MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
UNIVERSITE DE DOUALA

REPUBLIQUE DU CAMEROUN
Paix – Travail – Patrie

INSTITUT UNIVERSITAIRE DE TECHNOLOGIE

.....

DEPARTEMENT DE GENIE INFORMATIQUE

CONCEPTION ET MISE EN PRODUCTION D'UN SYSTEME DE GESTION DES DEMANDES D'EMPLOI A SAGA/SDV/SOCOPAO/DIT – GROUPE BOLLORE AFRICA LOGISTICS CAMEROUN

MEMOIRE EN VUE DE L'OBTENTION DU DIPLOME DE LICENCE DE TECHNOLOGIE
Option : GENIE LOGICIEL

Présenté et soutenu par

KWAYE NGUEMELIEU Gabriel Kant
Matricule : 09I03184

Sous la supervision de :

Monsieur **AYIMDJI Armel**
Encadreur académique

Monsieur **FONGUEM Eugène**
Encadreur professionnel

Année académique 2009/2010

A mes très chers parents
ENGA Sébastien et NGOUNOU Marie
Regardez le fruit de vos sacrifices, de votre affection et de votre amour.

Remerciements

Je remercie tout d'abord le Seigneur tout puissant de m'avoir guidé et de m'avoir donné la force nécessaire d'accomplir ce travail.

J'adresse mes sincères remerciements à M. **AYIMDJI Armel** pour tout le soutien qu'il m'a apporté pour la rédaction de ce document.

Je remercie M. **FONGUEM Eugène** pour sa disponibilité à me permettre de mieux m'intégrer dans l'équipe informatique, de m'avoir conduit à la recherche du terme de notre mémoire au sein du groupe BOLLORE, et de m'avoir prodigué des conseils et recommandation tout au long de ce travail.

Je remercie mes frères **Roger YOSSA**, **Francis TOUKO**, **Cyril MBODA**, **Cédric GAFFI** et mes sœurs **Frédie Carole FANGUEN**, **Judicaëlle LEUMENI**, **Aude-Ines NAMENI**, **Raïssa KAMENI**, **Ydavid Emma FANGUEN** pour leur interminable confiance et leur soutien moral q'ils ont su me prodiguer malgré les multiples difficultés.

Je remercie la famille **YOSSA** à Mélong pour son soutien nuits et jours.

Je tiens à remercier M. **TEUTIO Marcel** pour m'avoir permis de m'intégrer dans le milieu professionnel, pour m'avoir apporté des critiques, du savoir faire et son expertise dans le domaine informatique.

Je remercie M. **AKOA AMBASSA** pour ses conseils, son soutien moral et la relecture de ce mémoire.

Je remercie **FEUTO NJONKO Paul Brillant**, doctorant, pour ses critiques et conseils qui m'ont permis d'avancer dans ce travail.

Je remercie tous mes amis, toute la promotion 2010 de LICENCE pour leur encouragement et soutien inconditionnel pendant les moments difficiles.

Je ne saurais donner la liste exhaustive, je remercie de fond du cœur ceux qui, de près ou de loin, ont contribué et ont participé à la réalisation de ce document.

Avant propos

L'Institut Universitaire de Technologie de Douala (IUT de Douala) est un établissement professionnel de l'Université de Douala né de la réforme universitaire du 19 Janvier 1993 suivant l'arrêté N° 008/CAB/PR.

L'IUT de Douala a pour principale vocation de dispenser en formation initiale un enseignement dans les domaines industriels et commerciaux. A ce titre, il fournit aux entreprises et administrations des prestations de recherche appliquées, de service ou de formation professionnelle correspondant à ses activités, de préparer l'insertion des étudiants dans les milieux professionnels.

L'IUT de Douala forme en deux ans les étudiants qui par la suite obtiennent un diplôme Universitaire de Technologie (DUT) dans les filières de :

- Génie Industriel
- Génie Informatique
- Génie Civil
- Gestion des Entreprises et des Administrations
- Techniques de Commercialisation
- Génie des Télécommunications et Réseaux (non fonctionnel)
- Biologie Appliquée (non fonctionnel)

L'IUT de Douala prépare aussi les candidats au Brevet de Technicien Supérieur (BTS). Il dispose dans ce cycle des filières suivantes :

- Comptabilité et Gestion des Entreprises
- Informatique de gestion
- Secrétariat de Direction Bilingue
- Gestion commerciale et marketing

L'IUT de Douala forme également les étudiants en cycle de licence de technologie dans les filières :

- Génie logistique et transport (GLT),
- Génie informatique (GI),

- Gestion appliquée aux petites et moyennes organisations (GAPMO),
- Génie mécanique et productique (GMP),
- Génie industriel et maintenance (GIM),
- Génie électrique et informatique industrielle (GEII),
- Génie thermique et énergie.

L'IUT forme également dans les filières de professionnalisations qui comprennent :

- Génie Industriel
- Gestion des Entreprises et des Administrations
- Informatique
- Secrétariat Bureautique
- Techniques de Commercialisation

Afin d'obtenir son diplôme, chaque étudiant doit effectuer en fin de formation un stage académique d'une durée de deux à quatre mois à la fin duquel un rapport devra être rédigé et soutenu devant un jury. Telle est la raison du présent document sur la ***Gestion de la Cvthèque des demandes d'emploi au sein du Groupe BOLLORE*** après quatre mois que nous avons passé au département informatique de cette société.

L'objectif de ce stage est de nous permettre de mettre en pratique les connaissances acquises au cours de notre formation et aussi de nous adapter à l'environnement professionnel.

Résumé

La gestion des systèmes d'information est un artefact important dans la mise en œuvre des applications logicielles. A cette ère de la technologie avancée, le groupe Bolloré Cameroun doit mettre un terme aux techniques artisanes du monde informatique de la gestion de son information, notamment les demandes d'emploi en son sein. Ce mémoire traite du développement et du déploiement dans un environnement réel, d'une application pour la gestion des Curriculum Vitae au sin du Groupe Bolloré Cameroun. Nous avons adopté, suivant les standards de l'ingénierie logicielle, les étapes indispensables à la construction de logiciel en partant d'une méthode de modélisation UP dont le langage UML s'y greffe pour représenter de façon graphique et textuelle les entités du système. Ensuite du Model Driven Architecture (MDA), nous avons généré le code source complet à partir d'un modèle graphique.

Mots clés : Système d'information, Applications logicielles, Ingénierie logicielle, méthode de modélisation, UP, UML, langage de modélisation.

Abstract

The management of information system is an important artefact in a software application deployment. Nowadays, at the speedily growth of Technology, the Bollore Cameroon Group must end up with the old method of management, precisely its employment offers. This document talks about the development and deployment, in a real environment, of an application of resumes at the Bollore Group Cameroon. We have adopted, following engineering standard, steps in building software, from a modelling method UP which uses UML as modelling language. Then from the Model Driven Architecture (MDA), we have generated complete source code from a graphical model.

Keywords: Information System, software applications, Software engineering, modelling method, UP, UML, Modelling language.

Table des figures

INTRODUCTION GENERALE

Contexte

Les systèmes d'informations (SI) sont actuellement au cœur de l'innovation dans les entreprises. Ils doivent être à l'image des activités de ces entreprises : réactifs, évolutifs, performants. Le système d'information doit non seulement être capable de stocker et d'organiser un grand nombre de données, mais également de les traiter de façon intelligente pour aider à la prise de décision.

Les besoins en gestion des systèmes d'informations deviennent un atout pour toute entreprise émergente (PME ou grande entreprise). Or, l'industrie informatique est récente comparée à l'industrie métallurgique ou à l'industrie automobile.

Problématique

La place prépondérante des systèmes d'information au sein de l'activité économique la contraint à faire preuve de maturité, prise entre les attentes grandissantes des utilisateurs et les évolutions fulgurantes des technologies notamment dans la production (processeurs, mémoires vives, …) sur lesquelles elle repose. Pour parvenir à cette fin, les méthodes d'analyse, de conception et de déploiement doivent être mises en œuvre par les experts du domaine informatique en collaboration avec les futurs utilisateurs du système.

Vues les demandes grandissantes d'emploi en son sein, la Direction des Ressources Humaines (DRH) du groupe BOLLORE, afin de mieux gérer les différents Curriculum Vitae (CV) et les demandes d'emploi, doit s'acquérir de cette maturité en adoptant une attitude digne d'un processus industriel et en mettant un terme aux techniques artisanales du monde de l'informatique.

Contribution du mémoire

Face à cette insuffisance et cette méthode artisanale de gestion adoptée par la Direction des Ressources Humaines du groupe Bolloré, nous nous proposons dans ce mémoire de concevoir et de déployer un système qui permettra de gérer de façon efficace et optimale les

demandes d'emploi et le curriculum vitae de ses employés. Nous avons adopté les démarches de la construction logicielle selon les standards de l'IEEE. Notre architecture repose sur une méthode de modélisation couplée au langage de modélisation UML dans sa version 2.4.1.

Organisation du document

Ce document est reparti sur deux grandes parties :

La première partie décrit la société Bolloré et s'entend sur les chapitres :

- Le chapitre I présente le groupe Bolloré et ses différentes filiales.
- Dans le chapitre II, nous avons décrit comment s'est déroulé notre stage pendant cette période au sein du groupe Bolloré.

La deuxième partie est portée sur la conception et le déploiement de notre application de gestion des demandes d'emploi au sein du groupe Bolloré.

- Dans le chapitre I, nous partons du système actuel mis en place parallèlement avec le cahier des charges et nous dégageons les attentes du système ainsi que les modules principaux de notre application.
- Le chapitre II fait l'objet de l'étape des spécifications de notre application. Nous dégagerons les fonctionnalités précises du système, les profils des futurs utilisateurs ainsi que les spécifications détaillées de chaque cas d'utilisation de notre application. Nous produirons ensuite le diagramme des cas d'utilisation.
- Toute application doit être soumise à une analyse de la problématique, c'est l'objet du chapitre III de ce document. Il est question de produire un diagramme des classes en nous basant sur le langage de modélisation UML.
- Le chapitre IV présente les phases de conception et d'implémentation de notre application. Nous avons dégagé les aspects liés à la conception des objets et du système. La phase de l'implémentation présente les codes sources des classes précises de notre application.
- Enfin dans le chapitre V, nous avons procédé à l'étude de déploiement et de mise en production de notre application.

PARTIE I

PARTIE I : PRESENTATION DE L'ENTREPRISE ET DEROULEMENT DU STAGE

CHAPITRE I : PRESENTATION DE L'ENTREPRISE

I INTRODUCTION

SAGA/SDV/SOCOPAO[1] est né du groupement de trois entreprises, appartenants au prestigieux groupe français BOLLORE exerçant dans des secteurs variés. Ce groupement a vu le jour depuis le 01 janvier 2006 et la fusion n'a été effective qu'en septembre 2010. En fait il s'agit d'un groupement plus géographique que juridique car chacune des entreprises garde son statut juridique. Ce groupement est perceptible dans la mise en commun des ressources humaines puisque le personnel des trois entreprises, regroupé dans les sites communs travaille ensemble et utilise le même matériel.

II HISTORIQUE

La création de Société Delmas Vieljeux (SDV) est la résultante d'un long processus et peut être divisé en six étapes :

Tout commence en 1974 par l'installation au Cameroun de DELMAS VIELJEUX S.A.

En 1971, cette dernière fusionne avec les Chargeurs Réunis, une société d'armateurs, et donne donc naissance à la Société Navale et Commerciale DELMAS VIELJEUX (SNCDV).

Quatre années plus tard, un autre changement s'opère et la SNCDV devient la société Camerounaise de Manutention et de Transit (CAMTRANS).

En 1986, le groupe BOLLORE rachète la Société Commerciale d'Armement (SCAC) s'exerçant dans le transport logistique et le transit. Soucieuse de croître son domaine d'activité, le groupe BOLLORE rachète en 1989 la Société DELMAS VIELJEUX qui, à la suite d'une fusion avec la SCAC en 1991, donne naissance à SCAC-DELMAS-VIELJEUX (SDV). En 1993, la Société DELMAS Cameroun devient SCAC DELMAS VIELJEUX CAMEROUN.

Aujourd'hui cette société s'étend peu à peu sur le territoire national.

[1] Filiales du groupe BOLLORE au Cameroun

III SITUATION GEOGRAPHIQUE

Le siège social du groupe BOLLORE se trouve à Douala au lieu dit « zone portuaire Vallée Kotoko » entouré des entreprises telles GEODIS, Le Méridien, Panalpina, Maersk…Elle abrite ses services administratifs dans deux bâtiments situés face à face.

IV ACTIVITES ET EXTENSIONS

Les activités du groupe BOLLORE comprennent le Transit, étendu à l'import et l'export des colis par voie maritime, aérienne et routière. Cette activité s'étend sur les opérations de :

- ✓ Manutention qui est une opération d'embarquement ou de débarquement des colis et toutes les opérations s'y rattachant.
- ✓ Groupage maritime et ferroviaire qui consiste à mettre ensemble les colis de deux ou plusieurs clients dans un même conteneur vers une même direction.
- ✓ Logistique, c'est un ensemble des activités intéressant les ravitaillements, l'entretien, le transport.
- ✓ Acconage qui consiste à l'arrimage et au désarrimage des marchandises, de l'empotage, le dépotage des conteneurs, le chargement et le déchargement des navires.
- ✓ Magasinage, ici il s'agit de la mise sous entrepôt des marchandises.
- ✓ Shipping.

Elle exerce également des activités secondaires telles que l'agence de voyage qui réserve et vend des billets d'avion des compagnies aériennes pour lesquelles elle a reçu l'agrément.

Le groupe BOLLORE possède en plus quatre filiales : à Kribi, à Ngaoundéré, à Garoua et à Yaoundé.

V LES MOYENS

Il s'agit des ressources utilisées par l'entreprise pour atteindre son but et ses objectifs. Ils sont de plusieurs types :

- Les moyens humains

Le groupe BOLLORE comporte plus de 1000 employés répartis ainsi qui suit

Catégories socioprofessionnelles	Cadres Expatriés	Cadres Locaux	Agents de maîtrise	Agent d'exécution	TOTAL
Féminin	26	14	52	150	242
Masculin	61	25	152	550	788

Tableau 1 : Répartition personnels BOLLORE

- les moyens financiers

Ils sont constitués par le chiffre d'affaire provenant des prestations offertes, des capitaux propres et des capitaux étrangers utilisés par l'entreprise pour financer ses activités.

SDV détient un capital de 1 327 190 000 Frs CFA reparti entre les actionnaires français et locaux.

- Les moyens matériels

Ils représentent l'ensemble des équipements disponibles permettant les missions que l'entreprise s'est assignée :

Immobilisations incorporelles : licence d'exploitation, logiciels de gestion (SYDONIA, IRIS, SPOT, DELTA PAIE).

Immobilisations corporelles : mobilier de bureau (fauteuils, armoires,…), du matériel informatique, du matériel de transport, du parc automobile avec plus de 672 véhicules en engins.

VI ORGANISATION ET STRUCTURE GENERALE DE GROUPE BOLLORE

6.1 Organes de gestion

Le groupe Bolloré est un ensemble de Sociétés Anonymes administrée chacune de deux organes : Un Conseil d'Administration et une Direction Générale.

- Le conseil d'administration

Il est constitué de 08 membres dont 6 expatriés et 02 locaux. Il est dirigé par un Président du Conseil d'Administration (PCA). Les membres du Conseil d'Administration sont désignés par l'Assemblée Générale et sont révocables à tout moment.

Le Conseil d'Administration est investi des pouvoirs les plus étendus pour agir en toutes circonstances au nom de la société ; il est exercé dans la limite de l'objet social et sous réserve de ceux expressément attribués par la loi aux assemblées d'actionnaires sans que les dispositions des statuts ou de l'Assemblée Générale limitant ses pouvoirs puissent être opposés aux tiers. En outre, le Conseil d'Administration nomme parmi ou en dehors de ces membres un Directeur Général.

- La direction générale

Il existe une unique Direction Générale pour SAGA/SDV/SOCOPAO qui se compose comme suit :

Un Directeur Général (DG) nommé par le conseil d'administration pour une durée fixée librement par ledit conseil. Le DG gère, représente la société il définit les objectifs, fixe les différentes orientations et prend les décisions engageant l'entreprise

Un directeur général adjoint (DGA) nommé par le conseil d'administration sur proposition du DG ou du PCA. Il peut être déposé à tout moment par le conseil d'administration. Il assiste le directeur général dans l'exercice de ses fonctions et le remplace en cas d'absence.

6.2 Structure générale

Compte tenu de l'ampleur de ses fonctions, le groupe Bolloré s'est subdivisé en plusieurs directions :

- La division administrative et financière qui comprend des départements suivants : Comptabilité, Juridique et Contentieux, Informatique, Contrôle de Gestion, le contrôle débours maritime, le recouvrement.
- La direction des ressources humaines
- La direction du transit qui comprend les départements suivants : Export, Import, la direction GIE Atelier.

6.3 Organigramme

La structure organisationnelle du groupe Bolloré est du type Staff and Line. L'organigramme qui en découle obéit à la fois aux principes d'unités de commandement et au rôle de conseil et assistance. S'agissant de principe d'unité de commandement, il permet d'éviter les conflits d'autorité, dans la mesure où, chaque employé dépend d'un seul chef. Quand au rôle de conseil et d'assistance, il est assuré par l'assistance du Directeur Général qui joue un rôle de conseil sans pouvoirs de décision dans la gestion.

6.4 Le Département Informatique

Il s'occupe de l'installation, de la maintenance, de l'administration des réseaux, de l'administration et de la gestion des bases de données.

- ✓ Objectifs

Le service informatique du groupe Bolloré a pour principal objectifs de :

Garantir le bon fonctionnement et la sécurité des systèmes et réseaux informatiques.

Assurer la maintenance des services aux utilisateurs et l'administration des ressources interactives communes.

Offrir l'assistance technique nécessaire à la bonne utilisation des ressources aux utilisateurs.

Assurer un support technique aux autres services de la société.

Assurer le support informatique des projets expérimentaux.

Contribuer à la conception et à la mise en œuvre des systèmes et logiciels destinés à acquérir et à traiter en temps réel les données issues des détecteurs assemblés pour les expériences.

- ✓ Organisation

Le département informatique est hiérarchisé. A sa tête se trouve un chef de département, suivi de son adjoint et de la secrétaire. A la suite viennent les différentes cellules composées des responsables de cellules et des techniques. Voici son organigramme :

- Une cellule administrative

Elle est formée du chef de service informatique qui est entouré de son adjoint, d'une secrétaire, des chauffeurs chargés du déplacement du personnel dans les différents sites pour dépannage et du magasinier qui gère le stock des consommables.

- Une cellule Réseau

Elle s'occupe de la configuration en entrée de tout le réseau du groupe Bolloré de la configuration des équipements réseau tels que routeurs, switches, firewalls, onduleurs, modems etc. elle s'occupe également de l'interconnexion des différents sites et correspondants par un Réseau Privé Virtuel (Virtual Private Network[2]), fibre optique, VSAT[3] et bien d'autres moyens de communications. Elle gère également la VOIP[4] (Voice Over IP)

- Une cellule Applicatif

Elle s'occupe du développement des applications. Ce développement est fait pour les systèmes Microsoft Windows et pour les systèmes AS/400. l'AS/400 est un système qui utilisé dans les réseaux pour la fiabilité d'information et la précision dans le travail. Dans ce système, les terminaux sont utilisés comme clients pour accéder au système serveur.

Au sein de cette cellule, il existe aussi une autre sous cellule la Hotline.

La Hotline est un standard téléphonique qui permet de recevoir les problèmes des utilisateurs utilisant ainsi deux canaux : Ligne téléphone ou par mail. Ces problèmes sont par la suite analysés, et résolus. En cas de gros problème non résolu par la Hotline, celle-ci fait appel à la cellule la plus compétente.

- Une cellule micro

[2] Virtual Private Network (VPN) est un réseau qui interconnecte plusieurs sites, locaux ou distants et qui leur permet de communiquer dans un canal sécurisé.

[3] VSAT (Very Small Aperture Terminal) est une technique de communication par satellite bidirectionnelle qui utilise des antennes paraboliques dont le diamètre est inférieur à 3 mètres.

[4] VOIP (Voice Over IP) est une technologie qui permet de transporter de la voix sur le réseau Internet

Elle s'occupe de la configuration du réseau en interne. Elle gère le déploiement des PCs, des clients légers (WYSE), des imprimantes. Elle s'occupe également de l'administration serveur tels que contrôleurs de domaine, serveur d'impression, serveur TSE (Terminal Server Emulation) pour les WYSES, serveurs de messagerie (Lotus Domino, Microsoft Outlook) ; il s'occupe également de la gestion des comptes utilisateurs, de l'administration du réseau, des interventions à distance auprès des utilisateurs à travers des logiciels de télé intervention (PcAnywhere, Remote client, Dameware etc), des prises réseaux et des baies de brassages.

CHAPITRE II : DEROULEMENT DU STAGE

I INTRODUCTION

Notre stage au sein du Groupe Bolloré Africa Logistics s'est déroulé sur une période de 04 mois du 12 juillet au 12 novembre 2010. Au cours de cette période nous avons pu, sans peine majeure, nous habituer aux petits rouages de l'entreprise. L'assiduité ainsi que la collaboration du personnel de l'entreprise nous a permis davantage de bien appréhender et de mieux nous intégrer dans le milieu professionnel.

Dans cette partie de notre document, il est question pour nous de décrire les différentes tâches que nous avions au quotidien ainsi que les difficultés que nous avons rencontrées pendant cette période de stage.

II ACTIVITES QUOTIDIENNES

Dès notre arrivée au sein du groupe, le département informatique était en plein plongé dans un projet qui consistait en la migration du système de messagerie interne de l'entreprise. Il était question de migrer toutes les boîtes aux lettres de ses près de 1000 employés, de l'ancien système de messagerie Lotus Domino (côté serveur) / Lotus Notes (côté client) vers le nouveau système de messagerie actuel Microsoft Office Exchange (côte serveur) / Microsoft Office Outlook (côté client), sur une plate forme Windows Server 2003. Ce projet de migration était une décision du groupe Bolloré afin de répondre à certains problèmes techniques de compatibilité. A cette époque, le système de messagerie Lotus dont le fabricant IBM, concurrent du géant du logiciel Microsoft, tournait sur les plates formes Windows. Des incompatibilités étaient perceptibles à chaque niveau d'utilisation.

Il était question par la suite de former tous les utilisateurs dont les boîtes aux lettres avaient déjà été migrées, en l'utilisation du nouveau logiciel de messagerie (Outlook). Les formations ont effectivement débuté le 26 juillet. Elle était organisée en petit groupe de 10 utilisateurs en raison de 02 groupes par jour de mardi à jeudi, de 10h à 12h et de 15h30 à 17h30. 02 équipes de formateurs constituées des stagiaires se partageaient les formations en longueur de journée. Les utilisateurs qui étaient formés au cours de la semaine voyaient leurs boîtes de messagerie

migrée le jeudi soir. Les journées de vendredi et de lundi étaient réservées pour assister ceux qui avaient de la peine à s'adapter à cette nouvelle messagerie.

II PRINCIPALE DIFFICULTE

La principale difficulté que nous avons rencontrée au cours de notre séjour au sein du groupe Bolloré était notre adaptation à l'ambiance du département informatique ; nous avons toujours été habitués à travailler dans un environnement calme pendant les cours théoriques. Ce qui n'était pas le cas dans ce département à cause de la convergence accrue des appels des utilisateurs coincés en cours de travail dans cette nouvelle messagerie. Il serait préférable que la cellule Hotline soit délocalisée des autres cellules afin d'isoler les mouvements des appels des utilisateurs.

Conclusion partielle

Actuellement tout le personnel du groupe Bolloré filiale de Douala a déjà été migré, les utilisateurs des autres filiales des autres villes sont en cours de programmation.

Avec les techniques et les méthodes de l'analyse et de la mise en production des systèmes de l'information, acquises au cours de notre formation, la migration et la formation sur cette messagerie étaient très limitées et non adaptées à nos attentes. Il nous a semblé judicieux de choisir un autre domaine de difficulté auquel était confronté le Groupe Bolloré et dans lequel nous devions mettre en pratique nos connaissances et notre expérience afin de répondre à un problème réel.

Après plusieurs entretiens avec notre encadreur professionnel, nous sommes allés à la rencontre de la direction des ressources humaines. Plusieurs difficultés se faisaient ressentir parmi lesquelles la gestion des demandes d'emploi au sein du groupe.

Ce thème a attiré notre attention et nous y avons accentué notre travail tout le long de notre séjour. Mais avant de nous lancer dans le vif du sujet, il est judicieux de rappeler le processus, les démarches a adopter dans la réalisation d'un système informatique.

PARTIE II : LA DEMARCHE DU TRAVAIL

CHAPITRE I : LE PROCESSUS UNIFIE - PU

> *"Le processus unifié est un **processus de développement logiciel**, c'est-à-dire qu'il regroupe les activités à mener pour transformer les besoins d'un utilisateur en système logiciel" (Jacobson, Booch, Rumbaugh 1999)*

La conception d'un logiciel met en œuvre un ensemble d'activités qui a partir d'une demande d'informatisation de processus permettent la conception, la programmation et la mise sur pied d'un logiciel jusqu'à sa livraison au demandeur.
Une démarche de construction logicielle doit être une norme fondée sur un aspect méthodique et suivant des étapes bien définies dans le temps en fonction du choix de la méthode.

Le processus unifié est un processus de développement logiciel itératif, centré sur l'architecture, piloté par les cas d'utilisation et orienté vers la diminution des risques.

I - NOUVELLE APPROCHE DU PROCESSUS DE DEVELOPPEMENT

Dans une démarche traditionnelle (Exemple de la méthode Merise), le processus de développement était caractérisé par :

• Un processus de type séquentiel : développement organisé en phases qui regroupent des étapes, elles mêmes décomposées en tâche.

• Les niveaux de découpage coïncident : la fin d'une phase correspond à la conclusion de ses étapes, qui elles mêmes se terminent avec l'accomplissement des tâches qui les composent.

Dans une approche objet tout change :

• Le processus est de type itératif ;

• Les découpages ne coïncident pas : les activités (taches, phases, étapes, etc…) se déroulent dans plusieurs dimensions.

UP (Unified Process) est une méthode générique de développement de logiciel.

Générique signifie qu'il est nécessaire d'adapter UP au contexte du projet, de l'équipe, du domaine et/ou de l'organisation (exemple: R.UP ou X.UP). C'est, entre parenthèses, plus ou moins vrai pour toute méthode, qu'elle se définisse elle-même comme générique ou pas.

Il existe donc un certain nombre de méthodes issues d'UP.

Le processus unifié est un processus de développement logiciel : il regroupe les activités à mener pour transformer les besoins d'un utilisateur en système logiciel.

Caractéristiques essentielles du processus unifié :

- Le processus unifié est à base de composants,
- Le processus unifié utilise le langage UML (ensemble d'outils et de diagramme),
- Le processus unifié est piloté par les cas d'utilisation,
- Centré sur l'architecture,
- Itératif et incrémental.

II- LE PROCESSUS UNIFIE EST PILOTE PAR LES CAS D'UTILISATION

2.1- Présentation

L'objectif principal d'un système logiciel est de rendre service à ses utilisateurs ; il faut par conséquent bien comprendre les désirs et les besoins des futurs utilisateurs. Le processus de développement sera donc centré sur l'utilisateur. Le terme utilisateur ne désigne pas seulement les utilisateurs humains mais également les autres systèmes. L'utilisateur représente donc une personne ou une chose dialoguant avec le système en cours de développement.

Ce type d'interaction est appelé cas d'utilisation.

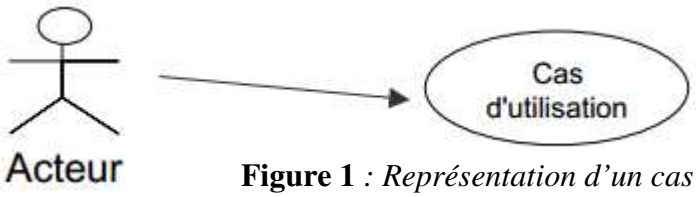


Figure 1 *: Représentation d'un cas d'utilisation*

Les cas d'utilisation font apparaître les besoins fonctionnels et leur ensemble constitue le modèle des cas d'utilisation qui décrit les fonctionnalités complètes du système.

2.2- Exemple : Guichet de banque.

Figure 2*: Exemple de cas d'utilisation : Guichet de banque*

On va se demander, en premier, quels sont les utilisateurs du système (pas forcément des utilisateurs physiques, mais plutôt des rôles). Ici, l'employé est aussi un client de la banque. On a donc une personne physique pour deux rôles. Nous ne sommes pas dans une approche de type fonctionnelle mais une approche pilotée par des cas d'utilisation.

2.3- Stratégie des cas d'utilisation.

Que doit pouvoir faire le système pour chaque utilisateur ?

Les cas d'utilisation ne sont pas un simple outil de spécification des besoins du système. Ils vont complètement guider le processus de développement à travers l'utilisation de modèles basés sur l'utilisation du langage UML.

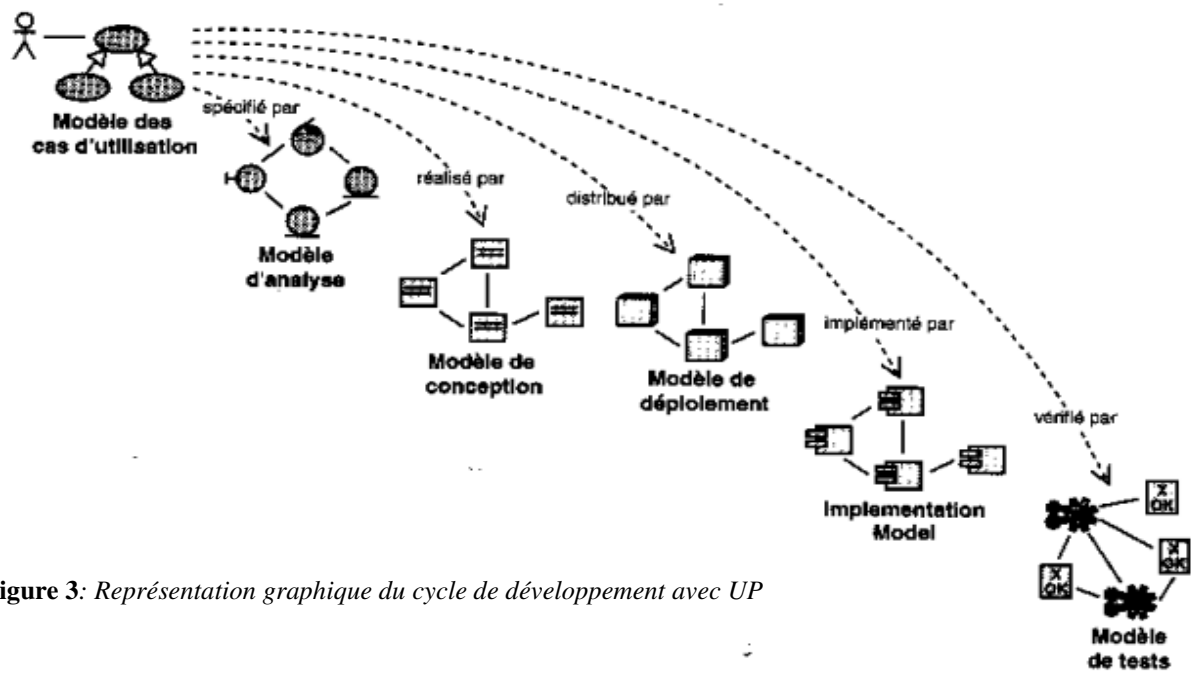


Figure 3*: Représentation graphique du cycle de développement avec UP*

- A partir du modèle des cas d'utilisation, les développeurs créent une série de modèles de conception et d'implémentation réalisant les cas d'utilisation.
- Chacun des modèles successifs est ensuite révisé pour en contrôler la conformité par rapport au modèle des cas d'utilisation.
- Enfin, les testeurs testent l'implémentation pour s'assurer que les composants du modèle d'implémentation mettent correctement en œuvre les cas d'utilisation.

Les cas d'utilisation garantissent la cohérence du processus de développement du système. S'il est vrai que les cas d'utilisation guident le processus de développement, ils ne

sont pas sélectionnés de façon isolée, mais doivent absolument être développés "en tandem" avec l'architecture du système.

III LE PROCESSUS UNIFIE EST CENTRE SUR L'ARCHITECTURE.

Dès le démarrage du processus, on aura une vue sur l'architecture à mettre en place. L'architecture d'un système logiciel peut être décrite comme les différentes vues du système qui doit être construit. L'architecture logicielle équivaut aux aspects statiques et dynamiques les plus significatifs du système. L'architecture émerge des besoins de l'entreprise, tels qu'ils sont exprimés par les utilisateurs et autres intervenants et tels qu'ils sont reflétés par les cas d'utilisation.

Elle subit également l'influence d'autres facteurs :

- la plate-forme sur laquelle devra s'exécuter le système ;
- les briques de bases réutilisables disponibles pour le développement ;
- les considérations de déploiement, les systèmes existants et les besoins non fonctionnels (performance, fiabilité,..)

3.1 Liens entre cas d'utilisation et architecture ?

Tout produit est à la fois forme et fonction. Les cas d'utilisation doivent une fois réalisés, trouver leur place dans l'architecture. L'architecture doit prévoir la réalisation de tous les cas d'utilisation. L'architecture et les cas d'utilisation doivent évoluer de façon concomitante.

1.1.1 Marche à suivre :

- L'architecte crée une ébauche grossière de l'architecture, en partant de l'aspect qui n'est pas propre aux cas d'utilisation (plate forme..). Bien que cette partie de l'architecture soit indépendante des cas d'utilisation. L'architecte doit avoir une compréhension globale de ceux ci avant d'en esquisser l'architecture.
- Il travaille ensuite, sur un sous ensemble des cas d'utilisations identifiés, ceux qui représentent les fonctions essentielles du système en cours de développement.
- L'architecture se dévoile peu à peu, au rythme de la spécification et de la maturation des cas d'utilisation, qui favorisent, à leur tour, le développement d'un nombre croissant de cas d'utilisation.

Ce processus se poursuit jusqu'à ce que l'architecture soit jugée stable.

IV LE PROCESSUS UNIFIE EST ITERATIF ET INCREMENTAL.

Le développement d'un produit logiciel destiné à la commercialisation est une vaste entreprise qui peut s'étendre sur plusieurs mois. On ne va pas tout développer d'un coup. On peut découper le travail en plusieurs parties qui sont autant de mini projets. Chacun d'entre eux représentant une itération qui donne lieu à un incrément.

Une itération désigne la succession des étapes de l'enchaînement d'activités, tandis qu'un incrément correspond à une avancée dans les différents stades de développement.

Le choix de ce qui doit être implémenté au cours d'une itération repose sur deux facteurs :

- Une itération prend en compte un certain nombre de cas d'utilisation qui ensemble, améliorent l'utilisabilité du produit à un certain stade de développement.
- L'itération traite en priorité les risques majeurs.

Un incrément constitue souvent un additif.

A chaque itération, les développeurs identifient et spécifient les cas d'utilisations pertinents, créent une conception en se laissant guider par l'architecture choisie, implémentent cette conception sous forme de composants et vérifie que ceux ci sont conformes aux cas d'utilisation. Dés qu'une itération répond aux objectifs fixés le développement passe à l'itération suivante.

Pour rentabiliser le développement il faut sélectionner les itérations nécessaires pour atteindre les objectifs du projet. Ces itérations devront se succéder dans un ordre logique.

Un projet réussi suivra un déroulement direct, établi dés le début par les développeurs et dont ils ne s'éloigneront que de façon très marginale. L'élimination des problèmes imprévus fait partie des objectifs de réduction des risques.

V AVANTAGES D'UN PROCESSUS ITERATIF CONTROLE.

- Permet de limiter les coûts, en termes de risques, aux strictes dépenses liées à une itération.

- Permet de limiter les risques de retard de mise sur le marché du produit développé (identification des problèmes dès les premiers stades de développement et non en phase de test comme avec l'approche « classique »).
- Permet d'accélérer le rythme de développement grâce à des objectifs clairs et à court terme.
- Permet de prendre en compte le fait que les besoins des utilisateurs et les exigences correspondantes ne peuvent être intégralement définis à l'avance et se dégagent peu à peu des itérations successives

L'architecture fournit la structure qui servira de cadre au travail effectué au cours des itérations, tandis que les cas d'utilisation définissent les objectifs et orientent le travail de chaque itération. Il ne faut donc pas mésestimer l'un des trois concepts.

VI LE CYCLE DE VIE DU PROCESSUS UNIFIE.

Le processus unifié répète un certain nombre de fois une série de cycles.
Tout cycle se conclut par la livraison d'une version du produit aux clients et s'articule en 4 phases : création, élaboration, construction et transition, chacune d'entre elles se subdivisant à son tour en itérations.

Chaque cycle se traduit par une nouvelle version du système. Ce produit se compose d'un corps de code source réparti sur plusieurs composants pouvant être compilés et exécutés et s'accompagne de manuels et de produits associés. Pour mener efficacement le cycle, les développeurs ont besoin de construire toutes les représentations du produit logiciel :

Modèle des cas d'utilisation	Expose les cas d'utilisation et leurs relations avec les utilisateurs
Modèle d'analyse	Détaille les cas d'utilisation et procède à une première répartition du comportement du système entre divers objets
Modèle de conception	Définit la structure statique du système sous forme de sous système, classes et interfaces ; Définit les cas d'utilisation réalisés sous forme de collaborations entre les sous systèmes les classes et les interfaces
Modèle d'implémentation	Intègre les composants (code source) et la correspondance entre les classes et les composants
Modèle de déploiement	Définit les nœuds physiques des ordinateurs et l'affectation de ces composants sur ces nœuds.

Modèle de test	Décrit les cas de test vérifiant les cas d'utilisation
Représentation de l'architecture	Description de l'architecture

Tableau I : Cycle de vie du développement avec UP

Tous ces modèles sont liés. Ensemble, ils représentent le système comme un tout. Les éléments de chacun des modèles présentent des dépendances de traçabilité ; ce qui facilite la compréhension et les modifications ultérieures.

5.1 Présentation du cycle de vie de UP.

Phase	Description et Enchaînement d'activités
Phase de création	Traduit une idée en vision de produit fini et présente une étude de rentabilité pour ce produit - Que va faire le système pour les utilisateurs ? - A quoi peut ressembler l'architecture d'un tel système ? - Quels sont l'organisation et les coûts du développement de ce produit ? On fait apparaître les principaux cas d'utilisation. L'architecture est provisoire, identification des risques majeurs et planification de la phase d'élaboration.
Phase d'élaboration	Permet de préciser la plupart des cas d'utilisation et de concevoir l'architecture du système. L'architecture doit être exprimée sous forme de vue de chacun des modèles. Emergence d'une architecture de référence. A l'issue de cette phase, le chef de projet doit être en mesure de prévoir les activités et d'estimer les ressources nécessaires à l'achèvement du projet.
Phase de construction	Moment où l'on construit le produit. L'architecture de référence se métamorphose en produit complet, elle est maintenant stable. Le produit contient tous les cas d'utilisation que les chefs de projet, en accord avec les utilisateurs ont décidé de mettre au point pour cette version. Celle ci doit encore avoir des anomalies qui peuvent être en partie résolue lors de la phase de transition.
Phase de transition	Le produit est en version bêta. Un groupe d'utilisateurs essaye le produit et détecte les anomalies et défauts. Cette phase suppose des activités comme la fabrication, la formation des utilisateurs clients, la mise en œuvre d'un service d'assistance et la correction des anomalies constatées.(où le report de leur correction à la version

	suivante)

APPROCHE DU LANGAGE UML

UML est la forme contractée de "*Unified Modeling Language*" qui peut se traduire par *Langage Unifié de Modélisation*. Il fournit les fondements pour spécifier, construire, visualiser et décrire les artefacts d'un système logiciel. Pour ce fait il se base sur une sémantique précise et sur une notation graphique expressive.

UML est un langage de modélisation objet et en tant que tel, il facilite l'expression et la communication de modèles en fournissant un ensemble de symboles (la notation) et de règles qui régissent l'assemblage de ces symboles (la syntaxe et la sémantique).

UML naît en janvier 1997 dans sa version 1.0 et a beaucoup évolué avec ses diverses fonctionnalités. Aujourd'hui il est à sa version 2.4.1 et apporte de nombreux changements qu'il s'agisse de correction ou d'ajout. Des incohérences entre les concepts présentés dans les documents traitant de la sémantique et de la notation ont été levées ; des définitions et des explications ont été clarifiées.

Un modèle est une simplification de la réalité qui permet de mieux comprendre le système à développer. La modélisation n'est pas un problème à solution unique. Bien souvent, le même problème analysé par des personnes différentes conduit à des modèles différents.

Les utilisateurs regardent et manipulent les modèles au moyen de vue graphiques. A chaque vue correspondent un ou plusieurs diagrammes.

UML dans sa version 2.4.1 définit 13 types de diagrammes classés en vues statiques et dynamiques.

Vue statique (diagramme statique) c'est-à-dire représenter le système dans son état physique

- Diagramme d'objet
- Diagramme de classe
- Diagramme de composants
- Diagramme de déploiement
- Diagramme de paquetage
- Diagramme de structure composite

Vue dynamique (diagramme dynamique) montrant le fonctionnement du système

- Diagramme de séquence
- Diagramme de communication (diagramme de collaboration en UML1.0)

- Diagramme de temps
- Diagramme global d'interaction

Diagrammes comportementaux

- Diagramme de cas d'utilisation
- Diagramme d'activité
- Diagramme d'états – transitions

➢ *Modélisation des cas d'utilisation*

Un cas d'utilisation spécifie une séquence d'interactions, avec ses variantes, entre les acteurs et le système, produisant un résultat satisfaisant pour un acteur particulier. On peut donc considérer un cas d'utilisation comme une abstraction de plusieurs chemins d'exécution d'une utilisation du système. La description d'un système est basée sur une instance de cas d'utilisation appelée Scénario. UML propose de décrire un cas d'utilisation en utilisant un bonhomme lié à une fonctionnalité du système par les flèches.

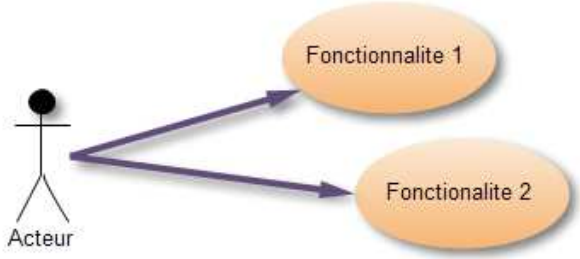


Figure 4*: Description de cas d'utilisation*

➢ *Modélisation des objets*

La modélisation objet consiste à créer une représentation abstraites, sous forme d'objets, d'entités ayant une existence matérielle (arbre, voitures, maison,…) ou bien virtuelle (sécurité sociale, compte bancaire, …). Un objet est caractérisé par plusieurs notions :

- les attributs : il s'agit des données caractérisant l'objet. Ce sont des variables stockant les informations d'état des objets.
- Les méthodes : les méthodes d'un objet caractérisent son comportement c'est-à-dire l'ensemble des actions que l'objet est à même de réaliser. Ces opérations permettent de rendre l'objet accessible de l'extérieure (manipulable par les autres objets externes). Les méthodes sont liées étroitement aux attributs car leurs actions peuvent dépendre de la valeur des attributs.

- L'identité : chaque objet possède une identité qui permet de le repérer de façon unique au sein du développement d'un système.

UML propose une manière de représenter un objet de façon graphique, sous forme de rectangle, dans lequel le nom de l'objet est souligné.

Figure 5*: Exemple d'instance de classe (Objet)*

- ***Les attributs d'un objet***

Ils représentent les caractéristiques de l'objet. L'ensemble des valeurs des attributs d'un objet constitue son état. UML propose de représenter les attributs d'un objet et de la valeur associée de la manière suivante

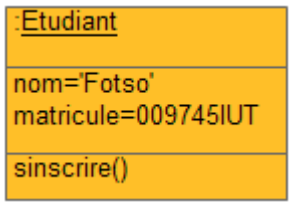


Figure 6*: Une instance d'objet avec ses attributs*

- ***Les liens entre objets***

Dans l'approche objet, les objets ne sont pas des corps inertes isolés. Bien au contraire même s'ils possèdent leurs caractéristiques propres par l'intermédiaire des valeurs de leurs attributs (ce qui constitue ce qu'on appelle état), ils ont la possibilité d'interagir entre eux grâce à leurs méthodes. UML propose de représenter les liens qui existent entre objets grâce à un trait continu.

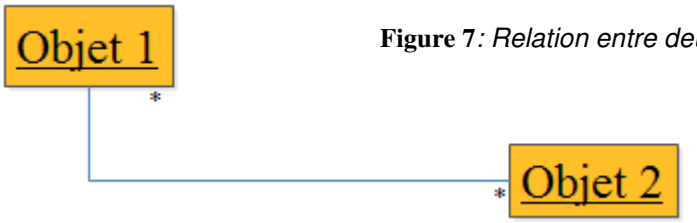


Figure 7*: Relation entre deux objets*

- ***Modélisation des classes***

On appelle classe la structure d'un objet c'est-à-dire la déclaration de l'ensemble des entités qui composeront un objet. Un objet est donc une instance d'une classe c'est-à-dire issu d'une

classe. La classe est alors vue comme un moule qui permet de fabriquer les objets ayant les mêmes attributs et des comportements différents. Une classe est composée de :

- d'attributs : il s'agit des données dont les valeurs représentent l'état des objets.
- Les méthodes : il s'agit des opérations applicables aux objets.

Si on définit la classe *animal*, les objets *oiseau, lion, poisson* seront les instanciations de cette classe. Ils ont chacun les mêmes caractéristiques un *poids*, une *taille*. Ils ont des comportements différents ; pour *se déplacer*, l'oiseau utilise ses ailles, le lion utilise ses pattes et le poisson utilise ses nageoires.

En UML, une classe se représente sous la forme d'un rectangle divisé en trois partitions. Le premier contient le nom donné à la classe (non souligné). Les attributs d'une classe sont définis par un nom, un type dans la seconde section. Dans le troisième compartiment sont répertoriés les comportements.

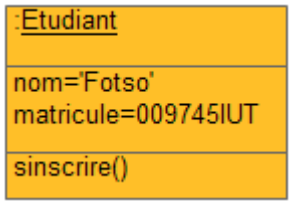


Figure 8*: Une classe complète*

La description d'un système est basée sur la notion d'objet et de classe. Un objet est la représentation abstraite.

PARTIE III : MISE EN PRODUCTION D'UN SYSTEME DE GESTION DES DEMANDES D'EMPLOI AU SEIN DU GROUPE BOLLORE

CHAPITRE I : DEFINITION DE L'EXISTANT ET CAHIER DES CHARGES

Le département des ressources humaines du groupe BOLLORE décide de mettre en œuvre un système informatisé de gestion des demandes d'emplois au sein de la société. A la date de mise en œuvre de ce système, le département des ressources humaines utilisait encore des méthodes artisanales de l'informatique pour satisfaire en partie ses attentes.

I DEFINITION DE L'EXISTANT

Dans cette section, nous définirons la direction des ressources humaines de Bolloré dans sa méthode actuelle de gestion de la banque des données des demandes d'emploi.

1.1 Description

La Direction des Ressources Humaines est confrontée à des difficultés qui mettent en cause les méthodes informatiques actuellement utilisées pour la gestion grandissante des demandes d'emploi. En effet, la DRH dispose de plusieurs chemises permettant de ranger les différents CV qui lui sont adressés. On peut distinguer les chemises portant les catégories de postes : informaticien, comptable, électricien, chauffeur…

Lorsqu'un CV est adressé à l'attention de la DRH, elle le range dans la chemise catégorie correspondante (la catégorie est le poste sollicité par le CV).

Les difficultés de gestion résident à ce stade ; lorsque la DRH souhaite effectuer des recherches portant sur un critère bien précis. Par exemple la liste des CV possédant au moins un diplôme de Baccalauréat ou encore la liste des CV ayant suivi une formation en bureautique informatique. La recherche devient fastidieuse et parfois longue et à la limite même décourageante vue la pile des CV présents dans les bureaux de la Direction des Ressources Humaines.

1.2 Processus de traitement des demandes d'emplois

La DRH a adopté une méthode artisanale dans le processus de gestion de la banque des CV disponibles en son sein. Lorsqu'un poste est vacant, la DRH se réfère d'abord à la

banque des CV en sa possession et y recherche le profil correspondant. Cette recherche est soldée par une probabilité très faible, à cause du fait que le profil recherché peut se trouver dans une catégorie autre que celle qui lui correspond. Pour cette raison, la DRH se trouve obliger de parcourir toutes les chemises bien que les autres ne correspondent pas à la catégorie recherchée.

Les échanges avec la direction des ressources humaines nous a permis d'entrer en possession du cahier des charges décrivant au mieux les besoins et les attentes afin d'améliorer la gestion de ces demandes d'emplois en leur proposant un système souple, cohérent et évolutif.

II CAHIER DES CHARGES

Nous avons reçu, sous un document les attentes de la DRH concernant cette facilité dans la gestion de ses demandes d'emploi.

2.1 Expression des besoins

« *Dans le souci d'informatiser le système de gestion des demandes d'emploi, le département des ressources humaines de SAGA/SDV/SOCOPAO a décidé de mettre en place une application dénommée CVTHEQUE BOLLORE CMR qui permettra d'améliorer la gestion et l'exploitation des demandes d'emploi.*

CVTHEQUE BOLLORE CMR permettra de :

- *Enregistrer et codifier les CV par :*
  - *Type de métier*
  - *Année de réception*
- *Rechercher les CV correspondant aux profils des postes à pouvoir*
- *Consulter, imprimer, supprimer les CV*
- *Effectuer les mises à jour de la base*
- *Obtenir les statistiques*
  - *Des CV enregistrés*
  - *Des CV sélectionnés*
  - *Des CV retenus*
  - *L'indice de satisfaction*
  - *Les CV reçus par année*
  - *Le nombre de CV par type de métier*

Les intervenants

Maîtrise d'ouvrage : Département des Ressources Humaines

Maîtrise d'œuvre : Service Informatique »

2.2 Réception du cahier des charges

La réception du cahier des charges a marqué notre engagement à apporter une solution informatique employant des méthodes modernes d'analyses et de conception de système de l'information, afin de permettre à la DRH d'abandonner la méthode artisanale et de se mettre en phase de la mondialisation et du modernisme que nous apportent les solutions fiables de l'informatique.

Nous traiterons dans la suite du document en suivant les étapes indispensables à la mise en œuvre des applications logicielles, partant de l'expression des besoins à la mise en exploitation d'un système informatique dans un environnement de production réelle. Cette démarche nous permettra de nous assurer de la qualité du logiciel suivant les standards de l'ingénierie logicielle.

CHAPITRE II : ETUDE DES SPECIFICATIONS DE L'APPLICATION

Selon les standards de l'ingénierie logicielle, nous dressons ci-dessous le tableau des étapes de réalisation.

Historique des évolutions

Tableau 2 : Historique des évolutions du DS

Version	*Date*	*Auteur*	*Section modifiée*	*Description*
1.0	26/07/2010	Gabriel Kant	Aucune	Cette partie a été établie à partir d'éléments initialement rédigés par le Département des Ressources Humaines
1.1	03/08/2010	Gabriel Kant	- Fonctionnalités - Profil utilisateur	

Documents associés

Tableau 3 : Documents fournis par la DRH

Documents	*Référence*
Cahier de charge fourni par Mr DJAGA Patrick	Ressources Humaines Groupe Bolloré

Cette partie du document décrit les spécifications du système à mettre en œuvre dans le cadre de l'informatisation des CV des demandeurs d'emploi au sein du groupe Bolloré Africa Logistics. Il comporte la description des fonctions majeures attendues, les exigences techniques et les cas d'utilisation du point de vue des divers utilisateurs.

I DOMAINE D'APPLICATION

Cette spécification est produite selon les règles de l'ingénierie logicielle, version 5.2 définies par l'Association for Computing Machinery (ACM) et l'Institute of Electronics and Electrical Engineers (IEEE). Les mises à jour doivent être effectuées par les auteurs, ou toute personne du groupe BOLLORE portant un intérêt au projet.

1.1 Interface utilisateur

Notre système sera utilisé par deux groupes d'utilisateurs : utilisateur et administrateurs du système. Chacun devra se loguer avec un compte et un mot de passe. La gestion des droits

d'accès se fait uniquement par les administrateurs. Chaque utilisateur disposera des droits en fonction de son statut au sein du groupe.

1.2 Différentes fonctionnalités de l'application CVTHEQUE

Pour chaque profil de notre application, nous définissons les différentes fonctionnalités de notre système : ce que les utilisateurs finaux peuvent effectuer comme tâche et qui décrivent les attentes de la DRH de Bolloré.

- Niveau utilisateur final (Agent de la DRH)

  Créer son compte -> Fonctionnalité 1

  Enregistrer un CV -> Fonctionnalité 2

  Supprimer un CV -> Fonctionnalité 3

  Rechercher un CV -> Fonctionnalité 4

  Modifier un CV -> Fonctionnalité 5

  Mettre à jour un CV -> Fonctionnalité 6

  Etudier, évaluer, noter un CV -> Fonctionnalité 7

  Visualiser les statistiques -> Fonctionnalité 8

  Consulter un CV -> Fonctionnalité 9

- Niveau Administrateur (Agent du département informatique)

  Activer les comptes utilisateurs -> Fonctionnalité 10

  Gérer l'ensemble du système avec base de données -> Fonctionnalité 11

II PROFIL DES UTILISATEURS

Le niveau de compétence nécessaire pour une utilisation optimale du logiciel CVTHEQUE Bolloré CMR reste somme toute basique. Un type d'interface est à prévoir. Cette interface sera ergonomique et fonctionnelle permettant à l'utilisateur de se connecter et d'interagir avec le système en fonction de son statut.

Nous disposerons au sein de notre application deux types de profils : les utilisateurs simples et les administrateurs.

Chaque profil est composé de :

- *Nom*
- *Prénom*
- *Email*
- *Téléphone*

- *Mot de passe,*

Et utilise le logiciel en fonction des droits d'accès qui lui sont attribués.

III PROFIL D'UN CV

Chaque demande d'emploi adressée à l'attention du département des ressources humaines doit comprendre les informations suivantes :

- Etat civil
  - Nom
  - Prénom
  - Date de naissance
  - Age
  - Sexe
  - Nationalité
  - Situation matrimoniale
  - Adresse email
  - Adresse postale
  - Numéro de téléphone
- Formation de base
  - Diplôme
  - Domaine d'études
- Formations professionnelles et complémentaires
  - Domaine
  - Intitulé formation
- Expériences professionnelles
  - Poste occupé
  - Période
  - Société
  - Attribution/Responsabilité
- Domaine de compétence
- Référence
  - Nom et prénom du référent
  - Adresse du référent
- Motivations

IV SPECIFICATIONS DETAILLEES ET SCENARIOS D'UTILISATION DES CAS PRECIS

Deux types de profil utilisateur sont définis dans cette application : les utilisateurs simples (acteurs principaux) et les administrateurs (qui sont les utilisateurs ayant des droits d'accès supplémentaires et dont la tâche principale réside à la maintenance, l'activation des comptes utilisateurs et la gestion de la base de données).

Les différentes fonctionnalités du système vis-à-vis des utilisateurs sont décrites dans les lignes suivantes.

4.1 Fonctionnalité 1

Titre : Créer son compte

Résumé : L'utilisateur crée son compte d'accès à l'application

Acteur : Utilisateur, administrateur

Pré condition : L'utilisateur est d'abord personnel de l'entreprise

Déclencheur : L'utilisateur souhaite enregistrer ou modifier un CV

Scénario 1.

Le système affiche le message d'accueil

L'utilisateur sélectionne le bouton de création du compte

L'utilisateur saisi les informations de création de son compte

Le système enregistre les données saisies (***Contrainte 1***)

L'administrateur valide le compte

Le système envoie un lien d'activation de compte dans la boîte email de l'utilisateur

L'utilisateur accède à son email et valide son compte en cliquant sur le lien

Le système enregistre l'activation du compte

Contrainte 1 : L'utilisateur doit posséder au moins une adresse email valide.

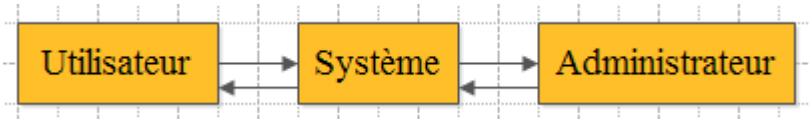


Figure 9: *Interaction de la fonctionnalité 1*

4.2 Fonctionnalité 2

Titre : Enregistrer un CV

Résumé : l'utilisateur enregistre un nouveau CV

Acteur : Utilisateur, Administrateur

Pré condition : L'utilisateur est d'abord employé confirmé de l'entreprise.

Déclencheur : L'utilisateur souhaite enregistrer ou modifier un CV qu'il a entre ses mains.

Scénario 2.

Le système affiche un message d'accueil

L'utilisateur s'authentifie à l'aide de son compte et mot de passe d'accès

Le système valide le compte

Le système affiche un menu en fonction des droits d'accès de l'utilisateur

L'utilisateur sélectionne le bouton d'enregistrement

L'utilisateur saisi les informations concernant le CV a enregistré

Le système valide les données saisies et attribut au CV un numéro d'ordre (***Contrainte 2***)

Le système envoie un mail d'accusé de réception dans la boîte email du demandeur (***Contrainte 3***)

Contrainte 2: Les données saisies doivent être cohérentes

Contrainte 3: Le demande doit fournir au moins une adresse email valide au moment de son enregistrement.

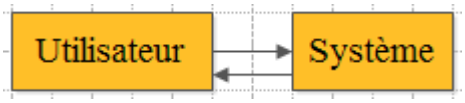


Figure 10*: Interaction de la fonctionnalité 2*

4.3 Fonctionnalité 3

Titre : Rechercher un CV

Résumé : L'utilisateur recherche un CV dans la banque des données.

Acteur : Utilisateur, Administrateur

Pré condition : L'utilisateur est d'abord personnel de l'entreprise

Déclencheur : L'utilisateur souhaite effectuer les opérations sur un CV et de ce fait il doit faire une recherche.

Scénario 3.

Le système affiche un message d'accueil

L'utilisateur s'authentifie à l'aide de son compte et mot de passe d'accès

Le système valide le compte

Le système affiche un menu en fonction des droits d'accès de l'utilisateur

L'utilisateur sélectionne le bouton de recherche

Le système demande le critère de recherche

L'utilisateur saisi le critère

Le système recherche dans la banque des données le/les CV (s) correspondants.

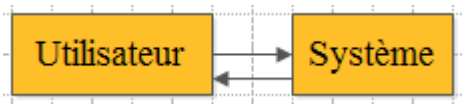


Figure 11*: Interaction de la fonctionnalité 3*

4.4 Fonctionnalité 4

Titre : Modifier un CV

Résumé : L'utilisateur recherche un CV pour y effectuer une modification

Acteur : Utilisateur, Administrateur

Pré condition : L'utilisateur possède un critère de recherche contenant le CV à rechercher

Déclencheur : L'utilisateur souhaite enregistrer ou modifier un CV

Scénario 4.

Le système affiche un message d'accueil

L'utilisateur s'authentifie à l'aide de son compte et mot de passe d'accès

Le système valide le compte

Le système affiche un menu en fonction des droits d'accès de l'utilisateur

L'utilisateur sélectionne le bouton de modification

Le système demande le numéro d'ordre du CV à modifier

L'utilisateur saisi le numéro

Le système valide le numéro

Le système affiche le CV correspondant.

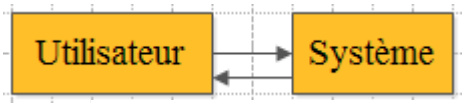


Figure 12*: Interaction de la fonctionnalité 4*

4.5 Fonctionnalité 5

Titre : Consulter un CV

Résumé : consultation d'un CV déjà enregistré.

Acteur : Utilisateur, Administrateur

Pré condition : L'utilisateur est d'abord personnel de l'entreprise

Déclencheur : L'utilisateur souhaite visualiser le contenu d'un CV déjà enregistré.

Scénario 5.

Le système affiche un message d'accueil

L'utilisateur s'authentifie à l'aide de son compte et mot de passe d'accès

Le système valide le compte

Le système affiche un menu en fonction des droits d'accès de l'utilisateur

L'utilisateur sélectionne le bouton de consultation

Le système demande le numéro d'ordre du CV à consulter

L'utilisateur saisi le numéro

Le système valide le numéro

Le système affiche le CV correspondant.

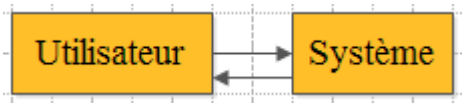


Figure 13*: Interaction de la fonctionnalité 5*

4.6 Fonctionnalité 6

Titre : Supprimer un CV

Résumé : Suppression d'un CV déjà enregistré.

Acteur : utilisateur, Administrateur

Pré condition : L'utilisateur est d'abord personnel de l'entreprise et doit posséder le numéro du CV du dossier à supprimer.

Déclencheur : Le CV ne possède aucun critère de sélection élaboré par le HR.

Scénario 6.

Le système affiche un message d'accueil

L'utilisateur s'authentifie à l'aide de son compte et mot de passe d'accès

Le système valide le compte

Le système affiche un menu en fonction des droits d'accès de l'utilisateur

L'utilisateur sélectionne le bouton de modification

Le système demande le numéro d'ordre du CV à supprimer

L'utilisateur saisi le numéro

Le système valide le numéro

Le système supprimer le CV correspondant.

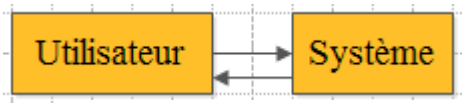


Figure 14*: Interaction de la fonctionnalité 6*

NB : Un CV supprimé peut encore être restauré et uniquement par l'administrateur

4.7 Fonctionnalité 7

Titre : Activer les comptes utilisateurs

Résumé : Activation d'un compte utilisateur pour lui permettre de se connecter au système.

Acteur : Administrateur

Pré condition : L'utilisateur a créé son compte d'accès attends l'activation par l'administrateur.

Déclencheur : Le Ressource Humaine souhaite ajouter un nouveau compte utilisateur.

Scénario 7.

Le système affiche un message d'accueil

L'administrateur s'authentifie à l'aide de son compte et mot de passe d'accès

Le système valide le compte

L'administrateur accède à la liste des comptes créés

L'administrateur attribue à chaque compte des droits d'utilisation du système

L'administrateur active le compte (***Contrainte 4***)

Le système valide la saisie de l'administrateur

Contrainte 4 : L'administrateur doit s'assurer que le compte à activer est bien celui d'un personnel de l'entreprise.

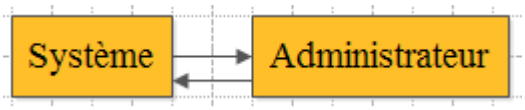


Figure 15*: Interaction de la fonctionnalité 7*

V DIAGRAMME DES CAS D'UTILISATION

Ce diagramme (voir Figure1) donne tous les cas d'utilisation, les différentes fonctionnalités vis-à-vis des utilisateurs finaux du système.

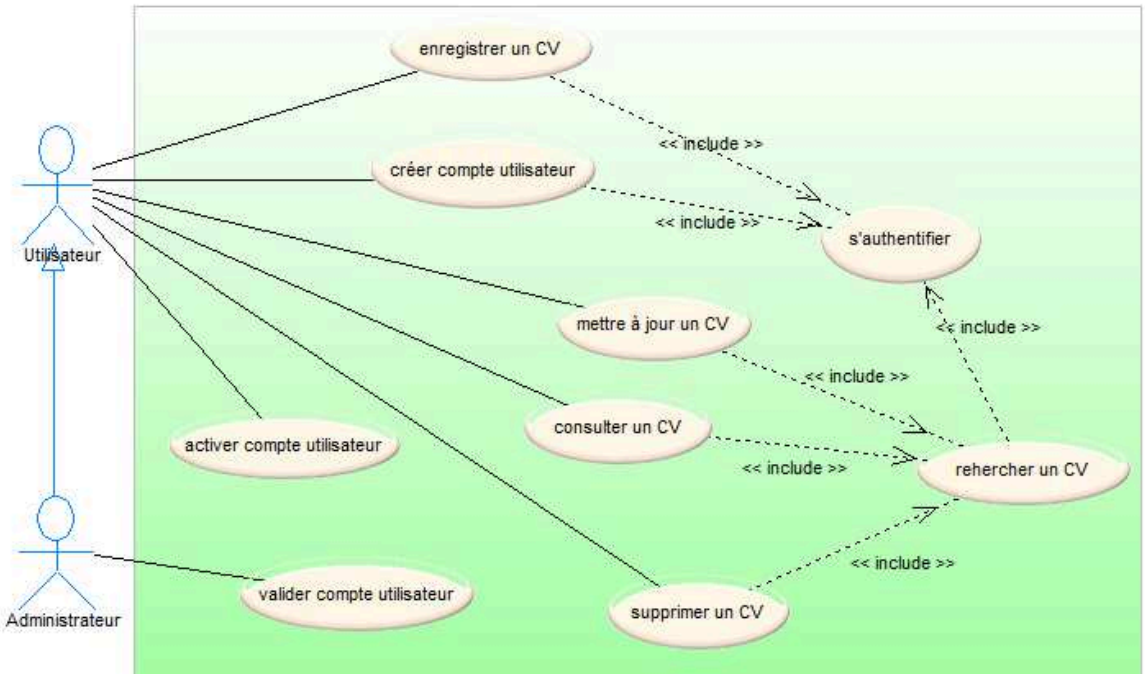


Figure 16*: Diagramme des cas d'utilisation de CVTHEQUE BOLLORE CMR*

CHAPITRE III : ANALYSE DE LA PROBLEMATIQUE

Il s'agit dans cette partie de dégager, suivant trois axes fonctionnels, les différentes entités permettant la construction de notre système, leur structure interne, ainsi que les relations entre elles. Nous nous accorderons principalement sur « ***ce que doit faire le système*** » en ne tenant pas compte des choix ni des contraintes informatiques. Cette absence de l'aspect technique nous permet de garantir que l'architecture du système est construite sur une réflexion axée sur les considérations métier et non sur les solutions informatiques. Ceci assurera une meilleure cohésion et une évolution plus souple de l'application et permettra d'envisager plusieurs architectures techniques possibles basées sur la même analyse du domaine métier.
A cette phase de l'analyse de notre application, il est primordial de dégager les différentes entités (ce sont les composantes humaines et objets qui interagissent entre eux pour faire fonctionner notre application), ainsi que les caractéristiques définissant le comportement de chacun, et les relations entre eux. Mais avant, nous rappellerons quelques notions sur les methodes de modelisation d'applications. langage objet Unified Modeling Language (UML : Langage Unifié de Modélisation) qui a été retenu pour la mise sur pied de notre application.

I ENTITES, STRUCTURES INTERNES, CYCLE DE VIE ET RELATIONS

Les entités sont les ensembles d'objets qui constituent notre système. Toutes les fonctionnalités du système sont basées autour d'elles. Chaque entité est constituée d'un ensemble d'attributs qui permettent de mieux décrire l'objet pendant son cycle de vie dans l'application. Nous dégageons dans cette section les principales entités de notre application, les noms des attributs ainsi que la codification (nom des variables).

- Entité CV (***newcv***)

Identifiant du CV (***id***) : valeur unique, permet d'identifier un et un seul CV à la fois
Civilité (***civil***) : Trois valeurs possibles ; Mme/Mlle/Mr
Nom (***name***) : Le nom d'état associé au CV
Prénom (***surname***) : Le prénom d'état civil associé au CV

Date de naissance (***dateOfbirth***) : La date de naissance associée au CV

- Diplôme (***Diplome***)

Identifiant du diplôme (***id***) : Permet d'identifier de façon unique un diplôme

Nom du domaine (***name***) : donne le nom du diplôme

- Domaine d'étude (***studydomain***)

Identifiant du domaine (***id***) : Permet d'identifier de façon unique un domaine d'étude

Nom du domaine (***name***) : Permet d'attribuer un nom à un domaine d'étude

- Formation de base (***training***)

Identifiant de la formation (***id***) : Permet d'identifier de façon unique une formation

Intitulé de la formation (***name***) : Le nom de la formation.

- Formations professionnelles

Identifiant de la formation (id)

Intitulé de la formation (name)

- Expérience professionnelle (***fieldcompetence***)

Identifiant de l'expérience (***id***) : Présente de façon unique une EP

Poste occupé (***poste_occupe***) : Précise le poste occupé pendant ces expériences

Début expérience (***date_debut***) : date de début de l'expérience à un poste

Fin expérience (***date_fin***) : Date de fin de l'expérience à un poste

Société (***societe***) : donne le nom de la société dans laquelle l'expérience a été acquise

Attribution (***responsabilite***) : principale activité menée pendant cette période de l'expérience

Référence (***reference***) : Nom et contact de l'entreprise dans laquelle l'expérience a été acquise

- Utilisateur (***User***)

Compte utilisateur (***email_user***) : Ici ce compte sera l'adresse email de l'utilisateur

Mot de passe (***password_user***)

Nom utilisateur (***nom_user***)

Prénom utilisateur (***prenom_user***)

Activation (***active***) : Savoir si un compte est actif (1) ou pas (0)

Validation (***valide***) : Savoir si un compte est déjà validé par l'administrateur

Date de création (***date_creation***) : Date à laquelle le compte a été créé

- Groupe (***Groupe***)

Identifiant du groupe (***id_groupe***)

Nom du groupe (***nom_groupe***)

- Paramètres de l'entreprise (***ParametreSociete***)

Nom de l'entreprise (***nom_ent***)

Téléphone de l'entreprise (***telephone_ent***)

- Etat matrimonial (***sMatrimoniale***)

Identifiant matrimonial (***id_matrimoniale***)

Etat (***etat_mat***)

Supprimer (***isDel***) : savoir si un état matrimoniale a été supprimé

- Etablissement (***Etablissement***)

Identifiant de l'entreprise (***id_etabl***)

Nom établissement (***nom_etabl***)

Adresse établissement (***adresse_etabl***)

Email établissement (***email_etabl***)

Téléphone établissement (***telephone_etabl***)

- Type de métier (***TypeMetier***)

Identifiant de type (***id_liste***)

Poste1 (***metier_1***)

Poste2 (***metier_2***)

Poste3 (***metier_3***)

- Téléphone (***Telephone***)

Numéro de téléphone (***numero***) :

Supprimer (***isDel***) : Savoir si un numéro de téléphone a été supprimé ou pas

- Adresse email associé à chaque CV (***Email***)

Champ email (***val_email***) : La valeur de l'adresse email

Suppression d'une adresse email (***isDeleted***) : Savoir si cette adresse est supprimée ou pas

- Adresse postale (***Adresse***)

Code postal associé (***code_postal***) : le code postal associé à cette adresse postale

La boîte postale (***b_postale***).

II DIAGRAMME DE CLASSES

Dans le diagramme qui suit, nous représentons les objets qui interviennent dans la résolution de notre problème ainsi que leurs associations. C'est de ce fait un réseau de classes et associations, qui modélisent la structure d'un objet, son rôle au sein du système ainsi que ses relations avec les autres objets.

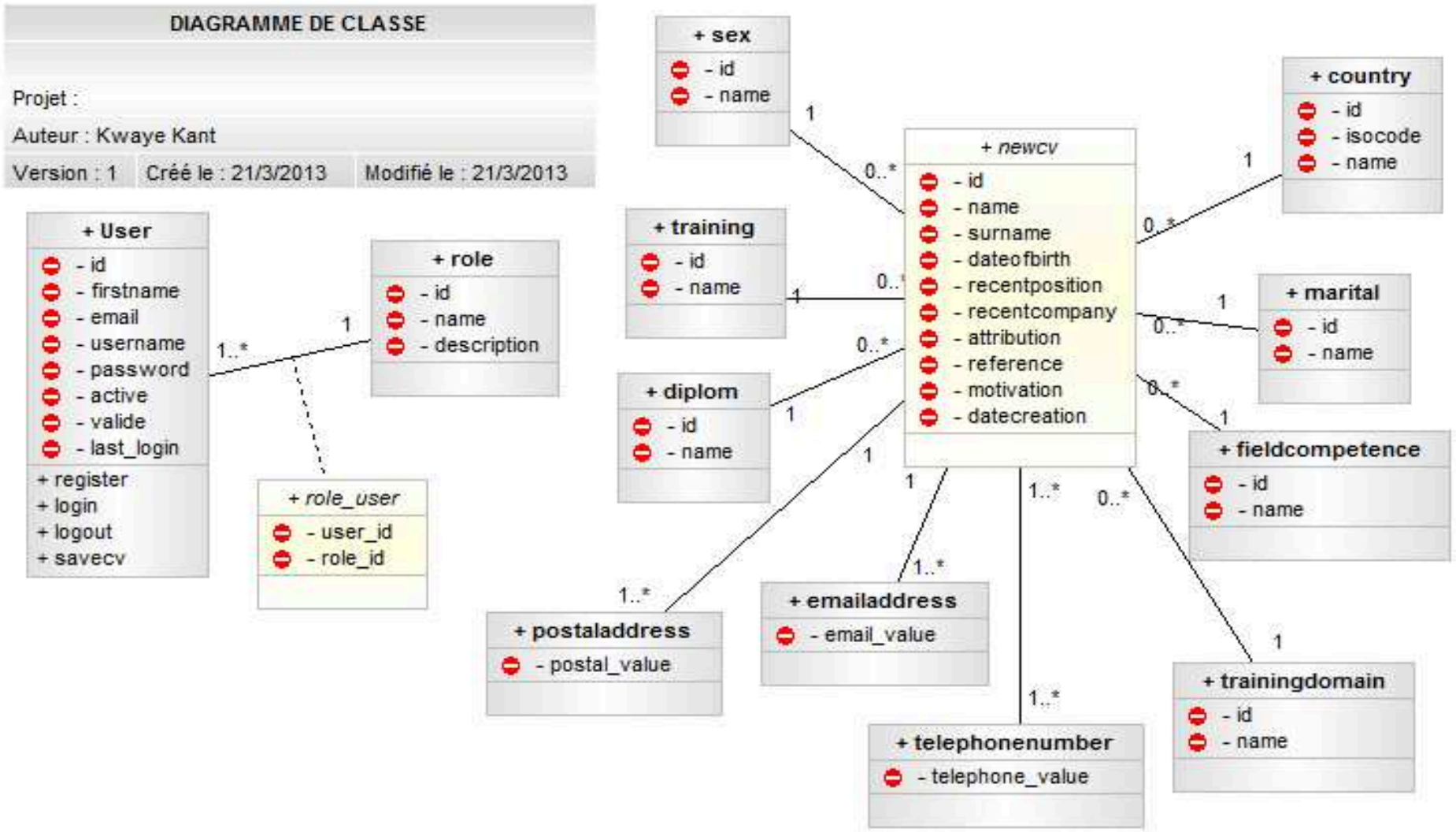


Figure 17*: Diagramme de classe de CVTHEQUE*

A l'issu des cas d'utilisations de notre système à déployer et des différentes entités y intervenant, nous avons pu produire le diagramme de classe à partir duquel nous avons fait ressortir ces entités ainsi que les relations qui les lient.

A partir de ce diagramme fourni, nous aborderons la proposition d'une solution adéquate à la problématique énoncée.

Modèle logique de données (voir annexe)

CHAPITRE IV : PHASE DE CONCEPTION ET D'IMPLEMENTATION

Dans cette section, nous allons décrire en tenant compte des choix architecturaux et des solutions d'implémentations, les langages de programmation et le système de gestion de la base de données (SGBD) choisis.

I PHASE DE CONCEPTION

Cette phase a pour objectif de s'accorder sur le *comment* et non plus sur le *quoi*. En d'autres termes, elle vise à trouver des solutions informatiques et techniques pour mettre en œuvre et construire le système analysé au cours des phases précédentes.

Il est important dans une démarche de construction logicielle de clarifier explicitement les possibilités des différentes architectures envisageables en suivant deux axes : la conception des objets (choix du langage de développement et de l'environnement de développement) et la conception du système (comment sera construire notre système. Quels types d'architectures) étudiés dans la phase d'analyse.

1.1 Conception des objets du système

Les produits de la conception objet sont constitués d'un ensemble de modèles qui sont une représentation détaillée des modèles objet de l'analyse, prenant en compte notamment les langages et les environnements de développement. Nous allons ajouter aux modèles objet de l'analyse, les détails liés à l'implémentation du système.

Dans la suite nous prendrons quelques cas précis des entités de notre diagramme de classe.

1.1.1 Entité Utilisateur

Un utilisateur devra se connecter au système en fournissant un login et un mot de passe. Mais avant il doit se créer un compte et le faire valider par un Administrateur. Nous disposerons une méthode *login()* qui prend deux arguments (*username, password*) et qui retourne un entier 1 si le compte et le mot de passe de l'utilisateur sont corrects et 0 sinon.

Public function action_login($username, $password)

Un objet Utilisateur se déconnecte au système en appelant la méthode *logout ()* qui va permettre de supprimer les variables de session login de l'utilisateur.

Void logout()

La méthode *save_cv()* retourne un entier 1 si l'enregistrement a bien été effectué et 0 sinon.

Int save_cv()

Delete_cv() prend en paramètre un objet de chaîne de caractère désignant le numéro du CV à supprimer puis retourne 1 si tout s'est bien et 0 sinon.

Int delete_cv(String numero_cv)

Pour mettre à jour un CV, un objet de type Utilisateur doit appeler la méthode update en lui passant un argument de type chaîne de caractère

Void update_cv (String numero_cv)

Nous représentons cela sous forme du graphe suivant :

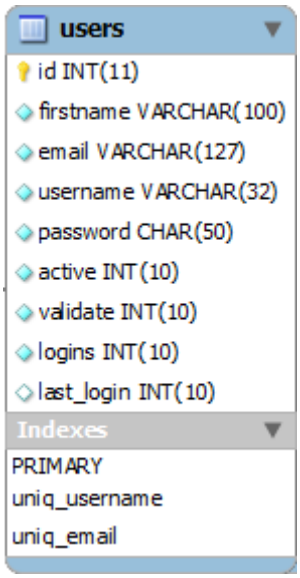


Figure 18: *Représentation complète de l'entité Utilisateur*

1.1.2 Entité newcv

Chaque CV est identifié de façon unique dans l'application grâce à son identifiant id_cv. Il possède un nom, un prénom et la date de naissance.

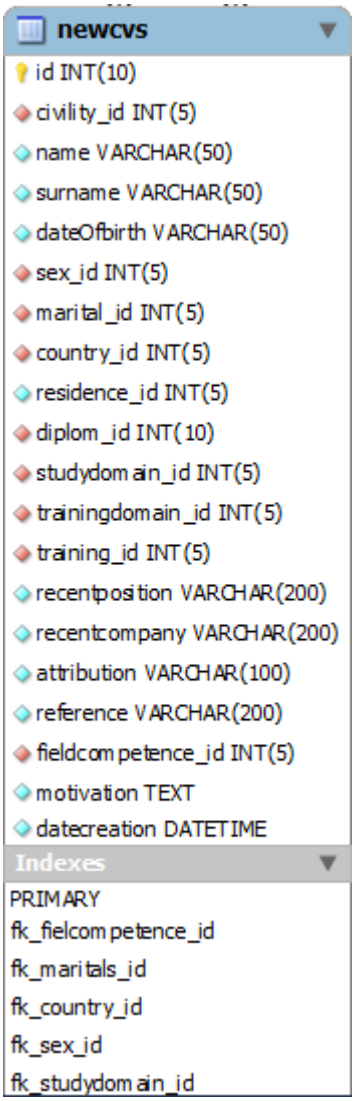


Figure 19*: Représentation complète de l'entité UnCV*

1.1.3 Des classes aux relations

Pour conceptualiser, nous utilisons le modèle objet et pour stocker nous utilisons le modèle relationnel. Dans cette partie nous allons passer du modèle objet au modèle relationnel.

Le modèle relationnel utilise structurellement trois concepts :

- La relation qui regroupe un ensemble d'attributs
- L'attribut qui est définit sur un domaine
- Le domaine qui est définit sur un ensemble de valeurs

La notion de domaine est identique à celle des types du diagramme des classes. La traduction du modèle objet vers le modèle relationnel doit donc tenir compte de deux concepts, la relation et les attributs.

En rappel de quelques règles de traduction:

- Chaque classe devient une relation.
- Chaque attribut de la classe devient un attribut de la relation.
- Chaque association entre deux objets est mémorisée et cette mémorisation est fonction de la cardinalité qui lie ces objets, plus précisément des maximums de ces objets.

Dans le tableau suivant, nous récapitulons les différentes possibilités de traduction des cardinalités.

Max A / Max B	**1**	**>1**
1	Si la clé de A = clé de B, ne rien faire Sinon Ajouter la clé de B dans la relation de A comme attribut ou Ajouter la clé de A dans la relation de B comme attribut.	Ajouter la clé de B dans la relation de A comme attribut
>1	Ajouter la clé de A dans la relation de B comme attribut	Créer une relation AB ayant comme attribut la clé de A et la clé de B

***Tableau 4* : Principe de traduction des cardinalités**

1.1.4 Modèle relationnel

Le schéma suivant est le modèle relationnel produit à partir du diagramme de classe en tenant compte des règles énoncées ci haut.

Relations

Utilisateur (**email_user**, nom_user, prenom_user, password_user, active, valide, date_creation)

UnCV (**id_cv**, civilite, nom_cv, prenom_cv, date_nais)

Email (**val_email**, isDel)

Adresse (**code_postal**, b_postale, ville)

Lexperience (**id_experience**, poste_occupe, date_debut, date_fin, societe, responsabilite)

TypeMetier (**id_liste**, nom_metier)

Smatrimoniale (id_matrimoniale, etat_mat, isDeleted)

Diplome (id_diplome, nom_diplome)

Pays (id_pays, nom_pays)

Telephone (numero, isDeleted)

DomaineEtude (id_domaine, nom_domaine)

Formation (id_formation, intitule, type_formation)

Etablissement (id_etabl, nom_etabl, adresse_etabl, email_etabl, telephone_etabl)

Groupe (id_groupe, nom_groupe)

AccessGroupe (save_cv, delete_cv, update_cv, codif_cv, find_cv, archive_cv, create_cv, valide_cv, suspend_cv, create_menu, publish_menu)

Menu (id_menu, nom_menu, link, position_menu)

Contenu (id_contenu, nom_contenu, content_contenu, isPublished, isDeleted)

ParametreSociete (nom_ent, adresse_ent, cigle_ent, siteweb_ent, email_ent, logo_ent, telephone_ent).

1.2 Conception du système

Comme annoncé ci-dessus, cette phase de conception vise à définir l'architecture globale du système aux niveaux matériel et logiciel.

1.2.1 Description globale des outils de développement

Dans cette section, nous allons décrire de manière succincte et comparative les différents langages de programmation de haut niveau avec les environnements de développement intégrés (IDE) et les différents systèmes de gestion de base de données (SGBD). Ensuite nous justifierons le choix pour la mise en œuvre de notre application.

1.2.1.1 Etude comparative des langages de programmation

Il existe plusieurs langages de programmation parmi lesquels on peut citer les plus célèbres : java, C++, Php/Html, ASP, C#. Cette multitude de langage nous confronte à une réelle difficulté à savoir le choix en fonction de la complexité et des exigences du système à développer.

Java développé par SUN et racheté par la firme ORACLE est un langage de haut niveau et dont la naissance à fait naître au sein de la communauté des développeurs un dynamisme de travail notamment de l'utilisation des concepts objets. Java permet le développement des applications lourdes orientée desktop (Java SE) et également des applications web (Java EE). Les contraintes matérielles de Bolloré ne nous permettent pas de développer les applications desktop à cause du système mis en place à savoir l'utilisation des WISE comme poste de travail côté client et dont il y a une restriction au niveau des installations.

C++ est également un langage de haut niveau et dont la genèse tire sa naissance du langage C, qui a été modifié en adoptant le concept de la programmation Orienté Objet. A cause des mêmes contraintes matérielles de Bolloré une application développée avec C++ ne peut installée.

Le C#, langage propriétaire de la firme Microsoft n'est pas portable et ne peut pas être exécutée sur les plate formes Unix et Linux. En plus les Environnements de Développements Intégrés (IDE) pour les applications C#, pour la plupart ne sont pas gratuit.

ASP pour des mêmes raisons que C# n'est pas portable, il n'est pas compatible pour l'instant avec les systèmes Unix et Linux.

Php/Html, HyperText Markup Language est un langage de développement orienté web. Il est libre, gratuit et portable c'est-à-dire qu'il peut s'exécuté sur toutes les plates formes (Linux/Windows, Unix, Mac). De plus on note une forte communauté de développeurs qui travaillent autour de ce projet. L'accès à une page web se fait via un navigateur web.

1.2.1.2 Justificatif du choix du langage de programmation

Après une étude comparative des différents langages de développement, notre choix porte sur le langage web PHP dans sa version 5.3 ainsi que le HTML5. Compte tenu des contraintes matérielles de Bolloré, une seule exigence est requise : un navigateur web sur les postes utilisateur. Etant donnée que le parc informatique de Bolloré coté utilisateur est équipé du navigateur Internet Explorer. Nous avons choisir de développer notre application en utilisant le langage PHP/Html ainsi que les technologies récentes du web (AJAX, modèle MVC, PDO).

1.2.1.3 Quel Framework PHP pour notre application ?

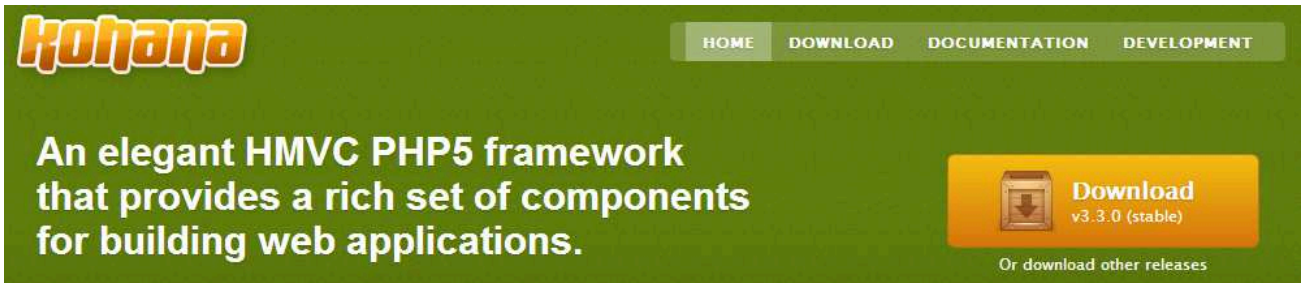
Un Framework est un ensemble de bibliothèque qui permet de faciliter le développement d'un logiciel, un Framework est surtout utile pour ne pas avoir à réécrire de temps en temps les même codes.

Il existe une multitude de Framework PHP parmi lesquels nous avons choisi le ***Framework Kohana.***

Le Framework Kohana est très facile à apprendre, léger et permet de développer rapidement des applications assez complexes. Ce qui est intéressant dans ce Framework est son système de protection globale XSS sur GET, POST, COOKIE, et SESSION ainsi que ses librairies ORM (*Object Relational Mapping*) simple à utiliser et son pattern. Ainsi les modules peuvent être développés séparément et assembles à la fin.

Il dispose de plusieurs Libraries et Helper qui répond à quasiment tous les besoins pour développer son application.

Figure 20: *Vue du Framework Kohana*



Kohana est actuellement dans sa version 3.3.0 et téléchargeable à l'adresse https://github.com/downloads/kohana/kohana/kohana-3.3.0.zip (lien valide à la rédaction de ce document).

Le code source de Kohana est maintenant par GitHub et tout le monde peut contribuer au développement des modules futurs.

1.2.1.4 Pré-requis et Installation de Kohana.

A Pré-requis au fonctionnement de Kohana.

Pour que Kohana fonctionne il y a un certain nombre de pré-requis qui doivent être vérifiés.

- Le système d'exploitation Linux – Ubuntu Server (confère Pagexxx – Choix du SE)
- Un serveur Web Apache (confère pagexxxx – Choix du serveur web)
  http://httpd.apache.org/docs/2.2/fr/install.html
- PHP (confère Pagexxx – Justificatif du choix du langage de programmation)
  http://www.php.net/manual/fr/install.unix.apache2.php
- MySQL (confère Pagexxx – Justificatif du SGBD)
  http://dev.mysql.com/doc/refman/5.0/fr/installing-binary.html
- PHPMyAdmin, pour l'administration aisée de la base de données.
  http://doc.ubuntu-fr.org/phpmyadmin

Nous vous renvoyons sur le site des Editeurs pour les procédures d'installation des différents composants.

Dans le cadre de cette démo, nous allons procéder à une installation basique du package-tout-en-un **EASYPHP** qui inclut à la fois ***MySQL-PHP-APACHE-PhpMyAdmin*** et sur un système d'exploitation Windows.

La procédure d'installation de EasyPHP est décrite ici *http://www.easyphp.org/save-easyphp-latest.php*

La fin de l'installation nous montre les fichiers et répertoires telle que présentés sur la figure suivante.

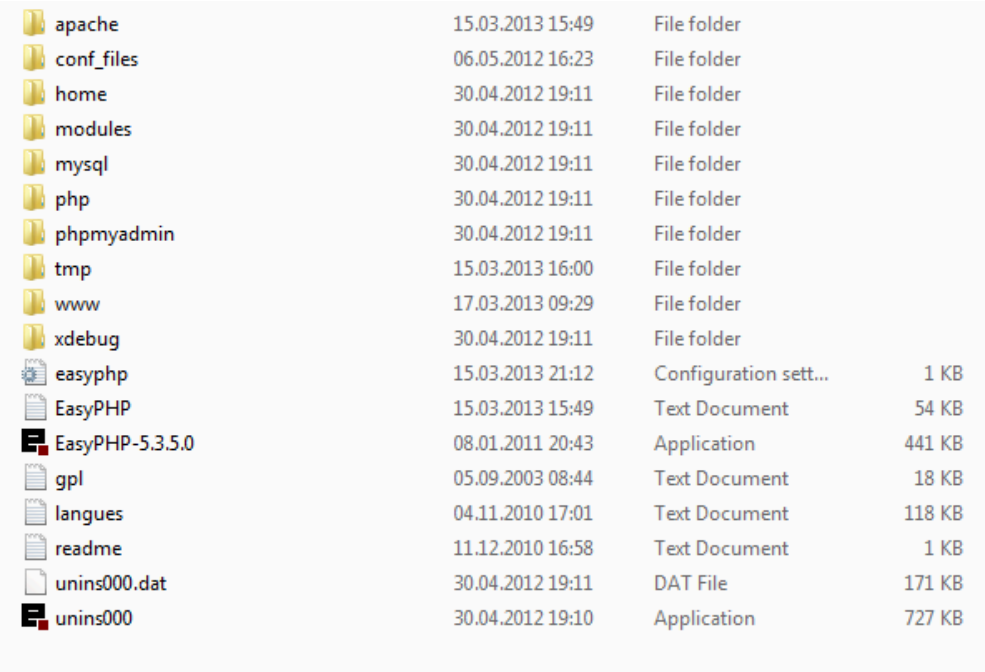


Figure 21: *Répertoire installation du serveur web apache*

B Installation de Kohana.

Une fois l'archive téléchargée, dezippons là à la racine (dans le répertoire ***/www/***) de notre site.

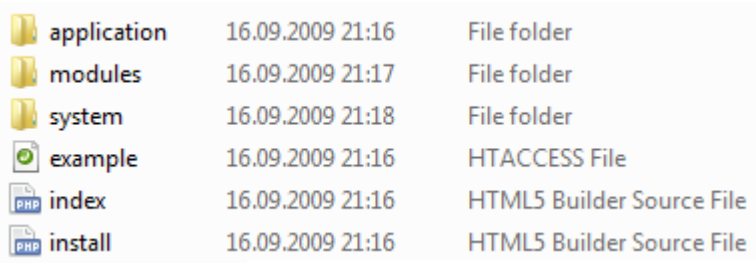


Figure 22: *Répertoire complet de Kohana*

Nous remarquons la présence des répertoires :

B.1 Application.

Il s'agit du répertoire dans lequel tous les codes de notre application sont logés.

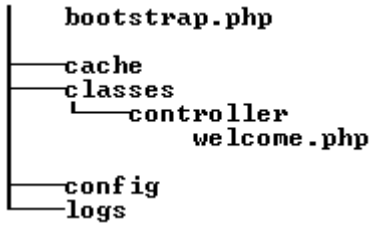


Figure 23: *Arborescence du répertoire Application de Kohana*

- Le fichier *bootstrap.php* permet de configurer l'application en ***définissant le chemin absolu de base, le contrôleur par défaut et toutes les configurations*** pour nécessaire pour faire fonctionner l'application.
- Le répertoire ***classes*** contient un répertoire ***controller*** qui contient tous les modules contrôleurs. Ici à l'imagine, ***welcome.php,*** le contrôleur par défaut de Kohana.
- Le répertoire ***config*** contient tous les fichiers de configuration directement liés au système. Par défaut il est vide. Par exemple pour configurer la base de données, il faut simplement copier ***/modules/database.php*** et puis éditer ce fichier avec les paramètres requis d'accès à la base de données.
- ***Logs*** est répertoire pour tracer les exceptions lors l'exécution du code.

B.2 Modules

Il existe des modules déjà développés avec Kohana, il suffit de l'activer dans l'application afin de l'utiliser. Le développeur peut également écrire ses propres modules.

B.3 System

Ce répertoire contient toutes les libraries utilisées par Kohana pour mettre en place l'ensemble de son système.

B.4 Example.HTACCESS

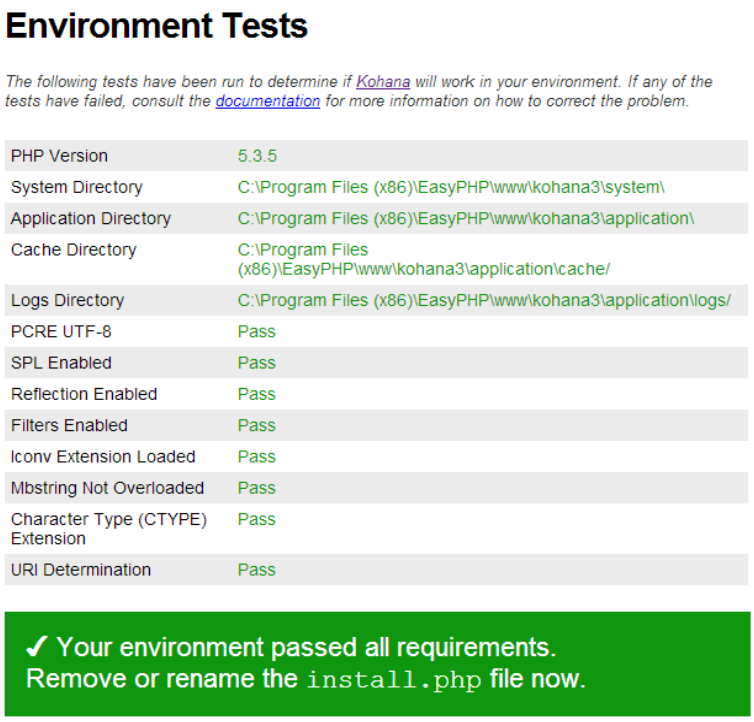
Ce fichier doit etre renome en .HTACCESS. Il contient les parametres de securite d'access illegal aa l'application. La modification qu'il faut apporter dans ce fichier c'est de modifier la ligne RewriteBase /kohana/ en RewriteBase / puisque nous avons installee Kohana aa la racine de notre serveur web. Au cas ouu nous l'avions installee dans un repertoire MonWeb alors la ligne ressemblerait aa ceci RewriteBase /MonWeb/.

B.5 Index.php

Est le fichier d'entrée de Kohana. C'est le premier fichier qui est ouvert lorsque nous appelons un site Kohana.

B.6 Install.php

C'est grâce aa fichier que Kohana s'installe pour la première fois. A la fin de l'installe nous devons supprimer ou renommer ce fichier.



Environment Tests

The following tests have been run to determine if Kohana will work in your environment. If any of the tests have failed, consult the documentation for more information on how to correct the problem.

PHP Version	5.3.5
System Directory	C:\Program Files (x86)\EasyPHP\www\kohana3\system\
Application Directory	C:\Program Files (x86)\EasyPHP\www\kohana3\application\
Cache Directory	C:\Program Files (x86)\EasyPHP\www\kohana3\application\cache/
Logs Directory	C:\Program Files (x86)\EasyPHP\www\kohana3\application\logs/
PCRE UTF-8	Pass
SPL Enabled	Pass
Reflection Enabled	Pass
Filters Enabled	Pass
Iconv Extension Loaded	Pass
Mbstring Not Overloaded	Pass
Character Type (CTYPE) Extension	Pass
URI Determination	Pass

✓ Your environment passed all requirements.
Remove or rename the `install.php` file now.

Figure 24: *Confirmation installation avec succès de Kohana*

1.2.1.5 Etude comparative des systèmes de gestion de la base de données (SGBD)

Le management des systèmes d'information tire son grand avantage dans l'utilisation des logiciels de gestion des banques de données. Ces logiciels permettent de conserver, de restituer et de traiter les données. Plusieurs logiciels de traitement des banques de données ont vu le jour. Dans les lignes suivantes, nous décrirons les différents SGBD du marché avec pour chacun son avantage et son inconvénient.

- *SQL Server*

SQL Server, c'est un SGBD propriétaire de Microsoft. Il n'est pas portable c'est-à-dire qu'il ne peut s'exécuter que les plates formes windows. Non libre et non gratuit.

Avantages

- Administration aisée
- Performant sous Windows en configuration par défaut
- Langage T-SQL très convivial, intégration de CLR
- Service web
- Sous SELECT possible dans clause FROM

Inconvénients

- Distribution fortement liée au Système d'exploitation
- Mono plateforme

- Pas de cluster
- Pas de prise en charge du LDAP

➢ *ORACLE*

C'est un SGBD performant, robuste et évolutive. On distingue les versions exécutables sur les différentes plates formes du marché. Il n'est pas libre et elle est commerciale.

Avantages

- Intégration LDAP
- Procédures stockées en PL-SQL
- Gestion centralisée de plusieurs instances
- Service web / Support XML

Inconvénients

- Prix élevé
- Administration complexe
- Porosité entre les schémas : difficile de faire cohabiter de nombreuses applications sans devoir créer plusieurs instances.

➢ *MySQL* (Licence : GPL et commerciale)

Mysql Community Server: Licence GPL

Avantages

- Solution très courante en hébergement public
- Très bonne intégration dans l'environnement Apache/Php
- OpenSource
- Support du cluster

Inconvénients

- Support incomplet des triggers et procédures stockées
- Pas d'héritage de tables
- Pas de vue matérialisée

➢ *PostgreSQL*

Avantages

- OpenSource et gratuit
- Héritage de tables

Inconvénient

- Pas de service web
- Sauvegarde peu évoluée

1.2.1.6 Justification du choix du SGBD

Notre application devra s'interfacer avec une base de données performante, robuste, évolutive et adaptable, libre, gratuit afin d'assurer une meilleure cohérence avec les différents composants du système. Le SGBD MySQL est mieux adapté à nos exigences et nous avons choisi de l'utiliser pour le développement de notre application.

1.2.1.7 Technologies

Il existe des technologies permettant la création des pages web dynamique. Nous avons choisi la technologie Web 2.0, la technologie AJAX, les librairies jQuery (écrites en Javascript), les feuilles de style CCS3.

1.2.1.8 Justification du choix du système d'exploitation

Ubuntu Server est une distribution Linux libre et gratuit dans sa version 6. Il intégré des fonctionnalités avancées de type serveur.

1.2.1.9 Configuration matérielle requise

- *Côté serveur*

La configuration minimale nécessaire pour le fonctionnement de notre application est :

  - Disque dur : 5 Go
  - Processeur : 513 MHz
  - Mémoire RAM : 128 Mo
  - Système d'exploitation : Linux, Distribution Ubuntu Server

- *Côté client*

Tout ordinateur, quelque soit sa caractéristique matérielle et possédant un navigateur web peut exécuter notre application.

1.2.1.10 Environnement de développement intégré

Il existe de nombreux IDE qui servent au développement des applications web et desktop. Eclipse, Notepad++, Zend Studio, PHPEdit, Netbeans.

Notre choix s'est porte sur NetBeans pour sa simplicité, la facilite d'intégration des plugins et sa souplesse.

1.2.2 Description de la solution cible

Logiquement le portail web est construit selon une architecture en trois couches ou tiers :

- ✓ La première couche est la couche de stockage, qui permet de stocker les contenus des pages et des comptes utilisateurs dans la base de données MySQL.

- ✓ La seconde couche est la couche de traitement, c'est cette couche qui gère les aspects liés à la gestion (création, suppression et mise à jour) des contenus et des comptes utilisateurs. Elle reste à l'écoute des requêtes des utilisateurs et communique avec les couches de stockage pour renvoyer les résultats qui sont affichés par la couche de présentation.
- ✓ Le troisième et dernier tiers est la couche de présentation, c'est la couche visible aux utilisateurs. Elle sert d'interface entre le système et les utilisateurs ainsi que les administrateurs et à partir de tout navigateur web.

II IMPLEMENTATION DU LOGICIEL

Il est question dans cette section de notre document de produire le code source nécessaire pour le codage de notre application, traduire les algorithmes et les structures de données vus pendant l'élaboration du diagramme de classe.

Nous listons ci-dessous quelques codes de certaines classes de notre application, dans le langage de programmation qui a été choisi (PHP 5.3).

2.1 Classe Utilisateur (User)

Cette classe permet de traiter toutes les informations relatives à un utilisateur. La méthode *login()* permet à un utilisateur de se connecter à notre application.

Code source de la fonction login(), confère Annexe Page XXXX

2.2 Création de compte utilisateur

Les codes de la figure ci-dessous permettent de créer un compte utilisateur et d'insérer les informations dans la base de données.

La méthode *action_register()* déclenche une action qui enregistre les cordonnées d'un nouvel utilisateur

Code source de la fonction action_register(), confère Annexe Page XXXX

2.3 Vérification des données saisies

Cette classe vérifie les données saisies pour la création de compte utilisateur avant la sauvegarde.

Code source de la classe User, confère Annexe Page XXXX

2.4 Récupération du titre de la page, exemple de code du Contrôleur

L'application CVTHEQUE BOLLORE CMR est un contenu dynamique. Les scripts interagissent avec la base de données pour charger les informations selon la requête de l'utilisation. Le code suivant permet d'interroger la base de données afin d'extraire une partie du code du *template.*

```
$companyname = new Model_Company;
$this->template->companyname = $companyname->get_companyname();
```

III PRESENTATION DE L'ENVIRONNEMENT DU PRODUIT

Le logiciel se présente sous la forme d'une interface web type PHP/HTML, l'utilisation de ce type de codage permet une portabilité d'un système d'exploitation à un autre. Il sera hébergé sur un serveur web lié à une base de données pour la gestion des différents CV.

3.1 Interface utilisateur

Chaque utilisateur du logiciel dispose des droits en fonction de son statut au sein de l'entreprise.

3.1.1 Interface matérielle

Le logiciel s'intègre et s'adapte automatiquement au matériel sur lequel il tourne. L'utilisation d'une interface web comme support pour notre application permet une portabilité complète avec les différents systèmes d'exploitation tels que Linux/Unix. La seule nécessité réside dans l'existence d'un réseau informatique LAN/WAN architecturé en étoile sur le modèle Client/Serveur. Les protocoles utilisés sont de type communication réseau : *Transport Control Protocol/Internet Protocol* (TCP/IP) pour la couche *OSI transport et réseau* et *HyperText Transfer Protocol* (http) pour la couche *OSI application*.

3.1.2 Interface logicielle

Notre application est une application maison, il ne s'interface pas avec un autre logiciel.

3.1.3 Interface de communication

La communication avec le système s'effectue via les technologies réseaux actuelles. Au sein de l'entreprise, l'application est accessible via son réseau local (Local Area Network : LAN) propre. Nous vous présentons sur la figure suivante une ébauche de l'architecture réseau mise en place.

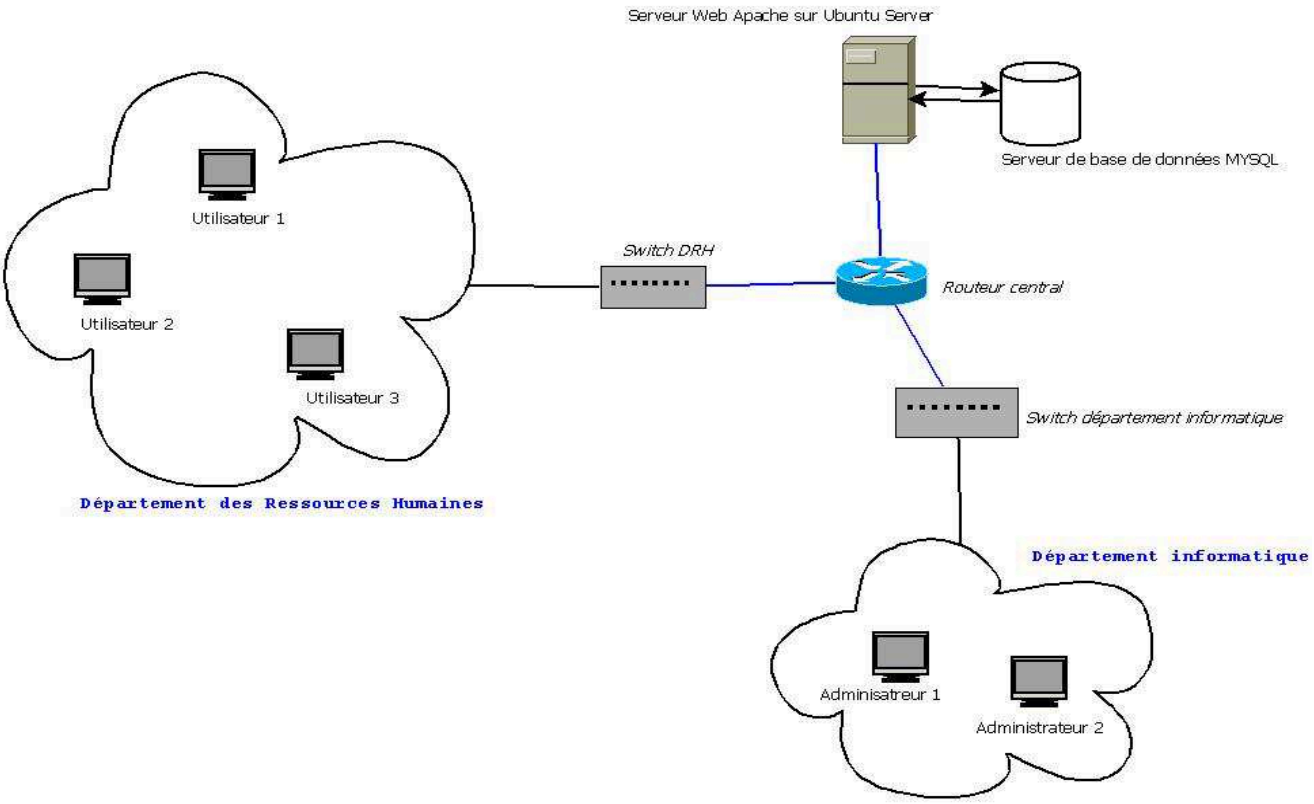


Figure 25*: Architecture réseau décrivant l'environnement de fonctionnement de notre application*

3.2 Interface Homme-Machine

Dans cette section, nous présenterons les séquences d'écran permettant la communication entre notre application et l'utilisateur. Nous l'illustrerons à travers les captures d'écran depuis l'interface fonctionnelle de l'application en nous basant sur les cas d'utilisations et des fonctionnalités systèmes précises. Le schéma ci-après est une architecture sur laquelle est basée cette partie de notre document, de l'interface aux infrastructures en prenant en compte les objets qui interviennent dans le système.

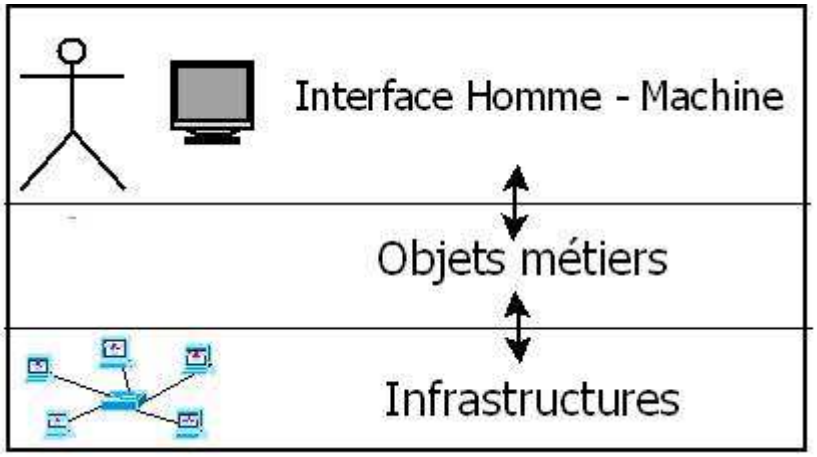


Figure 26*: Interface Homme - Machine*

3.3 Interface d'installation de notre application

Initialement, le dossier contenant les fichiers d'installation de notre application se présente comme décrit sur le schéma ci-dessous.

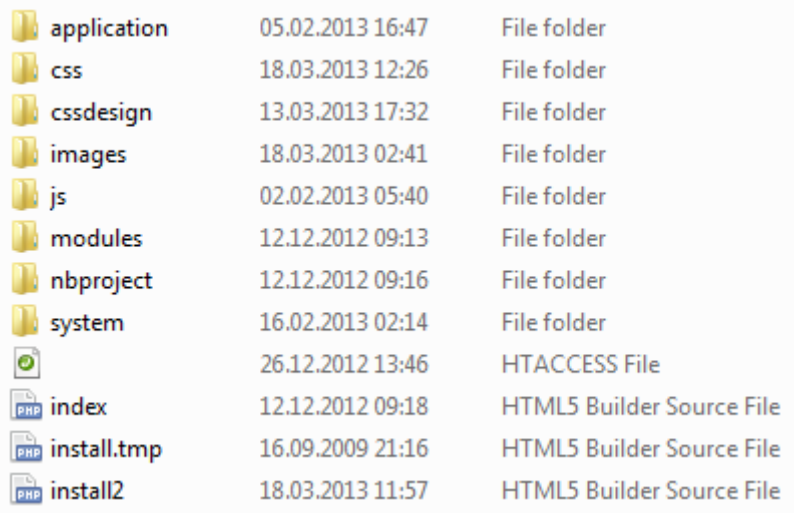


Figure 27*: Répertoire racine et les sous répertoires de l'application*

Le point d'entrée de notre application est le fichier *index.php*.

Code source de la classe User, confère Annexe Page XXXX

L'installation du Framework Kohana commence si le fichier *install.php* existe. Ce fichier devra être renommé ou supprimé après l'installation.

Le code ci-dessous installe l'application CVTHEQUE réellement.

```
public function action_index()
    {
        if(!file_exists(DOCROOT.'install2.php')) {
            Request::instance()->redirect('install/admin_setting');
        } else {
            Request::instance()->redirect(url::base().'showpublic');
        }
    }
```

Si le fichier *install2.php* existe alors l'on a déjà installé l'application, la page de connexion se lance. Sinon procéder a l'installation en chargeant *install/admin_setting*.

3.3.1 Création du compte Super Administrateur et Configuration de l'application.

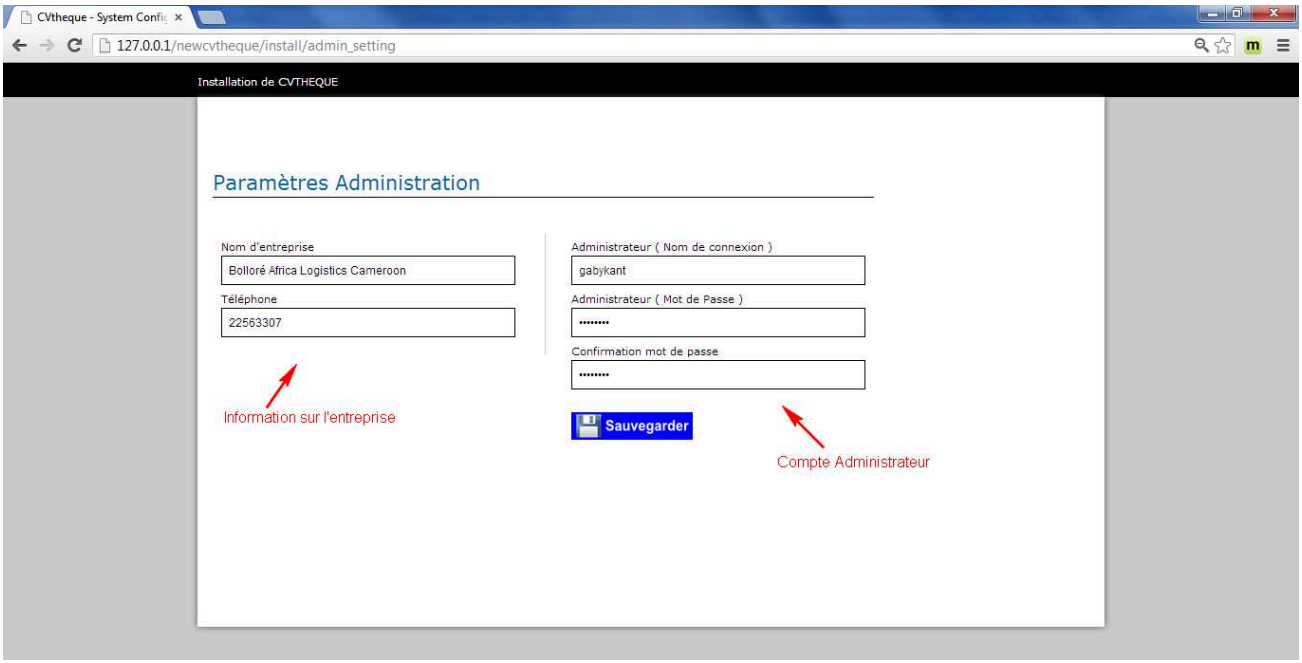


Figure 28: *Interface d'installation de l'application et création du compte Super Administrateur*

3.3.2 Interface de création de compte utilisateur.

L'utilisateur doit fournir son nom, son adresse email, son nom de connexion (*Identifiant*) et un mot de passe au choix.

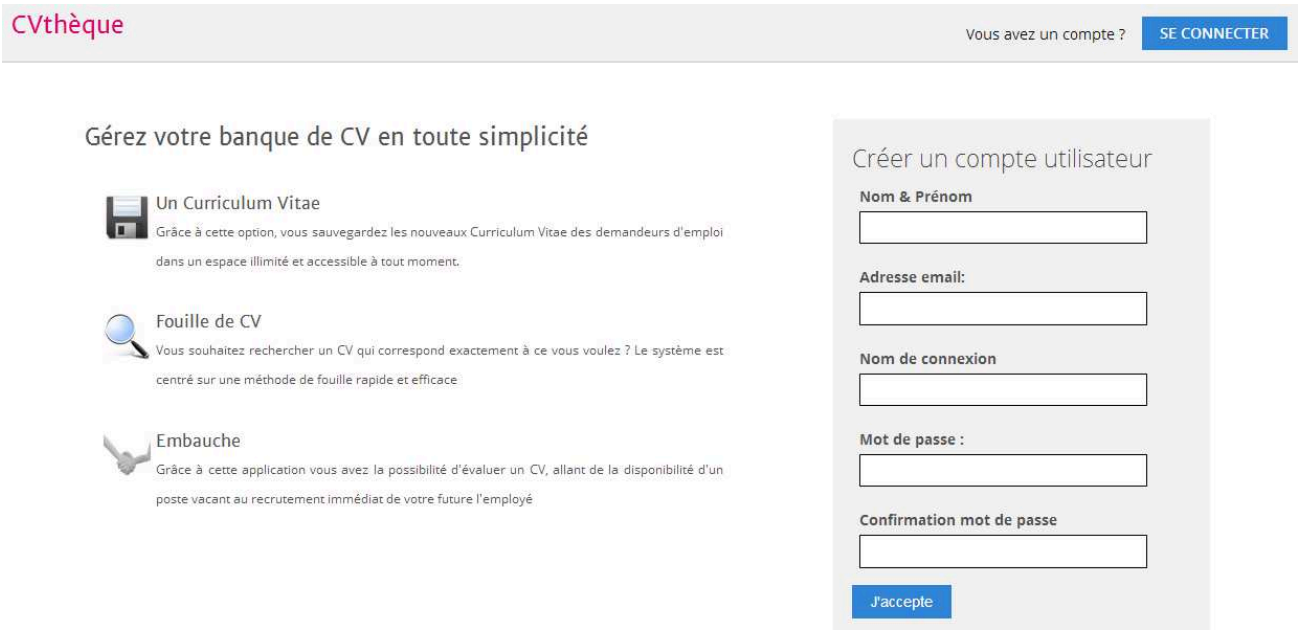


Figure 29: *Interface de création de compte utilisateur*

3.3.3 Interface de connexion

Connexion de l'utilisateur à l'aide de son identifiant et de son mot de passe

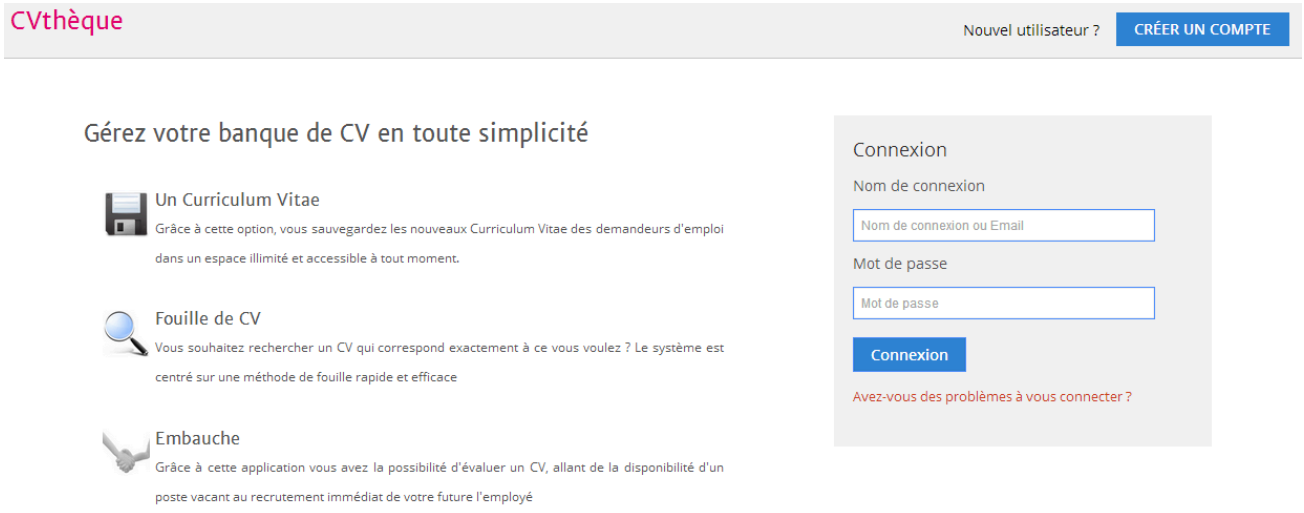


Figure 30: *Interface de connexion à l'application*

3.3.4 Interface de récupération de mot de passe perdu.

Lorsqu'un utilisateur ne se souvient plus de son mot de passe, le système lui demande d'introduire son adresse email et après soumission, un lien lui est envoyé par mail. Il devrait cliquer ce lien pour être invité aa renouveler son mot de passe.

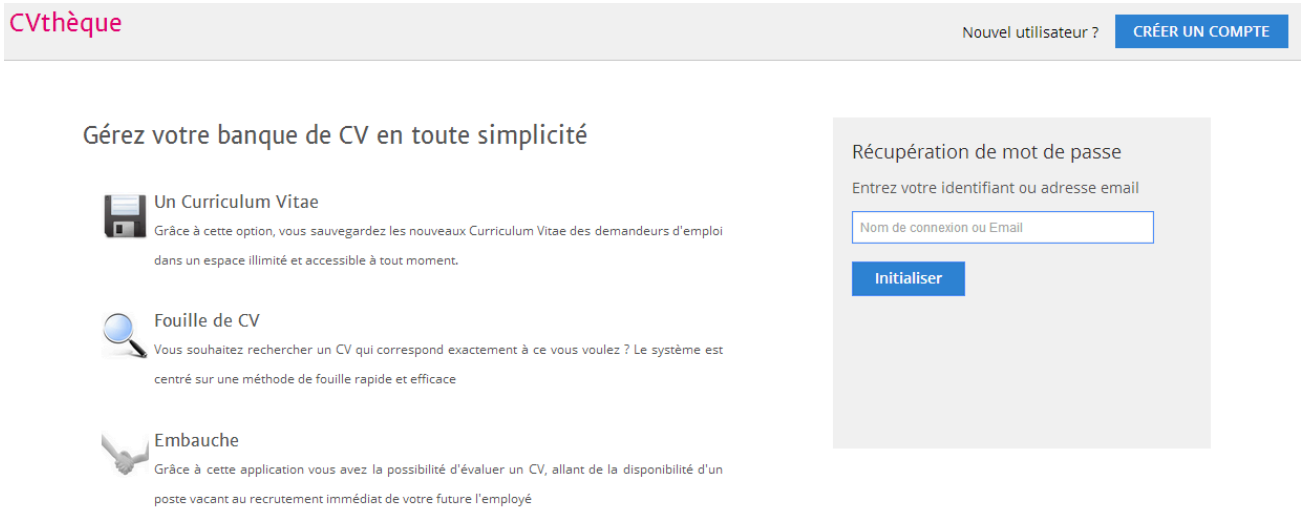


Figure 31: *Interface de récupération du mot de passe de l'utilisateur*

3.3.5 Interface finale de l'application (Tableau de bord)

Nous présentons ici l'interface finale du produit incluant tous les liens d'utilisations.

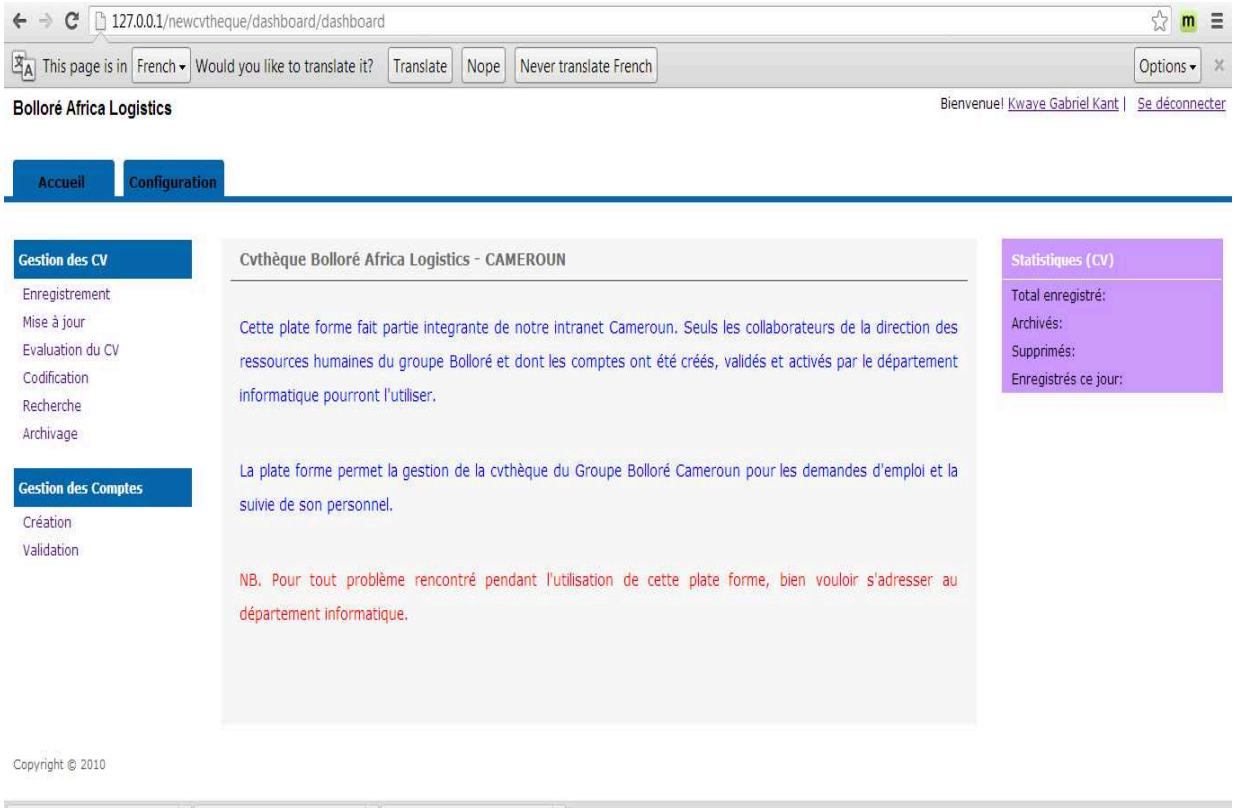
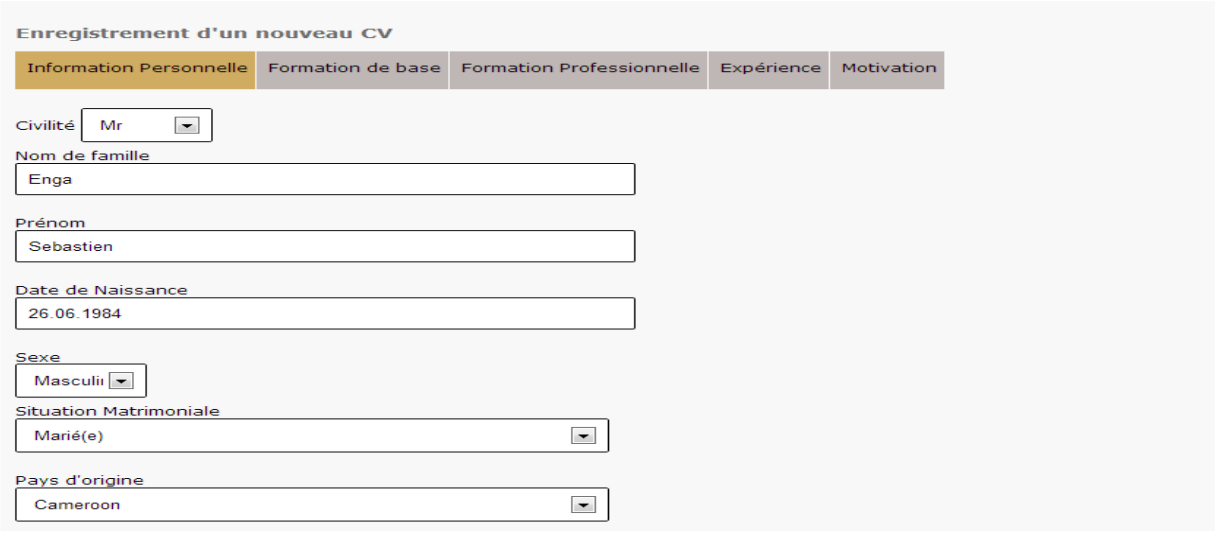


Figure 32: *Interface finale de l'application*

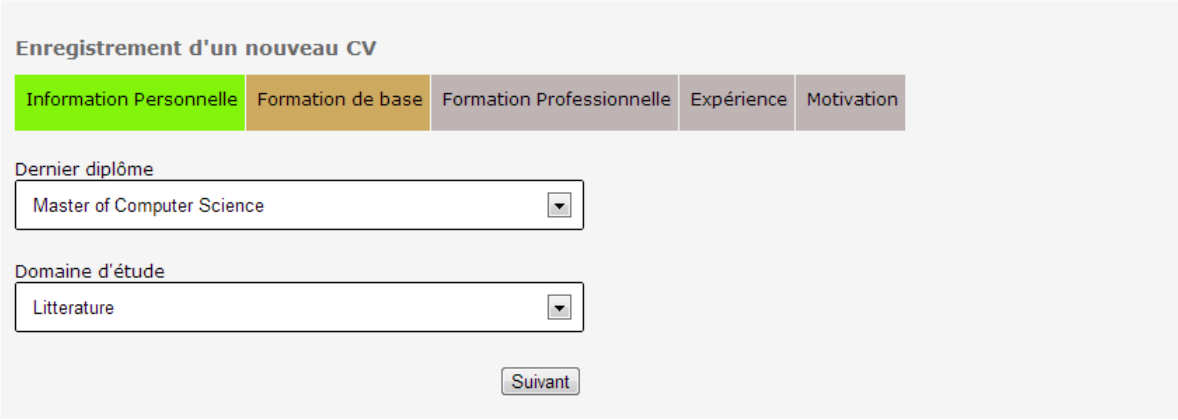
3.3.6 Interface du cas d'utilisateur « Enregistrer un nouveau CV »

Nous avons choisi de présenter les étapes d'enregistrement d'un nouveau au travers de notre application.

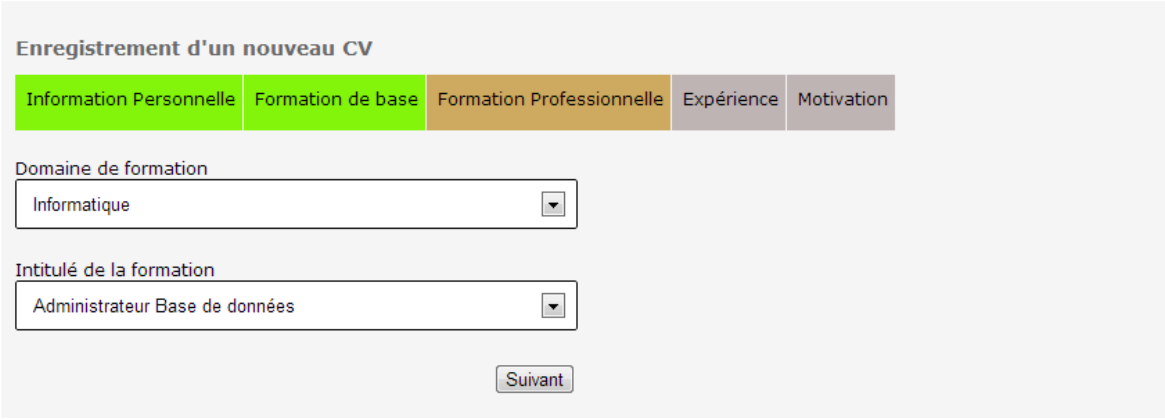
1er étape : Informations personnelles sur le candidat



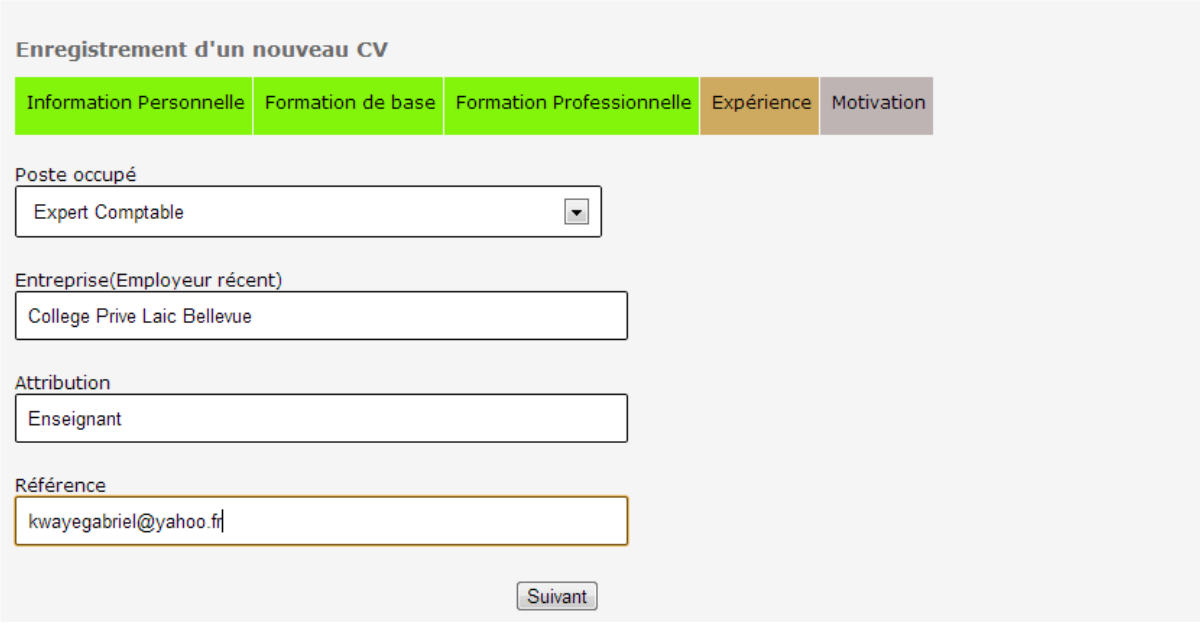
2ième étape : Formations de base du candidat



3ieme étape : Formation professionnelle du candidat



4ieme étape : Expérience professionnelle du candidat



5ieme étape : Motivations du candidat pour le poste sélectionne.

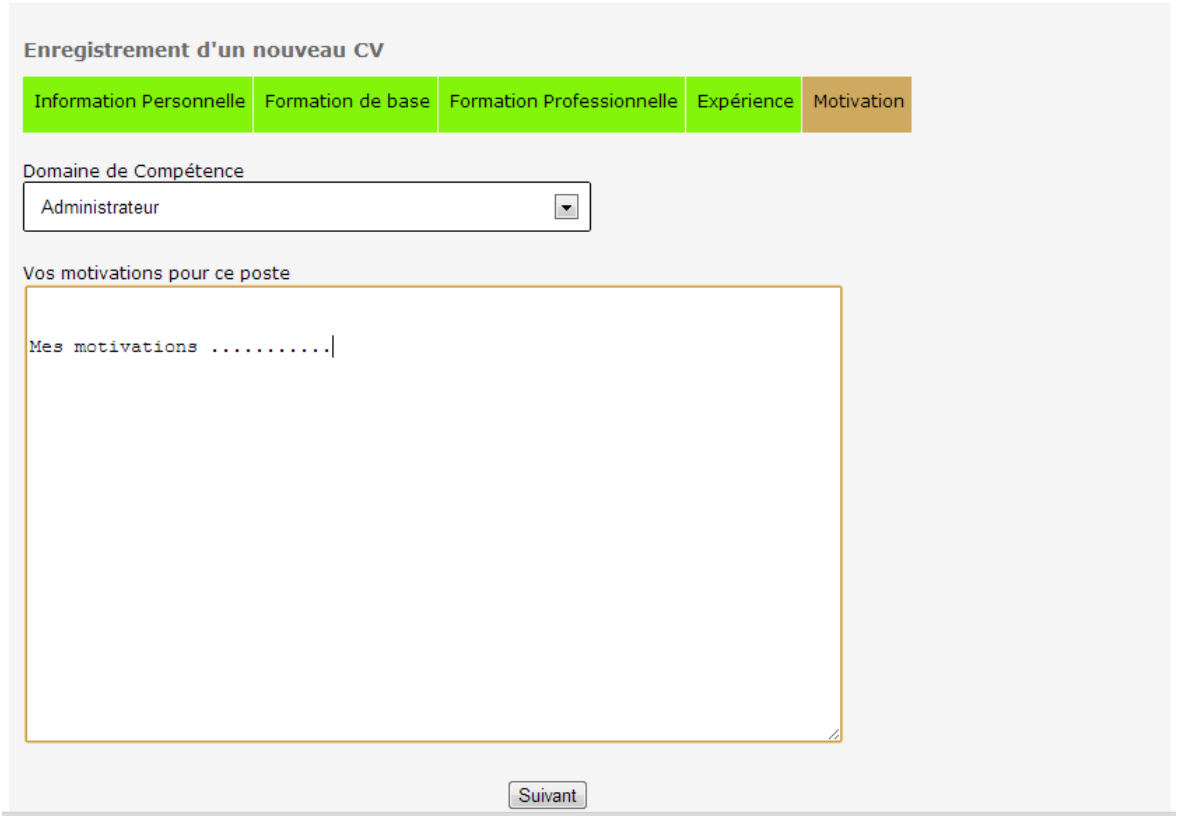


Figure 33*: Etapes d'enregistrement d'un nouveau CV*

CHAPITRE V : DEPLOIEMENT ET DE MISE EN PRODUCTION

I ETUDE DE DEPLOIEMENT

Dans cette section, nous décrirons les étapes de déploiement de notre application au sein du groupe Bolloré. Il est primordial de présenter le moyen architectural de déploiement de notre application. UML dans sa version 2.3 précise un diagramme pour illustrer cette phase de développement logiciel. La figure suivante est le diagramme de déploiement de l'application. Nous avons fait abstraction de l'architecture réseau LAN actuelle du groupe Bolloré ainsi que la technique d'adressage IP.

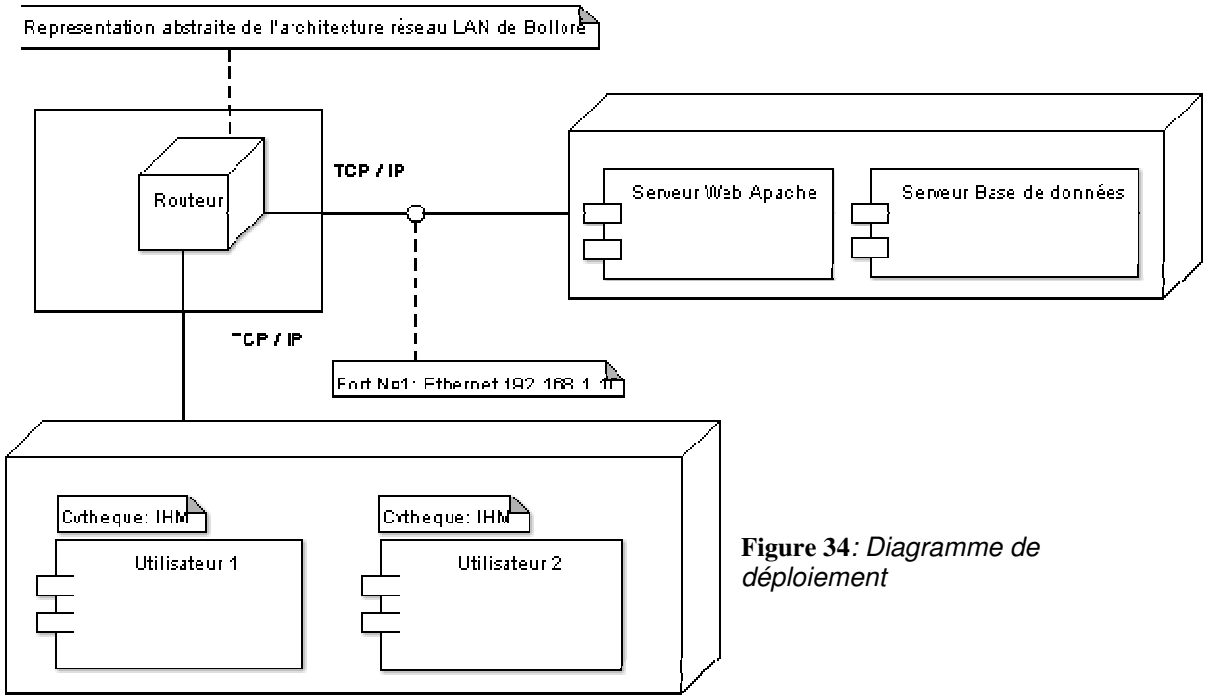


Figure 34: *Diagramme de déploiement*

Nous avons reçu les instructions du département informatique à utiliser l'architecture réseau LAN actuelle de Bolloré. Une étude de déploiement du réseau n'étant pas dans le cadre de notre travail et donc pas nécessaire.

- Le composant Serveur Web Apache décrit le nœud du réseau qui va héberger notre application web CVthèque. Pour rendre ce serveur fonctionnel et disponible, l'administrateur en charge du réseau LAN de Bolloré a attribué une adresse IP fixe à notre serveur.

II MISE EN PRODUCTION

Une fois notre architecture réseau mise en place, nous avons procédé à une vérification des connectivités en nous assurant de l'accessibilité de notre serveur à partir des postes utilisateurs devant utiliser l'application. Et puis nous utilisons un client FTP pour le transfert des fichiers au serveur.

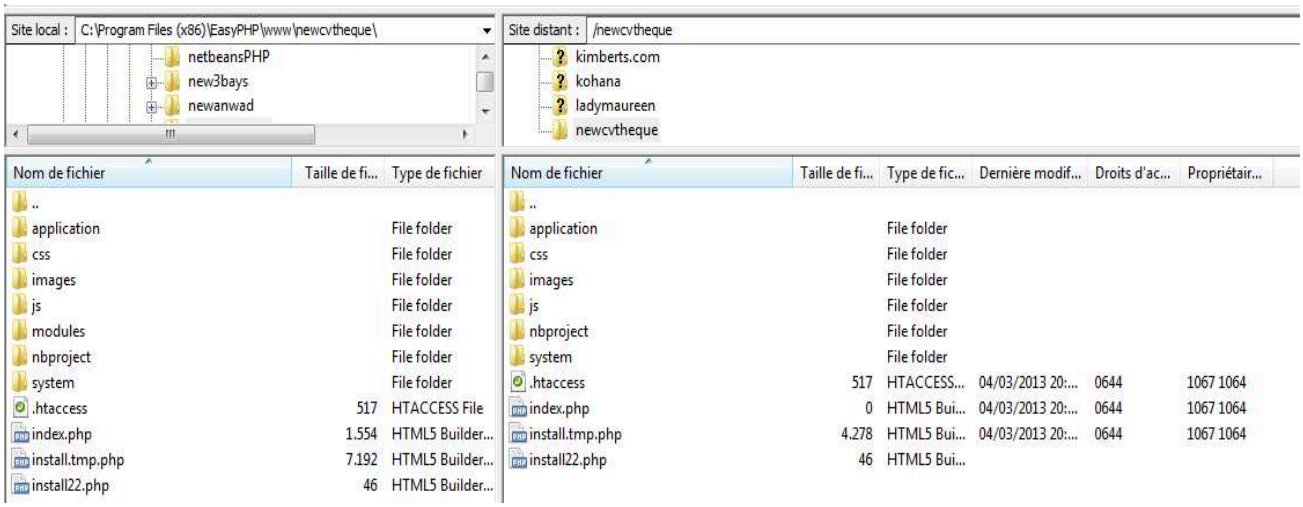


Figure 35*: Transfert des fichiers sur le serveur d'application a l'aide du client FTP FileZilla*

A partir d'un navigateur web (Internet Explorer ou Firefox ou GoogleChrome), nous saisissons l'adresse IP de notre serveur. Le navigateur nous conduit ensuite à la racine de notre application présente sur le serveur. Pour la première fois, nous sommes invités à installer l'application en suivant les étapes affichées à l'écran. A la fin de l'installation, nous devons supprimer ou renommer le fichier *install2.php* à la racine de notre application.

CONCLUSION GENERALE ET PERSPECTIVES

Conclusion

Dans une chaîne de production industrielle où tout est contrôlé avec les méthodes artisanales de l'informatique, l'utilisation des techniques de modélisations que nous apporte UML nous a permis, au terme de ce travail d'apporter notre contribution à l'amélioration de la méthode de gestion des demandes d'emploi au sein du groupe Bolloré. Nous sommes partis du cahier des charges que nous avons reçu de la Direction des Ressources Humaines et nous avons parcourus les étapes indispensables à la réalisation d'une application logicielle suivant les normes du standard de l'ingénierie logicielle, en adoptant les concepts de l'UML. Tout au long de notre travail, nous avons abordé les points suivants :

- **Spécification** : Nous avons délimité précisément le système et décrit les différentes manières de l'utiliser du point de vue des divers utilisateurs. A l'issu de cette phase nous avons élaboré le diagramme des cas d'utilisation.
- **Analyse** : Nous avons identifié les éléments intervenants dans le système, leurs structures internes ainsi que leurs relations. Au cours de cette phase, nous avons fait apparaître les caractéristiques métiers des objets suivants trois axes :
  - Un axe fonctionnel décrivant le savoir faire de chaque objet ;
  - Un axe statique représentant la description structurelle de l'objet ;
  - Un axe dynamique décrivant le cycle de vie de l'objet au cours de l'application (les états par lesquels passe l'objet ainsi que les événements qui lui sont envoyés).

  Ceci nous a conduit à l'élaboration du diagramme des classes de notre application.
- **Conception** : Dans cette phase nous avons déterminé la manière de résoudre le problème étudié par l'analyse et donc de proposer des solutions d'implémentation et de réalisation. Pour cela, plusieurs choix ont été effectués :
  - Le choix d'une architecture technique ;
  - Les décisions de performances et d'optimisation ;
  - Les stratégies de programmation et de persistance des données.

- **Implémentation** : Cette phase est l'étape prédominante du cycle de vie de notre système. Nous y avons traduit, dans le langage de programmation et la base de données appropriés, les structures de données et les algorithmes définis par la conception.
- **Déploiement et test** : Le diagramme de déploiement produit au cours de cette phase comporte les entités systèmes (LAN, architecture matérielle) à mettre en œuvre.

Le travail que nous avons effectué n'est pas arrivé à son terme, toutefois les fonctionnalités nécessaires pour une utilisation basique sont mises à la disposition du personnel du département informatique et du département des ressources humaines.

Perspectives

Cependant, il y a une amélioration à faire en ce qui concerne le niveau de sécurité, la structure fonctionnelle même de l'application ainsi que l'adaptabilité du code source aux exigences de la nouvelle technologie web (Web 2.0). Pour mener à bien ce travail et produire le code final prêt à une utilisation optimale, le groupe BOLLORE devra mettre à notre disposition des ressources matérielles nécessaires :

- Un ordinateur aux normes actuelles
- Un accès permanent à Internet pour la recherche de l'aide et outils utiles.
- Un contrat nous engageant dans ce projet
- Et bien d'autres outils pouvant nous aider dans la réalisation effective de ce travail.

ANNEXE

- ***Enregistrement d'un nouvel utilisateur (action_login).***

```
public function action_register()
{
      //  Nous chargeons le formulaire d'enregistrement
      $view = View::factory('publicview/register')
      //  Nous utilisons la variable notice pour afficher les erreurs sur le View
      ->bind('notice', $notice);

        // Est ce que l'utilisateur est déjà connecte ?
        if (Auth::instance()->logged_in()) {

            // Si Oui, charger le dashboard
            Request::instance()->redirect('dashboard/index');
        }

        if ($_POST) {    // Est ce qu'il y a une requete POST

            $_POST['logins'] = time();
            $user = new Model_User;
            $post = $user->validate_create($_POST);
            if($post->check()) {
                $user->values($post);
                $user->save();
                $user->add('roles', ORM::factory('role')->find(1));

                $userlogin = ORM::factory('user');
                $status = $userlogin->login($_POST);
            } else {
                $notice = $post->errors('validate');
            }
        }
        $this->template->left = View::factory('publicview/leftside');
        $this->template->right = $view;
        $this->template->content = View::factory('publicview/headerview');
    }
```

- ***Connexion d'un compte utilisateur***

```
public function login(array & $array, $redirect = FALSE)
{
        $array = Validate::factory($array)
            ->filter(TRUE, 'trim')
            ->rules('username', $this->_rules['username'])
            ->rules('password', $this->_rules['password']);

        // Nous commencons avec un etat de la connexion a FAUX (NON)
        $status = FALSE;

        if ($array->check())
        {
            // Chargeons l'entite  User de la base de donnees
            $this->where('username', '=', $array['username'])->find();
```

```
            if ($this->loaded() AND Auth::instance()->login($this,
$array['password']))
            {
                if (is_string($redirect))
                {
                    // Redirection du client après une connexion reussie
                    Request::instance()->redirect($redirect);
                }

                // Connexion reussie. Retour de l'etat de connexion TRUE
                $status = TRUE;
            }
            else
            {
                $array->error('username', 'invalid');
            }
        }

        return $status;
    }
```

- ***La classe User***

```
class Model_User extends Model_Auth_User{

    protected $_labels = array(
        'firstname' => 'First Name',
        'username' => 'Username',
        'email' => 'Email Address',
        'password' => 'Password',
        'password_confirm' => 'Confirm Password',
    );

    public function validate_create($array)
    {
        $array = Validate::factory($array)
            ->rules('username', $this->_rules['username'])
            ->rules('firstname', $this->_rules['firstname'])
            ->rules('password', $this->_rules['password'])
            ->rules('password_confirm', $this->_rules['password_confirm'])
            ->rules('email', $this->_rules['email']);

        foreach ($this->_callbacks as $key => $value) {
            foreach ($value as $validator) {
                $array->callback($key, $array($this, $validator));
            }
        }

        return $array;
    }
}
```

- ***Code source fichier index.php de Kohana***

```
<?php
$application = 'application';
```

```
$modules = 'modules';
$system = 'system';
define('EXT', '.php');

error_reporting(E_ALL | E_STRICT);

// Set the full path to the docroot
define('DOCROOT', realpath(dirname(__FILE__)).DIRECTORY_SEPARATOR);

// Make the application relative to the docroot
if ( ! is_dir($application) AND is_dir(DOCROOT.$application))
    $application = DOCROOT.$application;

// Make the modules relative to the docroot
if ( ! is_dir($modules) AND is_dir(DOCROOT.$modules))
    $modules = DOCROOT.$modules;

// Make the system relative to the docroot
if ( ! is_dir($system) AND is_dir(DOCROOT.$system))
    $system = DOCROOT.$system;

// Define the absolute paths for configured directories
define('APPPATH', realpath($application).DIRECTORY_SEPARATOR);
define('MODPATH', realpath($modules).DIRECTORY_SEPARATOR);
define('SYSPATH', realpath($system).DIRECTORY_SEPARATOR);

// Clean up the configuration vars
unset($application, $modules, $system);

if (file_exists('install'.EXT))
{
    // Load the installation check
    return include 'install'.EXT;
}

// Define the start time of the application
define('KOHANA_START_TIME', microtime(TRUE));

// Load the base, low-level functions
require SYSPATH.'base'.EXT;

// Load the core Kohana class
require SYSPATH.'classes/kohana/core'.EXT;

if (is_file(APPPATH.'classes/kohana'.EXT))
{
    // Application extends the core
    require APPPATH.'classes/kohana'.EXT;
}
else
{
    // Load empty core extension
    require SYSPATH.'classes/kohana'.EXT;
}

// Bootstrap the application
require APPPATH.'bootstrap'.EXT;
```

GLOSSAIRE

IUT : Institut Universitaire de Technologie

DUT : Diplôme Universitaire de Technologie

BTS : Brevet de Technicien Supérieur

GI : Génie Informatique

CV : Curriculum Vitae

UML : Unified Modelling Language

SI : Système d'Information

PME : Petite et Moyenne Entreprise

DRH : Direction des Ressources Humaines

IEEE: Internetworking of Electrical and Electronics Engineers

PU: Processus Unifie

RUP: Rational Unified Process

ACM : Association of Computing Machinery

CMR: Cameroon

SGBD: Système de Gestion des Base de Données

IDE : Integrated Development Environnement

Java : langage de programmation

C# : Langage de programmation breveté Microsoft

C+ : Langage de programmation

PHP : Langage de programmation

HTML : HyperText Transfer Protocol

ASP : Active-Server Page

AJAX : Asynchronous Javascript and XML

MVC : Model View Controller

PDO : PHP Data Objects

XSS : Cross-Site Scripting

ORM: Object-Relational Mapping

SQL: Structured Query Language

T-SQL: Transaction Structure Query Language

CLR: Common Language Runtime

LDAP: Lightweight Directory Access Protocol

XML: eXtensible Markup Language

GPL: General Public License

CSS3: Cascading Style Sheet

LAN: Local Area Network

WAN: World

TCP: Transfer Control Protocol

IP: Internet Protocol

OSI: Organisation des Standards Internationaux

BIBLIOGRAPHIE

[1] Nathalie LOPEZ, Jorge MIGUEIS, Emmanuel PICHON –***Intégrer UML dans vos projets*****.** Editions Eyrolles, 1999, 248 pages.

[2] A. LEFEBRE. – ***Web client-serveur***. ***Le triomphe du client léger***. Editions Eyrolles, 1998, 250 pages.

[3] T. ANDRO, J.-M. CHAUVET. – ***Objet métier***. Editions Eyrolles, 1998, 240 pages.

[4] Robert OGOR., ***Modélisation avec UML. ENST Bretagne***., Amazon, Mai 2003, 92 pages.

[5] Joseph CORTISARD, ***De UML à la base de données***. CNAM de Paris. 2006. 41 pages.

[6] FEUTO NJONKO Paul Brillant, ***Démarche des spécifications des architectures logicielles selon l'approche MDA : Application aux plates formes de datamining***, Mémoire de DEA, Université de Yaoundé I, HARMATTAN, Juin 2009, 53 pages.

Webographie

[1] http://uml.developpez.com/. (Juillet 2010)

[2] http://laurent-audibert.developpez.com/Cours-UML/html/Cours-UML.html. (Août 2010)

[3] http://laurent-piechocki.developpez.com/uml/tutoriel/lp/cours/. (Octobre 2010)

[4] http://www.misfu.com/information-cours-tutos-tutoriaux-Conception.html. (Décembre 2010)

[5] http://www.commentcamarche.net. (Janvier 2011)

[6] http://hosteddocs.ittoolbox.com/RP092305.pdf (Mars 2013)

[7] http://www.methodsandtools.com/archive/archive.php?id=32 (Mars 2013)

[8] http://fdigallo.online.fr/cours/uml.pdf (Mars 2013)

[9] http://thieum22.free.fr/Quest_RUP.htm (Mars 2013)

[10] http://kadreg.org/dotclear/index.php?post/2007/10/28/60-rup-methode-utilisant-uml (Mars 2013)

[11] http://fr.wikipedia.org/wiki/Unified_Process (Mars 2013)

[12] http://fr.wikipedia.org/wiki/Unified_Process (Mars 2013)

[13] http://www.commentcamarche.net/contents/merise/concintro.php3 (Mars 2013)

[14] http://fr.wikipedia.org/wiki/Zend_Framework (Mars 2013)