroup et MenuItem à partir d'un fichier de configuration xml

12.2.8.14 MenuDesigner

Construit une page Web de type Menu HTML. Utilise l'ensemble des classes décrites pour ça.

13 DOCUMENT 6 – 1.00 – ERGONOMIE DE L'INTERFACE CLIENT

13.1 Introduction

13.1.1 Objectifs du document

Le présent document a pour objectif de définir les règles d'ergonomie pour les développements d'applications en client riche et en client léger (pages web).

13.1.2 Utilisation de la norme

La norme doit être utilisée obligatoirement dans tous les nouveaux projets.

Pour tous les projets antérieurs, le développement doit suivre les normes du projet.

Pour ne pas respecter la norme définie dans ce dossier, il est obligatoire de préciser dans le cahier des charges la norme à appliquer sur le projet.

13.1.3 Domaine d'application de la norme

Tous les développements pour la Région PACA fait en C# ou ASP avec le FRAMEWORK .NET 2.0

13.2 Applications clients riches

13.2.1 Police – tailles des fenêtres – position des fenêtres - couleur d'écran

13.2.1.1 Règles

La police utilisée est MS sans sérif de taille 10.

La taille de la fenêtre est de 1024*768 pixels. L'écran doit être capable d'afficher toutes les informations

Les fenêtres doivent toujours être positionnées en haut à gauche, et non se placer de façon relative plus bas à droite de la fenêtre précédente.

La couleur de l'écran est Gray.

Le nombre de fenêtres ouvertes simultanément est limité à 5.

L'application doit lister les fenêtres actives dans un menu dédié.

13.2.1.2 Organisation

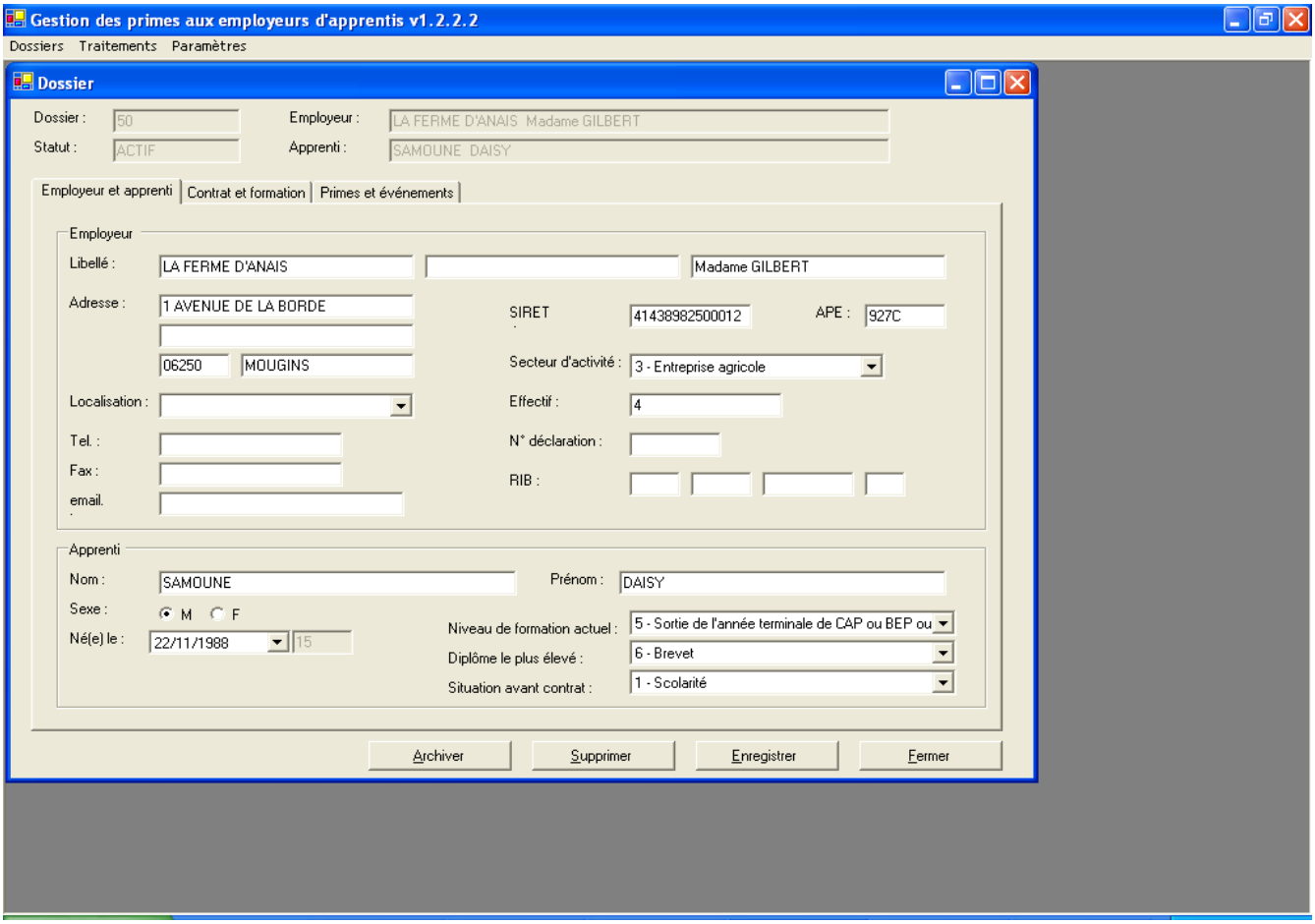
Les fenêtres sont organisées en trois parties :

Entête (non modifiable) contenant les informations résumant l'entité gérer par la fenêtre.

Corps (modifiable) : contient toutes les informations gérer par la fenêtre.

Ligne d'actions : contient tous les boutons d'actions (Edition, Enregistrer …).

13.2.1.3 Exemple

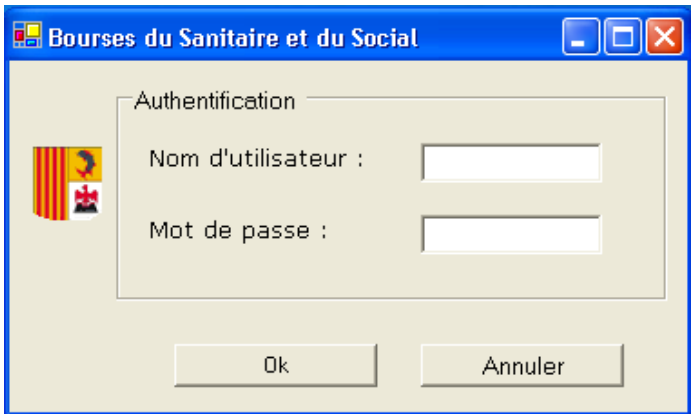


13.2.2 Fenêtre de connexion

13.2.2.1 Règles

Les boutons pour accéder à l'application sont « Ok » ou « Annuler » pour sortir.

13.2.2.2 Exemple



13.2.3 Menu

13.2.3.1 Règles

Le menu est uniquement métier. Aucune fonction du style « Edition → Copier/Coller ».

Selon le profil de la personne connectée certaines parties du menu seront grisées (inaccessibles). Le menu « Paramètres » sera accessible uniquement pour les personnes désignées comme administrateur de l'application.

L'entrée de chaque module applicatif sera effectuée par un écran de recherche.

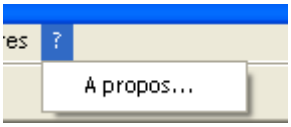
Le Menu est à mettre dans une fenêtre WinMenuGeneral dérivant de sopragroup.accelerate.module.CRPacaMenuGeneral.

13.2.3.2 Barre d'outils

Plus de barre d'outils comme il était utilisé précédemment dans les applications client serveur. Les fonctions enregistrer, ajouter, supprimer, liste de valeur seront gérées via des boutons.

13.2.3.3 Versions

Affichage du « ? » permettant d'afficher le nom de l'écran, son application et sa version sur 3 chiffres.



La version sera de la forme **chiffre.chiffre.chiffre*Lettre en minuscule.***

Exemple : 1.0.0.0a

Chaque modification d'une application en production entraînera l'évolution de son numéro de version.

- Le premier chiffre représente la livraison d'une application dans sa globalité. Il changera uniquement dans le cas d'évolutions majeures ou de modifications impactant l'application dans sa globalité (cas d'une migration technique).
- Le second représente un module ou un lot de l'application.
- Le troisième représente un livraison intermédiaire de ce lot.

- La lettre représente des corrections de bugs ou des petites évolutions (rajout d'un virgule, changement de libellé, déplacement d'un champ, etc.).

Exemple 1 : Livraison du module Gestion dans la nouvelle application PRV2 développée en .NET.

Sa version initiale sera : 1.1.1a.

Si correction de bugs mineurs, la version passera en 1.1.1b, puis 1.1.1c, etc.

S'il y a beaucoup de Bugs à corriger ou une évolution, correction importante à effectuer sur le lot 1 alors on regroupera la livraison non plus sous une lettre mais dans une livraison intermédiaire d'un lot/module. L'application passerait alors en 1.1.2a.

Exemple 2 : Livraison d'un autre module dans la nouvelle application PRFV2. Ce module sera livré en 2 étapes.

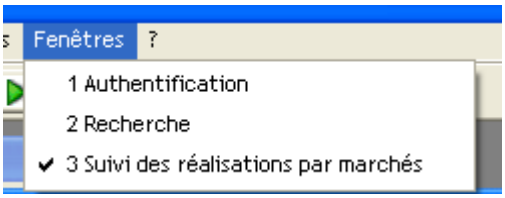
La version de l'application à la première étape sera : 1.2.1a.

La version de l'application à la deuxième étape sera : 1.2.2a.

La fenêtre A PROPOS n'est pas à faire, il existe une propriété dans la fenêtre CRPACAMenuGeneral pour ces changements.

13.2.3.4 Menu fenêtre

Affichage d'un menu fenêtre permettant de passer d'une fenêtre à une autre.



13.2.4 Gestion des DATAGRID

13.2.4.1 Formatage des données affichées

13.2.4.1.1 Données obligatoires

Garder le format de la donnée. Exemple : une date obligatoire doit être définie dans le DATATABLE comme étant de type date.

Les montants sont de type DECIMAL avec un formatage de la colonne en N2.

13.2.4.1.2 Données non obligatoires

Toutes les données non obligatoires doivent être formaté en STRING afin d'accepter les données NULL.

13.2.4.1.3 Formatage

Date : YYYY/MM/DD

Nombres : alignés à droite

Nombre réel : N2

Nombre entier : N0

13.2.4.2 Utilisation

Le tri doit être possible sur toutes les colonnes affichées (seulement pour les DATAGRID où les données sont affichées en ligne, et non pas pour ceux où les données sont affichées en colonne)

13.2.4.2.1 DATAGRID non modifiable et partiellement modifiable

Un DATAGRID non modifiable doit être en READONLY.

On ne doit pas pouvoir ajouter ou supprimer des lignes directement depuis le DATAGRID.

13.2.4.2.2 DATAGRID modifiable

Ajout illimité de lignes

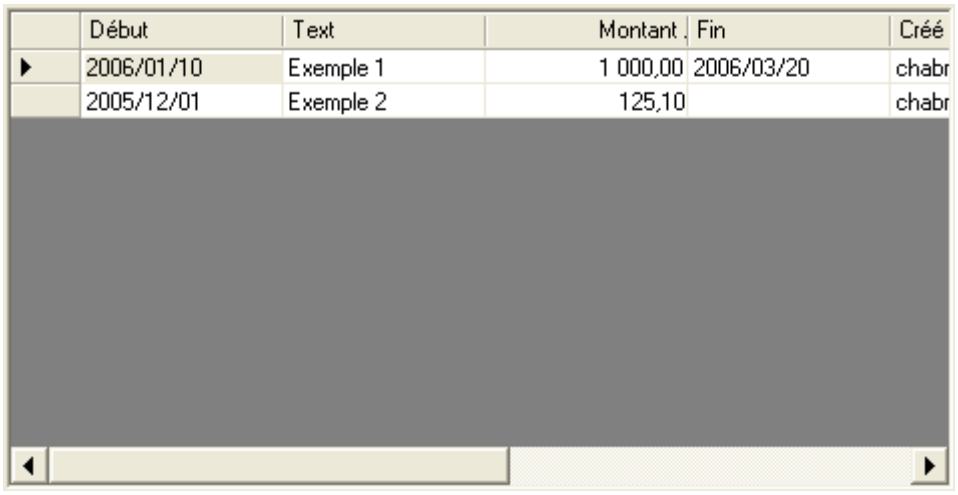
Suppression des lignes directement depuis le DATAGRID (hors problème technique).

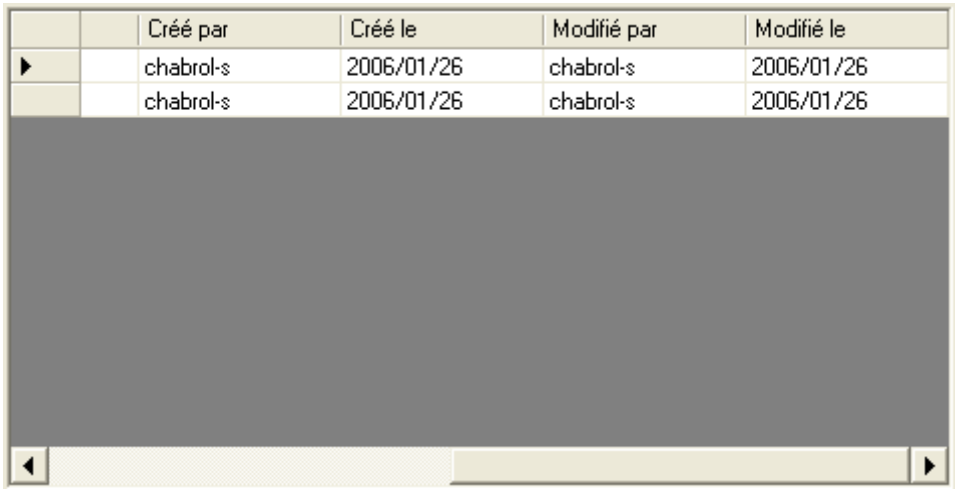
13.2.4.2.3 A ne pas oublier

Les 4 colonnes « Créé par », « Créé le », « Modifié par » et « Modifié le » sont systématiquement affichées dans tous les datagridset et doivent apparaitre en dernier. Le datagrid doit être organisé de telle sorte que ces 4 colonnes ne soient pas visibles immédiatement, sauf si on utilise l'ascenseur horizontal. Ces colonnes ne sont jamais modifiables.

La taille des colonnes est définie en fonction du nombre de données à afficher et de la longueur de la donnée à afficher.

13.2.4.3 Exemple





13.2.5 Gestion des onglets

13.2.5.1 Accessibilité

Seul les onglets accessibles doivent être affichés. Pour ce faire il suffit de donner, au TABCONTROL, les onglets visibles.

Pour un nouvel enregistrement, seul les onglets représentant l'entité principale sont affichés.

13.2.5.2 Exemples

13.2.6 Gestion des contrôles de saisie

13.2.6.1 Mise en page

Date : on doit pouvoir avoir une date NULL

Nombre : aligné à droite. Format N2 pour les montants sinon on utilisera le format N0.

Le premier élément d'un COMBOBOX doit être : « Choisissez… ».

13.2.6.2 Contrôles éditables

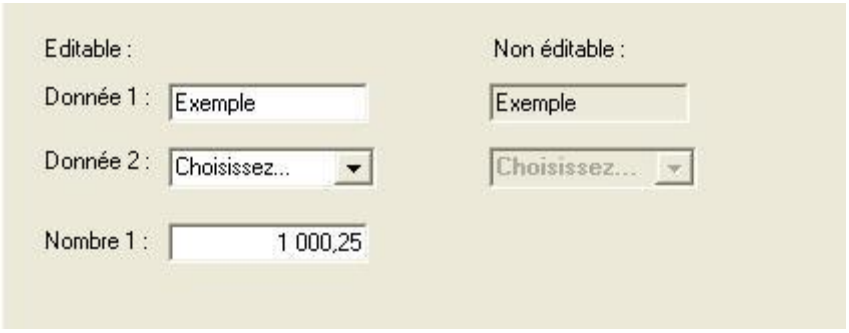
Nombre : reste aligné à droite par contre il faut enlever le formatage (N2 ou N0) lors de l'édition de la donnée.

13.2.6.3 Contrôles non éditables

TEXTBOX : doivent être en READONLY.

COMBOBOX et DATETIMEPICKER : ENABLE = FALSE, et le texte doit être en gras.

13.2.6.4 Exemple



13.2.7 Gestion des boutons

13.2.7.1 Les boutons

Les boutons d'Edition sont généralement placés à en bas à gauche sauf pour les écrans dédiés.

Les boutons actions sont placés en bas à droite. Le bouton FERMER est toujours le bouton le plus à droite. Les autres boutons sont placés par ordre d'importance et d'utilisation : le plus utilisé est à gauche, le moins utilisé est à coté du bouton FERMER.

Exemple : ENREGISTRER ; SUPPRIMER ; FERMER.

La taille des boutons doit être si possible de 95 * 24 pixels.

13.2.7.2 Accessibilité

Les boutons non utilisables doivent être « grisés ».

13.2.7.3 Exemples

Le bouton ENREGISTRER devient disponible

13.2.8 Messages et MESSAGEBOX

13.2.8.1 MESSAGEBOX

13.2.8.1.1 Barre des titres

Messages d'information : « INFORMATION ».

Messages d'alerte : « ATTENTION »

Messages d'erreur : « ERREUR »

13.2.8.1.2 Boutons

Messages d'erreur et d'information : 1 bouton « OK »

Messages d'alerte : 2 boutons « OUI » et « NON »

13.2.8.1.3 Icônes

Messages d'information : « MessageBoxIcon.Information »

Messages d'alerte : « MessageBoxIcon.Warning »

Messages d'erreur : « MessageBoxIcon.Error »

13.2.8.2 Messages

13.2.8.2.1 Changement de l'état de la fenêtre

Si l'utilisateur a fait des modifications dans la fenêtre sans avoir enregistré, afficher le message : « Voulez-vous enregistrer les modifications OUI/NON? »

Sinon n'afficher aucune message et sortir

13.2.8.2.2 Message avant la suppression

L'application doit demander une validation de l'action « supprimer » avant son exécution.

Le message est le suivant : « ATTENTION !!! Voulez-vous vraiment supprimer cet enregistrement ? ».

13.2.8.2.3 Autres messages

Message après l'échec d'une recherche : « Aucun résultat trouvé ».

Message lorsqu'il n'est pas possible d'ouvrir une fenêtre : « Vous ne pouvez avoir que 5 fenêtres actives ».

13.2.8.3 Exemples

13.2.9 Raccourcis

Conformément à la norme Microsoft, l'accès aux menus ou boutons sera accessible via des raccourcis claviers. Ceux-ci seront repérés par une lettre soulignée, en majuscule. L'accès sera alors possible par la combinaison des touches **Alt – Lettre**.

Exemples :

Alt N : a**N**nuler

Alt E : **E**nregistrer

Alt F : **F**ermer

Il faudra s'assurer que les raccourcis d'accès aux menus n'utilisent pas les raccourcis d'accès aux boutons.

Il sera possible, mais ce choix sera lié à l'application elle-même, de coder des raccourcis appellant directement un écran. L'accès sera alors possible par la combinaison des touches **Ctrl – lettre**.

Par exemple, dans l'application GPEA, l'appel à l'écran « n**O**uveau » sera possible par Ctrl -O.

Le menu sera alors, dans cet exemple, « Dossier » → « n**O**uveau – Ctrl O ».

13.2.10 Libellés

13.2.10.1 Règles

Tous les libellés sont suivis de « : ».

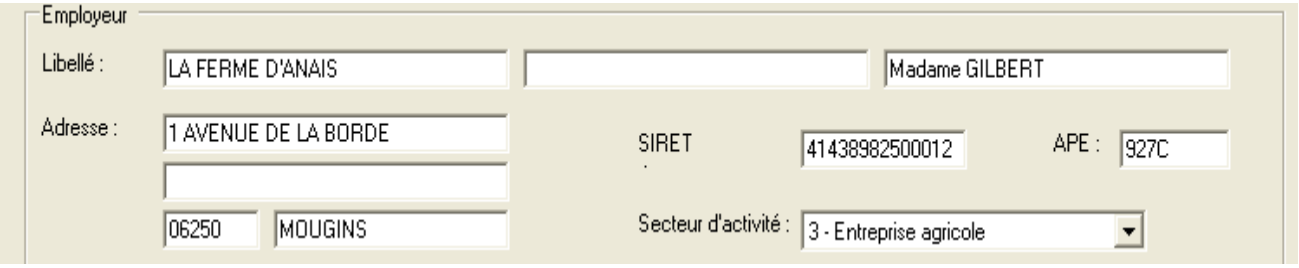
Ils sont alignés à gauche.

La première lettre est en majuscule.

Les mots ne sont pas coupés ou abrégés à part quelques abréviations bien définies. Par exemple :

Nom	**Abrégé**
Téléphone	Tél.
Numéro	N°

13.2.10.2 Exemples

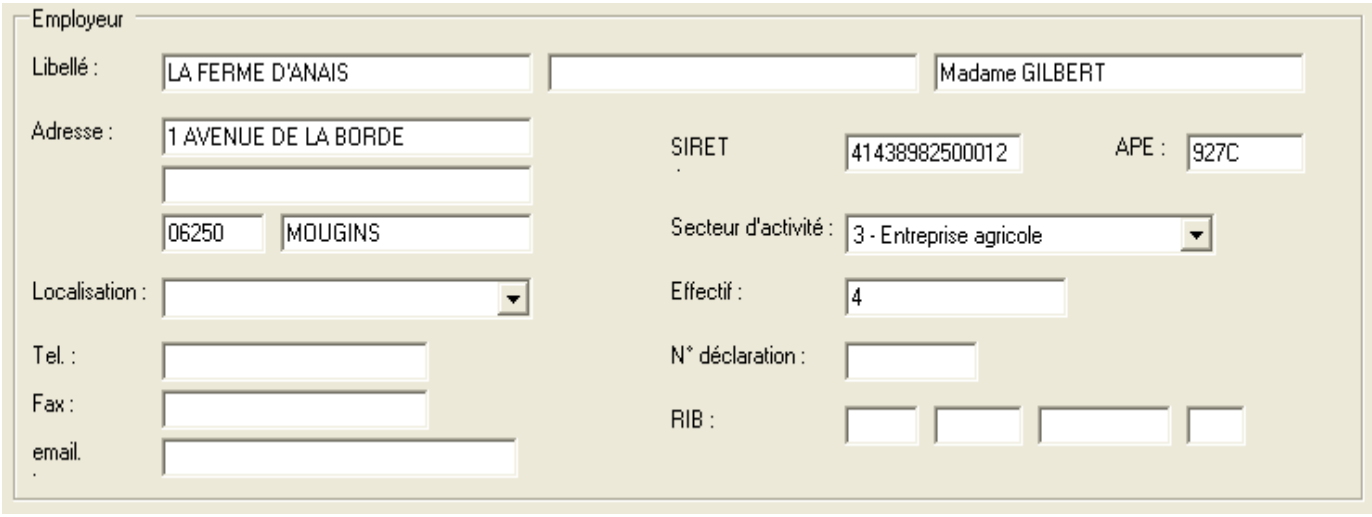


13.2.11 Champs

Ils sont alignés à gauche (voir ci-dessus).

Les champs non modifiables sont grisés et inaccessibles à l'utilisateur.

Les champs associés à une même notion sont regroupés dans un cadre (GroupBox).



13.2.12 Liste de valeurs

13.2.12.1 Règles

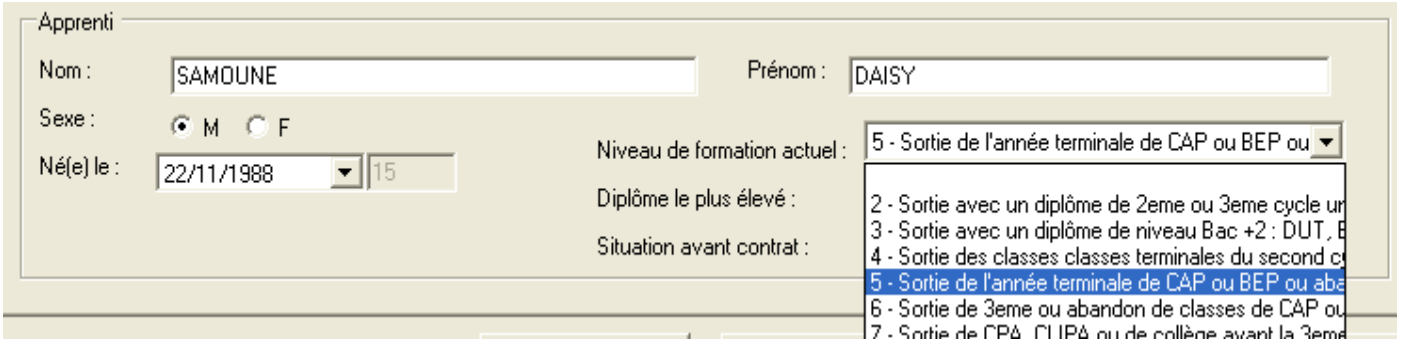
Les champs possédant des listes de valeurs sont gérés :

Par listBox si le nombre de valeurs n'est pas trop important.

Par popup dans le cas de listes de valeurs trop importantes (exemple : liste des communes). L'accès à cet écran de fera par l'icône placée à côté du champ. Il donnera l'accès à un écran de recherche.

Par Bouton radio dans le cas de valeurs non stockées en base et ne dépassant pas 3 choix possibles (exemple : choix du sexe).

13.2.12.2 Exemple



13.2.13 Navigation

Tous les champs de l'écran ainsi que les boutons peuvent être accessibles via le clavier. La navigation de champs en champs se fera tabulation (TAB) pour avancer et par MAJ+TAB pour reculer. L'ordonnancement des champs par la touche tabulation suit un ordre logique et s'organise par GroupBox.

13.2.14 Fenêtre de recherche

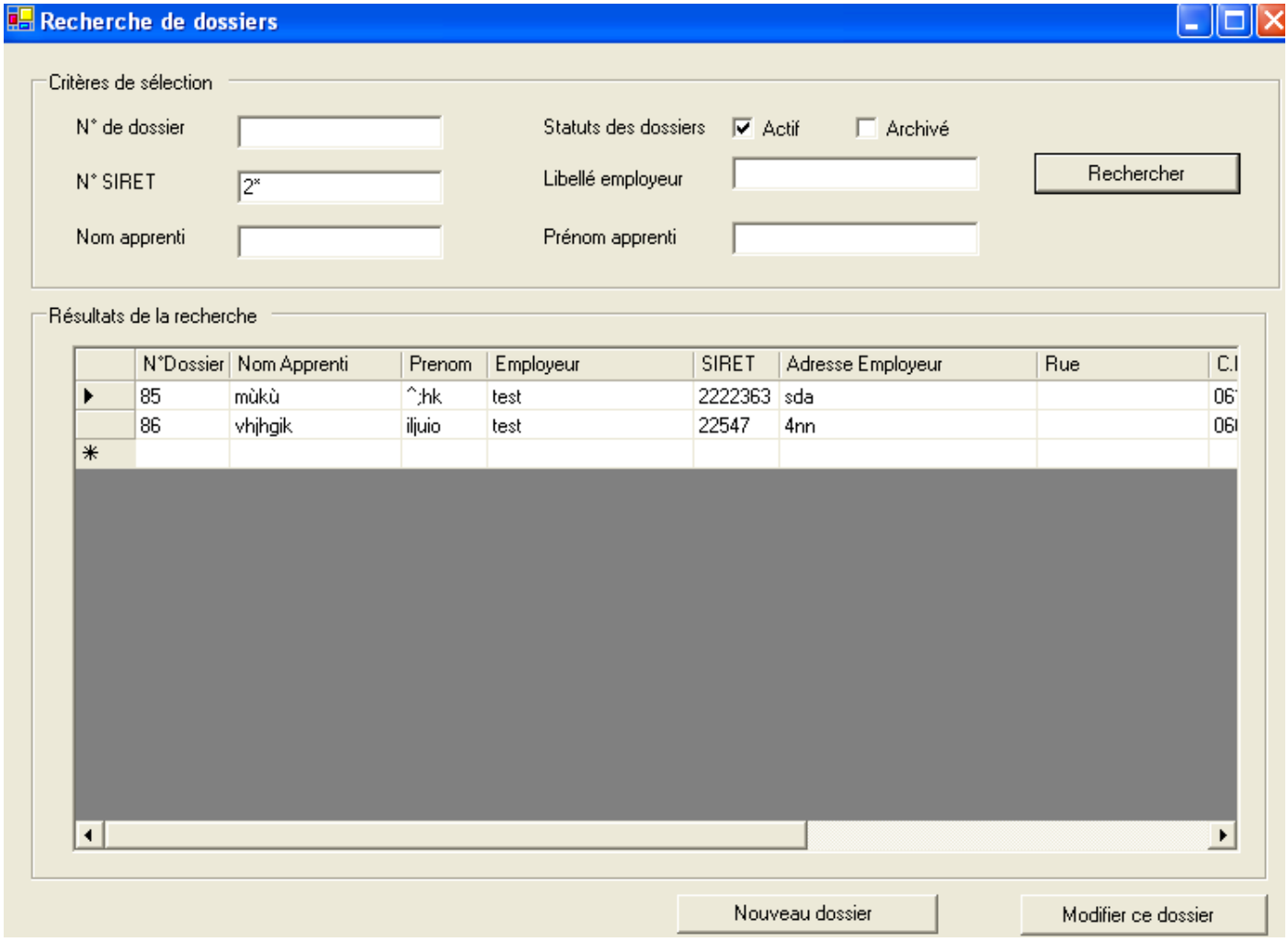
A chaque module développé doit être associé une fenêtre de recherche. Elle est composée de deux GroupBox :

Critères de sélection.

Résultat de la recherche = DataGrid.

Dans l'entête « critères de sélection » les utilisateurs renseignent un ou plusieurs de leurs critères de recherche. En cliquant sur le bouton **Rechercher**, ils déclenchent la recherche. Les résultats s'affichent dans la partie « Résultat de la recherche ». Un tri est possible par un « click » sur le nom de la colonne sur laquelle l'utilisateur souhaite effectuer un tri.

Cet écran est le point d'entrée de chaque module, l'accès aux autres écrans n'est, sinon, pas possible.



Dans le cas d'une recherche, le caractère indéfini sera représenté par « * ».

Pour accéder au détail d'une ligne, l'utilisateur peut :

Cliquer sur le bouton « modifier » ou « détail ».

Double-cliquer sur la ligne.

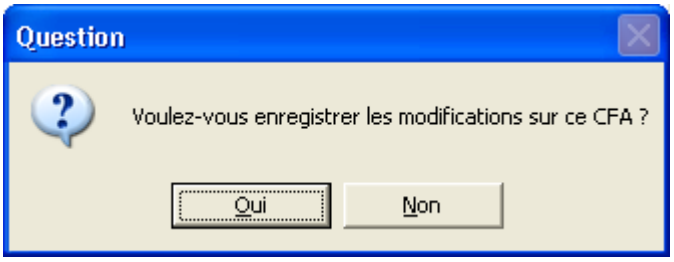
13.2.15 Règles transactionnelles

13.2.15.1 A la fermeture d'un écran

2 cas :

Aucune modification saisie : on sort sans message.

Une modification a été détectée, sans que l'utilisateur l'ait enregistrée : demander si on veut Enregistrer les modifications.



Si la réponse de l'utilisateur est :

Oui : On enregistre.

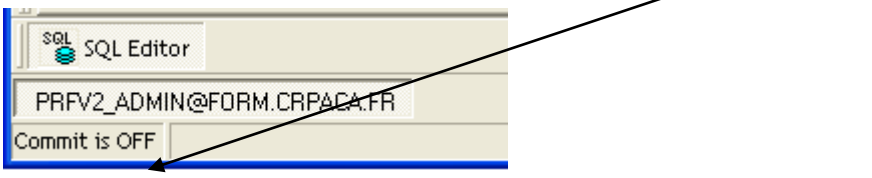
Non : On ne ignore les modifications.

13.2.15.2 Sur le bouton Enregistrer

N'afficher aucun message de confirmation.

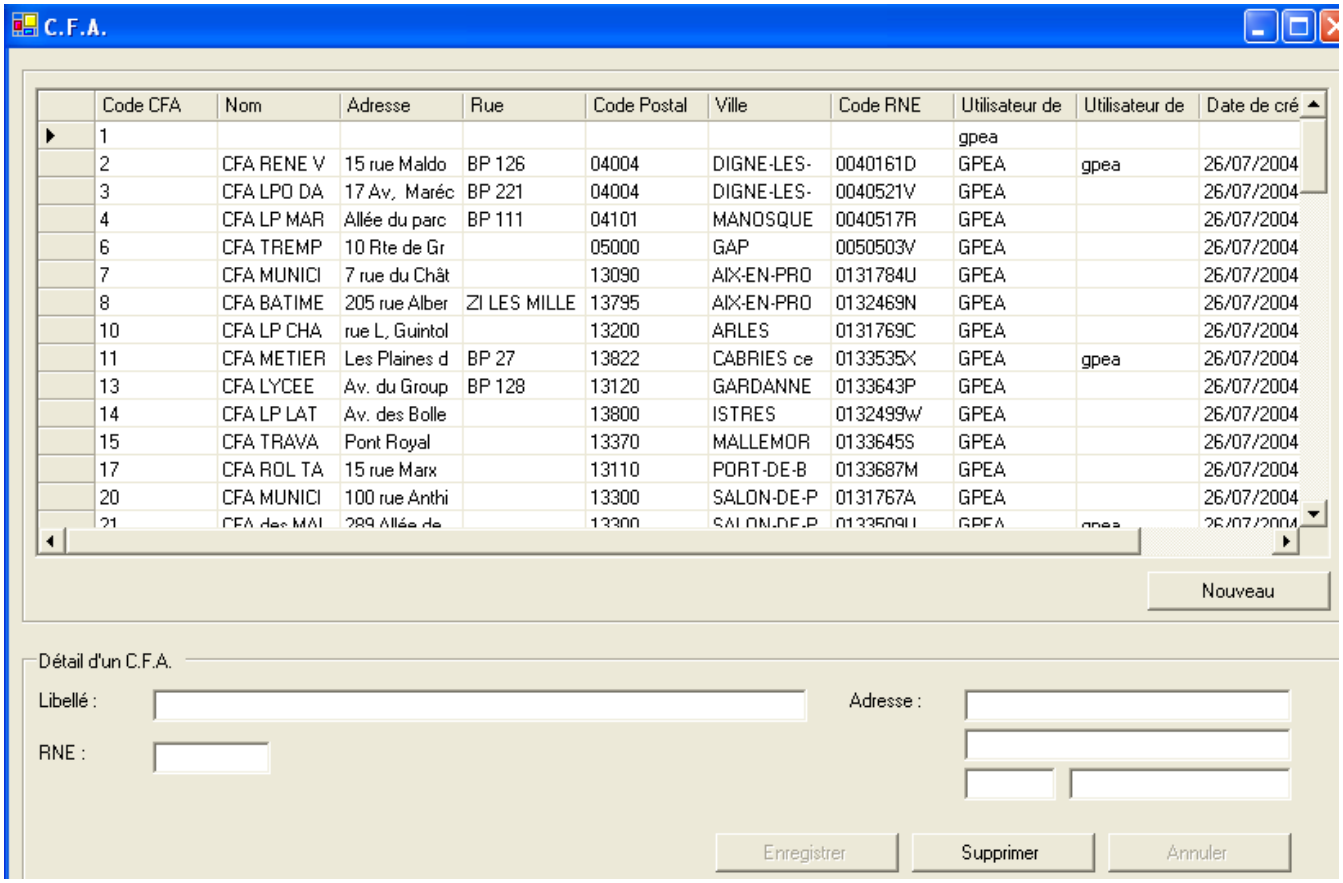
Si possible on affichera dans la barre d'avancement Windows, en bas à gauche, le message « Modification du dossier enregistrée».

Il faudra être attentif au rafraîchissement de cette de barre de tâches.



13.2.16 Ecrans de paramétrage

13.2.16.1 De style maître- détail « complexe »



Ajout d'une nouvelle ligne

Il faut utiliser pour cela le bouton « Nouveau ».

Lorsqu'on fait « nouveau », le bouton « enregistrer » ne doit pas être actif tant qu'on n'a rien saisi.

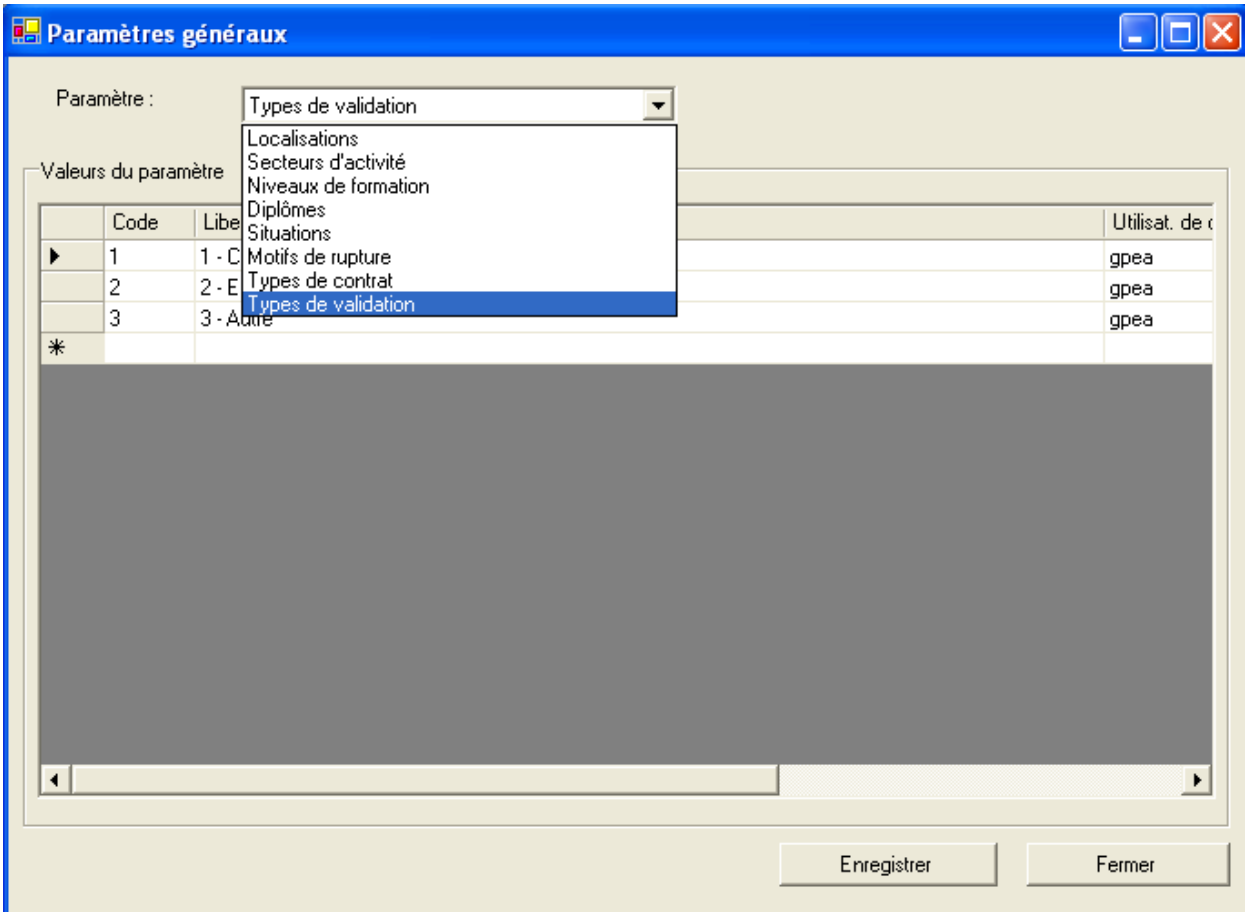
En création : positionner le curseur sur le premier champ de saisie.

Quand on change de paramètres : appliquer les mêmes règles que lors de la fermeture d'une fenêtre.

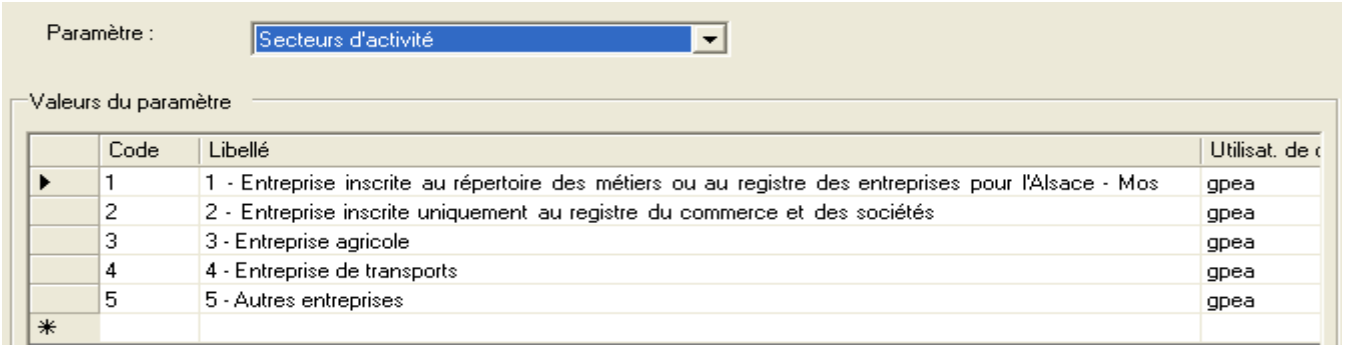
13.2.16.2 De style « simple »

Les paramètres représentés par un code et libellé pourront être regroupés dans un même écran « Paramètres généraux ».

Les boutons sur cet écran sont de droite à gauche, Fermer, Enregistrer, Annuler.

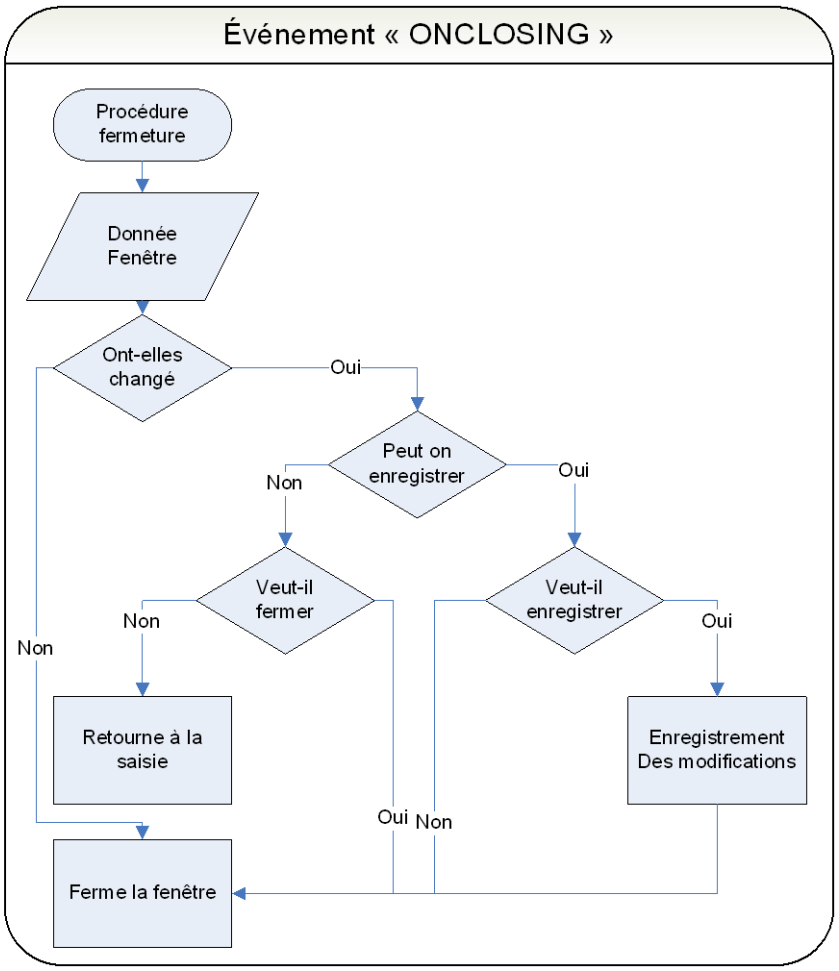


Autre choix de paramètres

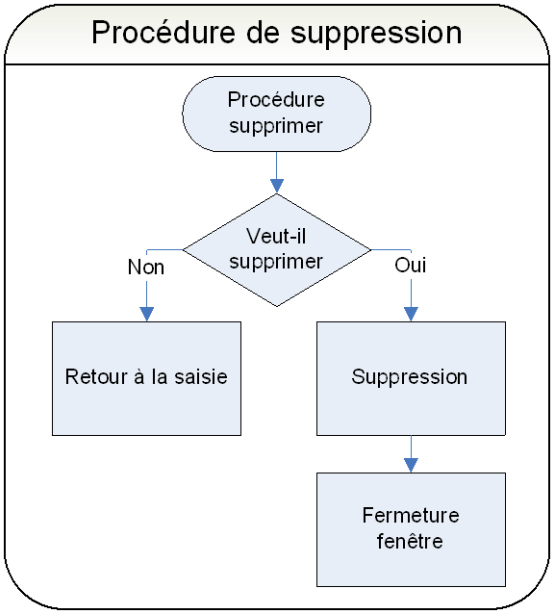


13.2.17 Procédures

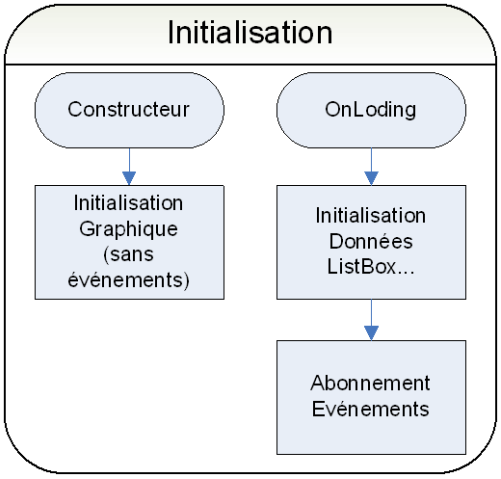
13.2.17.1 Procédure fermeture d'une fenêtre



13.2.17.2 Procédure de suppression

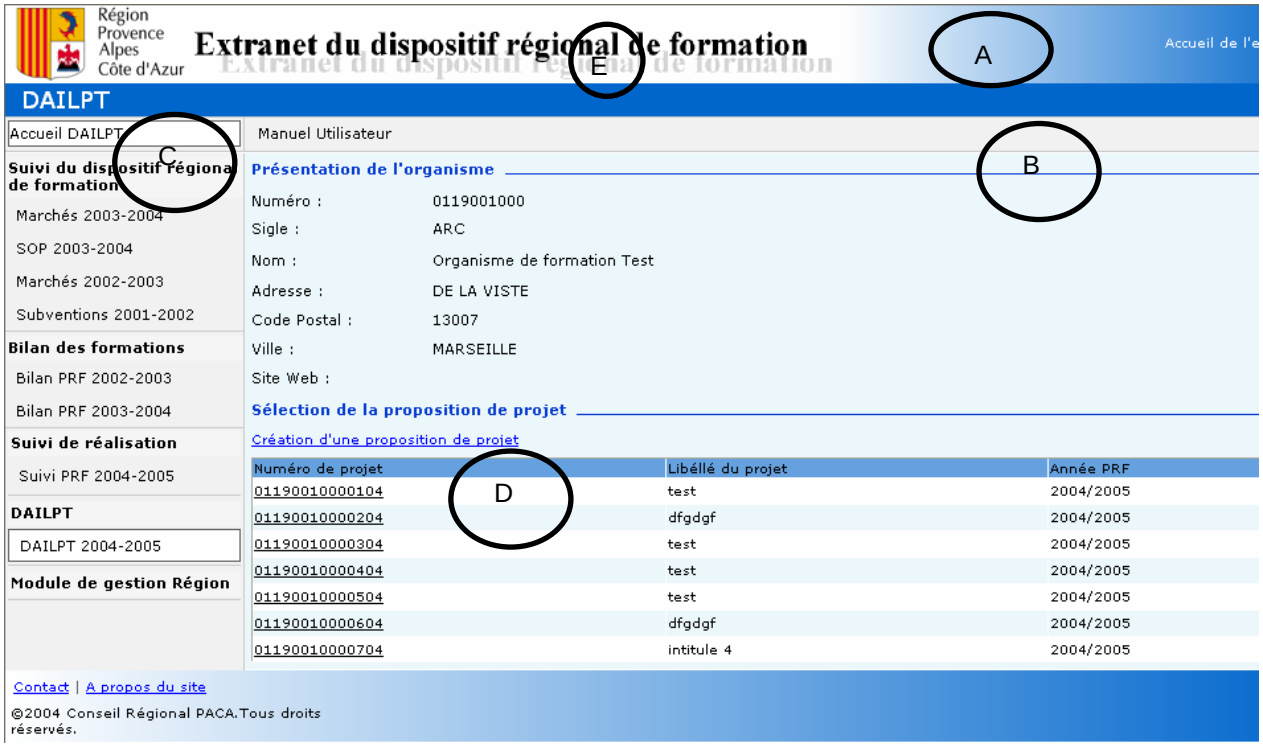


13.2.17.3 Procédure d'initialisation



13.3 Applications clients legers

13.3.1 Design général d'une page



La page peut-être découpée en 5 zones :

En-tête de page - A

En-tête du module - E

Menu - C

Page Principale - B

Pied de page – D

Les parties A, E, C et D sont apportés par la MASTER PAGE car ce sont des éléments fixes et communs à plusieurs pages. Il est préférable de ne pas dupliquer le code dans chaque page.

13.3.1.1 Décomposition

Une entête contenant :

Le logo de la région

Le sigle de l'application

Le nom détaillé de l'application

Le lien vers l'accueil

Un lien de déconnexion

Le titre de la page

Un menu positionné à gauche, il doit contenir un lien vers le manuel utilisateur.

A droite du menu, la partie métier de la page

En bas :

Un lien pour les contacts

Un lien « A propos du site »

Le copyright de la région.

13.3.1.2 Styles

Utilisation de la liste des styles CRPACA obligatoires.

Les styles les plus utilisés :

Titre de la page : style « bandeauTitre »

Le fond bleu ciel : style « formulaireBase »

les Labels de présentation des champs doivent être en gras

le titre des GROUPBOX : style « titreGroupBox » avec l'ajout d'une ligne bleu (<IMG height="1" src="/Commun/Images/1plblue.gif" width="100%">).

le dégrader du bas de page : style « gradient7 ».

13.3.2 Gestion des données

13.3.2.1 Affichage

Utilisation des mêmes règles de formatage que les clients riches. (Exemple, Nombre : alignement à droite, formatage N2 ou N0).

13.3.2.2 Enregistrement, suppression…

Affichage des mêmes messages que ceux des clients riches (§ 2.6)

Affichage du message informant l'utilisateur de la réussite de l'action : « L'action a été exécutée avec succès ».

13.3.3 Police – tailles des fenêtres – surbrillance – sélection d'un champ

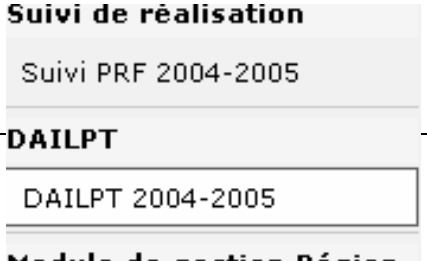
La police utilisée est le Verdana de taille 8.

L'écran doit être capable d'afficher toutes les informations dans une résolution de 800 x 600.

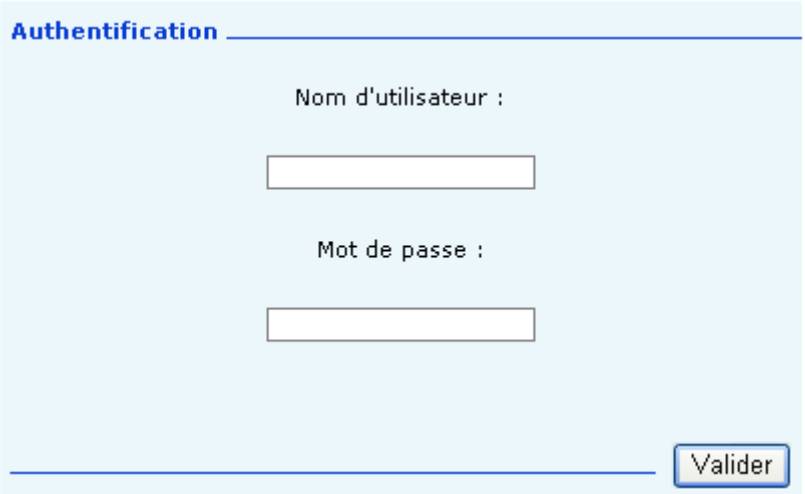
Le passage de la souris sur un item entraîne la surbrillance de celui-ci.

La couleur de surbrillance est le gris clair : #cccccc.

Afin d'indiquer à l'utilisateur la page sur laquelle il se trouve, celle-ci sera identifiée par une surbrillance « blanche ».



13.3.4 Fenêtre de connexion



L'accès au portail ne nécessite pas de connexion contrairement à l'accès aux différentes pages du site.

Les boutons pour accéder à l'application est « Valider ».

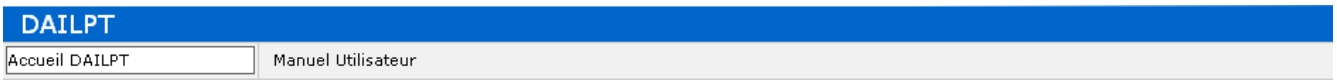
13.3.5 Zone « En-tête de la page » - A

Cette zone sera visible quelque soit la page appelée. Elle présente :

- En haut à gauche le logo de la Région. Un clic sur celui-ci dirigera l'utilisateur vers le site de la Région (www.cr-paca.fr).
- Au centre, le titre du portail. Un clic sur celui-ci ci dirigera l'utilisateur vers le portail.
- En haut à droite, l'accueil de l'extranet. Un clic sur celui-ci dirigera l'utilisateur vers le portail.

La couleur de fond de ce bandeau est un dégradé de blanc vers #0066cc.

13.3.6 Zone « En-tête du module » - E



Cette zone dépend du module sur lequel se trouvera l'utilisateur. Elle est composée :

d'un bandeau indiquant le titre du module. Les caractéristiques sont :

Police : Verdana 14

Couleur : Blanche

Couleur du bandeau : #0066cc

d'un bandeau « menu » permettant à tout moment de revenir sur l'accueil du module et de mettre à disposition, quelque soit la navigation à l'intérieur du module, les documents utiles à l'utilisateur. Les caractéristiques sont :

Police : Verdana 8

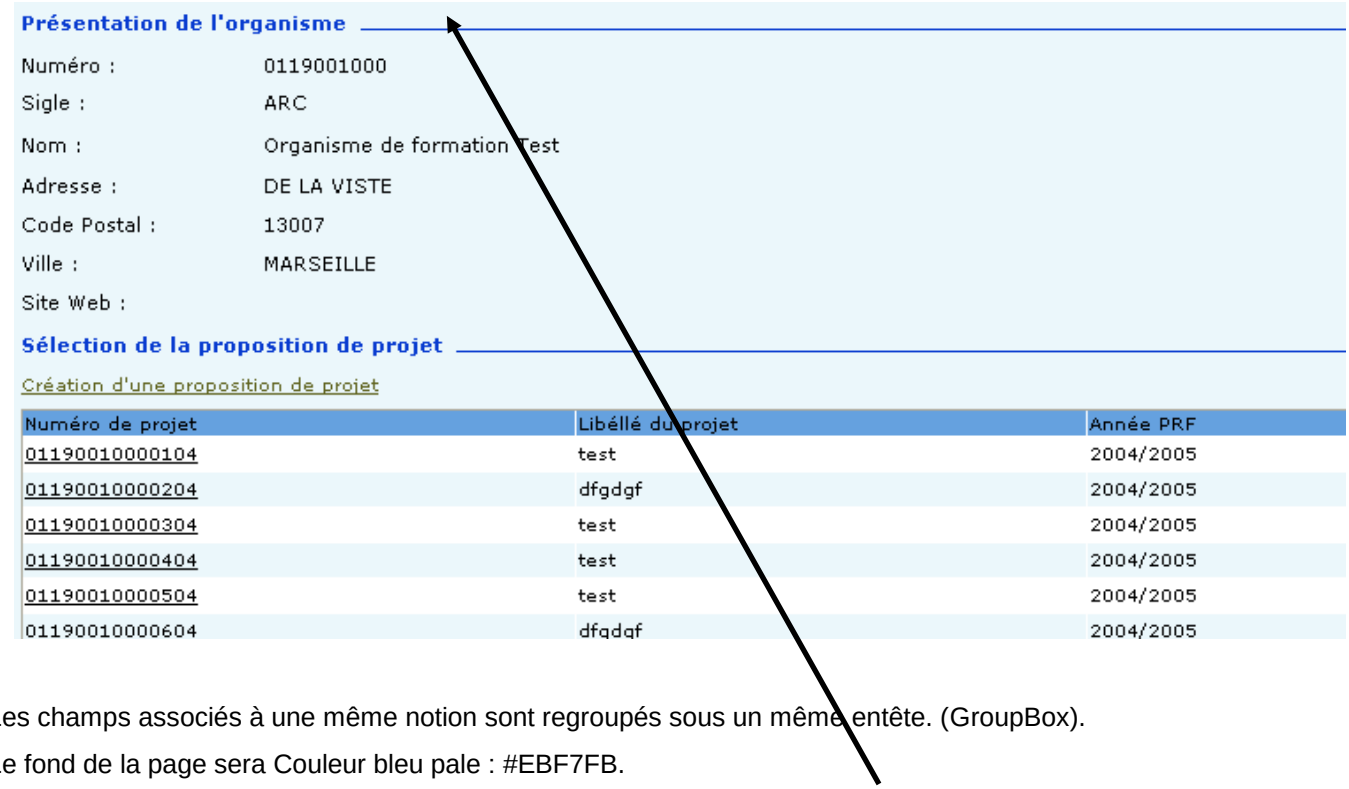
Couleur : Noir

Couleur du bandeau : #F1F1F1

13.3.7 Zone « Menu » - C

Police : Verdana 8 (gras et non gras)

13.3.8 Zone « Page Principale » - B



Les champs associés à une même notion sont regroupés sous un même entête. (GroupBox).

Le fond de la page sera Couleur bleu pale : #EBF7FB.

La police sera : Verdana 8.

Le police du titre du GroupBox est Verdana gras de 8, la couleur est la #093CD0, le trait est de la même couleur (#093CD0).

Chaque libellé sera suivi d'un blanc puis de « : ».

Les libellés sont alignés à gauche.

La première lettre est en majuscule.

Les champs non modifiables sont grisés et inaccessibles à l'utilisateur.

Les tableaux auront les caractéristiques suivantes :

Entête :

Police : Verdana 8

Couleur : noir

Les lignes du tableau

Police : Verdana 8

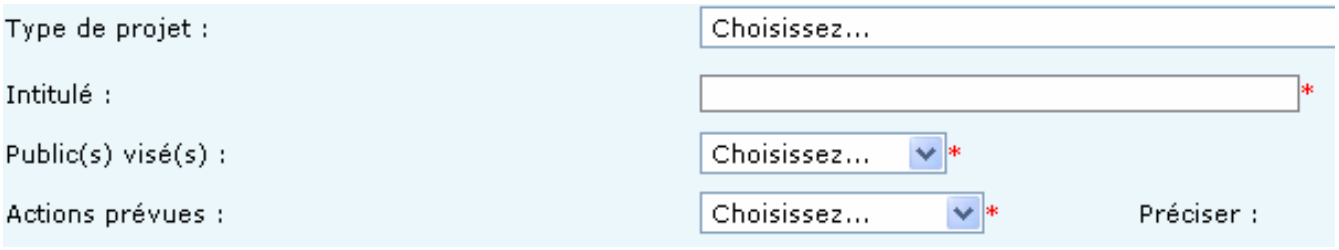
Couleur : noir

Les lignes seront de couleur :

Blanches.

bleu pale : #EBF7FB.

Les champs obligatoires seront suivi du « * »



Les mots ne sont pas coupés ou abrégés à part quelques abréviations bien définies. Par exemple :

Nom	**Abrégé**
Téléphone	Tél.
Numéro	N°

13.3.9 Zone « Pied de page » – D

Cette zone sera visible quelque soit la page appelée. Elle présente :

La couleur de fond de ce bandeau est un dégradé de blanc vers #0066cc.

Le lien **contact** permet de générer un mail à une adresse spécifiée.

A propos du site permet l'ouverture d'une pop-up indiquant le numéro de version de l'application.

La version sera de la forme **chiffre.chiffre.chiffre*Lettre en minuscule.***

Exemple : 1.0.0.0a

Chaque modification d'une application en production entraînera l'évolution de son numéro de version.

Le premier chiffre représente la livraison d'une application dans sa globalité. Il changera uniquement dans le cas d'évolutions majeures ou de modifications impactant l'application dans sa globalité (cas d'une migration technique).

Le second représente un module ou un lot de l'application.

Le troisième représente une livraison intermédiaire de ce lot.

La lettre représente des corrections de bugs ou des petites évolutions (rajout d'une virgule, changement de libellé, déplacement d'un champ, etc. …).

Exemple 1 : Livraison du module Gestion dans la nouvelle application PRV2 développée en .NET.

Sa version initiale sera : 1.1.1a.

Si correction de bugs mineurs, la version passera en 1.1.1b, puis 1.1.1c, etc. …

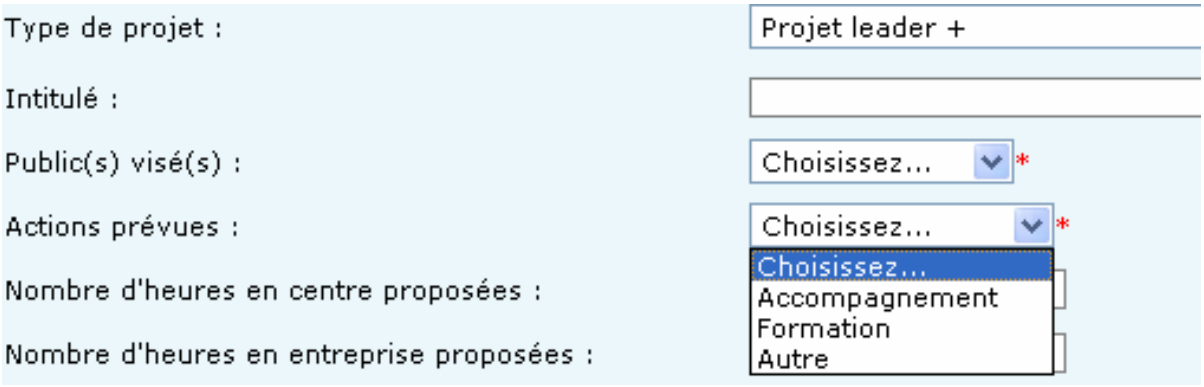
S'il y a beaucoup de Bugs à corriger ou une évolution, correction importante à effectuer sur le lot 1 alors on regroupera la livraison non plus sous une lettre mais dans une livraison intermédiaire d'un lot/module. L'application passerait alors en 1.1.2a.

Exemple 2 : Livraison d'un autre module dans la nouvelle application PRFV2. Ce module sera livré en 2 étapes.

La version de l'application à la première étape sera : 1.2.1a.

La version de l'application à la deuxième étape sera : 1.2.2a.

13.3.10 Liste de valeurs



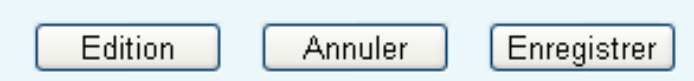
Les champs possédant des listes de valeurs sont gérés :

Par listBox si le nombre de valeurs n'est pas trop important.

Par popup dans le cas de listes de valeurs trop importantes (exemple : liste des communes). L'accès à cet écran de fera par l'icône placée à côté du champ. Il donnera l'accès à un écran de recherche.

Par Bouton radio dans le cas de valeurs non stockées en base et ne dépassant pas 3 choix possibles (exemple : choix du sexe).

13.3.11 Boutons



Tous les boutons seront de style « bouton standard » et de taille 80 pixels.

Ils seront positionnés en bas à droite de la page dans l'ordre suivant, de droite à gauche.

13.3.12 Champ date

L'utilisateur pourra :

Saisir directement la date

Choisir la date via le petit calendrier à droite du champ.

Le format de saisie des dates de sera JJ/MM/AAAA.

13.3.13 Navigation

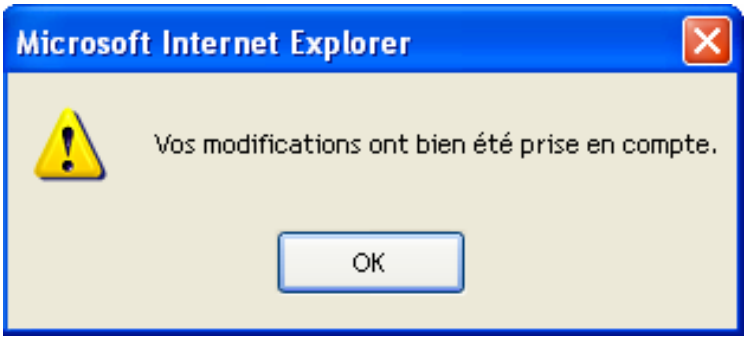
Tous les champs de l'écran ainsi que les boutons peuvent être accessibles via le clavier. La navigation de champs en champs se fera tabulation (TAB) pour avancer et par MAJ+TAB pour reculer. L'ordonnancement des champs par la touche tabulation suit un ordre logique et s'organise par GroupBox.

13.3.14 Règles transactionnelles

Sur le bouton Enregistrer

N'afficher aucun message de confirmation.

Afficher un message indiquant que les modifications ont été effectuées.



13.3.15 Template pour les pages web

En plus des modèles standards installés, disponibles dans les boîtes de dialogue « Nouveau projet » et « Ajouter un nouvel élément », vous pouvez accéder aux modèles que l'équipe de la TMA a pût être amené a développer. En effet, de façon générale, un Template personnalisé peut répondre à certaines problématiques telles que :

- utiliser une MasterPage particulière
- hériter d'une page de base
- inclure un cartouche de commentaires XML (nom utilisateur, date)
- faire référence aux namespace des autres couches de votre application (data, métier, ...)
- présenter un début de charte graphique dans leur <asp:Content> (un cadre et un titre)
- implémenter certaines méthodes d'initialisation
- …

13.3.15.1 Installation du Template

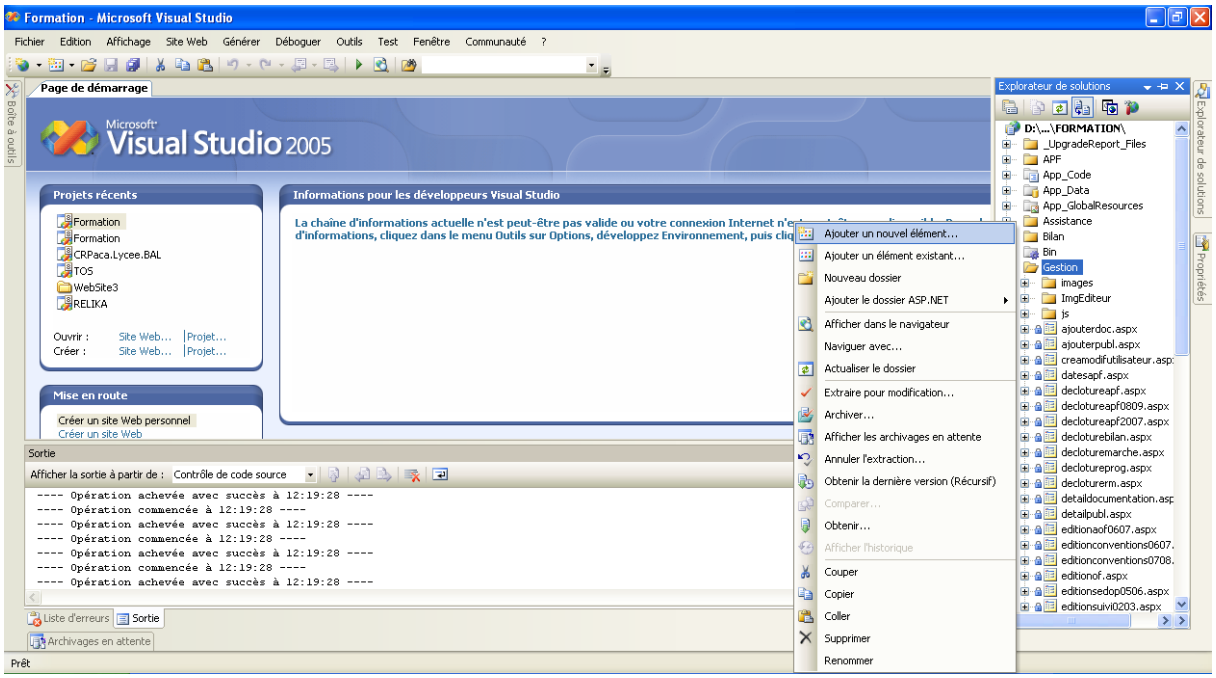
Pour avoir accès au template « CRPaca Web Form », il suffit de copier le fichier « O:\DSI\TMA\COMMUN\SOPRA\Capitalisation C#\CRPacaWebForm.zip » sur chaque poste de développement dans le répertoire de destination suivant : « C:\Documents and Settings\<CompteWindows>\Mes documents\Visual Studio 2005\Templates\ItemTemplates\Visual C# »

13.3.15.2 Création d'une nouvelle page web

Un « assistant » a donc été mis en place afin de faciliter la création de nouvelles pages web sous Microsoft Visual Studio .NET 2005.

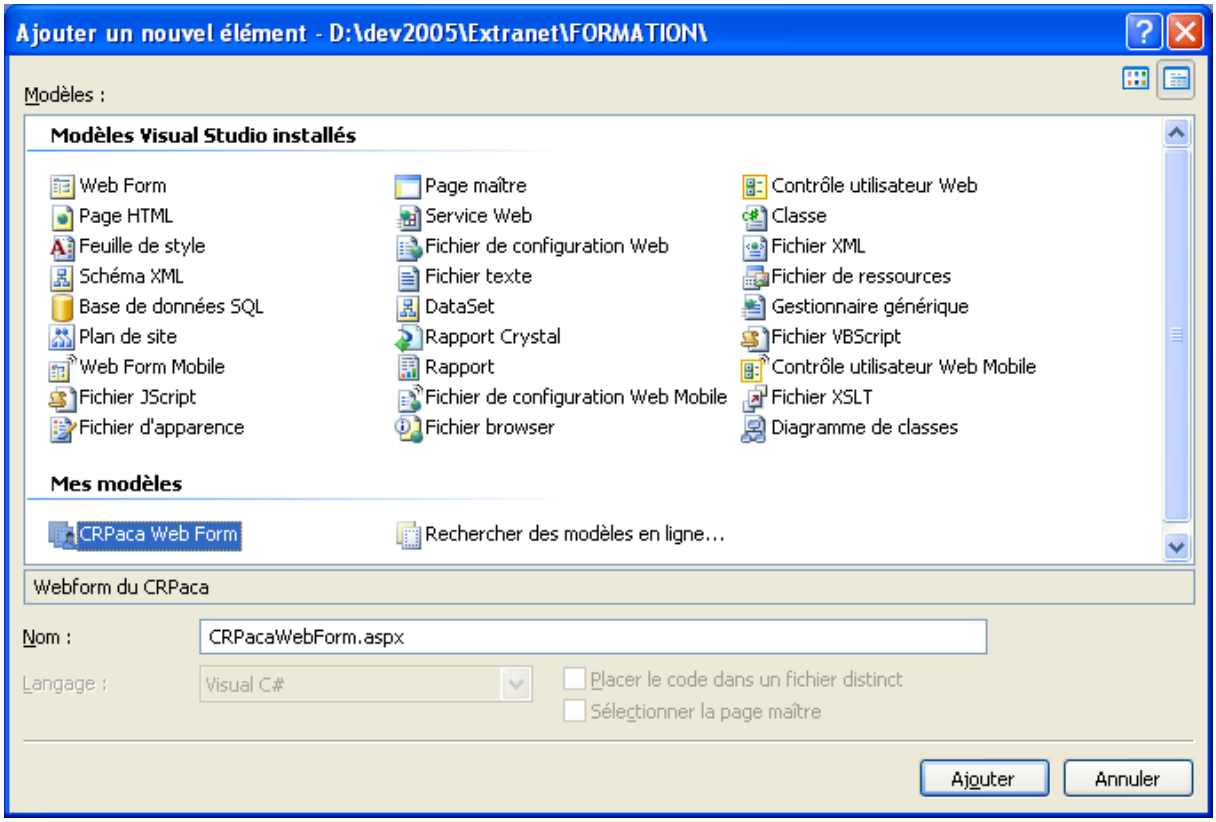
Dans l'explorateur de solutions, faire un click droit à l'endroit où doit être ajouté la page.

Sélectionner le menu « Ajouter », puis « Ajouter un nouvel élément »



Sélectionner « CRPaca Web Form »

Saisir le nom du fichier aspx puis cliquer sur « Ouvrir »



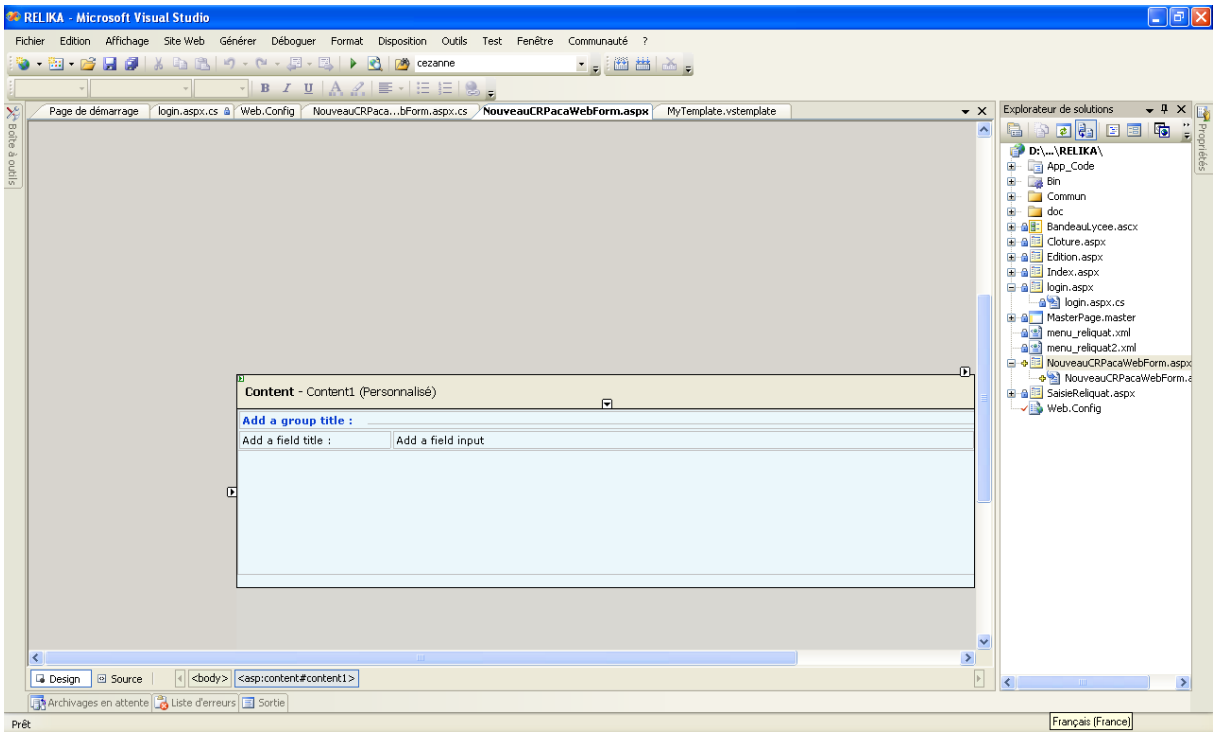
L'éditeur de Visual Studio affiche alors une page web type préconstruite avec :

- la Master Page en fond de page (comprenant : le bandeau du haut, le pied de page et le menu). Cette Master Page n'est pas modifiable à partir de cet emplacement.

- la zone réservée au formulaire de saisie, qui correspond au design de la page et qui peut bien évidemment être modifiée.

La page ainsi créée possède déjà le lien vers la Master Page et le « Look and Feel » du CR Paca.

Le dégradé de bleu présent sur le bandeau du haut et du bas ne s'affiche pas ici mais est présent à l'exécution de la page dans Internet Exploreur



13.3.15.3 Entête et pied de page

Le bandeau principal et le pied de page du formulaire web ainsi crée est « rempli » par le biais d'un composant (web control). Ce composant récupère un « bloc » de code html à une adresse donnée, et l'insère dans une page web. Ainsi, toutes les pages crées par cet assistant feront appel aux mêmes blocs html pour ces 2 parties de la page. Ces sources html pouvant être gérés par la DSI, on dispose d'un mécanisme de mise à jour automatique pour ces 2 zones sur l'ensemble du site.

13.3.15.4 Menus

Le développeur peut personnaliser le contenu (et la mise en forme) du menu de gauche et du menu du haut grâce à un fichier de XML de configuration. Le menu fait appel à ce fichier grâce à la fenêtre « propriétés » de Visual Studio.

Là aussi, ces menus sont réalisés par le biais de composants. Si la mise en forme de ces composant doit changer de manière globale, seulement une mise à jour du composant permettra de répercuter sur l'ensemble du site cette nouvelle mise en forme.

13.3.15.5 Styles

La page modèle importe aussi quelques feuilles de styles en cascades (css) par défaut :

Une feuille de style propre aux menus.

Une feuille de style propre à l'entête principale.

Une feuille de style propre au pied de page.

Une feuille de style dit « de base » pour l'ensemble des contenus de l'extranet.

L'ensemble de ces feuilles de styles, permet de réaliser l'ergonomie qui a été établie. Ces feuilles définissent quelques balises html, mais surtout contiennent des classes de styles que le développeur de pages doit utiliser. D'ailleurs le template en utilise déjà une bonne partie.

Bien entendu, si des styles supplémentaires sont nécessaires pour une page donnée, il est tout à fait possible d'inclure une nouvelle feuille de style.

14 DOCUMENT 6 – 1.01 – ERGONOMIE DE L'INTERFACE CLIENT

14.1 Introduction

14.1.1 Objectifs du document

Le présent document a pour objectif de définir les règles d'ergonomie pour les développements d'applications en client riche et en client léger (pages web).

14.1.2 Utilisation de la norme

La norme doit être utilisée obligatoirement dans tous les nouveaux projets.

Pour tous les projets antérieurs, le développement doit suivre les normes du projet.

Pour ne pas respecter la norme définie dans ce dossier, il est obligatoire de préciser dans le cahier des charges la norme à appliquer sur le projet.

14.1.3 Domaine d'application de la norme

Tous les développements pour la Région PACA fait en C# ou ASP avec le FRAMEWORK .NET 2.0.

14.2 Applications clients riches

14.2.1 Police – tailles des fenêtres – position des fenêtres - couleur d'écran

14.2.1.1 Règles

La police utilisée est MS sans sérif de taille 10.

La taille de la fenêtre principale est de 1280*1024 pixels.(Taille du MdiContainer).
Les formulaires qui sont donc contenus dans celle ci sont plus petits et s'appuient sur le template de fenêtre CRPacaNewForm (dont la taille est 1130*830).

L'écran doit être capable d'afficher toutes les informations.

Les fenêtres doivent toujours être positionnées en haut à gauche, et non se placer de façon relative plus bas à droite de la fenêtre précédente.

La couleur de l'écran est Gray.

Le nombre de fenêtres ouvertes simultanément est limité à 5.

L'application doit lister les fenêtres actives dans un menu dédié.

14.2.1.2 Organisation

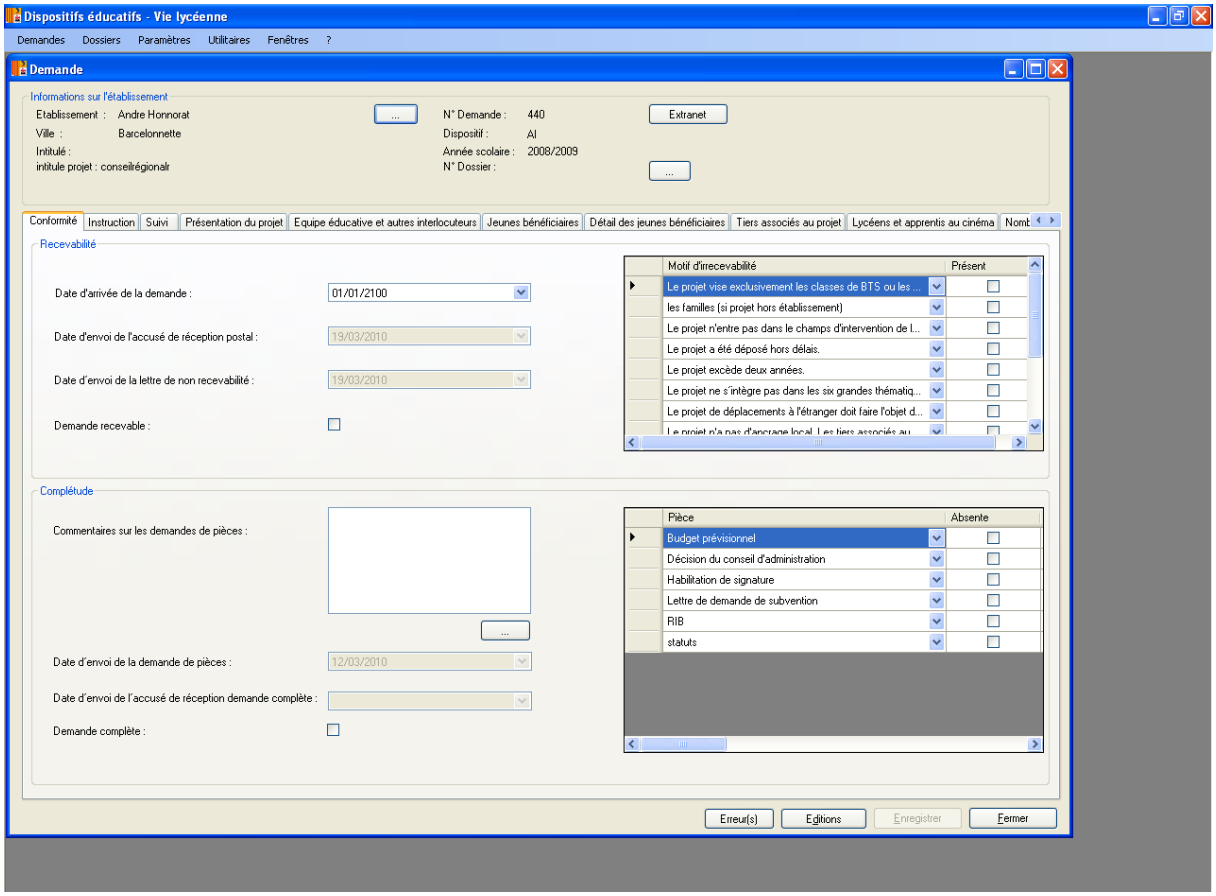
Les fenêtres sont organisées en trois parties :

Entête (non modifiable) contenant les informations résumant l'entité gérée par la fenêtre.

Corps (modifiable) : contient toutes les informations gérée par la fenêtre.

Ligne d'actions : contient tous les boutons d'actions (Edition, Enregistrer …).

14.2.1.3 Exemple

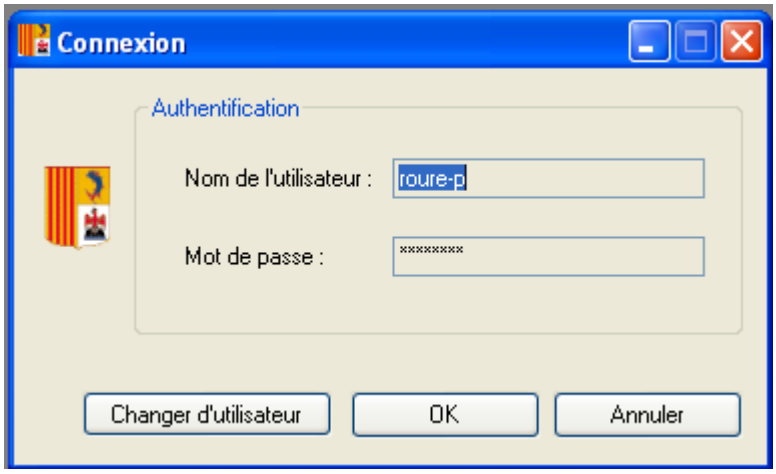


14.2.2 Fenêtre de connexion

14.2.2.1 Règles

Les boutons pour accéder à l'application sont « Ok » ou « Annuler » pour sortir ou « Changer d'utilisateur » pour se connecter avec un autre utilisateur que celui du compte Windows.

14.2.2.2 Exemple



14.2.3 Menu

14.2.3.1 Règles

Le menu est uniquement métier. Aucune fonction du style « Edition → Copier/Coller ».

Selon le profil de la personne connectée certaines parties du menu seront grisées (inaccessibles). Le menu « Paramètres » sera accessible uniquement pour les personnes désignées comme administrateur de l'application.

L'entrée de chaque module applicatif sera effectuée par un écran de recherche.

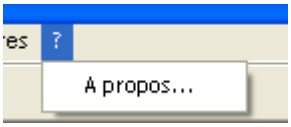
Le Menu est à mettre dans une fenêtre WinMenuGeneral dérivant de sopragroup.accelerate.module.CRPacaMenuGeneral.

14.2.3.2 Barre d'outils

Plus de barre d'outils comme il était utilisé précédemment dans les applications client serveur. Les fonctions enregistrer, ajouter, supprimer, liste de valeur seront gérées via des boutons.

14.2.3.3 Versions

Affichage du « ? » permettant d'afficher le nom de l'écran, son application et sa version sur 3 chiffres.



La version sera de la forme **chiffre.chiffre.chiffre*Lettre en minuscule.***

Exemple : 1.0.0a

Chaque modification d'une application en production entraînera l'évolution de son numéro de version.

- Le premier chiffre représente la livraison d'une application dans sa globalité. Il changera uniquement dans le cas d'évolutions majeures ou de modifications impactant l'application dans sa globalité (cas d'une migration technique).

- Le second représente un module ou un lot de l'application.
- Le troisième représente un livraison intermédiaire de ce lot.
- La lettre représente des corrections de bugs ou des petites évolutions (rajout d'une virgule, changement de libellé, déplacement d'un champ, etc.).

Exemple 1 : Livraison du module Gestion dans la nouvelle application PRV2 développée en .NET.

Sa version initiale sera : 1.1.1a.

Si correction de bugs mineurs, la version passera en 1.1.1b, puis 1.1.1c, etc.

S'il y a beaucoup de Bugs à corriger ou une évolution, correction importante à effectuer sur le lot 1 alors on regroupera la livraison non plus sous une lettre mais dans une livraison intermédiaire d'un lot/module. L'application passerait alors en 1.1.2a.

Exemple 2 : Livraison d'un autre module dans la nouvelle application PRFV2. Ce module sera livré en 2 étapes.

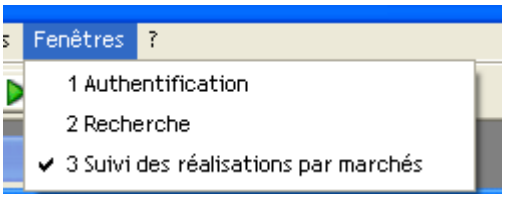
La version de l'application à la première étape sera : 1.2.1a.

La version de l'application à la deuxième étape sera : 1.2.2a.

La fenêtre A PROPOS n'est pas à faire, il existe une propriété dans la fenêtre CRPACAMenuGeneral pour ces changements.

14.2.3.4 Menu fenêtre

Affichage d'un menu fenêtre permettant de passer d'une fenêtre à une autre.



14.2.4 Gestion des DATAGRID

14.2.4.1 Formatage des données affichées

14.2.4.1.1 Données obligatoires

Garder le format de la donnée. Exemple : une date obligatoire doit être définie dans le DATATABLE comme étant de type date.

Les montants sont de type DECIMAL avec un formatage de la colonne en N2.

14.2.4.1.2 Données non obligatoires

Toutes les données non obligatoires doivent être formatées en STRING afin d'accepter les données NULL.

14.2.4.1.3 Formatage

Date : YYYY/MM/DD Ce n'est pas plutôt DD/MM/YYYY ?

Nombres : alignés à droite

Nombre réel : N2

Nombre entier : N0

14.2.4.2 Utilisation

Le tri doit être possible sur toutes les colonnes affichées (seulement pour les DATAGRID où les données sont affichées en ligne, et non pas pour ceux où les données sont affichées en colonne)

14.2.4.2.1 DATAGRID non modifiable et partiellement modifiable

Un DATAGRID non modifiable doit être en READONLY.

On ne doit pas pouvoir ajouter ou supprimer des lignes directement depuis le DATAGRID.

14.2.4.2.2 DATAGRID modifiable

Ajout illimité de lignes

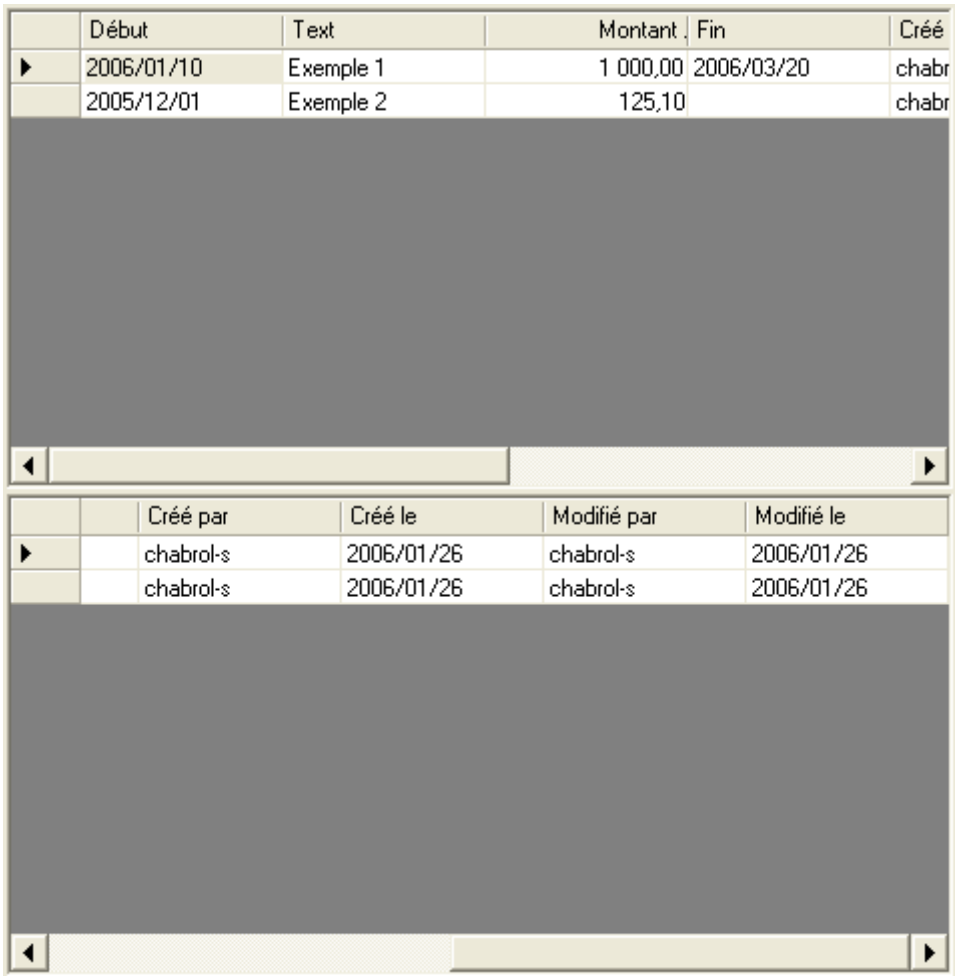
Suppression des lignes directement depuis le DATAGRID (hors problème technique).

14.2.4.2.3 Colonnes obligatoires dans tous les datagrids

Les 4 colonnes « Créé par », « Créé le », « Modifié par » et « Modifié le » sont systématiquement affichées dans tous les datagridset et doivent apparaitre en dernier. Le datagrid doit être organisé de telle sorte que ces 4 colonnes ne soient pas visibles immédiatement, sauf si on utilise l'ascenseur horizontal. Ces colonnes ne sont jamais modifiables.

La taille des colonnes est définie en fonction du nombre de données à afficher et de la longueur de la donnée à afficher.

14.2.4.3 Exemple



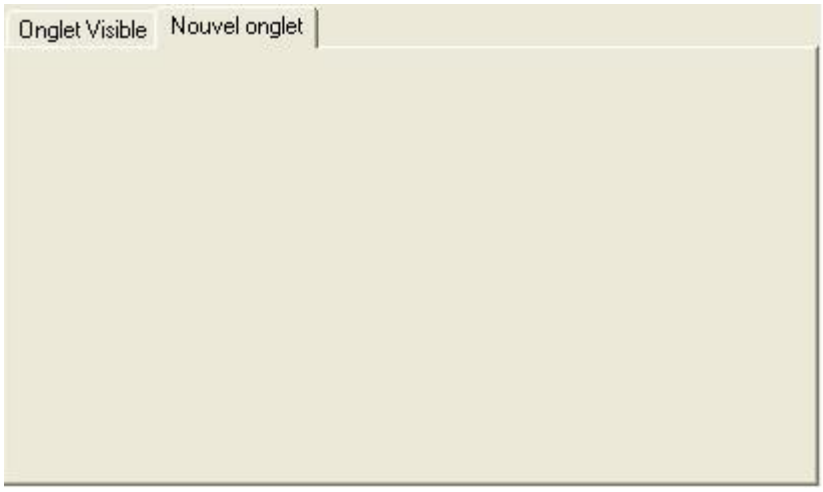
14.2.5 Gestion des onglets

14.2.5.1 Accessibilité

Seuls les onglets accessibles doivent être affichés. Pour ce faire il suffit de donner, au TABCONTROL, les onglets visibles.

Pour un nouvel enregistrement, seuls les onglets représentant l'entité principale sont affichés.

14.2.5.2 Exemples



14.2.6 Gestion des contrôles de saisie

14.2.6.1 Mise en page

Date : on doit pouvoir avoir une date NULL

Nombre : aligné à droite. Format N2 pour les montants sinon on utilisera le format N0.

Le premier élément d'un COMBOBOX doit être : « Choisissez… ».

14.2.6.2 Contrôles éditables

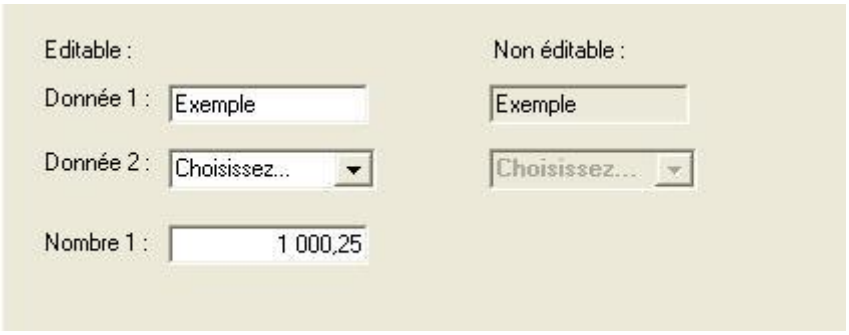
Nombre : reste aligné à droite par contre il faut enlever le formatage (N2 ou N0) lors de l'édition de la donnée.

14.2.6.3 Contrôles non éditables

TEXTBOX : doivent être en READONLY.

COMBOBOX et DATETIMEPICKER : ENABLE = FALSE, et le texte doit être en gras.

14.2.6.4 Exemple



14.2.7 Gestion des boutons

14.2.7.1 Les boutons

Les boutons d'Edition sont généralement placés à en bas à gauche sauf pour les écrans dédiés.

Les boutons actions sont placés en bas à droite. Le bouton FERMER est toujours le bouton le plus à droite. Les autres boutons sont placés par ordre d'importance et d'utilisation : le plus utilisé est à gauche, le moins utilisé est à coté du bouton FERMER.

Exemple : ENREGISTRER ; SUPPRIMER ; FERMER.

La taille des boutons doit être si possible de 95 * 24 pixels.

14.2.7.2 Accessibilité

Les boutons non utilisables doivent être « grisés ».

14.2.7.3 Exemples

Le bouton ENREGISTRER devient disponible

14.2.8 Messages et MESSAGEBOX

14.2.8.1 MESSAGEBOX

14.2.8.1.1 Barre des titres

Messages d'information : « Information ».

Messages d'alerte : « Attention »

Messages d'erreur : « Erreur »

14.2.8.1.2 Boutons

Messages d'erreur et d'information : 1 bouton « OK »

Messages d'alerte : 2 boutons « Oui » et « Non »

14.2.8.1.3 Icônes

Messages d'information : « MessageBoxIcon.Information »

Messages d'alerte : « MessageBoxIcon.Warning »

Messages d'erreur : « MessageBoxIcon.Error »

14.2.8.2 Messages

14.2.8.2.1 Changement de l'état de la fenêtre

Si l'utilisateur a fait des modifications dans la fenêtre sans avoir enregistré, afficher le message : « Voulez-vous enregistrer les modifications Oui/Non? »

Sinon n'afficher aucune message et sortir

14.2.8.2.2 Message avant la suppression

L'application doit demander une validation de l'action « supprimer » avant son exécution.

Le message est le suivant : « ATTENTION !!! Voulez-vous vraiment supprimer cet enregistrement ? ».

14.2.8.2.3 Autres messages

Message après l'échec d'une recherche : « Aucun résultat trouvé ».

Message lorsqu'il n'est pas possible d'ouvrir une fenêtre : « Vous ne pouvez avoir que 5 fenêtres actives ».

14.2.8.3 Exemples

14.2.9 Raccourcis

Conformément à la norme Microsoft, l'accès aux menus ou boutons sera accessible via des raccourcis claviers. Ceux-ci seront repérés par une lettre soulignée, en majuscule. L'accès sera alors possible par la combinaison des touches **Alt – Lettre**.

Exemples :

Alt N : a**N**nuler

Alt E : **E**nregistrer

Alt F : **F**ermer

Il faudra s'assurer que les raccourcis d'accès aux menus n'utilisent pas les raccourcis d'accès aux boutons.

Il sera possible, mais ce choix sera lié à l'application elle-même, de coder des raccourcis appelant directement un écran. L'accès sera alors possible par la combinaison des touches **Ctrl – lettre**.

Par exemple, dans l'application GPEA, l'appel à l'écran « n**O**uveau » sera possible par Ctrl -O.

Le menu sera alors, dans cet exemple, « Dossier » → « n**O**uveau – Ctrl O ».

14.2.10 Libellés

14.2.10.1 Règles

Tous les libellés sont suivis de « : ».

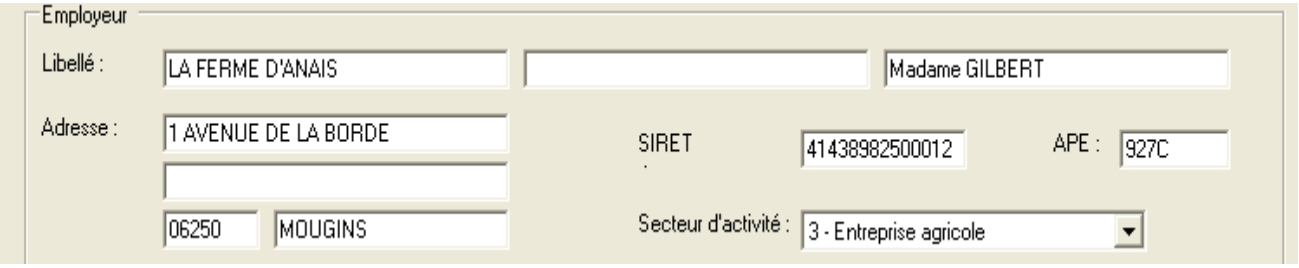
Ils sont alignés à gauche.

La première lettre est en majuscule.

Les mots ne sont pas coupés ou abrégés à part quelques abréviations bien définies. Par exemple :

Nom	**Abrégé**
Téléphone	Tél.
Numéro	N°

14.2.10.2 Exemples

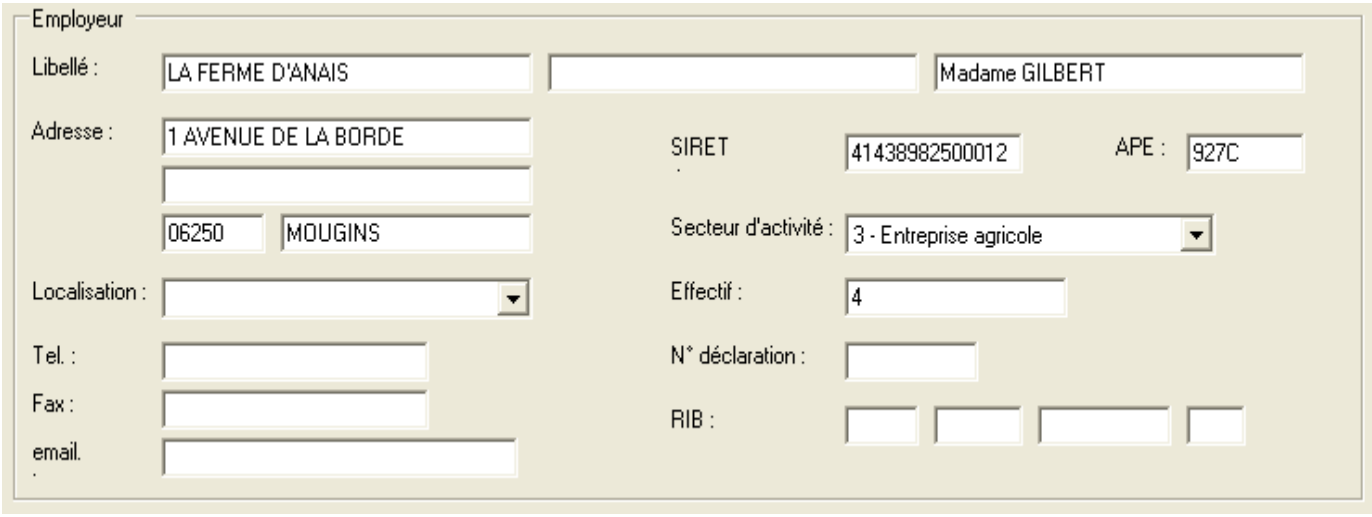


14.2.11 Champs

Ils sont alignés à gauche (voir ci-dessus).

Les champs non modifiables sont grisés et inaccessibles à l'utilisateur.

Les champs associés à une même notion sont regroupés dans un cadre (GroupBox).



14.2.12 Liste de valeurs

14.2.12.1 Règles

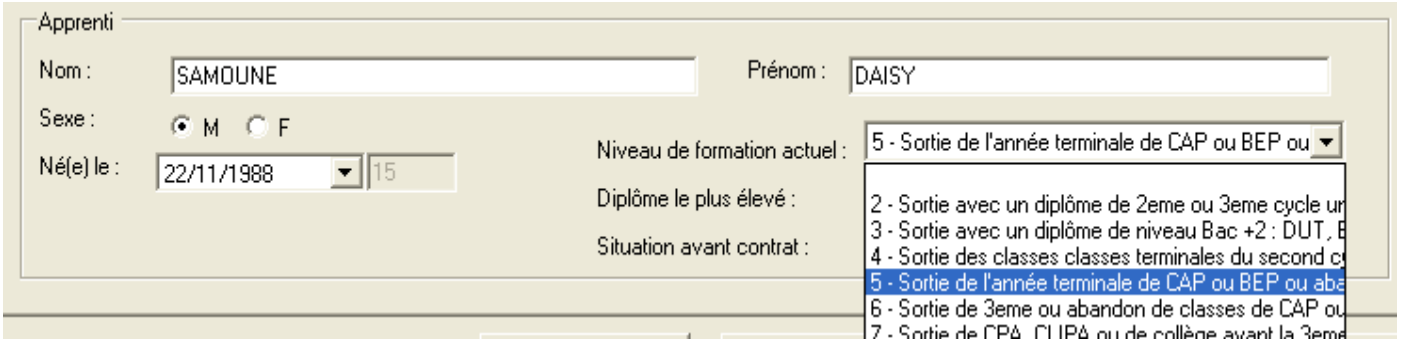
Les champs possédant des listes de valeurs sont gérés :

Par listBox si le nombre de valeurs n'est pas trop important.

Par popup dans le cas de listes de valeurs trop importantes (exemple : liste des communes). L'accès à cet écran de fera par l'icône placée à côté du champ. Il donnera l'accès à un écran de recherche.

Par Bouton radio dans le cas de valeurs non stockées en base et ne dépassant pas 3 choix possibles (exemple : choix du sexe).

14.2.12.2 Exemple



14.2.13 Navigation

Tous les champs de l'écran ainsi que les boutons peuvent être accessibles via le clavier. La navigation de champs en champs se fera tabulation (TAB) pour avancer et par MAJ+TAB pour reculer. L'ordonnancement des champs par la touche tabulation suit un ordre logique et s'organise par GroupBox.

14.2.14 Fenêtre de recherche

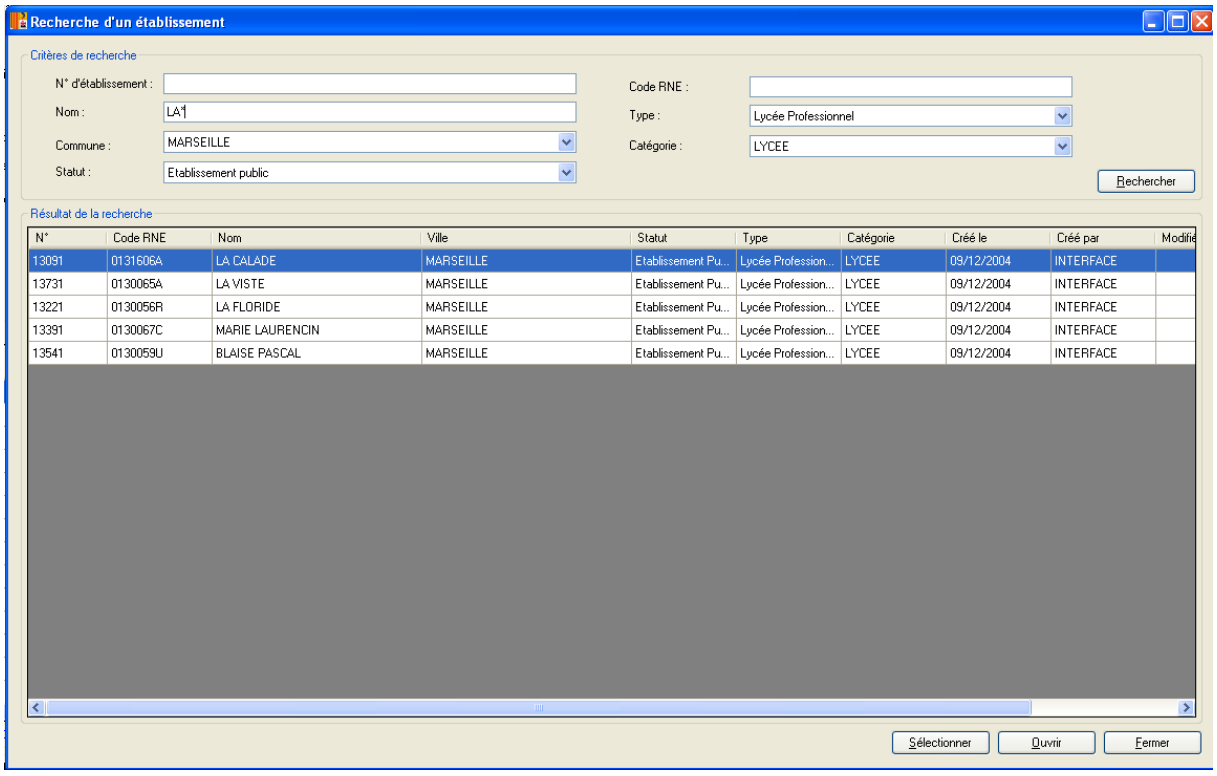
A chaque module développé doit être associé une fenêtre de recherche. Elle est composée de deux GroupBox :

Critères de sélection.

Résultat de la recherche = DataGrid.

Dans l'entête « critères de sélection » les utilisateurs renseignent un ou plusieurs de leurs critères de recherche. En cliquant sur le bouton **Rechercher**, ils déclenchent la recherche. Les résultats s'affichent dans la partie « Résultat de la recherche ». Un tri est possible par un « click » sur le nom de la colonne sur laquelle l'utilisateur souhaite effectuer un tri.

Cet écran est le point d'entrée de chaque module, l'accès aux autres écrans n'est, sinon, pas possible.



Dans le cas d'une recherche, le caractère indéfini sera représenté par « * ».

Pour accéder au détail d'une ligne, l'utilisateur peut :

Cliquer sur le bouton « modifier » ou « détail ».

Double-cliquer sur la ligne.

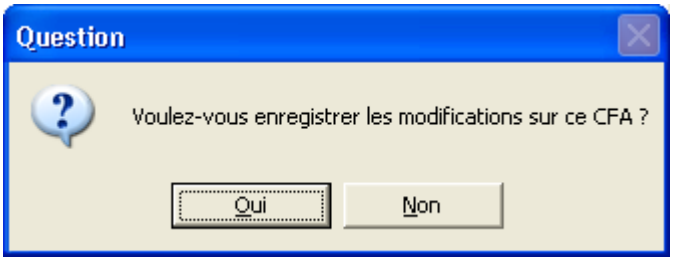
14.2.15 Règles transactionnelles

14.2.15.1 A la fermeture d'un écran

2 cas :

Aucune modification saisie : on sort sans message.

Une modification a été détectée, sans que l'utilisateur l'ait enregistrée : demander si on veut Enregistrer les modifications.



Si la réponse de l'utilisateur est :

Oui : On enregistre.

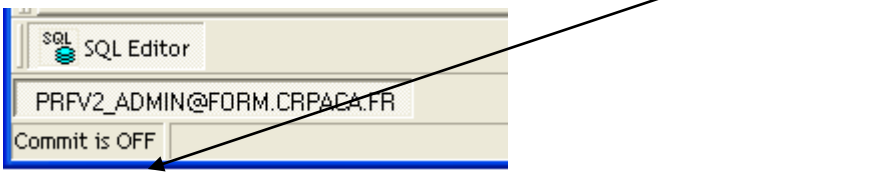
Non : On ignore les modifications.

14.2.15.2 Sur le bouton Enregistrer

N'afficher aucun message de confirmation.

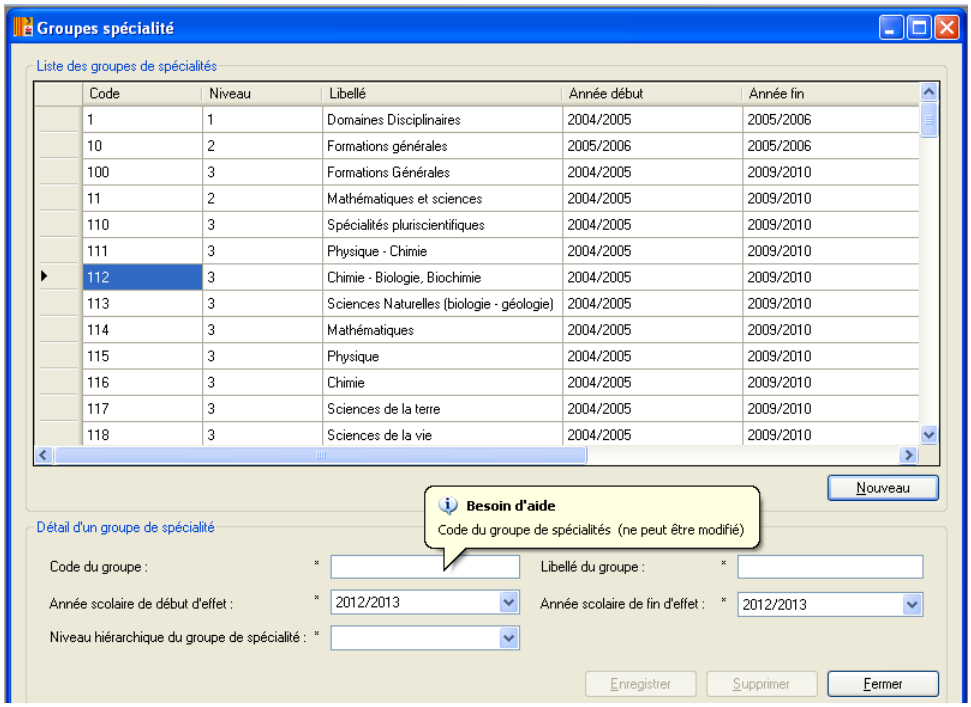
Si possible on affichera dans la barre d'avancement Windows, en bas à gauche, le message « Modification du dossier enregistrée».

Il faudra être attentif au rafraîchissement de cette de barre de tâches.



14.2.16 Ecrans de paramétrage

14.2.16.1 De style maître- détail « complexe »



Ajout d'une nouvelle ligne

Il faut utiliser pour cela le bouton « Nouveau ».

Lorsqu'on fait « nouveau », le bouton « enregistrer » ne doit pas être actif tant qu'on n'a rien saisi.

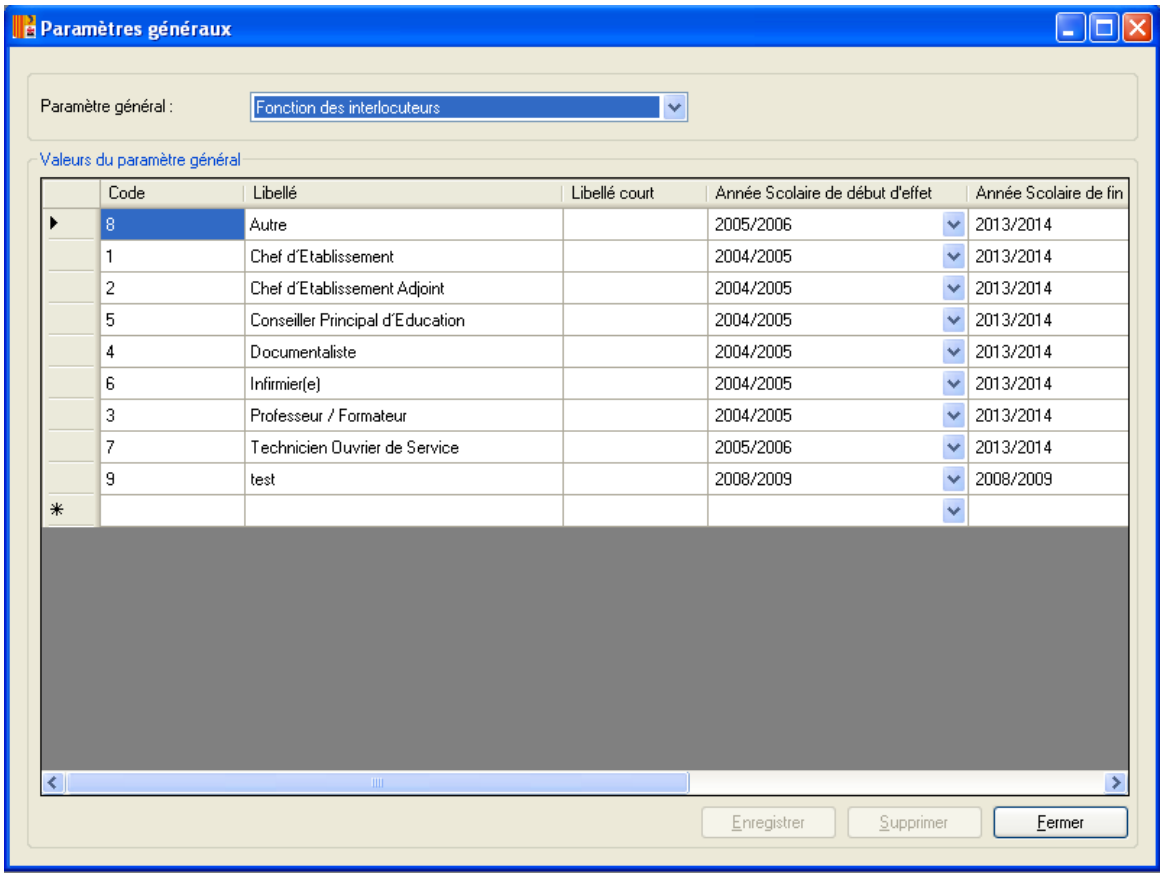
En création : positionner le curseur sur le premier champ de saisie.

Quand on change de paramètres : appliquer les mêmes règles que lors de la fermeture d'une fenêtre.

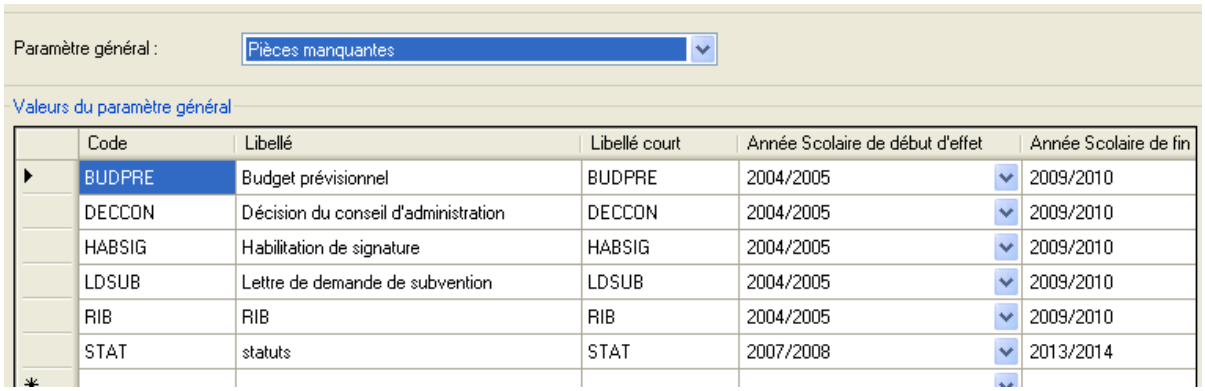
14.2.16.2 De style « simple »

Les paramètres représentés par un code et libellé pourront être regroupés dans un même écran « Paramètres généraux ».

Les boutons sur cet écran sont de droite à gauche, Fermer, Enregistrer, Annuler.

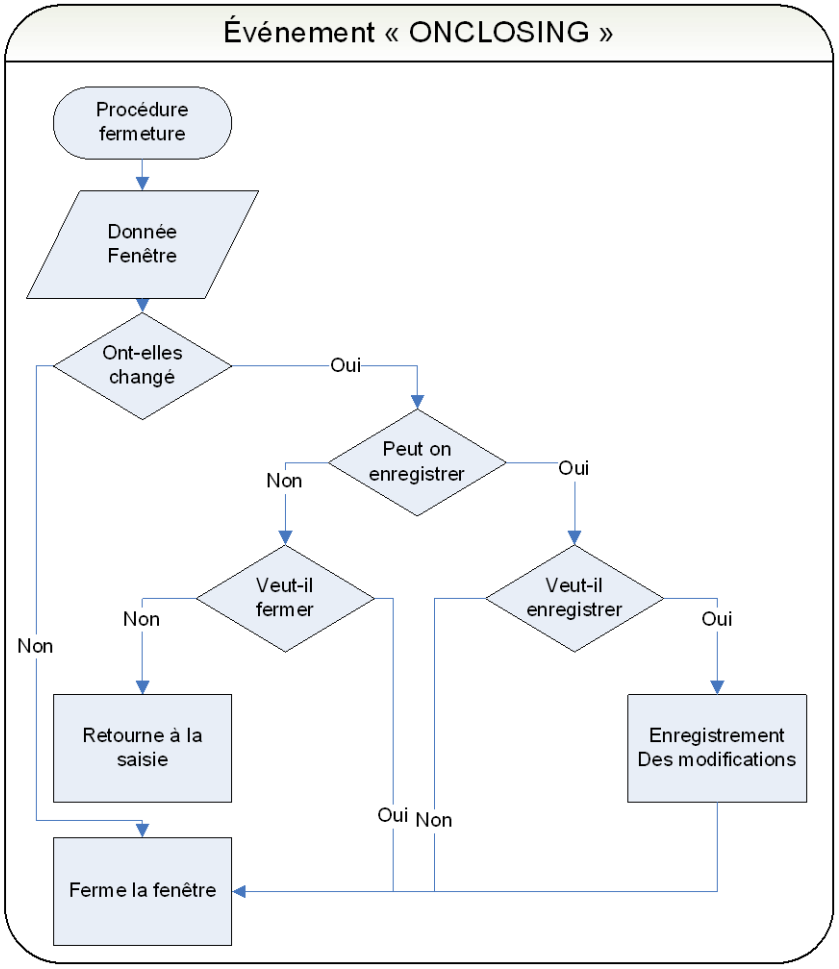


Autre choix de paramètres

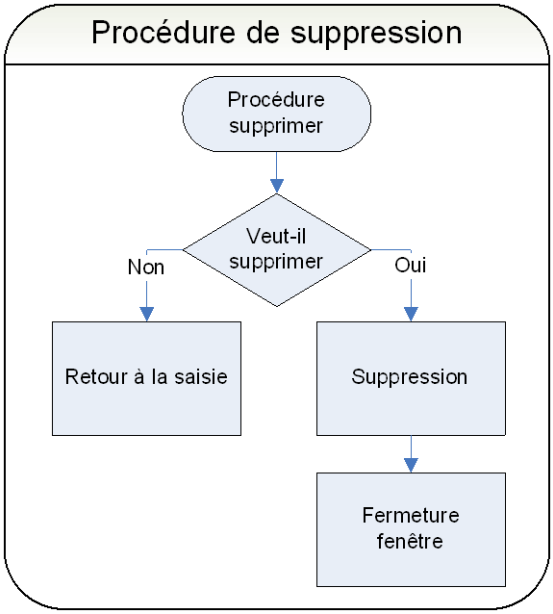


14.2.17 Procédures

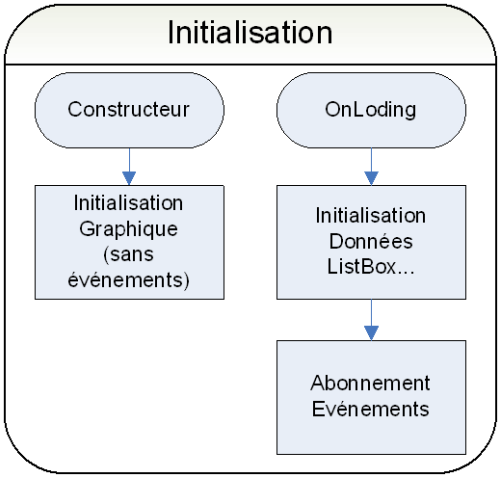
14.2.17.1 Procédure fermeture d'une fenêtre



14.2.17.2 Procédure de suppression

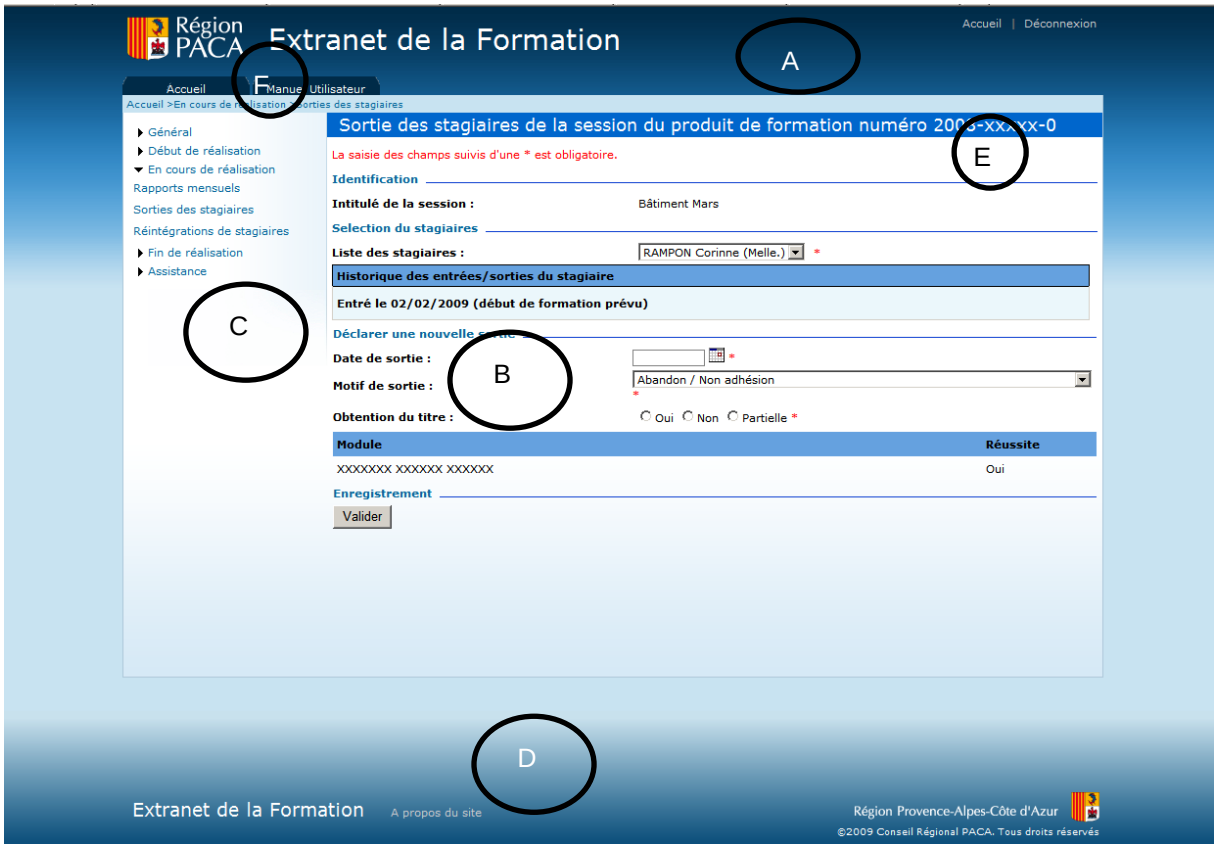


14.2.17.3 Procédure d'initialisation



14.3 Applications clients legers

14.3.1 Design général d'une page



La page peut-être découpée en 5 zones :

- En-tête de page - A
- Menu horizontal – F
- En-tête du module - E
- Menu - C
- Page Principale - B
- Pied de page – D

Les parties A, E, C, D et F sont apportés par la MASTER PAGE car ce sont des éléments fixes et communs à plusieurs pages. Il est préférable de ne pas dupliquer le code dans chaque page.

14.3.1.1 Décomposition

Un entête contenant :

- Le logo de la région
- Le sigle de l'application
- Le nom détaillé de l'application
- Le lien vers l'accueil
- Un lien de déconnexion
- Un menu horizontal, il doit contenir un lien vers le manuel utilisateur
- Un fil d'Ariane qui permet de connaître le chemin de la page actuelle

Le titre de la page

Un menu positionné à gauche

A droite du menu, la partie métier de la page

En bas :

- Un lien pour les contacts
- Un lien « A propos du site »
- Le copyright de la région.

14.3.1.2 Styles

Utilisation de la liste des styles CRPACA obligatoires.

Les styles les plus utilisés :

- Titre de la page : style « bandeauTitre »
- Le fond bleu ciel : style « formulaireBase »
- Les Labels de présentation des champs doivent être en gras
- Le titre des GROUPBOX : style « titreGroupBox » avec l'ajout d'une ligne bleu (<IMG height="1" src="/Commun/Images/1plblue.gif" width="100%">).
- Le dégradé du bas de page : style « gradient7 ».

14.3.2 Gestion des données

14.3.2.1 Affichage

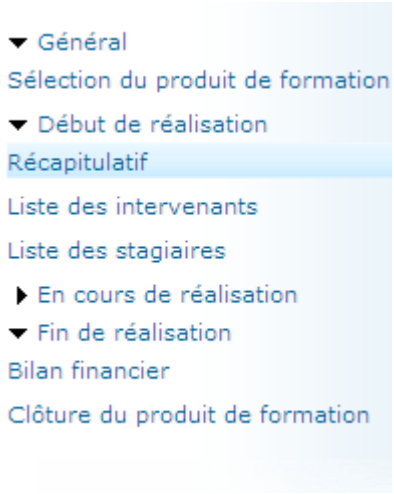
Utilisation des mêmes règles de formatage que les clients riches. (Exemple, Nombre : alignement à droite, formatage N2 ou N0).

14.3.2.2 Enregistrement, suppression…

Affichage des mêmes messages que ceux des clients riches (§ 2.6). Affichage du message informant l'utilisateur de la réussite de l'action : « L'action a été exécutée avec succès ».

14.3.3 Police – tailles des fenêtres – surbrillance – sélection d'un champ

La police utilisée est le Verdana de taille 8. L'écran doit être capable d'afficher toutes les informations dans une résolution de 1024 x 768. Le passage de la souris sur un item entraîne la surbrillance de celui-ci.



La surbrillance est effectuée par une image dégradée de bleu.

Afin d'indiquer à l'utilisateur la page sur laquelle il se trouve, celle-ci sera identifiée par une surbrillance « blanche ».

14.3.4 Fenêtre de connexion

L'accès au portail ne nécessite pas de connexion contrairement à l'accès aux différentes pages du site.

Le bouton pour accéder à l'application est « Valider ».

14.3.5 Zone « En-tête de la page » - A

Cette zone sera visible quelque soit la page appelée. Elle présente :

En haut à gauche le logo de la Région. Un clic sur celui-ci dirigera l'utilisateur vers le site de la Région (www.regionpaca.fr).

Au centre, le titre du portail. Un clic sur celui-ci ci dirigera l'utilisateur vers le portail.

En haut à droite, l'accueil de l'extranet. Un clic sur celui-ci dirigera l'utilisateur vers le portail.

14.3.6 Zone « Menu horizontal » - F

Cette zone dépend est composée :

D'un menu permettant d'accéder à l'accueil de l'application et au manuel utilisateur. Le style utilisé pour afficher cette partie du menu est le style MenuHorizontal.
D'un Fil d'Ariane permettant à tout moment de revenir sur l'accueil du module et de mettre à disposition, quelque soit la navigation à l'intérieur du module, le chemin de toutes les pages accédées. Le style utilisé pour afficher le composant est FilAriane.

14.3.7 Zone « En-tête du module » - E

Cette zone dépend du module sur lequel se trouvera l'utilisateur. Elle est composée d'un bandeau indiquant le titre du module. Les caractéristiques sont :

Police : Verdana 14

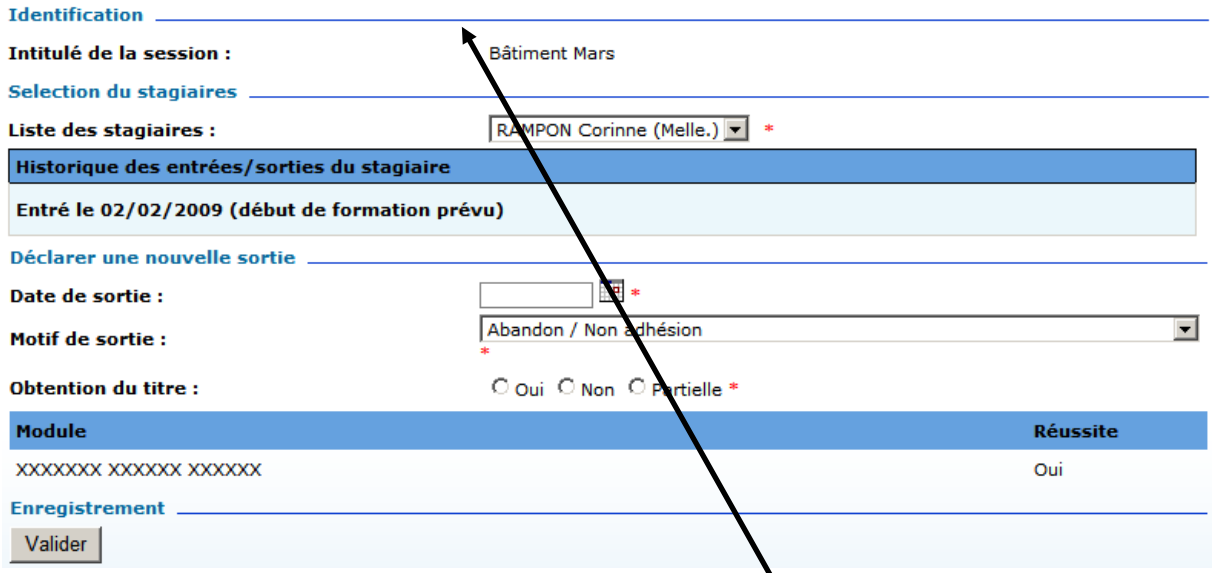
Couleur : Blanche

Couleur du bandeau : #0066cc

14.3.8 Zone « Menu » - C

Police : Verdana 8 (gras et non gras)

14.3.9 Zone « Page Principale » - B



Les champs associés à une même notion sont regroupés sous un même entête. (GroupBox).

Le fond de la page sera transparent (pour laisser apparaître le dégradé).

La police sera : Verdana 8.

Le police du titre du GroupBox est Verdana gras de 8, la couleur est la #093CD0, le trait est de la même couleur (#093CD0).

Chaque libellé sera suivi d'un blanc puis de « : ».

Les libellés sont alignés à gauche.

La première lettre est en majuscule.

Les champs non modifiables sont grisés et inaccessibles à l'utilisateur.

Les tableaux auront les caractéristiques suivantes :

Entête :

Police : Verdana 8

Couleur : noir

Les lignes du tableau

Police : Verdana 8

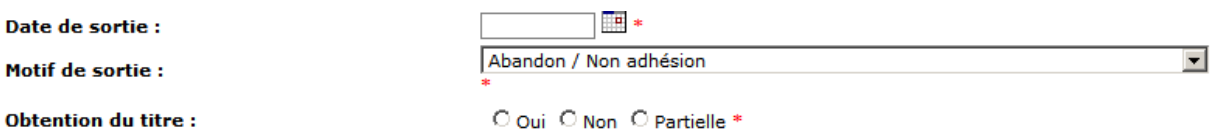
Couleur : noir

Les lignes seront de couleur :

Blanches.

bleu pale : #EBF7FB.

Les champs obligatoires seront suivi du « * »



Les mots ne sont pas coupés ou abrégés à part quelques abréviations bien définies. Par exemple :

Nom	**Abrégé**
Téléphone	Tél.
Numéro	N°

14.3.10 Zone « Pied de page » – D

Cette zone sera visible quelque soit la page appelée. Elle présente :

La couleur de fond de ce bandeau est un dégradé, le style du conteneur est nommé PiedDePage.

Le lien **contact** permet de générer un mail à une adresse spécifiée.

A propos du site permet l'ouverture d'une pop-up indiquant le numéro de version de l'application.

La version sera de la forme **chiffre.chiffre.chiffre*Lettre en minuscule.***

Exemple : 1.0.0.0a

Chaque modification d'une application en production entraînera l'évolution de son numéro de version.

Le premier chiffre représente la livraison d'une application dans sa globalité. Il changera uniquement dans le cas d'évolutions majeures ou de modifications impactant l'application dans sa globalité (cas d'une migration technique).

Le second représente un module ou un lot de l'application.

Le troisième représente une livraison intermédiaire de ce lot.

La lettre représente des corrections de bugs ou des petites évolutions (rajout d'une virgule, changement de libellé, déplacement d'un champ, etc. …).

Exemple 1 : Livraison du module Gestion dans la nouvelle application PRV2 développée en .NET.

Sa version initiale sera : 1.1.1a.

Si correction de bugs mineurs, la version passera en 1.1.1b, puis 1.1.1c, etc. …

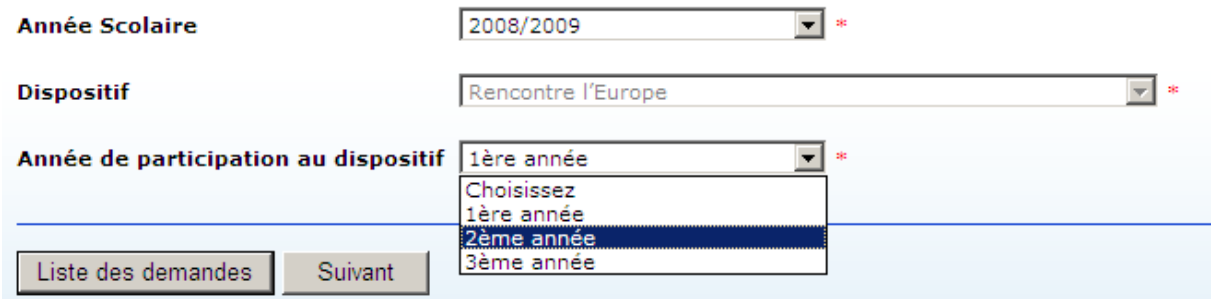
S'il y a beaucoup de Bugs à corriger ou une évolution, correction importante à effectuer sur le lot 1 alors on regroupera la livraison non plus sous une lettre mais dans une livraison intermédiaire d'un lot/module. L'application passerait alors en 1.1.2a.

Exemple 2 : Livraison d'un autre module dans la nouvelle application PRFV2. Ce module sera livré en 2 étapes.

La version de l'application à la première étape sera : 1.2.1a.

La version de l'application à la deuxième étape sera : 1.2.2a.

14.3.11 Liste de valeurs



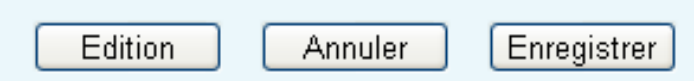
Les champs possédant des listes de valeurs sont gérés :

Par listBox si le nombre de valeurs n'est pas trop important.

Par popup dans le cas de listes de valeurs trop importantes (exemple : liste des communes). L'accès à cet écran de fera par l'icône placée à côté du champ. Il donnera l'accès à un écran de recherche.

Par Bouton radio dans le cas de valeurs non stockées en base et ne dépassant pas 3 choix possibles (exemple : choix du sexe).

14.3.12 Boutons



Tous les boutons seront de style « bouton standard » et de taille 80 pixels.

Ils seront positionnés en bas à droite de la page dans l'ordre suivant, de droite à gauche.

14.3.13 Champ date

L'utilisateur pourra :

Saisir directement la date

Choisir la date via le petit calendrier à droite du champ.

Le format de saisie des dates de sera JJ/MM/AAAA.

14.3.14 Navigation

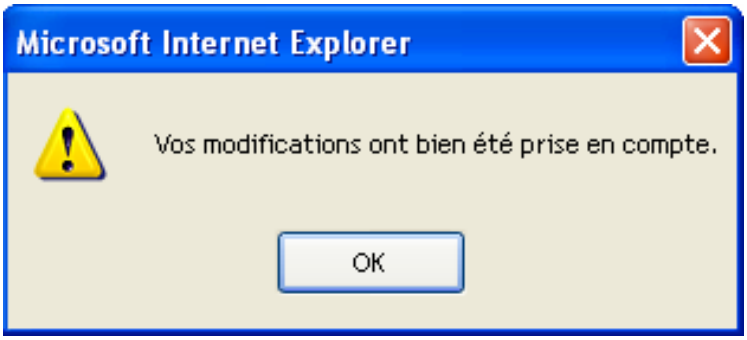
Tous les champs de l'écran ainsi que les boutons peuvent être accessibles via le clavier. La navigation de champs en champs se fera tabulation (TAB) pour avancer et par MAJ+TAB pour reculer. L'ordonnancement des champs par la touche tabulation suit un ordre logique et s'organise par GroupBox.

14.3.15 Règles transactionnelles

Sur le bouton Enregistrer

N'afficher aucun message de confirmation.

Afficher un message indiquant que les modifications ont été effectuées.



14.3.16 Template pour les pages web

En plus des modèles standards installés, disponibles dans les boîtes de dialogue « Nouveau projet » et « Ajouter un nouvel élément », vous pouvez accéder aux modèles que l'équipe de la TMA a pû être amené à développer. En effet, de façon générale, un Template personnalisé peut répondre à certaines problématiques telles que :

- utiliser une MasterPage particulière
- hériter d'une page de base
- inclure un cartouche de commentaires XML (nom utilisateur, date)
- faire référence aux namespace des autres couches de votre application (data, métier, ...)
- présenter un début de charte graphique dans leur <asp:Content> (un cadre et un titre)
- implémenter certaines méthodes d'initialisation
- …

14.3.16.1 Installation du Template

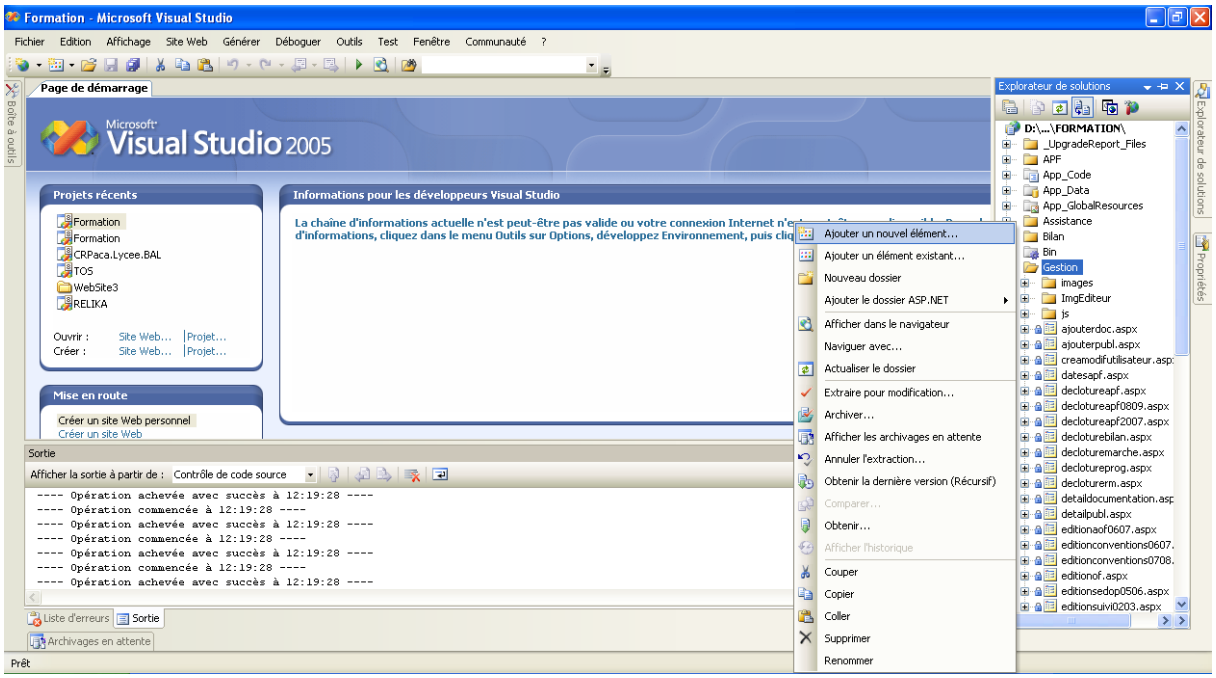
Pour avoir accès au template « CRPaca Web Form », il suffit de copier le fichier « O:\Espaces Departementaux\TMA\COMMUN\SOPRA\Capitalisation\Capitalisation C#\CRPacaWebForm.zip » sur chaque poste de développement dans le répertoire de destination suivant : « C:\Documents and Settings\<CompteWindows>\Mes documents\Visual Studio 2005\Templates\ItemTemplates\Visual C# »

14.3.16.2 Création d'une nouvelle page web

Un « assistant » a donc été mis en place afin de faciliter la création de nouvelles pages web sous Microsoft Visual Studio .NET 2005.

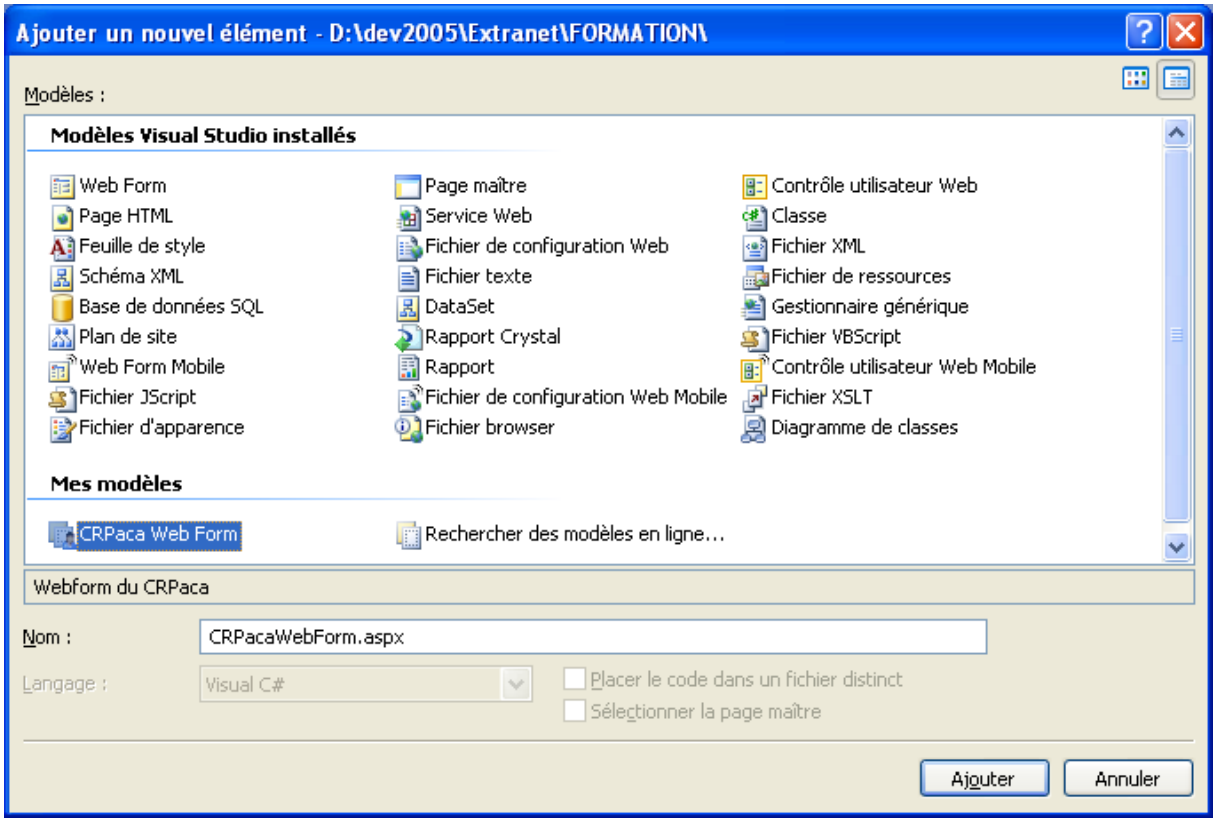
Dans l'explorateur de solutions, faire un click droit à l'endroit où doit être ajouté la page.

Sélectionner le menu « Ajouter », puis « Ajouter un nouvel élément »



Sélectionner « CRPaca Web Form »

Saisir le nom du fichier aspx puis cliquer sur « Ouvrir »



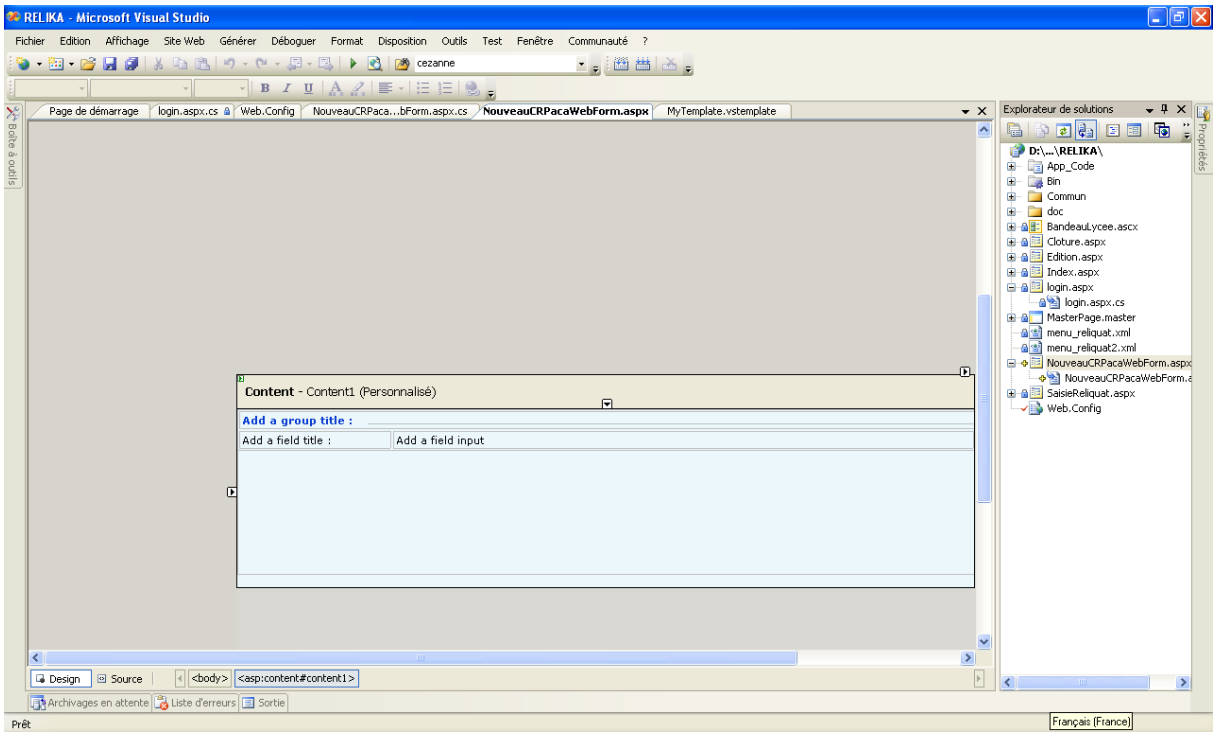
L'éditeur de Visual Studio affiche alors une page web type préconstruite avec :

- la Master Page en fond de page (comprenant : le bandeau du haut, le pied de page et le menu). Cette Master Page n'est pas modifiable à partir de cet emplacement.

- la zone réservée au formulaire de saisie, qui correspond au design de la page et qui peut bien évidemment être modifiée.

La page ainsi créée possède déjà le lien vers la Master Page et le « Look and Feel » du CR Paca.

Le dégradé de bleu présent sur le bandeau du haut et du bas ne s'affiche pas ici mais est présent à l'exécution de la page dans Internet Explorer



14.3.16.3 Entête et pied de page

Le bandeau principal et le pied de page du formulaire web ainsi crée est « rempli » par le biais d'un composant (web control). Ce composant récupère un « bloc » de code html à une adresse donnée, et l'insère dans une page web. Ainsi, toutes les pages créées par cet assistant feront appel aux mêmes blocs html pour ces 2 parties de la page. Ces sources html pouvant être gérés par la DSI, on dispose d'un mécanisme de mise à jour automatique pour ces 2 zones sur l'ensemble du site.

14.3.16.4 Menus

Le développeur peut personnaliser le contenu (et la mise en forme) du menu de gauche et du menu du haut et du fil d'Ariane grâce à un fichier siteMap de configuration. Le menu fait appel à ce fichier grâce à la fenêtre « propriétés » de Visual Studio : on fixe la DataSourceID :

(Expressions)	
DataBindings	(Collection)
DataSourceID	**SiteMapSource**

Comportement	
EnableViewState	True
ShowStartingNode	**False**
SiteMapProvider	**DefaultProvider**
StartFromCurrentN	False
StartingNodeOffsel	0
StartingNodeUrl	
Divers	
(ID)	**SiteMapSource**

Dans le Web.config global on pourra déclarer pour chaque module un siteMapProvider :

```
<siteMap defaultProvider="DefaultProvider">
    <providers>
        <add name="DefaultProvider" type="System.Web.XmlSiteMapProvider"
      siteMapFile="~/Web.sitemap" />
        <add name="SuiviAFPA2009Provider" type="System.Web.XmlSiteMapProvider"
             siteMapFile="~/Suivi/SuiviAFPA2009/Web.sitemap"/>
    </providers>
</siteMap>
```

Et suivant le module auquel on se connecte charger le bon provider (en conservant le même menu) dans la masterpage.

Le fait d'utiliser des composants .Net Natif va nous permettre d'uniformiser la gestion de notre composant sur l'ensemble du site.

14.3.16.5 Styles

La page modèle importe aussi quelques feuilles de styles en cascades (css) par défaut :

Une feuille de style propre aux menus.

Une feuille de style dit « de base » pour l'ensemble des contenus de l'extranet.

L'ensemble de ces feuilles de styles, permet de réaliser l'ergonomie qui a été établie. Ces feuilles définissent quelques balises html, mais surtout contiennent des classes de styles que le développeur de pages doit utiliser. D'ailleurs le template en utilise déjà une bonne partie.

Bien entendu, si des styles supplémentaires sont nécessaires pour une page donnée, il est tout à fait possible d'inclure une nouvelle feuille de style.

15 DOCUMENT 6 – 2.00 – ERGONOMIE DE L'INTERFACE CLIENT

15.1 Introduction

15.1.1 Objectifs du document

Le présent document a pour objectif de définir les règles d'ergonomie pour les développements d'applications en client riche et en client léger (pages web).

15.1.2 Utilisation de la norme

La norme doit être utilisée obligatoirement dans tous les nouveaux projets.

Pour tous les projets antérieurs, le développement doit suivre les normes du projet.

Pour ne pas respecter la norme définie dans ce dossier, il est obligatoire de préciser dans le cahier des charges la norme à appliquer sur le projet.

15.1.3 Domaine d'application de la norme

Tous les développements pour la Région PACA sont faits en C# 4.0, WPF 4.0 ou ASP.NET pour la couche IHM avec le FRAMEWORK .NET 4.0.

15.2 Applications clients riches

15.2.1 Police – tailles des fenêtres – position des fenêtres - couleur d'écran

15.2.1.1 Règles

La police utilisée est Calibri de taille 10.

Le titre des fenêtres et fenêtres secondaires sont de taille 18 en Gras

La taille de la fenêtre principale est de 1280*1024 pixels. L'écran doit être capable d'afficher toutes les informations

La taille des fenêtres secondaires est de 1130*830 pixels.

Les fenêtres doivent toujours être positionnées en haut à gauche, et non se placer de façon relative plus bas à droite de la fenêtre précédente. Chaque fenêtre doit passer au premier plan lorsque l'utilisateur passe son curseur sur une partie de cette fenêtre si celle-ci est placée en arrière plan.

La couleur de fond de la fenêtre principale est Noir (#FF202020). Le thème de fond pour les fenêtres secondaires est un dégradé de Noir (#FF202020) vers du Gris (#FF545454).

Le nombre de fenêtres ouvertes simultanément est limité à 5.

L'application doit lister les fenêtres actives dans un menu dédié.

Les fenêtres secondaires apparaissent avec une animation de fondu.

15.2.1.2 Organisation

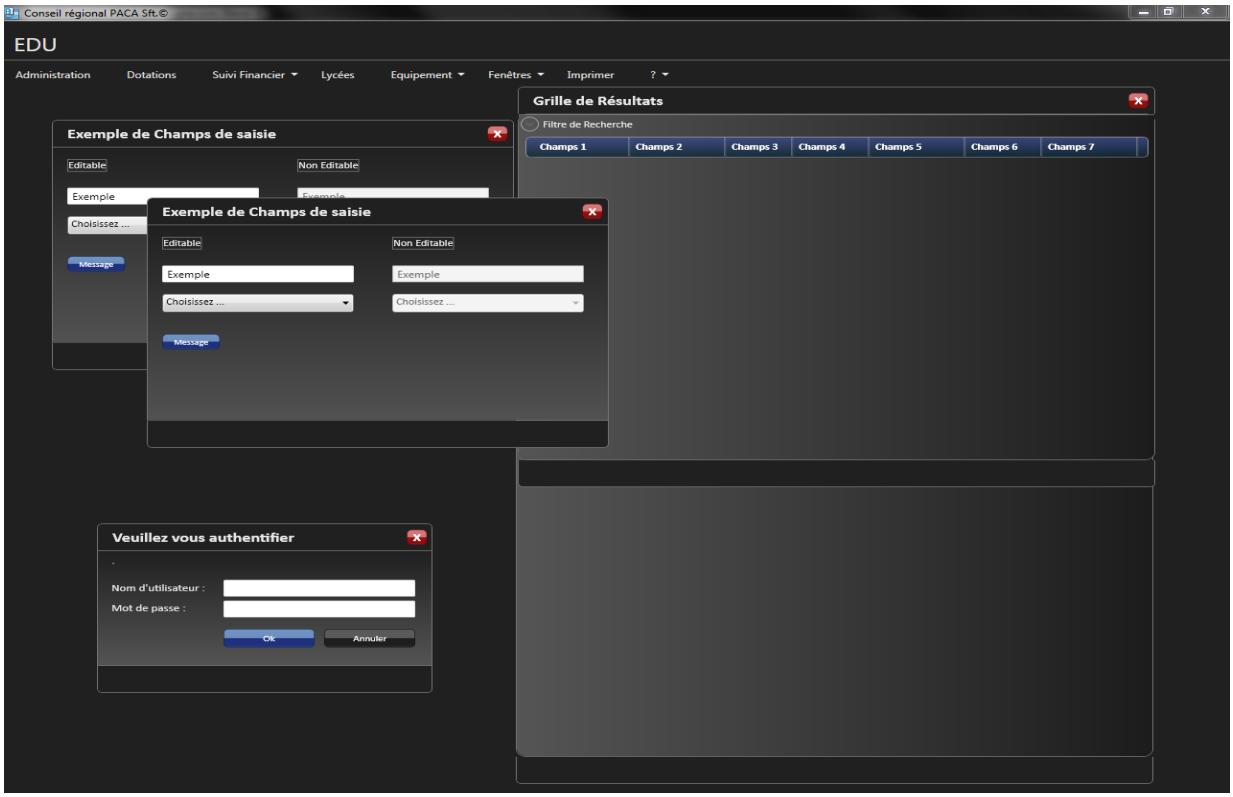
Les fenêtres sont organisées en trois parties :

Entête (non modifiable) contenant les informations résumant l'entité gérer par la fenêtre.

Corps (modifiable) : contient toutes les informations gérées par la fenêtre.

Ligne d'actions : contient tous les boutons d'actions (Edition, Enregistrer …).

15.2.1.3 Exemple

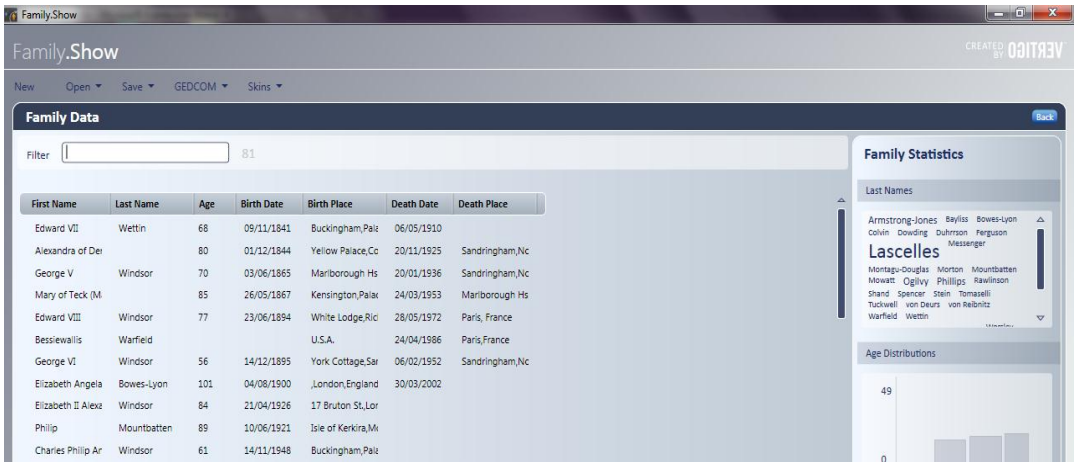


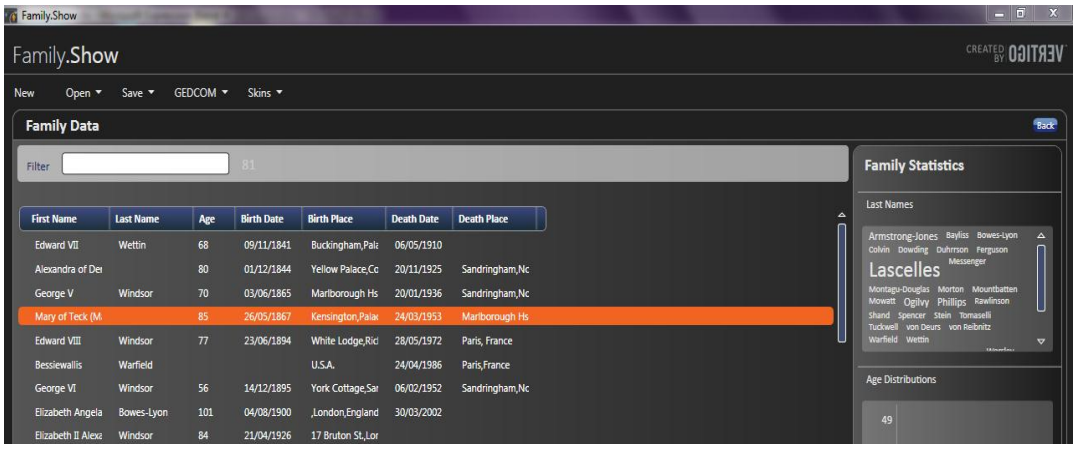
15.2.2 Changement de skin

15.2.2.1 Règles

Les applications client riche offrent la possibilité de changer de skin (thème) à l'utilisateur final. Il lui sera possible de passer du skin Black au skin Silver. Ces modifications ergonomiques sont sauvegardées par le système au prochain redémarrage du logiciel par cet utilisateur.

15.2.2.2 Exemple





Les différents thèmes sont décrits dans des fichiers de ressources WPF. Une seule implémentation est suffisante si les deux fichiers ressources sont à jour. Le changement de thème en « runtime » ne sera en fait que l'application de l'un ou l'autre des fichiers de ressources.

15.2.3 Fenêtre de connexion

15.2.3.1 Règles

Les boutons pour accéder à l'application sont « Ok » ou « Annuler » pour sortir ou « Changer d'utilisateur » pour se connecter avec un autre utilisateur que celui du compte Windows.

15.2.3.2 Exemple

15.2.4 Menu

15.2.4.1 Règles

Le menu est uniquement métier. Aucune fonction du style « Edition → Copier/Coller ».

Selon le profil de la personne connectée certaines parties du menu seront grisées (inaccessibles). Le menu « Paramètres » sera accessible uniquement pour les personnes désignées comme administrateur de l'application.

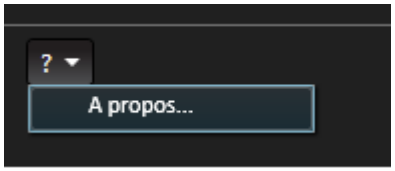
L'entrée de chaque module applicatif sera effectuée par un écran de recherche.

15.2.4.2 Barre d'outils

Plus de barre d'outils comme il était utilisé précédemment dans les applications client serveur. Les fonctions enregistrer, ajouter, supprimer, liste de valeur seront gérées via des boutons.

15.2.4.3 Versions

Affichage du « ? » permettant d'afficher le nom de l'écran, son application et sa version sur 3 chiffres.



La version sera de la forme **chiffre.chiffre.chiffre*Lettre en minuscule.***

Exemple : 1.0.0.0a

Chaque modification d'une application en production entraînera l'évolution de son numéro de version.

- Le premier chiffre représente la livraison d'une application dans sa globalité. Il changera uniquement dans le cas d'évolutions majeures ou de modifications impactant l'application dans sa globalité (cas d'une migration technique).
- Le second représente un module ou un lot de l'application.
- Le troisième représente une livraison intermédiaire de ce lot.

- La lettre représente des corrections de bugs ou des petites évolutions (rajout d'une virgule, changement de libellé, déplacement d'un champ, etc.).

Exemple 1 : Livraison du module Gestion dans la nouvelle application PRV2 développée en .NET.

Sa version initiale sera : 1.1.1a.

Si correction de bugs mineurs, la version passera en 1.1.1b, puis 1.1.1c, etc.

S'il y a beaucoup de Bugs à corriger ou une évolution, correction importante à effectuer sur le lot 1 alors on regroupera la livraison non plus sous une lettre mais dans une livraison intermédiaire d'un lot/module. L'application passerait alors en 1.1.2a.

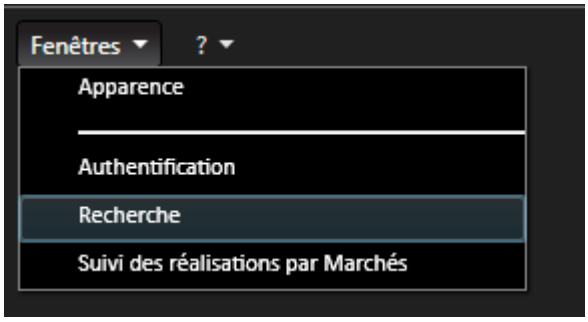
Exemple 2 : Livraison d'un autre module dans la nouvelle application PRFV2. Ce module sera livré en 2 étapes.

La version de l'application à la première étape sera : 1.2.1a.

La version de l'application à la deuxième étape sera : 1.2.2a.

15.2.4.4 Menu fenêtre

Affichage d'un menu fenêtre permettant de passer d'une fenêtre à une autre.



15.2.5 Gestion des DATAGRID

15.2.5.1 Formatage des données affichées

15.2.5.1.1 Données obligatoires

Garder le format de la donnée. Exemple : une date obligatoire doit être définie dans le DATATABLE comme étant de type date.

Les montants sont de type DECIMAL avec un formatage de la colonne en N2.

15.2.5.1.2 Données non obligatoires

Toutes les données non obligatoires doivent être formaté en STRING afin d'accepter les données NULL.

15.2.5.1.3 Formatage

Date : DD/MM/YYYY

Nombres : alignés à droite

Nombre réel : N2

Nombre entier : N0

15.2.5.2 Utilisation

Le tri doit être possible sur toutes les colonnes affichées (seulement pour les DATAGRID où les données sont affichées en ligne, et non pas pour ceux où les données sont affichées en colonne)

15.2.5.2.1 DATAGRID non modifiable et partiellement modifiable

Un DATAGRID non modifiable doit être en READONLY.

On ne doit pas pouvoir ajouter ou supprimer des lignes directement depuis le DATAGRID.

15.2.5.2.2 DATAGRID modifiable

Ajout illimité de lignes

Suppression des lignes directement depuis le DATAGRID (hors problème technique).

Une colonne modifiable pourra comporter dans son entête un pictogramme d'édition.

15.2.5.2.3 Colonnes obligatoires dans tous les datagrids

Les 4 colonnes « Créé par », « Créé le », « Modifié par » et « Modifié le » sont systématiquement affichées dans tous les datagridset et doivent apparaitre en dernier. Le datagrid doit être organisé de telle sorte que ces 4 colonnes ne soient pas visibles immédiatement, sauf si on utilise l'ascenseur horizontal. Ces colonnes ne sont jamais modifiables.

La taille des colonnes est définie en fonction du nombre de données à afficher et de la longueur de la donnée à afficher.

15.2.5.3 Format

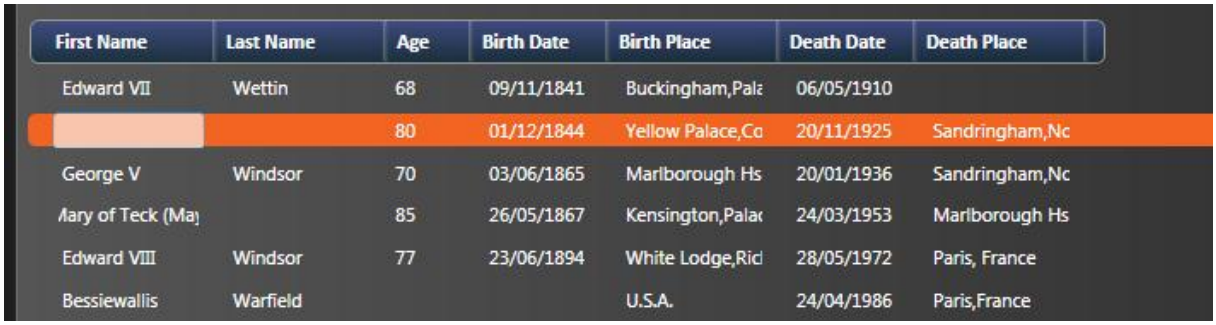
La couleur de fond du DataGrid sera par défaut celle du parent : Dégradé de Noir (#FF202020) vers du Gris (#FF545454).

La couleur de sélection d'une ligne est le orange #FFF16422.

Les entêtes de colonne doivent avoir une police blanche sur fond bleu. Les colonnes sont séparées par un espace de 5 pixels.

Les champs saisissables dans les datagrids doivent respecter les règles des champs de saisie présentée au paragraphe 2.6.

15.2.5.4 Exemple



L'écran ci-dessus montre un champ éditable dans la grille de résultats. Ce champ respecte les règles définis pour les champs de saisie à savoir la « surbrillance » lorsque le contrôle prend le focus et lorsque l'utilisateur survole le contrôle avec son curseur.

15.2.6 Gestion des onglets

15.2.6.1 Accessibilité

Seuls les onglets accessibles doivent être affichés. Pour ce faire il suffit de donner, au TABCONTROL, les onglets visibles.

Pour un nouvel enregistrement, seuls les onglets représentant l'entité principale sont affichés.

Le format des onglets doit respecter les règles de couleur des fenêtres secondaires :

15.2.6.2 Exemples

15.2.7 Gestion des contrôles de saisie

15.2.7.1 Mise en page

Date : on doit pouvoir avoir une date NULL

Nombre : aligné à droite. Format N2 pour les montants sinon on utilisera le format N0.

Le premier élément d'un COMBOBOX doit être : « Choisissez… ».

15.2.7.2 Contrôles éditables

Nombre : reste aligné à droite par contre il faut enlever le formatage (N2 ou N0) lors de l'édition de la donnée.

15.2.7.3 Contrôles non éditables

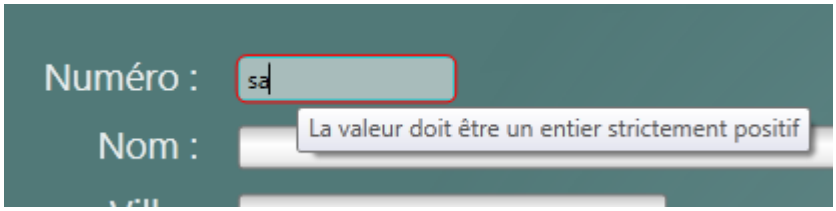
TEXTBOX : doivent être en READONLY.

COMBOBOX et DATETIMEPICKER : ENABLE = FALSE, et le texte doit être en gras.

15.2.7.4 Contrôles de Validation

Au moment de la saisie d'un champ, le système doit effectuer un contrôle sur le format attendu, et vérifie certaines règles de gestion.

Si une erreur apparaît au moment de la saisie le contrôle passe en mode erreur et un contour rouge doit apparaître. Le message associé à l'erreur s'affiche dans un Tooltip.



15.2.7.5 Exemples

15.2.8 Gestion des boutons

15.2.8.1 Les boutons

Les boutons d'Edition sont généralement placés à en bas à gauche sauf pour les écrans dédiés.

Les boutons actions sont placés en bas à droite. Le bouton FERMER sera placé en haut à droite de la fenêtre et aura l'apparence d'une croix blanche dans un bouton rouge.

Les autres boutons sont placés par ordre d'importance et d'utilisation : le plus utilisé est à gauche, le moins utilisé est le plus à droite de la liste des boutons.

15.2.8.2 Format

Les boutons « Enregistrer », « OK », « Oui » seront de couleur bleu.

Les boutons « Supprimer », « Quitter » seront de couleur rouge.

Les autres boutons seront de couleur noir.

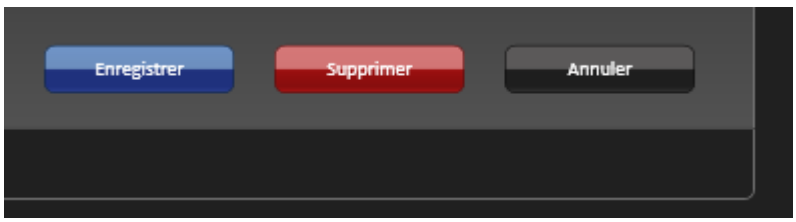
La taille des boutons doit être si possible de 95 * 24 pixels.

15.2.8.3 Accessibilité

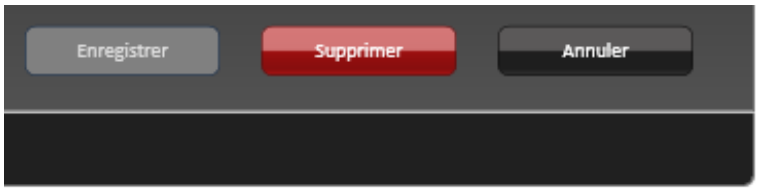
Les boutons non utilisables doivent être « grisés » ; C'est-à-dire sur fond gris.

15.2.8.4 Exemples

Les trois boutons sont actifs.



Dans le cas où le bouton « Enregistrer » est désactivé :



15.2.9 Messages et MESSAGEBOX

15.2.9.1 MESSAGEBOX

Le composant MessageBox utilisé dans les applications client riche est le contrôle natif du framework .NET 4.0. Il peut se configurer des façons suivantes :

15.2.9.1.1 Barre des titres

Messages d'information : « Information ».

Messages d'alerte : « Attention »

Messages d'erreur : « Erreur »

15.2.9.1.2 Boutons

Messages d'erreur et d'information : 1 bouton « OK »

Messages d'alerte : 2 boutons « Oui » et « Non »

15.2.9.1.3 Icônes

Messages d'information : « MessageBoxIcon.Information »

Messages d'alerte : « MessageBoxIcon.Warning »

Messages d'erreur : « MessageBoxIcon.Error »

15.2.9.2 Messages

15.2.9.2.1 Changement de l'état de la fenêtre

Si l'utilisateur a fait des modifications dans la fenêtre sans avoir enregistré, afficher le message : « Voulez-vous enregistrer les modifications Oui/Non? »

Sinon n'afficher aucun message et sortir

15.2.9.2.2 Message avant la suppression

L'application doit demander une validation de l'action « supprimer » avant son exécution.

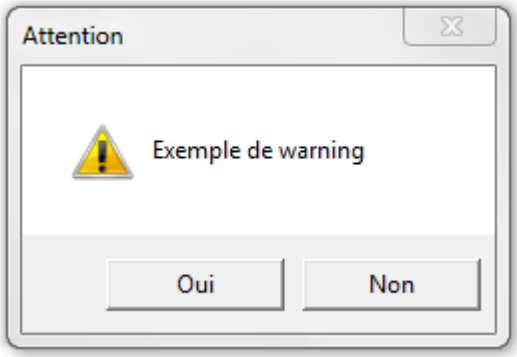
Le message est le suivant : « ATTENTION !!! Voulez-vous vraiment supprimer cet enregistrement ? ».

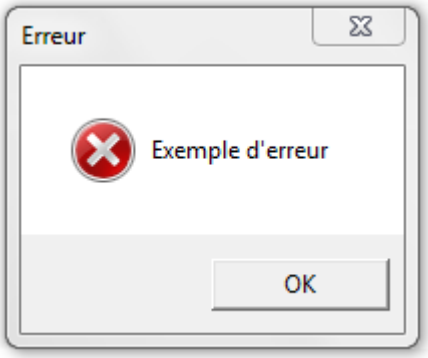
15.2.9.2.3 Autres messages

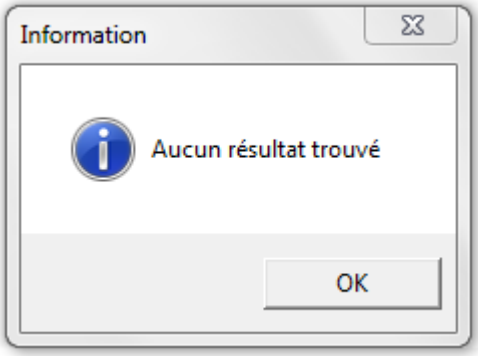
Message après l'échec d'une recherche : « Aucun résultat trouvé ».

Message lorsqu'il n'est pas possible d'ouvrir une fenêtre : « Vous ne pouvez avoir que 5 fenêtres actives ».

15.2.9.3 Exemples







15.2.10 Raccourcis

Conformément à la norme Microsoft, l'accès aux menus ou boutons sera accessible via des raccourcis claviers. Ceux-ci seront repérés par une lettre soulignée, en majuscule. L'accès sera alors possible par la combinaison des touches **Alt – Lettre**.

Exemples :

Alt N : a<u>N</u>nuler

Alt E : <u>E</u>nregistrer

Alt F : <u>F</u>ermer

Il faudra s'assurer que les raccourcis d'accès aux menus n'utilisent pas les raccourcis d'accès aux boutons.

Il sera possible, mais ce choix sera lié à l'application elle-même, de coder des raccourcis appelant directement un écran. L'accès sera alors possible par la combinaison des touches **Ctrl – lettre**.

Par exemple, dans l'application GPEA, l'appel à l'écran « n<u>O</u>uveau » sera possible par Ctrl -O.

Le menu sera alors, dans cet exemple, « Dossier » → « n<u>O</u>uveau – Ctrl O ».

15.2.11 Libellés

15.2.11.1 Règles

Tous les libellés sont suivis de « : ».

Ils sont alignés à gauche.

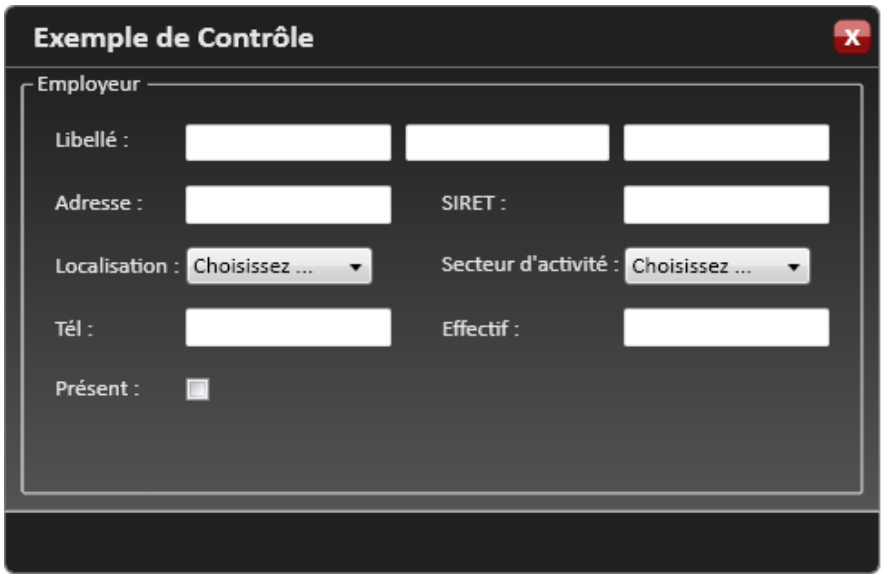
La première lettre est en majuscule.

Les mots ne sont pas coupés ou abrégés à part quelques abréviations bien définies. Par exemple :

Nom	**Abrégé**
Téléphone	Tél.

Numéro	N°

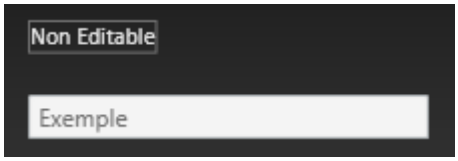
15.2.11.2 Exemples



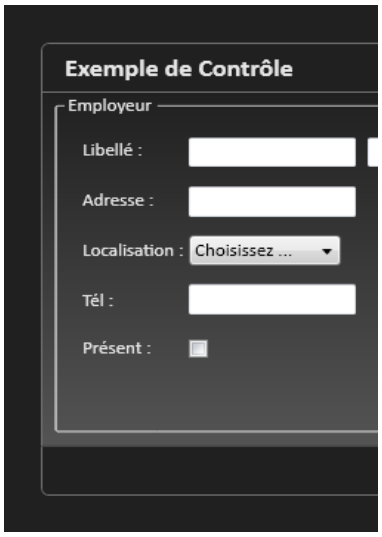
15.2.12 Champs

Ils sont alignés à gauche (voir ci-dessus).

Les champs non modifiables sont grisés et inaccessibles à l'utilisateur.



Les champs associés à une même notion sont regroupés dans un cadre (GroupBox).



Les champs de saisie doivent se griser lorsque l'utilisateur passe le curseur sur le champ et lorsque ce champ garde le focus :

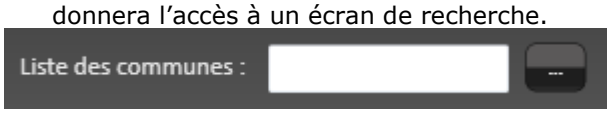
15.2.13 Liste de valeurs

15.2.13.1 Règles

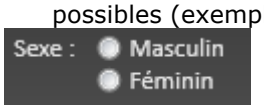
Les champs possédant des listes de valeurs sont gérés :

Par Combobox si le nombre de valeurs n'est pas trop important.

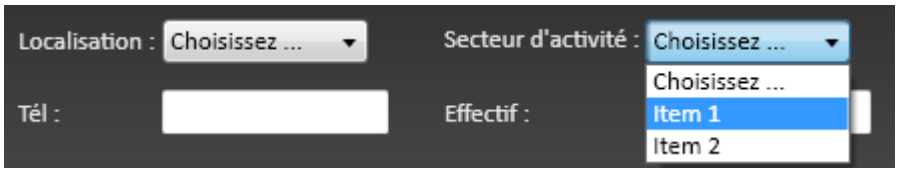
Par des fenêtres secondaires dans le cas de listes de valeurs trop importantes (exemple : liste des communes). L'accès à cet écran se fera par une icône placée à côté du champ. Il donnera l'accès à un écran de recherche.



Par Bouton radio dans le cas de valeurs non stockées en base et ne dépassant pas 3 choix possibles (exemple : choix du sexe).



15.2.13.2 Exemple



15.2.14 Navigation

Tous les champs de l'écran ainsi que les boutons peuvent être accessibles via le clavier. La navigation de champs en champs se fera tabulation (TAB) pour avancer et par MAJ+TAB pour reculer. L'ordonnancement des champs par la touche tabulation suit un ordre logique et s'organise par GroupBox.

15.2.15 Fenêtre de recherche

A chaque module développé doit être associé une fenêtre de recherche. Elle est composée de deux parties :

Critères de sélection.

Résultat de la recherche = DataGrid.

La partie « critères de sélection » ou « filtre de recherche » est cachée dans un composant « Expander » pour une meilleur lecture des résultats lorsque le nombre de critères de recherche est important. Ce composant sera affiché par défaut lors de l'ouverture de la fenêtre secondaire.

Dans l'entête « critères de sélection » les utilisateurs renseignent un ou plusieurs de leurs critères de recherche. En cliquant sur le bouton **Rechercher**, ils déclenchent la recherche. Les résultats s'affichent dans la partie « Résultat de la recherche ». Un tri est possible par un « click » sur le nom de la colonne sur laquelle l'utilisateur souhaite effectuer un tri.

La hauteur de la partie « résultats » est figé. Des « scrollbars » ou ascenseurs seront visibles en bas et à droite de la fenêtre de résultat pour faire défiler la grille.

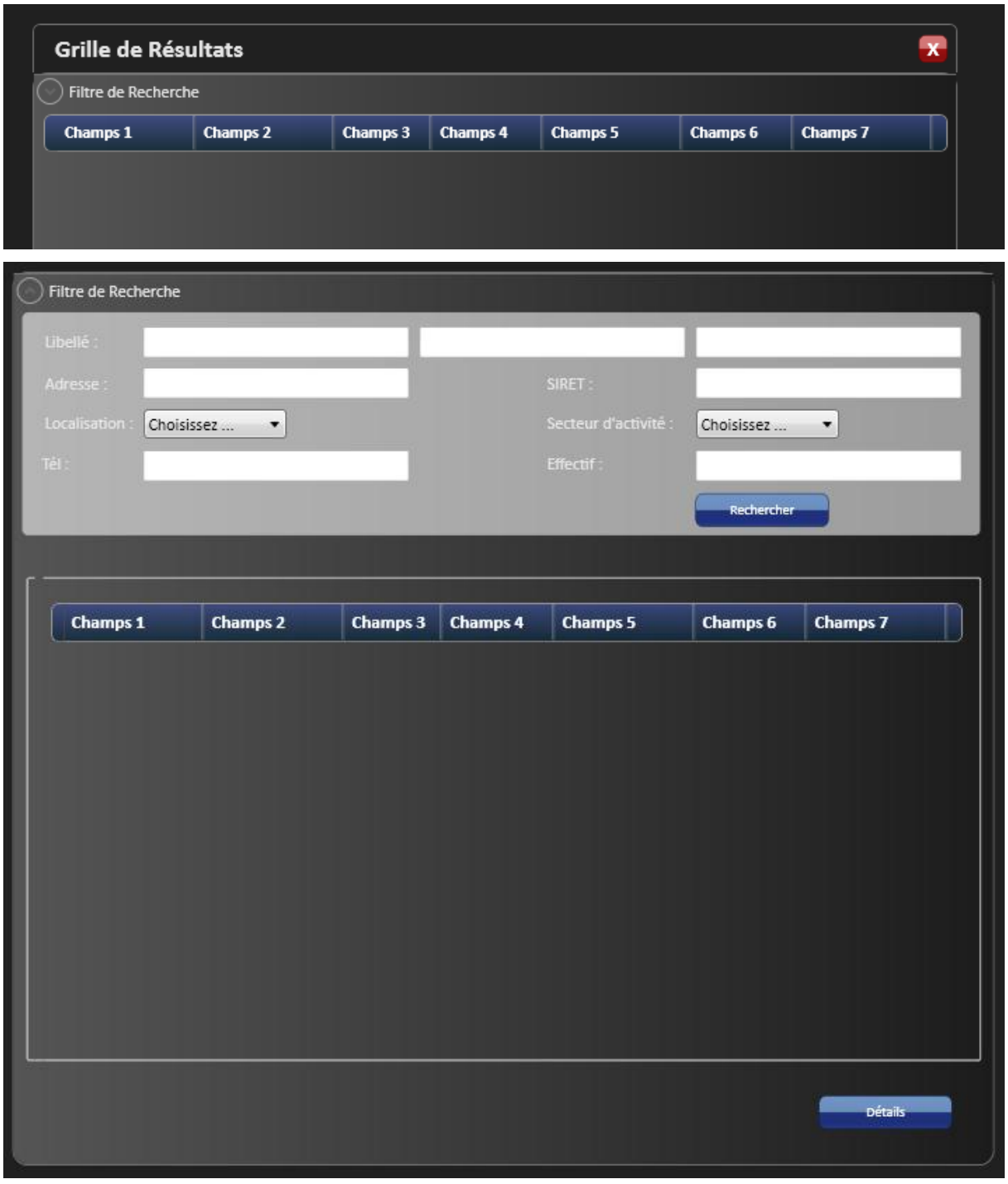
L'utilisation du CheckBox à trois valeurs sera permise uniquement dans les critères de recherche.

Valeur négative : 

Valeur indéterminée, qui sera la valeur par défaut dans les écrans de recherche : 

Valeur positive : 

Cet écran est le point d'entrée de chaque module, l'accès aux autres écrans n'est, sinon, pas possible.



Dans le cas d'une recherche, le caractère indéfini sera représenté par « * ».

Pour accéder au détail d'une ligne, l'utilisateur peut :

Cliquer sur le bouton « modifier » ou « détail ».

Double-cliquer sur la ligne.

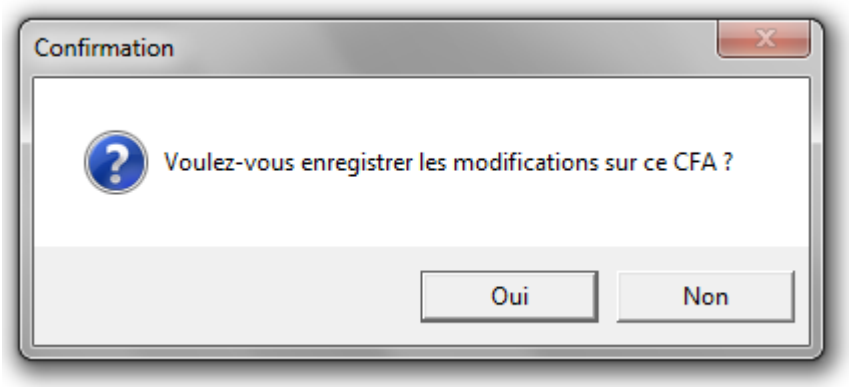
15.2.16 Règles transactionnelles

15.2.16.1 A la fermeture d'un écran

2 cas :

Aucune modification saisie : on sort sans message.

Une modification a été détectée, sans que l'utilisateur l'ait enregistrée : demander si on veut Enregistrer les modifications.



Si la réponse de l'utilisateur est :

Oui : On enregistre.

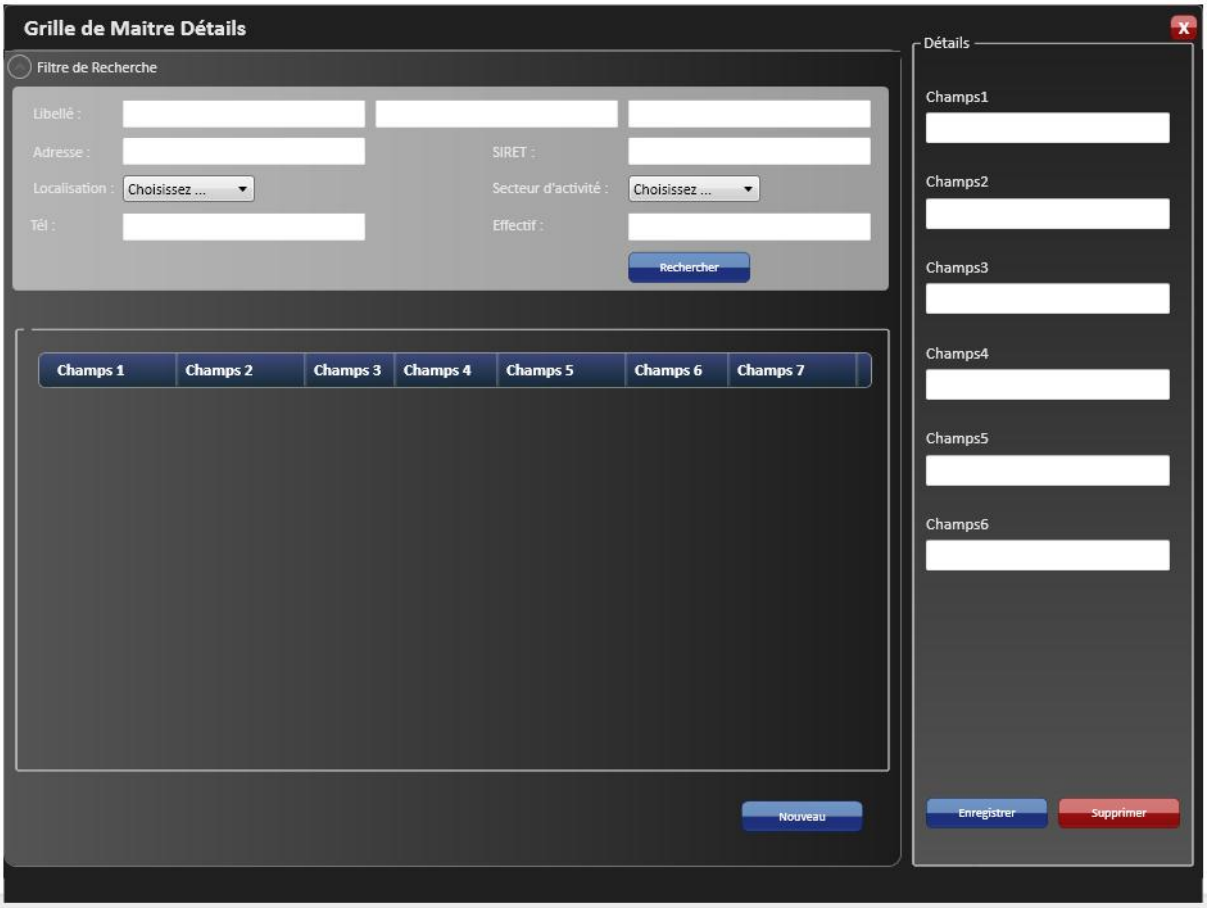
Non : On ignore les modifications.

15.2.16.2 Sur le bouton Enregistrer

N'afficher aucun message de confirmation

15.2.17 Ecrans de paramétrage

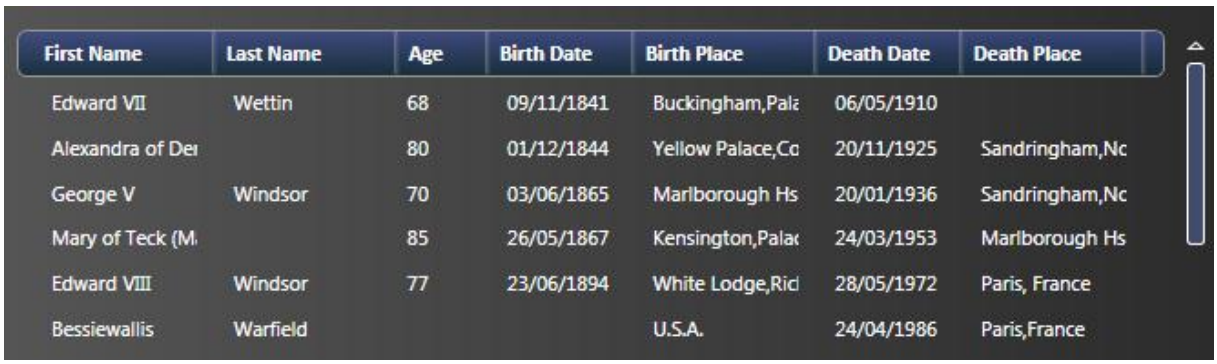
15.2.17.1 De style maître- détail « complexe »



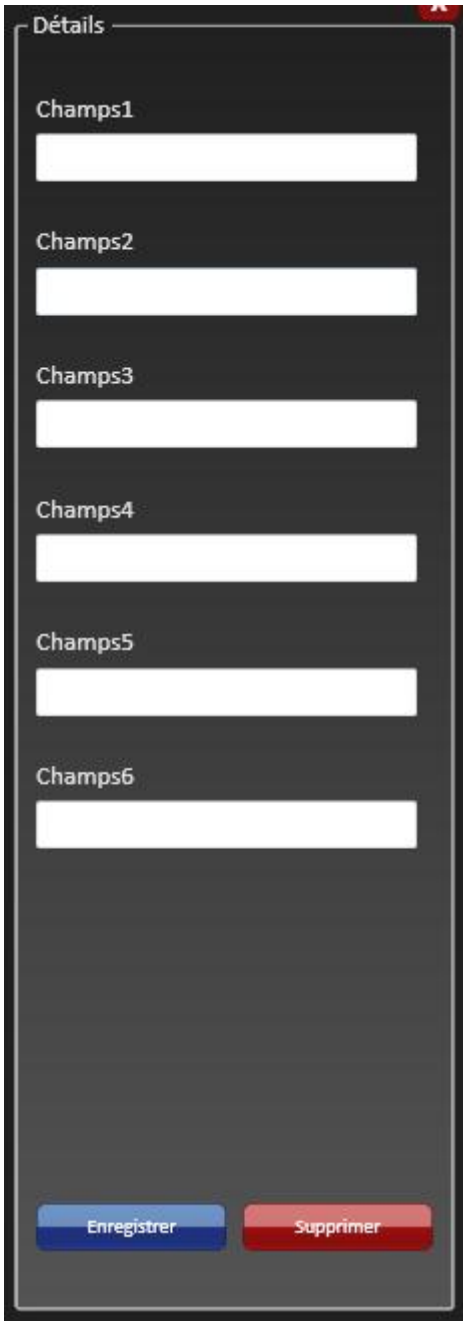
L'écran maître-détail « complexe » se compose d'une sous fenêtre secondaire de grille de résultat comme décrite au paragraphe 2.4 et d'une sous-fenêtre secondaire à droite. Cette deuxième sous-fenêtre permettra d'afficher et/ou de modifier l'élément sélectionné dans la sous-fenêtre de résultat. Elle devra respecter les normes de format des champs de saisie, à l'exception des libellés de champ qui seront positionnés au dessus du champ sans « : ».

La zone grille de résultat est limité par une bordure fixe. Si la grille dépasse à droite ou en bas, une « scrollbar » de couleur bleu foncé apparaît.

Sous-fenêtre de résultat.



Sous-fenêtre de détail



Ajout d'une nouvelle ligne

Il faut utiliser pour cela le bouton « Nouveau ».

Lorsqu'on fait « nouveau », le bouton « enregistrer » ne doit pas être actif tant qu'on n'a rien saisi.

En création : positionner le curseur sur le premier champ de saisie.

Quand on change de paramètres : appliquer les mêmes règles que lors de la fermeture d'une fenêtre.

15.2.17.2 De style « simple »

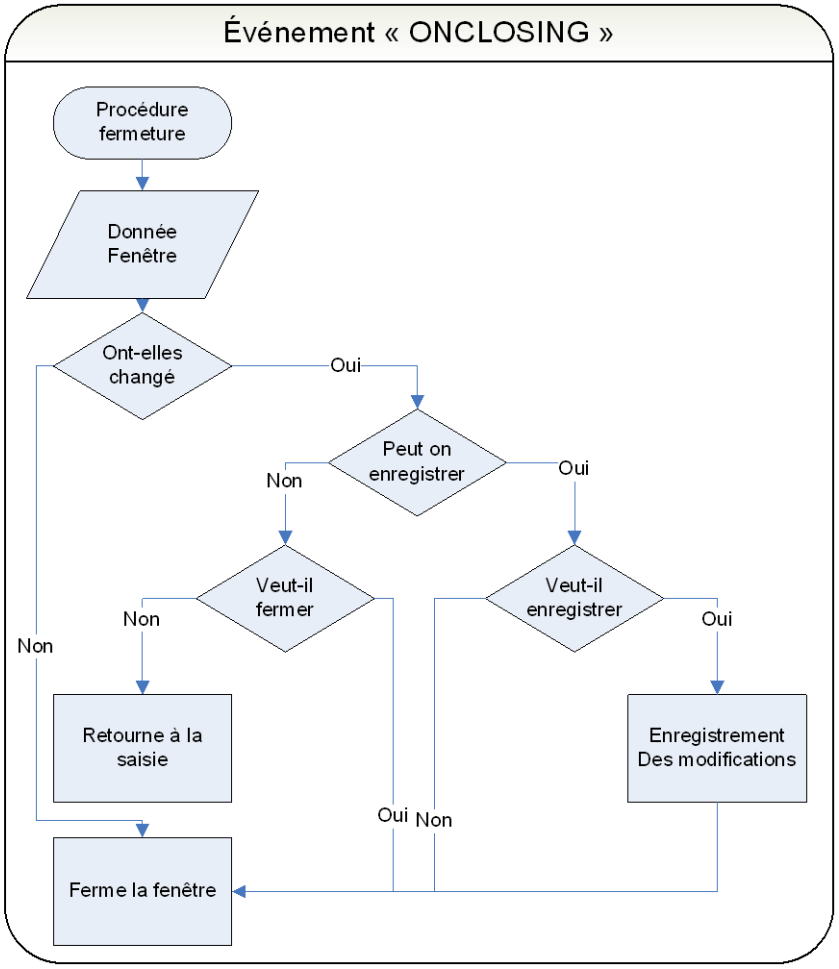
Les paramètres représentés par un code et libellé pourront être regroupés dans un même écran « Paramètres généraux ».

Les boutons sur cet écran sont de droite à gauche, Enregistrer, Supprimer.

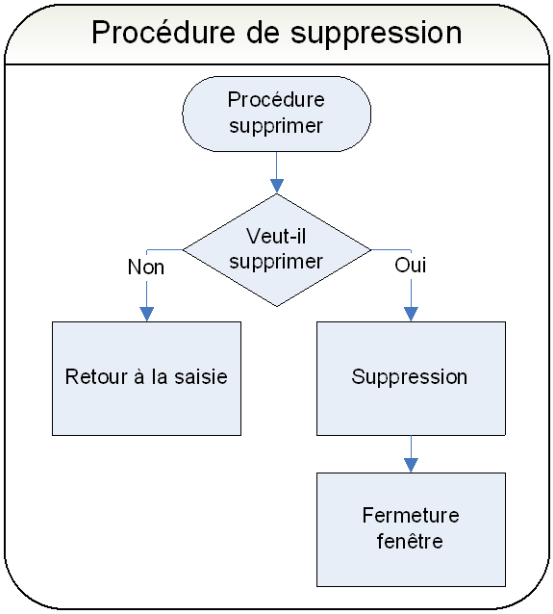

Paramètres Généraux
Paramètre Général :
Choisissez ...
Champs 1
Champs 2
Champs 3
Champs 4
Champs 5
Champs 6
Champs 7
Enregistrer
Supprimer

15.2.18 Procédures

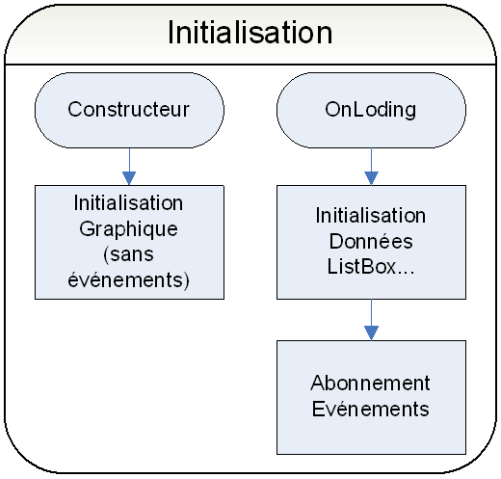
15.2.18.1 Procédure fermeture d'une fenêtre



15.2.18.2 Procédure de suppression

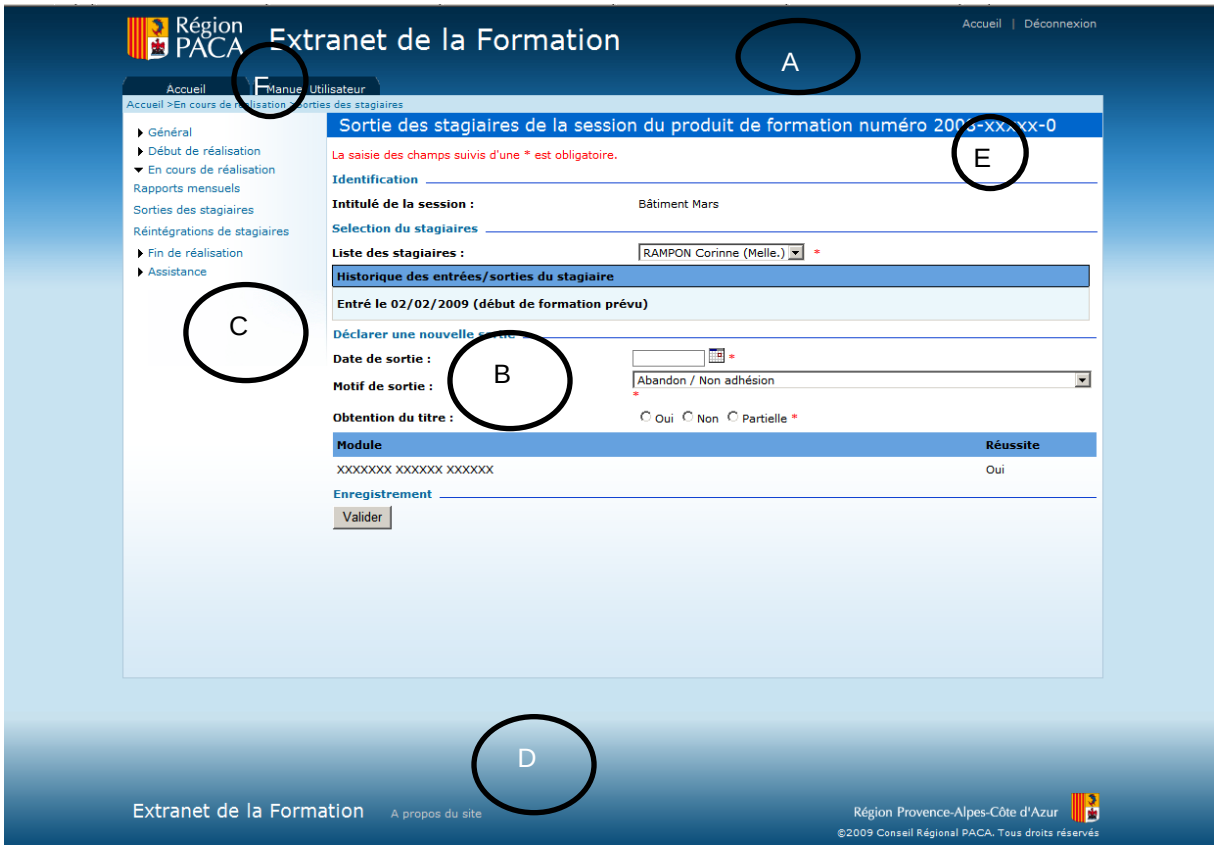


15.2.18.3 Procédure d'initialisation



15.3 Applications clients legers

15.3.1 Design général d'une page



La page peut-être découpée en 5 zones :

- En-tête de page - A
- Menu horizontal – F
- En-tête du module - E
- Menu - C
- Page Principale - B
- Pied de page – D

Les parties A, E, C, D et F sont apportés par la MASTER PAGE car ce sont des éléments fixes et communs à plusieurs pages. Il est préférable de ne pas dupliquer le code dans chaque page.

15.3.1.1 Décomposition

Un entête contenant :

- Le logo de la région
- Le sigle de l'application
- Le nom détaillé de l'application
- Le lien vers l'accueil
- Un lien de déconnexion
- Un menu horizontal, il doit contenir un lien vers le manuel utilisateur

Un fil d'Ariane qui permet de connaître le chemin de la page actuelle

Le titre de la page

Un menu positionné à gauche

A droite du menu, la partie métier de la page

En bas :

Un lien pour les contacts

Un lien « A propos du site »

Le copyright de la région.

15.3.1.2 Styles

Utilisation de la liste des styles CRPACA obligatoires.

Les styles les plus utilisés :

Titre de la page : style « bandeauTitre »

Le fond bleu ciel : style « formulaireBase »

Les Labels de présentation des champs doivent être en gras

Le titre des GROUPBOX : style « titreGroupBox » avec l'ajout d'une ligne bleu (<IMG height="1" src="/Commun/Images/1plblue.gif" width="100%">).

Le dégrader du bas de page : style « gradient7 ».

15.3.2 Gestion des données

15.3.2.1 Affichage

Utilisation des mêmes règles de formatage que les clients riches. (Exemple, Nombre : alignement à droite, formatage N2 ou N0).

15.3.2.2 Enregistrement, suppression…

Affichage des mêmes messages que ceux des clients riches (§ 2.6)

Affichage du message informant l'utilisateur de la réussite de l'action : « L'action a été exécutée avec succès ».

15.3.3 Police – tailles des fenêtres – surbrillance – sélection d'un champ

La police utilisée est le Verdana de taille 8.

L'écran doit être capable d'afficher toutes les informations dans une résolution de 1024 x 768.

Le passage de la souris sur un item entraîne la surbrillance de celui-ci.

La surbrillance est effectuée par une image dégradée de bleu.

Afin d'indiquer à l'utilisateur la page sur laquelle il se trouve, celle-ci sera identifiée par une surbrillance « blanche ».

15.3.4 Fenêtre de connexion

L'accès au portail ne nécessite pas de connexion contrairement à l'accès aux différentes pages du site.

Les boutons pour accéder à l'application est « Valider ».

15.3.5 Zone « En-tête de la page » - A

Cette zone sera visible quelque soit la page appelée. Elle présente :

- En haut à gauche le logo de la Région. Un clic sur celui-ci dirigera l'utilisateur vers le site de la Région (www.regionpaca.fr).

Au centre, le titre du portail. Un clic sur celui-ci ci dirigera l'utilisateur vers le portail.

En haut à droite, l'accueil de l'extranet. Un clic sur celui-ci dirigera l'utilisateur vers le portail.

15.3.6 Zone « Menu horizontal » - F

Cette zone dépend est composée :

d'un menu permettant d'accéder à l'accueil de l'application et au manuel utilisateur. Le style utilisé pour afficher cette partie du menu est le style MenuHorizontal.

d'un Fil d'Ariane permettant à tout moment de revenir sur l'accueil du module et de mettre à disposition, quelque soit la navigation à l'intérieur du module, le chemin de toutes les pages accédées. Le style utilisé pour afficher le composant est FilAriane.

15.3.7 Zone « En-tête du module » - E

Cette zone dépend du module sur lequel se trouvera l'utilisateur. Elle est composée d'un bandeau indiquant le titre du module. Les caractéristiques sont :

Police : Verdana 14

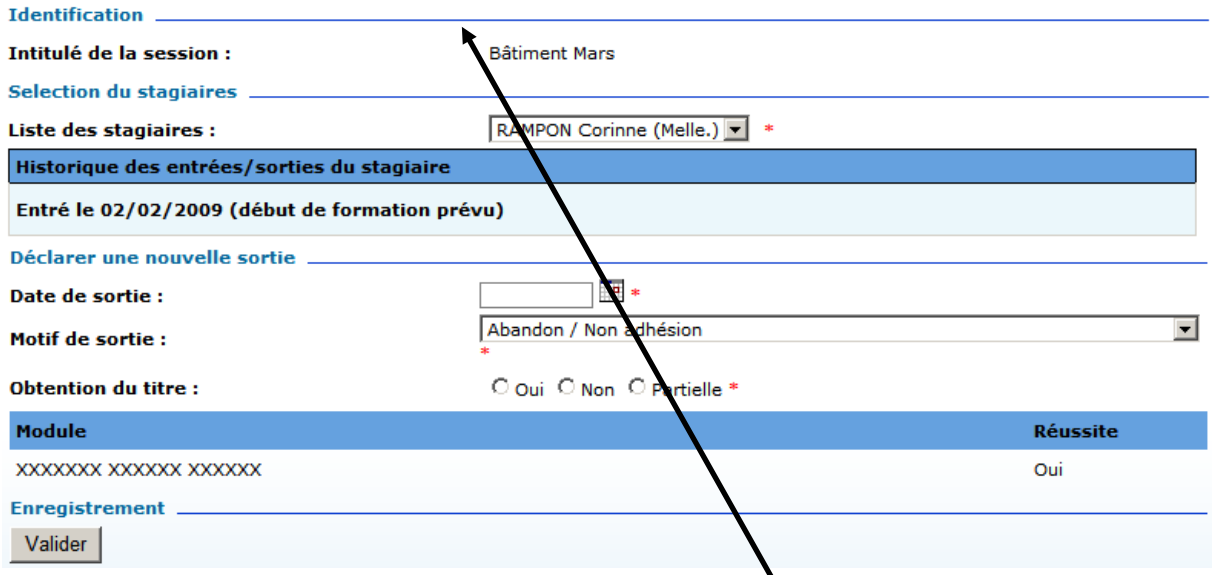
Couleur : Blanche

Couleur du bandeau : #0066cc

15.3.8 Zone « Menu » - C

Police : Verdana 8 (gras et non gras)

15.3.9 Zone « Page Principale » - B



Les champs associés à une même notion sont regroupés sous un même entête. (GroupBox).

Le fond de la page sera transparent (pour laisser apparaître le dégradé).

La police sera : Verdana 8.

Le police du titre du GroupBox est Verdana gras de 8, la couleur est la #093CD0, le trait est de la même couleur (#093CD0).

Chaque libellé sera suivi d'un blanc puis de « : ».

Les libellés sont alignés à gauche.

La première lettre est en majuscule.

Les champs non modifiables sont grisés et inaccessibles à l'utilisateur.

Les tableaux auront les caractéristiques suivantes :

Entête :

Police : Verdana 8

Couleur : noir

Les lignes du tableau

Police : Verdana 8

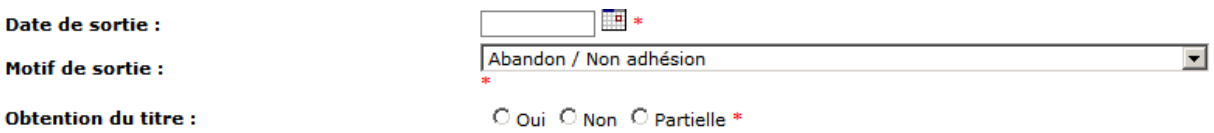
Couleur : noir

Les lignes seront de couleur :

Blanches.

bleu pale : #EBF7FB.

Les champs obligatoires seront suivi du « * »



Les mots ne sont pas coupés ou abrégés à part quelques abréviations bien définies. Par exemple :

Nom	Abrégé

Téléphone	Tél.
Numéro	N°

15.3.10 Zone « Pied de page » – D

Cette zone sera visible quelque soit la page appelée. Elle présente :

La couleur de fond de ce bandeau est un dégradé, le style du conteneur est nommé PiedDePage.

Le lien **contact** permet de générer un mail à une adresse spécifiée.

A propos du site permet l'ouverture d'un pop-up indiquant le numéro de version de l'application.

La version sera de la forme **chiffre.chiffre.chiffre*Lettre en minuscule.***

Exemple : 1.0.0.0a

Chaque modification d'une application en production entraînera l'évolution de son numéro de version.

Le premier chiffre représente la livraison d'une application dans sa globalité. Il changera uniquement dans le cas d'évolutions majeures ou de modifications impactant l'application dans sa globalité (cas d'une migration technique).

Le second représente un module ou un lot de l'application.

Le troisième représente une livraison intermédiaire de ce lot.

La lettre représente des corrections de bugs ou des petites évolutions (rajout d'une virgule, changement de libellé, déplacement d'un champ, etc. …).

Exemple 1 : Livraison du module Gestion dans la nouvelle application PRV2 développée en .NET.

Sa version initiale sera : 1.1.1a.

Si correction de bugs mineurs, la version passera en 1.1.1b, puis 1.1.1c, etc. …

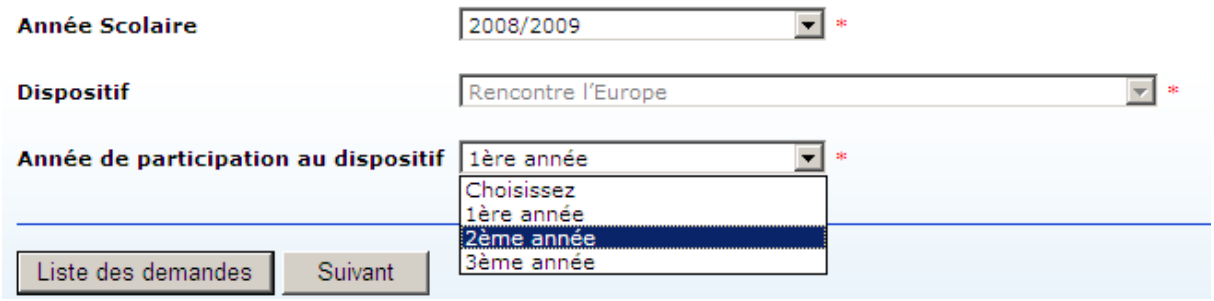
S'il y a beaucoup de Bugs à corriger ou une évolution, correction importante à effectuer sur le lot 1 alors on regroupera la livraison non plus sous une lettre mais dans une livraison intermédiaire d'un lot/module. L'application passerait alors en 1.1.2a.

Exemple 2 : Livraison d'un autre module dans la nouvelle application PRFV2. Ce module sera livré en 2 étapes.

La version de l'application à la première étape sera : 1.2.1a.

La version de l'application à la deuxième étape sera : 1.2.2a.

15.3.11 Liste de valeurs



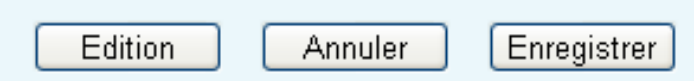
Les champs possédant des listes de valeurs sont gérés :

Par listBox si le nombre de valeurs n'est pas trop important.

Par popup dans le cas de listes de valeurs trop importantes (exemple : liste des communes). L'accès à cet écran de fera par l'icône placée à côté du champ. Il donnera l'accès à un écran de recherche.

Par Bouton radio dans le cas de valeurs non stockées en base et ne dépassant pas 3 choix possibles (exemple : choix du sexe).

15.3.12 Boutons



Tous les boutons seront de style « bouton standard » et de taille 80 pixels.

Ils seront positionnés en bas à droite de la page dans l'ordre suivant, de droite à gauche.

15.3.13 Champ date

L'utilisateur pourra :

Saisir directement la date

Choisir la date via le petit calendrier à droite du champ.

Le format de saisie des dates de sera JJ/MM/AAAA.

15.3.14 Navigation

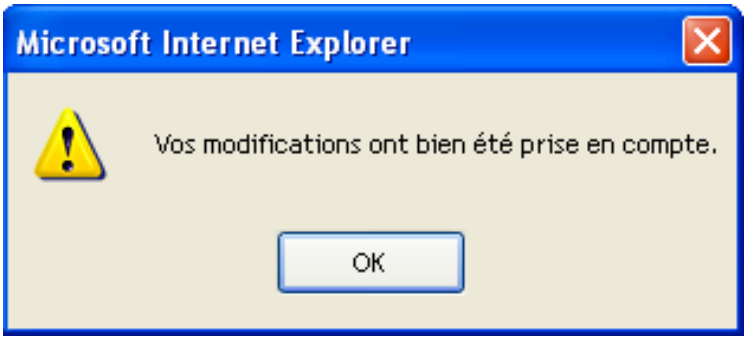
Tous les champs de l'écran ainsi que les boutons peuvent être accessibles via le clavier. La navigation de champs en champs se fera tabulation (TAB) pour avancer et par MAJ+TAB pour reculer. L'ordonnancement des champs par la touche tabulation suit un ordre logique et s'organise par GroupBox.

15.3.15 Règles transactionnelles

Sur le bouton Enregistrer

N'afficher aucun message de confirmation.

Afficher un message indiquant que les modifications ont été effectuées.



15.3.16 Template pour les pages web

En plus des modèles standards installés, disponibles dans les boîtes de dialogue « Nouveau projet » et « Ajouter un nouvel élément », vous pouvez accéder aux modèles que l'équipe de la TMA a pût être amené à développer. En effet, de façon générale, un Template personnalisé peut répondre à certaines problématiques telles que :

- utiliser une MasterPage particulière
- hériter d'une page de base
- inclure un cartouche de commentaires XML (nom utilisateur, date)
- faire référence aux namespace des autres couches de votre application (data, métier, ...)
- présenter un début de charte graphique dans leur <asp:Content> (un cadre et un titre)
- implémenter certaines méthodes d'initialisation
- …

15.3.16.1 Installation du Template

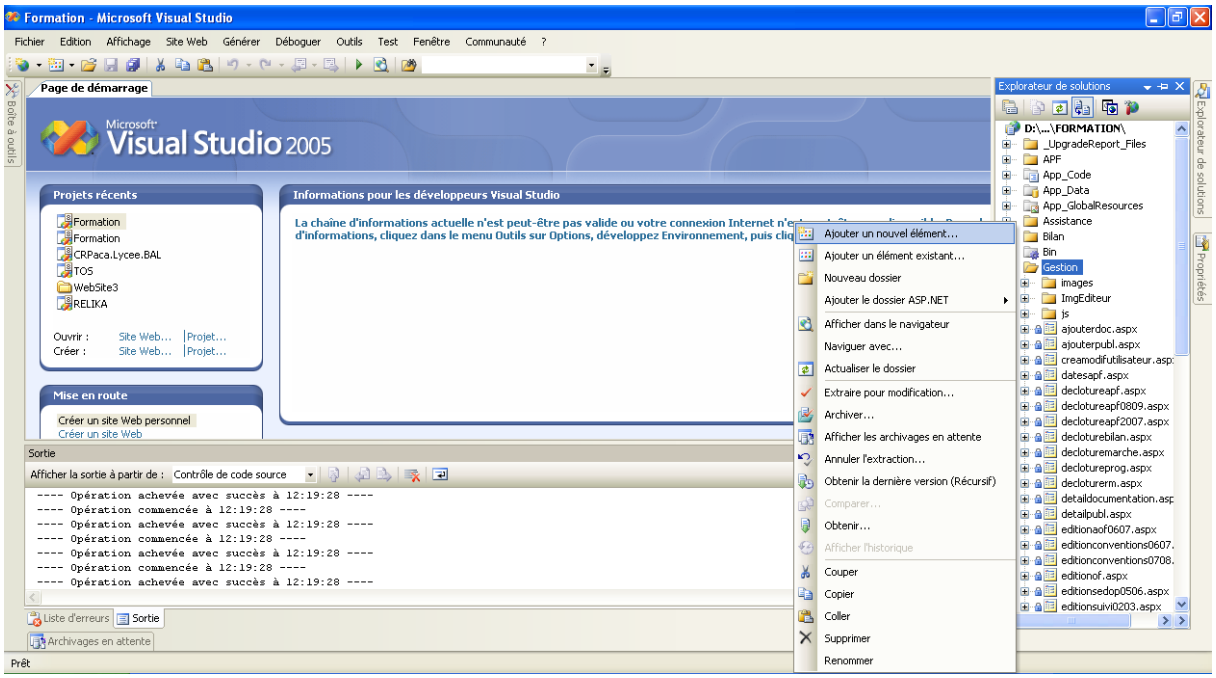
Pour avoir accès au Template « CRPaca Web Form », il suffit de copier le fichier « O:\DSI\TMA\COMMUN\SOPRA\Capitalisation C#\CRPacaWebForm.zip » sur chaque poste de développement dans le répertoire de destination suivant : « C:\Documents and Settings\<CompteWindows>\Mes documents\Visual Studio 2005\Templates\ItemTemplates\Visual C# »

15.3.16.2 Création d'une nouvelle page web

Un « assistant » a donc été mis en place afin de faciliter la création de nouvelles pages web sous Microsoft Visual Studio .NET 2005.

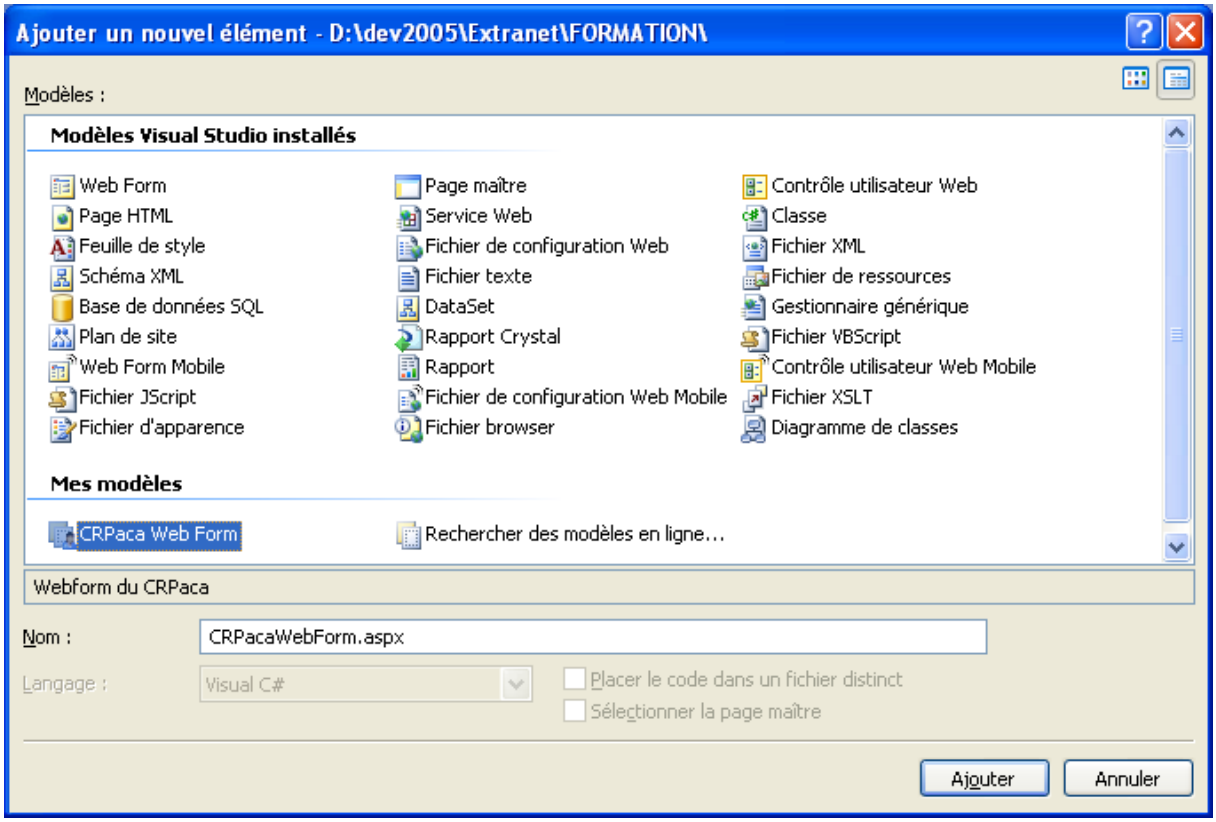
Dans l'explorateur de solutions, faire un click droit à l'endroit où doit être ajouté la page.

Sélectionner le menu « Ajouter », puis « Ajouter un nouvel élément »



Sélectionner « CRPaca Web Form »

Saisir le nom du fichier aspx puis cliquer sur « Ouvrir »



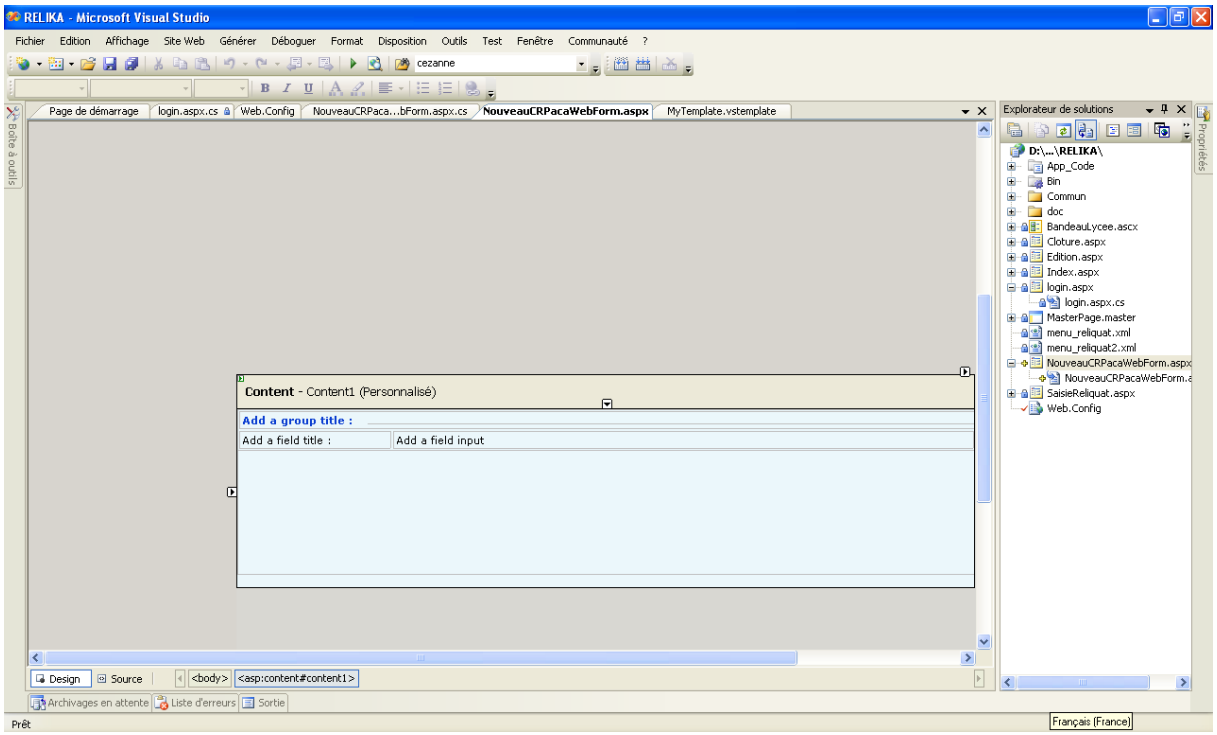
L'éditeur de Visual Studio affiche alors une page web type préconstruite avec :

- la Master Page en fond de page (comprenant : le bandeau du haut, le pied de page et le menu). Cette Master Page n'est pas modifiable à partir de cet emplacement.

- la zone réservée au formulaire de saisie, qui correspond au design de la page et qui peut bien évidemment être modifiée.

La page ainsi créée possède déjà le lien vers la Master Page et le « Look and Feel » du CR Paca.

Le dégradé de bleu présent sur le bandeau du haut et du bas ne s'affiche pas ici mais est présent à l'exécution de la page dans Internet Explorer



15.3.16.3 Entête et pied de page

Le bandeau principal et le pied de page du formulaire web ainsi crée est « rempli » par le biais d'un composant (web control). Ce composant récupère un « bloc » de code html à une adresse donnée, et l'insère dans une page web. Ainsi, toutes les pages crées par cet assistant feront appel aux mêmes blocs html pour ces 2 parties de la page. Ces sources html pouvant être gérés par la DSI, on dispose d'un mécanisme de mise à jour automatique pour ces 2 zones sur l'ensemble du site.

15.3.16.4 Menus

Le développeur peut personnaliser le contenu (et la mise en forme) du menu de gauche et du menu du haut et du fil d'Ariane grâce à un fichier siteMap de configuration. Le menu fait appel à ce fichier grâce à la fenêtre « propriétés » de Visual Studio : on fixe la DataSourceID :

(Expressions)	
DataBindings	(Collection)
DataSourceID	**SiteMapSource**

Comportement	
EnableViewState	True
ShowStartingNode	**False**
SiteMapProvider	**DefaultProvider**
StartFromCurrentN	False
StartingNodeOffsel	0
StartingNodeUrl	
Divers	
(ID)	**SiteMapSource**

Dans le Web.config global on pourra déclarer pour chaque module un siteMapProvider :

```
<siteMap defaultProvider="DefaultProvider">
    <providers>
        <add name="DefaultProvider" type="System.Web.XmlSiteMapProvider"
      siteMapFile="~/Web.sitemap" />
        <add name="SuiviAFPA2009Provider" type="System.Web.XmlSiteMapProvider"
             siteMapFile="~/Suivi/SuiviAFPA2009/Web.sitemap"/>
    </providers>
</siteMap>
```

Et suivant le module auquel on se connecte charger le bon provider (en conservant le même menu) dans la masterpage.

Le fait d'utiliser des composants .Net Natif va nous permettre d'uniformiser la gestion de notre composant sur l'ensemble du site.

15.3.16.5 Styles

La page modèle importe aussi quelques feuilles de styles en cascades (css) par défaut :

Une feuille de style propre aux menus.

Une feuille de style dit « de base » pour l'ensemble des contenus de l'extranet.

L'ensemble de ces feuilles de styles, permet de réaliser l'ergonomie qui a été établie. Ces feuilles définissent quelques balises html, mais surtout contiennent des classes de styles que le développeur de pages doit utiliser. D'ailleurs le Template en utilise déjà une bonne partie.

Bien entendu, si des styles supplémentaires sont nécessaires pour une page donnée, il est tout à fait possible d'inclure une nouvelle feuille de style.

16 DOCUMENT 7 – 1.00 – MISE EN PRODUCTION DES APPLICATIONS

16.1 Introduction

16.1.1 Objectifs du document

Le présent document a pour objectif de définir les règles de mise en production pour les développements d'applications en client riche et en client léger (pages web).

16.1.2 Domaine d'application

Tous les développements pour la Région PACA fait en C# ou ASP avec le FRAMEWORK .NET 1.1 avant 2008 et 2.0 après

16.2 Applications clients riches

16.2.1 Procédures

Il faut différencier les procédures :

- Première mise en production
- Mise en production d'une nouvelle version

16.2.1.1 Première mise en production

Pour la première mise en production

Le SAD ou l'équipe des prestataires

- Informe les différents acteurs d'une prochaine mise en production, au minimum une semaine avant la date réelle de la mise en production, en précisant la date de mise en production. Ce message est envoyé à
  - L'équipe système
  - Le SPTS
  - L'exploitation, en copie pour information

L'équipe systèmes et le SPTS

- Prévoit les ressources nécessaires à cette mise en production
- Attend la demande de mise en production

Le SAD ou l'équipe des prestataires

- Crée les répertoires de livraison sur le serveur de production
- Copie la version dans les répertoires de livraison
- Remplit le formulaire de demande de mise en production
- Envoie la demande de mise en production à l'équipe systèmes et ou SPTS, copie à l'exploitation.

Le SPTS

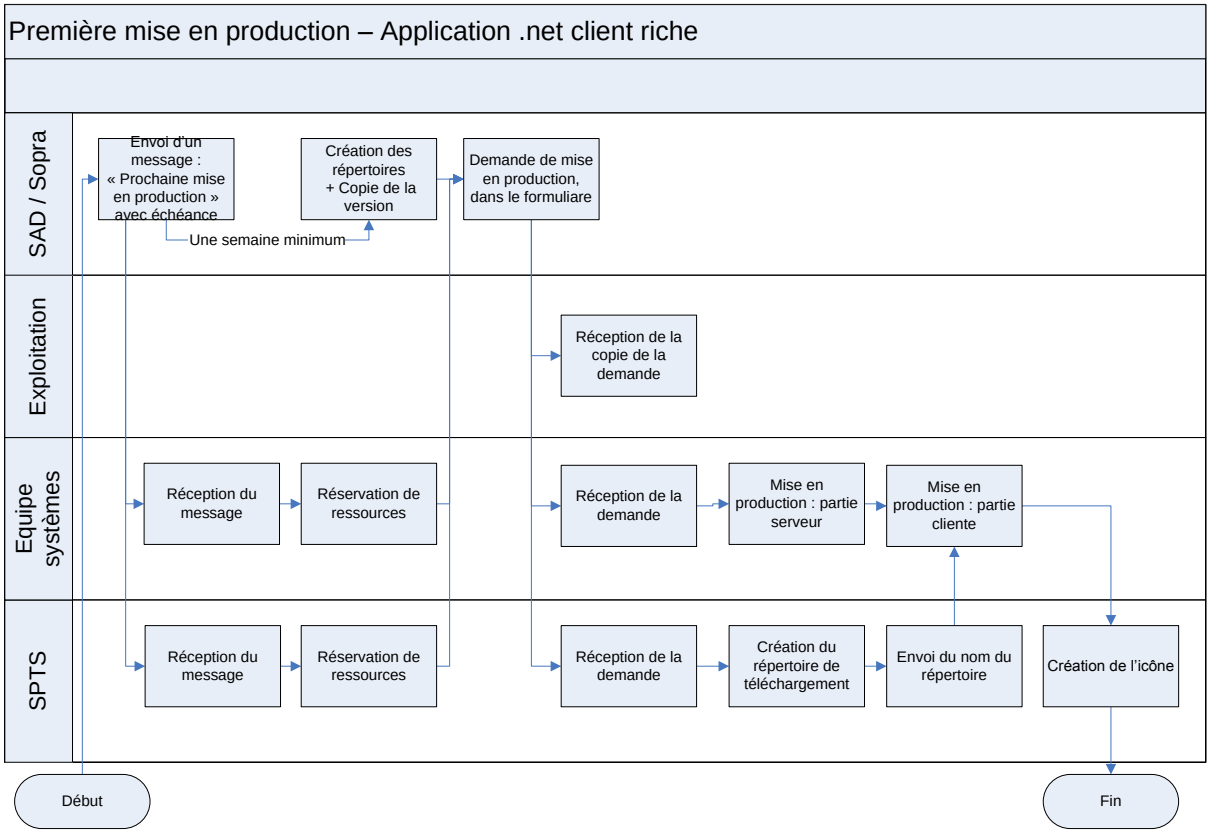
- Crée le répertoire de téléchargement
- Envoie le nom du répertoire de téléchargement à l'équipe systèmes

L'équipe systèmes

- Réceptionne la demande de mise en production à l'équipe systèmes, copie à l'exploitation.
- Crée les répertoires de production sur le serveur de production
- Crée le(s) service(s) demandé(s)
- Fait la mise en production de la partie serveur
- Fait la mise en production de la partie cliente
- Informe le SPTS que la mise en production est faite

Le SPTS

- Crée l'icône de l'application
- Déploie l'application aux différents utilisateurs (s'il a la liste)



16.2.1.2 Mise en production d'une nouvelle version

Pour la mise en production d'une nouvelle version

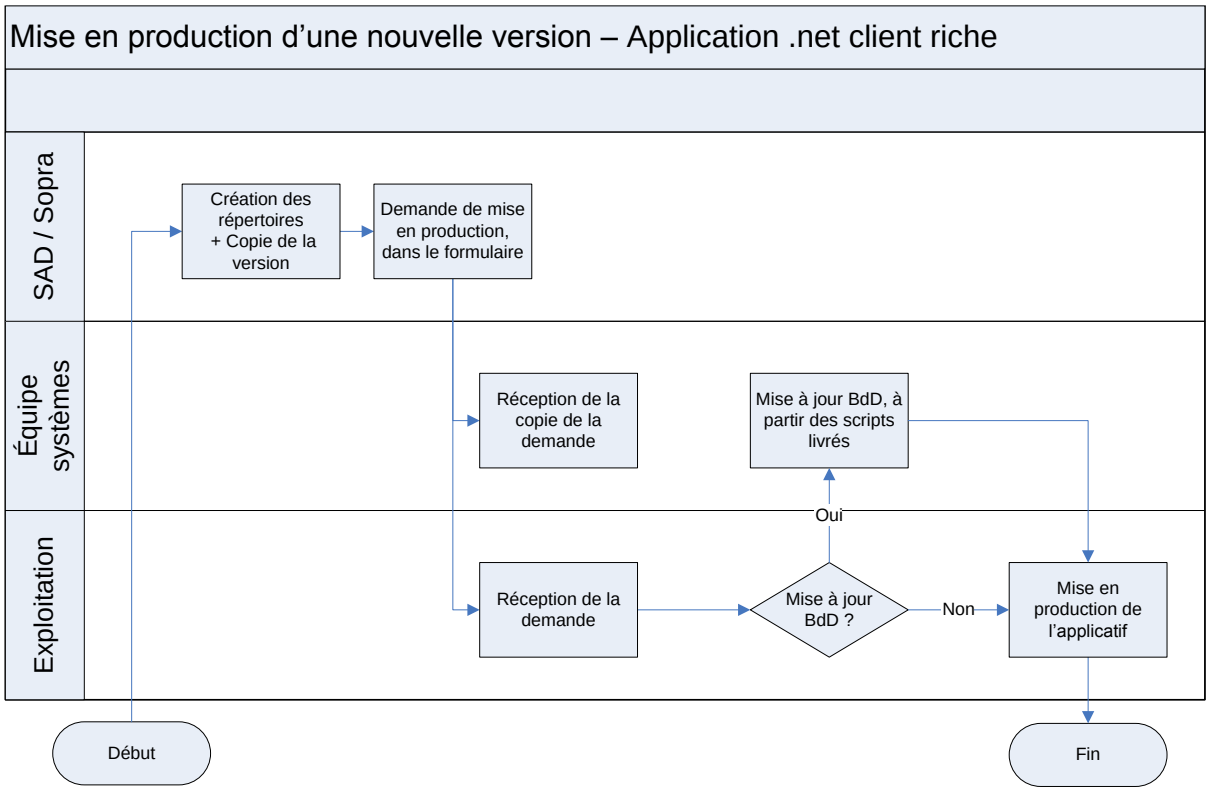
Le SAD ou l'équipe des prestataires

- Crée les répertoires de livraison sur le serveur de production, s'ils n'existent pas
- Copie la version dans les répertoires adéquats
- Remplit le formulaire de demande de mise en production
- Envoie la demande de mise en production à l'exploitation, copie à l'équipe systèmes.

L'exploitation

- Réceptionne la demande de mise en production à l'équipe systèmes, copie à l'exploitation.

- Si la mise en production nécessite une mise à jour de la base de données, envoi de la demande à l'équipe systèmes, qui assurera la modification de la base de données.
- Fait la mise en production



16.2.1.3 Formulaire de mise en production

Un formulaire standard de mise en production doit être utilisé. C'est ce fichier Word qui sera envoyé à l'équipe systèmes ou à l'exploitation.

16.2.2 Répertoires à créer

Chaque acteur a en charge la création des répertoires sur le serveur de production.

16.2.2.1 Livraison :

Le SAD et l'équipe des prestataires ont en charge la création des répertoires utilisés pour le dépôt des versions à mettre en production.

\\SERVEUR\Mise en production\COMMUN\DOMAINE\host	HOST commun à mettre à jour pour l'appli APPLI
\\SERVEUR\Mise en production\SPECIF\DOMAINE\APPLI\host	HOST specif à mettre à jour pour l'appli APPLI
\\SERVEUR\Mise en production\SPECIF\DOMAINE\APPLI\client	CLIENT à mettre à jour pour l'appli APPLI
\\SERVEUR\Mise en production \progs\Commun\SocleTechniquePacav2\Host	Socle technique

16.2.2.2 Chemins de production

L'équipe systèmes a en charge la création des répertoires utilisés pour la mise en production des applications

D:\applis\progs\commun\DOMAINE\host	HOST commun pour le domaine DOMAINE
D:\applis\progs\specif\DOMAINE\APPLI\host	HOST spécifique à l'application APPLI
D:\applis\progs\specif\DOMAINE\APPLI\client	CLIENT de l'application APPLI
D:\applis\progs\commun\SocleTechniquePacav2\Host	HOST du socle technique

16.2.2.3 Pour mémo - Nom des services

CRPACA.DOMAINE	HOST commun pour le domaine DOMAINE
CRPACA.APPLI	HOST spécifique à l'appli APPLI

16.2.3 Exemples de formulaires et Répertoires

16.2.3.1 BOURSSAN

16.2.3.1.1 Formulaire

Date de demande	01/03/06
Domaine	SanitaireSocial
Application	BOURSSAN
Version	1.02
Mise à jour HOST Commun	NON
Nom du service commun	
Mise à jour HOST Spécif.	OUI
Nom du service specif	CRPaca.SanitaireSocial.BALService
Mise à jour client	OUI
Mise à jour de la base	NON
Commentaires	

16.2.3.1.2 Livraison

\\CEZANNE\Mise en Production\progs\Specif\SanitaireSocial\BOURSSAN\Host
\\CEZANNE\Mise en Production\progs\Specif\SanitaireSocial\BOURSSAN\Client

16.2.3.1.3 Production

D:\applis\progs\specif\SANITAIRESOCIAL\BOURSSAN\host
D:\applis\progs\specif\SANITAIRESOCIAL\BOURSSAN\client
D:\applis\progs\commun\SocleTechniquePacav2\Host

16.2.3.2 PRF

16.2.3.2.1 Formulaire

Date de demande	01/03/06
Domaine	Formation
Application	PRF
Version	2.01
Mise à jour HOST Commun	OUI
Nom du service commun	CRPaca.formation.BALService
Mise à jour HOST Spécif.	NON
Nom du service specif	
Mise à jour client	OUI
Mise à jour de la base	NON
Commentaires	

16.2.3.2.2 Livraison

\\CEZANNE\Mise en Production\progs\Commun\Formation\HostV2
\\CEZANNE\Mise en Production\progs\Specif\Formation\PRF\client

16.2.3.2.3 Production

D:\applis\progs\Commun\Formation\HostV2
D:\applis\progs\Specif\Formation\PRF\client
D:\applis\progs\commun\SocleTechniquePacav2\Host

16.3 Applications clients legers (web)

16.3.1 Procédures

Il faut différencier les procédures :

- Première mise en production d'un nouveau domaine
- Mise en production d'une nouvelle version ou d'une nouvelle application dans un domaine déjà existant.

16.3.1.1 Première mise en production d'un nouveau domaine

Pour la première mise en production

Le SAD ou l'équipe des prestataires

- Informe les différents acteurs d'une prochaine mise en production, au minimum une semaine avant la date réelle de la mise en production, en précisant la date de mise en production. Ce message est envoyé à
  - L'équipe système
  - L'exploitation, en copie pour information

L'équipe systèmes

- Prévoit les ressources nécessaires à cette mise en production
- Attend la demande de mise en production

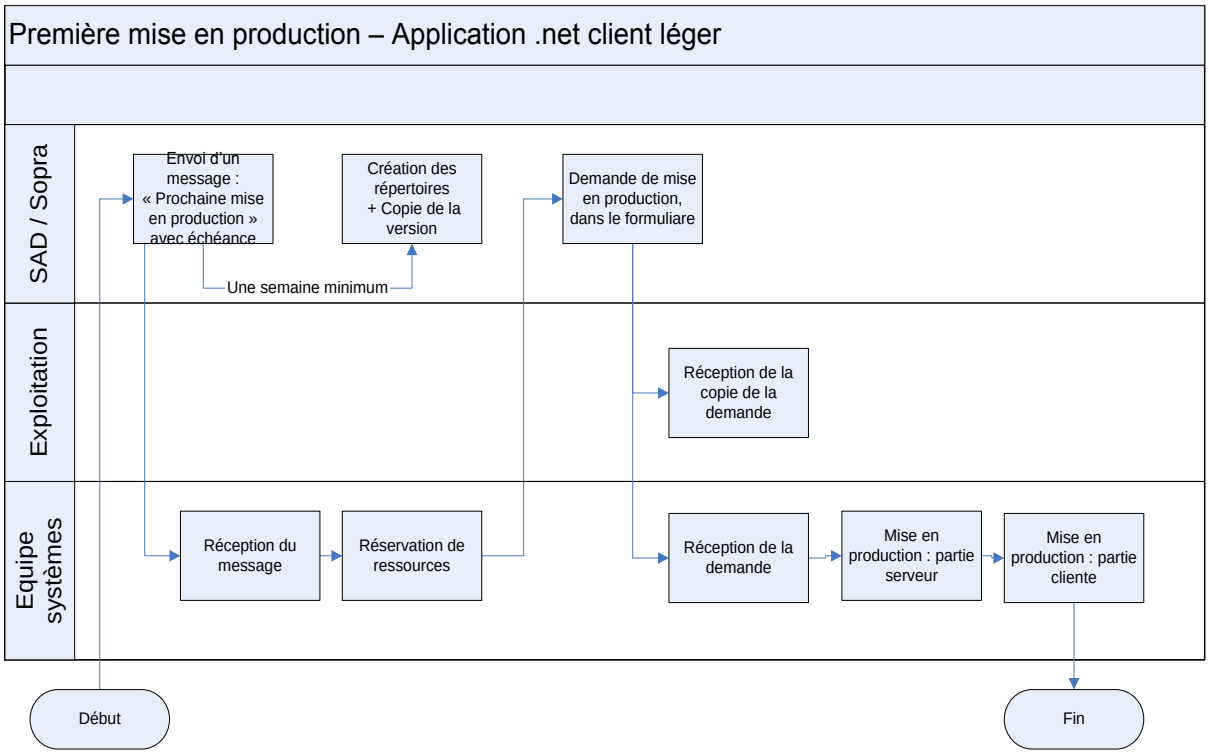
Le SAD ou l'équipe des prestataires

- Copie la version dans les répertoires de livraison
- Remplit le formulaire de demande de mise en production
- Envoie la demande de mise en production à l'équipe systèmes, copie à l'exploitation.

L'équipe systèmes

- Réceptionne la demande de mise en production à l'équipe systèmes, copie à l'exploitation.
- Crée le site Web correspondant au nouveau domaine.
- Crée les répertoires de production sur le serveur de production
- Crée le(s) service(s) demandé(s)
- Fait la mise en production de la partie serveur

- Fait la mise en production de la partie cliente
- Informe le SAD ou l'équipe des prestataires que la mise en production est faite

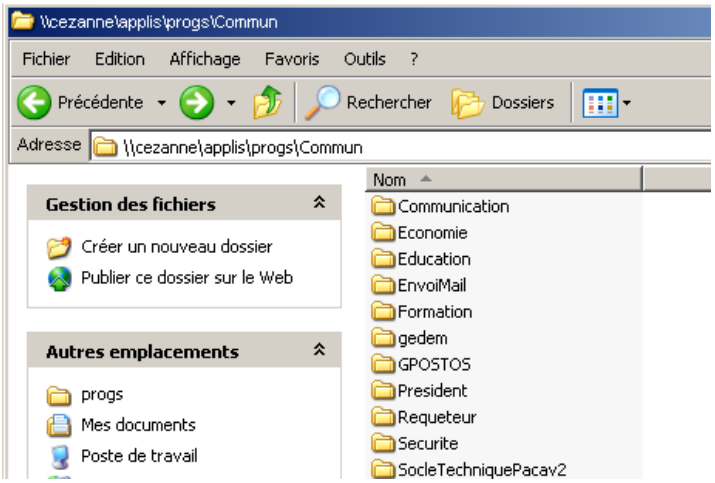


16.3.1.2 Mise en production d'une nouvelle version

Pour la mise en production d'une nouvelle version, la procédure se découpe ainsi :

- La partie Service (Host)
- La partie Client (Pages Web)

16.3.1.2.1 La partie service (Host)



Le SAD ou l'équipe des prestataires

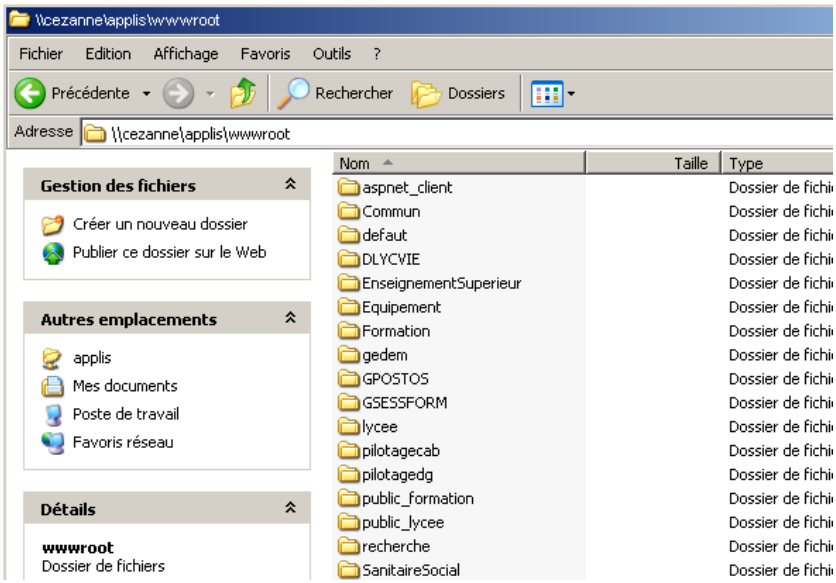
- Crée les répertoires de livraison sur le serveur de production, s'ils n'existent pas

- Copie la version dans les répertoires adéquats
- Remplit le formulaire de demande de mise en production
- Envoie la demande de mise en production à l'exploitation, copie à l'équipe systèmes.

L'exploitation

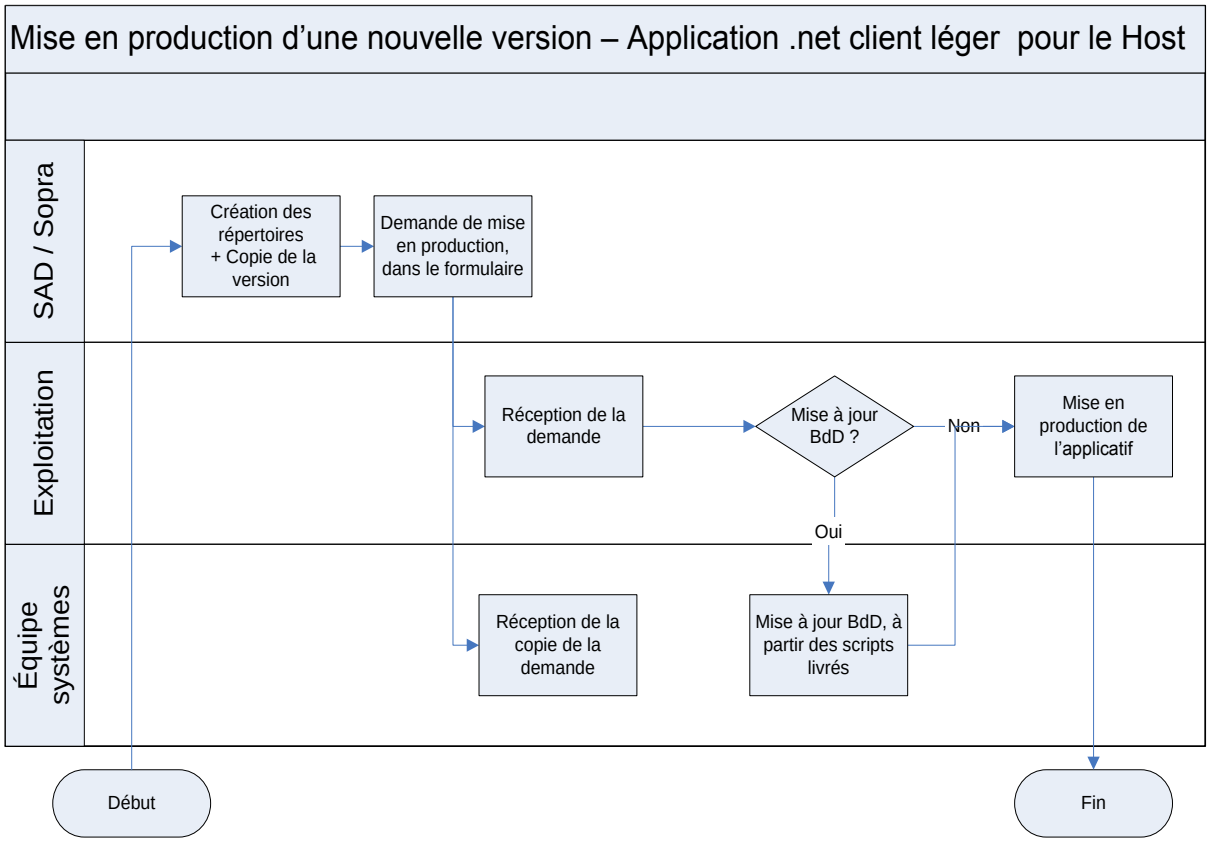
- Réceptionne la demande de mise en production à l'équipe systèmes, copie à l'exploitation.
- Si la mise en production nécessite une mise à jour de la Base de données, envoi de la demande à l'équipe systèmes, qui assurera la modification de la base de données.
- Fait la mise en production

16.3.1.2.2 La partie client (Pages Web)



Le SAD ou l'équipe des prestataires

- Crée les répertoires de production sur le serveur de production, s'ils n'existent pas
- Copie la version dans les répertoires adéquats



16.3.1.3 Formulaire de mise en production

Un formulaire standard de mise en production doit être utilisé. C'est ce fichier Word qui sera envoyé à l'équipe systèmes ou à l'exploitation.

16.3.2 Répertoires à créer

Chaque acteur a en charge la création des répertoires sur le serveur de production.

16.3.2.1 Livraison de la partie Host :

Le SAD et l'équipe des prestataires ont en charge la création des répertoires utilisés pour le dépôt des versions à mettre en production.

\\SERVEUR\Mise en production\progs\commun\DOMAINE\host	HOST commun à mettre à jour pour l'appli APPLI
\\SERVEUR\Mise en production\progs\commun\DOMAINE\APPLI\host	HOST Specif à mettre à jour pour l'appli APPLI
\\SERVEUR\Mise en production\wwwroot\APPLI	CLIENT à mettre à jour pour l'appli APPLI
\\SERVEUR\Mise en production\SocleTechniquePacav2\Host	Socle technique

16.3.2.2 Chemins de production pour la partie Client (Pages Web)

L'équipe systèmes a en charge la création des répertoires utilisés pour la mise en production des applications

\\SERVEUR\applis\wwwroot\APPLI \bin	Dll utilisées dans l'application
\\SERVEUR\applis\wwwroot\APPLI	CLIENT de l'application APPLI

16.3.3 Exemples de formulaires et Répertoires

16.3.3.1 BOURSSAN XNET

16.3.3.1.1 Formulaire

Date de demande	01/03/06
Domaine	SanitaireSocial
Application	BOURSSAN
Version	1.02
Mise à jour HOST Commun	OUI
Nom du service commun	CRPaca.SanitaireSocial.BALService
Mise à jour HOST Spécif.	NON
Nom du service specif	
Mise à jour de la base	NON
Commentaires	

16.3.3.1.2 Livraison

\\CEZANNE\Mise en production\progs\Specif\SANITAIRESOCIAL\BOURSAN\host
\\cezanne\Mise en Production\wwwroot\SanitaireSocial

16.3.3.1.3 Production

\\CEZANNE\applis\wwwroot\SanitaireSocial

17 DOCUMENT 8 – 1.00 – AUTHENTIFICATION AD

17.1 Introduction

Ce document décrit la mise en œuvre de l'authentification via Active Directory des applications .NET.

17.2 Rappel du contexte

La DSI a émis le souhait de mettre en place une authentification aux applications internes dédiées aux agents région, basée sur l'annuaire Active Directory (AD), ce afin d'éviter de gérer pour chaque application, le user de connexion ainsi que ses données propres

17.2.1 Principes de base

Le code permettant d'authentifier un utilisateur et de récupérer les infos d'un utilisateur dans l'AD est placé dans le projet CRPaca.Securisation de la solution CRPaca.Outil. Pour l'heure, ce code permet de :

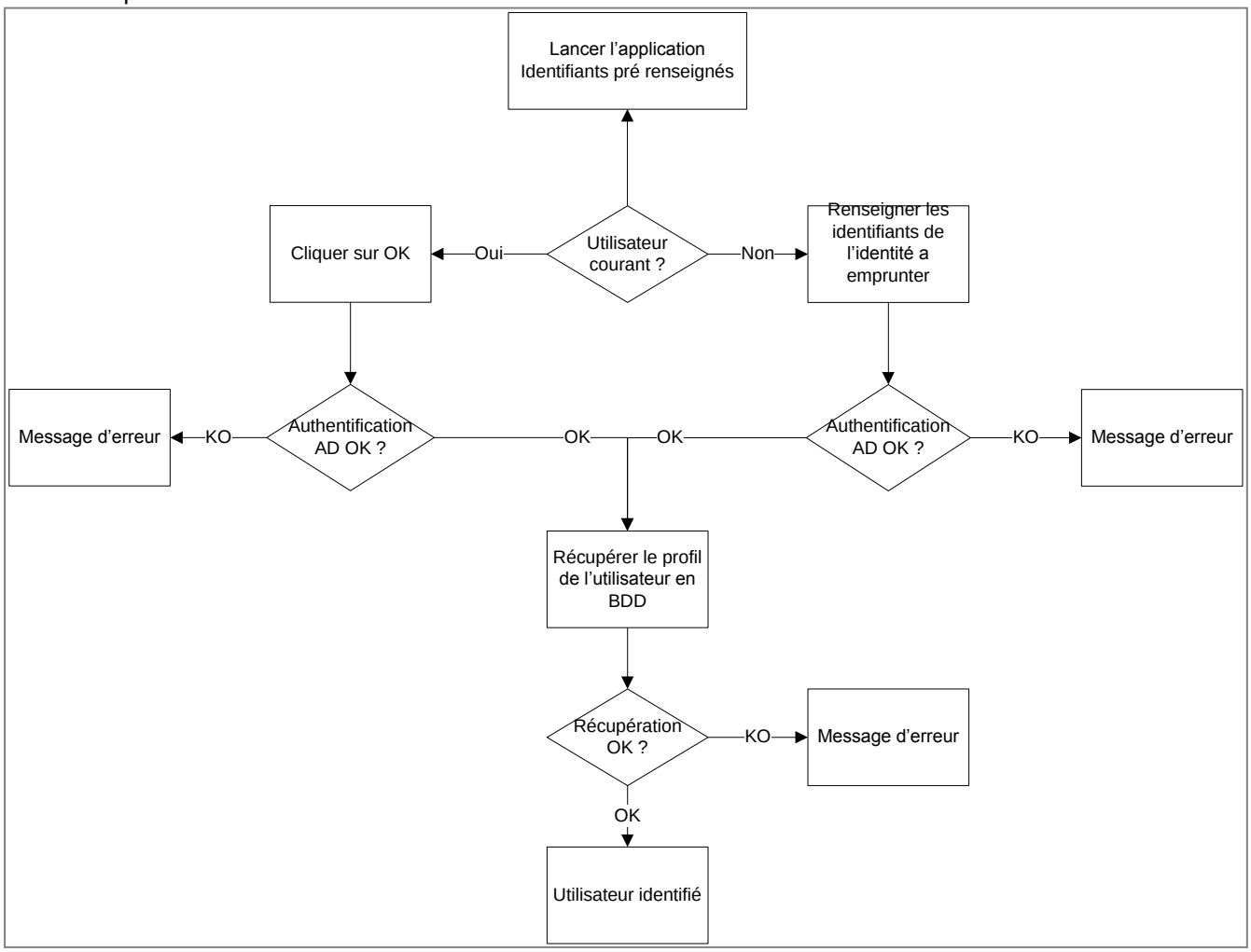
- Authentifier l'utilisateur dont la session Windows est active.
- Authentifier un utilisateur autre que celui dont la session Windows est active.
- Récupérer les informations d'un compte AD en fonction de son login. (Cette fonctionnalité permet de récupérer les informations renseignées dans l'AD et de les stocker dans les différentes tables UTILISATEUR des différentes applications.

Ce code est basé sur les namespaces *System.Security.Principal* et *System.DirectoryServices*. La table « utilisateur » de chaque application ne sert donc plus qu'a stocker des informations sur le profil de l'utilisateur.

Evolutions possibles :

A terme, ces profils peuvent être matérialisés par l'appartenance à des groupes AD. Il est possible techniquement de connaitre les groupes dont fait parti un compte AD. De ce fait des groupes AdminGPEA, ou UtilisateurSimpleGPEA peuvent être créés pour gérer les profils. De cette manière, la gestion des comptes est en partie externalisée. Une interface peut être développée afin de gérer l'ajout ou la suppression de compte dans ces groupes représentants les différents profils. Cependant, il faut s'assurer en amont que les limitations en terme de sécurité ne soit pas trop restrictives pour pouvoir développer ces fonctionnalités.

Principe d'authentification:



Attention !

Client lourd (WinForm) : Utiliser les méthodes qui ne sont pas suffixées de « WithImpersonation ».

Client léger (WebForm) : Utiliser les méthodes suffixées de « WithImpersonation » pour palier au problème de « double-hop »

17.2.2 Problème de « Double-Hop » (double bond)

Le scénario est simple : En général, on modifie une application Web pour authentifier l'utilisateur courant. A cet effet, on modifie le Web.config, en y insérant :

```
<authentication mode="Windows"/>
<identity impersonate="true"/>
```

Et on active l'authentification « intégrée windows ». A partir de là, on s'étonne de ne pouvoir accéder à aucune ressource distante du serveur Web… Le problème se résume ainsi :

IIS ne demande pas explicitement les identifiants à l'utilisateur. Ce sont les identifiants utilisés pour se connecter à sa session Windows qui sont utilisés. Le premier « bond » se produit ici : l'identification de l'utilisateur est passée, de manière sécurisée, du client au serveur Web (IIS). Le second « bond » intervient quand le serveur Web souhaite accéder à une ressource distante comme un serveur SQL ou toute ressource située sur un autre serveur. Le serveur Web se servirait alors de l'identification de l'utilisateur (sans le lui demander) pour accéder à n'importe quelle ressource, selon le désire du développeur. Ce n'est pas permit pour des raisons de sécurité évidentes. Le site suivant explique très bien ce phénomène : Concerning the credentials double hop issue. Il y en a d'autres qui traitent également très bien du sujet. On y trouve des solutions comme l'utilisation de l'authentification de base, plutôt que l'authentification intégrée. Mais cette authentification de base fait passer les identifiants en clair sur le réseau… ce qui est inenvisageable pour des questions de sécurité. La deuxième solution est d'activer la délégation de service. Cela a été fait, mais nous n'avons pas eu le résultat escompté, malgré une investigation poussée. Nous avons donc choisit d'utiliser une identité tierce pour accéder aux ressources distantes. Ainsi le problème de « double-hop » est évité.

17.3 Le projet CRPaca.Securisation

Ce projet permet de se servir des méthodes de connexion à l'AD et laisse la possibilité de faire évoluer et d'adapter ces méthodes en fonction des besoins qui pourraient survenir.

Pour pouvoir utiliser les méthodes de ce projet, il faut ajouter au projet utilisateur, la référence « CRPaca.Securisation » qui se trouve dans Win32\CRPaca.Outil\CRPaca.Securisation. Cette référence permet d'instancier un objet de la classe unique, et d'utiliser ses méthodes.

Principes techniques :

Dans le cas de l'authentification de l'utilisateur dont la session est actuellement active, on se sert de l'objet WindowsIdentity. Grâce à sa méthode « GetCurrent() », il est possible de récupérer l'utilisateur courant et récupérer ses informations dans l'AD grâce à une requête LDAP basé sur son « SID » (son identifiant unique et sécurisé). Ce cas du « SID » n'est vrai que pour le client lourd. Pour le client léger, il est nécessaire d'utiliser les méthodes suffixées de « WithImpersonation » qui pallient au problème de « double-hop ».

Dans le cas où on se connecte avec des identifiants différents de l'utilisateur connecté à la session Windows., il est possible de faire une recherche du compte grâce au Login et au Mot de passe saisis.

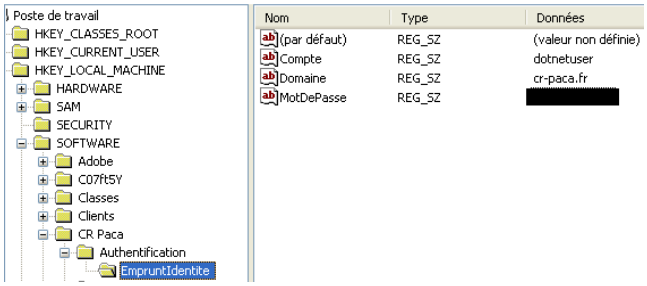
Rappel : Attention !

Client lourd (WinForm) : Utiliser les méthodes qui NE SONT PAS suffixées de « WithImpersonation ».

Client léger (WebForm) : Utiliser les méthodes suffixées de « WithImpersonation » pour palier au problème de « double-hop »

Les méthodes suffixées de « WithImpersonation » exécutent leur code grâce à une identité empruntée qui à l'heure où le document est écrit est « dotnetuser ». Ce compte est paramétrable : Les méthodes « d'impersonification » récupère les informations de l'identité à emprunter dans une clé de registre qui contient trois valeurs :

(HKEY_LOCAL_MACHINE\SOFTWARE\CR Paca\Authentification\EmpruntIdentite)
Nom du domaine
Nom du compte
Password



17.4 Exemple de modification du code du client

La méthode d'authentification d'origine du client a évolué au fil des évolutions et n'est pas forcément la même dans chaque application. Il convient donc d'adapter l'exemple de code ci-dessous à la situation rencontrée. Mais le principe reste dans l'absolu le même. Il suffit d'encapsuler l'authentification applicative par l'authentification AD.

```
//Instanciation de la table contenant les info de l'utilisateur
DataTable l_dttInfosUser = new DataTable("InfoUtilisateur");
//Si l'utilisateur en cours correspond au compte dont la session windows est active
if (this.tbUser.Text == Context.User.Identity.Name)
{
    //Création de l'objet contenant les methodes d'authentification
    SecurisationEchanges l_SecurisationEchanges = new SecurisationEchanges();
    //Authentification
    l_dttInfosUser = l_SecurisationEchanges.IsAuthenticatedCurrent();
    //si l'objet DataTable retourné ne contient aucune ligne, l'authentification AD a échoué
    if (l_dttInfosUser.Rows.Count > 0)
    {
        //On récupère le nom du compte, sans le nom de domaine
        int l_iposition DuDernierSeparateur = Context.User.Identity.Name.LastIndexOf(@"\");
        string l_strLoginSessionWindows = Context.User.Identity.Name.Substring(l_iposition DuDernierSeparateur + 1);

        //On appelle la méthode de récupération du profil (méthode d'origine du client)
        this.AuthentifierProfil(l_strLoginSessionWindows);
    }
    else //Dans le cas ou l'authentification est en échec, on affiche un message d'erreur.
    {
        this.lblInfo.Text = "Votre compte n'existe pas dans l'annuaire du CR PACA.";
    }
}
else//Cas de l'emprunt d'identifiants
{
    //On récupère le nom du compte, sans le nom de domaine
    int l_iposition DuDernierSeparateur = this.tbUser.Text.LastIndexOf(@"\");
    string l_strLoginSessionWindows = this.tbUser.Text.Substring(l_iposition DuDernierSeparateur + 1);
    //Création de l'objet contenant les methodes d'authentification
    CRPaca.Securisation.SecurisationEchanges l_SecurisationEchanges = new SecurisationEchanges();
    //Authentification
    l_dttInfosUser = l_SecurisationEchanges.IsAuthenticatedOther(l_strLoginSessionWindows,
                                                                 this.tbPassword.Text,
                                                                 ConfigurationSettings.AppSettings["NomDomaine"]);
    //si l'objet DataTable retourné ne contient aucune ligne, l'authentification AD a échoué
    if (l_dttInfosUser.Rows.Count > 0)
    {
        //On appelle la méthode de récupération du profil (méthode d'origine du client)
        this.AuthentifierProfil(l_strLoginSessionWindows);
    }
    else//Dans le cas ou l'authentification est en échec, on affiche un message d'erreur.
    {
        this.lblInfo.Text = "Votre compte n'existe pas dans l'annuaire du CR PACA.";
    }
}
```

17.5 Ressources

- Active Directory & .Net 2.0

http://webman.developpez.com/articles/dotnet/activedirectory/vbnet/?page=sommaire

- Procédure : Utiliser l'authentification par formulaires avec Active Directory

http://msdn.microsoft.com/fr-fr/library/aa302397.aspx

- Get Active Directory entry from SID

http://weblogs.asp.net/marianor/archive/2007/11/11/get-active-directory-entry-from-sid.aspx

- Understanding Kerberos Double Hop

http://blogs.technet.com/askds/archive/2008/06/13/understanding-kerberos-double-hop.aspx

- Concerning the credentials double hop issue

http://blogs.msdn.com/nunos/archive/2004/03/12/88468.aspx

- Building Secure ASP.NET Applications: Authentication, Authorization, and Secure Communication

http://msdn.microsoft.com/en-us/library/aa302415.aspx

- How To: Use Windows Authentication in ASP.NET 2.0

http://msdn.microsoft.com/fr-fr/library/ms998358(en-us).aspx

- How To: Use Impersonation and Delegation in ASP.NET 2.0

http://msdn.microsoft.com/fr-fr/library/ms998351(en-us).aspx

- Utiliser C# et Active Directory

http://morpheus.developpez.com/addotnet/ADCSharp/

18 DOCUMENT 9 – 1.00 – REGLES ET RECOMMANDATIONS TFS

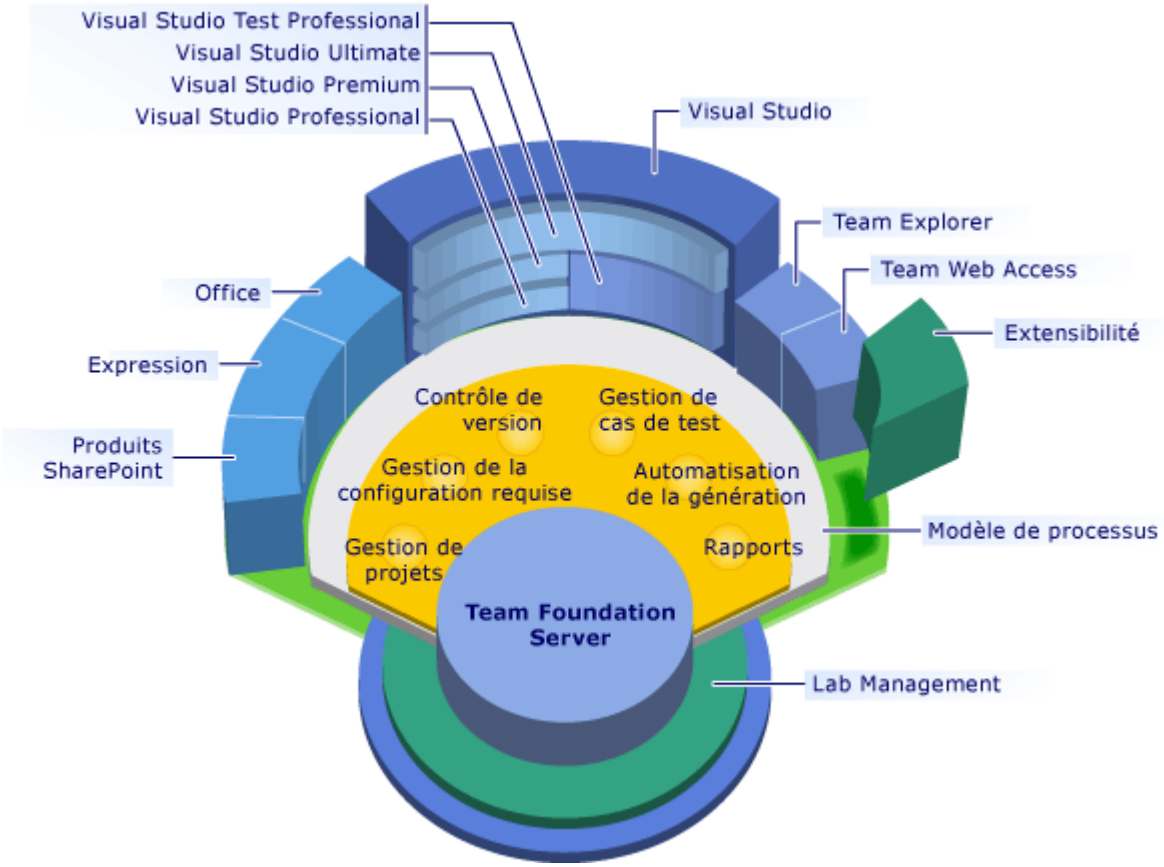
18.1 Introduction

Le présent document constitue les règles et recommandations de l'utilisation de Team Foundation Server 2010 de Microsoft. L'objectif est de proposer au développeur un document expliquant les grands principes de la gestion du contrôle de version. Cet outil remplace Visual Source Safe.

Ces règles et recommandations sont reprises du référentiel documentaire de Microsoft disponible sur le site http://msdn.microsoft.com/fr-fr/library/ms181368.aspx.

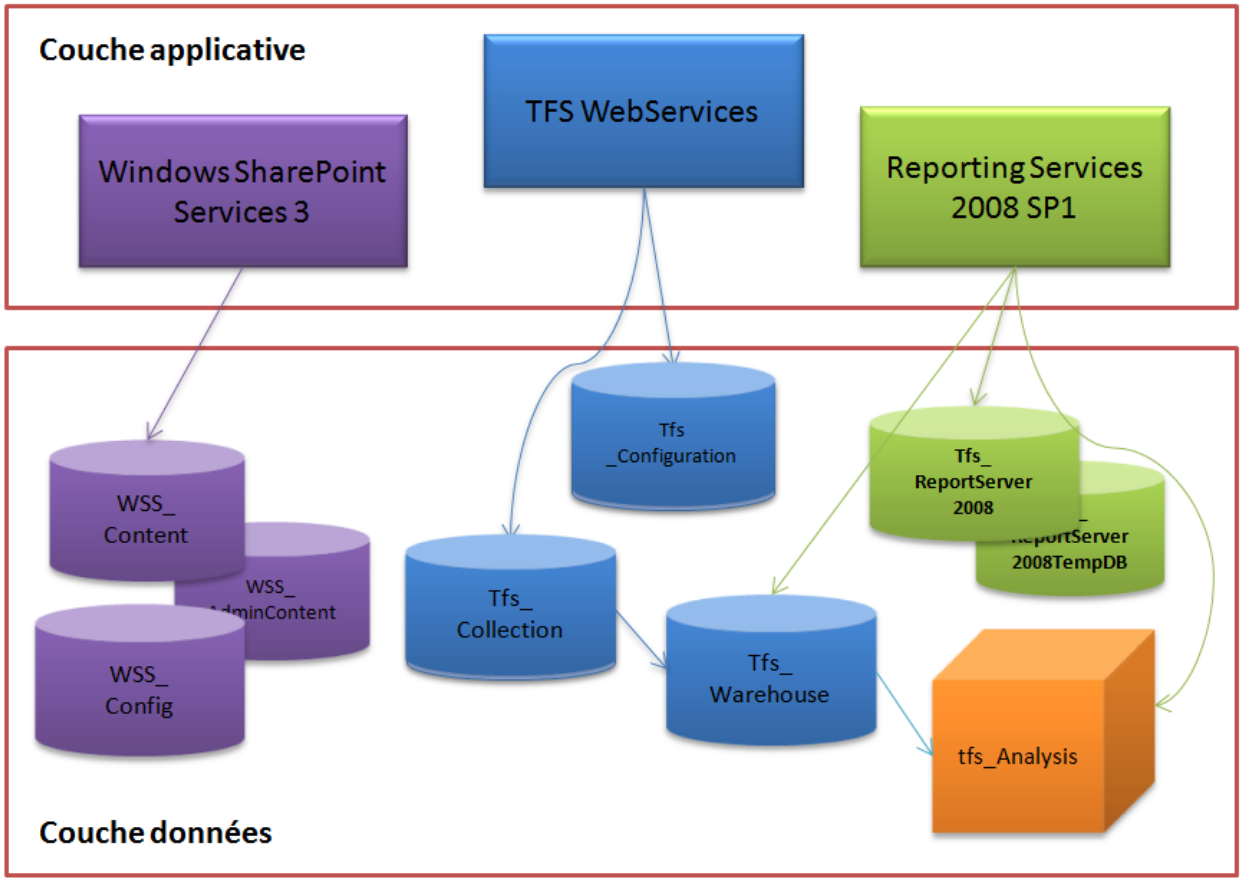
18.2 Description de l'outil

Microsoft Visual Studio Team Foundation Server, élément central des outils de gestion du cycle de vie des applications, assure le contrôle de version et fournit un système de génération, ainsi que des outils et métriques pour la gestion et l'organisation des projets. Avec une licence Team Explorer, vous pouvez afficher et mettre à jour des données dans Team Foundation Server à l'aide de Team Web Access, Visual Studio et d'autres environnements de développement intégrés. Vous pouvez accéder à quelques-unes de ces données sans licence Team Explorer en utilisant un navigateur Web pour consulter des portails du projet d'équipe.



Pour plus d'informations sur les clients Team Foundation Server, consultez Utilisation de clients Team Foundation.

L'architecture applicative de TFS se décrit de la façon suivante :



Dans le contexte du conseil régional PACA, Les couches applicative et donnée sont installées sur le même serveur :

- **Nom** : TEAMTFS
- **IP** : 10.130.2.92
- **OS** : Windows 2008 R2 Standard / 64 bits / FR

18.3 Configuration du poste client

18.3.1 Guide d'installation des postes clients

Le tableau ci-dessous récapitule la liste des outils dont l'installation est requise sur les postes de développement (surtout pour les Services Packs).

TYPE DE POSTE	NOM OUTIL	DESCRIPTIF OUTIL
NON DEVELOPPEURS	TEAM EXPLORER 2010	CLIENT DE CONNECTION AU SERVEUR TEAM FOUNDATION. CELUI-CI INSTALLE EGALEMENT LES API NECESSAIRES AU BON FONCTIONNEMENT DES AUTRES OUTILS.
DEVELOPPEURS	VISUAL STUDIO 2010	ENVIRONNEMENT DE DEVELOPPEMENT / INCLUS TEAM EXPLORER
NON VISUAL STUDIO 2005+	MSSCCI PROVIDER	AJOUT LA PASSERELLE PERMETTANT DE CONNECTER VS 2003 ET AUTRES OUTILS A TEAM EXPLORER (ET DONC A TEAM FOUNDATION SERVER).
TOUS	TEAM FOUNDATION SERVER POWER TOOLS	AJOUTE UN ENSEMBLE D'OUTILS ET DE FONCTIONNALITES (EN ANGLAIS) POUR ENRICHIR LES INTERACTIONS AVEC LE SERVEUR. IL EST NOTAMMENT PRATIQUE D'INSTALLER LES « EXTENSIONS SHELLS ».
POSTES ADMIN	TEAM FOUNDATION SERVER SIDEKICKS	PERMET D'ADMINISTRER LES ESPACES DE TRAVAIL, DEVERROUILLER LES FICHIERS VERROUILLES…
POSTES ADMIN	TFS ADMIN TOOL	CENTRALISE LA GESTION DES DROITS SUR LES PROJETS D'EQUIPE

18.3.2 Créer un nouveau projet d'équipe (Team Project)

Un projet d'équipe (Team Project) est créé à chaque nouvelle application. Cette entité propre à TFS devient le référentiel projet dans lequel on trouvera les documents, le code source ainsi que toute la gestion du projet. Seul un membre habilité sera responsable de la création d'un team project. Celui-ci aura les droits d'un Administrateur applicatif.

Pour créer un nouveau projet d'équipe :

- Démarrez Team Explorer avec un compte Administrateur applicatif
- Connectez-vous au serveur TFS puis démarrez le processus de création de projet d'équipe grâce au menu contextuel

La procédure de création d'un projet d'équipe est décrite à l'adresse URL suivante :
http://msdn.microsoft.com/fr-fr/library/ms181477.aspx.

18.3.3 Politique du nommage des projets d'équipe

La politique de nommage doit, concernant les projets d'équipe, suivre les règles ci dessous.

Attention, le nom d'un Team Project est définitif. La suppression physique des teams projects se fait directement sur le serveur en ligne de commande avec la commande tfsdeletproject.exe

On ne crée qu'un seul Team Project par domaine fonctionnel :

- Ex : DF_EnseignementSuperieur
- EX : DF_Lycee

(DF pour Domaine Fonctionnel)

On ne crée qu'un seul Team Project pour les Frameworks et composants tiers

- Ex : EXT_Sopragroup.Accelerate
- Ex : EXT_CRPACA.Framework

(EXT pour Externe)

18.3.4 Organisation des projets d'équipe

Suite à l'étude de migration du référentiel code source, une nouvelle organisation des projets est proposée. Le but étant :

- De réorganiser l'ensemble des projets actuels par domaine fonctionnel,
- De laisser toujours à la racine du Team Project un répertoire DEV (qui servira pour la future gestion des branches).

Aujourd'hui, dans Visual Source Safe l'arborescence exisante est la suivante:

- BAL
  - Project1
  - Project2
  - Project3
- Extranet
  - Project2
  - Project4
- Win32
  - Project 1
- Rapports
  - Project2
  - Project3

Les nouveaux projets devront respecter cette arborescence :

- DF_ProjetDEquipe1
  - DEV
    - BAL
    - Win32
- DF_ProjetDEquipe2
  - DEV
    - BAL
    - Extranet
    - Rapports

Remarque :

- Les composants multi-référencés (ex : Framework et composants tiers, Framework Sopra…) seront isolés dans des Projets d'équipes dédiés (et non pas au milieu du référentiel tel que c'est le cas aujourd'hui)
- Pour les utiliser, il sera nécessaire de créer des espaces de travail (WORKSPACES) multi-mappés
  - Ex :
    - WK_ProjectDequipe1
      - => $/DF_ProjectDequipe1/DEV
      - =>$/EXT_SopraGroupe.Accelerate/ComposantA

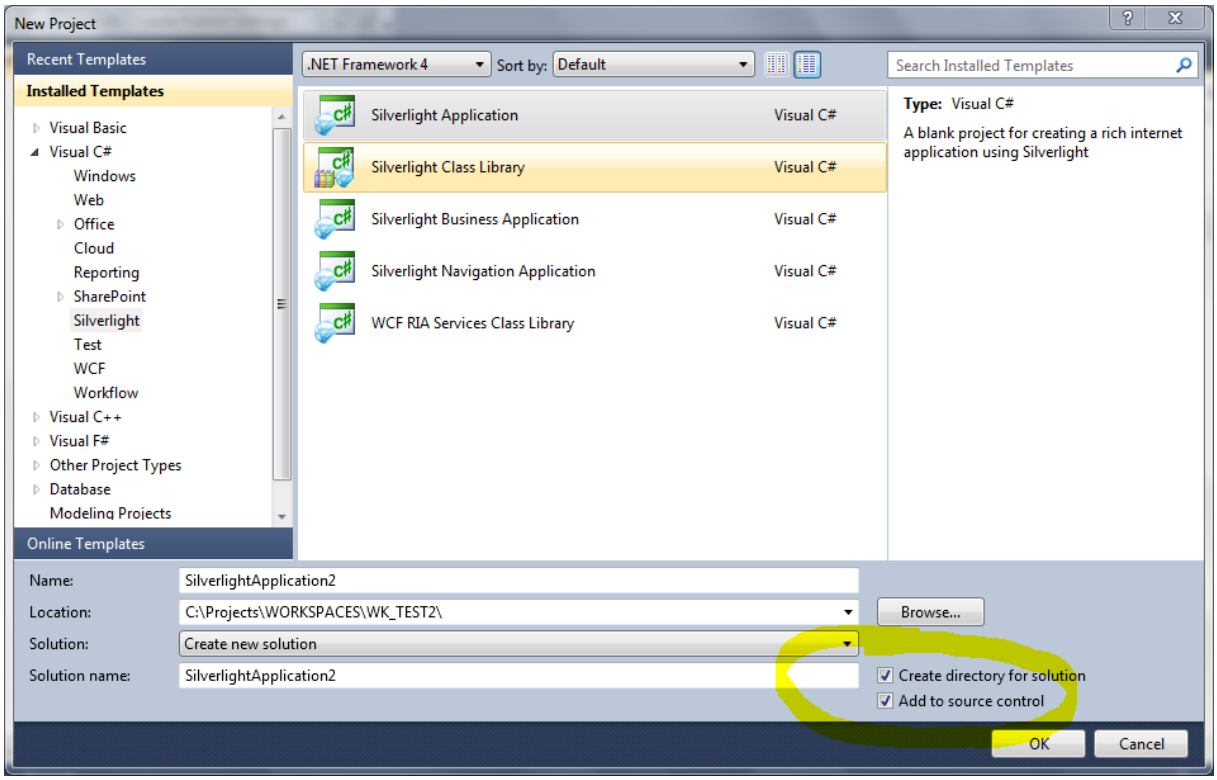
18.3.5 Création d'un espace de travail et Obtention des fichiers sur le poste client

La procédure de création d'un espace de travail et l'obtention des fichiers est décrite dans la documentation suivante : http://msdn.microsoft.com/fr-fr/library/ms181384.aspx

Les espaces de travail doivent respecter la règle de nommage suivante : Le nom des espaces de travail sont préfixés de « WS_ » ex : WS_XXXX.

18.3.6 Ajout un nouveau projet Visual Studio dans TFS

Lors de la création d'un projet Visual Studio, le développeur doit cocher la case « ajouter au control de code source » en bas à droite de l'écran de création



19 DOCUMENT 10 – 2.00 – GUIDE DE GESTION DE LA CONFIGURATION

19.1 Introduction

19.1.1 Objectif

La gestion de configuration consiste à gérer méthodiquement les éléments constitutifs du projet, de manière à garantir :

- la traçabilité des interventions
- la cohérence de l'information entre les états intermédiaires et l'état final durant tout le cycle de développement du projet

Les bénéfices de l'application de ces procédures de gestion de configuration sont les suivants :

- Prévenir la régression fonctionnelle
- Faciliter la réutilisation de composants
- Réduire les coûts de maintenance
- Améliorer la qualité générale du produit final

Ce document décrit en particulier les différentes étapes de la gestion de configuration des sources et des livrables, du développement à la livraison finale.

Il établit les méthodes et procédures pour :

- Organiser la Gestion de Configuration
- Identifier les différents référentiels de configuration dans le processus de développement, puis tout au long du cycle de vie du logiciel
- Contrôler les modifications des référentiels de configuration et gérer leurs différentes versions
- Administrer ces référentiels de configuration en fournissant des états de référence et permettre les audits de configuration

19.1.2 Scope

Les activités de GC se produisent pendant toutes les phases du cycle de développement.

Les domaines d'application actuellement en cours de gestion de configuration sont les suivantes:

- SQL/PLQSL
- .NET
- Reporting services
- Documentations

19.1.3 Définitions et abréviations

Gestion de la configuration	Activités nécessaires pour établir et maintenir l'intégrité des produits de travail internes et externes qui sont sujettes à modification lors du projet, et s'assurer que toute version d'un produit du travail qui a été placé sous GC peut être recréée à tout moment.
Eléments ou article de configuration	Ce sont les demandes de travail ou autres documents similaires, les exigences, spécifications, le code source, les procédures de test, la documentation de conception, les exécutables, les compilateurs et ainsi de suite. Certains produits de travail du projet doivent être "gérés et contrôlés" mais ils ne sont pas placées sous la gestion des configurations formelles, ils sont contrôlés par des versions, avec les dernières versions disponibles pour l'équipe de projet et autres parties intéressées.
Responsable de la configuration	Le chef de projet ou le responsable, désigné, de la gestion de configuration.

Révision	Une révision a pour rôle de stocker l'état intermédiaire d'un article de configuration. Elle correspond, par exemple, à un lot de corrections d'anomalies détectées dans une source logicielle
Version	une version définit l'état des fonctionnalités d'un logiciel. Un changement de version est lié à des évolutions fonctionnelles
Tronc ou Trunk	Le TRONC constitue la branche principale du projet. Elle contient, à certains moments, la version la plus avancée de l'ensemble des éléments de configuration. D'une manière générale, il est recommandé de ne jamais développer à partir du TRONC - ne pas avoir d'espace de travail qui résulte d'un check out du TRONC. Le TRONC ne doit contenir que des versions applicables, stables. La structure de configuration est donc complète dans le TRONC
Label	Un label permet de fixer une configuration. Un label représente une configuration figée de l'application. Il peut s'agir d'une configuration correspondant à une livraison, à un jalon …
Branche	Une branche est une copie totale ou partielle du TRONC. La création de branches permet de réaliser des développements de natures différentes en parallèle, sans impact entre les branches.

GC	Gestion de la configuration
EC	Eléments de configuration
RC	Responsable de la configuration

19.1.4 Outil de gestion de configuration

Le référentiel de l'outil Visual source safe est la base de données regroupant l'ensemble des composants logiciels gérés en configuration. Il stocke toutes les actions effectuées sur chacun de ces composants. Il existe un référentiel par projet ou par groupe d'applications liées entre elles. *Voir le guide DSIG404 – VSS*.

La version des documents de type Business Object est gérée de manière manuelle selon la procédure décrite au chapitre 5.1.5.4.

La version des référentiels Business Object est gérée de manière manuelle selon la procédure décrite au chapitre 5.1.5.4

La version des packages PL/SQL, des tables, de tous les objets base de données est gérée de façon manuelle selon la procédure décrite au chapitre 5.1.5.4.

La version de la documentation est gérée de manière manuelle selon la procédure de gestion documentaire décrite dans le document DSIP117 et stockée dans la SCP.

19.1.5 Références

- DSIS404 – Gestion de la configuration
- DSIP112 – Gestion des modifications
- DISG404 - VSS
- DSIP102 – nommage des documents projets
- DSIP103 – Structure de contrôle projet
- DSIP117 – Gestion documentaire

19.2 acteurs et roles

Les processus de GC mis en place par les équipes du SAD doivent être rigoureusement respectés. La qualité des livraisons est liée à la volonté de toute une équipe de se conformer à la discipline et à la méthodologie mises en place dans le cadre de ce projet.

L'utilisation du processus de GC est différente suivant l'activité de la personne. Un acteur du projet peut avoir plusieurs rôles parmi ceux définis ci-après.

Remarque : Selon la taille des projets, le responsable de la configuration, l'intégrateur et le chef de projet pourra être une seule et même personne.

19.2.1 Le Chef de Projet - CP

Le chef de projet coordonne et pilote son équipe et l'ensemble du processus GC. Il est garant de la mise en œuvre de la gestion de configuration sur le projet, alloue les ressources requises pour cette activité et, valide le PGC rédigé par le Responsable de la configuration

Il est « coresponsable » de l'application du Plan GC.

19.2.2 Le Responsable de la gestion de configuration - RC

- élabore le Plan GC, le maintient, le partage avec tous autres acteurs du projet et, par délégation, le valide
- effectue les habilitations des utilisateurs dans le Référentiel GC
- administre les outils de gestion de configuration
- définit les environnements
- réalise les « version de base » et assure la cohérence de la réalisation des paquets livrables
- communique et informe les développeurs des évolutions sur les activités de gestion de configuration les concernant
- met à jour le Plan GC lors de chaque évolution du processus

19.2.3 L'intégrateur - RI

Il est chargé de récupérer et de valider les modifications effectuées par les développeurs puis de les intégrer afin de construire une version de base cohérente (tests d'intégration).

Il rend compte de son activité au Responsable GC et au Chef de Projet.

Il a pour rôle de :

- Récupérer les modifications par lots cohérents depuis les branches de développement
- Réaliser les tests d'intégration
- Mettre à jour les spécifications techniques
- Communiquer et informer les développeurs des évolutions sur les activités de gestion de configuration les concernant

19.2.4 Le développeur - DP

Il travaille dans son espace de travail, uniquement sur des révisions d'articles en configuration. Il ne modifie pas de configurations.

Il rend compte de son activité au Responsable GC et au Chef de Projet. De ce fait il contribue activement au processus d'intégration continue.

Il doit pouvoir (en fonction de ses droits) :

- Créer une branche de développement dans le Référentiel. Concrètement, le développeur accède au référentiel de GC, sélectionne tous les objets nécessaires à sa modification et stocke ces objets dans une branche temporaire
- Effectuer le rapatriement des sources associées aux objets dans son environnement de travail
- Mettre à jour sa copie locale en intégrant les modifications présentes dans le Référentiel mais encore absentes de sa copie locale
- Mettre à disposition ses modifications dans le Référentiel pour validation et partage
- Ajouter de nouveaux fichiers ou répertoires
- Afficher les différences entre la version locale et celle du Référentiel

- Consulter l'historique des révisions (ou des lots)
- Visualiser des différences entre révisions (ou lots)

19.3 Norme d'utilisation

19.3.1 Arborescence

19.3.1.1 Dans le cas d'un développement réalisé par le SAD

L'arborescence de configuration de l'ensemble des projets du SAD doit être identique et suivre une même logique. Elle doit permettre, à tout moment, de revenir à une version ou de connaître les éléments constituant une version. Elle doit également permettre de gérer à la fois de la maintenance corrective ou évolutive tout en garantissant une intégrité des versions.

Dans la SCP, dans le répertoire Livraisons, créer manuellement 3 répertoires :

- Production (tronc)
- Développement (pour gérer les développements en cours)
- Maintenance (pour gérer les opérations de maintenance)

Le répertoire Production est structuré comme suit :

- V X.X.X (identifie la version en production)
  - BDD
  - Ecrans
  - Editions
  - Rapport
  - Autres
  - Fichier Excel constituant le suivi de la livraison (DSIT504)

19.3.1.2 Dans le cas d'un développement non réalisé par le SAD

Seul le fichier de suivi des livraisons et inventaires sont présents dans le répertoire suivi des livraisons de la SCP.

19.3.2 Fonctionnement

Une branche est dénommée : BM (branche de maintenance) ou BD (branche de développement) – [N° de la demande] – Version

19.3.2.1 Dans le cadre d'une maintenance corrective

- Dupliquer les produits que l'on doit modifier dans le répertoire Maintenance.
- Créer le répertoire BM – N° de la demande – Version
- Qualifier, recetter et valider la correction
- Dupliquer dans le tronc Production du répertoire VX.X.X en VX.X.X+1
- Reporter les modifications du répertoire BM – N° de la demande – Version. Ce report se fait de façon manuelle ou à travers l'outil VSS.
- Mettre à jour le suivi des livraisons ou la matrice de traçabilité

19.3.2.2 Dans le cadre d'une maintenance évolutive

- Dupliquer les produits que l'on doit modifier dans le répertoire Développement.
- Créer le répertoire BD – [N° de la demande] – Version

- Qualifier, recetter et valider le développement
- Dupliquer dans le tronc Production du répertoire VX.X.X en VX.X.X+1
- Reporter les modifications du répertoire BD – [N° de la demande] – Version. Ce report se fait de façon manuelle ou à travers l'outil VSS.
- Mettre à jour le suivi des livraisons ou la matrice de traçabilité

19.3.3 Nommage

Type	Nommage	Exemple
BRANCHE (Maintenance)	BM – N° de la demande ou du BUG – VX.X.X	BM – 1239 – V1.0.1
BRANCHE (Développement)	BD – [N° de la demande] - N° de la version en cours de préparation	BD – 1.2.0

19.3.4 Gestion de l'inventaire et des livraisons

19.3.4.1 Principe

L'inventaire de configuration (DSIT503) est maintenu à jour en temps réel pour ce qui est de création ou de la suppression des articles des configurations. Il est maintenu à jour au besoin et par lot pour la traçabilité entre les articles de configuration.

Le suivi des livraisons (DSIT504) est mis à jour et complété à chaque livraison logicielle ou documentaire.

19.3.4.2 Inventaire

L'inventaire permet d'effectuer une photo à un instant t de ce qui constitue la version en production. Il doit identifier les versions de l'application en production avec l'ensemble des composants liés (documentaire ou logiciel) mais il doit également identifiés les versions des outils système qui permettent le fonctionnement de l'application (version de IE, de la base de données, de l'outil BO, …).

Chaque enregistrement dans l'inventaire de configuration regroupe les éléments suivants :

Système

Nom	Description	Obligatoire
N°	Identifiant séquentiel du numéro dans l'inventaire	O
Label	Nom du produit	O
Version	Version du produit	O
Version de l'application	Version de l'application dans laquelle le composant s'inscrit	O
Géré par	Identifie la personne gérant le composant	N
Demande	Identifie si le composant est lié à une demande précise (cas d'une évolution)	N
Description	Décrit le composant synthétique	N
Commentaire	Décrit le commentaire	N

Logiciel

Nom	Description	Obligatoire
N°	Identifiant séquentiel du numéro dans l'inventaire	O
Label	Nom du composant	O
Type	Type du composant (document, librairie, web, win32, BO, Script SQL, Ecran, Edition)	O
Version	Version du composant	O

Version de l'application	Version de l'application dans laquelle le composant s'inscrit	O
Géré par	Identifie la personne gérant le composant	N
Demande	Identifie si le composant est lié à une demande précise (cas d'une évolution)	N
Description	Décrit le composant synthétique	N
Commentaire	Décrit le commentaire	N

19.3.4.3 Suivi des livraisons

Le suivi des livraisons permet de connaitre l'ensemble des versions applicatives livrées ainsi que les documents composant ces livraisons.

Chaque enregistrement dans le suivi des livraisons regroupe les éléments suivants :

Nom	Description	Obligatoire
N°	Identifiant séquentiel de la livraison	O
Version	N° de la version livrée	O
Responsable	Qui est responsable de la livraison	O
Origine	Origine de la livraison (BUG, Evolution, contractuelle, autre)	O
Date de la livraison	Date de la livraison	O
Liée à la demande	Identifie si la livraison est liée à une demande précise (cas d'une évolution ou d'un BUG)	N
Vers du PP		O
Vers des exigences		O
Vers des SFG		O
Vers des SFD		O
Vers du plan de test		O
Vers des SFT		O
Description	Décrit le composant synthétique	N
Commentaire	Décrit le commentaire	N

19.4 Droits d'accès et Responsabilités

19.4.1 Matrice des responsabilités

Actions	Qui fait	Qui valide
Définir le plan de GC	RC	CP
Mise en place de l'outillage GC	RC	CP
Gestion des changements	RC	CP
Former sur la configuration	RC, RI	CP
Gérer les versions	RI	RC

Gérer les branches	RC	CP
Gestion des audits	RC, CP	CP
Maintient de la configuration	RI	RC, CP
Utilisation de la configuration	DP	RC
Gestion inventaire de configuration	RC	CP

19.4.2 Droits d'accès à l'arborescence de configuration

	TRONC	**BRANCHES**
Lecture	CP, DP, RC, RI	CP, DP, RC, RI
Ecriture	CP, RC, RI	CP, DP, RC, RI

19.5 activites

19.5.1 Identification des éléments de configuration

Que doit-on gérer en configuration ? Tous les éléments de base nécessaires à la construction d'une version finale d'un logiciel devront être pris en compte (sources, documents, script, propriétés, données de paramétrage, binaire, exécutables)

Cette identification décrit les éléments de configuration officiellement contrôlés au cours du projet. Les éléments de configuration sont des éléments clés qui sont produits et gérés au cours du projet. Ce sont les composants logiciels et la documentation.

- Un produit livrable est fait d'un (la plupart du temps) ou plusieurs éléments de configuration.
- Une base/version de référence est établie à chaque livraison ou bien à chaque jalon (VPD) du projet. La liste des documents et des composants logiciels avec leur version qui crée la version de base est copié dans la matrice de traçabilité des exigences.

Au démarrage d'un projet le responsable de la configuration effectue un inventaire des différents éléments de configuration. Le modèle de cet inventaire est le document DSIT503 – inventaire.xls. Il est stocké directement dans le répertoire de la SCP, Livraisons. (cf chapitre 3.4)

Cet inventaire doit être mis à jour par le responsable de la configuration durant tout le projet, et ce à chaque nouvelle livraison. Il permet de connaître les versions de chaque article composant la version en production.

La configuration et la gestion des environnements de développement, recette, qualification, pré-production et production est géré et est sous la responsabilité de l'exploitation.

Les éléments gérés en configuration sont listés dans le tableau ci-dessous.

Produit	**Unique ID et emplacement**	**Phase du projet**	**Responsable**
Demande client	Id de la demande dans Gedem ou autres outils	Cas fonctionnels et exigences	CP
Plan projet	dans la SCP	Cas fonctionnels et exigences	CP
Conception générale	dans la SCP	Conception générale	CP
Architecture	dans la SCP	Conception générale	CP
Spécifications détaillées	dans la SCP	Spécifications détaillées	CP

Spécifications techniques	dans la SCP	Spécifications détaillées	CP
Cahier de recette	dans la SCP	Spécifications détaillées	CP
sources et exécutables	dans VSS dans la SCP	Réalisation	RGC

Dans le cas d'un développement réalisé par un prestataire, la gestion de l'inventaire des composants logiciels est gérée par le prestataire et est sous sa responsabilité.

19.5.2 Nommage des éléments de configuration

Le nommage des documents et logiciels est géré conformément aux documents:

- DSIS404 – Gestion de la configuration
- DSIP102 – Nommage des documents projets
- DSIP103 – Structure de contrôle projet
- DSIP117 – Gestion documentaire

19.5.3 Stockage des éléments de configuration

Chaque projet est stocké sur le portail CMMI (http://portail-sites.cr-paca.fr/SiteDirectory/dsi/cmmi/default.aspx) dans le répertoire projet.

Dans ce répertoire se trouve les répertoires de chacun des projets du SAD.

La référence de l'ensemble des projets se trouvent dans le fichier DSISAD – Inventaires des projets.xls sous le répertoire Projet.

Eléments	**Format**	**Emplacement de stockage**	**Stockage de la version précédente**	**Sécurité**
Documentation	Word, Excel, Powerpoint .sql .pl/sql Rapport BO	SCP	Dans le même répertoire de la SCP	Droits d'accès individuels
Source code	.html .NET ….	VSS	VSS	Accès en écriture restreint aux développeurs et à l'équipe projet

19.5.4 Gestion des éléments hors gestion de configuration

Cette section regroupe les articles qui ne sont pas sous gestion de configuration, mais seulement versionnés. Ces documents seront contrôlés et maintenus dans la SCP.

• **Nom du document**	• **Responsable**
• Plan de gestion projet	• CP
• Plan projet	• CP
• Plan de gestion de la configuration	• CP
• Revues	• CP
• Planning projet	• CP
• Estimation	• CP

• Risques	• CP
• Suivi des actions	• CP
• Matrice de traçabilité	• CP
• Stratégie des tests	• CP
• Plan de tests	• CP

19.5.5 Contrôle des éléments

Pour les documents:

- DSIP102 – nommage des documents projets
- DSIP103 – Structure de contrôle projet
- DSIP117 – gestion documentaire

Pour les logiciels

- Gestion de la configuration dans l'outil VSS (DSIG404 – VSS)

19.5.5.1 Version de référence

Une version de référence est une base fiable sur laquelle on peut s'appuyer (reconstitution de version), c'est un point stable dans le processus de livraison, une photo à un instant T de l'application

Comment l'identifier ?

- En définissant
  - les objectifs de cette nouvelle version de base
  - son cycle de vie et donc les périodicités des générations des versions de base
  - Le rôle des différents acteurs GC tout au long de ce cycle de vie
  - Le système (lot fonctionnel) et sous systèmes éventuels
  - La méthode d'archivage après livraison

Le cycle de vie d'une version de base est matérialisé par différentes itérations.

- Depuis la phase « cas fonctionnels et exigences » jusqu'à la phase de Validation
- A cette étape on initialise donc l'organisation du processus de GC, en posant les briques de base du PGC.

L'ensemble formé des éléments gérés en configuration, des demandes de changements, de la version de référence et des versions de base constituent le référentiel projet.

L'ensemble formé des espaces de travail et, d'intégration travaillent avec le référentiel projet

Le processus de versionning des parties logiciels (écrans, html, éditions) est effectué de manière automatique par l'outil de gestion de configuration logiciel.

A tout moment il est possible aussi bien de visualiser les versions les plus récentes des articles sujets à la gestion de configuration que les versions antérieures de celui-ci, pouvant inclure la comparaison de deux révisions distinctes, dans le but de voir les différences entre les deux.

La version de référence est tracée dans l'outil de suivi des livraisons (DSIT504). Dans ce document se trouve associé la version du produit, la version des documents liés à cette version. (cf chapitre 3.4)

Chaque version de référence est liée à une mise à jour de l'inventaire.

Le processus de versionning des documents « word, excel » et des documents de type sql, pl/sql est géré de façon manuelle. Ils font également partis de la version de référence.

19.5.5.2 Gestion manuelle de la configuration

A chaque modification d'un élément de configuration (doc, package, etc …) non gérés à travers un outil de GC la règle suivante doit être appliquée.

- Archivage et sauvegarde de la version en cours, dans la SCP dans le répertoire adéquate, appelé version n. Cette version ne sera plus touchée.
- Duplication de la version n en la nommant version n+1
- Travail sur la version n+1
- Recette/validation de la version n+1
- Mise en production la version n+1 qui devient la version courante du projet et qui devient, en vue de prochaine modification la version n.

Ainsi si on constate des effets indésirables issus de la version n+1, on pourra toujours revenir de manière simple à la version n.

19.5.5.3 Gestion automatique de la configuration

Voir le guide DISG404 – VSS.doc

19.5.5.4 Identification des versions/révisions

La gestion d'articles s'effectue au travers de révisions.

Pour un article donné, la première version représente le moment où il est incorporé au système de gestion de configuration.

Les changements successifs d'articles sont étiquetés comme révision de ce dernier. Dans l'outil de gestion de configuration, à chaque changement correspond un identifiant interne de révision. De plus, tout changement doit obligatoirement être accompagné d'un commentaire.

De cette manière, le système de gestion de la configuration maintient un historique de toutes les évolutions de l'article, pouvant être accédé à n'importe quel moment et sur la révision voulue.

L'outil de gestion de configuration garantit la traçabilité totale : qui, pourquoi et quand.

Quand un article est ajouté à la gestion de la configuration ou qu'il est modifié, cette action est accompagnée obligatoirement d'une description de cette action.

De plus, le fait de procéder à l'ajout ou la modification d'un article implique de manière intrinsèque une étiquette interne d'identification, aussi bien sur l'article lui-même que sur la modification effectuée, sur la base de la révision de cet article. Cette identification sera effectuée à l'aide d'un cartouche d'entête qui tracera les différentes modifications effectuées et leur version.

Ce cartouche pourra se présenter comme suit :

- Sujet de la modification :
- Liée à la demande N° :
- Date :
- Traité par :
- Action réalisée :
- Versions/révisions :

19.5.5.5 Codifications des versions/révision

Logicielle

La version sera de la forme **chiffre.chiffre.chiffre.**

Exemple : 1.0.0

Chaque modification d'une application en production entraînera l'évolution de son numéro de version.

- Le premier chiffre représente la livraison d'une application dans sa globalité. Il changera uniquement dans le cas d'évolutions majeures ou de modifications impactant l'application dans sa globalité (cas d'une migration technique).

- Le second représente un module ou un lot de l'application ; une livraison majeure ou un ensemble de correctif, évolution suffisamment important.
- Le troisième chiffre représente des corrections de bugs ou des petites évolutions (rajout d'une virgule, changement de libellé, déplacement d'un champ, etc.). C'est la révision d'une version.

Exemple 1 : Livraison d'une nouvelle application YYY.

Sa version initiale sera : 1.0.0.

Si correction de bugs mineurs, la version passera en 1.0.1, puis 1.0.2, etc.

S'il y a beaucoup de Bugs à corriger ou une évolution, correction importante à effectuer alors on regroupera la livraison dans une livraison d'un lot/module. L'application passerait alors en 1.1.0.

Si l'application est livrée en lots : le premier lot sera 1.1.0 puis 1.2.0 etc …

Si l'application est fortement impactée dans son ensemble et qu'une livraison globale est utilise, la version passera en 2.0.0.

Documentaire

Voir le document DSIP117 – Gestion documentaire. Chapitre 2.4.2

19.5.5.6 Procédure de livraison

Chaque livraison fait l'objet de l'établissement d'une fiche livraison contenant, entre autre, la liste de tous les composants livrés et leur version (VPDs).

Chaque livraison fait l'objet de mise à jour de l'inventaire des éléments de configuration via le fichier DSIT503 avec la ou les dernières versions en production et une mise à jour du fichier de suivi des livraisons DSIT504 – Suivi des livraisons.xls qui se trouvent dans répertoire SCP/Livraison/Production représentant une version livrée.

19.5.5.7 Gestion des demandes

Consulter également le document DSIP112 – Gestion des modifications.

Dans ce chapitre ne sont traitées que les demandes demandant une modification logicielle.

Une demande peut arriver par l'outil de gestion des demandes GEDEM ou bien par une autre voie telle que le mail.

Elle doit être tracée par le chef de projet dans l'outil DSIT501 – suivi action.xls (onglet demande ou évolution)

Si la demande concerne une évolution majeure sur le projet, avec un impact lourd en terme de charge et délai celle devrai être validée par le client et le Responsable projet.

Identification des demandes.

Avant de qualifier la demande, le chef de projet identifie :

- L'origine : client, internes, sous traitants ou partenaires
- Différencie les demandes ne nécessitant pas de modification de logiciel (demande de support)
  - une demande d'information
  - une demande d'assistance contractuelle ou non contractuelle
  - une demande de formation
  - une réclamation
- Des demandes nécessitant une modification de logiciel, donc un processus de développement ou une modification de paramétrage
  - une anomalie (bloquante ou non bloquante): maintenance corrective
  - une demande d'évolution: maintenance évolutive
  - une demande d'amélioration de l'application : maintenance préventive

Qualification des demandes

Le chef de projet qualifie donc cette demande de changement logicielle, l'attribue à un développeur et lui affecte une date de fin de réalisation, dans le respect des indicateurs définis.

Cycle de vie d'une demande

	PHASES/MISSIONS	Intervenants	Méthode appliquée	Produit utilisé. Module
10	Demande d'intervention	client		
20	Prise en compte et qualification de la demande : anomalie évolution information formation …	SAD	PGC du Projet	Gestion des demandes (GEDEM) DSIT501 – suivi action.xls et / ou Défaut lors de recette
30	Recherche éventuelle de l'impact de la modification	CP RC		
40	Constitution du lot dans l'environnement de développement	Développeurs,		VSS Manuelle
50	Développement/Maintenance du code	Développeurs		Outils de développements
60	Tests unitaires : BdD à jour	Développeurs		
70	Application de la modification sur la branche principale	Développeurs,	PGC du Projet	
80	Constitution de l'environnement de tests/recette	Intégrateurs, CP	PGC du Projet	
90	Tests d'intégration dans l'environnement de Recette Plan test Tests de montée en charge	CP, Testeurs	Portail CMMI	
91	Tests fonctionnels et de non régression dans l'environnement de Recette	CP, Testeurs	Portail CMMI	
100	Constitution du lot livré par mise à disposition de ce lot de modification pour recette client Fiche de livraison	RC CP	PGC	
110	Recette client Fiche de livraison signée	CP, Utilisateurs		
120	Mise en production	CP Client		Produit
130	Clôture de la demande	CP SAD	PGC du Projet	Gestion des demandes (GEDEM) Et / ou Recette

19.5.5.8 Gestion des développements

Pour maintenir une version en exploitation et réaliser en même temps des évolutions sur cette version, deux cycles de développement vont donc évoluer en parallèle.

La maintenance corrective est continuelle et doit être réalisée au fur et à mesure des incidents. Le projet de développement est réalisé sur une période plus longue et doit être livré en un seul lot sous forme d'une nouvelle version.

De ce fait, une branche de développement appelée Développement sera créée pour la maintenance évolutive ; la branche principale sera utilisée pour recevoir les développements validé et en production.

Le RC veillera que à ce choix soit bien respecté.

Report des corrections

Il est courant qu'un projet doive maintenir une version en production (Version dite N-1), construire une version applicable cible (Version N) et engage des études sur une version à venir (Version N+1).

Il peut s'avérer nécessaire de reporter des corrections réalisées sur une version dans les autres versions. Cette activité est dénommée « Report de correction ».

Plusieurs cas de figures existent :

- Des anomalies détectées en version N-1 entrainent des modifications qu'il est important de corriger dans les versions N et N+ 1. Les anomalies sont corrigées sur la branche de maintenance, elles sont reportées sur la branche développement courante de la version cible (Version N) et sur la branche de développement de la version future (N+1). Les vérifications et validations sont également à planifier pour ces deux versions.
- De même, un défaut découvert en version N, peut être à reporter en version N+1. Dans ce cas la correction est réalisée en branche de développement de N, testée et reportée en branche de développement de la N+1.
- Le rétro-report, consistant à reporter une correction de défaut découvert en version N ou N+1 sur une version antérieure peut également être considéré. Cela est notamment le cas si le défaut entraine des corruptions de données ou de dépendance entre objets difficiles à corriger par la suite.

Le RC est en charge de définir ave le CP et ou le RI, pour les corrections à fort impact (structure de base, cohérences des données en production, arrêt de service), les reports de correction à effectuer et les branches où s'effectueront les reports.

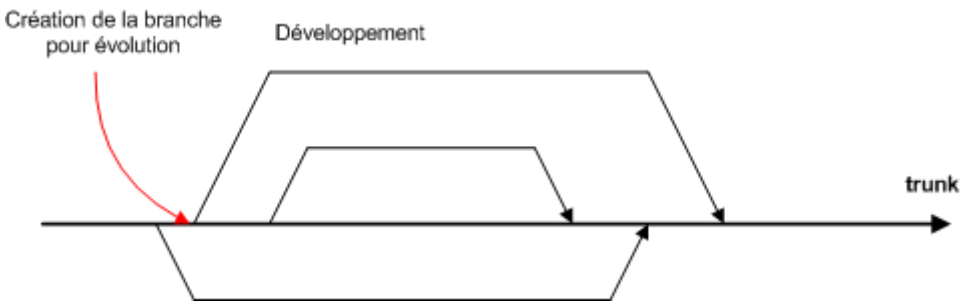
Report des fonctionnalités

De même que pour les corrections, il est possible que des évolutions mineures soient apportées sur des versions majeures ou mineures.

Il est espéré que ces évolutions soient toujours présentes dans les versions futures. Il sera donc nécessaire de planifier les reports de fonctionnalité, de même que les reports de corrections.

Réalisation des développements

Suite à la demande d'évolution ou de correction d'anomalie, une branche temporaire est créée par le développeur (cf chapitre 3.2).



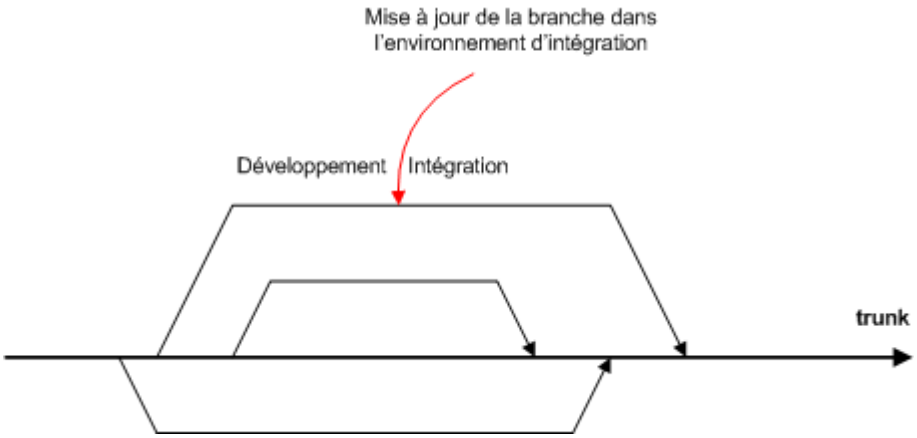
On n'utilise pas le verrouillage des objets mais la méthode « copier-modifier-fusionner » en particulier dans le cas de maintenance corrective et évolutive.

Cette branche est identifiée par le n° de la demande et sa version. Elle contient uniquement les composants qui vont évoluer et les données nécessaires à l'exécution et au test du composant.

Le développeur récupère cette branche dans son environnement de développement et réalise les modifications nécessaires.

Ces modifications sont ensuit reportées dans l'environnement d'intégration par le développeur pour validation :

- Si les tests sont valides, le développeur avertit la MOA pour livraison en recette
- Sinon, il reprend ses modifications à partir de son environnement de développement



19.5.5.9 Implémentation des changements

Cette tache a pour objectif de réaliser l'intégration des développements réalisés par les différents développeurs.

Elle est effectuée « au fil de l'eau » sous la responsabilité de l'intégrateur.

Préparation de la qualification

Cette étape consiste à préparer la qualification et la livraison des composants concernés par un lot de modifications donné. Les sources impactées par cet ensemble de modifications constitueront le contenu d'une livraison.

Qualification

Cette étape consiste à qualifier l'ensemble des sources concernés par une livraison d'un lot de modifications donné, les éventuels outils de reprise des données et à vérifier la non régression fonctionnelle.

Livraisons d'un correctif en recette

Les étapes successives effectuées sont les suivantes :

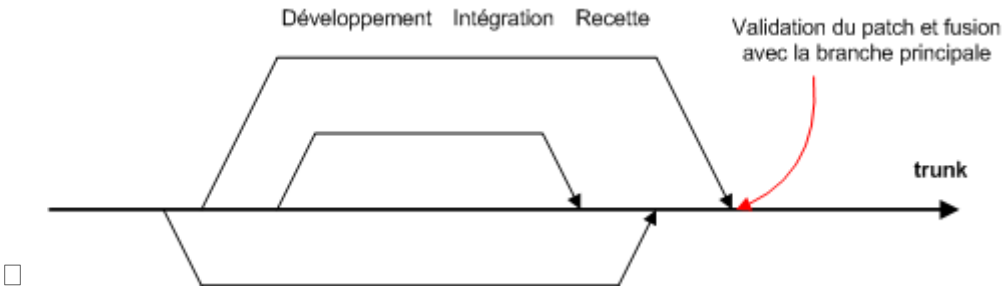
- Rédiger les documents à livrer :
  - Fiche de livraison et validation de la livraison (VPDs)

Le développeur notifie par mail l'équipe d'exploitation pour le déploiement du patch avec la documentation d'installation.

Le développeur exécute le scénario.

Si le patch est validé, une fusion avec résolution des conflits est effectuée sur la branche principale (trunk).

Si le patch est annulé, les modifications réalisées sur la branche de développement ne sont pas reportées sur le trunk.



Lors de la création du tag (voir DSIG404 – VSS.doc), on précise dans le commentaire :

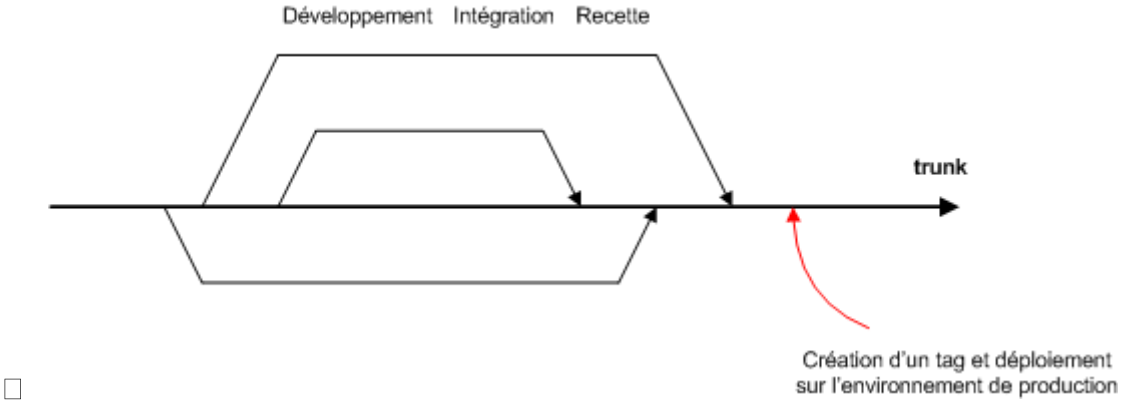
- L'origine de la modification logicielle :
  - Correction d'anomalie (ANO)
  - Demande d'évolution (EVO)
- La version en cours
- Le n° de la demande
- Le libellé

De ce fait, tous les objets référencés dans ce tag auront la même identification (n° de révision et commentaire identique).

Déploiement d'une version dans l'environnement de production

Les développements validés sont régulièrement déployés dans l'environnement de production :

- Pour un développement corrigeant une anomalie, le tag nouvellement créé suite à la modification est déployé dans l'environnement cible
- Pour des évolutions moins urgentes, on copie la version de référence dans un tag avec toutes les révisions identifiées pour les patchs à installer puis on déploie ce tag dans l'environnement cible



La mise à jour des autres environnements est réalisée de façon similaire

Les modifications sont implémentées sous le contrôle du responsable de la configuration en accord avec les règles suivantes :

Pour les documents :

- Répertoire : SCP (voir DSIP103 – Structure de contrôle projet)
- Version : conforme au guide (voir DSIP117 – gestion documentaire)

Pour la partie logicielle

- Gestion de la configuration : VSS (voir DSIG404 – VSS)

Pour le suivi

- Note de livraison, mise à jour inventaire, mis à jour suivi de livraison, mis à jour matrice de traçabilité

19.5.5.10 Gestion des incidents/modifications

Cette partie est gérée comme la gestion des demandes.

19.5.5.11 Gestion des environnements

La gestion des environnements et leur configuration est effectuée et est à la charge de l'exploitation.

Lors de l'inventaire au début du projet les versions de chaque système, base de données sont répertoriés.

Les développements projets sont prévus pour fonctionner dans les spécifications de ces environnements techniques.

Si des changements sont effectués l'exploitation doit le chef de projet ou le responsable de la configuration des changements afin que ceux-ci puissent identifier les éventuels impacts.

Une mise à jour de l'inventaire sera alors effectuée par le RC.

19.5.5.12 Gestion de la documentation

Les documentations d'exploitation et utilisateurs seront gérées en configuration dans la SCP.

En voici la liste :

- La note de livraison
- Le guide d'installation de l'application
- Guide utilisateur et d'exploitation

L'outil de production de la documentation est Word.

19.5.6 Autre gestion de configuration

S'il n'y a pas d'outils pour créer et stocker la documentation, la gestion de la configuration sera suivie à travers des versions de base identifiées dans un fichier Excel. Ce fichier sera stocké dans la SCP (Livraison). Des réunions régulières seront organisées avec l'équipe projet afin de passer en revue la liste des éléments de configuration (documents, logiciels). Le résultat de ces réunions sera une mise à jour du fichier Excel.

19.5.7 Audit de configuration

Les audits d'intégrité sont des audits informels afin de vérifier si les éléments de configuration (documents et composants logiciels) sont correctement gérés et si l'intégrité de tous les éléments de configuration est maintenue:

- Tous les documents sont approuvés au bon moment et conservé à la bonne place?
- L'historique en première page est à jour avec la raison de changements?
- Le nommage et le versionning sont en phase avec la procédure de documentation?
- Les composants logiciels modifiés sont correctement stockés comme nécessaires à l'accomplissement de la version exécutable?
- Les changements sont correctement identifiés dans des composants logiciels ("cartouche", des marqueurs, des raisons, date, nom du développeur)?
- Les documents de conception sont modifiés et sont en phase par rapport aux composants logiciels?

L'audit d'intégrité sera effectué par le contrôleur de configuration sur une base régulière afin de vérifier l'intégrité des éléments de configuration (maintenances documentation et composants logiciels) pour des projets ou sur un périmètre défini (une ou plusieurs modifications, exigences), y compris la mise en œuvre correcte et la traçabilité des changements. Ces audits seront effectués avec:

- le plan de gestion de configuration,
- les outils de configuration de la documentation et des logiciels,
- Les exigences de la matrice de traçabilité
- Les vérifications de l'intégrité de la configuration

Les audits seront exécutés durant les RGP entre le responsable qualité et le chef de projet. Les résultats seront stockés et suivis dans les rapports d'examen de l'AQ

19.5.8 Gestion de la configuration d'un logiciel acheté

Si un logiciel est acquis à un sous-traitant, éditeur, client, autre projet ou tout autre source, la démarche pour l'intégrer est la suivante :

Réception du logiciel, teste avec la participation de l'équipe de développement. La gestion de la configuration reste sous la responsabilité du fournisseur.

Les modifications sur le logiciel du fournisseur sont effectuées à travers des achats/commandes à ce fournisseur.

Le fournisseur participe dans les processus de modification du logiciel en tant que assistance et support uniquement

19.6 Planning des activites de GC

Tâche/activité	Participants	Quand
Identification de la configuration		
Définir l'inventaire des éléments de la configuration de la version de base	CP/RGC	Avant la VPD 2
Identifier la documentation	CP/RGC	
Audit de configuration		
Audit de configuration DSIP116	CP/RGC	10-15 jours avant les VPDs
Revue de configuration DSIT102	CP/RGC/RQP	Lors des RGP

19.7 Ressources

Outils	Description	Type d'éléments
VSS	Outil de gestion des sources	Code sources
Intranet - SCP	Structure de contrôle projet sur l'intranet	Gestion du projet
Portail CMMI	Référentiel documentaire sur l'intranet	Documentation technique, fonctionnelle, qualité
GEDEM	Outil de gestion des demandes	Demande client
Microsoft Outlook	Mail	Gestion de projet
MS Project DSIT501	Planning	Planning détaillé du projet
DSIT401, 404	Cahier de recette, contrôle des tests unitaires	Tests

19.8 Plan de gestion de la configuration

Le plan de maintenance est maintenu durant le projet par le chef de projet ou le responsable de la configuration.

Chaque projet doit posséder un plan de gestion de configuration (PGC).

Il est initialisé à partir du modèle DSIM412.

Le chef de projet et le responsable qualité contrôle régulièrement que les activités de gestion de configuration sont correctement effectuées et gérées (outil DSIT102)

19.9 Bonnes pratiques

Ces remarques sont le signe d'une bonne hygiène de développement.

- Toujours faire une mise a jour avant de se lancer dans une modification.
- Ne pas conserver une modification non versionnée pendant une longue durée.
- Ne pas versionner quelque chose qui ne compile pas

19.10 Traçabilité de la configuration

La traçabilité de la configuration est effectuée à travers la matrice de traçabilité, le suivi des livraisons, l'inventaire et l'outil de gestion de configuration Visual Source Safe.

La cohérence de la traçabilité impose que chaque article de configuration :

- Soit lié à une version
- Possède une version
- Soit associé à une exigence dans le cas d'une évolution ou du changement sur une exigence
- Soit lié à une demande dans le cas d'une correction

Evolution d'une exigence, nouvelle exigence	**Cas d'une maintenance corrective**

• Dans le fichier de matrice de traçabilité
  • Mise du fichier avec la nouvelle exigence ou l'exigence modifié (via le suivi des changements).
  • Mise à jour de la version des documents impactés (SFG, SFD, etc …)
  • Mise à jour de la zone fiche de livraison et version, associée à cette exigence.
• Mise à jour documentation
• Mise à jour de l'inventaire
• Mise à jour du suivi des livraisons
• Mise à jour de la SCP si nouvelle version de référence
	• Mise à jour documentation
	• Mise à jour de l'inventaire
	• Mise à jour du suivi des livraisons
	• Mise à jour de la SCP si nouvelle version de référence

19.11 Annexes

19.11.1 Inventaire

N°	Label	Type	Version de l'application	Géré par	demande	Description	Commentaire
S1	CRPaca.Apprentissage						
CI1	CRPaca.Apprentissage.BAL	Librairie	V1.1	BLP		Couche Métier GPEAV2	
CI2	GPEAV2	Win32	V1.1	BLP			
CI3	GPEAV2.ObjAppli	Librairie	V1.1	BLP			
		Document	V1.1	BLP		Cahier des charges	
		Document	V1.1	BLP		Spécifications fonctionnelles	
		Script SQL	V1.1	BLP		Script PL/SQL	
		Script SQL	V1.1	BLP		Script BD	

19.11.2 Suivi des livraisons

N°	Version livrée	Resp.	Origine	Date de la livraison	liée à la N° demande	Vers. du PP	Vers. des exigences	Version des SFG	Version des SFD	Version des SFT	Version des plans de tests	Description de la livraison	Commentaire
1													
2													
3													
4													

20 DOCUMENT 11 – 2.00 – ACCESSIBILITE EN ASP .NET - DOCUMENT DE REFERENCE

20.1 Introduction

Ce document a pour but de présenter dans un contexte d'accessibilité pour les développements WEB du Conseil régional PACA, les directives et critères à respecter ainsi que et les solutions préconisées permettant de répondre aux contraintes d'accessibilité.

20.2 Documents de référence

Les documents officiels ayant servi de base à la rédaction de ce dernier sont :

Document	Lien
Présentation du Référentiel Général d'Accessibilité pour les Administrations	http://references.modernisation.gouv.fr/rgaa-accessibilite/
Référentiel Général d'Accessibilité pour les Administrations RGAA	http://references.modernisation.gouv.fr/sites/default/files/RGAA_v2.2.1.pdf
Référentiel Général d'Accessibilité pour les Administrations RGAA Annexe 1 : Critères de succès	http://references.modernisation.gouv.fr/sites/default/files/RGAA-v2.2_Annexe1-Criteres.pdf
Référentiel Général d'Accessibilité pour les Administrations RGAA Annexe 2 : Tests de conformité au RGAA	http://references.modernisation.gouv.fr/sites/default/files/RGAA-v2.2_Annexe2-Tests.pdf
Référentiel Général d'Accessibilité pour les Administrations RGAA Annexe 3 : Grilles de correspondance entre les critères de succès et les tests de conformité	http://references.modernisation.gouv.fr/sites/default/files/RGAA-v2.2_Annexe3-Grilles.pdf

Remarque : Les recommandations présentes dans ce document deviennent prioritaires par rapport aux autres documents de normes pour tout projet intégrant des contraintes d'accessibilité.

20.3 Contexte

20.3.1 L'accessibilité

20.3.1.1 Définition

Tim Berners-Lee, directeur du W3C et inventeur du WWW définit l'accessibilité ainsi :

" Mettre le Web et ses services à la disposition de tous les individus, quel que soit leur matériel ou logiciel, leur langue maternelle, leur culture, leur localisation géographique, ou leur aptitudes physiques ou mentales".

20.3.1.2 Historique

L'accessibilité Web dépend de plusieurs composantes interdépendantes:

- le contenu Web, c'est-à-dire « l'ensemble de l'information et de l'expérience sensorielle qui est communiquée à l'utilisateur par le biais d'un agent utilisateur, incluant le code ou le balisage qui définit la structure du contenu, sa présentation et les interactions avec lui »;

- les agents utilisateurs qui exploitent le contenu et le restituent aux internautes : navigateurs, plugins, etc.,
- les technologies d'assistance logicielles (lecteurs d'écran, logiciels d'agrandissement, etc.) et matérielles (claviers adaptés, commutateurs, etc.),
- les outils d'édition du contenu,
- les outils d'évaluation de l'accessibilité Web,
- les développeurs de contenu, d'agents utilisateurs, de technologies d'assistance, d'outils d'édition et d'évaluation.

Pour permettre le développement de l'accessibilité à travers ces composants, le W3C a créé des recommandations à travers le projet Web Accessibility Initiative (WAI) créé en 1996.

20.3.1.2.1 La norme d'accessibilité internationale

Les objectifs du WAI sont les suivants :

- Vérifier l'accessibilité des technologies W3C
- Produire des recommandations
- Contribuer aux développements d'outils
- Coordonner la R&D sur l'accessibilité

Les recommandations du WAI sont :

- WCAG (Web Content Accessibility Guidelines) :
- Les recommandations en ce qui concerne le contenu s'adressent à tous les distributeurs de contenu sur le Web. Ces directives se nomment les *Web Content Accessibility Guidelines*. Succédant aux WCAG1.0 publiées en 1999, **les WCAG2.0 sont la recommandation officielle depuis le 11 décembre 2008.**
- ATAG (Authoring Tools Accessibility Guidelines) :
- Recommandations destinées aux outils d'éditions de contenus web (CMS, Editeurs type dreamweaver...)
- UAAG (User Agent Accessibility Guidelines) :
- Recommandations destinées aux Agents Utilisateurs (navigateurs web, lecteur multimedia, plugins et lecteurs d'écran...)
- WAI-ARIA(Accessible Rich Internet Applications) :
- Spécifications pour rendre des applications Internet riches (AJAX...) compatible avec les Technologies d'assistance.

20.3.1.2.2 Les méthodes d'application

La prise en compte des directives fixées dans le WCAG nécessite d'adopter une méthode d'évaluation et de déploiement. Ces méthodes peuvent être définies à l'échelle nationale ou à une échelle géographique plus large, par des organismes privés ou publics. Par exemple :

- Accessiweb (France) et Anysurfer (Belgique) sont des méthodes produites par des organismes de labellisation privés, inspirées des WCAG2.0 ;
- le référentiel général d'accessibilité pour les administrations (RGAA) (France) est un référentiel public de déploiement des WCAG2.0 ;

- la « méthodologie unifiée d'évaluation de l'accessibilité du Web » (UWEM) est une méthode européenne permettant une évaluation de l'application de WCAG1.0 aux niveaux A et AA.
- Les directives ATAG peuvent s'appuyer partiellement sur les méthodes d'application de WCAG. Une méthode d'application ATAG1.0 a été développée en France au sein d'Accessiweb. Concernant ATAG2.0, des techniques d'implémentation sont en cours de rédaction.

20.3.2 L'accessibilité en France

20.3.2.1 La réglementation obligatoire et la méthode d'application

20.3.2.1.1 Aspect réglementaire

En France, l'article 47 de la loi n° 2005-102 du 11 février 2005 pour l'égalité des droits et des chances, la participation et la citoyenneté des personnes handicapées, fait de l'accessibilité une exigence pour tous les services de communication publique en ligne de l'État, des collectivités territoriales et des établissements publics qui en dépendent. Il stipule que les informations diffusées par ces services doivent être accessibles à tous. Le Référentiel Général d'Accessibilité pour les Administrations (RGAA) permettra de rendre progressivement accessible l'ensemble des informations fournies par ces services.

Le décret d'application a été publié le 14 mai 2009 (décret n°2009-546 du 14 mai 2009)

- « Il décrit les modalités de contrôle permettant aux collectivités publiques mentionnées au premier alinéa de vérifier que leurs services de communication publique en ligne sont bien conformes à ces règles ».
- Il s'appuie sur le Référentiel Général d'Accessibilité pour les Administrations (RGAA) pour les modalités techniques de mise en œuvre.
- Il impose une mise en œuvre de l'accessibilité dans un délai de deux ans (à partir de la publication du décret) pour les services de communication publique en ligne de l'Etat et des établissements publics qui en dépendent, et de trois ans pour les services de communication publique en ligne des collectivités territoriales et des établissements publics qui en dépendent.

L'arrêté a été publié le 29 octobre 2009.

- « Le référentiel général d'accessibilité pour les administrations, prévu à l'article 1er du décret du 14 mai 2009 susvisé, est approuvé. »

20.3.2.1.2 RGAA comme méthode d'application – définition et objectifs

Le RGAA est un recueil de règles et de bonnes pratiques qui visent à améliorer l'accessibilité des sites Web des administrations.

Il constitue un référentiel d'application des standards internationaux WCAG 2 0.

Il ne constitue pas une nouvelle norme offre des explications et des tests pour la vérification de la mise en œuvre des standards internationaux d'accessibilité.

Il propose notamment :

- une présentation des règles pour l'accessibilité des contenus Web (WCAG 2.0)
- un guide d'accompagnement, destiné aux responsables des projets de mise en conformité
- des critères de succès, ensemble d'exigences que doit respecter un site Web pour être accessible
- des tests de conformité, qui permettent de vérifier l'accessibilité des contenus Web.

20.3.2.2 Directives, critères, niveaux

Les règles pour l'accessibilité des contenus du Web, qui sont proposées à travers ce référentiel, reposent sur le WCAG 2.0 (Web Content Accessibility Guidelines 2.0), rédigé par la WAI. Les WCAG 2.0 adoptent une approche thématique proposant 12 directives structurantes selon 4 principes fondamentaux :

1. des contenus perceptibles,
2. des contenus utilisables,
3. des contenus compréhensibles,
4. des contenus robustes,

Représentant 61 critères de succès WCAG 2.0.

Les 12 directives sont présentées ci-dessous.

- Cadres
- Couleurs
- Formulaires
- Images
- Multimédia
- Navigation
- Présentation
- Scripts
- Standards
- Structure
- Tableaux
- Text

Par ailleurs, trois niveaux de conformité ont été définis : A (le plus bas), AA et AAA (le plus élevé). »

Chacune des 12 directives se décompose en un ou plusieurs « critère de succès » de niveau A, AA ou AAA, qui deviennent la base d'évaluation d'accessibilité du site.

187 tests de conformité issus des « techniques et failures », documents non normatifs associés aux WCAG 2.0 permettent la vérification de conformité d'une page.

Niveau	Définition de la conformité	Critères
A	Pour une conformité de niveau A (le niveau minimal), la page Web satisfait à tous les critères de succès de niveau A ou une version de remplacement est fournie	Critères de succès essentiels pouvant raisonnablement s'appliquer à toutes les ressources Web.
AA	Pour une conformité de niveau AA, la page Web remplit tous les critères de succès de niveau A et AA ou une version de remplacement conforme au niveau AA est fournie.	Critères de succès pouvant raisonnablement s'appliquer à toutes les ressources Web.
AAA	Pour une conformité de niveau AAA, la page Web remplit tous les critères de succès de niveau A, AA et AAA ou une version de remplacement conforme au niveau AAA est fournie	Critères de succès ne s'appliquant pas à toutes les ressources Web.

Le niveau recommandé par l'Union Européenne est le niveau AA. C'est également le niveau attendu pour les sites concernés par le RGAA et à ce titre, pour être conforme au RGAA, il est nécessaire de valider l'ensemble des tests ayant un niveau WCAG déduit A et AA. Les critères de succès associés au niveau AAA peuvent être pris en compte dans certains contextes, lorsque cela est possible et pertinent.

20.3.3 Origine du document

Le présent document fait suite à l'audit mener par la Société TEMESIS (Expert en accessibilité) sur les sites Extranet de la Région, ainsi qu'à la volonté de la Région de s'inscrire dans une démarche d'accessibilité

La démarche de Sopra Group pour élaborer ce document à été la suivante :

Nous sommes repartis des 187 tests unitaires qui permettent la vérification de la mise en conformité d'une application WEB, nous avons étudié les solutions afin de répondre à ces critères en se focalisant sur le contexte PACA et l'expérience que nous en avons.

20.4 Pré-requis

Connaissance des langages HTML, Javascript et CSS,

Connaissance de la technologie ASP.Net, du langage C#.

20.5 Critères et Solutions - Contenu des directives & solutions

20.5.1 Cadres

Niv	N°	Critère	Solution
A	[CAD]-01	Absence de cadres non titrés.	L'utilisation de cadre (balise frame) est dépréciée en ASP.Net, si cependant il y a un besoin ponctuel alors il faut obligatoirement lui affecter une balise title et une balise alt. http://www.accessiweb.org/fr/guide_accessiweb/guide-accessiweb-glossaire.html#cadre
	[CAD]-02	Pertinence des titres donnés aux cadres.	Le titre du cadre doit être pertinent par rapport au contenu

20.5.2 Couleurs

Niv	N°	Critère	Solution
A	[COU]-01	Présence d'un autre moyen que la couleur pour identifier un contenu auquel il est fait référence textuellement.	Là où l'on identifie les éléments de la page par la couleur ajouter un style en plus. Par exemple, le texte non accessible : En rouge, les œuvres majeures de cet auteur : ** …* Peut être rendu accessible comme cela : En rouge et en gras, les œuvres majeures de cet auteur : ** …*
	[COU]-02	Présence d'un autre moyen que la couleur pour identifier un contenu auquel il est fait référence dans un élément non textuel.	Identique à [COU]-01 mais dans le cadre d'une image ou d'un composant silverlight.

	[COU]-03	Présence d'un moyen de transmission de l'information autre qu'une mise en couleur réalisée par des styles CSS.	Tout élément HTML ayant des styles utilisant au moins l'une des propriétés CSS suivantes : • color, • background-color, • background, • border-color, • border, • outline-color, • outline. Si le style apporte de l'information il n'est pas le seul moyen d'apporter cette information (on peut ici aussi rajouter du gras) Le contenu non accessible : * *<td style="background-color: yellow> </td>* Peut être rendu accessible comme cela : * *<td><[[Fichier:yellow.png|Icône médaille d'or]]</td>* *<td>Médaille d'or</td>* Pour l'item courant du menu mettre un title sur celui-ci (en plus de la couleur pour l'identifier)
	[COU]-04	Présence d'un moyen de transmission de l'information autre que l'utilisation de la couleur dans les éléments non textuels.	Pour les éléments non textuels : • img, • input de type image, • applet, • object, • embed, ou code javascript générant un des éléments précédents, s'ils contiennent de la couleur et que celle-ci est porteuse d'informations alors il faut détailler par un autre moyen ces informations. Sur les zones de couleurs porteuses d'infos mettre des textes écris sur les zones ou un identifiant graphique en plus.

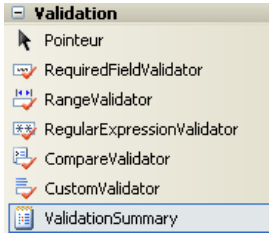
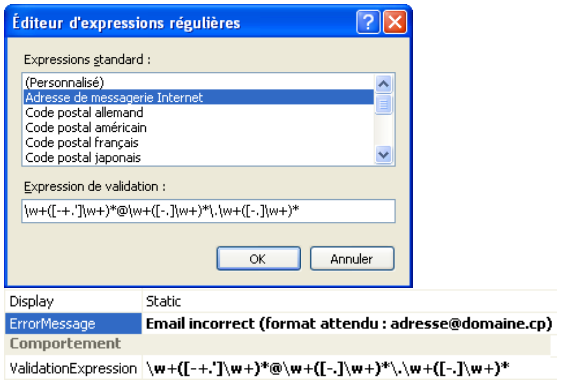
Niv	N°	Critère	Solution
AA	[COU]-05	Valeur du rapport de contraste du texte contenu dans des éléments non textuels. (minimum)	Pour les éléments non textuels : • img, • input de type image, • applet, • object, • embed, ou propriété CSS générant un des éléments précédents, ou code javascript générant un des éléments précédents s'ils contiennent du texte vérifier la taille (la taille finale du texte affichée doit être inférieure à 150%, ou 120% gras, de la taille du texte par défaut spécifiée par les styles de la page, ou, en son absence, de la taille fixée couramment par un navigateur). Pour les couleurs il faut vérifier avec l'outil Color Contrast Analyser (cf § outils de vérification) que le rapport est supérieur ou égal à 4,5.
	[COU]-06	Valeur du rapport de contraste du texte contenu dans des éléments non textuels utilisés comme fond d'éléments HTML. (minimum)	Tout élément HTML ayant des styles associés, utilisant au moins l'une des propriétés CSS suivantes : • background, • background-image, • list, • list-style-image La taille du texte contenu doit être inférieure à 150%, ou 120% gras, de la taille du texte par défaut spécifiée par les styles de la page, ou, en son absence, de la taille fixée couramment par un navigateur. Si la charte est ultérieure à la publication du RGAA (2008) alors vérifier le contraste avec Color Contrast Analyser (rapport supérieur ou égal à 4,5).

	[COU]-07	Valeur du rapport de contraste du texte contenu dans un segment de texte. (minimum)	Tout segment de texte (mot, groupe de mots, phrase, bloc de texte), contenu ou non dans un élément HTML, ou généré via du code javascript ou des feuilles de styles, et mis en couleur par des styles utilisant au moins l'une des propriétés CSS suivantes : • color, • background, • background-color, • background-image, • content, • list, • list-style-image. Appliquer les mêmes tests que ci-dessus.
	[COU]-08	Valeur du rapport de contraste du texte agrandi contenu dans des éléments non textuels. (minimum)	Tout élément : • img, • input de type image, • applet, • object, • embed, ou propriété CSS générant un des éléments précédents, ou code javascript générant un des éléments précédents. La taille finale du texte affichée doit être supérieure ou égale à 150%, ou 120% gras, de la taille du texte par défaut spécifiée par les styles de la page, ou, en son absence, de la taille fixée couramment par un navigateur. Si la charte est ultérieure à la publication du RGAA (2008) alors vérifier que le rapport de contraste entre la couleur du texte agrandi et celle de son arrière plan est supérieur ou égal à 3 (avec Color Contrast Analyser) .
	[COU]-09	Valeur du rapport de contraste du texte agrandi contenu dans des éléments non textuels utilisés comme fond d'éléments HTML. (minimum)	Tout élément HTML non textuel ayant des styles associés, utilisant au moins l'une des propriétés CSS suivantes : • background, • background-image, • list, • list-style-image. Appliquer les mêmes tests que ci-dessus.

	[COU]-10	Valeur du rapport de contraste du texte agrandi contenu dans un segment de texte. (minimum)	Tout segment de texte (mot, groupe de mots, phrase, bloc de texte), contenu ou non dans un élément HTML, ou généré via du code javascript ou des feuilles de styles, et mis en couleur par des styles utilisant au moins l'une des propriétés CSS suivantes : • color, • background, • background-color, • background-image, • content, • list, • list-style-image. Appliquer les mêmes tests que ci-dessus.

Niv	N°	Critère	Solution
AAA	[COU]-11	Valeur du rapport de contraste du texte contenu dans des éléments non textuels (améliorée)	Reprendre le test [COU]-05 mais le rapport de contraste entre la couleur du texte et celle de son arrière plan est supérieur ou égal à 7.
	[COU]-12	Valeur du rapport de contraste du texte contenu dans des éléments non textuels utilisés comme fond d'éléments HTML (améliorée)	Reprendre le test [COU]-06 mais le rapport de contraste entre la couleur du texte et celle de son arrière plan est supérieur ou égal à 7.
	[COU]-13	Valeur du rapport de contraste du texte contenu dans un segment de texte (améliorée)	Reprendre le test [COU]-07 mais le rapport de contraste entre la couleur du texte et celle de son arrière plan est supérieur ou égal à 7.
	[COU]-14	Valeur du rapport de contraste du texte agrandi contenu dans des éléments non textuels (améliorée)	Reprendre le test [COU]-08 mais le rapport de contraste entre la couleur du texte et celle de son arrière plan est supérieur ou égal à 4.5.
	[COU]-15	Valeur du rapport de contraste du texte agrandi contenu dans des éléments non textuels utilisés comme fond d'éléments HTML (améliorée)	Reprendre le test [COU]-09 mais le rapport de contraste entre la couleur du texte et celle de son arrière plan est supérieur ou égal à 4.5.
	[COU]-16	Valeur du rapport de contraste du texte agrandi contenu dans un segment de texte. (améliorée)	Reprendre le test [COU]-10 mais le rapport de contraste entre la couleur du texte et celle de son arrière plan est supérieur ou égal à 4.5.

20.5.3 Formulaires

Niv	N°	Critère	Solution
A	[FOR]-01	Possibilité d'identifier les erreurs de saisie.	La technologie ASP.Net propose par défaut des validators intégrés dans le Framework.  La bonne utilisation de ces composants permet de valider cette règle. ControlToValidate **MonTextBoxAValider** En utilisant les validators cités ci-dessus, avec quelques informations supplémentaires, on obtient le comportement voulu.  Il est nécessaire d'affecter dans la propriété *ErrorMessage* une indication avec le format attendu. Par ailleurs, affecter le *Display* à *Static* permettra d'avoir cette information dès l'entrée dans l'écran (suivant la nécessité).
	[FOR]-02	Présence d'information préalable sur le caractère obligatoire de certains champs de saisie et du type/format de saisie attendue si nécessaire.	L'utilisateur doit être averti du caractère obligatoire de l'élément et si nécessaire du format ou du type de saisie requis par au moins un des mécanismes suivant : • indication en début de formulaire et identifications des champs par un marqueur distinctif situé avant chaque élément de formulaire dans l'ordre du code source (ou après pour input type="checkbox", input type="radio") au sein d'une balise label associé à l'élément, • indication avant chaque élément de formulaire dans l'ordre du code source (ou après pour input type="checkbox", input type="radio") au sein d'une balise label associé à l'élément

	[FOR]-03	Positionnement correct des étiquettes par rapport aux champs dans les formulaires.	Les contrôles HTML qui font partis de ce test sont les suivants : • input type="text", ou le contrôle ASP Textbox • input type="password", ou le contrôle ASP Textbox (avec textmode = password) • input type="checkbox", ou le contrôle ASP Checkbox • input type="file", • input type="radio", ou le contrôle ASP RadioButton • select, ou le contrôle ASP Dropdownlist • textarea, ou le contrôle ASP Textbox (avec textmode = multiline) Pour chacun des composants listés s'il faut lui associer une étiquette alors elle doit se trouver alignée à droite accolée au composant.

	[FOR]-04	Regroupement d'éléments de formulaire via l'élément fieldset.	S'il y a une « cohérence » dans un groupe de contrôles alors utiliser le composant Fieldset, par exemple ici dans l'encart « créer un rapport mensuel » il y a une logique fonctionnelle à associer les contrôles relatifs au mois, à l'année (que ce soit les composants label ou select) :
			 
			On peut créer trois fieldset Créer un rapport / Modifier un rapport / Clôturer un rapport.
			`<fieldset>`
			`<legend>Créer un rapport mensuel</legend>`
			`<asp:label AssociatedControlID="MAListe" Text="Mois" />`
			`<asp:dropdownlist id="MAListe" />`
			`...`
			`</fieldset>`

	[FOR]-05	Absence d'élément fieldset sans élément legend.	Voir ci-dessus la bonne syntaxe pour le fieldset.
	[FOR]-06	Pertinence du contenu de l'élément legend dans l'élément fieldset.	Mettre une balise legend pertinente (qui soit fonctionnellement parlante : exemple ci-dessus à minima)
	[FOR]-07	Regroupement d'éléments option dans un élément select via l'élément optgroup.	**PB TECHNIQUE PEUT ETRE RESOLU PAR UN DEVELOPPEMENT DE CUSTOM CONTROL DROPDOWNLIST AVEC DES OPTGROUP**
	[FOR]-08	Présence d'un attribut label sur l'élément optgroup.	**IDEM FOR 07**

	[FOR]-09	Pertinence du contenu de l'attribut label de l'élément optgroup.	**IDEM FOR 07**
	[FOR]-10	Absence d'élément de formulaire sans identifiant.	Les contrôles HTML qui font partis de ce test sont les suivants : • input type="text", ou le contrôle ASP Textbox • input type="password", ou le contrôle ASP Textbox (avec textmode = password) • input type="checkbox", ou le contrôle ASP Checkbox • input type="file", • input type="radio", ou le contrôle ASP RadioButton • select, ou le contrôle ASP Dropdownlist • textarea, ou le contrôle ASP Textbox (avec textmode = multiline) Plus généralement, ceux sont tous les contrôles ASP.Net qui sont placés dans les formulaires de saisie. Pour chacun des composants listés il faut mettre un identifiant non vide et unique ainsi qu'un attribut title dont la valeur donne la fonction exacte de l'élément.
	[FOR]-11	Absence d'élément de formulaire sans étiquette associée.	Pour les composants des formulaires s'ils possèdent un label associé il faut que celui-ci possède une référence. Mettre un attribut title à l'élément. Par exemple pour un composant « asp:Label » fixer l'attribut « AssociatedControlID » au textbox ou à la dropdownlist associée. Pour un label html c'est dans l'attribut for qu'il faut mettre cette donnée.
	[FOR]-12	Pertinence des étiquettes d'élément de formulaire.	Tous les éléments Label ASP ou HTML doivent être testés. Le contenu ou l'attribut title du label doit donner la fonction exacte de l'élément de formulaire auquel il se rapporte. (description de l'élément pointé dans le cadre du test [FOR]-11)

Niv	N°	Critère	Solution
AA	[FOR]-13	Présence d'informations ou de suggestions facilitant la correction des erreurs de saisie	Sur tous les éléments des formulaires PACA, l'utilisation correcte du panel de validators fournis par le framework et décrit dans le [FOR]-01 permet d'indiquer les éléments corrigeant l'anomalie.
	[FOR]-14	Présence de mécanismes permettant de vérifier, modifier ou confirmer les données à caractère juridique, financier, personnel.	Dans les pages contenant les données à caractère personnel (exemple : ajout d'un utilisateur) présence d'une page récapitulative avant validation et traitement. Demander une confirmation avant un traitement « à risque » (suppression).

Niv	N°	Critère	Solution
AAA	[FOR]-15	Présence de mécanismes permettant de vérifier, modifier ou confirmer tous types de données saisies par l'utilisateur.	Pour chaque formulaire appliquer les mêmes règles que pour [FOR]-14
	[FOR]-16	Présence d'une page d'aide ou d'un mécanisme d'aide contextuelle pour la saisie des formulaires.	Pour chaque contrôles asp:textbox ou input type=text présence d'une aide à la saisie sous l'une des formes suivantes : • présence d'une page d'aide, • présence d'un assistant de saisie • présence d'un correcteur orthographique ou de suggestions lors de la saisie • présence si nécessaire d'informations ou d'exemples sur les formats ou les types de saisie requise. • présence d'indication en début

20.5.4 Images

Niv	N°	Critère	Solution
A	[IMA]-01	Présence de l'attribut alt.	Présence d'un attribut alt sur les composants : • img, • area, • input type='image', • asp:Image fixer l'attribut AlternateText • tout code javascript générant un des éléments précédant. • Par exemple :
	[IMA]-02	Pertinence de l'alternative textuelle aux images liens.	Pertinence du texte contenu dans l'alternative.
	[IMA]-03	Pertinence de l'alternative textuelle aux zones cliquables ou aux boutons graphiques.	Pour chaque image servant de lien le contenu de l'attribut alt doit permettre de comprendre l'action ou d'identifier la destination du lien. • Par exemple :
	[IMA]-04	Pertinence de l'alternative textuelle aux éléments non textuels.	• Pour tout élément : • img, • object • embed • tout code javascript générant un des éléments précédant. Si l'élément apporte visuellement une information, une alternative doit permettre de comprendre l'information que l'on souhaite faire passer par les moyens suivant : • le contenu de l'attribut alt, • le contenu alternatif avant la fermeture de l'élément dans le cas de l'élément object, • le contenu alternatif dans l'élément noembed dans le cas de l'élément embed, • le contenu de l'attribut alt d'une des images d'un groupe d'images formant un tout • le contenu textuel qui précède ou suit immédiatement l'élément ou juste une partie de celle-ci lorsqu'elle est retranscrite en totalité par le contenu servant de description longue associé à l'élément
	[IMA]-05	Pertinence de l'alternative textuelle vide aux éléments décoratifs.	Pour chaque image à but uniquement décoratif mettre un attribut alt vide. • Par exemple :
	[IMA]-06	Longueur du contenu des alternatives textuelles.	Pour chaque élément de type image l'attribut alt doit etre aussi précis et concis que possible.

	[IMA]-07	Existence d'une description longue pour les images le nécessitant.	Pour chaque image qui nécessite une explication plus conséquente mettre une description longue (un lien vers une page de description par exemple)
	[IMA]-08	Pertinence de la description longue pour les images le nécessitant.	Pertinence de l'élément ci-dessus.
	[IMA]-09	Présence de l'attribut longdesc pour établir une relation entre une image et sa description longue.	Dans le cadre d'une image html présence de l'attribut longdesc avec l'url de la description longue. Pour une image asp renseigner le DescriptionUrl. On peut aussi ajouter un lien au contenu immédiatement adjacent à l'élément permettant d'avoir accès à la description longue de l'élément.
	[IMA]-10	Présence d'une information de contexte et d'une solution d'accès pour les captcha lorsque l'alternative ne peut pas être communiquée	Hors contexte

Niv	N°	Critère	Solution
AA	[IMA]-11	Cohérence dans l'identification des alternatives textuelles et des étiquettes de formulaires.	Pour tout contrôle HTML et ASP.Net, si le contrôle apparait plusieurs fois dans une page ou un ensemble de page il garde une cohérence et la description est la même si le rôle de l'élément est identique.

20.5.5 Multimédia

Niv	N°	Critère	Solution
A	[MUL]-01	Accès à une information synthétique pour les contenus sonores, visuel animé ou les médias synchronisés	Hors contexte
	[MUL]-02	Présence de la transcription textuelle des contenus visuels animés, sonores ou des médias synchronisés.	Hors contexte
	[MUL]-03	Pertinence de la transcription textuelle des contenus visuels animés, sonores ou des médias synchronisés.	Hors contexte
	[MUL]-04	Présence d'une description audio synchronisée ou d'une transcription textuelle pour les contenus visuels animés et les médias synchronisés.	Hors contexte
	[MUL]-05	Pertinence de la description audio synchronisée des contenus visuels animés ou des médias synchronisés.	Hors contexte
	[MUL]-06	Possibilité de contrôler l'activation de la description audio synchronisée.	Hors contexte
	[MUL]-09	Présence du sous-titrage synchronisé des médias synchronisés qui ne sont pas diffusés en direct.	Hors contexte
	[MUL]-10	Pertinence du sous-titrage synchronisé des médias synchronisés.	Hors contexte
	[MUL]-11	Présence d'une alternative aux éléments applet et object.	Hors contexte
	[MUL]-12	Présence d'une alternative aux éléments embed.	Hors contexte
	[MUL]-13	Absence d'éléments provoquant des changements brusques de luminosité ou des effets de flash rouge à fréquence élevée.	Hors contexte
	[MUL]-14	Absence de code javascript provoquant des changements brusques de luminosité ou des effets de flash rouge à fréquence élevée.	Hors contexte
	[MUL]-15	Absence de mise en forme provoquant des changements brusques de luminosité ou des effets de flash rouge à fréquence élevée.	Hors contexte
	[MUL]-16	Compatibilité des éléments programmables avec les aides techniques.	Hors contexte
	[MUL]-19	Absence de l'élément blink.	Hors contexte
	[MUL]-20	Absence d'éléments provoquant des clignotements déclenchés automatiquement ne pouvant pas être arrêtés.	Hors contexte
	[MUL]-21	Absence de code javascript provoquant des clignotements déclenchés automatiquement ne pouvant pas être arrêtés.	Hors contexte
	[MUL]-22	Absence de mise en forme provoquant des clignotements déclenchés automatiquement ne pouvant pas être arrêtés.	Hors contexte
	[MUL]-23	Absence d'élément marquee.	Hors contexte
	[MUL]-24	Absence d'éléments affichant des mouvements déclenchés automatiquement ne pouvant pas être arrêtés.	Hors contexte
	[MUL]-25	Absence de code javascript provoquant des mouvements déclenchés automatiquement ne pouvant pas être arrêtés.	Hors contexte
	[MUL]-26	Absence de mise en forme provoquant des mouvements déclenchés automatiquement ne pouvant pas être arrêtés.	Hors contexte
	[MUL]-27	Indépendance du périphérique d'accès aux éléments object, embed, et applet.	Hors contexte
	[MUL]-28	Présence d'une alternative aux éléments object, applet et embed dépendant d'un périphérique.	Hors contexte
	[MUL]-29	Absence d'éléments déclenchant la lecture de son ne pouvant pas être arrêtée.	Hors contexte
	[MUL]-30	Absence d'élément bgsound.	Hors contexte

Niv	N°	Critère	Solution
AA	[MUL]-08	Présence d'une description audio synchronisée pour les contenus visuels animés ou les médias synchronisés.	Hors contexte
	[MUL]-18	Présence du sous-titrage synchronisé des médias synchronisés ou sonores diffusés en direct.	Hors contexte

Niv	N°	Critère	Solution
AAA	[MUL]-07	Présence d'une description audio synchronisée étendue pour les contenus visuels animés ou les médias synchronisés.	Hors contexte
	[MUL]-17	Absence totale de changements brusques de luminosité ou des effets flash rouge à fréquence élevée.	Hors contexte
	[MUL]-31	Présence de version en langue des signes française facilitant la compréhension des médias synchronisés.	Hors contexte
	[MUL]-32	Pertinence de la version en langue des signes française.	Hors contexte
	[MUL]-33	Niveau sonore de la piste de dialogue.	Hors contexte
	[MUL]-34	Présence d'un mécanisme pour personnaliser la couleur d'avant plan et d'arrière plan des blocs de texte.	Hors contexte

20.5.6 Navigation

Niv	N°	Critère	Solution
A	[NAV]-01	Accès aux liens textuels doublant les zones cliquables côté serveur.	Pour chaque image si celle-ci possède des zones cliquables (usemap) et que ces dernières sont doublées par des alternatives textuelles alors ces derniers doivent être immédiatement accessibles derrière l'élément. Exemple : *<img **alt="Liens de navigation du site"** src="menu.gif" width="500" height="212" usemap="#Map">* *<map name="Map">* *<area **alt="Archives"** shape="rect" coords="203,114,258,129" href="archives.aspx">* *<area **alt="Actu"** shape="rect" coords="277,113,348,129" href="actu.aspx/">* *<area **alt="A propos"** shape="rect" coords="364,113,401,128" href="apropos.aspx">* *<area **alt="Admin"** shape="rect" coords="418,114,488,130" href="admin.aspx">* *<area **alt="Google"** shape="rect" coords="-4,190,131,210" href="http://www.google.fr">* *</map>*
	[NAV]-02	Présence d'un avertissement préalable à l'ouverture de nouvelle fenêtre lors de l'utilisation de l'attribut target sur les liens textuels et les formulaires.	Sur chaque lien qui doit ouvrir une nouvelle page alors nécessité de prévenir l'utilisateur de l'une de ces manières : • contenu de son élément html parent si il s'agit d'un élément p ou li, • contenu du titre de hiérarchie (hx) précédent l'élément, • contenu de l'entête (th) qui lui est rattaché si l'élément est dans une cellule de tableau (td), • contenu des éléments de listes parents de l'élément dans une liste arborescente (ul,ol,dl), • contenu de l'attribut title de l'élément si celui-ci est plus grand que le contenu de l'élément seul

	[NAV]-03	Présence d'un avertissement préalable à l'ouverture de nouvelle fenêtre lors de l'utilisation de l'attribut target sur les images liens et les zones cliquables.	Si le lien contient uniquement un élément IMG mettre une alternative à l'élément. Le contenu de l'élément alt seul ou le contenu seul additionné à un contenu récupérable dans au moins un des contextes suivant : • contenu de son élément html parent s'il s'agit d'un élément p ou li, • contenu du titre de hiérarchie (hx) précédent l'élément, • contenu de l'entête (th) qui lui est rattaché si l'élément est dans une cellule de tableau (td), • contenu des éléments de listes parents de l'élément dans une liste arborescente (ul,ol,dl), doit contenir un avertissement signalant l'ouverture dans une nouvelle fenêtre
	[NAV]-04	Présence d'un avertissement préalable à l'ouverture de nouvelle fenêtre lors de l'utilisation de code javascript.	L'élément qui déclenche le code javascript doit contenir une information signalant l'ouverture d'une nouvelle fenêtre. Dans le cas contraire, le contenu de l'élément faisant office d'intitulé, récupéré dans un des contextes suivant : • contenu textuel de l'élément + contenu de son élément html parent s'il s'agit d'un élément p ou li, • contenu textuel de l'élément + contenu du titre de hiérarchie (hx) précédent l'élément, • contenu textuel de l'élément + contenu de l'entête (th) qui lui est rattaché si l'élément est dans une cellule de tableau (td), • contenu textuel de l'élément + contenu des éléments de listes parents de l'élément dans une liste arborescente (ul,ol,dl), • contenu de l'attribut title sur l'élément, • contenu de l'attribut alt pour images liens ou les zones cliquables, doit signaler l'ouverture dans une nouvelle fenêtre.
	[NAV]-05	Absence d'ouverture de nouvelles fenêtres sans action de l'utilisateur.	Pour tout code javascript il ne déclenche pas de redirection sans action de l'utilisateur.

	[NAV]-06	Absence de piège lors de la navigation clavier.	Respecter la bonne structure, la bonne linéarisation des tableaux nous permet d'avoir un ordre logique dans les éléments sans renseigner les tabIndex. Si l'un des composants pouvant avoir le focus n'est pas accessible par la tabulation alors préciser le tabIndex. Dans l'éventualité et suivant le besoin préciser aussi la propriété accesskey mais celle-ci doit être explicitement donnée à l'utilisateur. Exemple : *<table> <tr> <td> Intitulé 1 : Intitulé 2 : </td> </tr> <tr> <td> <input type=text id=champ1 /> <input type=text id=champ2 /> </td> </tr> </table>* N'est pas linéarisé les labels et leurs champs de saisi sont inversés. Choisir plutôt une solution : *<table> <tr> <td> Intitulé 1 : </td> <td> <input type=text id=champ1 /> </td> </tr> <tr> <td> Intitulé 2 :*	

			</td> <td> <input type=text id=champ2 /> </td> </tr> </table>	

	[NAV]-07	Absence d'élément meta provoquant un rafraîchissement automatique de la page.	Pour chaque élément ayant un attribut meta http-equiv='refresh' l'attribut content doit avoir une valeur entière comprise entre 0 et 72000 (cf annexe 2).
	[NAV]-08	Absence de code javascript provoquant un rafraîchissement automatique de la page ne pouvant pas être arrêté.	Si un javascript permet le rechargement de la page que celui-ci fait parti du comportement obligatoire pour le bon fonctionnement de l'application alors l'utilisateur doit avoir le contrôle sur ce rafraichissement par l'un des moyens suivant ou contournement : • possibilité d'arrêter et de reprendre le rafraîchissement, • possibilité d'ajuster librement la durée de rafraîchissement à un minimum de dix fois la durée initialement prévue, • possibilité d'étendre, par une action simple, la durée de rafraîchissement pendant une période d'au minimum vingt secondes au préalable à l'exécution du rafraîchissement, • le délai de rafraîchissement est supérieur à vingt heures,
	[NAV]-09	Absence d'éléments provoquant un rafraîchissement automatique de la page ne pouvant pas être arrêté	Pour chaque élément : • Script serveur • Object • Embed S'il provoque un rafraichissement de la page et qu'il est obligatoire pour le bon fonctionnement de l'application alors il doit être possible : • d'arrêter et de reprendre le rafraîchissement, • d'ajuster librement la durée de rafraîchissement à un minimum de dix fois la durée initialement prévu, • d'étendre, par une action simple, la durée de rafraîchissement pendant une période d'au minimum vingt secondes au préalable à l'exécution du rafraîchissement, • le délai de rafraîchissement doit être supérieur à vingt heures.
	[NAV]-10	Absence d'élément meta provoquant une redirection automatique de la page.	Hors contexte
	[NAV]-11	Absence de code javascript provoquant une redirection automatique de la page ne pouvant pas être arrêtée.	Si un javascript est utilisé pour faire une redirection automatique et qu'il est obligatoire au bon fonctionnement de la page alors l'utilisateur doit avoir la possibilité : • d'arrêter la redirection, • d'ajuster librement la durée préalable à la redirection à un minimum de dix fois la durée initialement prévu, • d'étendre, par une action simple, la durée préalable à la redirection pendant une période d'au minimum vingt secondes au préalable à l'exécution de la redirection, • le délai préalable à la redirection doit être supérieur à vingt heures,

	[NAV]-12	Absence d'éléments provoquant une redirection automatique de la page ne pouvant pas être arrêtée.	Pour chaque élément : • Script serveur • Object • Embed S'il est utilisé pour contrôler une redirection automatique alors l'utilisateur doit avoir la possibilité : • d'arrêter la redirection, • d'ajuster librement la durée préalable à la redirection à un minimum de dix fois la durée initialement prévu, • d'étendre, par une action simple, la durée préalable à la redirection pendant une période d'au minimum vingt secondes au préalable à l'exécution de la redirection, • le délai préalable à la redirection doit être supérieur à vingt heures.
	[NAV]-13	Possibilité d'identifier la destination ou l'action des liens et des boutons.	Pour chaque élément servant à rediriger l'utilisateur : • a, • area, • button, • input type="image", • input type="submit", • input type="button", • input type="reset". La lecture de l'intitulé doit permettre d'identifier le lien de destination. A défaut il doit être possible d'identifier cette cible avec au moins l'un des contenus suivant : • contenu de son élément html parent s'il s'agit d'un élément p ou li, • contenu du titre de hiérarchie (hx) précédent l'élément, • contenu de l'entête (th) qui lui est rattaché si l'élément est dans une cellule de tableau (td), • contenu des éléments de listes parents de l'élément dans une liste arborescente (ul,ol,dl), • contenu de l'attribut title de l'élément si celui si est plus long que l'intitulé du lien lui même

	[NAV]-15	Cohérence de la destination ou de l'action des liens ayant un intitulé identique.	Si deux liens ont un intitulé identique et qu'on retrouve les conditions : • contenu de leur élément html parent s'il s'agit d'un élément p ou li, • contenu du titre de hiérarchie (hx) précédent les éléments, • contenu de l'entête (th) qui leur est rattaché si l'élément est dans une cellule de tableau (td), • contenu des éléments de listes parents des éléments dans une liste arborescente (ul,ol,dl), Alors les éléments doivent avoir un attribut title dont le contenu est spécifique, plus long que l'intitulé du lien et dont la lecture doit permettre d'identifier la cible du lien.
	[NAV]-16	Absence de liens sans intitulé.	Tous les liens non vides doivent avoir un intitulé obtenu soir par contenu textuel soit par alternative textuelles des éléments graphiques contenus dans l'élément.
	[NAV]-24	Navigation au clavier dans un ordre logique par rapport au contenu.	La structure html du document doit permettre une navigation cohérente sur chacun des composants pouvant avoir le focus (voir NAV06)
	[NAV]-26	Présence des informations de format pour les documents en téléchargement.	Si un lien propose un téléchargement de document, le type du document doit se retrouver par l'un des moyens suivants : • contenu de son élément html parent s'il s'agit d'un élément p ou li, • contenu du titre de hiérarchie (hx) précédent l'élément, • contenu de l'entête (th) qui lui est rattaché si l'élément est dans une cellule de tableau (td), • contenu des éléments de listes parents de l'élément dans une liste arborescente (ul,ol,dl) • contenu de l'attribut title de l'élément,
	[NAV]-27	Présence des informations de poids pour les documents en téléchargement.	Si un lien propose un téléchargement de document, le type et le poids du document doit se retrouver par l'un des moyens suivants : • contenu de son élément html parent s'il s'agit d'un élément p ou li, • contenu du titre de hiérarchie (hx) précédent l'élément, • contenu de l'entête (th) qui lui est rattaché si l'élément est dans une cellule de tableau (td), • contenu des éléments de listes parents de l'élément dans une liste arborescente (ul,ol,dl) • contenu de l'attribut title de l'élément,
	[NAV]-28	Présence des informations de langue pour les documents en téléchargement.	Si un lien propose un téléchargement de document, la langue du document doit se retrouver par l'un des moyens suivants : • contenu de son élément html parent s'il s'agit d'un élément p ou li, • contenu du titre de hiérarchie (hx) précédent l'élément, • contenu de l'entête (th) qui lui est rattaché si l'élément est dans une cellule de tableau (td), • contenu des éléments de listes parents de l'élément dans une liste arborescente (ul,ol,dl) • contenu de l'attribut title de l'élément,

	[NAV]-29	Présence de regroupement pour les liens importants.	Si un composant html contient plusieurs liens alors tous les liens font partis d'un même groupe (zone de navigation, zone de contenu, zone d'outils)
	[NAV]-30	Présence d'un balisage permettant d'identifier les groupes de liens importants.	Si un composant html contient plusieurs liens alors tous les liens font partis d'un même groupe (zone de navigation, zone de contenu, zone d'outils). Par ailleurs une ancre doit etre présente sur l'élément implémenté de l'un des manières suivantes : • présence d'un attribut id non vide sur l'élément lui même, • présence d'un élément a, avec un attribut id ou name non vide, étant l'élément frère précédent le plus proche de l'élément, • présence d'un élément a, avec un attribut id ou name non vide, étant le premier enfant de l'élément
	[NAV]-31	Présence de liens d'évitement ou d'accès rapide aux groupes de liens importants.	Mettre un lien d'évitement dans la masterpage Aller au contenu
	[NAV]-32	Cohérence des liens d'évitement ou d'accès rapide aux groupes de liens importants.	Le lien d'évitement doit être fonctionnel et amener à l'endroit indiqué. La navigation au clavier doit se poursuivre à partir de l'endroit pointé.

Niv	N°	Critère	Solution
AA	[NAV]-17	Présence d'une page contenant le plan du site.	Créer une page web s'appuyant sur le fichier site map décrivant le plan du site.
	[NAV]-18	Cohérence du plan du site.	Créer une page web s'appuyant sur le fichier site map décrivant le plan du site avec des liens hypertextes.
	[NAV]-19	Présence d'un accès à la page contenant le plan du site depuis la page d'accueil.	Faire un lien redirigeant vers le plan du site depuis la page d'accueil.
	[NAV]-21	Présence de menus ou de barres de navigation.	Intégrer un menu dans la master page
	[NAV]-22	Cohérence de la position des menus et barres de navigation dans le code source de la structure HTML.	Identique : Intégrer un menu dans la master page
	[NAV]-23	Cohérence de la présentation des menus et barres de navigation.	Le menu intégré dans la master page ne doit pas changer de style.
	[NAV]-33	Ordre des liens d'évitement ou d'accès rapide dans le code source des pages.	S'il y a plusieurs liens d'évitement l'ordre des éléments doit rester le même partout
	[NAV]-34	Présence d'un moteur de recherche.	Hors contexte
	[NAV]-35	Possibilité de naviguer facilement dans un groupe de pages.	Hors contexte

Niv	N°	Critère	Solution
AAA	[NAV]-14	Possibilité d'identifier la destination ou l'action des liens et des boutons. (intitulé seul)	Pour chaque élément servant à rediriger l'utilisateur : • a, • area, • button, • input type="image", • input type="submit", • input type="button", • input type="reset". La lecture de l'intitulé doit permettre d'identifier le lien de destination.
	[NAV]-20	Présence d'un fil d'ariane.	Intégrer un élément fil d'ariane dans la master page
	[NAV]-25	Présence d'un avertissement préalable à l'ouverture de nouvelle fenêtre lors de l'utilisation d'éléments object ou embed.	Si on ouvre une nouvelle fenêtre lors de l'utilisation d'un composant object ou embed alors le contenu de l'élément doit signaler ce comportement
	[NAV]-36	Présence d'une indication de la position courante dans la navigation.	Mettre dans le menu la page courante en gras et en surbrillance

20.5.7 Présentation

Niv	N°	Critère	Solution
A	[PRé]-01	Absence de génération de contenus porteur d'information via les styles CSS.	Pour tout élément qui possède un style css si la propriété content est définie, le contenu ne doit pas être porteur d'informations
	[PRé]-02	Absence d'altération de la compréhension lors de la lecture d'un bloc d'informations lorsque les styles sont désactivés.	La compréhension de la page si on désactive les css reste la même
	[PRé]-03	Lisibilité des informations affichées comme fond d'éléments via les styles CSS lorsque les styles et/ou les images sont désactivés.	Pour chaque composant qui possède une propriété définie en css : • background, • background-image, • list, • list-style-image. Si l'image de fond est porteuse d'information alors il faut que cette information reste accessible si l'on désactive les css et/ou les images.
	[PRé]-04	Absence d'espaces utilisés pour séparer les lettres d'un mot.	Ne pas mettre d'espace ou de composant html simulant un espace entre les lettres d'un mot.
	[PRé]-06	Possibilité de remplacer les éléments non textuels par une mise en forme effectuée grâce aux styles CSS.	Hors contexte car l'élaboration de la charte graphique est antérieure à la parution du RGAA
	[PRé]-08	Absence d'attributs ou d'éléments HTML de présentation.	Ne mettre aucun des attributs suivants : align, alink, background, basefont, bgcolor, border, color, link, text, vlink ni les éléments : basefont, blink, center, font, marquee, s, strike, tt, u,
	[PRé]-09	Absence d'éléments HTML utilisés à des fins de présentation.	Les éléments suivants ne doivent pas être utilisés uniquement à des fins de présentation : a, abbr, acronym, address, area, bdo, blockquote, button, caption, cite, code, dd, dfn, dir, dl, dt, em, fieldset, form, h1 à h6, input, ins, kbd, label, legend, li, menu, ol, pre ou suite d'espace insécables, q, samp, select, strong, sub, sup, th, var, ul
	[PRé]-10	Maintien de la distinction visuelle des liens.	Le contraste des liens mis en forme par du css doit etre testé avec color contrast analyser : Si le ratio de contraste entre la couleur du texte des liens et celle du texte à proximité des liens est supérieur ou égal à 3 et qu'un élément de distinction autre que la couleur est visible lors du focus des liens (graisse, soulignement, icône, etc), le test est validé

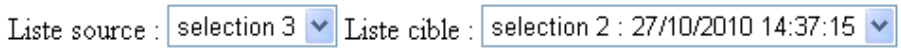
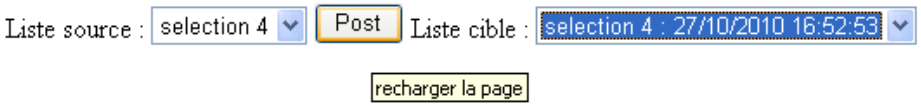
	[PRé]-11	Absence de suppression de l'effet visuel au focus des éléments.	Les propriétés : • outline, • outline-color, • outline-style, • outline-width Ne doivent pas être utilisées pour supprimer l'effet visuel rendu lors du focus de l'élément.
	[PRé]-18	Restitution correcte dans les lecteurs d'écran des éléments masqués.	

Niv	N°	Critère	Solution
AA	[PRé]-05	Absence de définition d'une couleur de texte sans définition d'une couleur de fond et inversement	Si les propriétés css suivantes sont présentes dans les composants texte de la page: • background, • background-color, • font, • color, • list, • list-style-image Si on définit une couleur de fond par css alors il faut définir une couleur de texte et inversement.
	[PRé]-13	Lisibilité du document en cas d'agrandissement de la taille du texte.	L'utilisateur doit avoir la possibilité de lire son document avec une taille de texte augmentée à 200% sans perte de lisibilité.
	[PRé]-14	Absence d'unités absolues ou de pixel dans les feuilles de styles pour la taille de caractère des éléments de formulaire.	Utiliser des tailles de caractères définies en % ou en em notamment sur tous les champs de formulaires, input, select et textarea. (tous composants asp contenant du texte). Par exemple on pourra attribuer à la propriété css font-size la valeur 0.8em ou 80% (le rapport se calcule alors par rapport à l'élément parent)

Niv	N°	Critère	Solution
AAA	[PRé]-07	Possibilité de remplacer les éléments non textuels par une mise en forme effectuée grâce aux styles CSS. (sans exceptions)	Pour chaque élément img object embed, si l'élément n'est pas purement décoratif et qu'il est possible de remplacer l'élément par une mise en forme CSS c'est une nécessité.
	[PRé]-12	Absence de justification du texte.	Aucun bloc de texte ne doit être justifié.
	[PRé]-15	Absence d'apparition de barre de défilement horizontale en affichage plein écran.	En plein écran on peut faire apparaître une barre de défilement cependant un mécanisme qui permet de changer la mise en forme et ne pas voir cette barre doit être présent (par le biais d'un changement de css accessible à l'utilisateur par exemple)
	[PRé]-16	Largeur des blocs de textes.	Tous les blocs de texte doivent avoir une largeur inférieure ou égale à 80 caractères.
	[PRé]-17	Valeur de l'espace entre les lignes et entre les paragraphes	La valeur de l'espacement entre les lignes défini doit être supérieure à 1,5 fois la taille du texte et la valeur de l'espacement entre les paragraphes doit être supérieure à 1,5 fois la taille de l'espacement entre les lignes A défaut un mécanisme permettant d'agrandir l'espacement entre les lignes et entre les paragraphes doit etre présent (exemple : deuxième css).

20.5.8 Scripts

Niv	N°	Critère	Solution
A	[SCR]-01	Mise à jour des alternatives aux éléments non textuels dans la page.	La modification par des scripts de la source (src pour une image) doit aussi modifier l'alternative alt (qui ne correspondrait plus).
	[SCR]-02	Universalité du gestionnaire d'évènement onclick.	L'événement OnClick doit être utilisé seulement sur les balises <a> (si on utilise cet événement sur les span utiliser ARIA)
	[SCR]-03	Universalité des gestionnaires d'évènements.	Pour les événements onMouseOver onMouseout il faut doubler les événements pour une accessibilité au clavier onMouseOut -> on rajoute un onBlur onMouseOver -> on rajoute un onFocus
	[SCR]-05	Absence de changements de contexte suite à une action de l'utilisateur sans validation explicite ou information préalable.	Gestionnaire d'événements onchange / Rechargement automatique d'une page sans action utilisation. Déplacement du focus sans que l'utilisateur soit averti. Au minimum le message d'alerte sinon les faux select. Changement de contexte ou de viewport
	[SCR]-06	Ordre d'accès au clavier aux contenus mis à jour dynamiquement en javascript.	Lorsque l'on génère du contenu il est nécessaire qu'il soit généré à la fin. A défaut il faut replacer le focus sur le bon élément pour que l'utilisateur puisse y avoir accès.
	[SCR]-07	Utilisation correcte du rôle des éléments.	Idem 02
	[SCR]-08	Présence d'une alternative au code javascript utilisant un gestionnaire d'événements sans équivalent universel ou une propriété propre à un périphérique.	Pour les images, lorsqu'on a posé une map dessus on doit avoir la possibilité de faire la même chose que ses onclick mais accessible au clavier (par exemple un bouton à coté)
	[SCR]-09	Absence de suppression du focus clavier à l'aide de code javascript.	Vérifier qu'aucun javascript n'empêche la prise de focus sur un composant.

	[SCR]-12	Présence d'une alternative au code javascript.	Le site web doit être fonctionnel si l'on désactive le javascript. On ne pourra pas utiliser l'ensemble des fonctionnalités des composants natifs. Par exemple la fonctionnalité d'AutopostBack présente dans de nombreux composant permettant d'automatiser le rechargement de la page pour l'utilisateur sera impossible. Pour autant il ne faudra pas pénaliser la majorité des utilisateurs et leur enlever des facilités auxquels ils se sont habitués. Par ailleurs l'utilisation de la balise <noscript> va permettre de ne pas polluer l'affichage de ces derniers. Prenons l'exemple le plus parlant une liste déroulante qui lorsqu'on sélectionne un élément recharge automatiquement une deuxième liste  Pour avoir cette fonctionnalité il faut avoir un navigateur autorisant le javascript. Dans le cas contraire le rechargement de la page ne se fera pas et la fonctionnalité sera perdue. Une solution : laisser le contrôle du rechargement à l'utilisateur par le biais d'un bouton qui n'apparaitra que lorsque le javascript est désactivé.  **<noscript>** **<asp:Button id="toReload" Text=" Post "** **title="recharger la page" runat="server"/>** **</noscript>** En annexe, on présentera la majorité des contrôles qui possèdent des fonctionnalités javascript, la préconisation pour l'utilisation de chacun d'eux.
	[SCR]-13	Accessibilité des contenus dynamiques en javascript.	Le code généré par du javascript doit être accessible.

Niv	N°	Critère	Solution
AAA	[SCR]-04	Possibilité de désactiver toute alerte non sollicitée ou toute mise à jour automatique d'un contenu de la page.	La plupart du temps cela concerne des messages informatifs sinon cela pose un problème technique qui sera détaillé plus bas.
	[SCR]-10	Absence de limite de temps pour compléter une tâche.	Dans un formulaire multi-partie publique (pas infos d'authentification) si l'utilisateur doit saisir plusieurs pages mais que la saisie dépend d'informations en session alors il est nécessaire que l'information soit stockée par un autre moyen que la session au cas où elle expire.
	[SCR]-11	Absence de perte d'informations lors de l'expiration des sessions authentifiées.	Critère non pris en compte car AAA / il doit être possible en début du formulaire d'augmenter la durée de session : la session est généralement globale à tous les utilisateurs mais on peut aussi gérer les temps de session en base de données par exemple.

20.5.9 Standards

Niv	N°	Critère	Solution
A	[STA]-01	Présence de la déclaration d'utilisation d'une DTD.	Mettre un DOCTYPE dans la master page
	[STA]-02	Conformité de la position de la déclaration d'utilisation d'une DTD.	L'instruction DOCTYPE doit être présente avant la balise html.
	[STA]-03	Conformité syntaxique de la déclaration d'utilisation d'une DTD.	L'instruction doit être définie suivant la syntaxe W3C : http://www.w3.org/QA/2002/04/valid-dtd-list.html
	[STA]-04	Validité du code HTML / XHTML au regard de la DTD déclarée.	La DTD déclarée au début de nos documents : http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd doit être validée.
	[STA]-05	Absence de composants obsolètes par rapport à la version des spécifications W3C utilisée.	Aucun composant ne doit être déclaré comme obsolète par rapport aux spécifications du W3C
	[STA]-06	Présence d'un titre dans la page.	Toutes les pages doivent posséder un élément title.
	[STA]-07	Pertinence du titre de la page.	L'élément title dans chacune des pages doit être pertinent (identifier le contenu et la fonction de la page).
	[STA]-08	Présence d'une langue de traitement.	Tout élément ayant un attribut lang (pour nous dans la masterpage) il doit contenir fr-fr.

20.5.10 Structure

Niv	N°	Critère	Solution
A	[STR]-01	Présence d'au moins un titre de hiérarchie de premier niveau (h1).	Mettre un <h1> contenant le titre dans la page
	[STR]-02	Pertinence du contenu des titres de hiérarchie.	Chaque balise : h1 -> h6 doit permettre d'identifier l'élément qu'il précède.
	[STR]-03	Absence d'interruption dans la hiérarchie de titres.	Bien respecter l'ordre des balises hx : le premier titre de hiérarchie précédant l'élément considéré (hx) dans l'ordre du code source doit être de niveau hn, hn-1 ou hn+x (x inférieur ou égal à 4),
	[STR]-04	Présence d'une hiérarchie de titres complète.	Bien structurer sa page avec un ou plusieurs titres de hiérarchie.
	[STR]-05	Absence de simulation visuelle de liste non ordonnée.	Toute liste doit être structurée sous forme de balise ul, li
	[STR]-06	Utilisation systématique de listes ordonnées pour les énumérations.	Toute liste ordonnée doit être structurée avec des balises ol, li
	[STR]-07	Balisage correct des listes de définitions.	Toute liste de définitions (structurée en alinéa de paragraphe) doit être implémentée avec les éléments dl, dt, dd.

	[STR]-08	Balisage correct des citations.	Les citations doivent être dans des balises <q>, si ce sont des block de citations <block-quote>
	[STR]-13	Accessibilité des documents bureautiques en téléchargement.	Pour tous les liens de téléchargement présents dans la page il faut que les documents pointés soient dans une version accessible (notamment concernant les alternatives aux images, les titres de hiérarchie, les listes, les formulaires, les tableaux de données, la langue et l'ordre de lecture) et qu'ils présentent une alternative au format html.

Niv	N°	Critère	Solution
AAA	[STR]-09	Balisage correct des abréviations présentes dans la page.	Les abréviations doivent être balisées avec <abbr> et comporter un attribut title où le mot non abrégé est indiqué.
	[STR]-10	Balisage correct des acronymes présents dans la page.	Les acronymes doivent être contenus dans une balise <acronym > et elle doit posséder un title avec la version complète du mot.
	[STR]-11	Pertinence de la version non abrégée de l'abréviation.	Le title contient la version complète
	[STR]-12	Pertinence de la version complète de l'acronyme.	Le title contient la version complète

20.5.11 Tableaux

Niv	N°	Critère	Solution
A	[TAB]-01	Présence des balises th pour indiquer les en-têtes de lignes et de colonnes dans les tableaux de données.	L'utilisation du nouveau composant asp :gridview permet de générer des balises th Si on devait faire un tableau en code html alors il ne faut pas oublier de mettre une structure en th pour les en-têtes : *<table>* *<tr>* *<th scope= col> En tête 1</th>* *< th scope= col> En tête 2</th>* *</tr>* *<tr>* *<td> valeur 1</td>* *<td> valeur 2</td>* *</tr>* *…* *<table>*
	[TAB]-02	Présence d'une relation entre les en-têtes (th) et les cellules (td) qui s'y rattachent dans un tableau de données simple grâce aux attributs id et headers ou scope.	Ajouter l'attribut scope=col dans les th
	[TAB]-03	Présence d'une relation entre les en-têtes (th) et les cellules (td) qui s'y rattachent dans un tableau de données complexe grâce aux attributs id et headers.	Si on a 2 lignes d'en têtes par tableau alors on doit mettre un id à chacun des th et chaque td doit posséder *th scope= col* un attribut headers contenant la valeur de l'id du th correspondant.
	[TAB]-04	Absence des éléments propres aux tableaux de données dans les tableaux de mise en page.	Les tableaux de mises en page doivent contenir un élément summary vide et ne doivent pas contenir les attributs th, caption, thead, tfoot, colgroup, scope, headers, axis
	[TAB]-05	Absence de tableaux de données ou de colonnes formatés à l'aide de texte	Ne pas simuler un formatage colonne ou tableau de données à l'aide d'espaces textuels.
	[TAB]-06	Linéarisation correcte des tableaux de mise en page.	Bonne linéarisation des tableaux : la lecture linéaire du contenu du tableau (cellule après cellule, dans l'ordre du code source) doit conserver les associations logiques présentes dans son rendu affiché.
	[TAB]-07	Présence d'un titre pour les tableaux de données.	Les tableaux de données doivent posséder un attribut Caption (propriété caption d'un gridview).
	[TAB]-08	Présence d'un résumé pour les tableaux de données.	Ajouter un élément summary dans les tableaux et les gridview (coté code aspx car il n'y a pas de propriété)
	[TAB]-09	Pertinence du titre du tableau de données.	Pertinence du Caption : sa lecture doit permettre de déduire le contenu et la fonction du tableau.
	[TAB]-10	Pertinence du résumé du tableau de données.	Pertinence du summary : sa lecture doit permettre d'expliciter l'organisation des données dans le tableau.

20.5.12 Textes

Niv	N°	Critère	Solution
A	[TEX]-03	Équivalence de l'information mise à disposition dans la version alternative.	L'alternative aux éléments embed, object doit contenir les mêmes informations que l'élément d'origine.
	[TEX]-05	Absence de syntaxes cryptiques par rapport au contenu de votre site.	Hors contexte
	[TEX]-07	Présence d'un moyen de transmission de l'information autre qu'une utilisation de la forme ou la position dans les éléments non textuels.	Hors Contexte
	[TEX]-08	Présence d'un autre moyen que la forme ou la position pour identifier un contenu auquel il est fait référence dans un élément non textuel.	Hors Contexte
	[TEX]-09	Présence d'un autre moyen que la forme ou la position pour identifier un contenu auquel il est fait référence textuellement.	Si un élément texte fait référence à un contenu en indiquant la forme et/ou la position de l'élément référencé alors cette information doit être accessible par un autre moyen.

Niv	N°	Critère	Solution
AA	[TEX]-01	Présence de l'indication des changements de langue dans le texte.	Hors Contexte
	[TEX]-02	Présence de l'indication des changements de langue dans les valeurs d'attributs HTML.	Corriger le calendrier. Pour chaque composant qui possède les attributs alt,summary,content pour les éléments méta,label pour les éléments option,value,name pour les éléments frame et iframe,title,standby pour les éléments object le texte contenu doit être en français.

Niv	N°	Critère	Solution
AAA	[TEX]-04	Présence de liens ou de définitions permettant d'avoir accès aux informations nécessaires à la compréhension des contenus.	Si un élément textuel n'est pas compréhensible par la totalité des utilisateurs de l'application parce qu'il fait parti du langage métier clarifier sa définition par l'un des moyens suivants : • un élément dfn • une liste de définition • la clarification du segment de texte dans le contenu textuel de la page
	[TEX]-06	Présence d'informations sur les mots par la mise à disposition de leur prononciation phonétique	Mettre un lien présentant la phonétique des mots qui peuvent présenter des difficultés à la prononciation
	[TEX]-10	Utilisation d'un style de rédaction simple et compréhensible de tous.	La compréhension des textes ne doit pas nécessiter un niveau d'éducation plus avancée que celui obtenu environ neuf ans après le début de la scolarisation (environ niveau 3ème), après la suppression des noms propres et des titres. A défaut l'utilisateur doit disposer des types de contenu suivants qui ne doivent pas nécessiter de capacité de lecture supérieure au niveau obtenu neuf ans après le début de la scolarisation (environ niveau 3ème) : • illustrations visuelles, symboles ou images facilitant la compréhension des contenus, • version en langue des signes française, • version sonore du segment de texte, • résumé rédigé de manière à ce que sa compréhension ne requière pas de capacité de lecture supérieure au premier cycle de l'enseignement secondaire

Remarques :

Les problèmes techniques non résolus nativement sont traités dans un document annexe à destination du développeur.

20.6 Outils de vérification

20.6.1 Type d'outil : Barres de validation

Ci-dessous une liste des barres de navigations permettant à intégrer dans Firefox ou IE :

- Firefox accessibility extension
  https://addons.mozilla.org/fr/firefox/addon/5809/
- Web developper (IE)
- Wave
  http://wave.webaim.org/toolbar
- Accessibility toobar (paciello group)
  http://www.paciellogroup.com/resources/wat-ie-about.html

20.6.2 Type d'outil : Service en ligne

Ci-dessous les services en ligne pour lesquels on spécifie une adresse de page et on obtient un rapport d'erreurs :

- OCAWA : ne valide qu'une page et pas RGAA -> valide WCAG en version gratuite
  http://www.ocawa.com/fr/Accueil.htm
- TAW3 : ne valide qu'une page mais WCAG 2
  http://www.tawdis.net/ingles.html?lang=en

20.6.3 Type d'outil : Logiciel

Il existe aussi des logiciels de validation qui fonctionnent sur l'ensemble d'un site :

- TAW3 : ne valide pour le moment que WCAG 1, gratuit, valide 10000 pages (fait un rapport complet)
  http://www.tawdis.net/ingles.html?lang=en
- Fujitsu Web Accessibility Inspector
  http://www.fujitsu.com/global/accessibility/assistance/wi/
- TIDY : nettoyage de pages au format W3C
  http://www.w3.org/People/Raggett/tidy/

20.6.4 Type d'outil : Analyseur de contraste

Color Contrast Analyser en version logiciel installé est l'outil préconisé.

http://www.wat-c.org/tools/CCA/1.1/index.html

20.6.5 Type d'outil : Lecteur d'écran

NVDA est gratuit, on peut l'installer sous windows, il est quasiment aussi performent que JAWS. A installer essentiellement pour de la Qualification logicielle.
http://www.nvda.fr/

20.6.6 Type d'outil : tester l'accessibilité en Silverlight / WPF

L'outil UI automation permet de regarder dans la couche accessibilité pour les applications WPF et silverlight.
http://uiautomationverify.codeplex.com/

20.7 La solution ARIA (Accessible Rich Internet Applications)

ARIA décrit comment ajouter de la sémantique et des métadonnées aux contenus HTML afin de rendre les contrôles d'interface et les contenus dynamiques plus accessibles. Par exemple, il devient possible d'identifier une liste de liens en tant que menu de navigation et d'indiquer si son état est plié ou déplié. Quoique conçu pour traiter de l'accessibilité en HTML, l'utilisation de WAI-ARIA n'est pas nécessairement restreinte au HTML mais peut être étendue à d'autres langages comme Scalable Vector Graphics (SVG).

ARIA permet aux pages Web (ou à des parties de pages) de se déclarer comme des applications plutôt que comme de simples documents statiques, par l'ajout de rôles, de propriétés ou d'états d'information vers des applications web dynamiques.

ARIA est destinée à être utilisée par les développeurs d'applications Web, les navigateurs web (ou agents utilisateurs), les technologies d'assistance (ou ATS), et les outils d'évaluation à l'accessibilité.

A vérifier tout de même la compatibilité avec les lecteurs d'écran.

20.8 Annexes

20.8.1 Détails des contrôles utilisant des scripts clients

Composant	Problème rencontré si javascript désactivé	Solution
Button	L'événement "js onClientclick" ne se lancera pas	Ne pas mettre de logique métier dans l'événement
Calendrier custom	Ne marche pas sans javascript. Le calendrier n'apparaitra pas ni l'aide à la saisie.	Autoriser la saisie clavier + implémenter une vérification de format
CheckBox	Autopostback lié à l'action du checkbox ne se déclenchera pas	Ne pas se servir de cette fonctionnalité ou si c'est une obligation mettre un bouton recharger
DropDownList	AutoPostBack + OptGroup	Ne pas se servir de cette fonctionnalité ou si c'est une obligation mettre un bouton recharger/ Pour le optgroup redévelopper un composant ul/li faux select ou un pre render
LinkButton	Autopostback lié à l'action sur le bouton ne se déclenchera pas	Préférer un bouton normal ou un href
ListBox	Autopostback lié à l'action sur la listbox ne se déclenchera pas	Ne pas se servir de cette fonctionnalité ou si c'est une obligation mettre un bouton recharger
TextBox	Peut déclencher un autopostback si utilisation de la fonction Texchanged	Ne pas se servir de cette fonctionnalité ou si c'est une obligation mettre un bouton recharger
GridView	Selected ne marche pas car il fait un postback	Ne pas se servir de cette fonctionnalité ou si c'est une obligation mettre un bouton recharger
Contrôles validateurs	Validation coté client ne se fait pas	Faire une validation coté serveur obligatoirement
Menu	Génère du code non accessible	Chercher un autre composant / accessible si on prend le composant 4.0
TreeView	Génère du code non accessible	Chercher un autre composant - cssfriendly dernière version / accessible si on prend le composant 4.0

20.8.2 Documentation

WAI

http://www.w3.org/WAI/wai-fr

Site regroupant l'ensemble des documents permettant de faciliter l'accessibilité.

WCAG

http://www.w3.org/tr/wcag

Web Content Accessibility Guidelines édités par le WAI.

IBM

http://www-03.ibm.com/able/guidelines/web/accessweb.html

Guides d'accessibilité destines aux développeurs

Accessiweb

http://www.accessiweb.org/fr/Label_Accessibilite/criteres_accessiweb/#liste

Définition de 133 critères AccessiWeb répartis en 3 niveaux (Bronze - Argent - Or)

Wikipedia

http://fr.wikipedia.org/wiki/Wikip%C3%A9dia:Atelier_accessibilit%C3%A9/Bonnes_pratiques

Atelier sur l'accessibilité, des exemples illustrés de bonnes pratiques

MSDN

http://msdn.microsoft.com/fr-fr/library/bb515247.aspx#designing_web_sites_for_accessibility

Description de la prise en charge de certains contrôles en ASP.Net

21 DOCUMENT 12 – 2.00 – GUIDE D'IMPLEMENTATION DE L'ACCESSIBILITE EN FRAMEWORK .NET

21.1 Introduction

21.1.1 Objectif du document

L'objectif de ce document est de proposer aux développeurs des solutions d'implémentations techniques des contrôles serveurs ASP.NET afin de les rendre accessibles au regard des critères d'accessibilité décrit dans le RGAA. Il s'inscrit dans la démarche d'accessibilité des applications Web Intranet/Extranet du Conseil régional PACA.

Par défaut, la plupart des contrôles serveur ASP.NET génèrent un balisage conforme aux directives d'accessibilité. Dans certains cas, il sera nécessaire de configurer un contrôle pour vous assurez que le contrôle génère un balisage accessible. Toutefois, il persiste des contrôles serveur ASP.NET qui ne peuvent pas générer un balisage conforme à certaines directives d'accessibilité (dans ce cas, si l'utilisation de ce contrôle est nécessaire, il faudra justifier son utilisation et le préciser dans une demande de dérogation en vue de l'attestation).

Les sections qui suivent répertorient les contrôles serveur ASP.NET et fournissent des informations sur l'accessibilité pour chacun d'eux. Dans le cas où le contrôle génère un HTML non accessible, il sera nécessaire de consulter le document « Document de référence - Accessibilité en ASP.Net.doc » afin d'analyser la faisabilité de mise en accessibilité dans le contexte Conseil régional PACA.

21.1.2 Documents de référence

Document	Lien
Accessibilité dans Visual Studio et ASP.NET	Accessibilité dans Visual Studio et ASP.NET
Référentiel Général d'Accessibilité pour les Administrations RGAA	http://references.modernisation.gouv.fr/sites/default/files/RGAA_v2.2.1.pdf
Référentiel Général d'Accessibilité pour les Administrations RGAA Annexe 1 : Critères de succès	http://references.modernisation.gouv.fr/sites/default/files/RGAA-v2.2_Annexe1-Criteres.pdf
Référentiel Général d'Accessibilité pour les Administrations RGAA Annexe 2 : Tests de conformité au RGAA	http://references.modernisation.gouv.fr/sites/default/files/RGAA-v2.2_Annexe2-Tests.pdf
Référentiel Général d'Accessibilité pour les Administrations RGAA Annexe 3 : Grilles de correspondance entre les critères de succès et les tests de conformité	http://references.modernisation.gouv.fr/sites/default/files/RGAA-v2.2_Annexe3-Grilles.pdf
Référentiel accessibilité CR PACA	

21.2 Problèmes techniques & solutions

Ce paragraphe décrit les problèmes techniques posés par l'utilisation de la technologie ASP.Net.

21.2.1 Composants liste non accessibles

La liste déroulante (ASP Dropdownlist) n'est pas un composant accessible car il ne prend pas en charge les éléments de groupe (optgroup). Cependant cette fonctionnalité n'est pas une obligation si on parvient à bien classer les éléments de la liste qui reste alors utilisable dans son ensemble. Il faudra pour contourner la règle privilégier un ordre précis alphabétique ou numérique plutôt que métier.

Si malgré cela la solution du tri ne permet pas d'obtenir une cohérence globale des valeurs alors il sera préférable d'utiliser des éléments ul, li et les mettre en forme via du css.

La dernière solution serait de redévelopper un composant dropdownlist pour lequel on customiserait le pre-render afin d'avoir la possibilité de proposer des optgroup.

21.2.2 Composant de navigation : menu

La solution utilisée actuellement est une modification du menu ou treeview proposé par le Framework par le biais d'une librairie CssFriendly permettant de produire, à l'aide d'adapter, du code que l'on peut rendre facilement accessible (le html produit est constitué essentiellement de ul li).

Cependant si des modifications du code javascript généré (gestionnaire d'événement au clavier) et de l'html par la librairie sont nécessaires elles sont facilement implémentables.

21.2.3 Changement de contexte sans validation explicite de l'utilisateur

Ce paragraphe concerne les changements de contexte suite à une action de l'utilisateur sans validation explicite ou information préalable.

Sur l'exemple de la liste déroulante qui lors de la sélection d'un élément charge automatiquement un deuxième composant liste, plusieurs solutions :

- Faux select ul/li décrit plus haut
- Ajout d'un bouton de validation entre les 2
- A minima mettre un titre prévenant du comportement
- Doubler les événements souris et du composant (onchange) par des événements clavier.

21.2.4 Présence d'une alternative au code javascript

La technologie ASP.Net ne permet pas de s'affranchir du code javascript généré. Par conséquent il est nécessaire de demander une dérogation à l'organisme procurant l'attestation d'accessibilité. Dans cette demande doit figurer la justification de l'impossibilité technique ainsi que le coût généré de la prise en compte de ce critère qui multiplierait à minima par 5 les charges de développement.

Il est à noter que chaque site doit avoir une page décrivant le niveau de conformité et l'attestation. La non-conformité due à la présence de javascript peut être atténuée par une bonne complétude de la gestion événementielle clavier.

21.3 Synthèse des contrôles

Ce tableau synthétise la compatibilité aux directives WCAG 2.0 des composants du Framework .NET 4.0.

3 niveaux de conformité sont précisés :

- Conforme nativement (CN) : le contrôle est utilisable sans avoir à apporter un quelconque paramétrage.
- Partiellement conforme (CP) : une vigilance doit être apportée à l'utilisation de ce contrôle.
- Non-conforme (NC) : aucun paramétrage au niveau contrôle serveur n'est possible pour rendre le contrôle accessible.

Composants	CN	CP	NC	Considérations Microsoft sur l'accessibilité	Recommandations
Button	X				
BulletedList		X		La propriété Target est restituée sous forme d'attribut target. (Voir Chapitre 4, Remarque 1.)	• Ne pas utiliser la propriété Target.
Calendar		X		La disposition du contrôle est restituée à l'aide d'un tableau HTML. (Voir Chapitre 4, Remarque 3.) Afin de restituer une légende pour la table, définissez les propriétés Caption et CaptionAlign du contrôle. Dans le cadre des versions d'ASP.NET antérieures à 4, le contrôle restitue un attribut disabled lorsque la propriété IsEnabled a la valeur « false ». (Voir Chapitre 4, Remarque 2.)	• Définissez impérativement les propriétés Caption et CaptionAlign du contrôle. • Définissez la propriété IsEnabled à false uniquement pour les éléments conçus pour l'entrée utilisateur, comme les éléments input et textarea. (Versions d'ASP.NET antérieures à 4)
CheckBox	X				
CheckBoxList		X		Lorsque la propriété RepeatLayout a la valeur Table (valeur par défaut), la disposition du contrôle est restituée en utilisant une table HTML. (Voir Chapitre 4, Remarque 3.) Par défaut, ces contrôles restituent des listes sous la forme d'éléments table. Toutefois, ils peuvent restituer à la place des éléments ul (liste non triée), des éléments ol (liste triée) ou des éléments span, selon la définition de la propriété RepeatLayout. Pour restituer un balisage sémantiquement correct, définissez la propriété RepeatLayout sur UnorderedList ou OrderedList. Les limitations de mise en forme suivantes s'appliquent si vous utilisez une liste triée ou une liste non triée : • L'orientation de la disposition doit être verticale. • Vous ne pouvez pas spécifier plusieurs colonnes. • Les en-têtes, les pieds de page et les séparateurs ne sont pas pris en charge.	• Définissez impérativement la propriété RepeatLayout sur UnorderedList ou OrderedList. • Définissez la mise en forme avec les feuilles de style CSS.
DropDownList			X	Le contrôle ne prend pas en charge l'élément optgroup, qui permet de subdiviser la liste en sections. La solution préconisée dans ce cas est l'implémentation d'un "faux" SELECT qui marche à base d'INPUT texte pour le champ de sélection et des UL/LI pour la liste d'éléments déroulés. Cette solution est pratique mais reste consommatrice de Javascript si vous avez plusieurs listes dans votre page. Dans ce cas on modifiera la propriété du javascript « OnChange ». Si vous ne pouvez pas mettre un « title », c'est une liste qui va vous rediriger sur un autre élément lorsque vous changez l'index. Dans le cadre des versions d'ASP.NET antérieures à 4, le contrôle restitue un attribut disabled lorsque la propriété IsEnabled a la valeur « false ». (Voir Chapitre 4, Remarque 2.)	• Ne pas utiliser le contrôle par défaut. • Implémentez un faux « select » à base d'INPUT texte pour le champ de sélection et des UL/LI pour la liste d'éléments déroulés.

HyperLink		X		Si vous affectez l'URL d'un fichier graphique à la propriété ImageUrl, définissez la propriété Text pour spécifier le texte de remplacement pour le graphique résultant. (Voir Chapitre 4, Remarque 5.) La propriété Target est restituée sous forme d'attribut target. (Voir Chapitre 4, Remarque 1.) Dans le cadre des versions d'ASP.NET antérieures à 4, le contrôle restitue un attribut disabled lorsque la propriété IsEnabled a la valeur « false ». (Voir Chapitre 4, Remarque 2.)	• Définissez impérativement la propriété Text pour spécifier le texte de remplacement. • Ne pas utiliser la propriété Target. • Définissez la propriété IsEnabled à false uniquement pour les éléments conçus pour l'entrée utilisateur, comme les éléments input et textarea. (versions d'ASP.NET antérieures à 4)
Image, ImageButton, ImageMap		X		Définissez la propriété AlternateText ou GenerateEmptyAlternateText afin de fournir une valeur pour l'attribut alt. (Voir Chapitre 4, Remarque 5.) Dans le cadre des versions d'ASP.NET antérieures à 4, le contrôle restitue un attribut disabled lorsque la propriété IsEnabled a la valeur « false ». (Voir Chapitre 4, Remarque 2.)	• Définissez impérativement la propriété AlternateText ou GenerateEmptyAlternateText.
Label		X		Pour utiliser le contrôle afin de restituer un élément label pour une zone de texte ou un autre contrôle conçu pour l'entrée utilisateur, définissez la propriété AssociatedControlID sur l'ID du contrôle à associer à l'étiquette. Dans le cadre des versions d'ASP.NET antérieures à 4, le contrôle restitue un attribut disabled lorsque la propriété IsEnabled a la valeur « false ». (Voir Chapitre 4, Remarque 2.)	• Définissez la propriété AssociatedControlID sur l'ID du contrôle à associer à l'étiquette. • Définissez la propriété IsEnabled à false uniquement pour les éléments conçus pour l'entrée utilisateur, comme les éléments input et textarea. (Versions d'ASP.NET antérieures à 4)
LinkButton, Panel		X		Dans le cadre des versions d'ASP.NET antérieures à 4, le contrôle restitue un attribut disabled lorsque la propriété IsEnabled a la valeur « false ». (Voir Chapitre 4, Remarque 2.)	• Définissez la propriété IsEnabled à false uniquement pour les éléments conçus pour l'entrée utilisateur, comme les éléments input et textarea. (Versions d'ASP.NET antérieures à 4)
ListBox	X				
RadioButton	X				

 RadioButtonList		X		Lorsque la propriété RepeatLayout a la valeur Table (valeur par défaut), la disposition du contrôle est restituée en utilisant une table HTML. (Voir Chapitre 4, Remarque 3.) Par défaut, ces contrôles restituent des listes sous la forme d'éléments table. Toutefois, ils peuvent restituer à la place des éléments ul (liste non triée), des éléments ol (liste triée) ou des éléments span, selon la définition de la propriété RepeatLayout. Pour restituer un balisage sémantiquement correct, définissez la propriété RepeatLayout sur UnorderedList ou OrderedList. Les limitations de mise en forme suivantes s'appliquent si vous utilisez une liste triée ou non triée : • L'orientation de la disposition doit être verticale, • Vous ne pouvez pas spécifier plusieurs colonnes, • Les en-têtes, les pieds de page et les séparateurs ne sont pas pris en charge.	• Définissez impérativement la propriété RepeatLayout sur UnorderedList ou OrderedList. • Définissez la mise en forme avec les feuilles de style CSS.
 Table, TableRow, TableCell, TableHeaderCell, TableHeaderRow TableFooterRow		X		Afin de restituer une légende pour la table, définissez les propriétés Caption et CaptionAlign du contrôle. Lorsque vous créez un objet Table, incluez les contrôles TableHeaderRow et TableHeaderCell.Les valeurs par défaut des contrôles TableHeaderRow et TableFooterRow conduisent le contrôle à restituer des éléments thead, tbody et tfoot. Dans les contrôles TableCell, définissez la propriété AssociatedHeaderCellID pour que le contrôle restitue un header qui associe la cellule à son en-tête. Définissez la propriété Scope du contrôle pour associer l'en-tête à la colonne de données correspondante. Dans le cadre des versions d'ASP.NET antérieures à 4, le contrôle restitue un attribut disabled lorsque la propriété IsEnabled a la valeur « false ». (Voir Chapitre 4, Remarque 2.)	• Définissez impérativement les propriétés Caption et CaptionAlign du contrôle. • Incluez impérativement les contrôles TableHeaderRow et TableHeaderCell. • Définissez impérativement dans les contrôles TableCell, la propriété AssociatedHeaderCellID. • Définissez impérativement la propriété Scope du contrôle pour associer l'en-tête à la colonne de données correspondante. • Définissez la propriété IsEnabled à false uniquement pour les éléments conçus pour l'entrée utilisateur, comme les éléments input et textarea. (Versions d'ASP.NET antérieures à 4)
 TextBox	X				
 DataList, DataListItem			X	La disposition du contrôle est restituée à l'aide d'un tableau HTML. (Voir Chapitre 4, Remarque 3.) Pour créer un balisage qui utilise des feuilles de style en cascade (CSS) plutôt qu'une table pour la disposition, utilisez le contrôle ListView. Dans le cadre des versions d'ASP.NET antérieures à 4, le contrôle restitue un attribut « disabled » lorsque la propriété IsEnabled a la valeur « false ». (Voir Chapitre 4, Remarque 2.)	• Ne pas utiliser ce contrôle. • Utilisez impérativement le contrôle ListView.

 DataPager		X		Si les objets de champ de pagineur sont configurés pour utiliser des images, vous ne pouvez pas spécifier explicitement un texte de remplacement pour les images. Les images utilisent les propriétés Text (telles que NextPageText) comme texte de remplacement. Vous pouvez également utiliser l'objet TemplatePagerField pour pouvoir définir exactement les éléments affichés par le « pagineur ».	• Utilisez l'objet TemplatePagerField pour pouvoir définir exactement les éléments affichés par le « pagineur ». • Utilisez impérativement les propriétés Text (telles que NextPageText) comme texte de remplacement sur les images.
 DetailsView		X		La disposition du contrôle est restituée à l'aide d'un tableau HTML. (Voir Chapitre 4, Remarque 3.) Pour créer un balisage qui utilise des feuilles de style en cascade (CSS) pour la disposition au lieu d'une table, utilisez les propriétés de modèle du contrôle. Si vous créez une colonne ButtonField et que vous spécifiez une image pour le bouton, vous ne pouvez pas spécifier explicitement le texte de remplacement pour l'image. L'image utilise la propriété Text du bouton comme texte de remplacement. Si vous activez la pagination et définissez les propriétés NextPageImageUrl et PreviousPageImageUrl, vous ne pouvez pas définir explicitement le texte de remplacement pour les images qui sont utilisées comme boutons de pagination. Utilisez à la place la propriété PagerTemplate pour pouvoir définir exactement ce qu'affiche le pagineur. Les zones de texte générées automatiquement lorsque vous définissez la propriété AutoGenerateEditButton du contrôle sur « true » n'ont aucun élément label associé. Pour éviter ceci, créez des champs de modèle et ajoutez vos propres contrôles Label et TextBox, puis créez l'association manuellement. Afin de restituer une légende pour la table, définissez les propriétés Caption et CaptionAlign du contrôle. Dans le cadre des versions d'ASP.NET antérieures à 4, le contrôle restitue un attribut « disabled » lorsque la propriété IsEnabled a la valeur « false ». (Voir Chapitre 4, Remarque 2.)	• Utilisez les feuilles de styles CSS pour la mise en forme. • Utilisez la propriété Text du bouton comme texte de remplacement sur l'objet image du contrôle colonne ButtonField. • Utilisez à la place la propriété PagerTemplate pour pouvoir définir exactement ce qu'affiche le pagineur lorsque vous activez la pagination et que vous définissez les propriétés NextPageImageUrl et PreviousPageImageUrl. • Créez impérativement des champs de modèle et ajoutez vos propres contrôles Label et TextBox, puis créez l'association manuellement lorsque vous définissez la propriété AutoGenerateEditButton du contrôle sur « true ». • Définissez impérativement les propriétés Caption et CaptionAlign du contrôle. • Définissez la propriété IsEnabled à false uniquement pour les éléments conçus pour l'entrée utilisateur, comme les éléments input et textarea. (Versions d'ASP.NET antérieures à 4).

FormView		X		La disposition du contrôle est restituée à l'aide d'un tableau HTML. Pour créer un balisage qui utilise des feuilles de style en cascade (CSS) pour la disposition au lieu d'une table, utilisez les propriétés de modèle du contrôle. (Voir Chapitre 4, Remarque 3 et Remarque 4.) Afin de restituer une légende pour la table, définissez les propriétés Caption et CaptionAlign du contrôle. Par défaut, ces contrôles restituent du HTML inclus dans un wrapper dans un élément table dont le but est d'appliquer des styles intralignes au contrôle entier. Vous appliquez des styles intralignes en définissant des propriétés telles que BackColor ou CssClass. Vous n'avez pas besoin d'utiliser des styles intralignes si vous utilisez des modèles pour spécifier comment le code HTML sera restitué pour ces contrôles. Dans ce cas, vous pouvez définir la propriété RenderOuterTable sur « false » pour empêcher la restitution du tableau externe. Dans le cadre des versions d'ASP.NET antérieures à 4, le contrôle restitue un attribut « disabled » lorsque la propriété IsEnabled a la valeur « false ». (Voir Remarque 2.)	• Utilisez les feuilles de styles CSS pour la mise en forme. • Définissez impérativement les propriétés Caption et CaptionAlign du contrôle. • Ne pas utiliser de style intraligne, Définissez la propriété RenderOuterTable sur « false ». • Définissez la propriété IsEnabled à false uniquement pour les éléments conçus pour l'entrée utilisateur, comme les éléments input et textarea. (Versions d'ASP.NET antérieures à 4)
GridView		X		Définissez la propriété TableSection de la propriété HeaderRow sur TableHeader pour que les éléments thead et tbody soient générés. Définissez la propriété RowHeaderColumn sur le nom d'un champ de données et la propriété UseAccessibleHeader sur « true » pour que le contrôle restitue les éléments th et les attributs scope (cela ne fonctionne que pour les champs liés, pas pour les champs de modèle). Si vous créez un objet ButtonField et que vous spécifiez une image pour le bouton, vous ne pouvez pas spécifier explicitement le texte de remplacement pour l'image. L'image utilise la propriété Text de l'objet comme texte de remplacement. Afin de restituer une légende pour la table, définissez les propriétés Caption et CaptionAlign du contrôle. Si vous activez la pagination et définissez les propriétés NextPageImageUrl et PreviousPageImageUrl, vous ne pouvez pas définir explicitement le texte de remplacement pour les images qui sont utilisées comme boutons de pagination. Utilisez à la place la propriété PagerTemplate pour pouvoir définir exactement ce qu'affiche le pagineur. Les zones de texte générées automatiquement lorsque vous définissez la propriété AutoGenerateEditButton du contrôle sur « true » n'ont aucun élément label associé. Pour éviter ceci, créez des colonnes de modèle et ajoutez vos propres contrôles Label et TextBox, puis créez l'association manuellement. Dans le cadre des versions d'ASP.NET antérieures à 4, le contrôle restitue un attribut « disabled » lorsque la propriété IsEnabled a la valeur « false ». (Voir Chapitre 4, Remarque 2.)	• Définissez la propriété TableSection de la propriété HeaderRow sur TableHeader. • Définissez la propriété RowHeaderColumn sur le nom d'un champ de données et la propriété UseAccessibleHeader sur « true ». • Utilisez la propriété Text du bouton comme texte de remplacement sur l'objet image du contrôle colonne ButtonField. • Définissez les propriétés Caption et CaptionAlign du contrôle. • Utilisez à la place la propriété PagerTemplate pour pouvoir définir exactement ce qu'affiche le pagineur lorsque vous activez la pagination et que vous définissez les propriétés NextPageImageUrl et PreviousPageImageUrl. • Créez impérativement des champs de modèle et ajoutez vos propres contrôles Label et TextBox, puis créez l'association manuellement lorsque vous définissez la propriété AutoGenerateEditButton du contrôle sur « true ». • Définissez la propriété IsEnabled à false uniquement pour les éléments conçus pour l'entrée utilisateur, comme les éléments input et textarea. (Versions d'ASP.NET antérieures à 4).

ListView		X		Dans le cadre des versions d'ASP.NET antérieures à 4, le contrôle restitue un attribut « disabled » lorsque la propriété IsEnabled a la valeur « false ». (Voir Chapitre 4, Remarque 2.	• Définissez la propriété IsEnabled à false uniquement pour les éléments conçus pour l'entrée utilisateur, comme les éléments input et textarea. (Versions d'ASP.NET antérieures à 4).
Contrôles validateurs				Attribuez des messages d'erreur significatifs aux propriétés Text et ErrorMessage. Ne leur affectez pas un astérisque (*).	• Attribuez des messages d'erreur significatifs aux propriétés Text et ErrorMessage. Ne leur affectez pas un astérisque (*).
ValidationSummary		X		Dans le cadre des versions d'ASP.NET antérieures à 4, le contrôle restitue un attribut « disabled » lorsque la propriété IsEnabled a la valeur « false ». (Voir Chapitre 4, Remarque 2.)	• Définissez la propriété IsEnabled à false uniquement pour les éléments conçus pour l'entrée utilisateur, comme les éléments input et textarea. (Versions d'ASP.NET antérieures à 4).

Menu		X		Par défaut, ce contrôle génère du code HTML conforme aux directives d'accessibilité qui présente les caractéristiques suivantes : • Le code HTML généré est structuré sous forme de liste non triée (élément ul). • CSS est utilisée pour la mise en forme visuelle. • Le menu se comporte conformément aux normes ARIA pour l'accès au clavier. • Les attributs de propriété et de rôle ARIA sont ajoutés au code HTML généré. Toutefois, le contrôle fournit la propriété RenderingMode pour la compatibilité descendante. Si vous définissez cette propriété sur Table, le contrôle générera du code HTML qui utilise des éléments Table pour la mise en forme, comme c'était le cas dans ASP.NET 3.5 et les versions antérieures Si l'attribut controlRenderingCompatibilityVersion de l'élément pages du fichier Web.config est défini sur 3.5 ou une valeur inférieure, ou si la propriété RenderingMode est définie sur Table, la disposition du contrôle est restituée à l'aide d'une table HTML. (Voir Chapitre 4, Remarque 3.) Le contrôle génère du HTML conforme aux normes ARIA si les conditions suivantes sont remplies : • L'attribut controlRenderingCompatibilityVersion de l'élément pages du fichier Web.config a la valeur 4.0 ou une valeur supérieure. • La propriété Menu..::..RenderingMode est définie sur MenuRenderingMode.Default ou MenuRenderingMode.List. Pour éviter les erreurs de validation de balisage, les attributs ARIA sont définis à l'aide de JavaScript. La propriété SkipLinkText permet que le contrôle entier soit ignoré par les lecteurs d'écran. Si la propriété SkipLinkText est définie, une image invisible s'affiche avec un texte de remplacement afin de permettre à l'utilisateur d'accéder à la fin du contrôle. Les lecteurs d'écran lisent le texte de remplacement à haute voix; l'image occupe uniquement un pixel. Vous pouvez définir la propriété SkipLinkText sur une chaîne vide ("") afin de fournir votre propre mécanisme pour ignorer le menu. La propriété SkipLinkText est définie par défaut sur « Liens Ignorer la navigation ». Dans le cas ou l'attribut SkipLinkText est supprimé ou si vous définissez manuellement l'ancre ctl0_menu_skiplink, rajoutez un lien d'évitement « Aller au contenu ». [NAV]-31 Dans le cadre des versions d'ASP.NET antérieures à 4, le contrôle restitue un attribut « disabled » lorsque la propriété IsEnabled a la valeur « false ». (Voir Chapitre 4, Remarque 2.)	• Ne pas utilisez la propriété RenderingMode avec la valeur Table. • Définissez impérativement l'attribut controlRenderingCompatibilityVersion de l'élément pages du fichier Web.config a la valeur 4.0 ou une valeur supérieure dans le cadre du ARIA. • Définissez impérativement la propriété Menu..::..RenderingMode avec la valeur MenuRenderingMode.Default ou MenuRenderingMode.List dans le cadre du ARIA. • Définissez la propriété IsEnabled à false uniquement pour les éléments conçus pour l'entrée utilisateur, comme les éléments input et textarea. (Versions d'ASP.NET antérieures à 4).

SiteMapPath		X		Pour que le contrôle restitue un lien Ignorer la navigation, définissez la propriété SkipLinkText du contrôle sur une chaîne. Dans le fichier XML de plan de site, définissez l'attribut description de chaque nœud sitemap pour fournir un titre que les lecteurs d'écran peuvent utiliser pour identifier des liens de navigation. Dans le cadre des versions d'ASP.NET antérieures à 4, le contrôle restitue un attribut « disabled » lorsque la propriété IsEnabled a la valeur « false ». (Voir Chapitre 4, Remarque 2.)	• Définissez la propriété SkipLinkText du contrôle sur une chaîne. • Définissez impérativement l'attribut description de chaque nœud sitemap. • Définissez la propriété IsEnabled à false uniquement pour les éléments conçus pour l'entrée utilisateur, comme les éléments input et textarea. (Versions d'ASP.NET antérieures à 4).
TreeView		X		Pour que le contrôle restitue un lien Ignorer la navigation, définissez la propriété SkipLinkText du contrôle. Si la propriété PopulateOnDemand d'un nœud est définie sur « true », le contrôle se comporte d'une façon qui est incompatible avec les directives d'accessibilité. La propriété Target est restituée sous forme d'attribut target. (Voir Chapitre 4, Remarque 1.) Dans le cadre des versions d'ASP.NET antérieures à 4, le contrôle restitue un attribut « disabled » lorsque la propriété IsEnabled a la valeur « false ». (Voir Chapitre 4, Remarque 2.)	• Définissez la propriété SkipLinkTextt • Ne pas définir la propriété PopulateOnDemand d'un nœud à « true ». • Ne pas utiliser la propriété Target. • Définissez la propriété IsEnabled à false uniquement pour les éléments conçus pour l'entrée utilisateur, comme les éléments input et textarea. (Versions d'ASP.NET antérieures à 4).

 ChangePassword		X		La disposition du contrôle est restituée à l'aide d'un tableau HTML. Pour créer un balisage qui utilise des feuilles de style en cascade (CSS) pour la disposition au lieu d'une table, utilisez les propriétés de modèle du contrôle. (Voir Chapitre 4, Remarque 3 et Remarque 4.) Si vous configurez le contrôle pour utiliser des images, affectez le texte de remplacement approprié à la propriété Text correspondante. Par exemple, si vous définissez la propriété ChangePasswordButtonType sur Image, définissez la propriété ChangePasswordButtonText sur la valeur du texte de remplacement. Si le contrôle ChangePassword n'est pas personnalisé avec les modèles, la propriété AccessKey du contrôle ChangePassword s'applique à la première zone de texte dans le contrôle. Si le contrôle ChangePassword est personnalisé avec des modèles, la propriété AccessKey est ignorée. Dans ce cas, définissez directement la propriété AccessKey de chaque contrôle enfant de modèle. La propriété TabIndex est rendue sur tous les contrôles TextBox dans le contrôle ChangePassword. Si le contrôle ChangePassword est personnalisé avec des modèles, la propriété TabIndex est ignorée. Par défaut, ces contrôles restituent du HTML inclus dans un wrapper dans un élément table dont le but est d'appliquer des styles intralignes au contrôle entier. Vous appliquez des styles intralignes en définissant des propriétés telles que BackColor ou CssClass. Vous n'avez pas besoin d'utiliser des styles intralignes si vous utilisez des modèles pour spécifier comment le code HTML sera restitué pour ces contrôles. Dans ce cas, vous pouvez définir la propriété RenderOuterTable sur « false » pour empêcher la restitution du tableau externe. • Dans le cadre des versions d'ASP.NET antérieures à 4, le contrôle restitue un attribut « disabled » lorsque la propriété IsEnabled a la valeur « false ». (Voir Chapitre 4, Remarque 2.)	• Utilisez les feuilles de styles CSS pour la mise en forme. • Définissez la propriété ChangePasswordButtonText sur la valeur du texte de remplacement si la propriété ChangePasswordButtonType est définit sur Image • Définissez directement la propriété AccessKey de chaque contrôle enfant de modèle si le contrôle ChangePassword est personnalisé avec des modèles. • Ne pas utiliser de style intraligne, Définissez la propriété RenderOuterTable sur « false ». • Définissez la propriété IsEnabled à false uniquement pour les éléments conçus pour l'entrée utilisateur, comme les éléments input et textarea. (Versions d'ASP.NET antérieures à 4).
 Login		X		La disposition du contrôle est restituée à l'aide d'un tableau HTML. Pour créer un balisage qui utilise des feuilles de style en cascade (CSS) pour la disposition au lieu d'une table, utilisez les propriétés de modèle du contrôle. (Voir Chapitre 4, Remarque 3 et Remarque 4.) Par défaut, ces contrôles restituent du HTML inclus dans un wrapper dans un élément table dont le but est d'appliquer des styles intralignes au contrôle entier. Vous appliquez des styles intralignes en définissant des propriétés telles que BackColor ou CssClass. Vous n'avez pas besoin d'utiliser des styles intralignes si vous utilisez des modèles pour spécifier comment le code HTML sera restitué pour ces contrôles. Dans ce cas, vous pouvez définir la propriété RenderOuterTable sur « false » pour empêcher la restitution du tableau externe.	• Utilisez des feuilles de style CSS pour la mise en forme. • Ne pas utiliser de style intraligne, • Définissez la propriété RenderOuterTable sur « false ».
 LoginName		X		Définissez la propriété ToolTip sur du texte tel que Connecté comme nom d'utilisateur afin que les lecteurs d'écran interprètent correctement le texte du contrôle. Dans le cadre des versions d'ASP.NET antérieures à 4, le contrôle restitue un attribut « disabled » lorsque la propriété IsEnabled a la valeur « false ». (Voir Chapitre 4, Remarque 2.)	• Définissez la propriété ToolTip sur du texte tel que Connecté comme nom d'utilisateur. • Définissez la propriété IsEnabled à false uniquement pour les éléments conçus pour l'entrée utilisateur, comme les éléments input et textarea. (Versions d'ASP.NET antérieures à 4).
 LoginStatus		X		Dans le cadre des versions d'ASP.NET antérieures à 4, le contrôle restitue un attribut « disabled » lorsque la propriété IsEnabled a la valeur « false ». (Voir Chapitre 4, Remarque 2.)	• Définissez la propriété IsEnabled à false uniquement pour les éléments conçus pour l'entrée utilisateur, comme les éléments input et textarea. (Versions d'ASP.NET antérieures à 4).

 PasswordRecovery		X		La disposition du contrôle est restituée à l'aide d'un tableau HTML. Pour créer un balisage qui utilise des feuilles de style en cascade (CSS) pour la disposition au lieu d'une table, utilisez les propriétés de modèle du contrôle. (Voir Chapitre 4, Remarque 3 et Remarque 4.) Lorsque la propriété SubmitButtonType a la valeur Image, la valeur de la propriété SubmitButtonText est utilisée comme texte de remplacement. Le contrôle ne restitue pas de paramètres de touche d'accès rapide ou d'index de tabulation. Par défaut, ces contrôles restituent du HTML inclus dans un wrapper dans un élément table dont le but est d'appliquer des styles intralignes au contrôle entier. Vous appliquez des styles intralignes en définissant des propriétés telles que BackColor ou CssClass. Vous n'avez pas besoin d'utiliser des styles intralignes si vous utilisez des modèles pour spécifier comment le code HTML sera restitué pour ces contrôles. Dans ce cas, vous pouvez définir la propriété RenderOuterTable sur « false » pour empêcher la restitution du tableau externe. Dans le cadre des versions d'ASP.NET antérieures à 4, le contrôle restitue un attribut « disabled » lorsque la propriété IsEnabled a la valeur « false ». (Voir Chapitre 4, Remarque 2.)	• Utilisez les feuilles de style CSS pour la mise en forme. • Définissez la propriété SubmitButtonText comme texte de remplacement lorsque la propriété SubmitButtonType a la valeur Image. • Ne pas utiliser de style intraligne, Définissez la propriété RenderOuterTable sur « false ». • Définissez la propriété IsEnabled à false uniquement pour les éléments conçus pour l'entrée utilisateur, comme les éléments input et textarea. (Versions d'ASP.NET antérieures à 4).

21.4 Remarques

Remarque 1 : Dans XHTML 1.1 ou HTML 4.0, l'attribut target n'est pas autorisé sur les éléments anchor.

Remarque 2 : Dans HTML 4.0 et XHTML 1.1, l'attribut disabled est autorisé uniquement pour les éléments conçus pour l'entrée utilisateur, comme les éléments input et textarea. Si l'attribut controlRenderingCompatibilityVersion de l'élément pages dans le fichier Web.config, est défini sur 3.5, et si la propriété IsEnabled a la valeur « false », certains contrôlent restituent un attribut « disabled » sur des éléments non conçus pour l'entrée utilisateur. Pour plus d'informations, consultez WebControl..::..SupportsDisabledAttribute.

Remarque 3 : L'utilisation de tables pour disposer les éléments visuels sur une page n'est pas sémantiquement correcte et peut ne pas être conforme à certaines directives d'accessibilité.

Remarque 4 : Même si le contrôle est personnalisé avec des modèles, le balisage rendu est contenu dans une table HTML. Pour empêcher la restitution de cet élément table, définissez la propriété RenderOuterTable sur « false ». Dans ce cas, vous ne pouvez pas définir des propriétés de style comme ForeColor.

Remarque 5 : L'attribut alt des éléments img doit décrire le contenu de l'image sauf si l'image est uniquement décorative. Pour les images décoratives, l'attribut alt doit être défini sur une chaîne vide ("").