Institut **S**upérieur
d'**I**nformatique, de
Modélisation et de
leurs **A**pplications

BP 10125
63173 Aubière Cedex

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

Stage chez Sopra Group

Présenté par : Julien PINGUET
Tuteur ISIMA : Claude MAZEL
Sous la direction de : Delphine JARIGE
Laurent LAFFERRÈRE
Pierre DASCHER

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 son 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 apportée lors des difficultés rencontrées.

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

Table des figures

II.1 Échange d'une message chiffré 8
II.2 Échange d'un message signé 9
II.3 Hiérarchie des certificats 11
II.4 Processus de création du SOD 13
II.5 Page HTML de saisie des groupes de données 14

III.1 Architecture du projet 18
III.2 Architecture 3 tiers 19
III.3 Utilisation des profils et droits des utilisateurs 21
III.4 Couches de .NET Framework 22
III.5 Interface de PowerAMC 24
III.6 Fonctionnement de l'application client-serveur 27
III.7 Fonctionnement de reporting services 28
III.8 Pattern Modèle-Vue-VueModèle 29
III.9 Écran de connexion 31
III.10 Processus de connexion 32
III.11 Structure de l'interface graphique 33
III.12 Menu latéral gauche 33

IV.1 Niveaux d'administration de l'application 38
IV.2 Hiérarchie de Sopra Group France 39
IV.3 Diagramme UML des instances 40
IV.4 Managing des instances 41
IV.5 Modification d'un domaine 42
IV.6 Association d'utilisateurs à une instance 43

Résumé

Mon stage de fin d'études à l'ISIMA avait pour but l'intégration d'une société de services en ingénierie informatique, dans laquelle j'ai participé à trois projets.

Tout d'abord, j'ai effectué de la tierce maintenance applicative sur des sujets orientés vers la sécurité et le chiffrement de données.

Ensuite, j'ai participé à la refonte d'une application de recrutement. J'ai aidé à la compréhension du besoin et j'ai effectué le développement de l'application et du modèle de donné.

Enfin, j'ai rejoint l'équipe du projet pour l'Imprimerie Nationale. J'ai implémenté plusieurs fonctionnalités du back office de l'application.

Mots clé : Sopra Group, Microsoft, VB.NET, Silverlight, SQL Server, WCF RIA Services

Abstract

During my ISIMA third year studies, my work placement's aim was to incorporate an information technology services and software development company, in which I participated in three projects.

I made first the third-party applications maintenance in some security and encryption oriented topics.

After, I participated in project to redesign an application for recruitment. I helped the comprehension of customer needs and I developped the application and database model.

At last I joined the Imprimerie Nationale team. I developed some back office fonctions of the application.

Keywords : Sopra Group, Microsoft, VB.NET, Silverlight, SQL Server, WCF RIA Services

Table des matières

Introduction **1**

I Introduction de l'étude **3**

I.1 Présentation de l'entreprise 3

I.1.1 Historique 3

I.1.2 Agence de Clermont-Ferrand 3

II Sides **5**

II.1 Présentation 5

II.2 Sides Lituanie : modification de la capture des empreintes digitales 5

II.2.1 Contexte 5

II.2.2 Contrôle de la date 6

II.3 Signature des données de la puce des passeports 6

II.3.1 Pré-requis 6

II.3.1.1 Le chiffrement 6

II.3.1.2 La signature 7

II.3.1.3 Le certificat 9

II.3.1.4 La PKI 9

II.3.2 EJBCA 10

II.3.2.1 Utilisation dans Sides 10

II.3.2.2 Documentation 10

II.3.3 Document security object 11

II.3.3.1 Application de test 12

II.3.3.2 Documentation de fonctionnement 12

III Limagrain **15**

III.1 Étude du problème 15

III.1.1 Contexte 15
III.1.2 Besoin de l'utilisateur 15
III.1.3 Solution actuelle 16
III.1.4 Solution envisagée 16
III.2 Conception de la solution 17
III.2.1 L'équipe du projet 17
III.2.2 La relation client 17
III.2.2.1 Assistance technique 17
III.2.2.2 Définition du besoin 17
III.2.3 Technologies et architecture 18
III.2.3.1 Socle 19
III.2.3.2 Langage de programmation 21
III.2.3.3 Base de données 23
III.2.3.4 Les services 25
III.2.3.5 Interface utilisateur 28
III.3 Résultats et discussions 30
III.3.1 Interface graphique utilisateur 30
III.3.1.1 Connexion 30
III.3.1.2 Structure 31
III.3.1.3 Les types d'écran 34
III.3.2 Base de données 34
III.3.2.1 Modèle de données 34
III.3.2.2 Importation des anciennes données 35
III.3.3 Aspect fonctionnel 35
III.3.3.1 Contact avec le client 35
III.3.3.2 Les spécifications 36
III.3.3.3 La communication 36

IV Imprimerie Nationale **37**
IV.1 Le contexte 37
IV.2 L'administration 37
IV.2.1 Le Front Office 38
IV.2.2 Le Back Office 38
IV.2.2.1 Administration 38
IV.2.2.2 PGIN 39

IV.3 Gestion des instances 39
IV.3.1 Présentation 39
IV.3.2 Les écrans 40
IV.3.2.1 Managing 40
IV.3.2.2 Affichage, modification et création 40
IV.3.2.3 Les utilisateurs 41
IV.4 Gestion des profils 42
IV.4.1 Généralités 42
IV.4.2 Différents niveaux 42
IV.4.2.1 Administration 43
IV.4.2.2 PGIN 43
IV.5 Résultats 44

Conclusion **45**

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.

XML : "Extensible Markup Language" langage de balisage générique.

Introduction

Le travail d'un ingénieur en informatique peut être fonctionnel avec une présence chez le client pour comprendre son besoin, mais peut aussi être technique ayant comme objectif le développement de l'application.

La société de service en ingénierie informatique n'est pas une exception, et l'une de ses spécificité est le changement régulier de projet. Les technologies et les domaines d'activité varient d'un projet à l'autre nécessitant de s'adapter selon les situations.

L'objectif de ce stage de fin d'étude est de mettre en pratique mes diverses connaissances acquises durant ma scolarité sur divers projets, à la fois dans le domaine technique que fonctionnel.

Dans ce rapport je présenterai mon intégration dans la société de service en ingénierie informatique qu'est Sopra Group et le travail que j'ai réalisé.

Je présenterai tout d'abord mes différentes tâches sur le projet Sides qui ma permis d'intégrer Sopra Group. Dans un second temps je présenterai le projet pour Limagrain. Enfin je présenterai le projet pour l'Imprimerie Nationale effectué en fin de stage.

I Introduction de l'étude

I.1 Présentation de l'entreprise

I.1.1 Historique

Sopra Group est une Société de Service en Ingénierie Informatique, couramment appelé SSII. Créée en 1968 par Pierre PASQUIER et François ODIN, il s'agit d'une des plus anciennes et importantes 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'affaires 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 services. 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 mène les projets Sides et SITI spécialisés dans la gestion de titres (documents d'identité, passeport, cartes à puce, ...), et travaille pour une quinzaine de gouvernements. Le groupe Limagrain, dont le siège social est situé Chappes, fait actuellement appel aux services de Sopra Group pour la réalisation du projet BOS.

II Sides

II.1 Présentation

Sides est un projet de gestion de titres sécurisés conçu par Sopra Group, plus particulièrement par l'agence de Clermont Ferrand. Il permet de produire des passeports ou des cartes d'identités pour une quinzaine de gouvernements comme les Philippines, la Lituanie, la Belgique, Monaco, ...

Pour chacun des pays et titres, une application a été développée. Il existe cependant une base commune pour plusieurs projets, mutualisant notamment pour les périphériques, l'intégration avec la base de données, les calculs mathématiques et le traitement des images.

Actuellement l'équipe Sides n'effectue que de la tierce maintenance applicative (TMA) sur les différentes applications existantes. L'objectif est de corriger les bugs éventuels ou d'ajouter de nouvelles fonctionnalités.

II.2 Sides Lituanie : modification de la capture des empreintes digitales

II.2.1 Contexte

L'application Sides à destination de la Lituanie permet la gestion de passeports. Ces passeports nécessitent la lecture des empreintes du porteur pour être valides. Lorsque les porteurs sont des personnes diplomatiques ou haut placées, celles-ci désireraient ne pas perdre de temps lors de création ou renouvellement de titres.

Pour satisfaire ces exigences, une personne du gouvernement munie d'un ordinateur

portable, de l'application Sides et du périphérique va effectuer la saisie des empreintes sur le lieu désiré : ville plus proche, lieu de travail, ...

II.2.2 Contrôle de la date

La première version de cette fonctionnalité permettait l'importation des empreintes les plus récentes présentes dans la base de données pour créer un nouveau passeport. Les empreintes peuvent venir d'une ancienne demande ou avoir été saisies pour une personne diplomatique comme expliqué précédemment.

Pour sécuriser l'application et le chargement d'anciennes empreintes, l'évolution de la fonctionnalité a pour objectif de contrôler la date d'importation des empreintes dans la base de données.

Pour réaliser ce contrôle, j'ai dû étudier le fonctionnement de l'application pour comprendre le processus de chargement des empreintes dans une nouvelle demande de passeport.

Il était nécessaire d'ajouter une nouvelle entrée dans le fichier de configuration définissant le nombre de jours durant lesquels les empreintes peuvent servir à renouveler un passeport.

II.3 Signature des données de la puce des passeports

II.3.1 Pré-requis

La gestion des identités de personnes, au niveau national, est un domaine très sensible qui requière une sécurité élevée des systèmes informatiques. La solution permettant de sécuriser les données consiste à chiffrer ou signer les données échangées, garantissant ainsi que les données ne pourront être découverte ou qu'elles n'ont pas été modifiée ou envoyées par une personne tiers. Ce chiffrement s'effectue grâce aux clés des certificats.

II.3.1.1 Le chiffrement

Pour assurer que personne ne puisse découvrir le message échangé entre deux personnes il est nécessaire de les chiffrer.

Cette méthode utilise deux entités : un couple de clés composé d'une clé publique (K) et d'une clé privée (K'), et un algorithme mathématique composé d'une fonction (F) et de sa réciproque (F'). Le chiffrement consiste à utiliser une clé dans l'algorithme, et le déchiffrement à utiliser la fonction inverse avec l'autre clé pour retrouver les données d'origine, comme le résume la fonction suivant :

$$F_K[\ F_{K'}(M)\ ] = M = F'_{K'}[\ F_K(M)\ ]$$

Le fonctionnement de l'échange d'un message chiffré est le suivant, schématisé par la figure II.1 :

1. Bob possède une paire de clé publique/privée, et Alice désire envoyer un message à Bob ;
2. Bob envoie sa clé publique à Alice ;
3. Alice chiffre son message grâce à cette clé publique ;
4. Alice envoie ensuite son message chiffré à Bob ;
5. Bob déchiffre le message chiffré grâce à sa clé privée. Personne d'autre ne peut le faire car il est le seul à posséder la clé privée.

La sécurité est basée sur deux principes : le chiffrement et la confidentialité.

Si l'algorithme de chiffrement est trop faible alors il sera possible de déchiffrer les données par calcul mathématiques. De plus, si la clé de chiffrement est trop petite alors la méthode "brute force", consistant à essayer toutes les combinaisons possibles, permettra aussi le déchiffrement

Si la clé privée du certificat est découverte alors les données seront instantanément déchiffrées, ainsi que les futurs messages. Il est donc nécessaire de renouveler les clés régulièrement.

II.3.1.2 La signature

La signature numérique permet de vérifier l'authenticité et la non-modification d'un message.

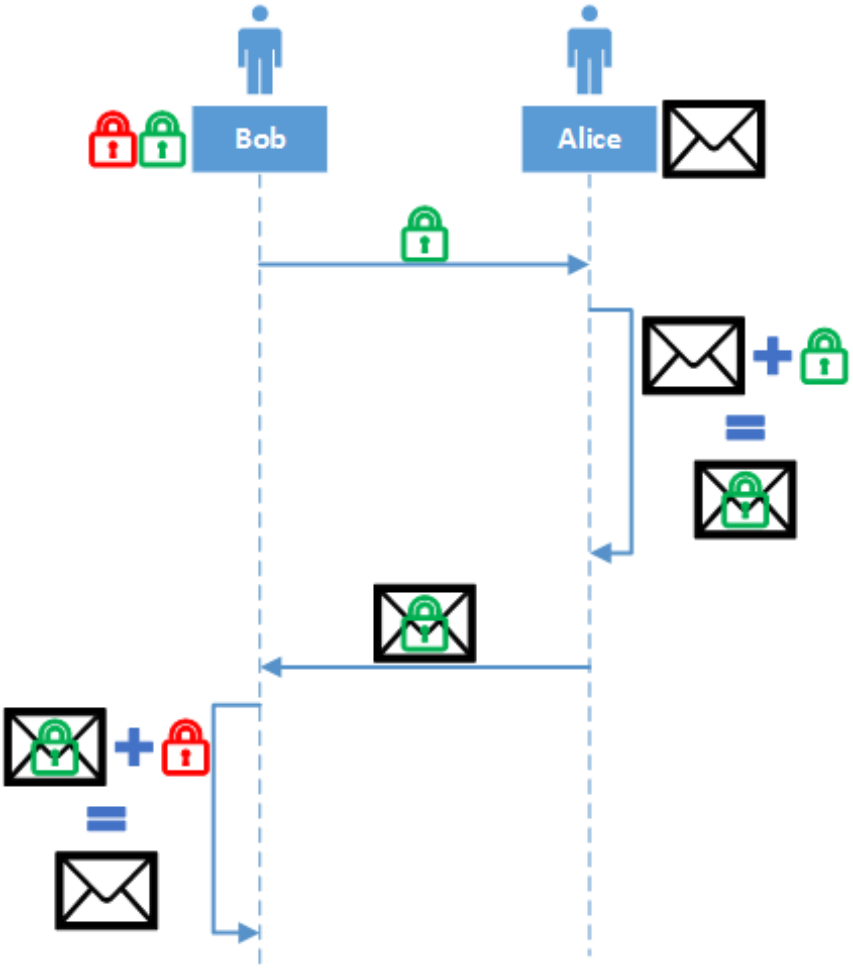

FIGURE II.1 – Échange d'une message chiffré

Cette méthode utilise une fonction hachage qui a pour objectif de créer une empreinte à taille fixe. Les algorithmes de hachage les plus utilisés sont MP5 et SHA.

Le fonctionnement est le suivant, schématisé par la figure II.2 :

1. Bob hache le message, puis chiffre ce hash avec sa clé privée, ce qui produit la signature du message ;
2. Bob envoie à Alice le message, la clé publique et la signature ;
3. Alice hache le message, et déchiffre la signature du message grâce à la clé publique. Ensuite elle compare les deux valeurs, qui seront égales si le message n'a pas été modifié et a été chiffré avec la clé privée associée à la clé publique.

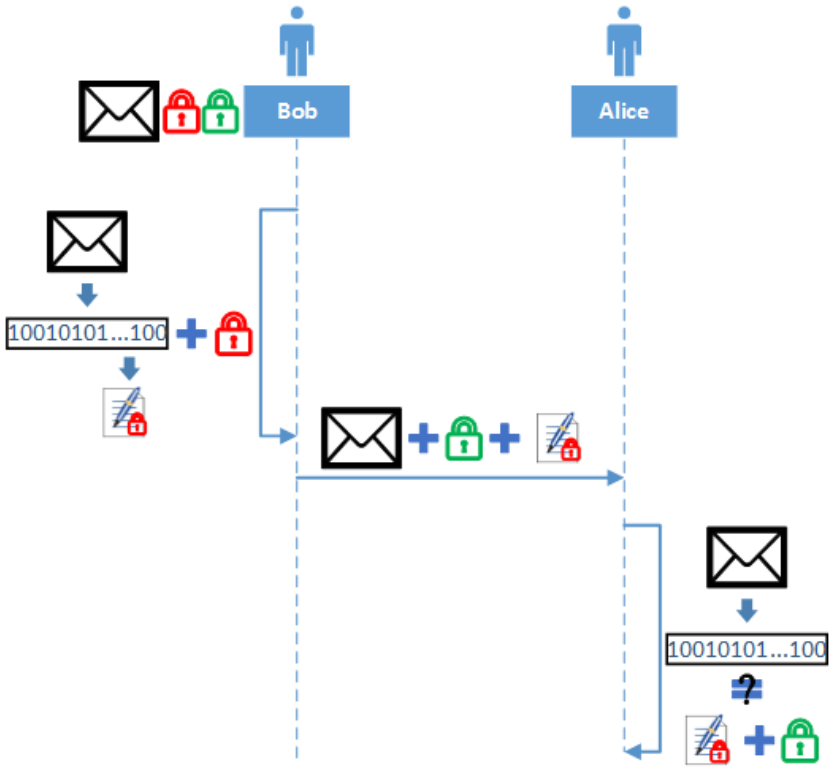

FIGURE II.2 – Échange d'un message signé

II.3.1.3 Le certificat

Un certificat est composé de :

- une clé publique, et éventuellement la clé privée associée ;
- des informations sur le propriétaire ;
- une signature.

Les certificats sont signés par une autorité de certification, aussi appelé CA pour Certificat Autority, qui permet de garantir l'authenticité des certificats. Seule l'autorité de certification racine est auto-signée.

II.3.1.4 La PKI

Une *infrastructure à clés publiques*, aussi appelé par son terme anglais *public key infrastructure*, permet la gestion de grandes quantités de certificats ou clés publiques. Elle fournie des services de création, de publication et de révocation de certificats. Cette infrastructure est principalement dans les systèmes informatiques utilisant de nombreux certificats.

II.3.2 EJBCA

II.3.2.1 Utilisation dans Sides

En production, le projet Sides utilise de nombreux certificats pour sécuriser les données. Pour cela une PKI est utilisée permettant la gestion des nombreux certificats.

Lorsque l'on souhaite tester l'application avant la mise en production, il est nécessaire de pouvoir produire des certificats. La mise en place d'une PKI peut s'avérer fastidieuse à intégrer et paramétrer pour fournir les mêmes services que la PKI de production.

Pour cela une PKI a été déployée sur les serveurs de tests, fournissant ainsi les fonctionnalités minimales utiles, notamment la production de certificats. La PKI open-source EJBCA est utilisée par l'équipe Sides et le client pour gérer des certificats de test.

II.3.2.2 Documentation

Mon second objectif dans l'équipe Sides a été de rédiger une documentation de production de certificats en utilisant EJBCA. Ces certificats doivent respecter les spécifications d'ICAO [1], International Civil Aviation Organization, qui définissent les procédures et conseils destinés aux états et fournisseurs de solutions de PKI.

La documentation sera utilisée par le client pour produire ses certificats de tests. Elle devait donc être la plus précise possible, détaillant toutes les étapes de la création, en spécifiant chacun des paramètres et options.

Afin de définir les paramètres que les certificats implémenteront, il est possible de définir des profils. J'ai tout d'abord créé deux profils qui seront utilisés pour le CSCA et le DS.

Le *CSCA*, pour Country Signing Certificate Autority, est le certificat de plus haut niveau, utilisé par le le pays. Il s'agit plus du plus critique car de lui dépend tous les autres certificats utilisés. Dans certains pays, celui-ci est stocké dans un ordinateur portable situé dans un coffre fort, qui est ouvert en cas de besoin notamment lors du renouvellement des certificats.

Le *DS*, pour Document Signer, est utilisé pour signer les documents sécurisés. Il est situé au second niveau dans la hiérarchie des certificats, et est signé par le certificat racine CSCA.

1. ICAO - Site web : `http://www.icao.int`

Des certificats, composés d'une clé publique et une clé privée, peuvent ensuite être créés. Ils peuvent être téléchargés sous le format PKCS#12 qui est un fichier à l'extension ".p12". L'utilisateur peut ensuite l'utiliser ou l'installer sur son environnement.

La figure II.3 représente la hiérarchie des différents certificats utilisés.

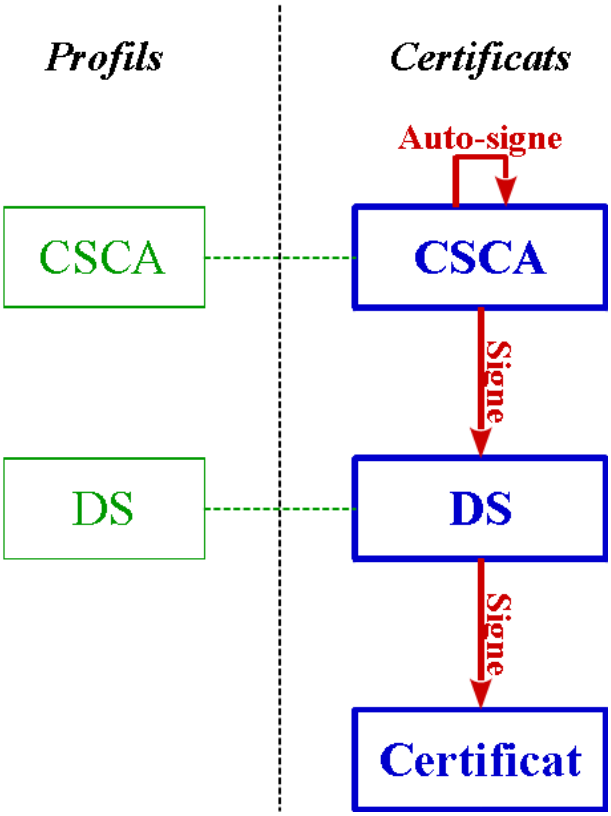

FIGURE II.3 – Hiérarchie des certificats

II.3.3 Document security object

Le *SOD*, Document Security Object, est la signature d'un document. Cette signature est générée en utilisant la clé privée d'un certificat de type DS, présenté dans la section précédente.

Dans ce cadre j'ai travaillé sur le SOD des cartes à puce, plus particulièrement sur la création de la signature. Ces cartes à puces contiennent :

1. des données, disposées dans seize groupes (DG, pour Data Group), contenant les informations comme l'identité et la photo d'une personne ;

2. la signature des données appelée SOD, créée à partir des différents DG et d'un certificat.

Le SOD permet de vérifier que les données de la carte à puce n'ont pas été modifiées ou altérées, garantissant ainsi l'authenticité des passeports produits par le système Sides. Le processus de vérification est similaire à la vérification de la signature d'un message présenté dans la section II.3.1.2 :

- On déchiffre le SOD avec la clé publique du DS ;
- On hache les données contenues dans les DGs ;
- Si les deux valeurs sont égales alors les données sont correctes.

II.3.3.1 Application de test

L'application web de création de signature est développée en JEE (Java Enterprise Edition). Une *servlet* est une classe java appelée lors d'une requête HTTP et permet de générer dynamiquement des données (généralement des pages HTML). Le projet web fournie plusieurs servlets permettant d'utiliser plusieurs méthodes de signature, comme la connexion à un HSM (Hardware Security Module) ou l'utilisation de certificat PKCS#12 décrit précédemment.

Dans l'objectif de fournir au client une application permettant de tester la création de signature SOD à partir d'un certificat PKCS#12, j'ai modifié le projet web initial pour ne garder que la fonctionnalité correspondante. L'épuration du code source et la correction de certains bugs ont permis de produire une solution fonctionnelle qui fourni les services de signature de données décrites dans la partie II.3.3.2. Il était nécessaire que je travaille sur le fichier de configuration utilisé par le programme pour que celui-ci puisse être utilisé par l'utilisateur sans difficultés.

II.3.3.2 Documentation de fonctionnement

Ensuite, j'ai rédigé une documentation permettant d'expliquer comment créer la signature à partie des différents groupes de données (DG). La création utilise l'application web déployée par le client, qui prend en paramètre les valeurs hexadécimales des groupes de données. La signature est ensuite retournée sous la forme d'un fichier binaire. Le figure II.4 schématise le processus de création de la signature SOD.

Une page HTML, dont une capture d'écran est présente sur la figure II.5, permet de faciliter la saisie manuelle des données et l'appel au service en proposant respectivement

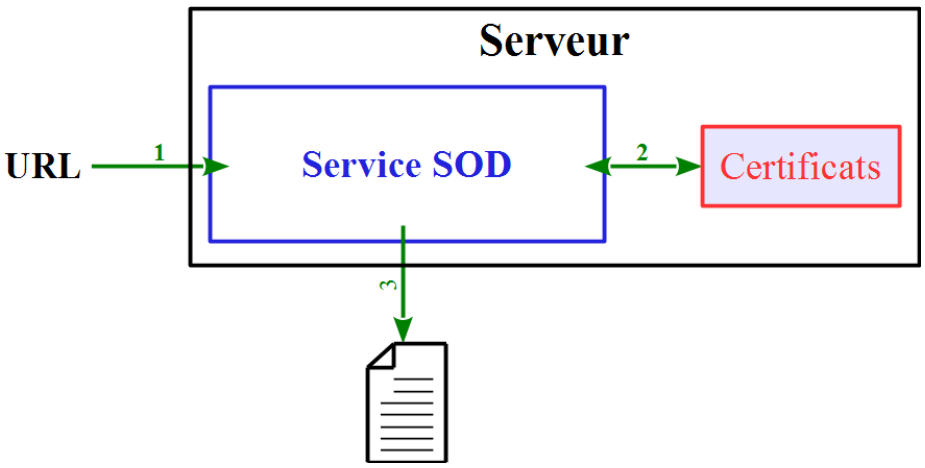

FIGURE II.4 – Processus de création du SOD

des zones de saisie et un bouton de soumission. La fenêtre de téléchargement s'affiche ensuite pour proposer d'ouvrir ou de sauvegarder le fichier constituant la signature.

Il est aussi possible d'appeler le service de manière automatique depuis un programme. Pour expliquer cela j'ai développé un exemple de script Python permettant de générer une signature.

```
import urllib

# parameters
params = urllib.urlencode({"dataGroup1": "02468ACE",
                           "dataGroup2": "13579BDF"})

# invoke with POST parameters
dataPOST = urllib.urlopen("http://localhost:8080/sod", params)

# saving SOD
fileSOD = open("D:\\Desktop\\sod.out", 'w')
fileSOD.write(dataPOST.read())
fileSOD.close()

dataPOST.close()
```

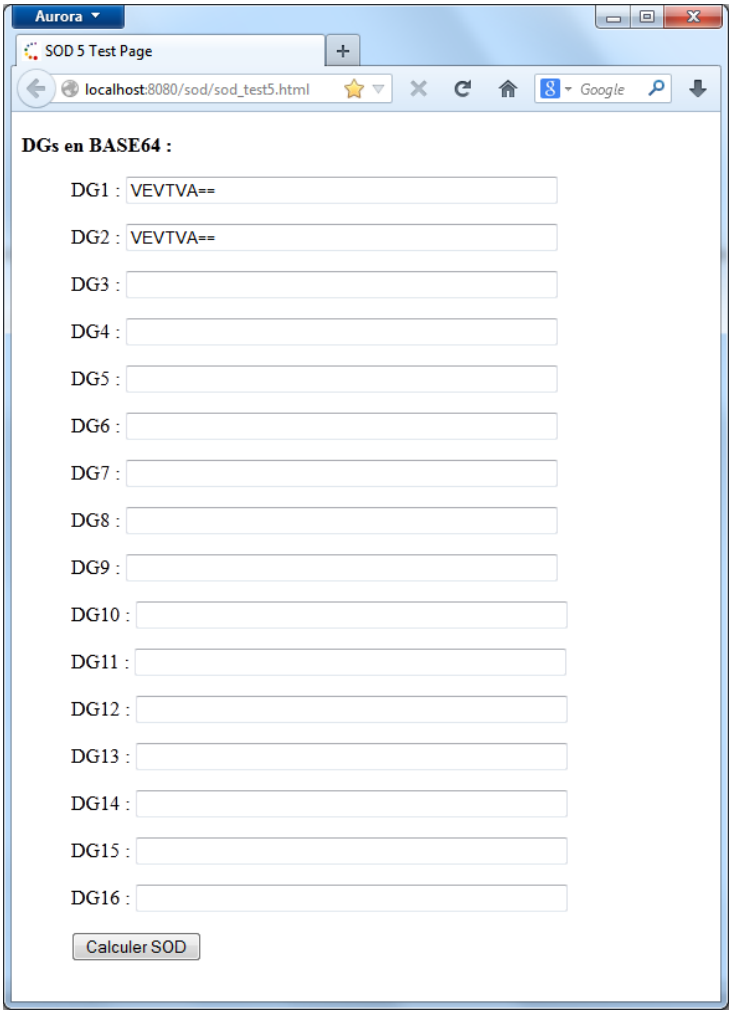

FIGURE II.5 – Page HTML de saisie des groupes de données

III Limagrain

III.1 Étude du problème

III.1.1 Contexte

Limagrain est un groupe coopératif agricole spécialisé dans les semences et les produits céréaliers.

Le groupe Limagrain possède un pic d'activité sur la saison estivale, période à laquelle les céréales sont plantées, entretenues et récoltées. Limagrain reçoit plus de 1.000 candidatures en moins d'un mois dont une grande partie des candidats effectuent un ou plusieurs contrats.

III.1.2 Besoin de l'utilisateur

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, union 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 et par 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 et qui n'ont 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. Le nombre de candidatures et contrats est en effet de plus en plus élevé au cours des années.

III.1.3 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.

III.1.4 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 d'abord 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'offrent 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.

III.2 Conception de la solution

III.2.1 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 relais 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 qui 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.

III.2.2 La relation client

III.2.2.1 Assistance technique

Le projet s'est réalisé en mode *assistance technique*. Il n'y a aucune date de livraison 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.

III.2.2.2 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, leurs types,

les contraintes, ainsi que les liens entre elles. Cette partie est importante car la structure de l'application est fortement liée à 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.

III.2.3 Technologies et architecture

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

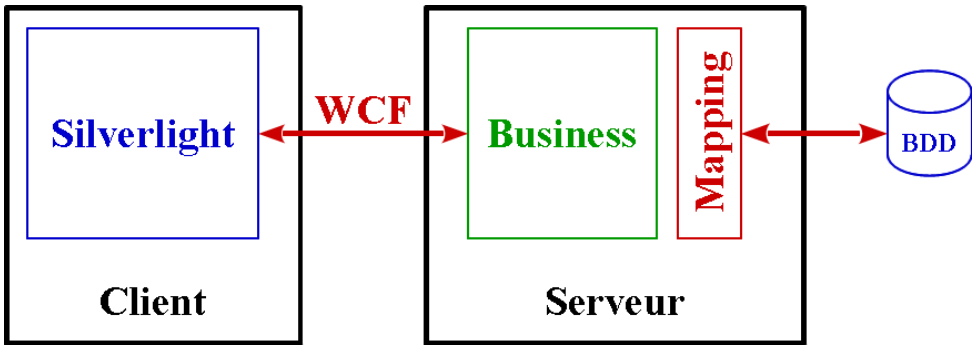

FIGURE III.1 – Architecture du projet

Le groupe Limagrain travaille dans un environnement Microsoft 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 évolutivité. De plus cela permet la séparation des problèmes diminuant ainsi la complexité.

Notre application est divisée en 3 couches, appelée *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 III.2.

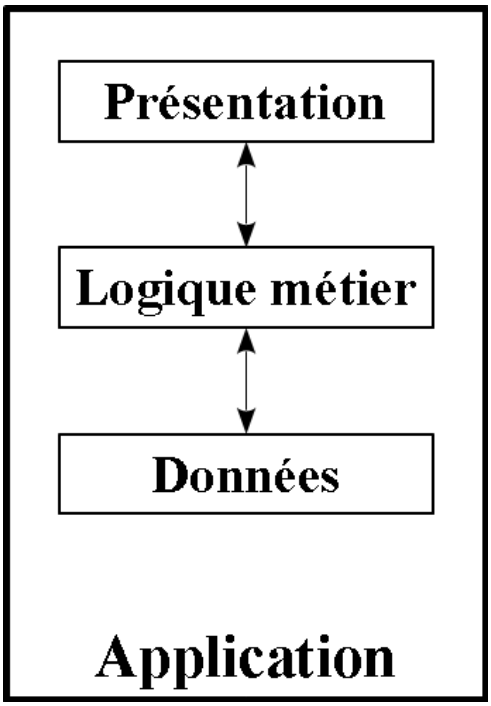

FIGURE III.2 – Architecture 3 tiers

III.2.3.1 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.

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'utilisation d'un seul et même compte pour tous les services connectés : Windows, la boite mail Outlook, cette application, ...

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 "ressources humaines" 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 droit directement, permet un gain de temps. En effet, l'administrateur des 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 III.3 représente le schéma UML des utilisateurs, profils et droits dans l'application.

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) aucune interaction entre elles.

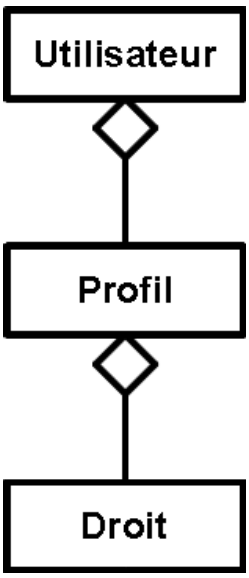

FIGURE III.3 – Utilisation des profils et droits des utilisateurs

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

III.2.3.2 Langage de programmation

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 constitué de nombreux composants, disposés en couches au fur et à mesure de ses versions, schématisés sur la figure III.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)), ...

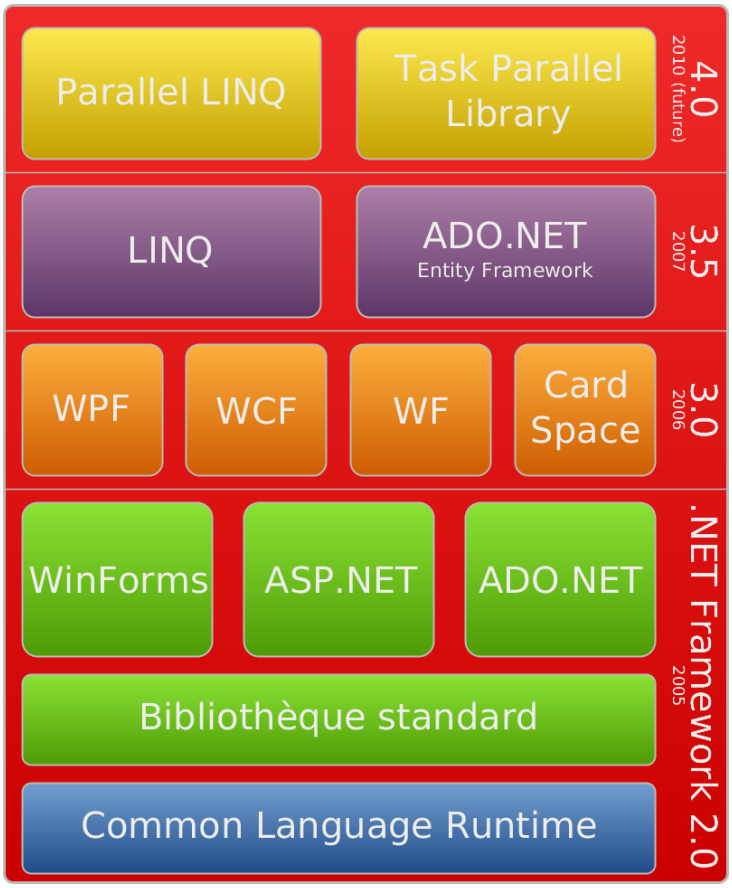

FIGURE III.4 – Couches de .NET Framework
Source : Wikipedia

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 la couche présentation.

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.

III.2.3.3 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.

PowerAMC

PowerAMC[2], version francophone de PowerDesigner, est un logiciel de modélisation de bases 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 III.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 d'effectuer les différentes modifications très simplement.

SQL Server

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

Mapping de la base de données

Le "modèle relationnel" est utilisé dans les systèmes de gestion de bases 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.

1. log4net - Site web : `http://logging.apache.org/log4net/`
2. PowerAMC - Site web : `http://www.sybase.fr/products/modelingdevelopment/poweramc`

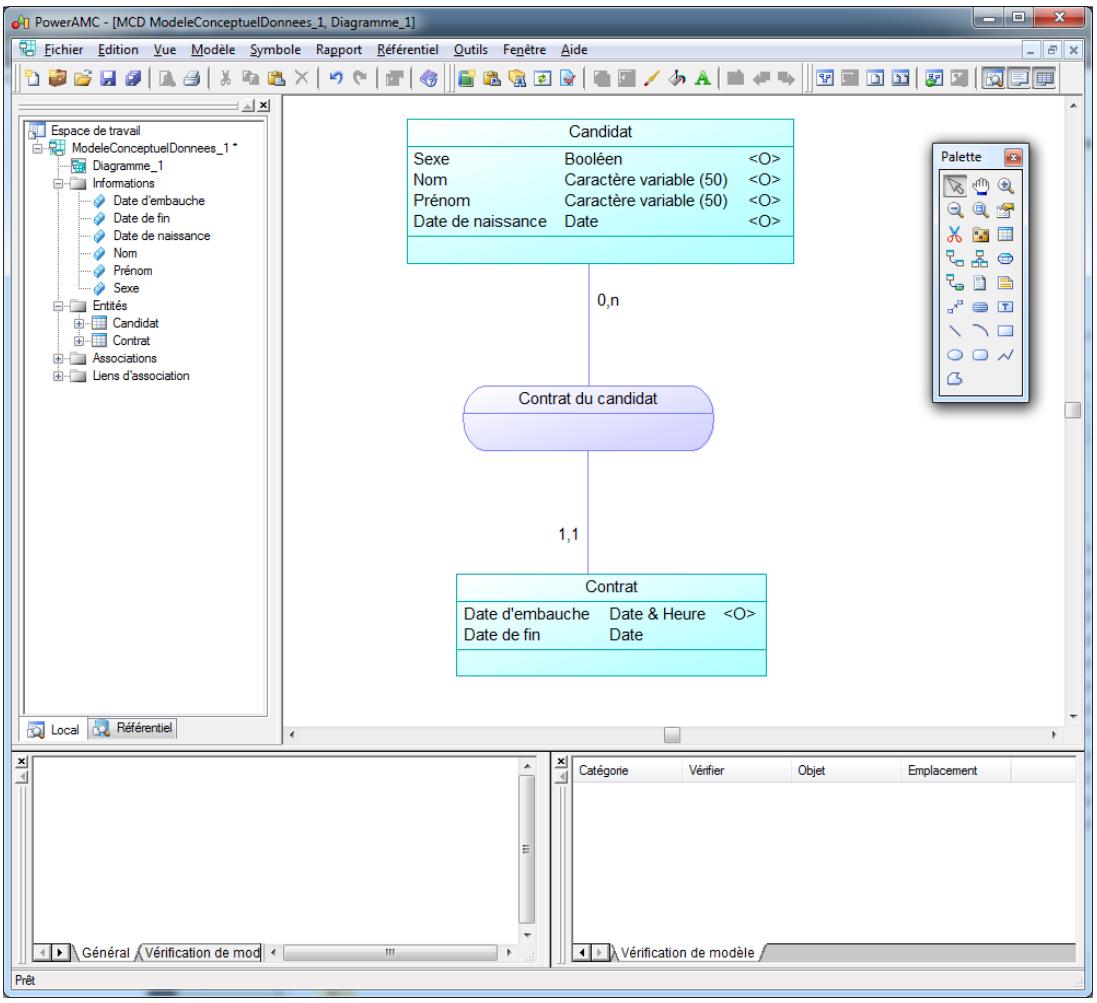

FIGURE III.5 – Interface de PowerAMC

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`).

Cette opération peut être faite à l'aide d'un framework, permettant 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 bases de données. Ainsi, la réaction, la recherche, la mise à jour ou la suppression (opérations CRUD) de données se font de manières 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.

III.2.3.4 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.

Client-Serveur

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 III.2.3.5 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.

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 sous 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 la 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 [WCF], 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.

WCF RIA Services

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

1. Soit on effectue la vérification côté serveur, impliquant l'échange inutile d'informations lorsque celles-ci sont fausses ;
2. Soit la vérification est faite côté 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 côté client et du côté serveur, mais cela impose de dupliquer le même code dans les deux couches.

Pour éviter ce problème, Microsoft propose le framework *WCF RIA Services* [RIA]. Cet outil duplique du code d'un projet pour le copier dans un autre, de façon automatique lors de la compilation. L'opération est schématisée sur la figure III.6. Ceci permet

d'avoir un code métier constamment à jour sans avoir à effectuer les modifications dans les différentes couches.

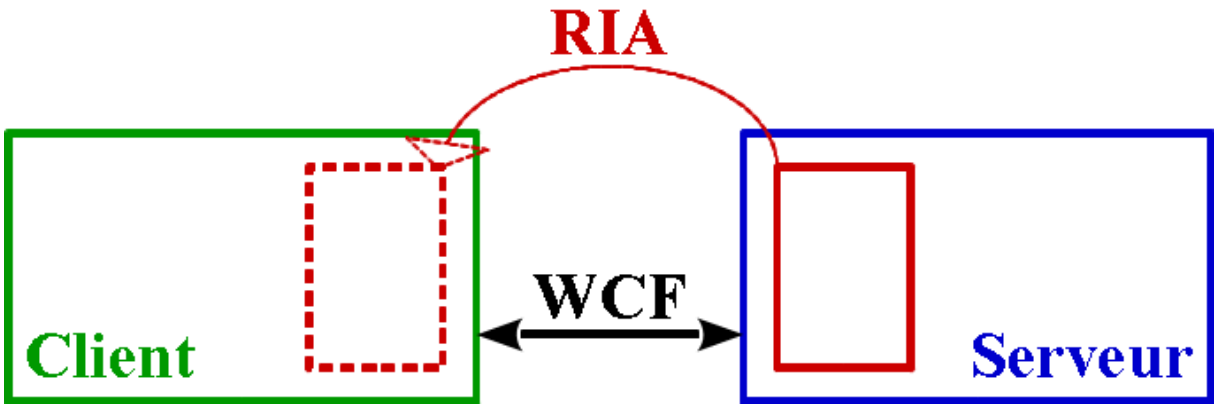

FIGURE III.6 – Fonctionnement de l'application client-serveur

Reporting services

Microsoft fournit un outil de génération de rapports appelé *Reporting services*, 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 III.7 et se déroule en plusieurs étapes :

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.

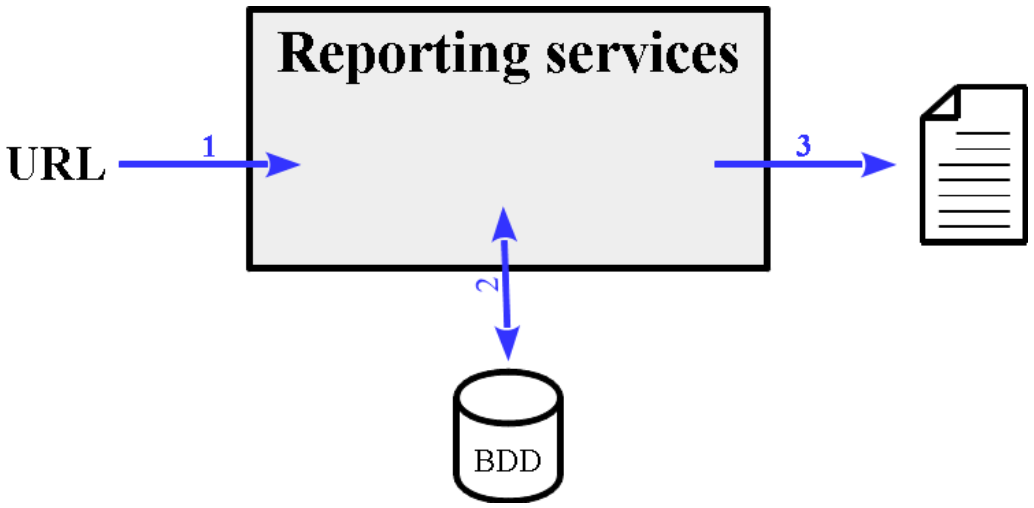

FIGURE III.7 – Fonctionnement de reporting services

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.

III.2.3.5 Interface utilisateur

Silverlight

Microsoft *Silverlight* [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.

Le pattern *Modèle-Vue-VueModèle* a été créé par Microsoft a destination de Silverlight et Windows Presentation Foundation (WPF). Il permet de structurer le développement d'interfaces graphiques en séparant la vue de la logique et de l'accès aux données, comme l'illustre le schéma de la figure III.8. Le couplage est plus faible qu'avec le pattern Modèle-Vue-Contrôleur (MVC), ce qui permet une maintenance, une réutilisabilité et des tests plus faciles.

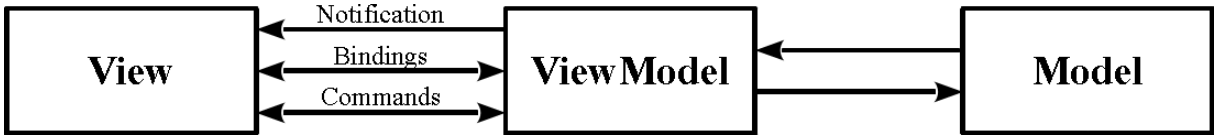

FIGURE III.8 – Pattern Modèle-Vue-VueModèle

La vue du pattern MVVM utilise le langage *XAML*, dérivé du langage XML, développé pour les besoins de Microsoft. Son aspect générique sans code source permet aux interfaces graphiques d'être réutilisées dans diverses application et divers langages (C#, VB.NET, ...).

L'exemple qui suit affiche une zone de saisie dont le contenu est bindé à la propriété `PersonName` du ViewModel. Ainsi une modification de la View mettra à jour automatiquement la propriété, et inversement une modification de la propriété du ViewModel mettra à jour la View.

```
<Window xmlns="...">
    <TextBox Text="{Binding Path=PersonName}"/>
</Window>
```

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*.

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 fournit une bibliothèque intégrée à ASP.NET appelée *SignalR*[3] 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.

III.3 Résultats et discussions

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.

III.3.1 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.

III.3.1.1 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.9.

3. SignalR - Site web : `http://signalr.net/`

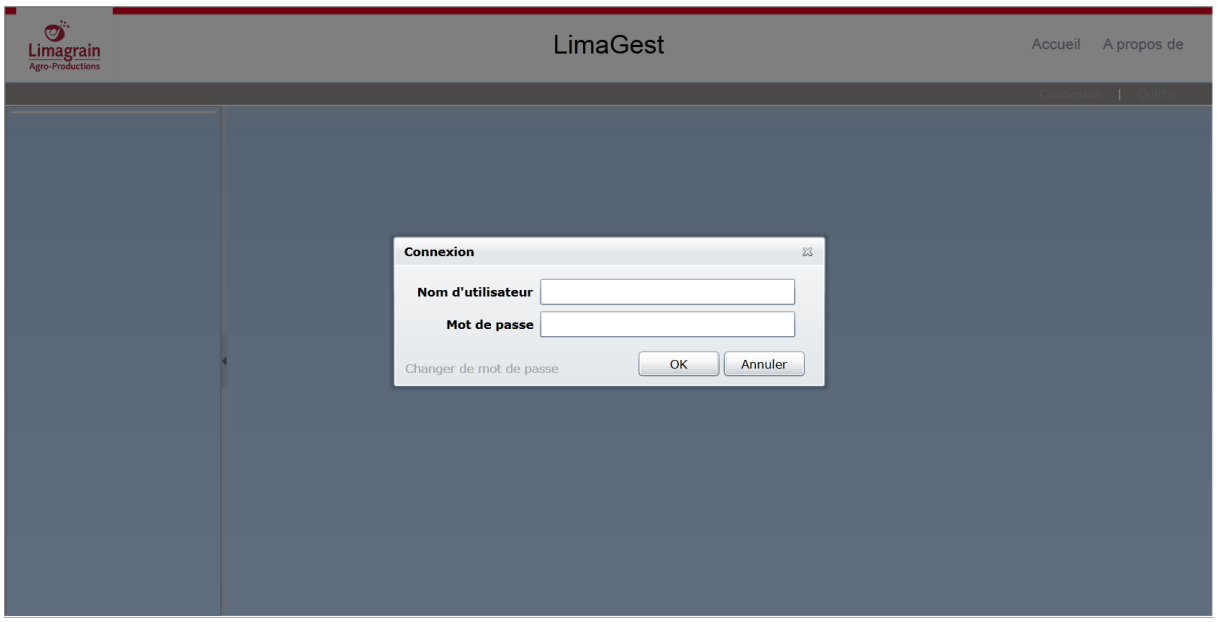

FIGURE III.9 – Écran de connexion

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.10.

III.3.1.2 Structure

La figure III.11 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.

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.

Menu latéral

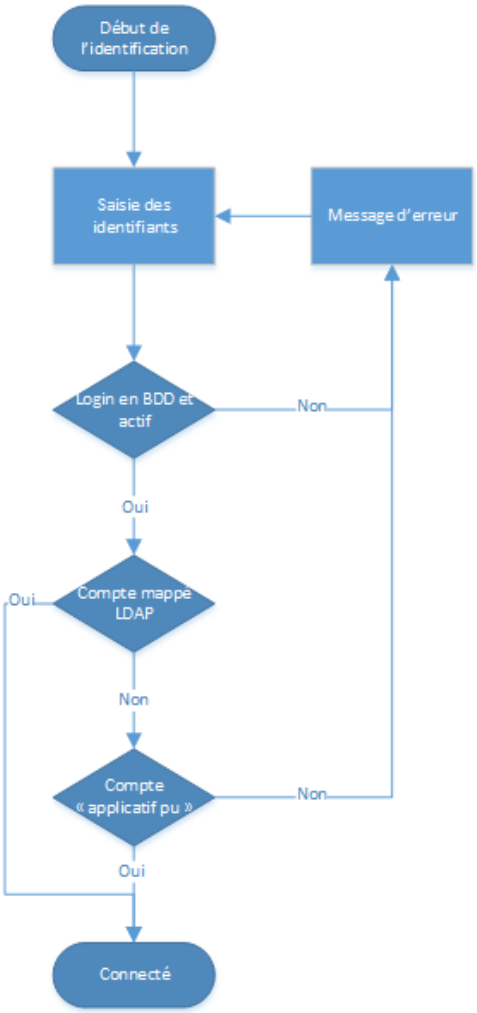

FIGURE III.10 – Processus de connexion

Une barre latérale située sur la partie gauche propose un *menu*. Chaque ligne de ce menu correspond à un module de l'application, comme expliqué précédemment. La figure III.12 est une capture d'écran du menu, montrant les différents menus de l'application.

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.12.

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

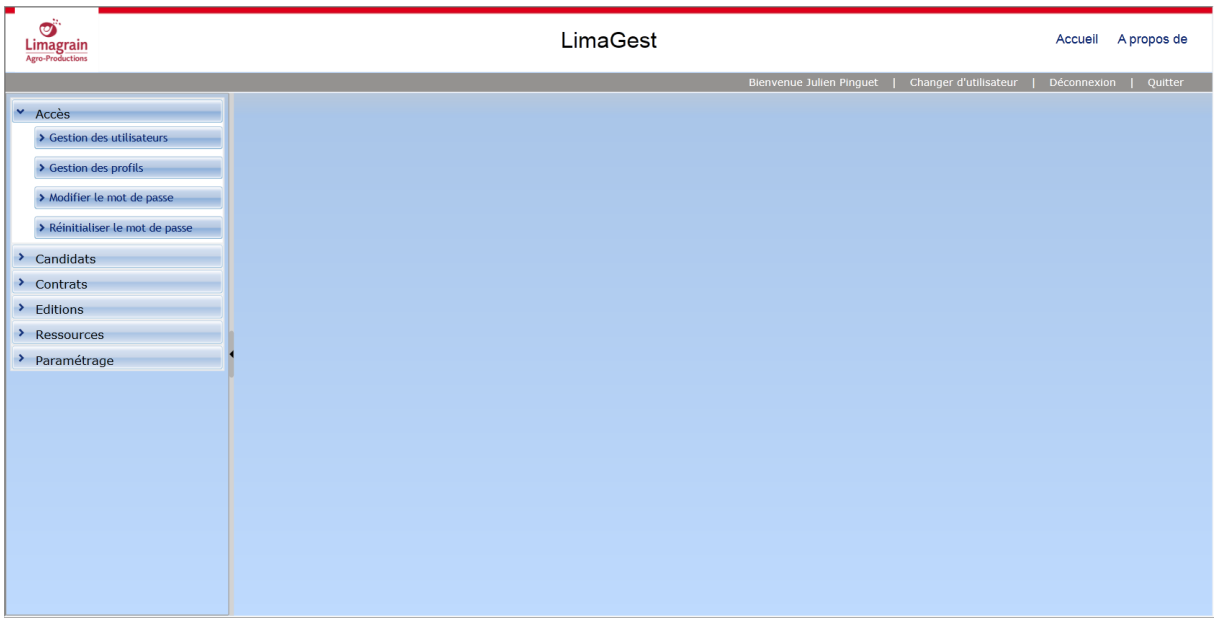

FIGURE III.11 – Structure de l'interface graphique

FIGURE III.12 – Menu latéral gauche

hiérarchiser les fonctionnalités par thème, séparant ainsi les actions liées aux candidats de celles liées aux contrats et encore de l'administration.

III.3.1.3 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.

III.3.2 Base de données

III.3.2.1 Modèle 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.3.2.2 Importation des anciennes données

Plusieurs milliers de candidats et contrats ont été saisis avec l'ancienne application par le service de recrutement. De nombreux candidats effectuent plusieurs contrats dans le groupe Limagrain. Pour ne pas avoir à ressaisir ces informations, il est nécessaire d'effectuer une importation des anciennes données dans la nouvelle base de données.

Le changement du modèle de données entre l'ancienne et la nouvelle version de la base de données impose certaines modifications.

Lorsque des champs sont ajoutés et qu'ils sont obligatoires, il est nécessaire de définir leur valeur. S'il s'agissait d'un simple champs (entier, chaîne de caractères, date, ...) alors j'ai soit spécifié la valeur par défaut (0, la chaîne vide, la date minimale, ...), soit défini une valeur spécifique que l'utilisateur devra mettre à jour (par exemple "A définir" ou la date du jour), selon ce que modélise le champ.

Si un champ change de type, comme par exemple le passage d'un stockage sous forme de chaîne de caractères (`String`) à entier (`int`), il est souvent nécessaire d'effectuer une conversion, si cela est possible. Ce changement de type permet de s'adapter aux besoins de l'utilisateur, ou améliorer la cohérence des données dans la base de données en empêchant le stockage d'informations invalides.

C'était le cas pour un numéro de sécurité social, initialement stocké sous la forme d'une chaîne de caractères, les données seront maintenant stockés sous la forme d'un entier à 15 chiffres.

III.3.3 Aspect fonctionnel

III.3.3.1 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 demander 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.3.3.2 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, par manque de budget et de charges sur le projet.

Cette porte ouverte a permis au client de modifier le périmètre du projet au cours du temps. De nouvelles fonctionnalités ont été demandées au fur et à mesure de l'avancement. De plus, certaines fonctionnalités ont dû être modifiées à plusieurs reprises pour répondre aux besoins du client.

III.3.3.3 La communication

Ce projet m'a amené à utiliser mes compétences de communication à plusieurs reprises et dans diverses situations.

Les besoins de l'utilisateur peuvent parfois prendre un temps important pour être réalisés techniquement. Il m'est donc arrivé de négocier avec le client pour concevoir une solution intermédiaire, avec le meilleur compromis entre temps de développement et fonctionnalité désirée.

Une fois la première ébauche de l'application terminée, une réunion a été faite chez le client en présence des diverses personnes du service de recrutement qui utiliseront l'application. Cette réunion avait pour objectif de présenter l'aspect général de l'application, ainsi que les différentes fonctionnalités implémentées. Il a fallu aussi répondre aux différentes interrogations du client, notamment la date à laquelle sera opérationnelle l'application.

IV Imprimerie Nationale

IV.1 Le contexte

L'objectif du projet est de concevoir une solution de gestion d'identité numérique. Ces identités sont stockées sur des supports physiques, tels que des badges ou des cartes à puce. Des services seront proposés à ces supports, il peut s'agir par exemple d'accès à un bâtiment, de chiffrement ou de signature de données sensibles, ...

L'Imprimerie Nationale désire vendre cette solution à divers clients dans le but de leur produire leurs supports. L'architecture retenue sera donc *multi-tenant*, c'est-à-dire que chaque client possèdera ses propres données. Cette solution sera hébergée à l'Imprimerie Nationale et chacun des clients pourra se connecter à l'application pour gérer ses supports et services.

Ce projet est réalisé par plusieurs sociétés :
- Keynectis pour l'infrastructure de gestion de clés
- Dictao pour l'activation des cartes à puce ;
- l'Imprimerie Nationale pour la personnalisation des identités ;
- et Sopra Group pour le développement du système de gestion d'identités.

L'équipe du projet est composé d'une quinzaine de personnes, majoritairement de l'agence de Clermont Ferrand, mais dont certains étaient rattachés à l'agence de Lyon ou Grenoble.

IV.2 L'administration

L'application est divisée en divers niveaux afin de séparer les différentes fonctionnalités. La figure IV.1 représente les différents niveaux présentés dans cette section.

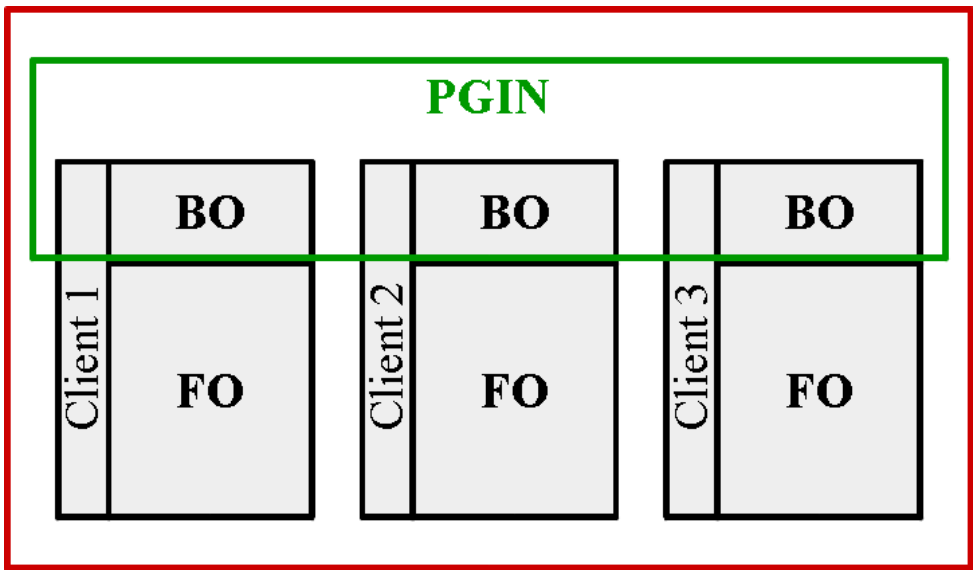

FIGURE IV.1 – Niveaux d'administration de l'application

IV.2.1 Le Front Office

Le *Front Office* (FO) est la partie accessible par les utilisateurs finaux. Elle propose les principales fonctionnalités principales de l'application. Dans notre cas il s'agit des fonctions de gestion des supports : demande, validation, révocation, ...

IV.2.2 Le Back Office

Le *Back Office* (BO) est quant à lui destinée aux administrateurs de l'application. L'objectif de cette partie est de gérer les utilisateurs et droits, mais aussi de gérer les données "statiques" qui apparaitront dans la partie front office comme par exemple l'ensemble des supports disponibles.

Il s'agit du niveau sur lequel j'ai travaillé durant mon stage et que je vous présenterai dans ce chapitre.

IV.2.2.1 Administration

Il est nécessaire que chaque client puisse administrer les proposer données qui lui sont propre, sans impacter les données des autres clients. Pour cela un niveau "Administrateur" a été mis en place.

IV.2.2.2 PGIN

L'imprimerie va produire des cartes pour l'ensemble de ses clients. Ainsi des données seront communes à chacune des clients et il est nécessaire de pouvoir les gérer efficacement. De plus, il est nécessaire de pouvoir administrer l'application de chacun des clients, en cas de problème ou pour effectuer des maintenances.

Dans cet objectif, un niveau "PGIN" (signifiant Plateforme de Personnalisation de l'Identité Numérique) a été mis en place afin d'administrer l'ensemble des données des clients. Il s'agit d'un niveau "super administrateur" au niveau de l'Imprimerie Nationale.

IV.3 Gestion des instances

IV.3.1 Présentation

Une instance, aussi appelé niveau hiérarchique, et une entité de l'entreprise. Les différentes instances forment une hiérarchie, représentée sous la forme d'un arbre. L'instance racine est la compagnie et est unique pour un client donné. Les instances sous-jacentes sont les divisions, et les feuilles sont les domaines. La solution actuelle se limite à trois niveaux dans la hiérarchie.

À titre d'exemple, on peut représenter Sopra Group France qui est composée de divisions (Rhône-Alpes Auvergne, ...) qui sont elles-même composées d'agences (Clermont Ferrand, Lyon, ...). La figure IV.2 représente la hiérarchie de Sopra Group France.

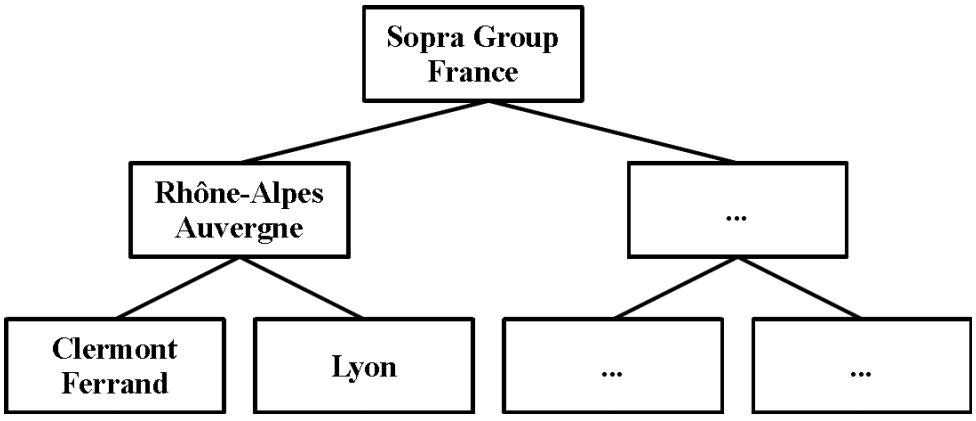

FIGURE IV.2 – Hiérarchie de Sopra Group France

IV.3.2 Les écrans

J'ai tout d'abord travaillé sur la gestion des instances. L'objectif est de pouvoir effectuer les opérations courantes (affichage, modification, création, suppression), ainsi que l'association des utilisateurs. La figure IV.3 est le diagramme UML des instances.

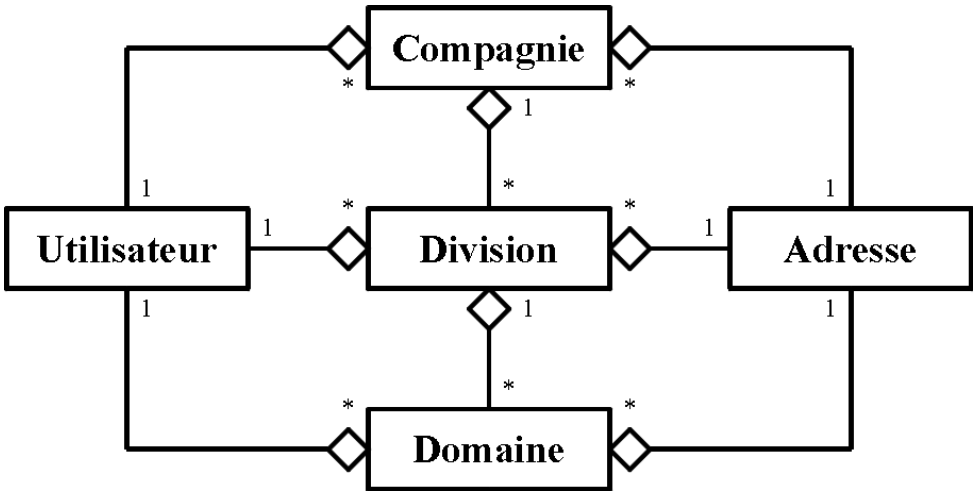

FIGURE IV.3 – Diagramme UML des instances

IV.3.2.1 Managing

L'écran d'accueil de la gestion affiche la hiérarchie des instances, présenté sur la figure IV.4 qui en est une capture d'écran. Des boutons d'action permettent d'agir sur l'instance sélectionnée.

La suppression d'une instance se fait en sélectionnant une instance dans la hiérarchie de l'écran puis en cliquant sur le bouton "Supprimer". Pour des raisons de sécurité, il est impossible de supprimer une instance si celle-ci possède des liens avec d'autres données dans la base de données. Par exemple il est impossible de supprimer une division s'il existe des domaines sous-jacents.

IV.3.2.2 Affichage, modification et création

Dans ce type d'actions, il existe deux types d'écran. L'affichage montre simplement à l'utilisateur les valeurs des différents champs. Les écrans de création et modification proposent quand à eux une zone de saisie, où les champs sont pré-remplies par la valeur dans le cas d'une modification.

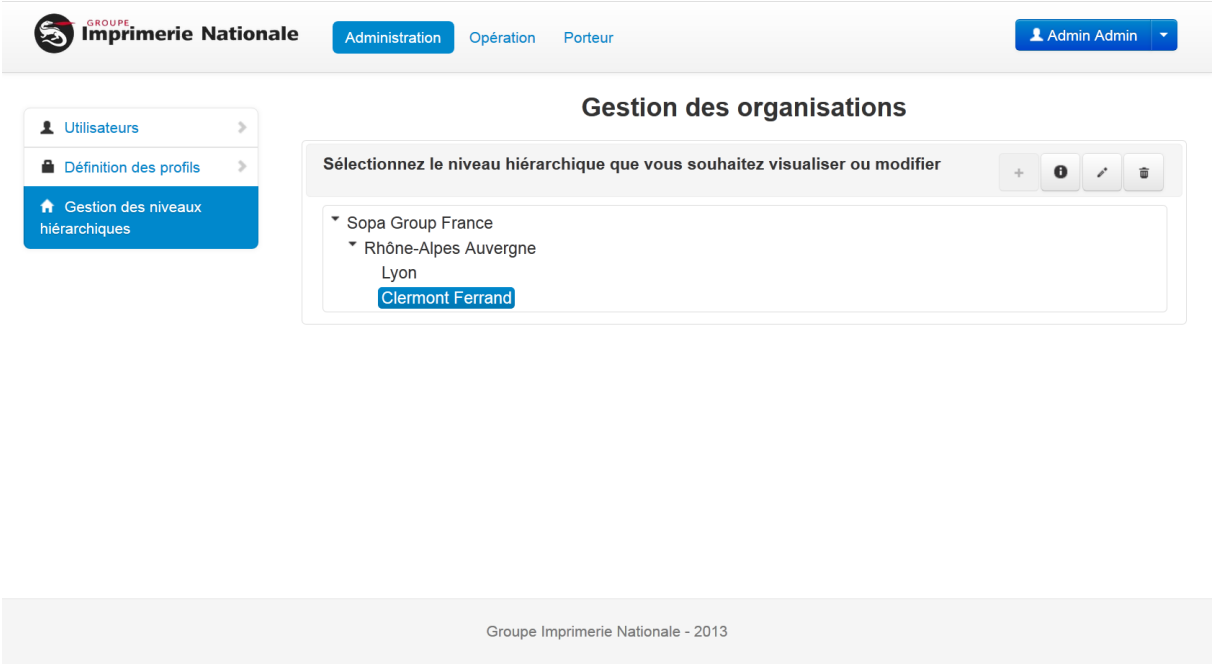

FIGURE IV.4 – Managing des instances

Dans les trois cas, l'utilisateur a accès aux informations des instances parentes en cliquant sur l'onglet correspondant. Par exemple lorsque l'on modifie une division, la division et la compagnie sont affichées (sans modification possible). Cela permet à l'utilisateur de connaître quelle instance il manipule et où elle se situe dans la hiérarchie. La figure IV.5 est une capture d'écran de la fenêtre de modification d'une division.

IV.3.2.3 Les utilisateurs

Chaque utilisateur de l'application est rattaché à une instance. La modification d'une instance permet aussi d'associer de nouveaux utilisateurs ou de supprimer des nouveaux utilisateurs à l'instance.

La figure IV.6 est l'écran d'affectation, composé de deux parties : la liste des utilisateurs disponibles avec un filtre de recherche, et la liste des utilisateurs associés. Les deux listes sont complémentaires, c'est-à-dire que l'association d'un utilisateur à l'instance le fera passer de la liste des disponibles à la liste des associés. L'association se fait en sélectionnant des utilisateurs dans une des liste puis en cliquant sur le bouton "flèche du haut" ou "flèche du bas" pour ajouter ou retirer des utilisateurs.

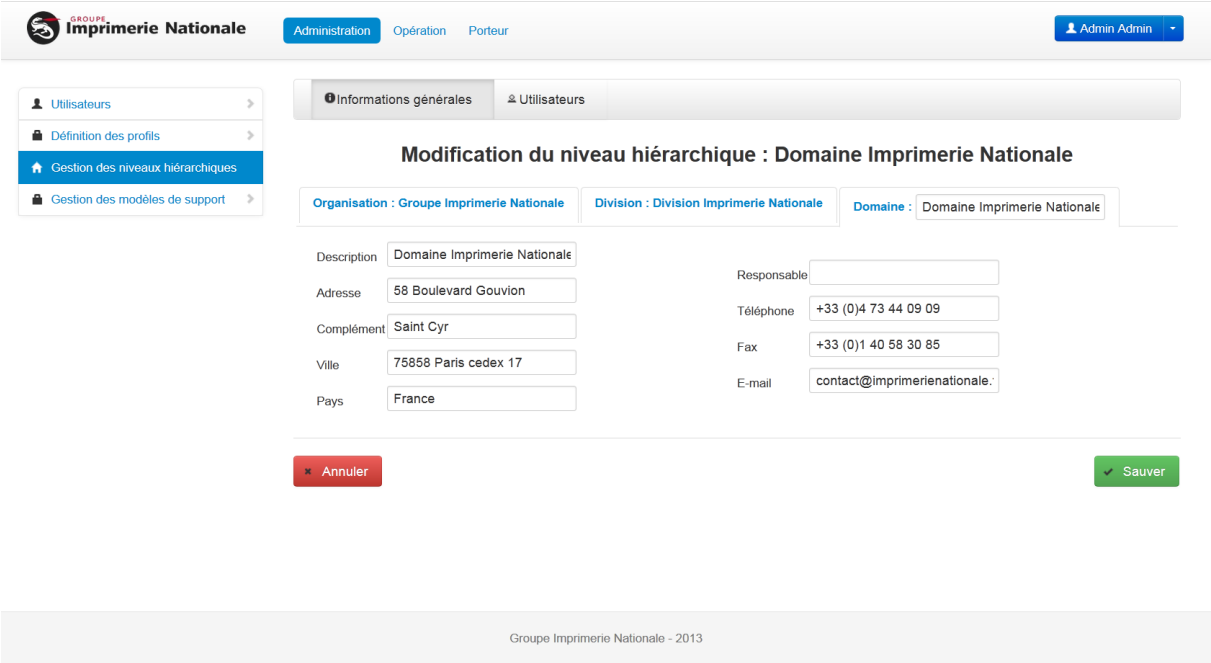

FIGURE IV.5 – Modification d'un domaine

IV.4 Gestion des profils

IV.4.1 Généralités

Les différentes restrictions dans l'application sont gérées par des droits. En fonction des droits que possède l'utilisateur, celui-ci verra des entrées dans les menus, des boutons d'action, ...cachés. Cette gestion des droits se fait finement par l'intermédiaire des profils, comme détaillé dans le chapitre précédent III.2.3.1.

L'écran d'affectation des utilisateurs est identique à celui de la figure IV.6. L'écran d'affectation des droits diffère uniquement sur les données affichées car ici on y manipule des droits.

IV.4.2 Différents niveaux

Il existe deux types de profils dans l'application :

- les profils dits "spécifiques" qui sont propres à un client ;
- les profils dits "communs" qui sont présent chez l'ensemble des clients.

Leur gestion varie en fonction du niveau dans lequel il se trouve.

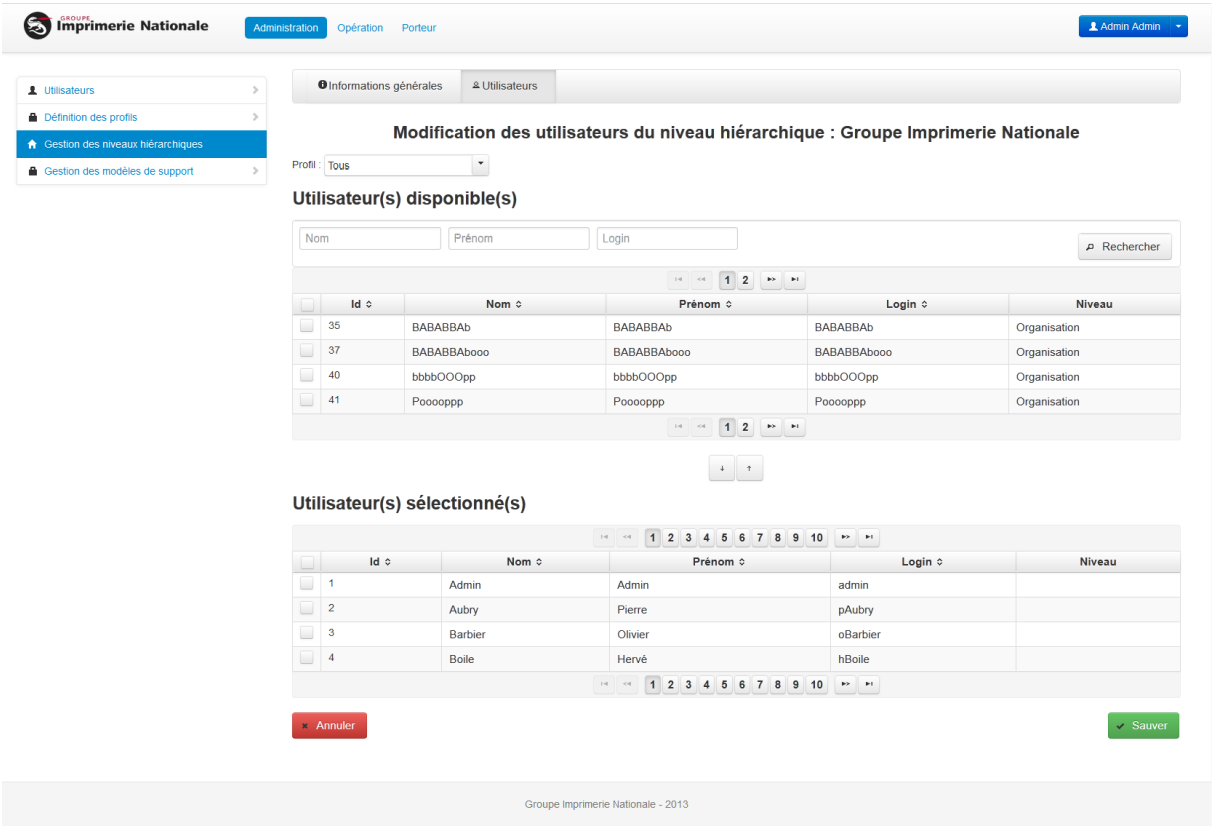

FIGURE IV.6 – Association d'utilisateurs à une instance

IV.4.2.1 Administration

Dans le niveau administration, propre à un client, les profils communs ont une gestion très limitée : il n'est possible de modifier que l'affectation des utilisateurs. Ces profils sont proposés pour fournir des "profils de base" que les clients pourront affecter aux utilisateurs dans le cadre d'une utilisation simple de l'application.

Concernant les profils spécifiques, il est possible d'en créer des nouveaux, de les supprimer, et de modifier l'affectation de droits et utilisateurs. Chaque client peut ensuite personnaliser ses accès et restrictions.

IV.4.2.2 PGIN

Spécifique

Les profils spécifiques permettent d'administrer les droits dans la partie PGIN de l'application.

Commun

Les profils communs demandent des opérations plus complexes dans la base de données. En effet, un profil commun est dupliqué sur l'ensemble des clients, et il est nécessaire que ces profils soient tous identiques. Une action sur un profil commun dans le niveau PGIN devra donc être répercutée chez chacun des clients.

Une problématique importante s'est soulevée lors de l'implémentation des fonctionnalités : retrouver le profil commun. Habituellement un profil est identifié par un identifié unique : la clé primaire. Or lorsque l'on va créer un nouveau profil commun celui-ci n'aura pas le même identifiant dans chacune des tables, en raison du fait que les identifiants sont auto-générés. La solution que j'ai utilisée est d'identifier le profil par une clé alternative, c'est à dire une valeur unique : le nom. Ce nom n'est modifiable par aucun client, comme expliqué précédemment dans la partie IV.4.2.1, ce qui permet d'utiliser le nom pour retrouver les différents profils communs dans les tables des clients.

IV.5 Résultats

Actuellement mes différents écrans répondent aux besoins du client. Ils proposent une partie des fonctionnalités présentes dans la partie back office de l'application permettant d'administrer les droits et les instances.

Le projet a été livré à la fin du mois d'août au client. Actuellement en phase de qualification, l'application sera mise en production à la fin du mois de septembre pour être proposée à d'éventuelles sociétés.

Conclusion

Intégré à une société de services en ingénierie informatique, j'ai participé à l'élaboration de trois projets distincts.

Les principaux problèmes rencontrés concernaient la découverte des différentes technologies utilisées, mais aussi les aspects fonctionnels comme la communication avec le client ou la compréhension du besoin.

Ce stage m'a permis de découvrir de nouvelles facettes du métier d'ingénieur que je n'ai pas abordé lors de mon stage de seconde année. Il a permis d'améliorer mes compétences fonctionnelles avec la communication et la compréhension du besoin, mais aussi techniques avec la découverte de nouvelles technologies.

Webographie

[WCF] Microsoft « Services de données WCF ». 2013. http://msdn.microsoft.com/fr-fr/library/vstudio/cc668792.aspx

[RIA] Microsoft « Services RIA WCF ». 2013. http://msdn.microsoft.com/fr-fr/library/ee707344%28v=vs.91%29.aspx

[Silverlight] Centre de développement Silverlight « Microsoft Silverlight ». 2013. http://msdn.microsoft.com/fr-fr/silverlight/bb187358.aspx