ISIMA
Institut Supérieur
d'Informatique, de
Modélisation et de
leurs Applications

BP 10125
63173 Aubière Cedex

Sopra group
Sopra Group

Parc Technologique de la Pardieu
Immeuble Vision II
3 Allée Pierre de Fermat
63170 Aubiere

Rapport d'ingénieur

Stage de 3ème année

*Filière :* Génie Logiciel et Systèmes Informatiques

# Développement d'une application de recrutement

*Présenté par :* Julien PINGUET
*Sous la direction de :* Claude MAZEL

Avril-Septembre 2013

# Remerciements

Je tiens à remercier les personnes qui m'ont accueilli et accompagnés durant ces 6 mois de stage chez Sopra Group.

Je remercie tout d'abord l'équipe du projet Sides, notamment Delphine JARIGE, pour mon accueil en ce début de stage.

J'adresse aussi mes remerciements à l'équipe du projet Limagest, sujet de cette étude : le chef de projet Laurent LAFFERRÈRE pour son accompagnement au niveau fonctionnel, ainsi que l'architecte Xavier CLEMENCE pour son aide apporté lors des difficultés rencontrées.

Enfin, je remercie l'équipe du projet de l'Imprimerie Nationale : Pierre DASCHET le chef du projet, Ambroise ROQUETTE l'architecte, mais aussi les développeurs, pour l'aide apporté lors de mon affectation au projet.

# Table des figures

# Résumé

Mon stage de fin d'études à l'ISIMA avait pour but l'intégration d'une société de services en ingénierie informatique, affecté à un projet de refonte d'une application de recrutement de saisonniers.

Tout d'abord, j'ai étudié les besoins du client, ainsi que l'ancienne solution, permettant de modéliser le modèle de données et la nouvelle application.

Après le choix des technologies, j'ai développé une première version fonctionnelle de l'application. Celle-ci a permis au client d'énumérer les problèmes et proposer des améliorations ergonomiques et fonctionnelles.

Une fois l'application opérationnelle et son déploiement effectué, il ne restait plus qu'à importer les données de l'ancienne base de données pour permettre une transition efficace.

**Mots clé :** Sopra Group, Microsoft, VB.NET, Silverlight, SQL Server

# Abstract

TODO

**Keywords :** Sopra Group, Microsoft, VB.NET, Silverlight, SQL Server

# Table des matières

# Glossaire

**BDD :** Base De Données.

**Client léger :** application fonctionnant dans un navigateur web, ne nécessitant aucune autre installation.

**Framework :** ensemble de composants logiciels permettant le développement le développement rapide d'applications.

# Introduction

Dans de nombreux contextes le gain de temps est le maître mot. Le domaine professionnel ne fait pas exception à la règle, imposant une productivité de plus en plus élevée. Ainsi l'utilisation d'outils de plus en plus performants permet de minimiser le temps passé à effectuer une tache.

Limagrain utilise un outil un outil de recrutement, développé il y a plus de dix ans par un membre de service ne possédant de compétence en développement. Dû au nombre de recrutements de plus en plus élevé, et l'augmentation de la charge de travail, Limagrain désire faire évoluer son outil pour faciliter le processus actuel.

Le passage des années, ainsi que l'évolution des besoins, impose une mise à niveau permanente des outils informatiques. Mais lorsque celui-ci utilise des technologies trop anciennes, et ne possède pas une architecture évolutive, il est rarement possible d'intervenir. La meilleure solution est donc une ré-implémentation complète de l'outil, amenant alors à une solution structurée qui pourra évoluer facilement aux cours des années.

L'objet de cette étude sera d'étudier les besoins du client, puis d'implémenter la nouvelle solution. Celle-ci devra remplir les mêmes fonctions que la solution actuelle, en permettant l'ajout de nouvelles.

L'étude de la solution actuelle et l'étude des besoins de l'utilisateur permettra de modéliser la nouvelle solution. L'implémentation d'une première version de l'application permettra de soulever les problèmes existants et d'émettre les points d'amélioration. Enfin la mise en production et l'importation des anciennes données a permis au client d'utiliser l'application.

Tout d'abord, j'analyserai précisément le sujet, en présentation l'entreprise, ainsi que la solution actuelle et celle envisagée. Dans un deuxième temps, je présenterai l'environne-

ment de travail et la structure du projet. Pour terminer, je présenterai la solution obtenue et les différents points d'amélioration possibles.

# I Introduction de l'étude

## I.1 Présentation de l'entreprise

### I.1.1 Historique

*Sopra Group* est une Sociétés de Service en Ingénierie Informatique, couramment appelé SSII. Créée en 1968 par Pierre PASQUIER et François ODIN, il s'agit d'un des plus anciennes et importante SSII. La société s'est d'abord révélée au niveau national grâce à de grands projets dans les domaines bancaires et avec le Ministère de l'Intérieur, avant de s'implanter au niveau européen et mondial. Avec plus de 14.300 collaborateurs en 2012, Sopra Group réalise un chiffre d'affaire supérieur à 1,2 milliards d'euros, principalement dans le domaine du conseil et des services en France.

### I.1.2 Agence de Clermont-Ferrand

Michelin est le plus gros employeur de la région Auvergne, mais fait aussi appel à de nombreuses sociétés extérieures comme les sociétés de service. Dû au nombre croissant de collaborateurs travaillant chez Michelin, la division Rhône-Alpes a créé l'agence de Clermont-Ferrand pour être au plus proche du client.

L'agence de Clermont-Ferrand travaille aussi sur plusieurs autres projets importants. Le pôle identité qui est spécialisé dans la gestion de titres (documents d'identité, passeport, cartes à puce, ...) grâce aux projets Sides et plus récemment SITI, travaillant pour une quinzaine de gouvernements. Le groupe Limagrain, dont le siège social est situé Chappes, fait actuellement appelle aux services de Sopra Group pour la réalisation du projet Boss.

## I.2 Étude du problème

### I.2.1 Besoin de l'utilisateur

Durant l'été, saison des récoltes des céréales, Limagrain emploie de nombreux travailleurs. Il s'agit principalement de contrats saisonniers ou à durée déterminée.

Le service de recrutement du groupe Limagrain a besoin d'enregistrer les informations de chacun de ses employés pour ne pas avoir à les ressaisir lors des prochains contrats. En effet, le service emploie de préférence d'anciens candidats. De plus, les contrats sont parfois renouvelés plusieurs fois durant la saison ou chaque année.

L'Urssaf, unions de recouvrement des cotisations de sécurité sociale et d'allocations familiales, impose la saisie d'un document : la DAPE, déclaration préalable à l'embauche. Pour chaque recrutement il est donc nécessaire de remplir ce document avec les informations du candidat ainsi que du contrat. Le client désire donc pouvoir générer automatiquement ces documents, lui évitant ainsi de les saisir manuellement un par un pour chaque contrat.

Pour faciliter le recrutement d'anciens candidats, ou la recherche de candidats disponibles, le client désire avoir différentes fonctionnalités de recherche. Il peut s'agir par exemple de la recherche de candidats disponibles à certaines périodes, ou les candidats ayant un contrat en cours dont il n'a pas fourni toutes les pièces justificatives.

Enfin pour pouvoir augmenter leur productivité, il sera nécessaire de produire une application ergonomique et facile d'utilisation permettant la saisie de nombreuses données le plus rapidement possible. Limagrain reçoit en effet plus de 1.000 candidatures en moins d'un mois dont une grande partie effectue des contrats.

### I.2.2 Solution actuelle

Le service de recrutement utilisait une petite application Access développée à partir d'un document Excel, de Microsoft Office. Ce document comporte une base de données ainsi qu'une interface de saisie.

Cette solution comporte de nombreux inconvénients.

Tout d'abord, la saisie des données est fastidieuse car l'ergonomie est très sommaire : les champs sont disposés de manière désordonnée sur la page. De plus, pour la saisie

d'un établissement par exemple, il était nécessaire de saisir son code, sans pouvoir voir directement son nom ou toute autre information.

Les fonctionnalités sont limitées, et se limitent principalement à la saisie. Les attentes de l'utilisateur sont donc très loin de ce que propose l'outil actuel.

Enfin, il est impossible de travailler en simultané avec cet outil car les sauvegardes multiples ne sont pas prises en compte.

### I.2.3 Solution envisagée

En étudiant la solution actuelle, les chefs de projet et architectes ont décidé de développer une nouvelle solution, en raison de l'impossibilité de maintenir et faire évoluer l'ancienne. Cette nouvelle solution devra répondre aux besoins de l'utilisateur, en corrigeant les inconvénients de l'ancienne ainsi qu'en apportant de nouvelles fonctionnalités.

Il est tout débord nécessaire de changer de base de données pour se tourner vers un système plus performant et centralisé. Un nouveau schéma de données adapté aux nouveaux besoins permettra de mieux structurer ces données.

Pour améliorer l'ergonomie de l'application, la nouvelle solution sera un *client léger*. Dans ce projet nous ne rencontrerons pas les inconvénients qu'offre cette solution, comme les performances ou les problèmes qui affecteront tous les utilisateurs. Par contre, un simple site web offre les avantages d'un déploiement unique, d'une maintenance simple, car cela ne se fait que sur le serveur et non sur chacun des ordinateurs des utilisateurs.

# II Conception de la solution

## II.1 L'équipe du projet

Pour réaliser ce projet, j'ai été intégré dans une petite équipe de trois personnes.

Un chef de projet qui est le relai entre Sopra Group et Limagrain afin d'établir la compréhension du besoin fonctionnel et technique. Sa charge est d'une demi-journée par semaine. De plus, il est chargé de la rédaction des spécifications de l'application. Pour des raisons de délais trop courts et charges limitées, ces spécifications ont été réalisées en parallèle du développement, malgré l'éloignement des bonnes pratiques.

Un architecte a pour objectif de suivre le développeur, conseillant les outils, technologies et méthodes utilisées. Sa charge sur le projet est aussi d'une demi-journée par semaine.

Et enfin moi-même, en tant que développeur, en charge du développement de l'application, à plein temps.

## II.2 La relation client

### II.2.1 Assistance technique

Le projet s'est réalisé en mode *assistance technique*. Il n'y a aucune date de livré imposée, par contre la charge (nombre de jours d'interventions) totale est fixe.

J'ai travaillé à raison de trois jours chez le client Limagrain et deux jours au siège de Sopra Group, profitant ainsi de la proximité du client pour éclaircir les points ambigus.

### II.2.2 Définition du besoin

La relation directe avec le client m'a permis de définir le besoin du client au cours des différentes étapes du projet.

La première étape a consisté à établir le modèle de données. Pour cela il a fallu étudier la solution existante pour déterminer les informations à mémoriser, leur type, les contraintes, ainsi que les liens entre elles. Cette partie est importante car la structure de l'application est fortement liée a ce modèle.

Deuxièmement, j'ai réalisé une première version de l'interface graphique de l'application. L'objectif est de proposer une première maquette au client qui pourra effectuer des critiques. L'ergonomie de la solution s'est ainsi optimisée au fur et à mesure des démonstrations.

Enfin, une fois l'application fonctionnelle, il était nécessaire d'importer les données existantes dans la nouvelle de données. Comme ces deux bases de données sont différentes, il était nécessaire d'être en contact avec le client pour faire correspondre les schémas.

## II.3 Technologies et architecture

Dans cette section je vous présenterai les différentes technologies utilisées dans le projet ainsi que leurs interactions. La figure II.1 schématise les différents composants et communications.

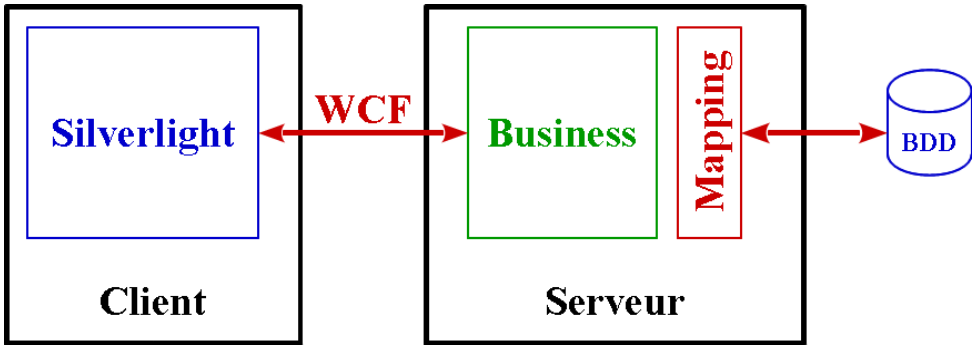


FIGURE II.1 – Architecture du projet

Le groupe Limagrain travaille dans un environnement Micosoft et utilise ainsi leurs technologies et logiciels, comme par exemple : Windows pour le système d'exploitation, Internet Explorer comme navigateur internet, Outlook comme messagerie, ...

Pour développer cette solution nous nous sommes donc tournés vers les technologies Microsoft, facilitant ainsi la compatibilité et l'intégration des composants.

L'architecture d'une application est importante, car cela définie sa maintenabilité et son l'évolutivité. De plus cela permet la séparation des problèmes diminuant ainsi la complexité.

Notre application est divisée en 3 couches, appelé *3-tiers*, qui est le modèle multi-tiers le plus utilisé. Cela permet de séparer l'accès aux données de la base, la partie métier effectuant les traitements, et l'interface de l'utilisateur, comme le représente la figure II.2.

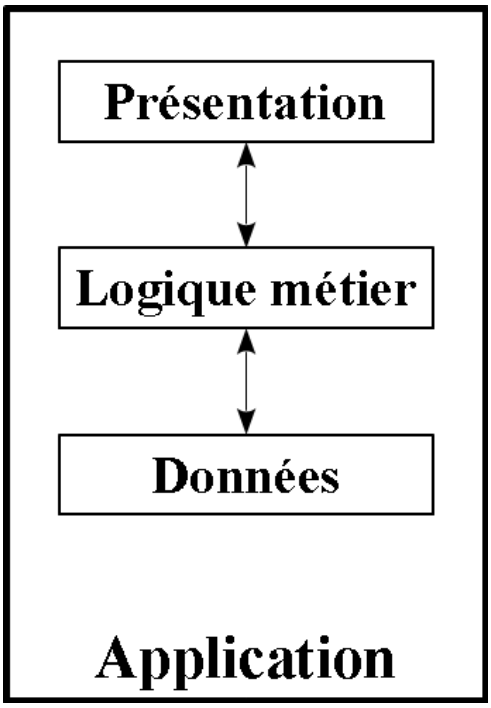


FIGURE II.2 – Architecture 3 tiers

### II.3.1 Socle

La solution développée se base sur un socle existant, développé dans le cadre d'un autre projet. Ceci nous a permis un gain de temps important car une partie importante du projet n'a dû être développé à nouveau. En contrepartie, les différentes technologies utilisées nous ont été imposées.

#### II.3.1.1 Gestion des utilisateurs

Le socle permet une gestion des utilisateurs de l'application. Il est possible de les créer, modifier ou supprimer, pour restreindre l'accès aux personnes autorisées. De plus, il est possible de connecter l'application à un LDAP.

Le *LDAP*, pour Lightweight Directory Access Protocol, est un protocole standard de gestion d'utilisateurs. Son objectif est de centraliser les informations des utilisateurs (nom, identifiant, mot de passe, ...) dans un annuaire.

Le principal avantage de cette solution est la mise en commun des comptes utilisateurs, ce qui permet l'utilisateur d'un seul et même compte pour tous les services connectés : Windows, la boite mail Outlook, cette application, ....

#### II.3.1.2 Gestion des droits

Il est aussi possible de gérer des droits dans l'application grâce à ce socle. L'administrateur peut ainsi contrôler les accès et les actions des utilisateurs, protégeant ainsi les informations confidentielles.

Chaque écran de l'application peut avoir leur accès ou certaines actions restreintes en fonction d'un *droit*. Ceux-ci peuvent prendre trois valeurs :

- "aucun accès", par défaut, qui interdit tout accès à l'écran ;
- "lecture" n'autorise que l'accès à l'écran, avec la possibilité d'effectuer des recherches ou d'afficher les détails ;
- "lecture et écriture" autorise toutes les actions possibles.

Un *profil* est un ensemble de droits, qui permet de définir un périmètre d'action. Par exemple un profil "administrateur" aura tous les droits dans l'application, un profil "manager" n'aura que des droits de lecture, ou encore un profil "ressource humaine" possèdera les droits liés aux candidats et contrats, ...

Chaque utilisateur possède un ou plusieurs profils, ce qui permet de définir l'ensemble de ses droits. L'utilisation de profils plutôt que de directement de droit permet un gain de temps, car la personne administrant les utilisateurs n'aura pas à affecter les nombreux droits aux nouveaux utilisateurs, et la modification d'un profil permet d'impacter l'ensemble des utilisateurs associés.

La figure II.3 représente le schéma UML des utilisateurs, profils et droits dans l'application.

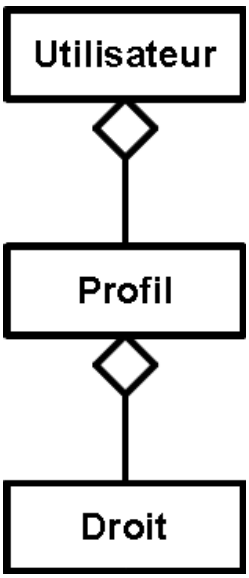


FIGURE II.3 – Utilisation des profils et droits des utilisateurs

#### II.3.1.3 Programmation modulaire

Un module est un projet indépendant effectuant une tâche particulière. Le socle offre la possibilité de programmer de façon modulaire, où chacun des modules ne possèdent (quasiment) aucunes interactions entre elles.

Ainsi pour programmer de manière structurée, plusieurs modules distincts ont été développés. Ceci permet de séparer l'implémentation de tâches distinctes.

### II.3.2 Langage de programmation

#### II.3.2.1 Microsoft .NET Framework

*.NET Framework* est une plateforme application proposée par Microsoft. Cette technologie est comparable et directement concurrente de Java d'Oracle.

Ce framework est constitués de nombreux composants, disposés en couches au fur et à mesure de ses versions, schématisés sur la figure II.4 :

1. Le moteur d'exécution appelé Common Language Runtime (CLR) permet de compiler le code course en un langage intermédiaire appelé Microsoft Intermediate Language (MSIL). Ce code est ensuite compilé à la volée lors de la première exécution grâce au compilateur "Just In Time" (JIT) ;
2. Une bibliothèque de classes, offrant des fonctionnalités de base pour les différentes applications ;
3. Plusieurs couches supplémentaires proposant des outils de développement d'interfaces graphique (WinForms), d'accès aux données (ADO.NET (Entity Framework)), ...

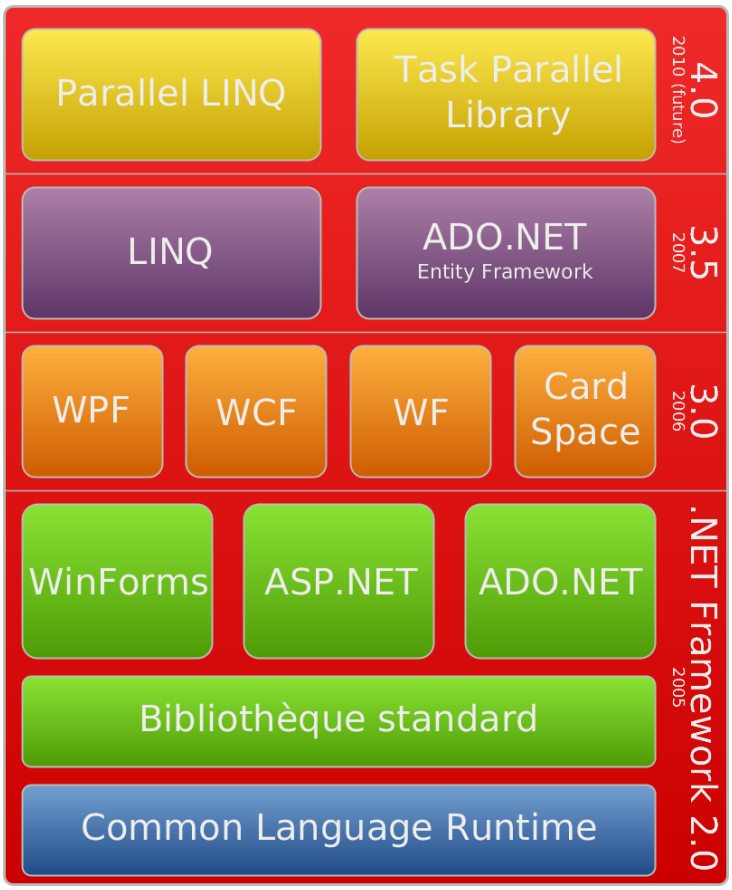


FIGURE II.4 – Couches de .NET Framework
Source : http://fr.wikipedia.org/wiki/Framework_.NET

#### II.3.2.2 VB.NET

*Visual Basic .NET* est un langage de programmation développé par Microsoft. Il s'agit d'une évolution majeure de Visual Basic 6, introduisant notamment l'aspect orienté

objet. De plus, le code est compilé dans le langage intermédiaire que les différents langages fonctionnant sur la machine virtuelle .Net. Ce langage est très proche du C#, à la syntaxe près.

Le projet a été développé dans ce langage, aussi bien la couche métier que dans la couche présentation.

#### II.3.2.3 Apache log4net

Les *log* sont les informations mémorisées lors du fonctionnement d'un programme informatique. Ils permettent de garder des traces d'exécution, notamment lors du démarrage, de l'arrêt, ou d'erreurs. Les log peuvent faciliter le débogage et la recherche d'anomalies. Un ou plusieurs fichiers texte sont générés donnant accès à ces informations.

Le framework *log4net*[1] est un port sous Microsoft .NET du projet libre et open-source "Apache log4j". Il permet de gérer simplement les log dans l'application.

### II.3.3 Base de données

Une *base de données* permet de stocker un grand nombre d'informations ayant des natures différentes. Les données de même nature sont stockées dans des tables, où un champ possède un type et représente une valeur.

#### II.3.3.1 PowerAMC

*PowerAMC*, version francophone de PowerDesigner, est un logiciel de modélisation de base de données produit par la société Sybase. Il permet d'établir facilement un modèle de données à partir de son interface graphique, visible sur la figure II.5. De plus, les scripts générés sont compatibles avec les différents types et versions de bases de données, ce qui permet de s'abstraire de certaines contraintes de compatibilité.

Ce logiciel est utilisé par Sopra Group sur les différents projets. Je l'ai donc utilisé sur ce projet. Il m'a permis de concevoir, mais aussi effectuer les différentes modifications très simplement.

#### II.3.3.2 SQL Server

Il existe de nombreux système de gestion de base de données : Oracle, MySQL, PostgreSQL, ...Nous avons utilisé *Microsoft SQL Server*, par demande du client car il s'agit d'un standard pour Limagrain.

1. log4net - Site web : http ://logging.apache.org/log4net/

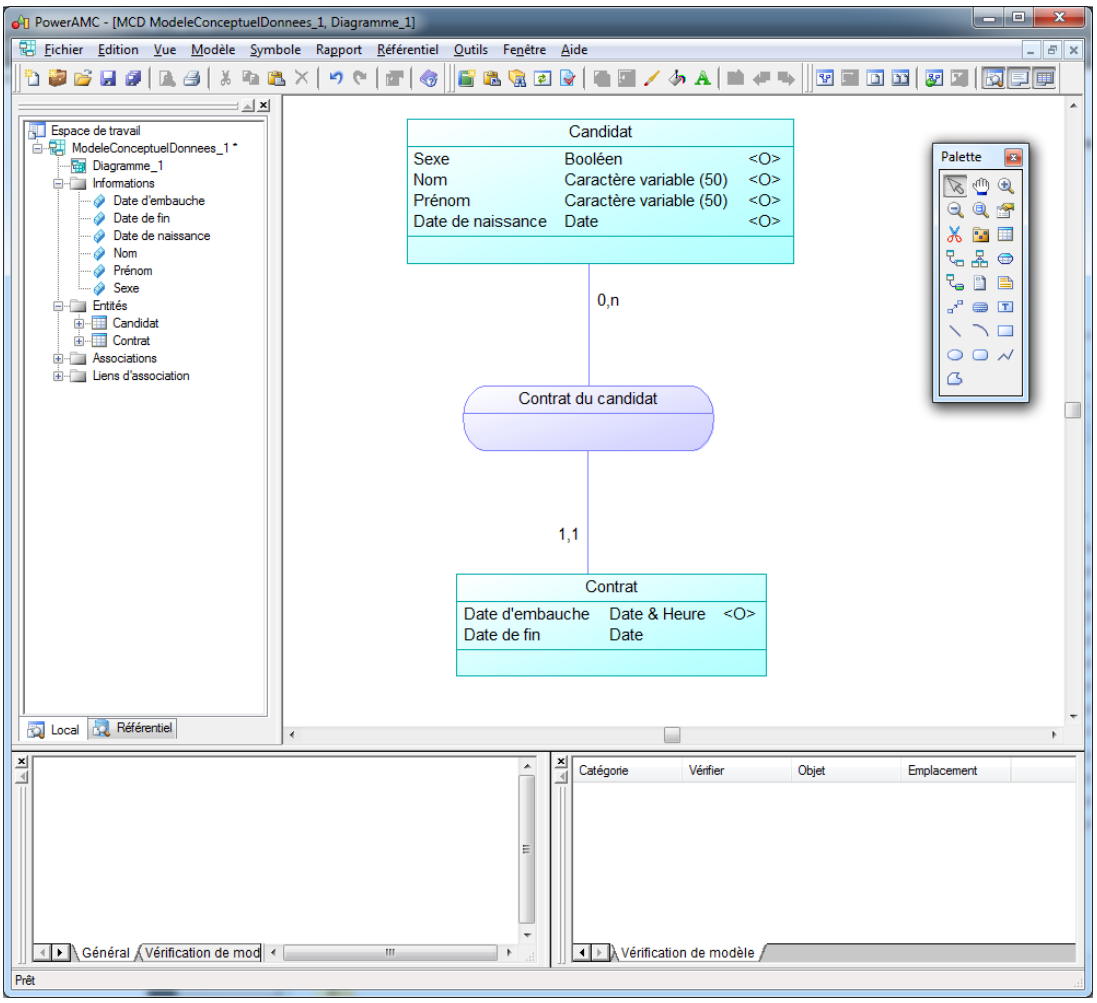


FIGURE II.5 – Interface de PowerAMC

### II.3.3.3 Mapping de la base de données

**Définition**

Le "modèle relationnel" est utilisé dans les systèmes de gestion de base de données (SGBD) pour rassembler un ensemble d'informations. Les données (clés) sont dupliquées entre les tables et l'accès aux relations s'effectue ensuite grâce à des jointures entre les tables.

Le "modèle objet", quant à lui, est utilisé dans la programmation orientée objet. Les données sont modélisées sous la forme d'objets, entités complexes ayant des comportements et des relations entre elles.

Le *mapping objet-relationnel* consiste à interfacer le modèle relationnel d'une base de

données avec le modèle orienté objet d'un programme informatique. Généralement, une classe modélisera une table, et attribut d'objet modélisera un champ d'une table, avec un type similaire (par exemple `String` pour `varchar`).

**Entity Framework**

Cette opération peut être faite à l'aide d'un framework, permettent de s'abstraire de la base de données, automatisant et réduisant ainsi la duplication de code. L'objectif est de faciliter le développement, augmenter la maintenabilité du programme, ou encore s'abstenir du type de base de données. Ainsi, la réaction, la recherche, la mise à jour, ou la suppression (opérations CRUD) de données se font de manière simplifiées voir transparentes pour l'utilisateur.

Microsoft propose plusieurs framework, et c'est *Entity Framework* qui a été utilisé. Il est intégré à Visual Studio, ce qui permet une génération et un paramétrage facile d'utilisation. De plus, il permet l'interaction avec LINQ (Language-Integrated Query), extension du langage permettant de faire des requêtes sur des ensembles de données en s'abstrayant du type.

## II.3.4 Les services

La couche de *service* est la partie métier de l'application. Elle permet de séparer le code source lié à l'affichage destiné à l'utilisateur, du code métier effectuant des traitements dans l'application.

### II.3.4.1 Client-Serveur

**Nécessité**

Il est nécessaire de faire communiquer la couche cliente de l'application avec la couche métier en raison de l'utilisation de Silverlight, comme je l'expliquerai dans la section II.3.5.1 ci-après. Ces deux couches ne peuvent communiquer directement entre elles car l'une est exécutée sur le serveur et l'autre sur l'ordinateur de l'utilisateur. La solution consiste à utiliser un service web.

**Web service**

Un *service web* (ou *web service*) est un programme informatique permettant la communication et l'échange d'informations entre des systèmes hétérogènes et distribués (local, réseau, internet, ...), exposant des fonctionnalités.

L'échange d'informations entre le client et le serveur se fait par sérialisation, consistant à coder les informations contenues en mémoire. Cela peut se faire sous le format texte (XML, JSON, ...) ou même au format binaire. L'objectif est d'échanger des données génériques et abstraites qui pourront être représentées dans n'importe quel langage.

L'exemple qui suit montre un exemple de sérialisation d'un même objet sous le format XML :

```
<menu id="file" value="File">
    <popup>
        <menuitem value="New" onclick="CreateNewDoc()" />
        <menuitem value="Open" onclick="OpenDoc()" />
        <menuitem value="Close" onclick="CloseDoc()" />
    </popup>
</menu>
```

et JSON :

```
{
    "menu": {
        "id": "file",
        "value": "File",
        "popup": {
            "menuitem": [
                { "value": "New", "onclick": "CreateNewDoc()" },
                { "value": "Open", "onclick": "OpenDoc()" },
                { "value": "Close", "onclick": "CloseDoc()" }
            ]
        }
    }
}
```

**Windows Communication Foundation**

*Windows Communication Foundation*, couramment appelé sous ses initiales WCF, est la couche de communication de .NET Framework, apparue dans la version 3.0. Cette technologie respecte les normes standards des services web, ce qui lui permet d'appeler ou d'être appelé par des technologies différentes (Java, Python, ...).

Ce framework a été utilisé pour réaliser le service web de l'application, exposant ainsi différentes fonctions utiles à la couche d'affichage.

### II.3.4.2 WCF RIA Services

Il est souvent nécessaire de posséder une logique applicative à la fois du coté serveur et du coté client. C'est le cas par exemple lorsque l'on souhaite vérifier la validité des données avant de les insérer en base de données :

1. Soit on effectue la vérification coté serveur, impliquant l'échange inutile d'informations lorsque celles-ci sont fausses ;
2. Soit la vérification est faite coté client, ce qui empêcherait le bon contrôle des flux d'entrée avant le traitement, ce qui pourrait amener à des erreurs ;
3. Soit on effectue la même vérification à la fois du coté client et du coté serveur, mais cela impose de dupliquer le même code dans les deux couches.

Pour éviter ce problème, Microsoft propose le framework *WCF RIA Services*. Cet outil duplique du code d'un projet pour le copier dans un autre, de façon automatiquement lors de la compilation. L'opération est schématisé sur la figure II.6. Ceci permet d'avoir un code métier constamment à jour sans avoir à effectuer les modifications dans les différentes couches.

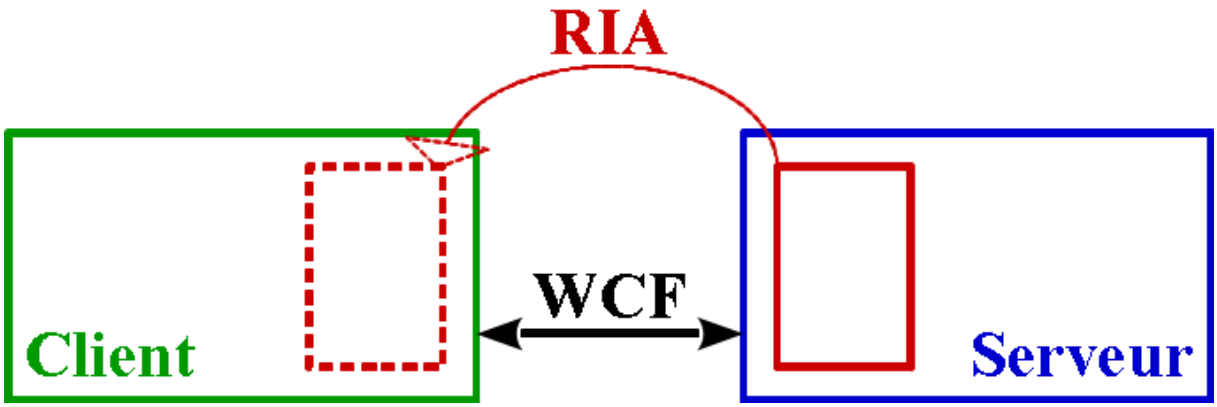


FIGURE II.6 – Fonctionnement de l'application client-serveur

### II.3.4.3 Reporting services

Microsoft fourni un outil de génération de rapports appelés *Reporting services*. C'est un outil est intégré à la suite SQL Server. Les fichiers source utilisés par le service se

présentent sous la forme d'un fichier XML. Visual Studio propose un éditeur graphique simplifiant son édition, la mise en forme ainsi que la création de requêtes.

Le fonctionnement du service est représenté par la figure II.7 et se déroule en plusieurs étapes :

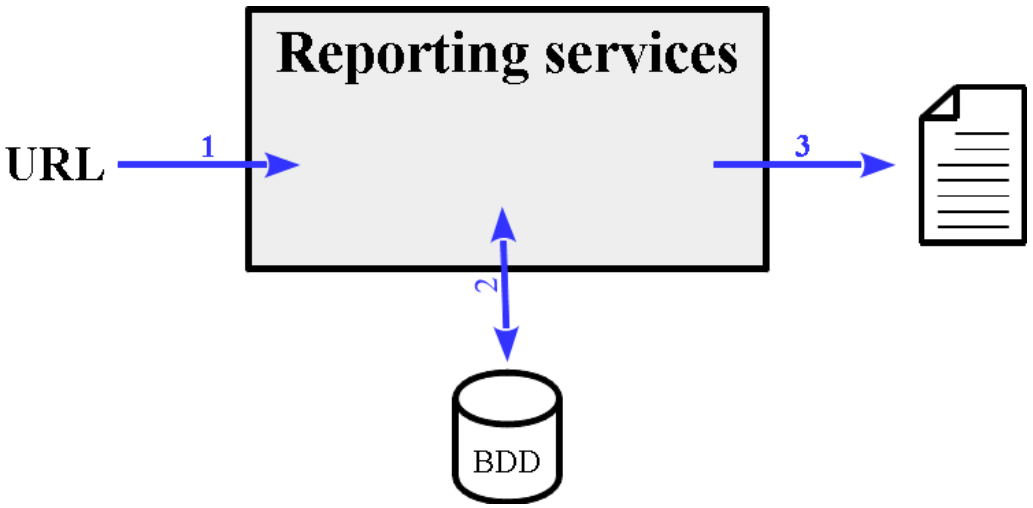


FIGURE II.7 – Fonctionnement de reporting services

1. On appelle le service à l'aide de son adresse IP ou son nom de domaine. Il est possible de passer des paramètres dans l'URL en utilisant la méthode GET. Par exemple, si le service est accessible à l'URL `http://nomserveur/reports/`, l'appel de `http://nomserveur/reports/ID=5&name=toto` permet de spécifier les paramètres `ID` et `name` avec les valeurs respectives `5` et `toto`.
2. Le serveur effectue les requêtes nécessaires dans la base de données afin de récupérer les données dynamiques du rapport.
3. Le rapport est généré puis retourné au client au format HTML, PDF ou autre.

Ce type de service a été utilisé dans le projet. Il est ainsi facile de générer les divers documents à imprimer avec les informations correspondantes aux contrats et candidats. Le serveur de rapports est installé sur la même machine que celle de la base de données, pour ne pas avoir à installer inutilement SQL Server, mais aussi parce que tous les serveurs sont installés sur la même machine.

### II.3.5 Interface utilisateur

#### II.3.5.1 Silverlight

Microsoft *Silverlight* est un plugin pour navigateur web. Il permet le développement d'applications riches et de pousser plus loin l'expérience utilisateur du web 2.0, au même titre qu'Adobe Flash dont il se veut une alternative. Initialement prévu pour des applications web dans un navigateur, les programmes peuvent être téléchargés pour être utilisés directement sur l'ordinateur ("out of browser"), et permettent aussi le développement d'applications pour Windows Phone 7.

Cette technologie nécessite l'installation du plugin, qui est un sous-ensemble de Microsoft .NET Framework. Les applications sont ainsi cross-browser (Internet Explorer, Firefox, Chrome, ...) et cross-platform (Windows, mais aussi OS-X et Linux via le projet open-source Moonlight). De plus les applications fonctionnent dans une "sandbox" ("bac à sable") ce qui permet de garantir une sécurité accrue pour l'utilisateur.

Une des spécifications de Silverlight est l'impossibilité d'appeler des services web de manière synchrone, c'est à dire que la fonction appelante attend la réponse. En effet, lors d'un appel à un service le programme continue son exécution et un événement est émis lors de la réception de la réponse. Cette solution permet de ne pas bloquer l'interface de l'utilisateur qui pourrait penser à un plantage de l'application. Mais l'inconvénient est l'augmentation de la complexité de programmation car il est nécessaire de contrôler plusieurs fils d'exécution en parallèle.

#### II.3.5.2 ASP.NET

Un projet "Web Silverlight" nécessite la création d'un projet web, en plus du projet purement Silverlight. En effet, ce projet web permet d'encapsuler le Silverlight pour le transmettre au navigateur du client. Il est réalisé en *ASP.NET* qui est une technologie Microsoft de programmation web, permettant de créer des sites dynamiques.

Il est nécessaire d'utiliser un serveur web pour déployer le projet ASP.NET. Nous avons encore utilisé une technologie Microsoft : *Internet Information Services*, plus couramment appelé *IIS*.

#### II.3.5.3 SignalR

Le temps réel dans une application web permet au client de recevoir des données au moment où elles sont publiées. Un exemple courant est une discussion instantanée : le

serveur doit informer le client lors de la réception d'un message, sans que celui-ci n'ait à rafraichir la page continuellement.

Avant l'apparition du HTML5, il était nécessaire d'utiliser des techniques JavaScript en envoyant continuellement des requêtes du client vers le serveur. Le HTML5 a apporté cette fonctionnalité grâce aux WebSocket, canal de communication bidirectionnel entre un navigateur web et un serveur web.

Microsoft fourni une bibliothèque intégrée à ASP.NET appelée *SignalR*[2] permettant l'ajout de fonctionnalités de temps réel dans une application web. Elle est basée sur les WebSocket et du JavaScript, facilitant ainsi son utilisation.

2. SignalR - Site web : http ://signalr.net/

# III Résultats - Discussions

Je vais maintenant vous présenter dans ce chapitre la solution obtenue après le développement et livré au client. De plus je présenterai mes impressions durant sa réalisation, dans le domaine technique et fonctionnel.

## III.1 Interface graphique utilisateur

J'ai été libre de programmer la première version de l'interface graphique, comportant les fonctionnalités demandées par le client. Ce dernier a tout de même participé à sa réalisation en effectuant diverses critiques et demandes de réagencement des éléments amenant à une interface graphique optimale et ergonomique.

Je présenterai dans cette section la structure de l'affichage et les différents types d'écran de la solution.

### III.1.1 Connexion

L'application restreint son accès aux personnes autorisées, c'est à dire possédant des identifiants valides. Lorsque l'on se connecte à l'application, la première fenêtre qui s'affiche est l'écran de connexion, que l'on peut apercevoir sur la capture d'écran de la figure III.1.

L'application peut se connecter à un serveur LDAP permettant la gestion des utilisateurs. Lorsque l'utilisateur a saisi son identifiant et son mot de passe, l'application vérifie d'abord s'il s'agit d'un compte LDAP en envoyant une requête au serveur LDAP. Si ce n'est pas le cas, alors l'application va vérifier s'il s'agit d'un compte "applicatif pur" propre à l'application. Ce processus est schématisé par le diagramme de flux présent sur la figure III.2.

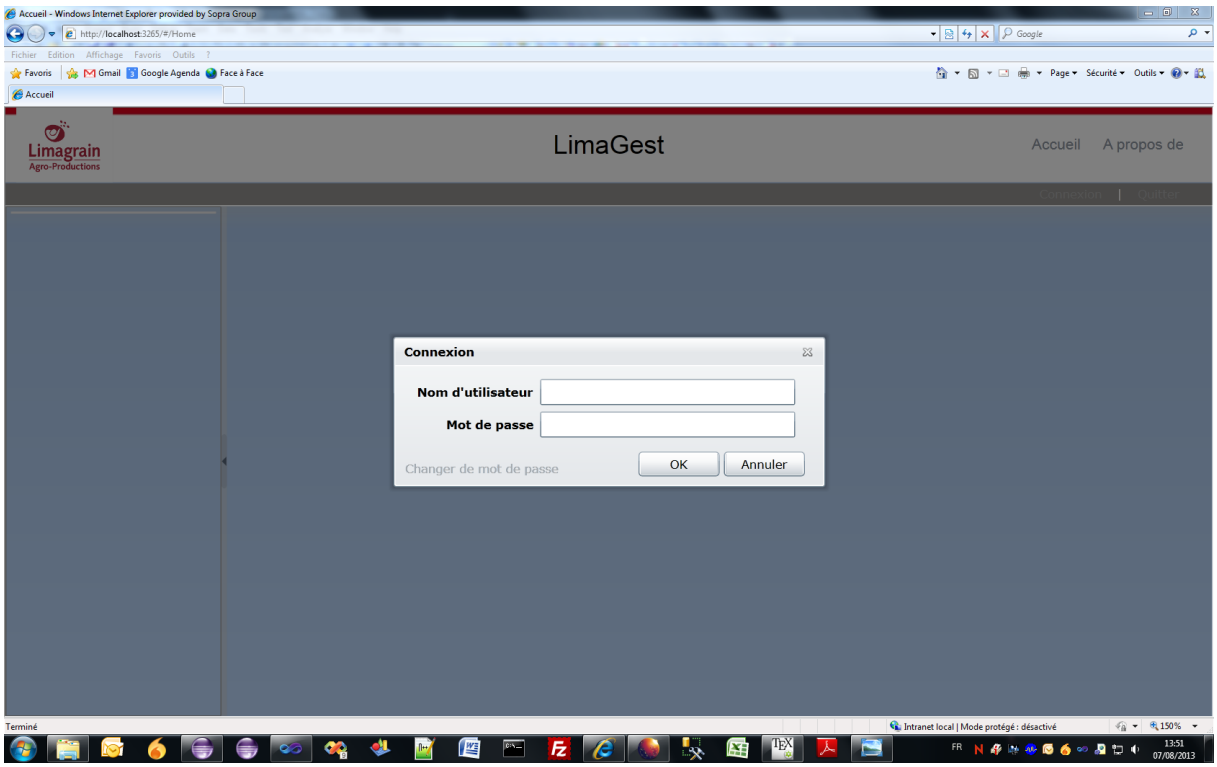


FIGURE III.1 – Écran de connexion

### III.1.2 Structure

La figure III.3 montre la structure de l'interface graphique, avec les trois parties principales : la barre supérieure, le menu latéral, et le contenu où s'affichent les fenêtres.

#### III.1.2.1 Barre supérieure

La partie supérieure de l'écran possèdes deux fonctions. La première est l'affichage des détails de l'application, tels que le logo de l'entreprise, le nom de l'application, et des liens généraux. Cela permet d'identifier l'application utilisée. La seconde est liée au compte de l'utilisateur, avec des liens permettant de changer d'utilisateur, de se déconnecter, et permettant de quitter l'application.

#### III.1.2.2 Menu latéral

Une barre latéral située sur la partie gauche propose un *menu*. Chaque ligne de ce menu correspond à un module de l'application, comme expliqué dans la partie II.3.1.3. La figure III.4 est une capture d'écran du menu, montrant les différents menus de l'application.

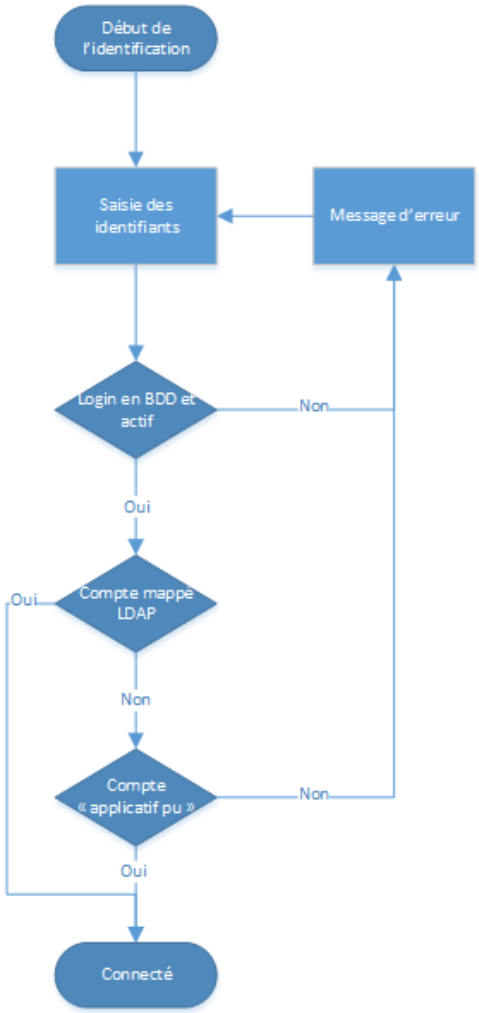


FIGURE III.2 – Processus de connexion

De plus, un sous-menu de second niveau apparait lorsque l'on sélectionne un menu. Il s'agit de chacun des écrans principaux du module correspondant. Par exemple, le menu "Accès" possède quatre sous-menus correspondants à quatre fonctionnalités, comme le montre la capture d'écran de la figure III.4.

Ce menu facilite la navigation entre les différentes fonctionnalités de l'application sans avoir à retourner sur la page d'accueil. Les sous-menus quant à eux permettent de hiérarchiser les fonctionnalités par thème, séparant ainsi les actions liées aux candidats de celles liées aux contrats et encore de l'administration.


FIGURE III.3 – Structure de l'interface graphique

### III.1.3 Les types d'écran

Pour ne pas perdre l'utilisateur, deux types d'écrans ont été utilisés, permettant de garder le même style dans l'application.

Un écran d'affichage de données comporte un tableau permettant d'afficher une liste de valeurs. Il y a une zone de recherche permettant de filtrer les données. De plus, des boutons situés en bas de l'écran permettent d'effectuer des actions sur le ou les élément(s) sélectionné(s) : affichage des détails, modification, suppression, ...

L'affichage, la modification ou la création d'un élément est une fenêtre s'affichant par-dessus l'écran actuel sous la forme d'un pop-up. Lorsqu'il s'agit d'une édition les éléments sont des "zones de texte" ou des "listes déroulantes", sinon ce sont simplement des "labels".

Deux écrans de "paramétrage" sont différents des autres. Ils permettent de gérer les tables "statiques" contenant les valeurs des différentes listes déroulantes que l'on peut trouver dans l'application. Ils sont composés de plusieurs tableaux affichant les données présentes, avec des boutons permettant l'ajout, la modification ou la suppression de valeurs.

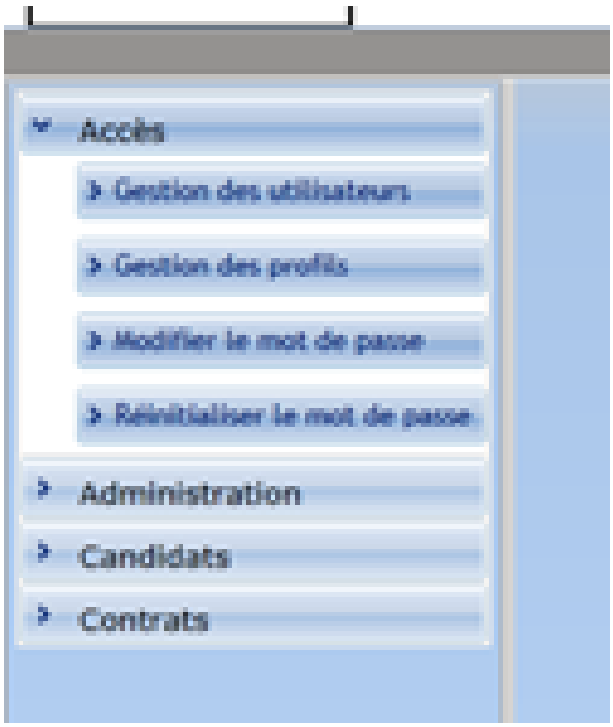


FIGURE III.4 – Menu latéral gauche

## III.2 Base de données

Le modèle de données actuel satisfait bien le besoin du client. Il représente correctement les différentes données à enregistrer, avec les contraintes et dépendances imposées.

Cependant il reste quelques améliorations possibles. En effet, le client a tendance à ajouter, modifier ou supprimer certaines spécifications au court du projet, imposant des changes plus ou moins importants dans le modèle. Ces changements impactent aussi les différentes couches de l'application, du mapping à l'interface graphique, ce qui peut faire perdre un temps important en fin de projet. Pour ne pas retarder la livraison du lot 1, les modifications "non bloquantes" n'ont pas été apportées et le seront lors du lot 2.

## III.3 Aspect fonctionnel

### III.3.1 Contact avec le client

Travailler en assistance chez le client m'a permis de découvrir un univers totalement différent de celui de l'agence Sopra Group.

L'avantage est d'être en contact direct avec le client, ce qui permet d'avoir des informations en temps réel. Cela évite donc de devoir appeler ou échanger des mails, et les explications sont plus claires.

En contrepartie, le client a tendance à demande l'état d'avancement du projet plus régulièrement. Cette proximité procure une pression supplémentaire, notamment lorsque le projet est en retard sur les prévisions.

### III.3.2 Les spécifications

Le principal problème rencontré durant ce projet est l'absence de périmètre dans les spécifications. Celles-ci ont en effet été rédigées en parallèle, voir même en se basant sur les résultats de l'application. Cette porte ouverte a permis au client de demander de nouvelles fonctionnalités au fur et à mesure de l'avancement.

# Conclusion

Ce projet avait pour objectif le développement d'une nouvelle application de recrutement à destination du service recrutement du groupe Limagrain. L'application devait reprendre les fonctionnalités de l'ancien outil, en y ajoutant de nouvelles fonctionnalités pour satisfaire aux besoins de l'utilisateur.

La première version de l'application est actuellement terminée et en production chez le client. Les fonctionnalités prévues dans le premier lot ont été implémentées. Un second lot est prévu pour le mois de septembre, ajoutant de nouvelles fonctionnalités et appliquant les modifications éventuelles.

Certaines améliorations sont possibles, notamment au niveau ergonomie, mais cela nécessiterait l'intervention d'une personne spécialisée.

Les principaux problèmes rencontrés concernaient la découverte des différentes technologies utilisées, notamment le socle utilisé sur lequel se base les différents projets.

Ce projet m'a permis d'améliorer ma communication et la compréhension du besoin, du fait de la proximité directe avec le client. De plus j'ai pu découvrir de nouvelles technologies approfondissant ainsi mes connaissances. J'ai pu ainsi découvrir le métier d'ingénieur avec ses différents aspects fonctionnels et techniques.

# Webographie

TODO