Ministère de l'enseigneMent supérieur
Université du 7 Novembre à Carthage

RAPPORT DE PROJET DE FIN D'ETUDES
POUR L'OBTENTION DU DIPLOME DE LA LICENCE

Filière: Informatique Appliquée

Réalisation d'une application Web CRM

Entreprise: DISCOVERY INFORMATIQUE

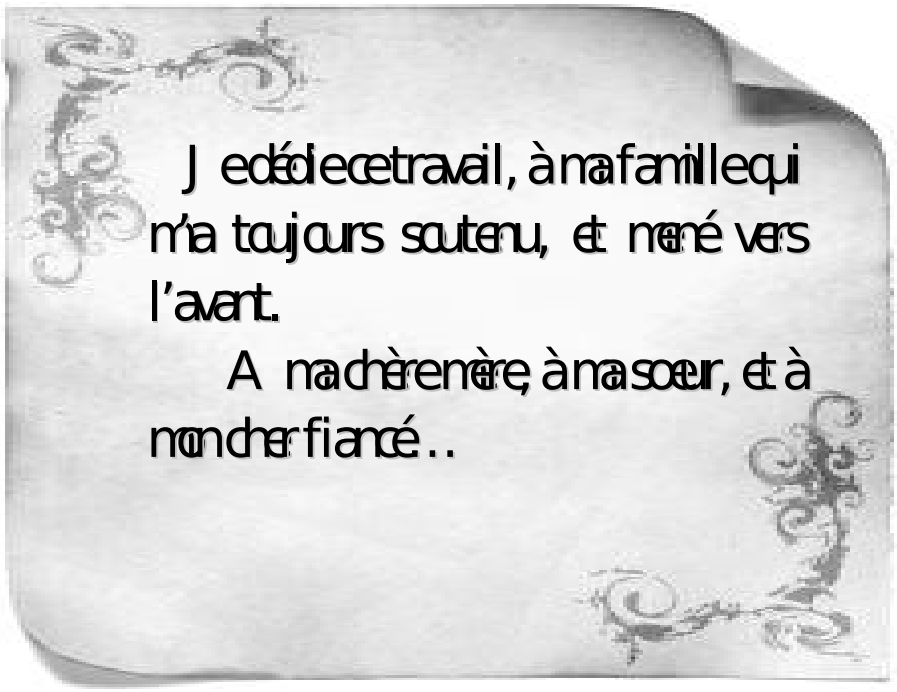
Réalisé par : **Mlle. Nassim Hssairi**
M. Med Jihed Belhadj

Sous la direction de : **M. Aymen Bouchhima**
M. Elfaleh Tarek

2009 - 2010

DEDICACES DE BELHADJ MED JIHED

DEDICACES DE HSSAIRI NASSIM



Remerciement

C’est un plaisir et un moment très agréable de rendre hommage et de formuler des remerciements aux personnes qui, d’une manière ou d’une autre, ***ont apporté leur soutien et contribué à finaliser ce travail.***

Nous tenons, tout d’abord, à exprimer nos sincères gratitudes envers notre encadreur Mr Aymen BOUCHHIMA, pour les conseils qu’il n’a cessé ***s de*** *nous prodiguer, sa compréhension et la confiance qu’il a toujours témoignée à* ***notre égard.***

Nos sincères remerciements et notre profond respect à tout le pe de Discovery Informatique qui nous ont accueilli, fourni le cadre néces la *réalisation de notre projet de fin d’études, en nous faisant partager leur* ***grande expérience et en se rendant toujours disponibles malgré leur***

En particulier Mr Karim GHORBEL, responsable financier qui nous *donné l’opportunité d’effectuer ce sta****ge.***

Nous remercions aussi Mr Tarek EL FALEH, responsable activité, d nouvelles technologies.

Nous remercions également Mr Ghazi BOUJNAH pour son estimab *collaboration et le soutien qu’il nous a apporté.*

Nous adressons également nos sincères remercieme*nts à l’ensem* ***monsieurs les membres du jury.***

Nous associons à ces remerciements tous les enseignants qui on *contribué à notre formation ainsi qu’à toutes les personnes qui travaillent à l’ESTI****.***

Table des matières

Introduction .....

Chapitre I. Présentation Générale .....
I.1. Introduction .....
I.2. Présentation de l’organisme d’accueil .....
I.2.1 Les outils techniques utilisés par Discovery .....
I.2.2 Organigramme .....
I.3. Concepts Customer Relationship Management (Gestionnaires de R
I.3.1 Définition d’un CRM .....
I.3.2 Pourquoi utiliser une solution CRM .....
I.3.3 Fonctionnalités d'un CRM .....
I.4. Présentation du sujet .....
I.5. Conclusion .....

Chapitre II. Analyse des besoins et spécifications .....
II.1. Introduction .....
II.2. Objectif .....
II.3. Etude de l’existant .....
II.3.1 Description de l’existant informatique .....
II.4. Spécification des exigences .....
II.4.1 Liste des exigences .....
II.4.2 Scénarios et des cas d’utilisation « module recrutement » .....
II.4.3 Scénarios et des cas d’utilisation « module espace employées
II.5. Conclusion .....

Chapitre III. Conception .....
III.1. Introduction .....
III.2. Conception .....
III.2.1. Module « Recrutement » .....
III.2.2. Module «Espace Employée » .....
III.3. Structure de la base de donnée.....
III.3.1. Dictionnaire des données de la base des données .....
III.4. Conclusion .....

Chapitre IV. Etat de l’art .....
IV.1. Introduction .....
IV.1.1. Présentation du SugarCRM .....
IV.2. Architecture de l’application « MVC » .....
IV.3. Comparaison entre les technologies CRM .....
IV.4. Conclusion .....

Chapitre V. Réalisation .....
V.1. Introduction .....
V.2. Environnement de travail .....
V.2.1. Environnement matériel .....
V.2.2. Environnement logiciel .....
V.3. Choix techniques .....
V.3.1. Choix du langage .....
V.3.2. Choix de la technologie de sécurité .....
V.4. Gestion du projet .....
V.5. Phase d'implémentation .....
V.5.1. Contraintes .....
V.5.2. La programmation avec Sugar et MVC .....
V.6. Phase de tests et validation .....
V.7. Conclusion .....

Chapitre VI. Interface de l'application .....
VI.1. Introduction .....
VI.2. Interfaces de l'application .....
VI.3. Conclusion .....

Conclusion Générale .....

Bibliographie .....

Netographie. .....

Annexe A : Vue d'ensemble sur les différents répertoires et fichier de Su

Annexe B : Le développement du moteur de recherche des candidats ..

Table des figures

Figure 1. Structure et hiérarchie de l'organisation .....
Figure 2. Diagramme de contexte .....
Figure 3. Diagramme des cas d'utilisation globale.....
Figure 4. Diagramme des cas d'utilisation « Gestion de la planification p
Figure 5. Diagramme des cas d'utilisation « Gestion de recrutement » ..
Figure 6. Diagramme des cas d'utilisation «Gestion de proposition d'un
Figure 7. Diagramme des cas d'utilisation «Gestion de la saisie des cv»
Figure 8. Diagramme du contexte.....
Figure 9. Diagramme Cas Utilisateurs globale .....
Figure 10. Diagramme Cas Utilisateurs Gestion d'affectation des emplo
Figure 11. Diagramme Cas Utilisateurs des taches employées .....
Figure 12. Cas Utilisateur de gestion des réclamations .....
Figure 13. Diagramme de classes .....
Figure 14. Diagramme d'activité relatif au cas d'utilisation globale de sy
Figure 15. Diagramme d'activité relatif au cas d'utilisation «Gestion de
Figure 16. Diagramme de séquences relatif au cas d'utilisation «Gestion
Figure 17. Diagramme d'activité relatif au cas d'utilisation « Gestion de
Figure 18. Diagramme de séquences relatif au cas d'utilisation «Gestion
Figure 19. Les éléments impliqués dans le MVC de SugarCRM.....
Figure 20. La navigation entre les modules .....
Figure 21. Module Builder fait partie de la console «Developer Tools» de
Figure 22. L'utilisateur commence par créer et nommer le nouveau pac
Figure 23. Les répertoires différents à l'intérieur du SugarCRM .....
Figure 24. L'architecture MVC d'une application web .....
Figure 25. Interface d'authentification .....
Figure 26. Chronologique de la réalisation de notre projet .....
Figure 27. Intégré un module dans SugarCRM .....
Figure 28. Recherche d'un candidat .....
Figure 29. Sélection d'un élément de compétence .....
Figure 30. Résultat .....
Figure 31. Vérification du résultat.....
Figure 32. Ajouter un candidat .....
Figure 33. Déposer les cv et voir historique cv.....
Figure 34. Interface remplir Timesheet .....
Figure 35. Remplir fiche d'intervention .....

Table des Tableau

Tableau 1. Dictionnaire des données .....
Tableau 2. Liste des tests pour valider notre application .....

Introduction

Comme nous vivons dans un monde où le marché informatique se développe très rapidement et les technologies se perfectionnent, le besoin d'informatiser les tâches internes dans les entreprises ne cesse d'accroître. Et ce, pour faciliter les manipulations et gagner du temps. Les CRM (Customer Relationship Management) se trouvent au cœur même de cette évolution, qui modifie de manière significative, la gestion des systèmes d'information et la conception des applications.

Dans cette perspective, et dans le cadre de notre projet de fin d'études, réalisé au sein de l'entreprise Discovery Informatique, nous allons nous intéresser au développement d'une solution de gestion automatisée de candidatures au recrutement et de l'espace employée afin de l'intégrer dans le portail CRM de l'entreprise.

Et dans ce contexte, nous allons procéder comme suit :

Chapitre1 : Présentation générale

Dans ce chapitre, nous présentons l'entreprise d'accueil, la problématique à résoudre et quelques concepts liés au CRM.

Chapitre 2 : Analyse des besoins et spécifications

Ce chapitre détaille la spécification générale de notre futur système avec Identification des différentes fonctionnalités de l'application. Il présente notamment différents cas d'utilisation du projet de point de vue utilisateur.

Chapitre 3 : Conception

Ce chapitre présente la conception générale et détaillée de notre futur système.

Chapitre 4 : Etude de l'art

Dans ce chapitre on met l'accent sur les différentes technologies existantes utilisées pour l'élaboration du projet (pendant la réalisation).

Chapitre 5 : Réalisation

Dans ce chapitre nous présentons les étapes suivies jusqu'à l'implémentation finale de notre système.

Chapitre 6 : Interface de l’application

Ce chapitre présente les principaux choix ergonomiques adoptés en présentant quelques interfaces graphiques de l’application.

Chapitre I : Présentation Générale

I.1. Introduction

Dans ce chapitre, nous commençons par la présentation de l'organisme accueillant. Puis, nous entamons les différentes activités ainsi que le cadre de travail qui englobe également l'environnement matériel et logiciel du projet.

I.2. Présentation de l'organisme d'accueil

Discovery Informatique est une société du groupe « Discovery Datasoft Group ». C'est une SSII Spécialisée dans l'ingénierie logicielle et l'intégration de solutions de gestion intégrées ERP et de solutions de gestion d'entreprise.

Créée en 1993, Discovery est aujourd'hui leader dans son domaine d'activité et la première certifiée ISO 9001 sur le marché Maghrébin. Discovery opère également sur le marché Européen en menant des projets en sous-traitance ou en délégation de compétences avec une expertise en gestion de projet et une démarche qualité.

I.2.1. Organigramme

La figure 1 présente l'organigramme de la société. Nous sommes intervenus au niveau de l'unité « Nouvelles technologies » dont la vocation est d'intégrer les nouvelles technologies au sein de la solution informatique proposée par la société.

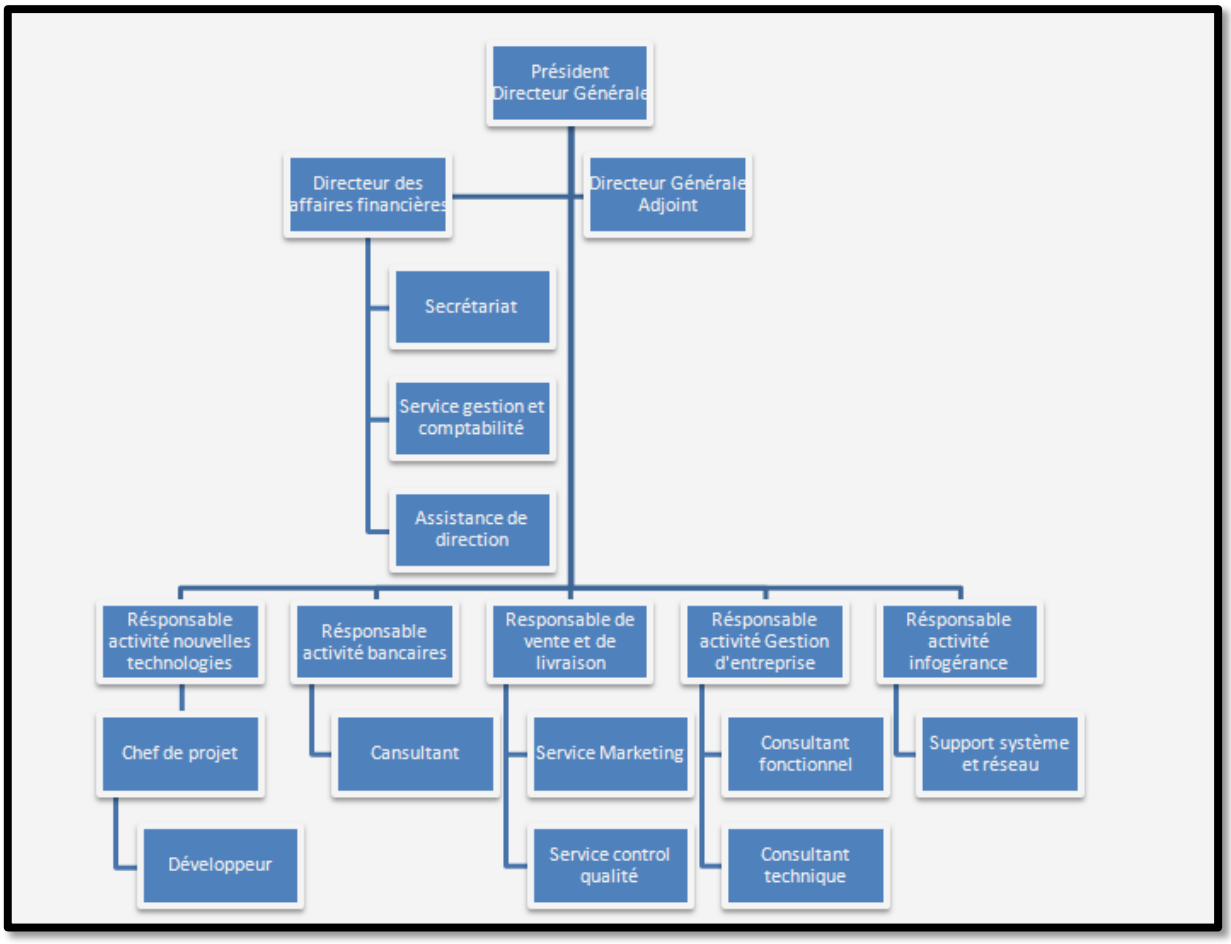


Figure 1. Structure et hiérarchie de l'organisation

I.3. Concepts Customer Relationship Management (Gestionnaires de Relation Client)

Associées aux techniques de marketing modernes, les entreprises ne peuvent plus échapper aux technologies et outils du moment à savoir dans notre cas les Gestionnaires de Relation Client (GRC) dont le terme le plus couramment utilisé est CRM pour Customer Relationship Management.

Dans un premier temps, nous allons définir ce qu'est un CRM. Puis, nous verrons en détail les principales fonctionnalités de cet outil. Enfin, nous observerons les différents acteurs du marché des CRM.

I.3.1. Définition d'un CRM

Un CRM, soit Customer Relationship Management ou encore GRC pour Gestion de la Relation Client, est un outil révolutionnaire dans le domaine du marketing. En effet, un CRM dépasse l'approche marketing traditionnelle focalisée sur la relation entre un produit/service et son client ou encore la notoriété et l'image de marque.

Il existe de nombreuses fonctionnalités offertes par le CRM telles que la gestion des prospects, gestion des clients et de leurs achats, etc. Il est possible pour l'entreprise de personnaliser l'offre, sous forme de publicités ou encore d'e-mails afin de fidéliser au maximum le client, d'améliorer sa satisfaction et par la même occasion d'accroître l'efficacité des employés.

I.3.2. Pourquoi utiliser une solution CRM

De nombreuses sociétés et organismes de secteurs d'activité variés font appel à des spécialistes pour mettre en place une solution CRM.

La mise en place d'une solution CRM au sein d'une entreprise est donc une démarche résolument stratégique basée sur le désir d'augmenter la satisfaction du cli ent et avec pour conséquence la croissance de l'activité et des profits.

I.3.2.1 Avantages d'un logiciel CRM en ligne

- Tous les employés peuvent y accéder en même temps, quelque soit leurs emplacements.
- Pas de problème en cas de panne matériel d'un ou de plusieurs ordinateurs (logiciel externe).
- Interface unifiée, regroupant toutes les fonctionnalités nécessaires à l'activité.

I.3.3. Fonctionnalités d'un CRM

I.3.3.1 Module Vente

Le Module Vente gère tous les aspects de la vente (relation client) :

- Informations principales d'un client ou d'un prospect.
- Mise à jour et suivi des contrats et des affaires en cours.

- Gestion des références produits et des tarifs.
- Organisation du plan de vente.
- Gestion du cycle de vie d'un client.

I.3.3.2 Module Marketing

Le module Marketing analyse et gère le comportement des clients en fonction des différentes méthodes de communication :

- Envoi de documents (fax, e-mail, papier, sms) en masse.
- Sauvegarde de documents (commercial/marketing).
- Suivi et analyse des secteurs de chaque vendeur
- Analyse de l'entreprise par rapport à la concurrence.
- Mise à jour et suivi du site internet.
- Gestion des appels téléphoniques.
- Analyse de la transmission de l'information dans l'entreprise.

I.3.3.3 Module Gestion

Le Module Gestion gère au sein de l'entreprise l'ensemble de la documentation :

- Stockage des documents.
- Analyse et suivi des événements.
- Import/export d'une base de données informatique.
- Gestion du courrier électronique.
- Agenda mutualisé de l'entreprise.
- Indicateurs sur le fonctionnement de l'entreprise.

I.3.3.4 Module Service

Le Module Service regroupe les services proposés par les éditeurs de CRM :

- Gestion des commissions vendeurs.
- Gestion de plusieurs devises.
- Hiérarchie sur les droits d'accès.

- Gestion du partage de l'information.
- Possibilité de travailler sur PC portable.
- Possibilité de travailler lorsque le système n'est pas connecté.
- Association possible avec d'autres logiciels.
- Modification possible des champs de données.

I.4. Présentation du sujet

Notre sujet de projet de fin d'étude consiste à étendre une application Web CRM basée sur la plateforme SugarCrm en ajoutant deux modules CRM.

Le premier est un module de recrutement. Il représente un sujet d'actualité qui commence à se propager dans plusieurs entreprises informatisées, on le retrouve aussi, dans les média interactifs de recrutement comme keejob.com.

Le deuxième module représente un sujet primordial dans chaque entreprise qui est la gestion d'un groupe d'employées (fiche d'intervention, timesheet) et ces projets ont pour objectif afin d'augmenter la fiabilité de ce module, d'assurer des gains de temps et de contrôler les intervenants de suivre les affaires en cours ainsi que la répartition des coûts et du temps entre elles et aussi de donnés un accès au client pour voir le calendrier de chaque employée de champ.

I.5. Conclusion

Dans ce chapitre nous avons présenté l'organisme d'accueil ainsi que ses différentes activités et nous avons également présenté le sujet de notre projet. Notre sujet sera expliqué plus en détaille dans le chapitre suivant, en exposant les besoins et les objectifs à atteindre.

Chapitre II. Analyse des besoins et spécifications

II.1. Introduction

Dans ce chapitre, nous présentons les motivations qui sont derrières l'idée de ce projet. Nous détaillons la liste exhaustive des fonctionnalités attendues par l'application à réaliser et les objectifs visés par ce projet. Nous présentons à la fin quelques diagrammes des cas d'utilisation pour la modélisation des fonctionnalités de notre application.

II.2. Objectif

Notre projet consiste à étendre une solution CRM existante est accessible à travers le portail web de Discovery en ajoutant deux modules.

Notre tâche consiste à trouver une solution pour rendre facile et rapide une recherche spécifique d'un profil de candidat, pour un éventuel recrutement. Nous allons donc gérer des candidatures des différentes ressources, en fonction de leurs compétences de leurs diplômes de leurs certifications et de leur disponibilité.

Ces deux solutions de gestion de candidature au recrutement seront intégrées à l'ensemble de fonctionnalités qu'offre SUGAR CRM.

D'autre part la deuxième solution (Gestion des employées) se compose de deux modules Fiche intervention et Timesheet qui répondent à la demande mise par la direction générale (Mr. Elfaleh Tarek), de disposer d'un module lui permettant :

- De minimiser les coûts, la perte de temps et assurer la disponibilité de main d'œuvre.
- De garder une trace électronique des projets ou des tâches effectuées par les intervenants en automatisant ces actions et en consultant l'état d'avancement de chaque tâche faite chez les clients.
- De facilité le travaille de chaque chef de projet pour gérer leurs équipes pour voir les employées disponibles dans une date fixe.

 De garantir la sécurité en limitant l'accès pour chaque participant, à un ensemble de procédures et d'activités, en fonction des rôles qu'il tient.

Sachant que ces modules CRM sera mis en place dans le portail de Discovery ; ce projet entre dans le cadre d'extension de solutions pour la gestion de la relation avec le client.

II.3. Etude de l'existant

Nous allons, tout d'abord, étudier l'existant informatique et fonctionnel de Discovery. Cette étude nous permettra de déployer progressivement le projet et de ressortir quelques critiques, à partir desquelles nous allons expliciter la solution adéquate à implémenter.

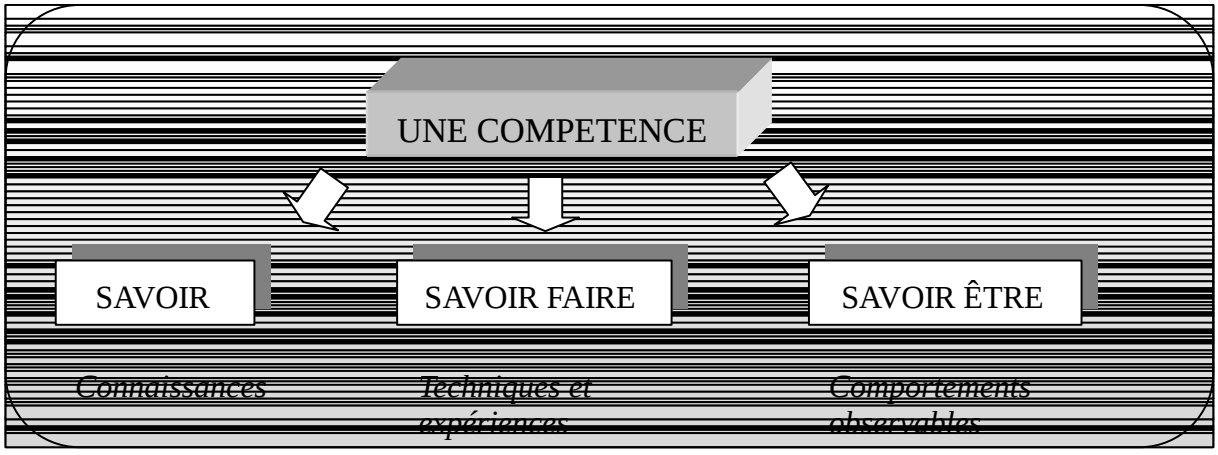
II.3.1. Description de l'existant informatique

Discovery a adopté le logiciel SugarCRM et a décidé de créer des modules sur-mesure qui répondent à ses besoins.

Et après plusieurs années de travail, les activités de DISCOVERY ont évolué et le nombre de ses projets et de ses activités est devenu très important. Pour réussir, la société a toujours su s'entourer de compétences à fort potentiel. Suite à sa grande et large activité, elle recourt souvent au recrutement pour occuper de nouveaux emplois.

Tout emploi existe pour produire des résultats, et pour produire des résultats, il faut mobiliser de la compétence.

Une compétence est l'ensemble de trois composants mises en œuvre dans une situation donnée pour produire un résultat : un savoir, un savoir faire et un savoir être.



C'est pourquoi le facteur humain représente une ressource importante pour la survie et le suivi des projets de l'entreprise.

Pour chaque mission, Discovery a besoin de profils spécifiques de candidats. Un responsable de recrutement affecte une mission à un des employés en se basant sur sa connaissance des compétences et expériences de ses employés.

Cette opération s'effectue oralement mais si la recherche n'est pas aboutissant ou la personne correspondante au profil recherché n'est pas disponible, Discovery lance des appels d'offre, fait de la publicité et emploie d'autres moyens afin de déceler le bon profil des candidats.

Avant on utilisait des simples offres d'emploi dans la presse et en envoyait par postes les candidatures et donc c'était apaisant de rechercher dans tous ces documents le profil qu'on recherche exactement.

Et si les activités, les projets, les clients ont bien évolué de plus le nombre des personnelles aussi est devenue très important pareillement pour la gestion des tâches des employés, avant on travaillait sur des papiers sur lesquels chaque employées écrit en détaille tout le travail qu'il a réalisé mais cela n'était pas pratique vu qu'on ne pouvait pas savoir son avancement en temps réel.

Cette opération s'effectue chaque jour oralement ou d'une manière téléphonique et ce processus a besoins d'être automatiser pour faciliter les tâches de tous les acteurs de ce processus.

- **Le client** : appelle le Service technique de Discovery pour passer une réclamation.
- **Service Technique** : appelle le chef de projet et lui informe de la réclamation du client.
- **Le chef de projet** : doit proposer une solution et affecter un consultant technique qui soit disponible
- **Le consultant technique** : se dirige aux clients et fait sa mission proposée et retourne à Discovery à la fin de la journée pour remplir sa fiche d'intervention.

Et pour résoudre tout ces problèmes on va intégrer ces modules CRM pour donner une solution automatiser et fiable qui répond à tous ces besoins.

II.4. Spécification des exigences

II.4.1. Liste des exigences

Dans cette section, nous identifions une liste des exigences fonctionnelles et non fonctionnelles du projet.

II.4.1.1 Exigences fonctionnelles

- Une solution rapide et efficace pour lancer des recherches automatiques.
- Maîtriser l'allocation des compétences et connaissances de son personnel et des candidats externes à l'entreprise.
- Gestion de l'historique de tous les CV pour garder des traces de ces derniers.
- Gérer les emplois du temps de chaque personne travaillant à Discovery
- Informatiser la tâche fiche d'intervention.
- Gérer le déroulement de chaque tâche du projet.
- Gérer les réclamations des clients.

II.4.1.2 Exigences non fonctionnelles

- Garder sa clientèle et les fidéliser en les satisfaisant au mieux.
- Elargir sa gamme de clients en offrant des services rapides et faciles à manipuler.
- Gagner du temps à l'intérieur de l'entreprise.
- Eviter les coûts et pertes engendrées par l'ancien moyen de recrutement.
- Gagner en terme d'allocation de ressource, dans le cas par exemple où l'on affecte la bonne mission à chaque profil, chacun effectuera au mieux sa tâche et apportera plus de solutions intéressantes à son projet et donc à l'Entreprise.

II.4.2. Scénarios et des cas d'utilisation « module recrutement »

II.4.2.1 Identification des acteurs

La mise en marche du système nécessite essentiellement trois acteurs :

- **Gestionnaire des candidats** : il se représente comme un membre du personnel de Discovery, il est chargé de gérer les ressources humaines. Son rôle consiste à affecter chaque nouvelle candidature ou un employé de Discovery au poste qui le convient.

- **Candidat internaute** : c'est l'un des acteurs qui va interroger le système à chaque disposition d'une nouvelle candidature.

- **Client** : il a pour fonction l'autorisation d'accéder au système pour rechercher un profil anonyme dans la base des CV de Discovery.

II.4.2.2 Diagramme de contexte

Les flux représentés ci-dessous entre système et acteurs sont pris en charge par le diagramme de contexte dynamique. Ce diagramme montre une vue globale du déroulement de notre projet.

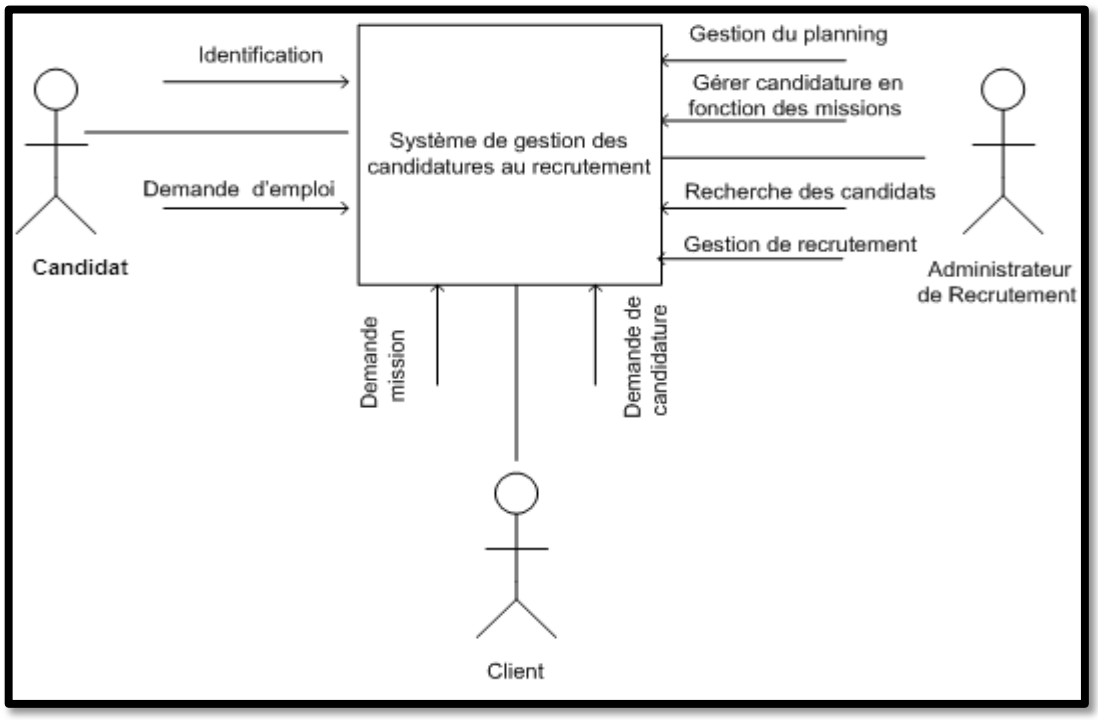


Figure 2. Diagramme de contexte

II.4.2.3 Les cas d'utilisation

Le diagramme de cas d'utilisation représente la structure des grandes fonctionnalités nécessaires aux utilisateurs du système.

C'est le premier diagramme du modèle UML, celui où s'assure la relation entre l'utilisateur et les objets que le système met en œuvre.

Un diagramme de cas d'utilisation capture le comportement d'un système, d'un sous - système, d'une classe ou d'un composant tel qu'un utilisateur extérieur le voit. Il scinde la fonctionnalité du système en unités cohérentes, les cas d'utilisation, ay ant un sens pour les acteurs.

Les cas d'utilisation permettent d'exprimer le besoin des utilisateurs d'un système, ils constituent donc une vision orientée besoins utilisateur contrairement à une vision informatique.

- *Diagramme des cas d'utilisation global :*

Le diagramme qui suit représente un cas d'utilisation globale de l'application.

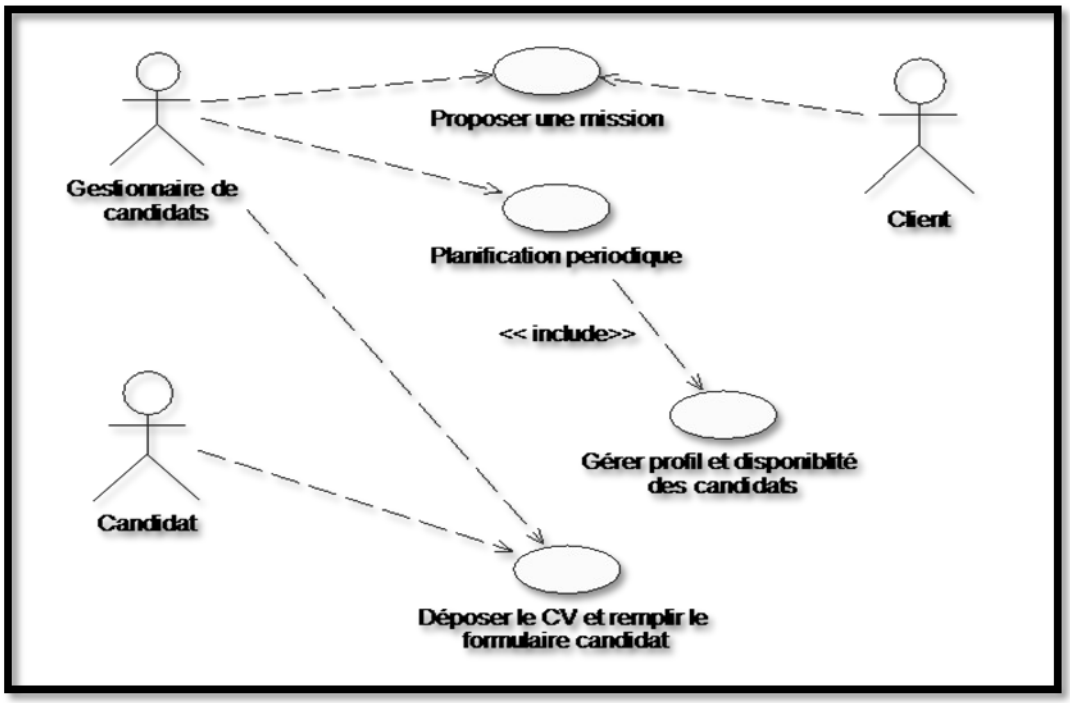


Figure 3. Diagramme des cas d'utilisation globale

- *Diagramme du cas d'utilisation « Gestion de la planification périodique » :*

Spécification : L'objectif de ce cas d'utilisation est de planifier les projets en cours de Discovery, et organiser les missions de chaque projet, et de gérer la disponibilité des employés.

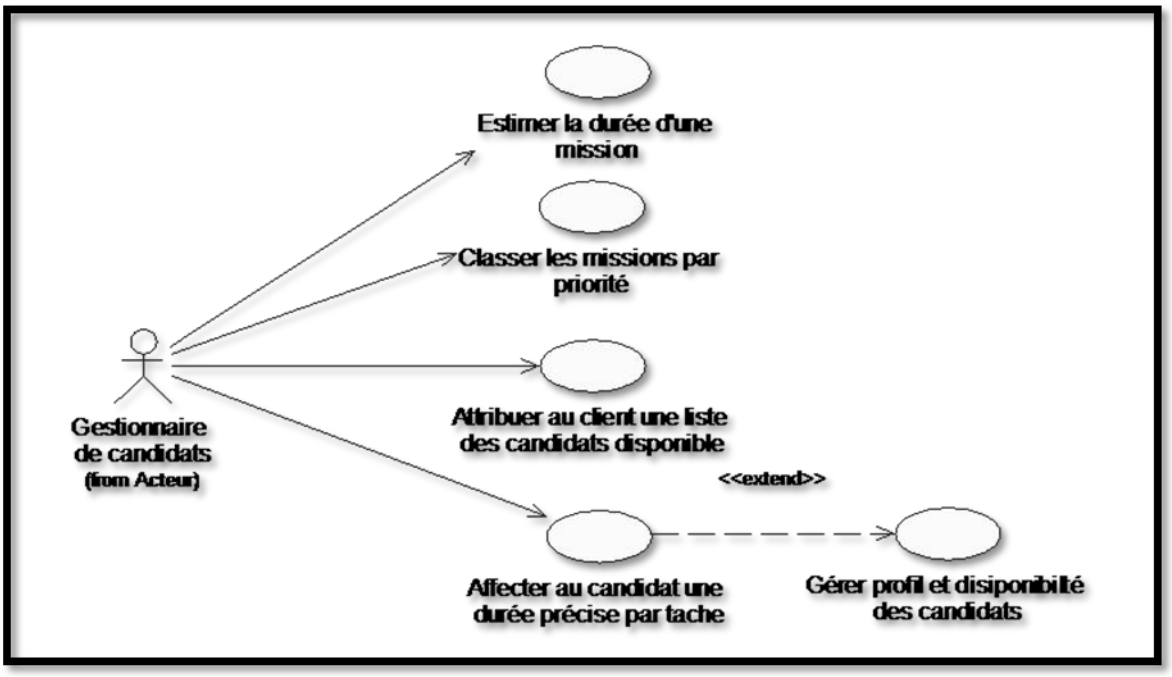


Figure 4. Diagramme des cas d'utilisation « Gestion de la planification périodique »

- *Diagramme du cas d'utilisation « Gestion de recrutement »*

Spécification : L'objectif de ce cas d'utilisation est que lors d'un éventuel recrutement, le recruteur demande au système de vérifier si une ressource disponible existe dans la base ; dans le cas contraire, il lance un appel d'offre pour déposer de nouvelles candidatures.

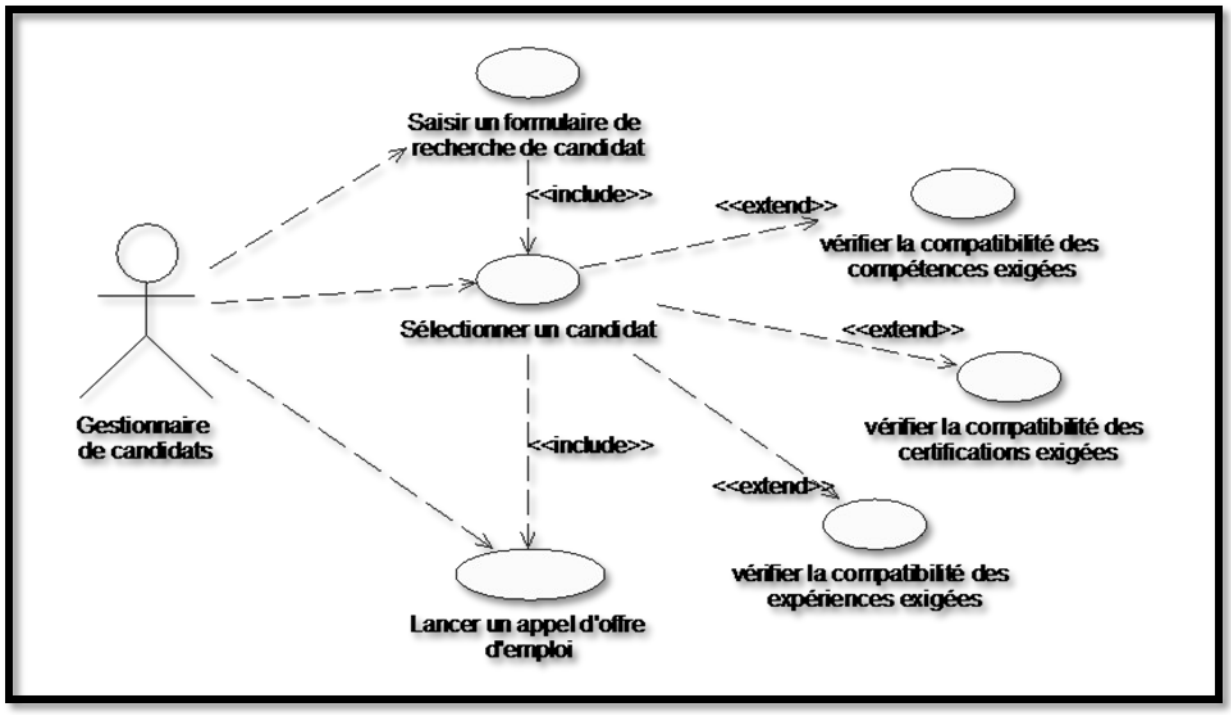


Figure 5. Diagramme des cas d'utilisation « Gestion de recrutement »

- *Diagramme du cas d'utilisation «Gestion de proposition d'une mission» :*

Spécification : L'objectif de ce cas d'utilisation est de visualiser l'interaction entre le client et notre système. Le client remplit des formulaires en ligne qui seront plus tard traités par le système.

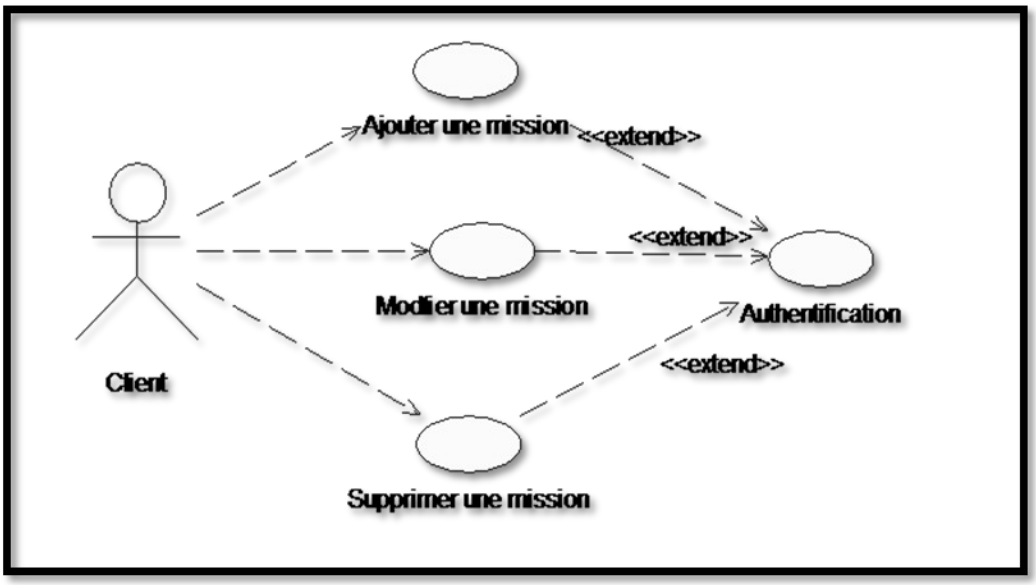


Figure 6. Diagramme des cas d'utilisation «Gestion de proposition d'une mission»

- *Diagramme du cas d'utilisation «Gestion de la saisie des cv» :*

Spécification **:** Ce cas d'utilisation représente la synchronisation entre la base du web et la base opérationnelle afin de mettre à jour nos données.

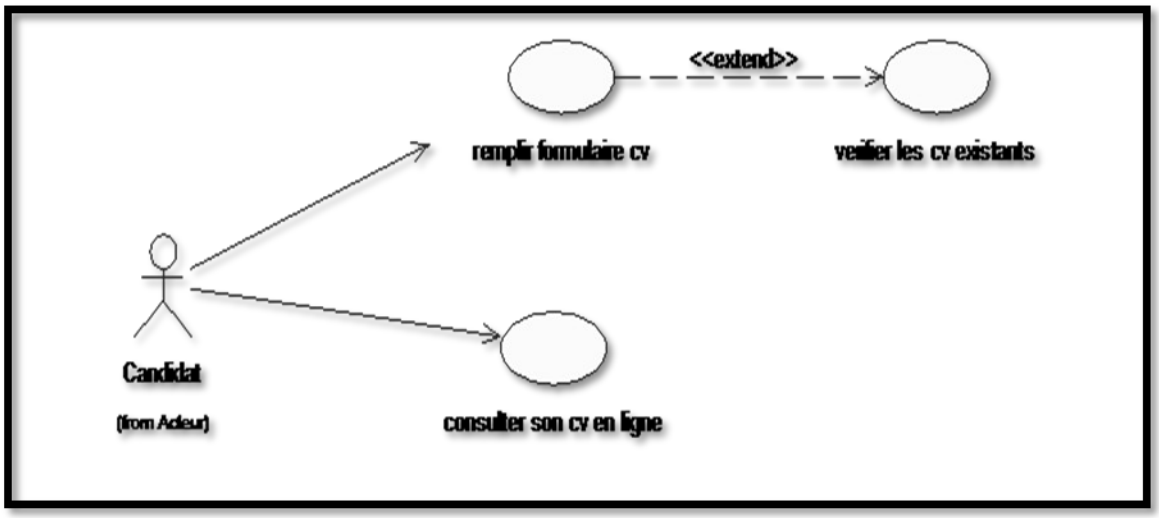


Figure 7. Diagramme des cas d'utilisation «Gestion de la saisie des cv»

II.4.2.4 Raffinement des cas d'utilisation

- *Raffinement du premier cas d'utilisation :*

Intitulé du cas d'utilisation	Gestion de la planification périodique
Acteur	Gestionnaire de candidats
Pré-condition	Localisation à l'application entant qu'Administrateur.
Post-condition	Des projets planifiés
Description du cas d'utilisation	Le gestionnaire de candidats rectifie et planifie ces projets en décrivant tous ces projets.

- *Raffinement du deuxième cas d'utilisation:*

Intitulé du cas d'utilisation	Gestion de recrutement
Acteur	Gestionnaire de candidats
Pré-condition	Localisation à l'application entant qu'Admin.
Post-condition	Recruter des candidats ou attribuer des missions aux employés
Description du cas d'utilisation	Lors d'une nouvelle mission ou projet le recruteur interroge la base de cv pour trouver le profil du candidat adéquat, sinon il lance un appel d'offre

- *Raffinement du troisième cas d'utilisation :*

Intitulé du cas d'utilisation	Gestion de proposition d'une mission
Acteur	Client
Pré-condition	Identification
Post-condition	Manipuler ces missions
Description du cas d'utilisation	Un client peut demander une mission ou mettre à jour l'une de ces missions qui existe déjà ou bien annuler l'une de ces missions.

- *Raffinement du quatriéme cas d'utilisation :*

Intitulé du cas d'utilisation	Gestion de saisie de cv
Acteur	Candidat
Pré-condition	Dépôt de nouveaux cv qu'il n'existe pas
Post-condition	Mise à jour de la base de cv
Description du cas d'utilisation	Les candidats demande un recrutement en ligne, il saisit alors un formulaire de cv. saisies les cv comme ils peuvent aussi consulter afin de mettre à jour leur cv pour toute modification.

II.4.3. Scénarios et cas d'utilisation «module espace employées»

II.4.3.1 Identification des acteurs

- **Chef de projet** : c'est un membre de Discovery , son rôle est de gérer son groupe et le déroulement de chaque projet.
- **Client** : Il a pour fonction l'autorisation d'accéder au système pour voir le déroulement de son projet et de nous contacter.
- **Service Technique** : C'est un service destiné pour les clien ts, son rôle est d'enregistrer les réclamations.
- **Employée** : c'est un employée de champ il est désigné comme un consultant technico fonctionnel, son rôle est de réparer et intégrer des logiciels chez les clients.

II.4.3.2 Diagramme de contexte

Ce diagramme montre une vue globale du déroulement de notre projet.

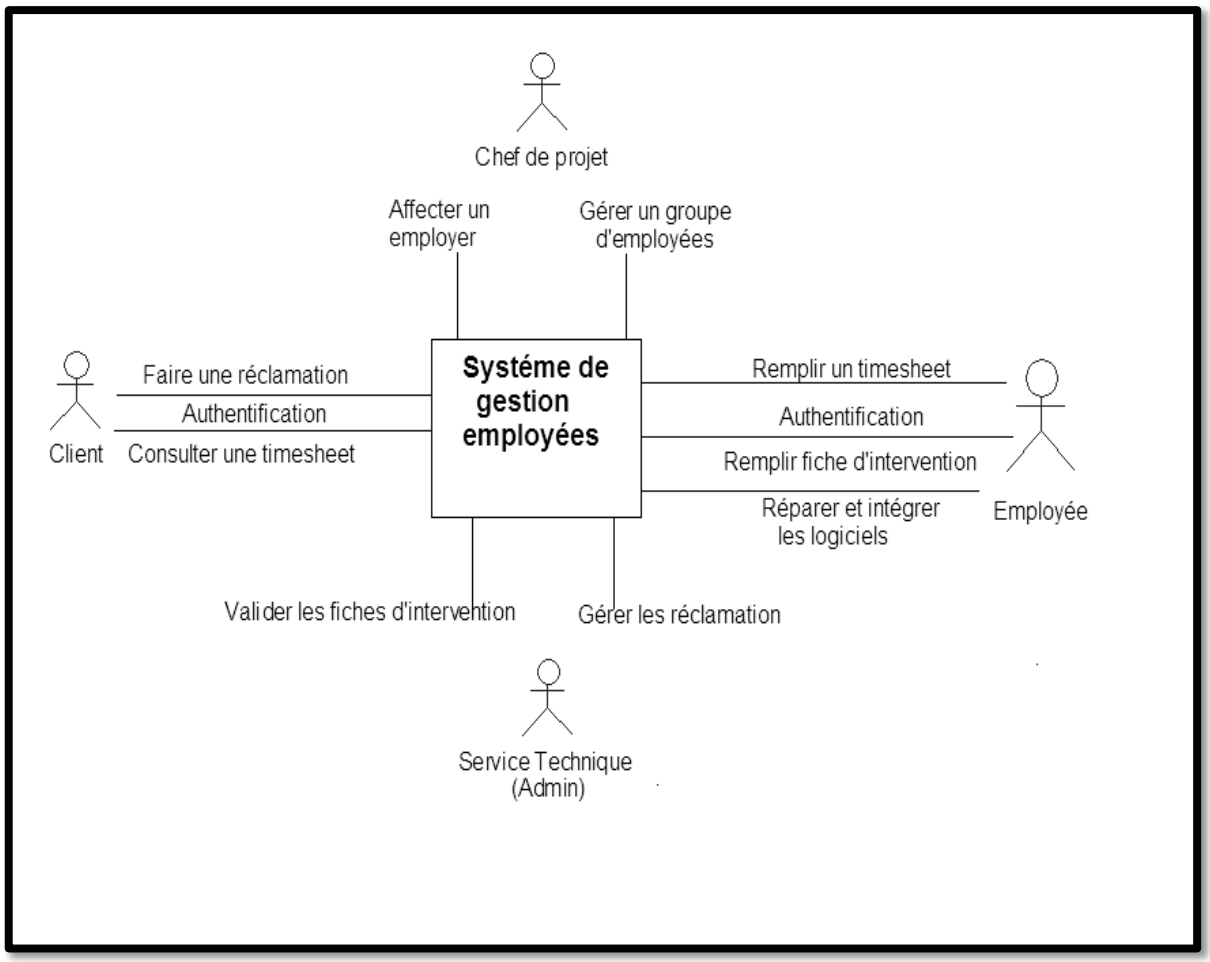


Figure 8. Diagramme du contexte

II.4.3.3 Les cas d'utilisation

Comme on a expliqué dans le module précédent, l'utilité des cas utilisations est d'exprimer les besoins des utilisateurs d'un système.

Ces diagrammes ne présentent pas des solutions d'implémentation mais ils identifient les utilisateurs du système (acteurs) et leurs interactions du système.

- *Diagramme des cas d'utilisation global :*

Le diagramme qui suit représente un cas d'utilisation globale de l'application.

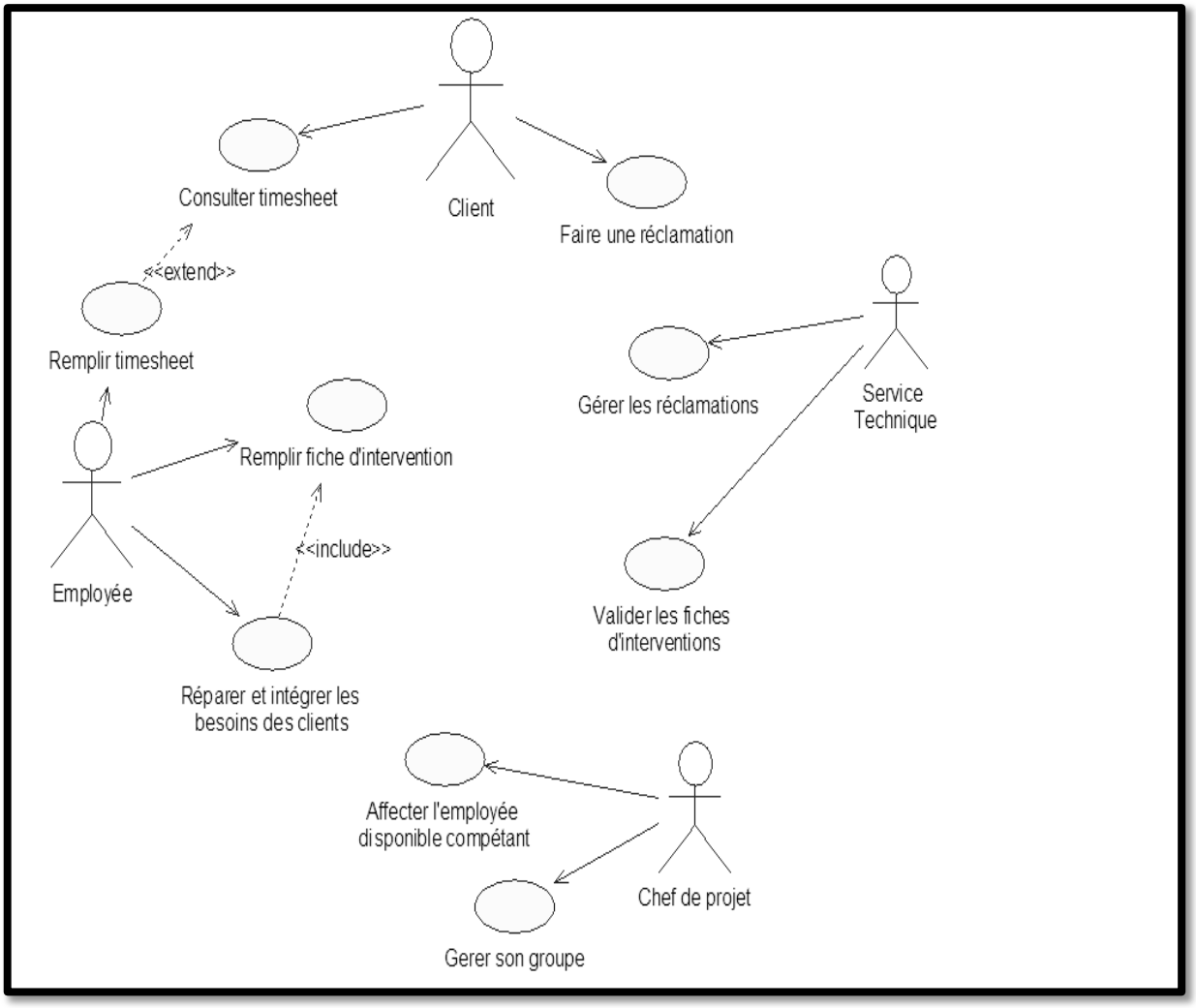


Figure 9. Diagramme Cas Utilisateurs globale

- *Diagramme Cas utilisateur Gestion d'affectation des employées :*

Spécification : Ce cas d'utilisation représente l'interaction du chef de projet et notre système, ce diagramme explique l'enchainement du cas utilisateur « Affecter l'employée disponible compétant ».

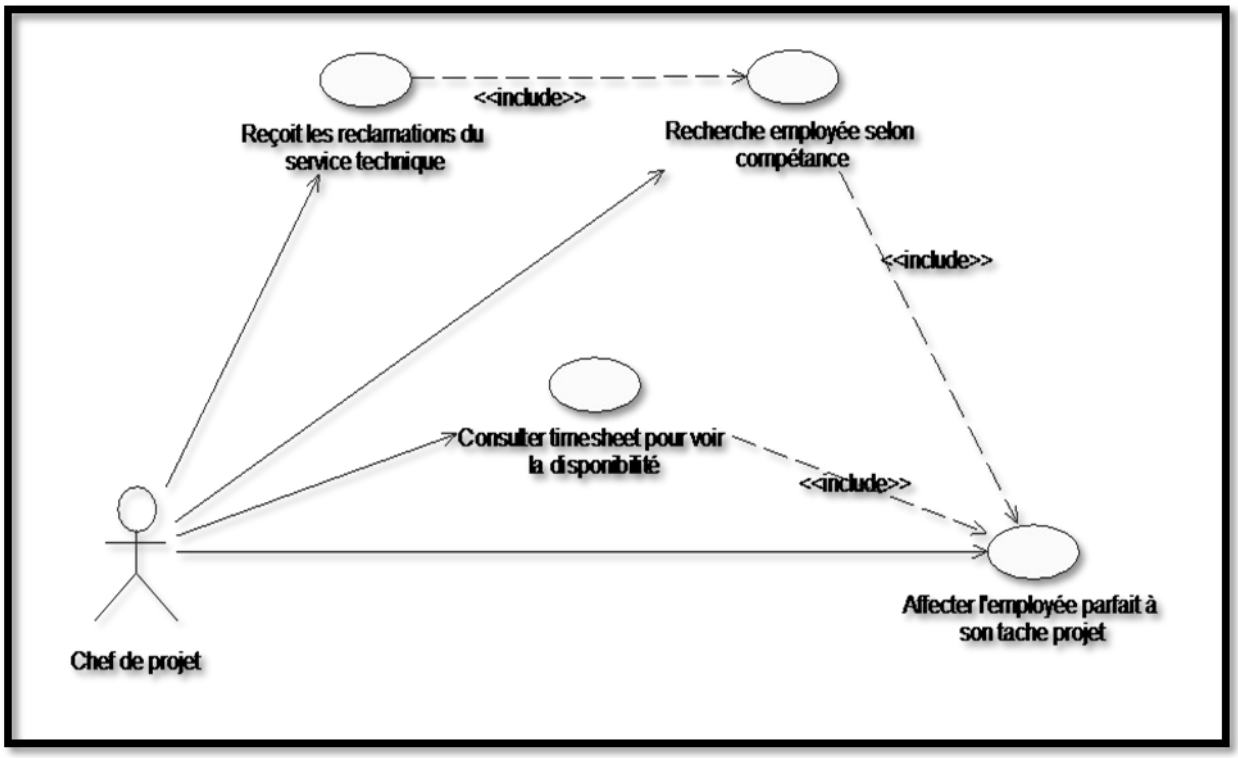


Figure 10. Diagramme Cas Utilisateurs Gestion d'affectation des employées

- *Diagramme Cas utilisateur des taches employées :*

Spécification : Ce cas d'utilisation représente l'interaction de chaque employée avec notre système. Ce diagramme pressente les tâches employées.

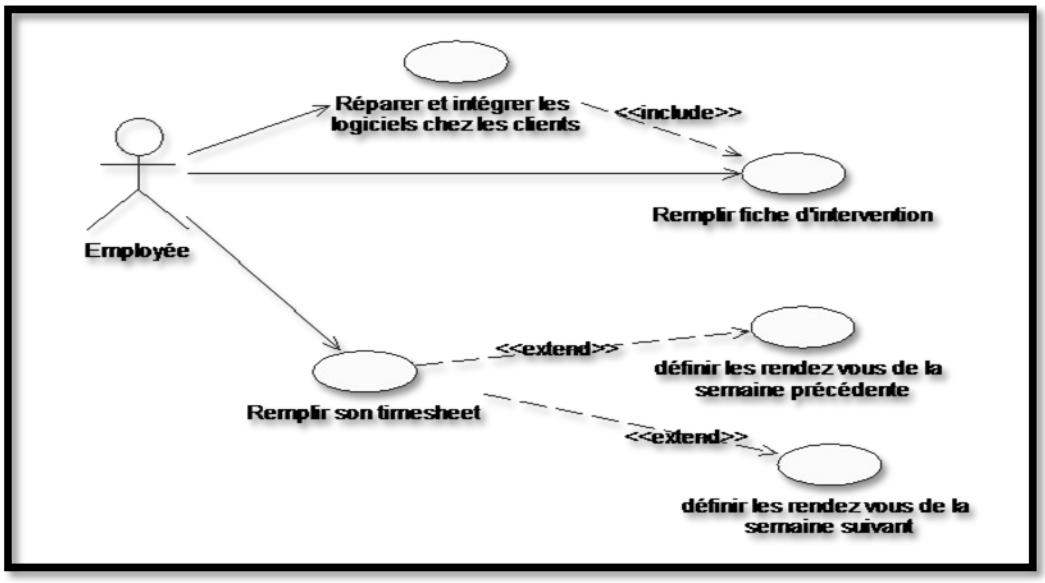


Figure 11. Diagramme Cas Utilisateurs des taches employées

- *Diagramme gestion des réclamations :*

~~*Spécification*~~ : Ce cas utilisateurs représente l'interaction du Client et Service Technique avec notre système. Ce diagramme explique le rôle d'une timesheet dans la gestion des réclamations.

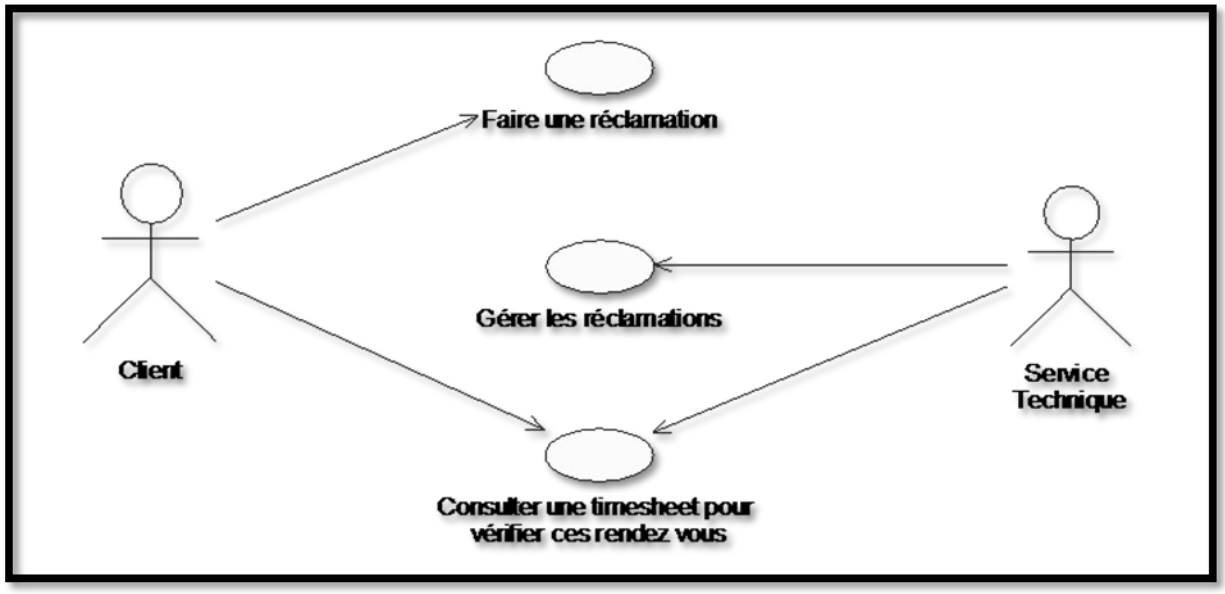


Figure 12. Cas Utilisateur de gestion des réclamations

II.4.3.4 Raffinement des cas d'utilisation

- *Raffinement du premier cas d'utilisation :*

Intitulé du cas d'utilisation	Gestion d'affectation des employées
Acteur	Chef de projet
Pré-condition	Le chef de projet reçoit une réclamation du service technique.
Post-condition	Selon les compétences de chaque membre de son groupe el leur disponibilité le chef de projet affecte l'employée.
Description du cas d'utilisation	Le chef de projet reçoit une réclamation alors il faut la gérer rapidement et donne la solution pour la satisfaction des clients.

- *Raffinement du deuxième cas d'utilisation :*

Intitulé du cas d'utilisation:	Gestion des taches employées
Acteur	Employée
Pré-condition	L'employée reçoit une mission à faire du chef de projet.
Post-condition	L'employée faire son travail demandé et remplir son fiche de chaque intervention faite, aussi remplir son emploie du temps de chaque semaine.
Description du cas d'utilisation	Les taches d'employée et leur interaction avec les acteurs du système.

- *Raffinement du troisième cas d'utilisation :*

Intitulé du cas d'utilisation	Gestion des réclamations
Acteur	Client et service technique
Pré-condition	Le client a eu un problème.
Post-condition	Le client exprime son besoin au service technique
Description du cas d'utilisation	Comment la gestion des réclamations et le rôle d'un timesheet pour minimiser le nombre des réclamations.

II.5. Conclusion

Nous avons étudié l'environnement de notre projet et analysé l'existant et les besoins de la société tout au long de ce deuxième chapitre d'analyse des besoins et spécification.

Dans le chapitre suivant, nous allons présenter la conception de notre application, en présentant quelques diagrammes illustrant les différentes fonctionnalités de notre application.

Chapitre III Conception

III.1. Introduction

La phase de conception est considérée parmi les phases les plus importantes dans le cycle de vie d'un logiciel. Il s'agit en premier lieu de présenter une modélisation par étape de la conception de notre application et décrire finalement l'architecture de la base de données.

III.2. Conception

Les diagrammes UML de modélisation dynamique nous montrent l'évolution du système et les interactions entre objets. Nous allons présenter comme exemple de cette modélisation un ensemble de diagramme d'activité et diagramme de séquence ensuite on va décrir e l'architecture de la base de données.

Le diagramme d'activités n'est autre que la transcription dans UML de la représentation du processus telle qu'elle a été élaborée lors du travail qui a préparé la modélisation : il montre l'enchaînement des activités qui concourent au processus.

Le diagramme de séquence représente les interactions entre objets au cours du temps par des messages (synchrone et asynchrone).

III.2.1. Module « Recrutement »

III.2.1.1 Diagramme de classe

Le diagramme de classes est considéré comme le plus important de la modélisation orientée objet, il est le seul obligatoire lors d'une telle modélisation.

Le diagramme de classes montre la structure interne du système. Il permet de fournir une représentation abstraite des objets du système qui vont interagir ensemble pour réaliser les cas d'utilisation. Il s'agit d'une vue statique car on ne tient pas compte du facteur temporel dans le comportement du système. Les principaux éléments de cette vue statique sont les classes et leurs relations : association, généralisation et plusieurs types de dépendances, telles que la réalisation et l'utilisation.

Une classe de conception est composée par :

- **Attribut** : chaque attribut d'une classe est le même pour chaque instance de cette classe.
- **Méthodes** : elle définit le comportement d'une classe elle-même, et non le comportement de ses instances qui peut être différent.

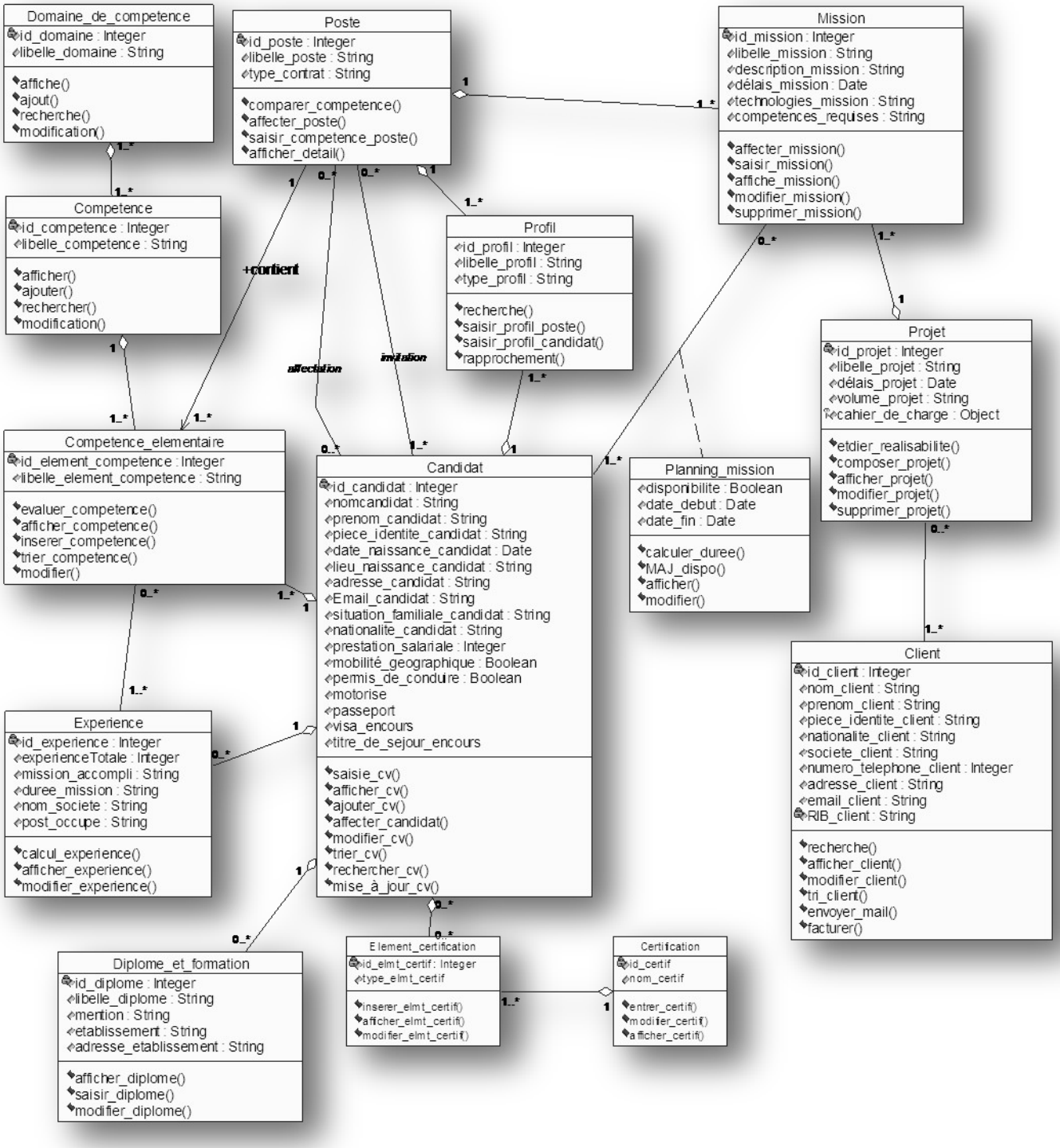


Figure 13. Diagramme de classes

- **Candidat** : classe entité qui représente les différents candidats qui postulent leurs cv, les attributs de cette classe représentent les différents champs dans le cv.

- **Diplôme et formation** : classe entité qui représente les différents diplômes et/ou formations qu'a suivi chaque candidat. Cette classe regroupe donc les différents diplômes des différents candidats.

- **Etablissement** : classe entité, contient tous les établissements reliés aux différents diplômes.

- **Expérience** : classe entité qui représente les différentes expériences de chaque candidat.

- **Domaine compétence** : classe entité qui représente le domaine de compétence pour chaque compétence.

- **Compétence** : classe entité qui représente les différentes compétences que peut avoir un candidat.

- **Compétence élémentaire** : classe entité qui représente les différents éléments de compétence du candidat.

- **Poste** : classe entité qui représente les différents postes auxquels pourrait postuler un candidat. Il contient différents compétences élémentaires exigés par le poste, il a un profil poste, et est relié à plusieurs mission.

- **Profil** : classe entité qui représente les profils poste et candidat.

- **Mission** : classe entité qui représente les différentes missions qui peuvent être affectées à différents candidats.

- **Projet** : classe entité qui regroupe les différentes missions par projet ; cette classe représente les projets sur lesquels travaille l'entreprise.

- **Planning Mission** : classe entité qui représente l'emploi du temps de chaque candidat affecté à une mission, pour pouvoir gérer sa disponibilité.

- **Client** : classe entité qui représente les différents clients de l'entreprise.

- **Certification** : classe entité qui représente les différentes certifications que pourrait avoir chaque candidat et qui regroupe différents éléments de certification.

- **Elément certification** : classe entité qui représente les différents éléments de certification que pourrait avoir chaque candidat.

III.2.1.2 Diagramme d'activité globale

Pour des raisons de lisibilité, nous allons schématiser les figures «Diagramme d'activité de la gestion de la demande de recrutement» et «Diagramme d'activité globale du système» en largeur de la page. La figure suivante représente le diagramme d'activité global de la gestion de candidatures aux recrutements.

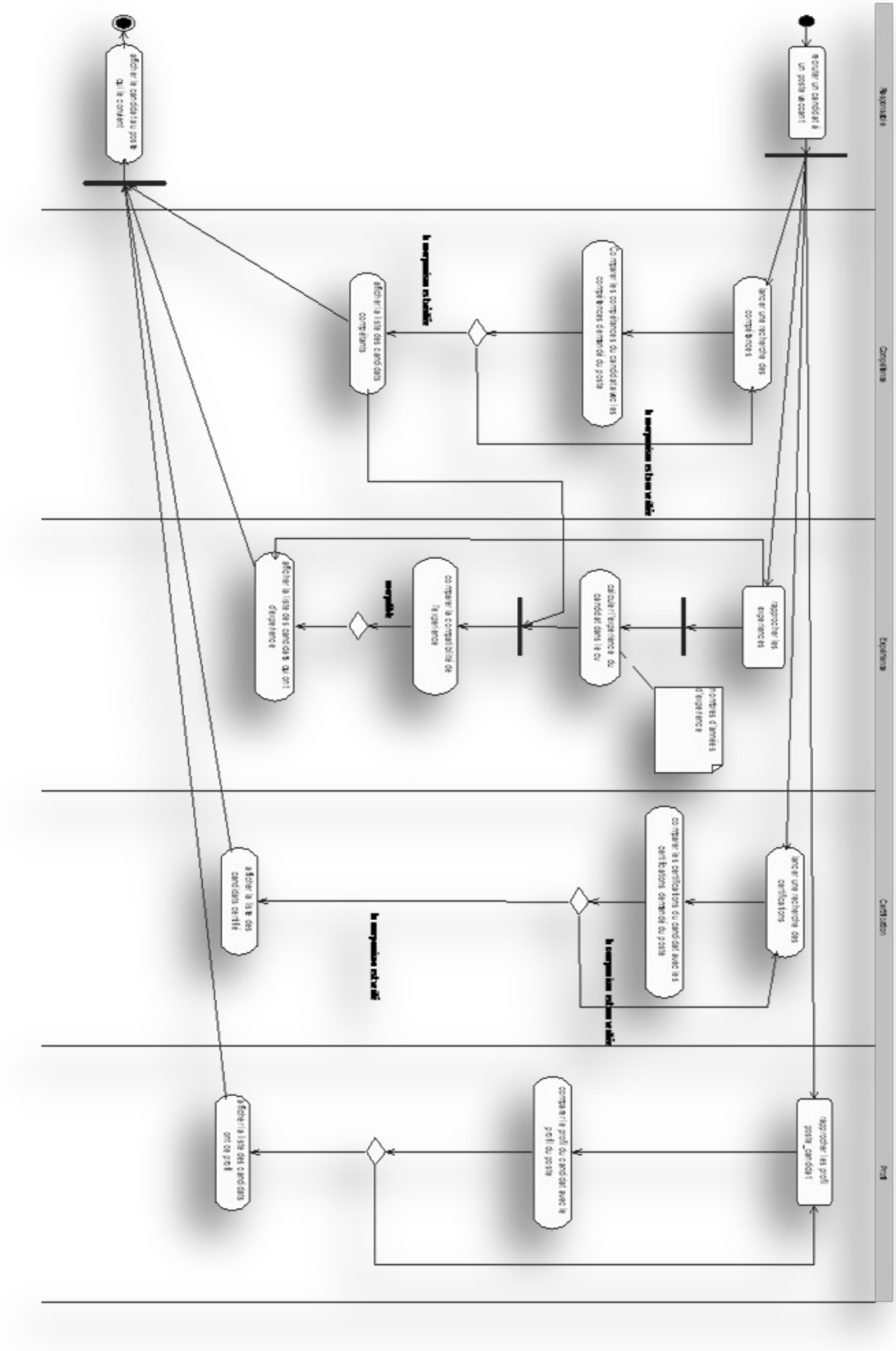


Figure 14. Diagramme d'activité relatif au cas d'utilisation globale de système

III.2.1.3 Diagrammes d'activité et de séquence détaillés

- *Analyse du cas d'utilisation «Gestion de la demande de recrutement» :*

Diagramme d'activité

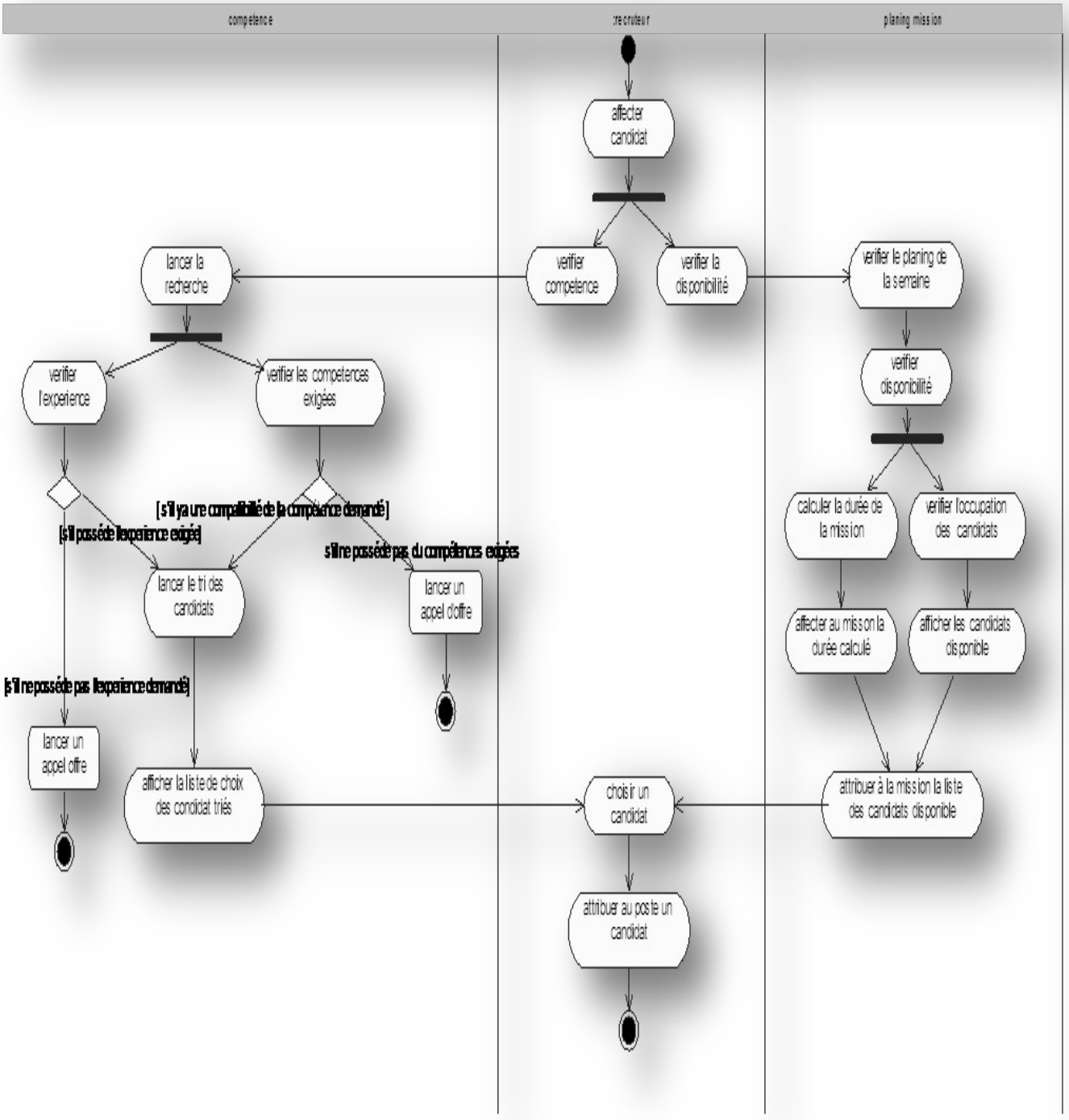


Figure 15. Diagramme d'activité relatif au cas d'utilisation «Gestion de la demande de recr

Diagramme de séquence

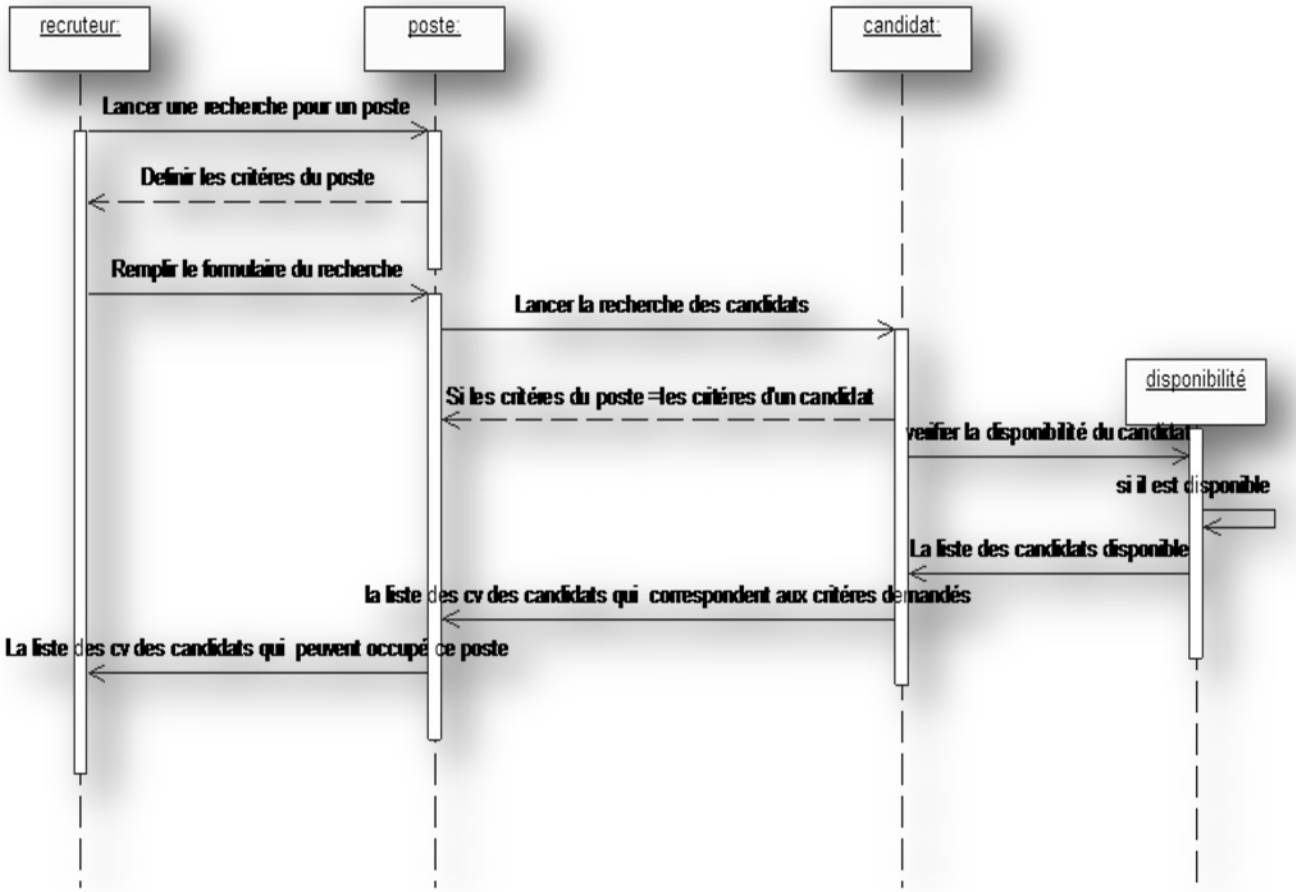


Figure 16. Diagramme de séquences relatif au cas d'utilisation «Gestion de la demande de r

- *Analyse du cas d'utilisation «Gestion de la demande de recrutement» :*

Diagramme d'activité

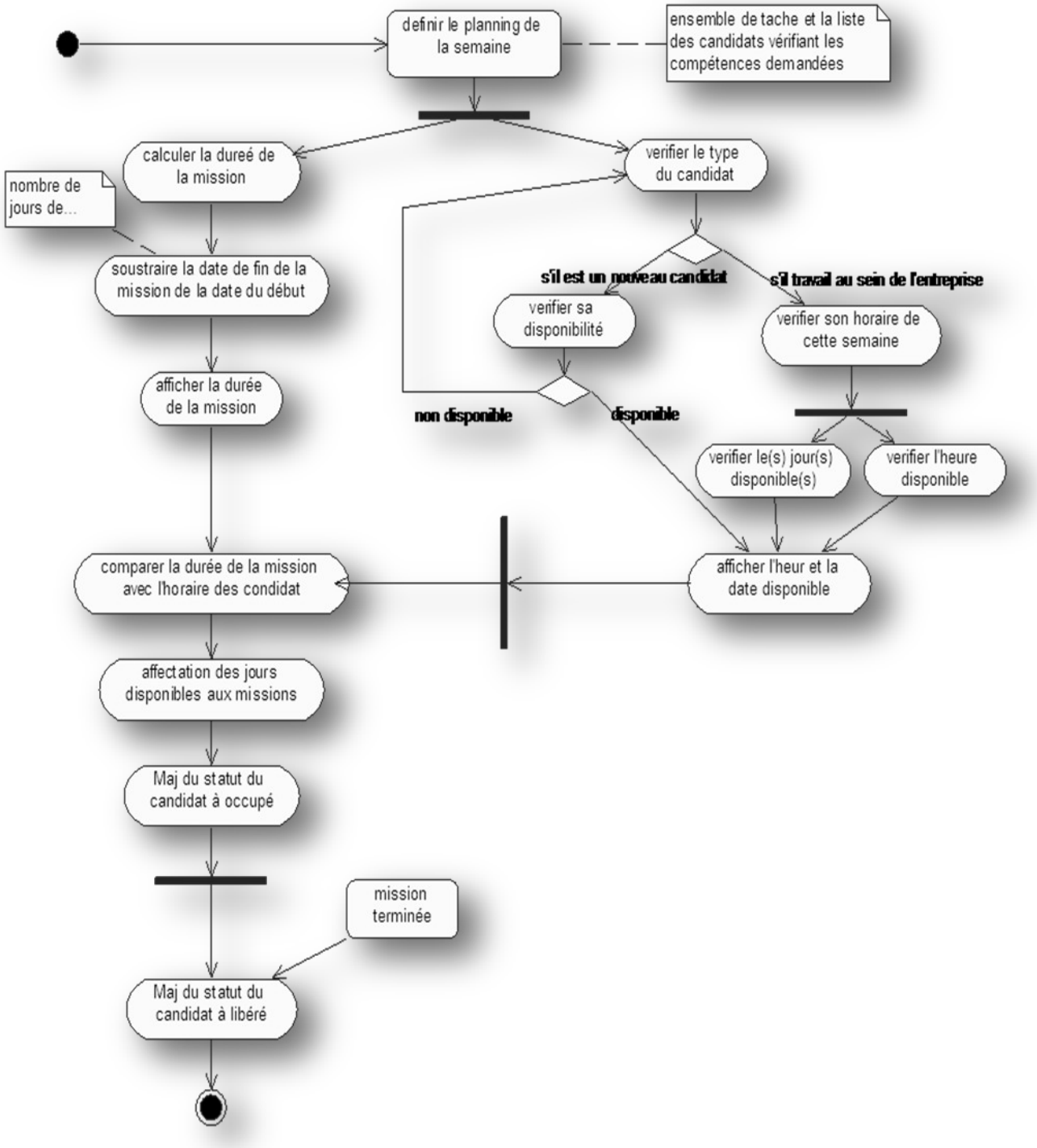


Figure 17. Diagramme d'activité relatif au cas d'utilisation « Gestion de la planification pér

Diagramme de séquence

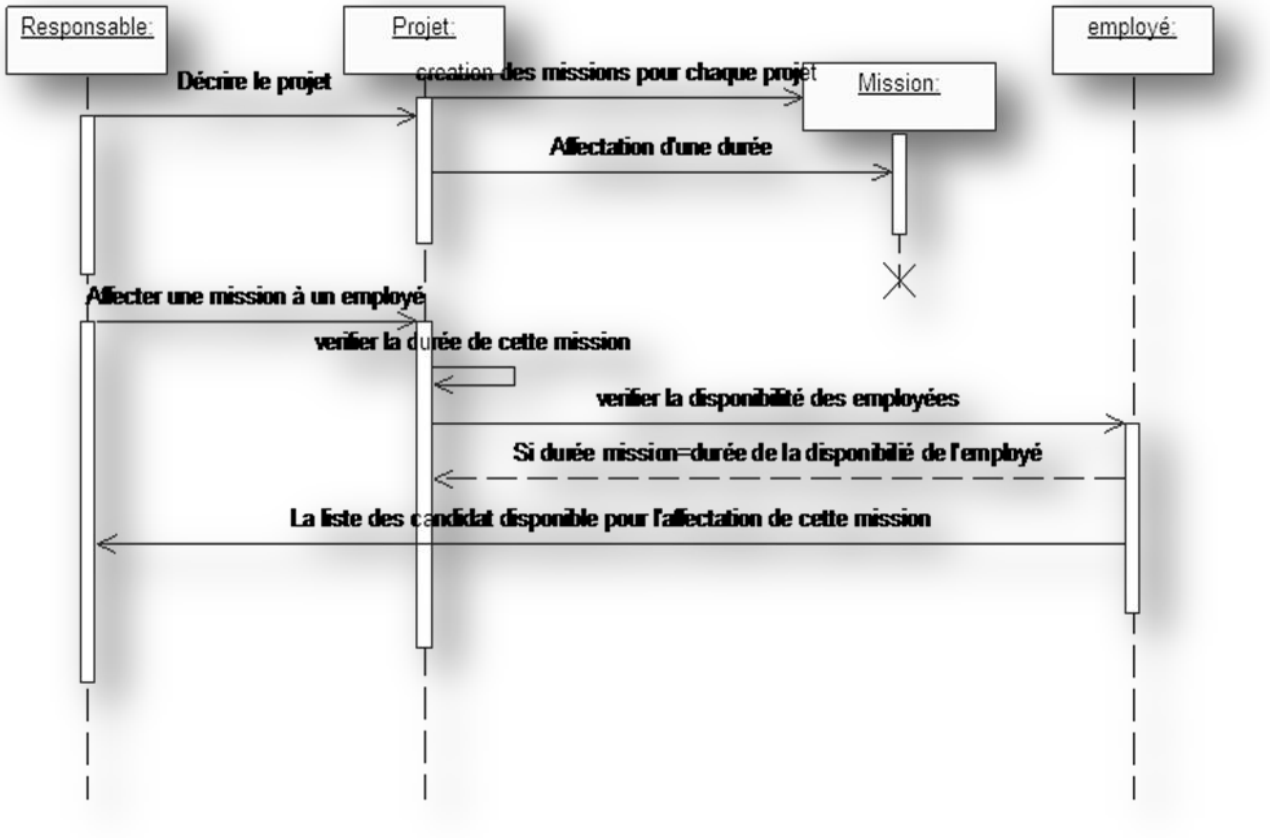


Figure 18. Diagramme de séquences relatif au cas d'utilisation «Gestion de la planification p

III.2.2. Module «Espace Employée »

III.2.2.1 Diagramme de classe

Dans cette partie on va présenter une liste des classes pour montrer la structure interne du module Espace employée.

- **Timesheet** : classe entité, c'est-à-dire les feuilles de temps, en premier lieu ils représentent une durée de travail réalisée par l'employée à une date donnée.
- **Employée** : classe entité hérite de la classe user, le timsheet traduit une relation avec la classe Employée pour gérer ses tâches et son emplois de temps de chaque semaine. Cette classe représente une liste des employés.
- **Client** : classe entité qui représente les différents clients de l'entreprise.
- **Fiche intervention** : classe entité représente le travail fait par l'employé chez le client.

- **Chef Projet** : classe entité hérite de la classe user et qui a un statut différent, elle présente le chef de projet qui affecte l'employé.
- **Projet** : classe entité qui représente une liste des projets des clients.
- **Tâche projet** : classe entité représente les tâches d'un projet.
- **Service Technique** : classe entité qui représente un utilisateur qui a le droit de recevoir du client une liste de réclamation.
- **Réclamation** : classe entité qui représente une liste des réclamations des clients.
- **Semaine** : classe entité qui représente les jours de la semaine

Voici le diagramme de classe qui représente les classes et les associations de notre système.

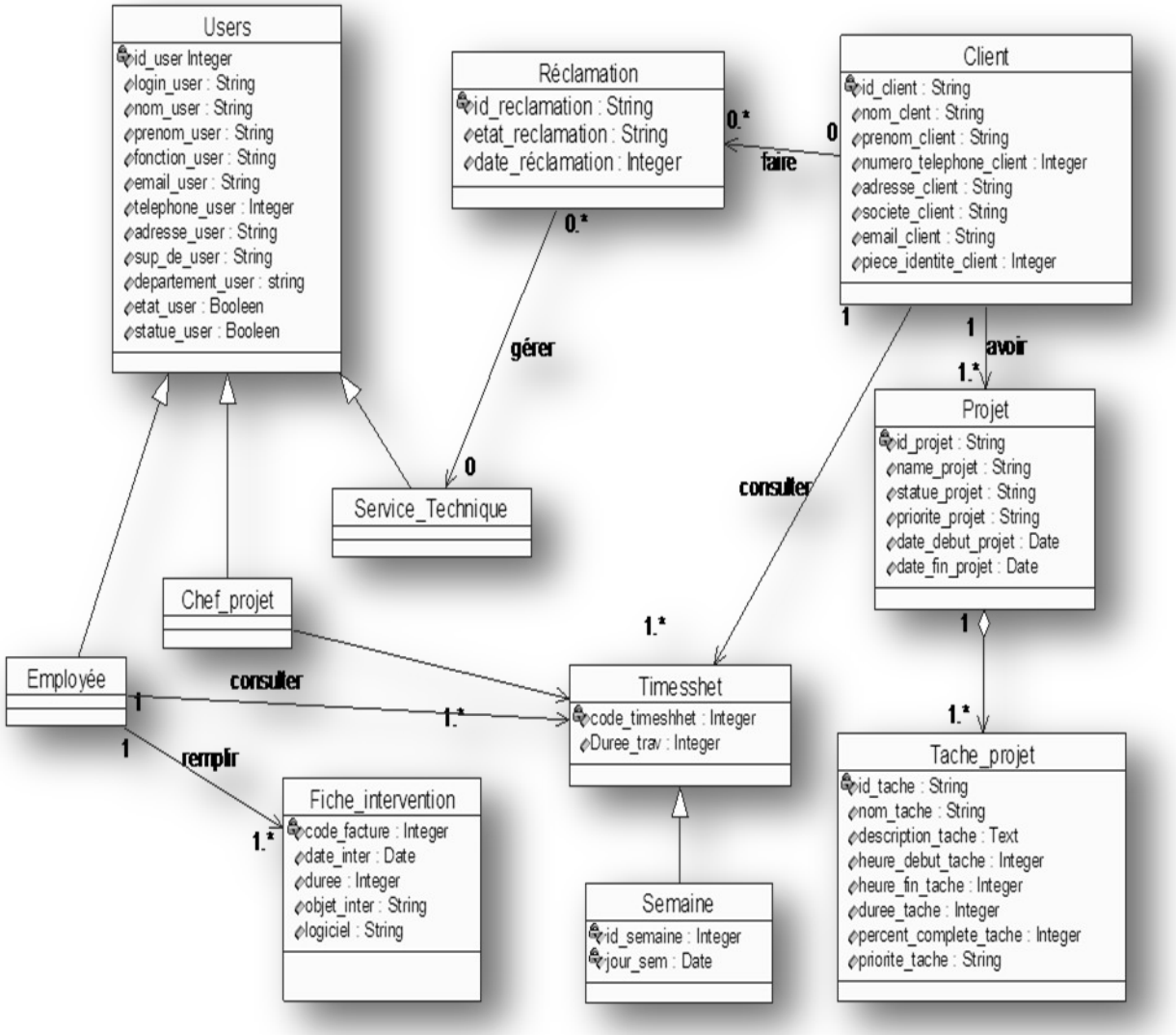


Figure 19. Diagramme de classe « gestion employée »

III.2.2.2 Diagramme d'activité globale

Pour des raisons de lisibilité, nous allons schématiser seulement le diagramme d'activité globale du module gestion d'employées.

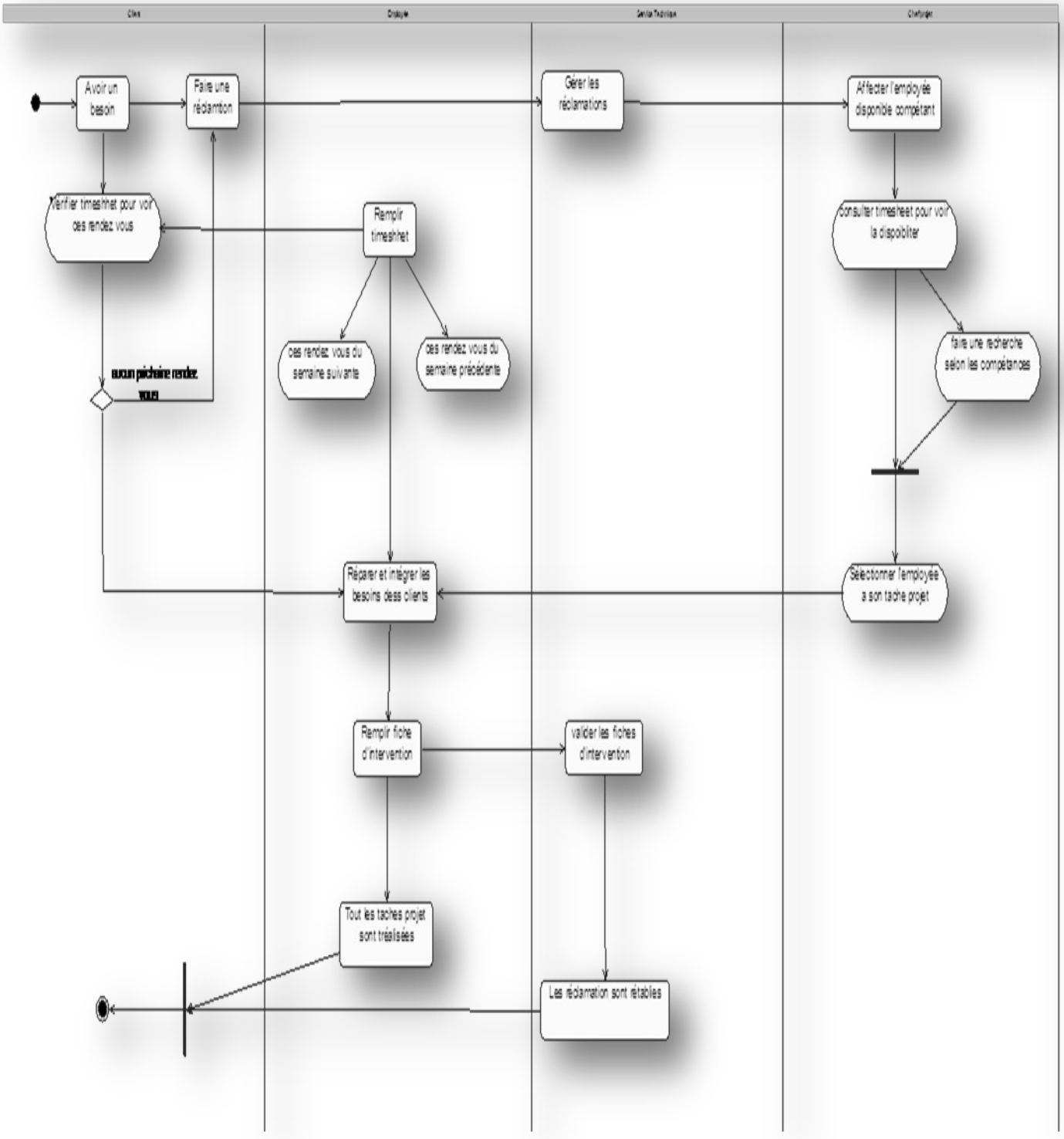


Figure 20. Diagramme d'activité global

III.3. Structure de la base de donnée

Nous allons présenter la structure des tables pour le deuxième module. Les clés primaires sont soulignées et les clés étrangères sont précédées par un dièse.

- **Client** : (id_client, nom_client, prenom_client, numero_telephone_client, adresse_client, societe_client, email_client, piece_identite_client).
- **Users** : (id_user, login_user, prenom_user, fonction_user, email_user, telephone_user, adresse_user, sup_de_user, departement_user, etat_user, statue_user).
- **Fiche_Intervention** : (code_facture, date_inter, duree, objet_inter, logiciel, #id_user, #id_client, #id_tache_projet).
- **Réclamation** : (id_reclamation, etat_reclamation date_réclamation, #id_client).
- **Projet** : (id_projet, name_projet, statue_projet, priorite_projet, date_debut_projet, date_fin_projet).
- **Tâche_projet** : (id_tache_projet, nom_tache, description_tache, heure_debut_tache, heure_fin_tache, duree_tache, percent_complete_tache, priorite_tache, #id_projet).
- **Timesheet**: (id_timesheet, duree_trav, #id_user #id_projet #id_tache_projet).
- **Semaine** : (id_semaine, jour_sem, #id_timeshhet).

III.3.1. Dictionnaire des données de la base des données

Nom des champs	Libellé	Type de données
Id_client **(clé primaire)**	Identificateur du client	Numéro Auto
Nom_client	Nom du client	Texte
Prenom_client	Prénom du client	Texte
Numero_telephone_client	Numéro de téléphone de l'agent	Numérique
Adresse_client	Adresse du client	Texte

Nom des champs	Libellé	Type de données
Societe_client,	Société du client	Texte
Email_client,	Email du client	Texte
Piece_identite_client	Carte d'identité du client	Numérique
Id_user **(clé primaire)**	Identificateur de l'utilisateur	Numéro Auto
Login_user	Login de l'utilisateur	Texte
Prenom_user	Prénom de l'utilisateur	Texte
Fonction_user	Fonction de l'utilisateur	Texte
Email_user	Email de l'utilisateur	Texte
Telephone_user	Téléphone de l'utilisateur	Numérique
Adresse_user	Adresse de l'utilisateur	Texte
Sup_de_user	Superviseur de l'utilisateur	Texte
Departement_user	Département de l'utilisateur	Texte
Etat_user	Etat de l'utilisateur	Numérique
Statue_user	Statue de l'utilisateur	Texte
Code_facture,	Code de la facture	Numérique
Date_inter	Date de l'intervention	Date
Duree_inter	Durée de l'intervention	Numérique
Objet_inter	Objet de l'intervention	Numérique
Logiciel_rép	Logiciel réparé	Texte
Id_reclamation **(clé primaire)**	Identificateur de la réclamation	Numéro Auto
Etat_reclamation	Etat de la réclamation	Texte
Date_réclamation	Date de la réclamation	Date
Id_projet **(clé primaire)**	Identificateur du projet	Numéro Auto
Name_projet	Nom du projet	Texte
Statue_projet	Statue du projet	Texte
Priorite_projet	Priorité du projet	Texte
Date_debut_projet	Date de début du projet	Date

Nom des champs	Libellé	Type de données
Date_fin_proje	Date de fin du projet	Date
~~Id_tache_projet~~ **(clé primaire)**	Identificateur de la tache projet	Numéro Auto
Nom_tache	Nom de la tâche	Texte
Description_tache,	Description de la tache	Texte
Heure_debut_tache	Heure de début de la tache	Numérique
Heure_fin_tache	Heure de la fin de tache	Numérique
Duree_tache	Durée de la tache	Numérique
Percent_complete_ tache	Pourcentage complète de la tache	Numérique
Priorite_tache	Etat d'un PDA	Texte
Id_timesheet **(clé primaire)**	Identificateur du type d'une question	Numéro Auto
Duree_trav	Durée du travaille	Texte
Id_semaine	Identificateur de la semaine	Numéro Auto
Jour_sem **(clé primaire)**	Journée de la semaine	Date

Tableau 1. Dictionnaire des données

III.4. Conclusion

Au cours de ce chapitre, nous avons conçu notre base de données ainsi que les différents composants de notre système.

Maintenant, notre application est prête à être codée. Mais tout le prochain chapitre sera la mise en évidence des technologies utilisées dans notre application.

Chapitre IV : Etat de l'art

IV.1. Introduction

Le choix des technologies utilisées dans une application agit sans doute sur la qualité du travail réalisé. Le succès ou l'échec d'un travail quelconque dépend, en partie, du choix des technologies employées. Nous allons exposer dans ce chapitre la technologie envisageable et relative au développement de notre application, c'est le « SugarCRM ». Aussi nous allons présenter des petites comparaisons entre le SugarCRM et les autres éditeurs de CRM.

IV.1.1. Présentation du SugarCRM

- **Qu'est ce que SugarCRM ?**

SugarCRM est une solution CRM Open Source. Il comprend des modules de gestion des Forces de Vente (SFA: Sales Force Automation), d'automatisation du marketing (EMA), de reporting, et de gestion du service client. Il apporte des solutions de Gestion de la Relation Cliente adaptées aux besoins des entreprises. SugarCRM permet une démarche précise et une approche personnalisée pour chaque client. A partir de l'analyse des besoins de l'entreprise, enjeux et contraintes, nous pouvons le personnaliser et ajouter de nouveaux modules. Voici la conception des applications dans SugarCRM :

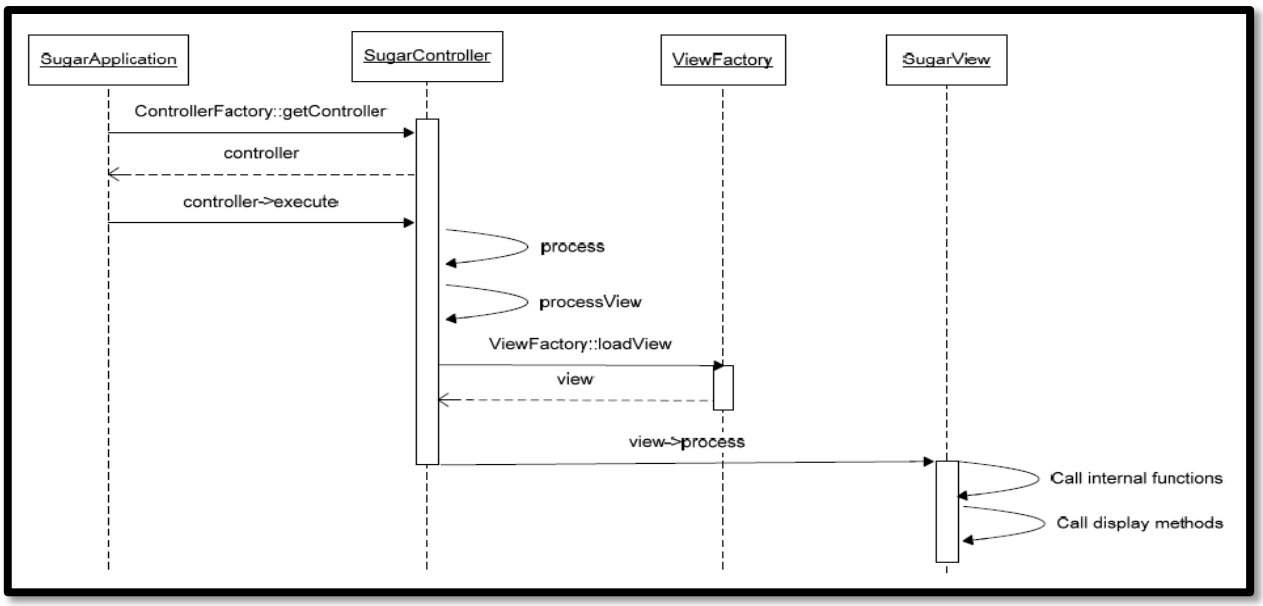


Figure 19. Les éléments impliqués dans le MVC de SugarCRM

Le schéma de la figure est un diagramme de séquence qui met en lumière quelques-uns des principaux éléments impliqués dans le Framework SugarCRM.

IV.1.1.1 Méthode pour intégrer un nouveau module

La personnalisation des applications peut se faire de deux manières : Nous pouvons créer de nouveaux modules en utilisant le « Module Builder » ou personnaliser des modules existants en utilisant le « Studio » parmi les outils du développeur (developper tools) directement à partir de l'interface SugarCRM. Le « Module Builder » est un générateur, qui nous permet de construire de nouveaux modules pour étendre les fonctionnalités de SugarCRM. La fonctionnalité du Module Builder permet aux programmeurs de créer des modules et de créer des relations entre ces modules sans écrire du code. Parfois, la personnalisation d'un module peut nécessiter une modification dans le code source.

Nous pouvons aussi construire et personnaliser de nouveaux modules entièrement à partir du code source de l'application tout en respectant l'architecture MVC qu'adapte SugarCRM, et les points d'entrées sans avoir à utiliser le générateur du Builder.

SugarCRM est basé sur une structure modulaire et sur la sécurité des points d'entrée de l'application (par exemple index.php : qui est l'aiguilleur de l'application permet d'accéder aux différentes interfaces). Avant chaque accès l'utilisateur doit obligatoirement passer pour s'authentifier. Cette demande d'authentification est affichée lors de la première demande de la page de SugarCRM ou lorsqu'un utilisateur entre directement dans la barre d'adresse de l'URL voulu et qui devra lui ouvrir la fenêtre d'un module donné; dans ce cas il sera redirigé vers cette page pour s'authentifier en premier lieu. Une fois identifié, l'utilisateur pourra naviguer librement parmi les modules et les fonctionnalités du logiciel.

Dans ce qui suit, nous allons illustrer une démonstration de l'utilisation du logiciel SugarCRM :

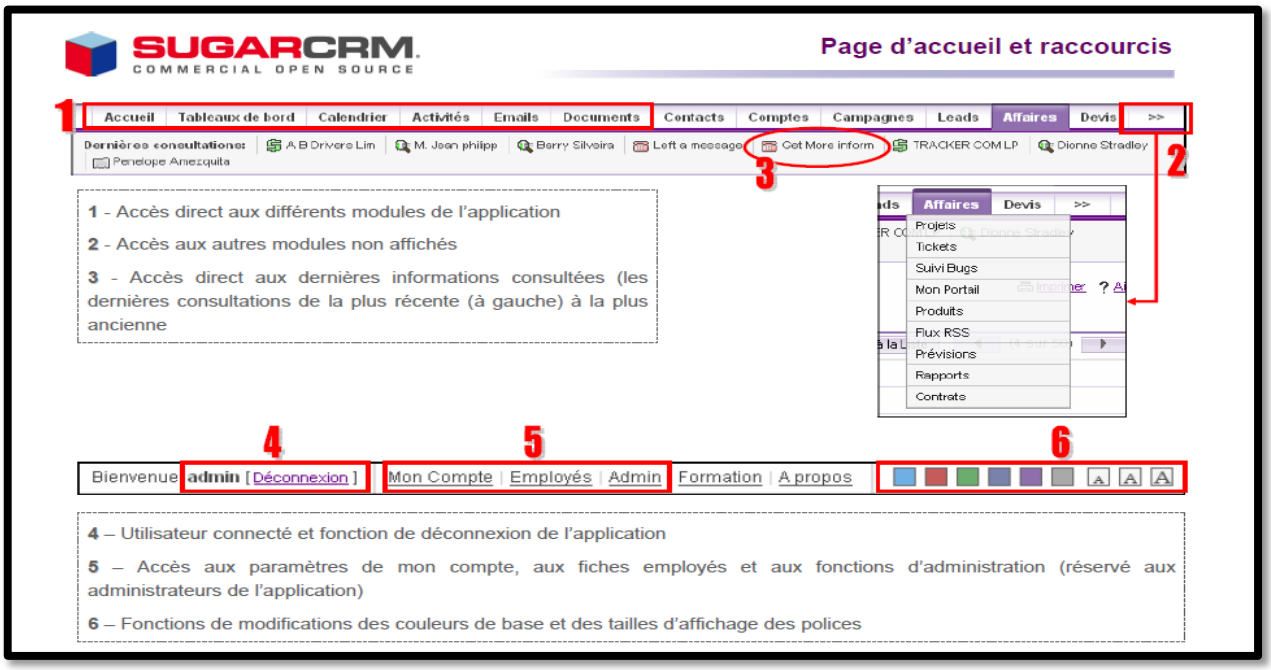


Figure 20. La navigation entre les modules

Si l'utilisateur identifié est l'administrateur « admin » alors il aura en plus, un accès à la page « Admin » où plusieurs fonctionnalités sont proposées, parmi lesquelles nous avons le « developper tools » :

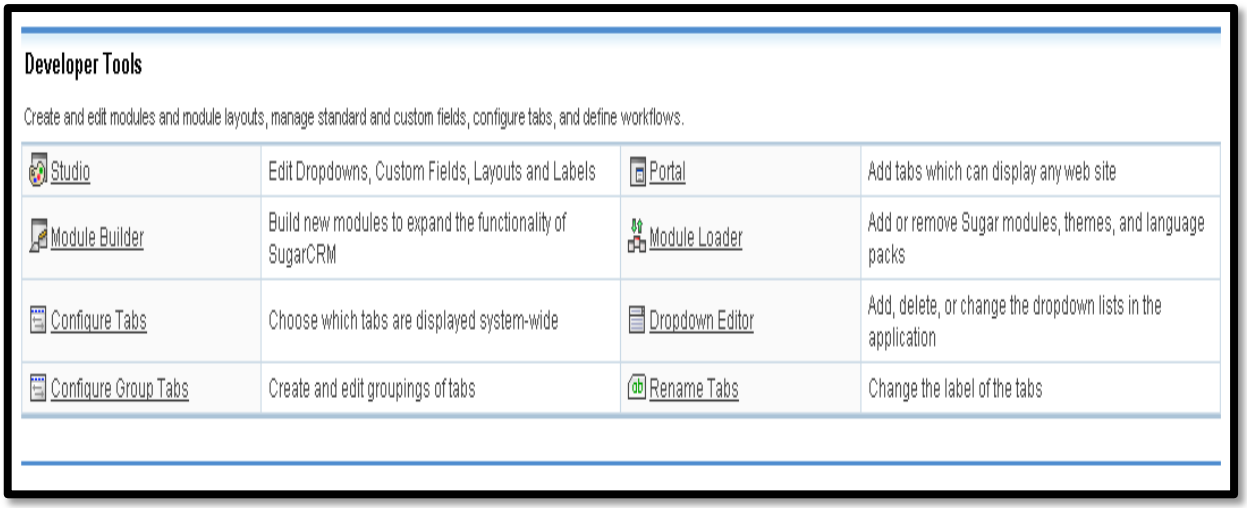


Figure 21. Module Builder fait partie de la console «Developer Tools» de l'administration d

A partir de là, nous pouvons accéder au « Module Builder » pour créer de nouveaux modules, ou accéder à d'autres fonctionnalités du « Developper Tools ».

Les figures suivantes nous montrent des aperçus de création grâce au générateur Module Builder :

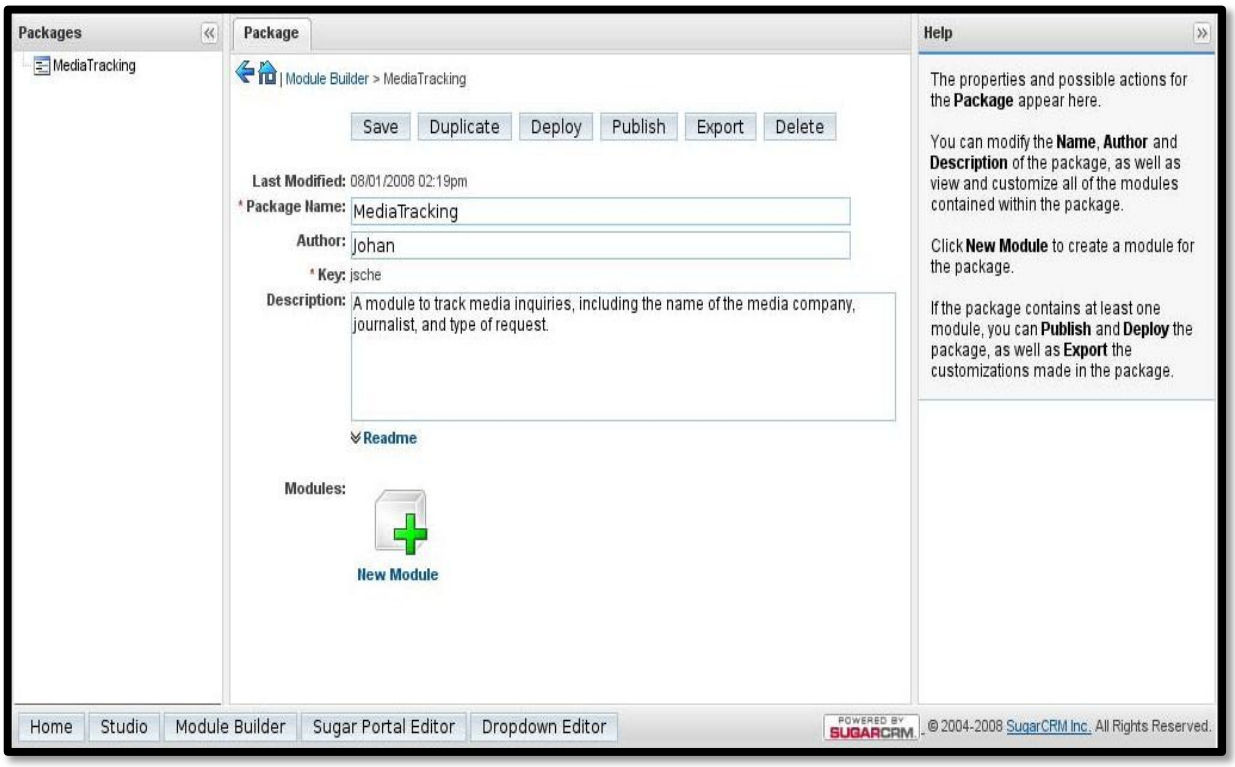


Figure 22. L'utilisateur commence par créer et nommer le nouveau package des modules p

Une fois le nouveau package créé et enregistré, un écran est présenté à l'utilisateur pour créer un module personnalisé. Lors de la sélection de l'icône "Nouveau module", un écran apparait, indiquant six modèles d'objets différents. Pour chaque module, nous pouvons créer, en plus des champs par défaut, des champs personnalisés, nous pouvons aussi les publier grâce à l'Edit View. Nous pouvons gérer les « relationship » du module avec les autres modules. Plusieurs autres manipulations divers et variées sont possibles avec le logiciel SugarCRM, (Vous trouverez de plus amples détails dans la partie Annexe A).

IV.1.1.2 La structure des répertoires du code source de SugarCRM

Cette figure présente une liste des sous-répertoires sous le dossier Sugarcrm :

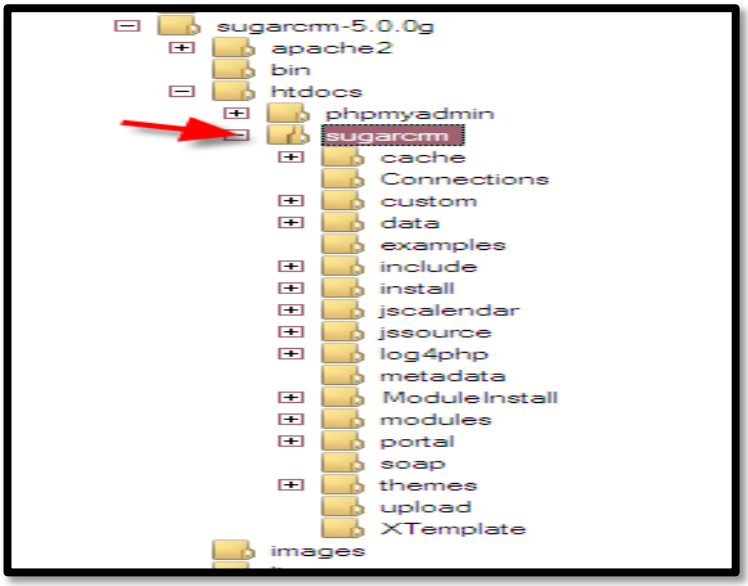


Figure 23. Les répertoires différents à l'intérieur du SugarCRM

Le code de SugarCRM réside dans des répertoires différents à l'intérieur de la Sugar Installation. La structure des répertoires au sein de l'application du Sugar se compose des répertoires suivants:

- **custom** : Magasins de mise à niveau de sécurité des personnalisations telles que les définitions de champ, présentation de l'interface utilisateur et le logique métier.
- **data** : système de fichiers de clés se trouvent dans ce dossier, et plus particulièrement la classe de base qui est SugarBean qui contrôle l'application logique par défaut pour tous les objets des entreprises de sugar.
- **include** : comporte beaucoup de fonctions utilitaires du Sugar, ainsi que d'autres bibliothèques que Sugar utilise dans le cadre de ses opérations. Plus particulièrement, dans ce répertoire, on trouve le fichier « utils.php » qui contient toutes les fonctions utilitaires les plus largement utilisées.
- **Metadata** : Ce fichier contient les relations métadonnées de tous les liens des relations entre les données et les objets métiers (Business Object).
- **modules** : Ce dossier contient tous les modules du système. Custom modules installés par le biais de Module Loader existe ici aussi.

- *Concepts clés :*

Tels sont les principaux dossiers, de classes et de l'application des concepts qui composent la plate-forme de Sugar.

- **Controller** : Il dirige toutes les demandes de pages. Il peut être relu plusieurs fois, dans chaque module pour modifier le comportement par défaut. Il s'appuie sur des paramètres de point d'entrée, pour servir la page appropriée.

- **Views** : Un jeu de l'interface utilisateur, des actions gérées par le contrôleur, les vues par défaut dans Sugar comportent les DetailView, EditView et ListView.

- **Display Strings** : SugarCRM est entièrement internationalisé et localisable. Chaque pack de langue a son propre ensemble de chaînes d'affichage qui est la base de la langue de localisation.

- **Dropdown Lists**: ils sont représentées comme des **'nom'**=>**'valeur'** paires de tableaux situés dans l'application strings comme indiqué ci-dessus.

La valeur du **'nom'** est stockée dans la base de données et la **'valeur'** est affiché à l'utilisateur dans l'interface utilisateur. On peut créer et modifier des listes déroulantes (dropdown lists) et leurs valeurs via l'interface utilisateur dans l'outil d'admin Studio.

- *Fichiers :*

- **sugarBean.php :** Ce fichier se trouve sous le «<sugar root> / data» contient le dossier SugarBean classe de base utilisée par tous les objets de l'entité ou d'un module. Tout module qui lit, écrit ou affiche des données va étendre cette classe. Le SugarBean joue le levage lourd pour les données d'interactions, des relations de manutention, etc.

- **modules.php :** Le fichier modules.php est un fichier critique à Sugar. Il contient plusieurs variables qui définissent quels modules seront actifs et utilisables dans la demande.

- *Points d'entrée :*

Le point d'entrée principal de l'interface utilisateur pour Sugar se fait par **index.php** situé à la racine du dossier Sugar (root sugar folder). Il y a trois paramètres principaux pour la plupart des appels à l'intérieur du Sugar, qui sont :

- **module** : Le module est accessible dans le cadre de l'appel.

- **action** : L'action à prendre par l'application dans le module.

- **record** (dossier): Le dossier est accessible à l'id.

IV.2. Architecture de l'application « MVC »

L'architecture de notre application est l'architecture MVC « **M**odel-**V**iew-**C**ontroller ». Il s'agit d' une méthode de conception qui crée une séparation entre le logique métier ou de traitement « business-logic » et la logique de présentation « display logic ».

- **Model** : C'est l'objet de données construit par l'entreprise ou par l'application logique nécessaire pour la présenter à l'interface utilisateur. Pour SugarCRM, il est représenté par la SugarBean et toutes les sous-classes de la SugarBean.

- **View** : Il s'agit de la couche d'affichage, qui est responsable de rendre les données du modèle de l'utilisateur final.

- **Controller** : C'est la couche qui gère les utilisateurs des événements tels que «Enregistrer Candidat» et détermine la logique des mesures à prendre pour construire le modèle, et à la charge de rendre les données au utilisateur final.

 Utiliser l'architecture MVC dans les applications Web/PHP :

Le modèle MVC (Modèle-Vue-Contrôleur) cherche à séparer nettement les couches présentation, traitement et accès aux données. Une application web respectant ce modèle sera architecturée de la façon suivante :

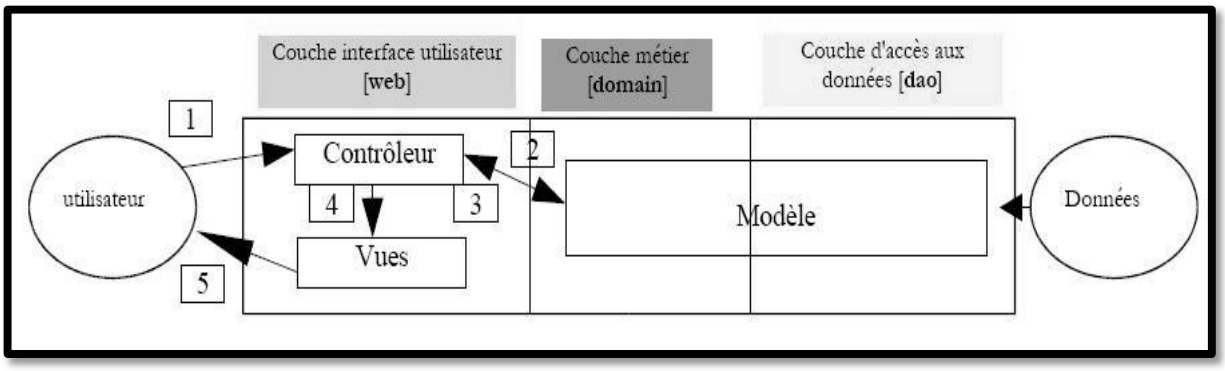


Figure 24. L'architecture MVC d'une application web

Le traitement d'une demande d'un client se déroule selon les étapes suivantes :

1. Le client fait une demande au contrôleur. Ce contrôleur voit passer toutes les demandes des clients. C'est la porte d'entrée de l'application. C'est le C de MVC.

2. Le contrôleur traite cette demande. Pour ce faire, il peut avoir besoin de l'aide de la couche métier, ce qu'on appelle le modèle M dans la structure MVC.

3. Le contrôleur reçoit une réponse de la couche métier. La demande du client a été traitée. Celle-ci peut appeler plusieurs réponses possibles. L'exemple classique est une page d'erreurs si la demande n'a pu être traitée correctement ; une page de confirmation sinon.

4. Le contrôleur choisit la réponse (= vue) à envoyer au client. Celle-ci est le plus souvent une page contenant des éléments dynamiques. Le contrôleur fournit ceux-ci à la vue.

5. La vue est envoyée au client. C'est le V de MVC.

Le modèle MVC adapté au Web est une réponse à une division des responsabilités :

- la partie Vue est prise en charge par un script (PHP ou autre langage) générant du HTML (sans autre logique de traitement)
- la partie contrôleur est représentée par un script PHP qui déclenche des traitements liés aux services (Use Case) auquel il est attaché (Un contrôleur par Use Case)
- la partie Modèle est représentée par des scripts PHP gérant les accès au SGBDR. Cela peut être par exemple des classes Métier qui implémentent des fonctions vers un SGBDR.

IV.3. Comparaison entre les technologies CRM

Dans le monde de CRM on retrouve plusieurs solutions ; il y en a celles qui sont payantes et d'autres qui sont libres et notre choix de Sugar CRM repose sur plusieurs raisons. Au fait il n'existe pas de meilleure solution CRM mais plutôt il existe la meilleure solution qui répond à nos besoins. Tout d'abord on définit les solutions CRM les plus répondues sur le marché. On commence avec les solutions payantes :

- **Microsoft Dynamic On Demand** : elle ne fonctionne que sur Internet Explorer et en plus elle est un peu cher par rapport aux autres solutions c'est à partir de 42 € par mois, en plus, plus on ajoute des fonctionnalités plus le prix augmente.
- **Centric CRM** : c'est une solution Open Source qui est utilisé par plus 500 compagnies mais la version gratuite vient seulement avec 5 licences d'utilisateurs ce qui ne l'a rend pas pratique pour une entreprise qui contient plus de 100 employés.
- **Tilliolive** : elle aussi c'est une solution Open Source mais elle n'est pas ergonomique c'est à dire les utilisateurs simple qui ne s'y connaissent pas vraiment en informatique auront des difficultés à l'utiliser et vu que nos utilisateurs finaux sont des personnes ordinaires donc cette solution ne sera pas optimale pour nous.

- **Sugar CRM** : c'est la solution la plus utilisée et la plus répondue. Sugar a été téléchargé plus de 3 millions de fois et plus de 12000 sociétés l'utilisent, en plus il est écrit en php et est compatible avec la base de données Mysql.

Sugar profite d'une plate-forme largement déployée et facilement accessible. En plus nous avons choisi Sugar pour la facilité d'intégrations de modules qui répondent à nos besoins

IV.4. Conclusion

Ce chapitre nous a permis d'étudier la technologie SugarCRM pour l'élaboration de notre système. En outre la comparaison entre ces CRM nous a permis de dégager celle la plus adéquate pour la réalisation de notre projet.

Chapitre V : Réalisation

V.1. Introduction

La phase de réalisation est une étape très importante dans le cycle de vie de notre application. Cette phase permet de concrétiser notre projet par le développement des interfaces et par des réalisations concrètes des fonctionnalités du système. Pou réaliser notre application nous avons en recourt à plusieurs outils de développement.

Dans ce chapitre, nous présentons en premier lieu l'environnement de travail. En second lieu, nous présentant le choix techniques que nous avons adopté pour le développement de notre application.

V.2. Environnement de travail

V.2.1. Environnement matériel

Tout au long de la période de stage nous avons disposé de deux ordinateurs, un ordinateur pour chacun avec un serveur CRM.

- *Le premier ordinateur :*

Marque : DELL VOSTRO 1015
Système : Microsoft Windows XP Professionnel version 2002 service pack3
Ordinateur : INTEL ® Core™2 Duo CPU T 6570 @ 2.10 GHz
1,19 GHz 2,96 Go de RAM

- *Le deuxième ordinateur :*

Marque : Mac OS X version 10.6.3
Système : Macintosh HD
Ordinateur : INTEL ® Core™2 Duo 2.26 GHz
2 Go 1067 MHz DDR3

V.2.2. Environnement logiciel

- *Dreamweaver :*

Dreamweaver fut l'un des premiers éditeurs HTML. Ces innovations le propulsèrent rapidement comme l'un des principaux éditeurs de site web, aussi bien utilisable par le néophyte que par le professionnel.

Dreamweaver offre deux modes de conception par son menu affichage. L'utilisateur peut choisir entre un mode création permettant d'effectuer la mise en page directement à l'aide d'outils simples, comparables à un logiciel de traitement de texte (insertion de tableau, d'image, etc.) et un autre mode de code. Il est également possible d'afficher et d'éditer directement le code (HTML ou autre) qui compose la page.

- *Rational Rose :*

Pour la conception et la modélisation des diagrammes UML nous avons utilisé le logiciel Rational Rose. Comme tous les produit Rational Rose, il fournit un langage de modélisation commun à votre équipe et permet d'accélérer la création de logiciels de qualité.

IBM Rational Rose Enterprise est l'un des produits les plus complets de la gamme Rational Rose. Tous les produits Rational Rose prennent en charge le langage UML (Unified Modeling Language), mais ils se différencient de par les technologies d'implémentation qu'ils prennent en charge.

V.3. Choix techniques

V.3.1. Choix du langage

- *PHP :*

Notre application a été développée en PHP. Un extrait du code est présenté à l'annexe B. Il concerne le développement du moteur de recherche des candidats.

PHP (HyPertext Preprocessor) est un langage de programmation qui s'intègre dans nos pages HTML. Il permet entre autres de rendre automatiques des tâches répétitives, notamment grâce à la communication avec une base de données.

- *HTML :*

HTML (**H**yper **T**ext **M**arkup **L**anguage / langage hypertexte) est le langage dans lequel sont écrites les pages du web. Un site web est constitué d'un ou plusieurs documents HTML, appelées aussi pages.

Pour se déplacer d'une page à l'autre dans nos modules on passe par l'intermédiaire d'hyperliens. Pour ajouter des objets graphiques on utilise le HTML d'autre part pour tester des pages web html en local, il suffit d'ouvrir le fichier dans un navigateur.

Le HTML n'est pas un langage de programmation comme le C+. Les langages dynamiques comme PHP et Javascript vont d' ailleurs générer des pages HTML statiques.

- *Java script :*

Javascript est un langage de programmation de scripts principalement utilisé pour les pages web interactives comme les pages HTML. Javascript est exécuté sur l'ordinateur de l'internaute par le navigateur lui-même. C'est une extension du langage HTML qui est incluse dans le code.

Ce langage est un langage de programmation qui permet d'apporter des améliorations au langage HTML en permettant d'exécuter des commandes. Ce code est directement écrit dans la page HTML, c'est un langage peu évolué qui ne permet aucune confidentialité au niveau des codes.

Dans notre application on a codé de plusieurs fonctions javascript par exemple : pour l'interaction des pages en envoyant des variables dans l'adresse URL pour filtrer la résultat de la requête on utilisant la méthode POST ou GET.

- *Ajax :*

AJAX (Asynchronous Javascript And Xml) il désigne un nouveau type de conception de pages Web permettant l'actualisation de certaines données d'une page sans procéder au rechargement total de cette page. AJAX n'est pas une technologie en elle-même, mais un terme qui évoque l'utilisation conjointe d'un ensemble de technologies libres couramment utilisées sur le Web

Les applications AJAX peuvent être utilisées au sein des navigateurs Web qui supportent les technologies décrites précédemment. Parmi eux, on trouve Mozilla Firefox, Internet Explorer et on a utilisé ce concept pour recharger un seul bloc dans la page timesheet.

V.3.2. Choix de la technologie de sécurité

La sécurité de l'application est assurée par les sessions de SugarCRM et les points d'entrées. En effet, tous les points d'entrée dans l'application de sugar sont prédéfinis, afin que des mesures de sécurité et d'authentification soient appliquées de manière cohérente à travers l'ensemble de l'application.

Le passage obligatoire de nos pages développés, par le point d'entrée index.php, permet de rediriger tout utilisateur non authentifié, à la page d'authentification de sugarCRM :

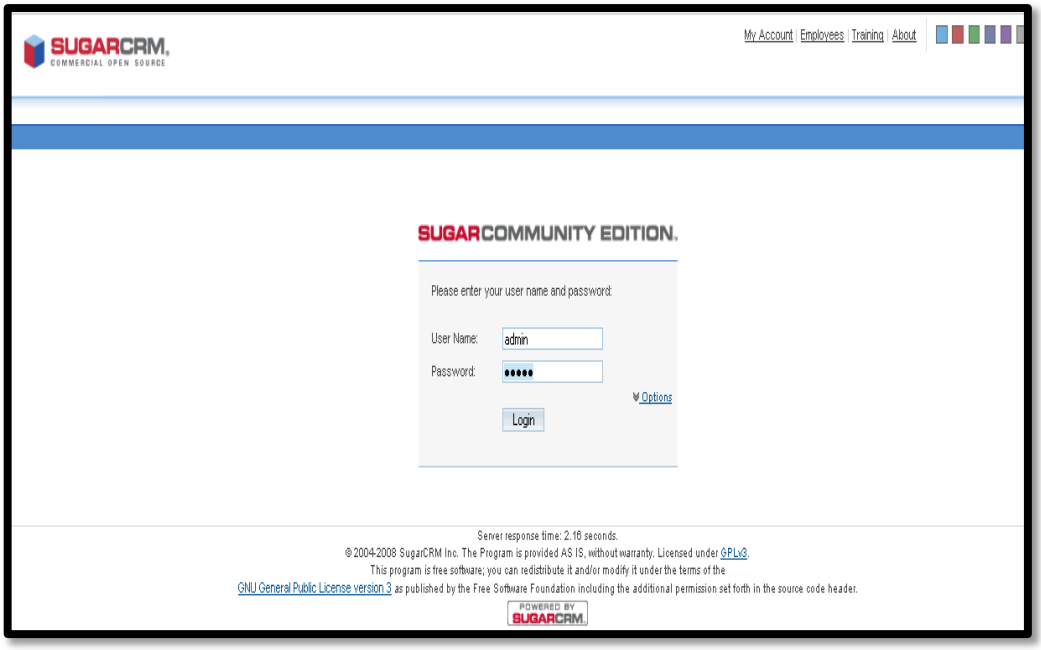


Figure 25. Interface d'authentification

V.4. Gestion du projet

Dans cette partie nous représentons l'organigramme chronologique de la réalisation de notre projet. Cet organigramme comporte les principales tâches effectuées tout au long du projet. Nous avons commencé durant trois semaines par analyser l'existant et spécifier les besoins, puis nous avons passé à la partie conception qui a duré quatre semaines. Vers la fin de cette dernière nous avons commencé la plus grande partie (plus que dix semaines) c'est la partie de programmation et la réalisation. Pour la partie du rapport nous l'avons crée au fur et a mesure de l'avancement du travail (les deux dernières mois).

Le tableau qui illustre la répartition du travail tout au long de ce projet est présenté dans la figure ci-dessous:

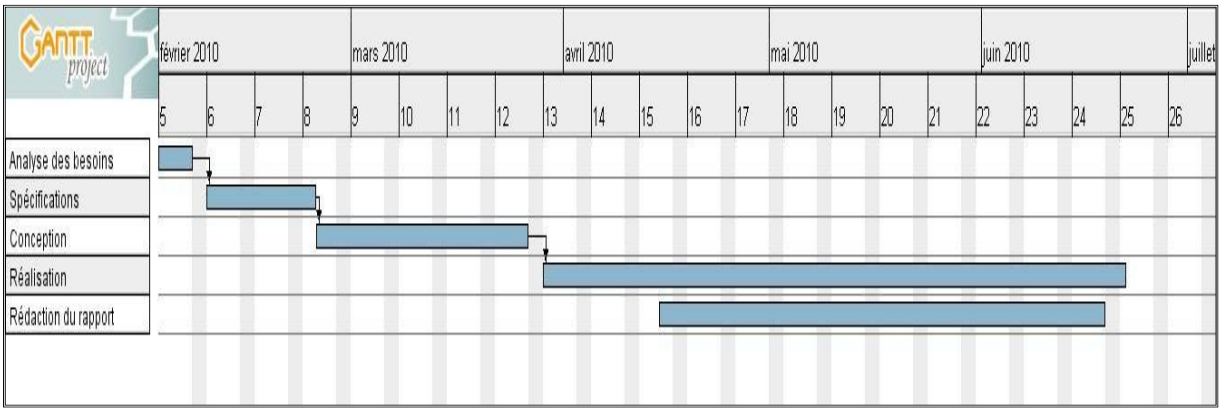


Figure 26. Chronologique de la réalisation de notre projet

V.5. Phase d'implémentation

V.5.1. Contraintes

Dans le cas du développement d'une simple application web, le développeur est libre de développer sans aucune limitation ni contrainte une application correspondante exactement à ses différents besoins. Mais dans notre cas, nous allons développer une application qui doit être intégrée à l'ensemble des modules et solutions qu'offrent SugarCRM.

La plateforme de SugarCRM n'est guère simple ou facile à comprendre. La compréhension de l'architecture de la plateforme, la maîtrise de l'ensemble de ses dossiers, le cheminement des informations et la répartition du code dans SugarCRM sont relativement compliqués. Cette plateforme est conçue en MVC cette architecture est très bien adapté aux grands projets tel que le CRM de Sugar.

Notre plus grande contrainte est de se limiter dans le développement de l'application selon l'architecture de la plateforme, ses bibliothèques, ses classes et ses fonctions. On peut bien entendu, développer d'autres classes et fonctions dans cette plateforme, on va donc ajouter du code et le répartir pas à pas sur les différentes couches du modèle.

La figure suivante représente une solution proposée pour l'intégration d'un nouveau module.

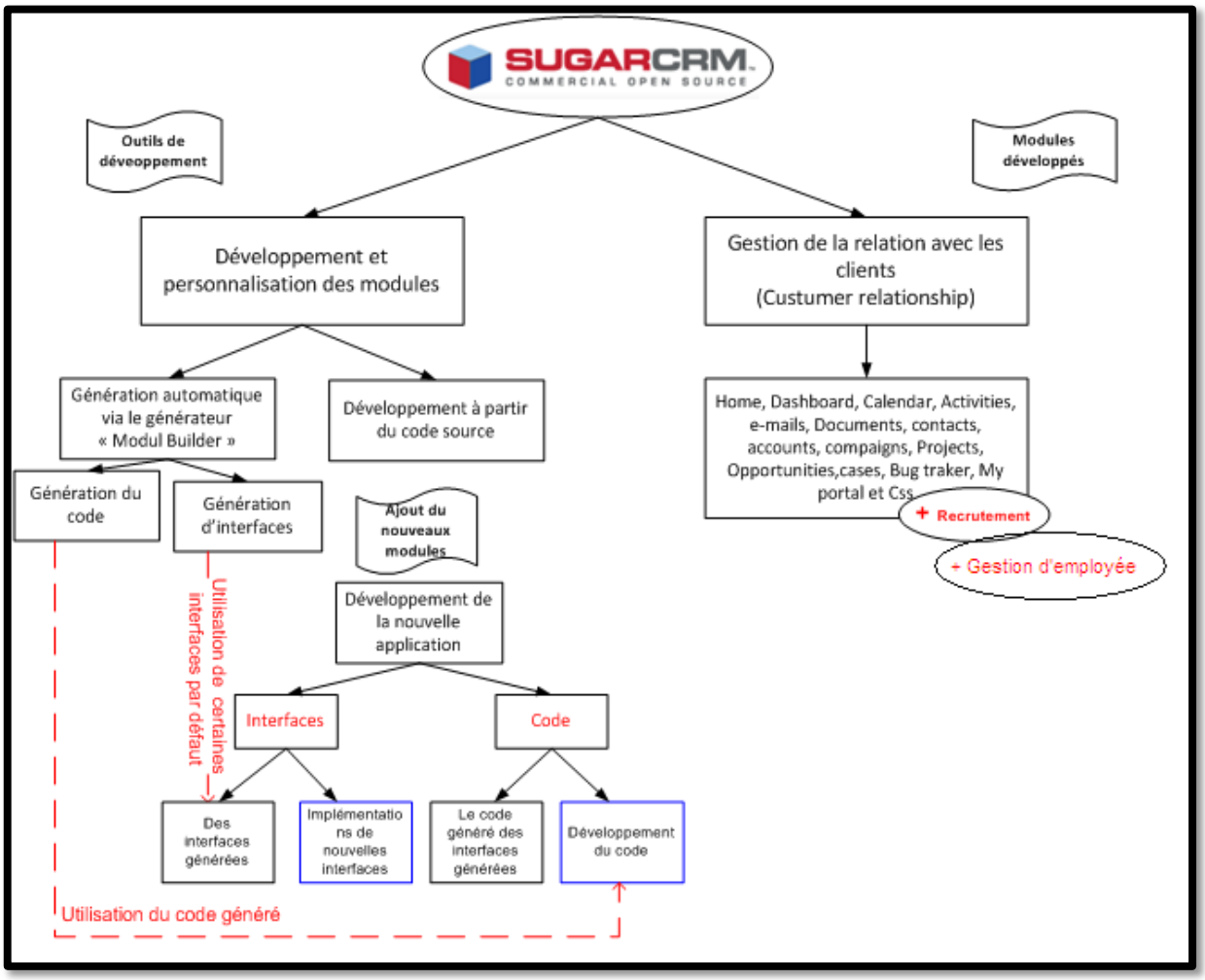


Figure 27. Intégré un module dans SugarCRM

La partie droite représente les fonctionnalités offertes par SugarCRM, à un simple utilisateur, et la partie gauche représente les fonctionnalités offertes au développeur. Nous entant que développeurs nous devront s'intéresser à la partie gauche, du dessin, pour bien sûre, aboutir à la fin de notre développement à la réalisation de notre application dans la partie droite du schéma, c.à.d. l'ajout de notre application (recrutement et gestion d'employée) dans l'interface utilisateur, comme une fonctionnalité et des modules de plus, parmi, les fonctionnalités et modules qu'offre SugarCRM.

V.5.2. La programmation avec Sugar et MVC

Avec l'avènement de SugarCRM 5.0, une nouvelle architecture MVC est née. Cette architecture a été désignée pour éliminer les tâches pénibles dans la construction des modules dans Sugar. Ainsi, au lieu d'avoir à tracer manuellement des modèles et de mettre en place leurs interactions et leurs relations, on pourra facilement tirer profit du Framework à l'aide des modèles standardisés et les fichiers de définitions pour la construction de divers vues. Le système est aussi très extensible permettant de nouvelles vues et Templates personnalisés.

V.5.2.1 MVC (Model-View-Controller):

MVC est un modèle de design qui crée une séparation nette entre le logique métier et la logique d'affichage.

- **Le Model** : C'est l'objet de données construite par l'entreprise logique de l'application / besoin de présenter à l'interface utilisateur. Pour SugarCRM, il est représenté par le SugarBean et toutes les classes de la SugarBean

- **La Vue** : Il s'agit de la couche d'affichage qui est chargé de rendre les données du modèle à l'utilisateur final.

- **Le Contrôleur** : C'est la couche qui gère les événements d'utilisateurs tels que «Enregistrer» et détermine les actions à prendre la logique métier pour construire le modèle, et c'est pour rendre les données à l'utilisateur final.

V.5.2.2 SugarCRM 5.0 Mis en œuvre avec MVC :

- *Le Model :*

Les objets Sugar étendent la notion de sous classement un peu plus loin et vous permet de sous classer la vardefs. Cela comprend l'héritage de champs, les relations, les index et les fichiers de langue, mais à la différence sous classement on n'est pas limité à un héritage unique. S'il y avait un objet de Sugar pour les champs qui sont utilisés dans chaque module comme id, supprimé, date_modified, etc.

On peut avoir notre module hérité de deux objets : de base de Sugar et de la personne objet du Sugar. Maintenant, disons que le type de base possède un champ "nom" de longueur 10. Et Candidat a un champ "nom" avec une longueur de 20. Si on hérite de la base d'abord, puis de l'entreprise sur le terrain sera de longueur 20. Maintenant disons qu'on a défini un

champ "nom" dans notre module qui est de longueur 60. Notre module sera toujours la priorité sur toutes les valeurs fournies par des objets du Sugar.

Actuellement, il existe 4 types de Sugarobject modèles :

- **Contacts Personne** (Contacts, Prospects, Leads)
- **Issue** (Bugs, Cases)
- **Société** (Accounts)
- **De base** (General Fields)

Les interfaces SugarObject et les modèles SugarObject sont très semblables les uns aux autres, mais la principale distinction est que les modèles ont une classe de base, on peut les sous classer alors que ce n'est pas le cas des interfaces. Si on regarde dans la structure du fichier, on remarquera que les modèles comprennent de nombreux fichiers supplémentaires, y compris un répertoire de métadonnées complètes. Ceci est principalement utilisé par ModuleBuilder pour obtenir les mises par défaut à partir de celui-ci.

Structure du fichier : **include/SugarObjects/interfaces**

include/SugarObjects/templates

- *Le Contrôleur :*

La version 5.0 introduit un concept de contrôleur en cascade. Il ya un contrôleur principal appelé SugarController qui gère toutes les actions de base d'un module d'édition et des vues de détail pour l'archivage d'un enregistrement. Chaque module peut remplacer ce SugarController en ajoutant un fichier controller.php dans son répertoire. Ce fichier devrait étendre le SugarController et la convention de nommage pour les classes. A l'intérieur du contrôleur on définit les méthodes d'action.

Il y a plusieurs mécanismes de contrôle qu'un développeur peut utiliser pour remplacer le traitement de commande. Par exemple, si un développeur a voulu créer une sauvegarde de nouvelles actions, il y a 3 endroits où ils pourraient éventuellement remplacer.

- **Action_save** : ce serait le plus large cahier des charges et donnerait à l'utilisateur un contrôle total sur le processus d'enregistrement.
- **Pre_save** : un utilisateur peut réécrire les paramètres de la forme.

- **Post_save** : c'est là où la vue est le programme d'installation en cours. À ce point, le développeur peut définir une URL de redirection, faire un peu de post traitement enregistrer ou fixer un point de vue différent.

On peut également ajouter un contrôleur personnalisé qui devrait étendre le contrôleur modules si le contrôleur de ce type existe déjà. Par exemple, si on souhaite étendre le contrôleur à un module qui vient avec sortie SugarCRM 5.0, on doit vérifier si ce module a déjà un contrôleur de module spécifique fourni avec le produit. Si oui, on peut étendre à partir de cette classe de contrôleur sinon on étend de la classe SugarController. Dans les deux cas, on doit placer le fichier contrôleur personnalisé classe custom / modules / <MyModule> / Controller.php au lieu du répertoire du module. Cela rend notre personnalisation forte. Par exemple, si une requête arrive pour detailview c'est ainsi que le contrôleur se chargera de cette demande.

- Démarrer dans index.php nous chargeons de l'instance SugarApplication
- SugarApplication instancie SugarControllerFactory
- ChargesSugarControllerFactory le contrôleur approprié
- Vérifiez les custom / modules / <MyModule> / controller.php
- S'il n'est pas trouvé, vérifier les modules / <MyModule> / controller.php
- S'il n'est pas trouvé, la charge SugarController.php
- Appel sur les mesures appropriées
- Vérifiez les modules / <MyModule> / <MyAction>. Php
- S'il n'est pas trouvé, vérifier la méthode action_ <MyAction> dans le contrôleur.
- S'il n'est pas trouvé, recherchez un action_file_mapping
- S'il n'est pas trouvé, rapport d'erreur "L'action n'est pas défini „

On ce qui concerne le module « Timesheet» on a eu recourt à plusieurs instance de contrôleur de base de Sugar comme pour la sécurité on a fait appel à sécurité.php c'est grâce à ce contrôleur que toute connestion sur le serveur sugar se fait en toute sécurité et que chaque utilisateur a des droits bien définis. Cela n'empêche qu'on rajoute des contrôleurs qui répondent à nos besoins tels que timesheet.php pour enregistrer les projets et les tâches de chaque employé, aussi pour pouvoir consulter son avancement et le temps qui passe sur chaque projet.

Aussi pour le module « recrutement » on a utilisé la même procédure on a utilisé basic search pour faire une recherche basique d'un candidat c'est à dire selon le nom seulement et

aussi Advanced search pour faire une recherche avancé selon le nom, créer par et le mail. En ce qui concerne les contrôleurs qu'on a ajouté on a rajouté des méthodes de recherche pour les candidats selon le diplôme les certifications les compétences, etc.

- *View :*

Les vues sont pour la manipulation d'affichage des informations dans le navigateur. Elles ne sont pas seulement limitées à des données HTML. Comme avec les contrôleurs il ya une classe par défaut appelé SugarView qui met en œuvre un grand nombre de la logique de base pour des vues telles que la manipulation des en-têtes et pieds de page.

En tant que développeur, si on veut créer une vue personnalisée qu'on place une vue. <view_name>. Fichier php dans une vue de sous-répertoire dans le module. Si un sous-répertoire vues n'existe pas, on doit en créer un. Dans le fichier, on doit créer une classe nommée: <Module>. Par exemple, pour un affichage de liste dans des candidats du module « recrutement » de la classe serait ContactsViewList. Il faut notez que la première lettre de chaque mot est en majuscule et toutes les autres lettres sont minuscules.

Principalement au niveau de l'affichage on a eu recourt pour des classes qui sont par défaut dans Sugar tel que :

- **Listviewdefs** (taille de l'affichage taille des tableaux les valeurs par défaut)
- **Editviewdefs** (il faudrait avoir au moins un champ non vide afin de pouvoir enregistrer un candidat)
- **Detailsviewdefs** (des boutons pour ajouter ou supprimer).

V.6. Phase de tests et validation

En informatique, un test désigne une procédure de vérification partielle d'un système informatique. Le but en est de trouver un nombre maximum de comportements problématiques du logiciel, car il est impossible de prouver qu'un logiciel fonctionne bien dans tous les cas. Plus d'erreurs sont trouvées, plus il y a de chances qu'il y ait davantage d'erreurs dans le composant logiciel visé. Les tests de vérification ou de validation visent à s'assurer que ce système réagit de la façon prévue par ses concepteurs (spécifications) ou est conforme aux attentes du client l'ayant commandé (besoins), respectivement.

Dans cette partie on va présenter une liste des modules de test, scénarios de test et le résultat obtenu à partir du tableau suivant :

Tableau 2. Liste des tests pour valider notre application

	Module	Scénarios de test	Résultat obtenu
1	Connexion à l'application	Entrer un login et un mot de passe correcte	Exécution correcte
		Entrer un login et un mot de passe faux	Exécution correcte
2	Connexion à la BD	Sélectionner des données de la BD	Exécution correcte
3	Gestion des données statiques	Insertion des données statique	Exécution correcte
		Consultation des données	Exécution correcte
4	Gestion des interfaces graphique	Insertion des données alphabétique à la place des numérique	Exécution correcte
5	Gestion des exceptions	Insertion des données existantes	Exécution correcte
		Sélection des données inexistantes dans la BD	Exécution correcte
		Transfert correct des informations entre les interfaces	Exécution correcte
		Mise à jour de BD à chaque modification	Exécution correcte
		Envoyer une requête au serveur	Exécution correcte
		Insertion au BD	Exécution correcte

V.7. Conclusion

Dans ce chapitre nous avons présenté en détails le développement de notre système. Nous avons commencé par présenter l'environnement matériel et logiciel, les choix techniques et les outils de travail sur lesquels se base notre application.

De plus, nous avons exposé l'organigramme chronologique du déroulement de notre travail et enfin nous avons conclu par les scénarios de test et de validation de l'application.

Chapitre VI. Interface de l'application

VI.1. Introduction

Comme toute application, notre application doit être facile à utiliser et compréhensive par tout type de cible. Dans ce qui suit, nous présentons quelques interfaces pour faire expliquer les principales fonctionnalités des différents modules de notre projet.

VI.2. Interfaces de l'application

Cette figure représente l'interface principale du module gestion des candidats (Recrutement). Elle présente un formulaire de recherche selon compétence, expérience, diplôme, etc.

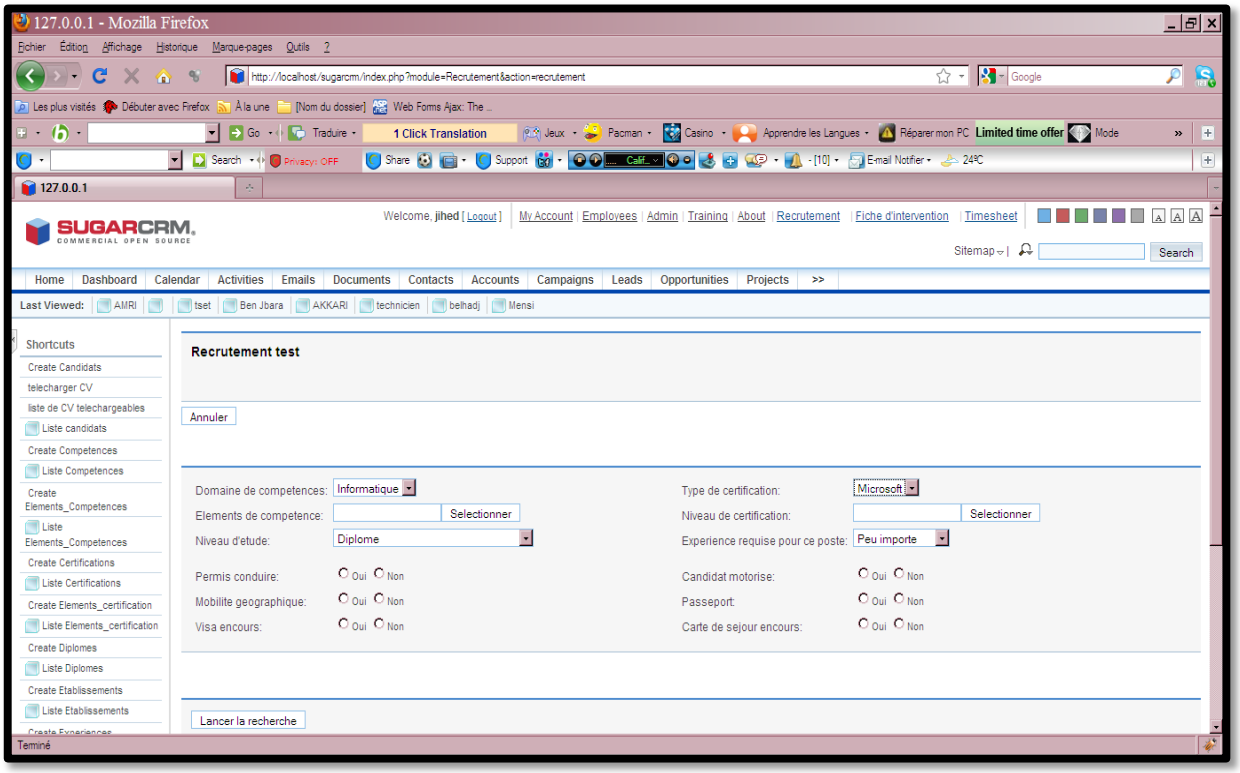


Figure 28. Recherche d'un candidat

Et dans la figure suivante on va faire une recherche selon les compétences comme exemple.

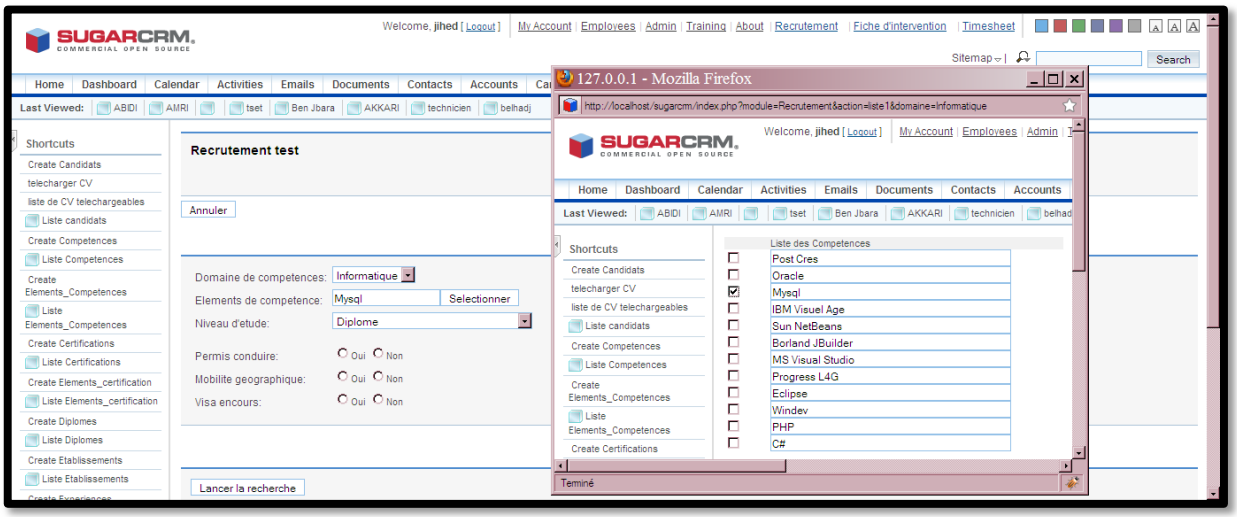


Figure 29. Sélection d'un élément de compétence

Le gestionnaire de recrutement sélectionne d'abord un domaine de compétence (Informatique, Gestion, Finance). Puis il sélectionne un élément de compétence du domaine sélectionné pour exemple (Mysql). Et il lance la recherche.

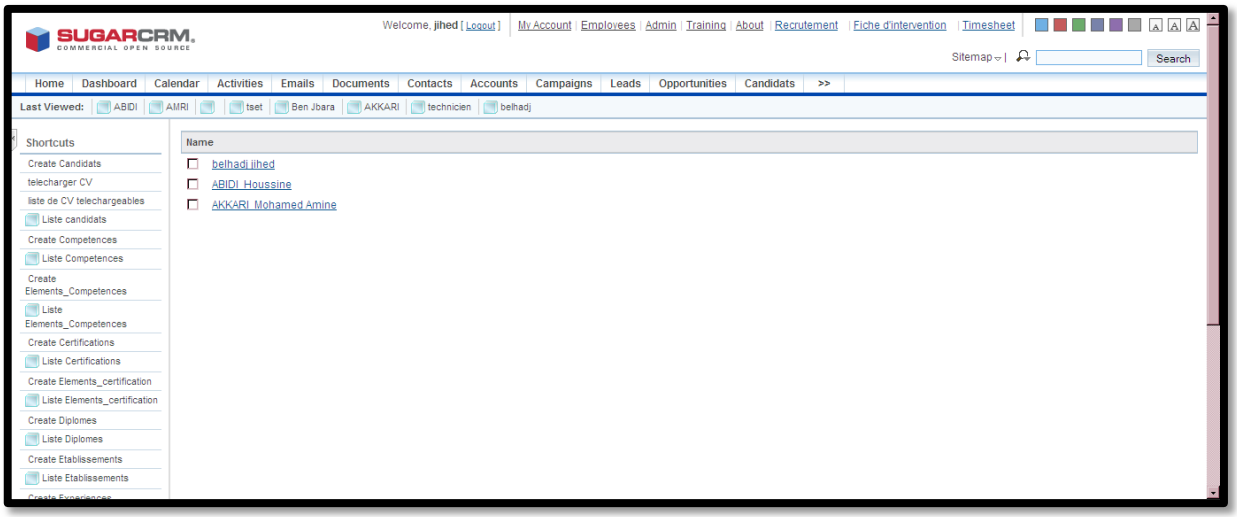


Figure 30. Résultat

Voici le résultat de la recherche, une liste des candidats qui ont une expérience «Mysql ».

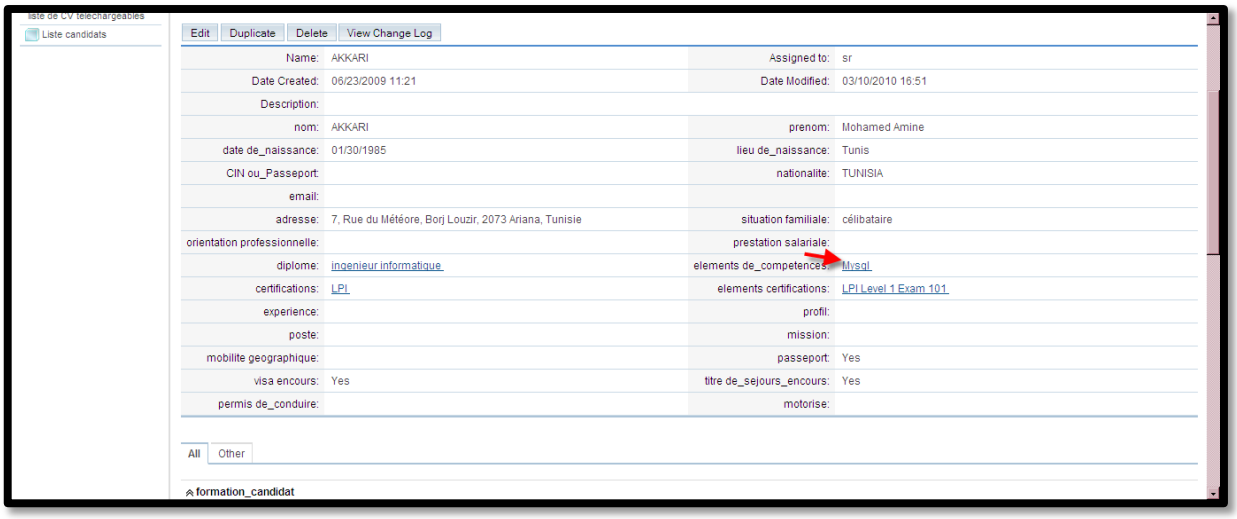


Figure 31. Vérification du résultat

Le gestionnaire de candidat peut vérifier on utilisant la page view de candidat ou il peut consulter le CV.

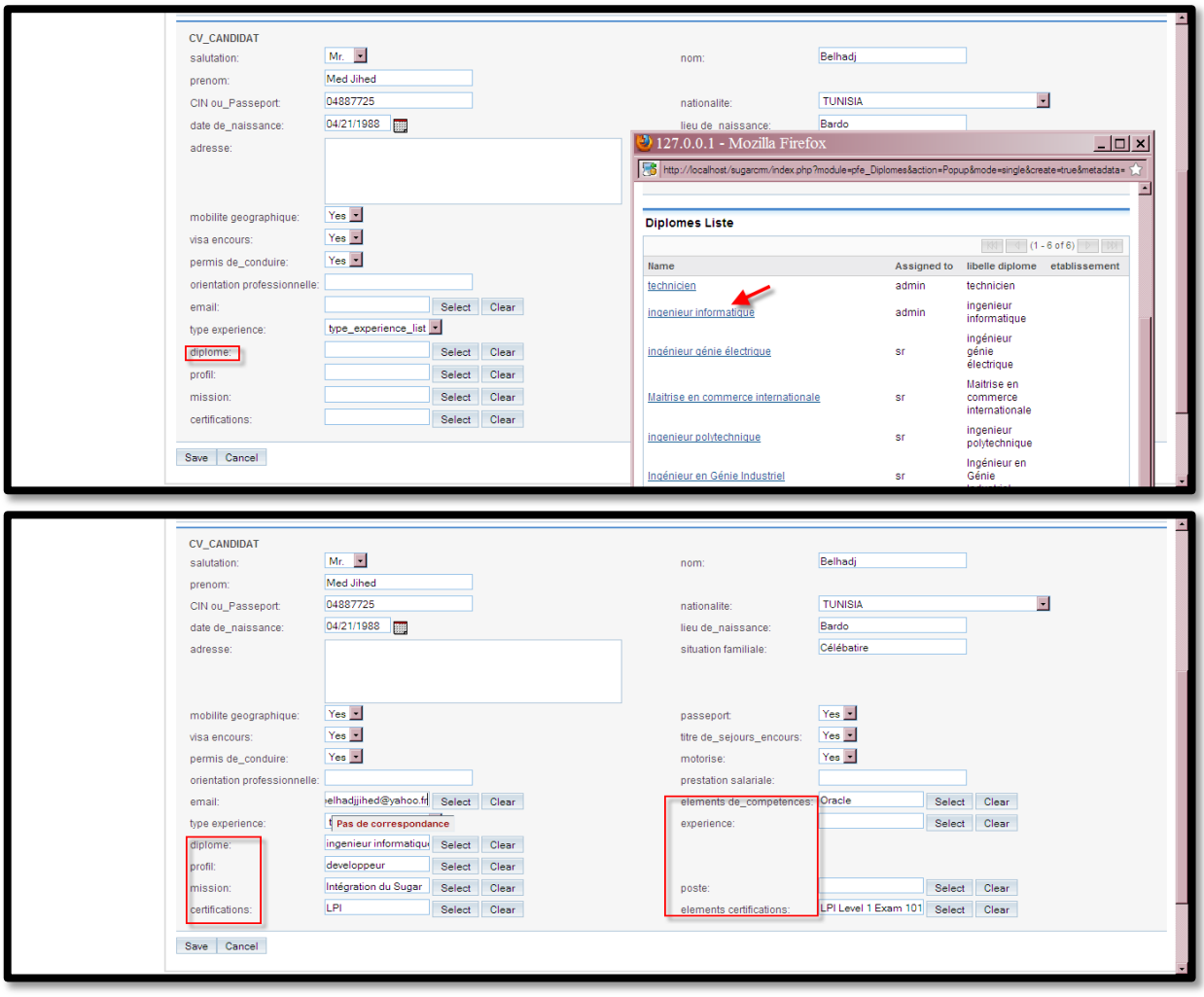


Figure 32. Ajouter un candidat

D'autre part voici l'interface d'ajout d'un candidat, le candidat doit remplir ces attributs avec des sous listes de compétences, diplôme, expérience, etc. Dans notre exemple on a un popup qui présente une liste de diplômes. Cette interface permet pour chaque candidat de télécharger son CV et voir l'historique de ces CV.

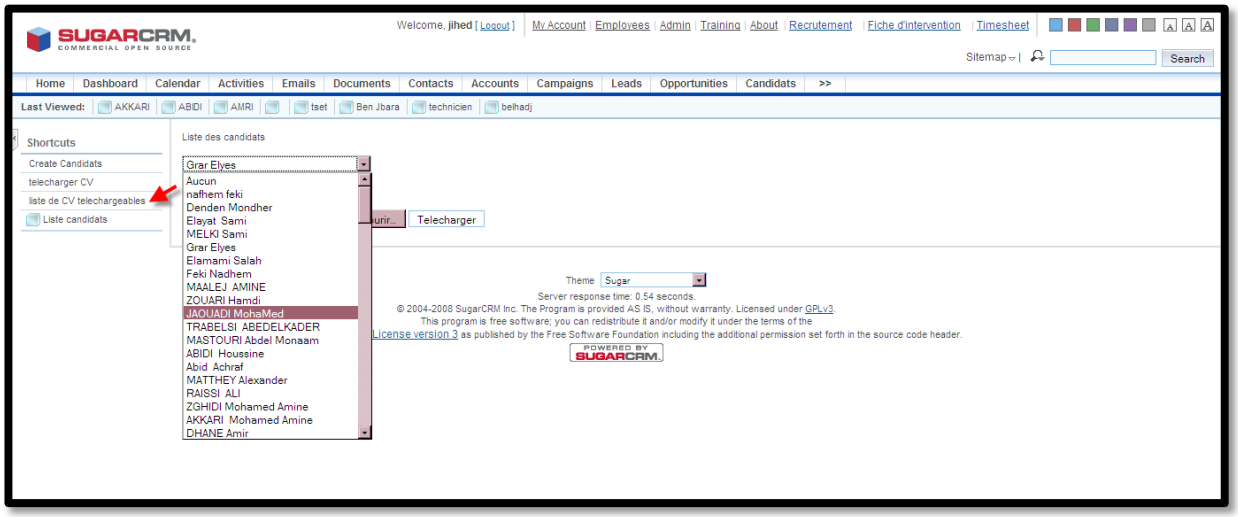


Figure 33. Déposer les cv et voir historique cv

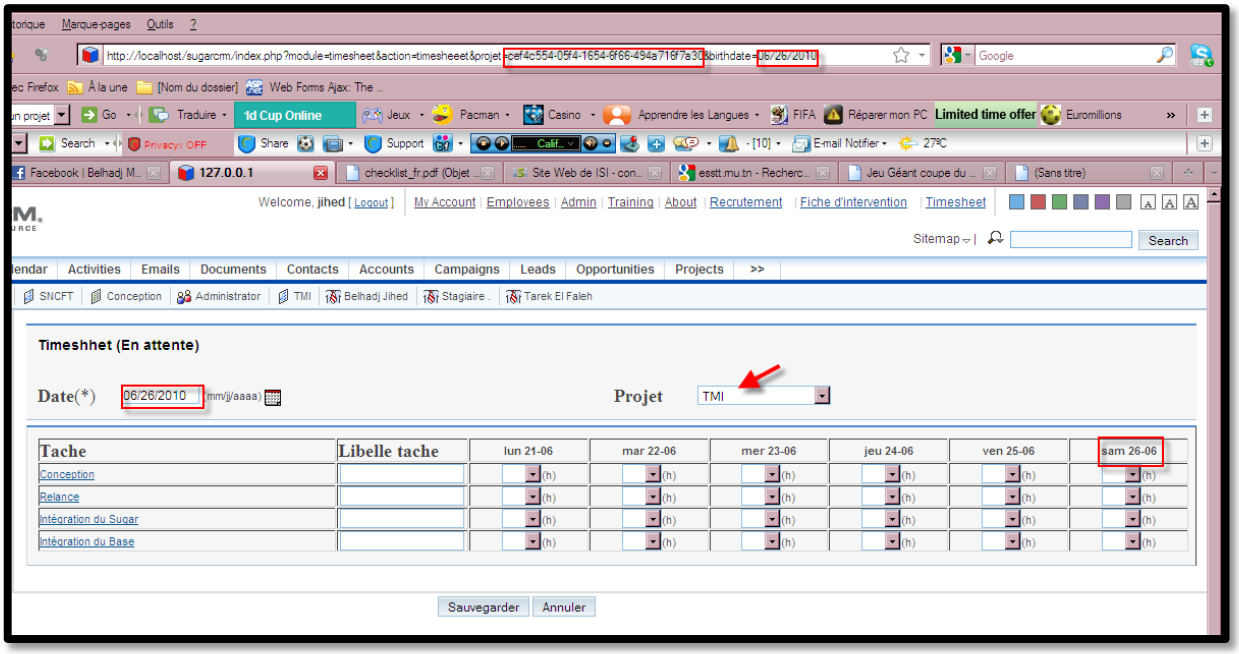


Figure 34. Interface remplir Timesheet

Voici l'interface de remplir son timesheet, cette interface peut être recharger suivant deux champs clés la date et le projet.

Après la sélectionne d'une date précis e notre page se recharge et une semaine s'affiche (Si vous sélectionnez le 26/06/2010 qui est un samedi notre système affiche tout les jours de cette semaine). D'autre part on a un autre filtre selon les projets sélectionnés.

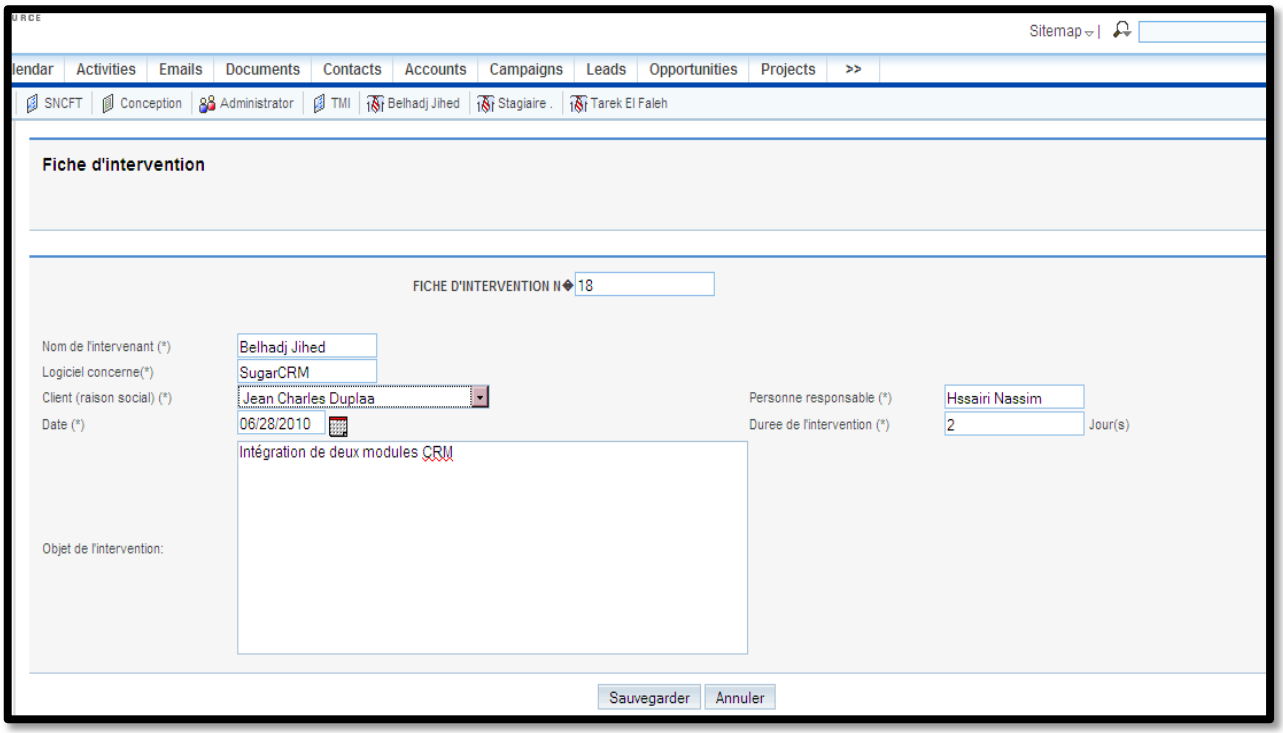


Figure 35. Remplir fiche d'intervention

Cette interface est une copie identique da la version papier de la fiche d'intervention, l'employée après chaque intervention il remplit ce formulaire pour être valider par l'administration.

VI.3. Conclusion

Dans ce chapitre nous avons présenté les principales interfaces de notre application et nous avons également exposé les différentes étapes à suivre par l'utilisateur, et il est à noter que ce travail nous a posé quelques problèmes mais enfin nous sommes contentés, dans ce travail, d'arriver à réaliser les principaux modules en répondan t aux contraintes et en répondant parfaitement à l'attente de l'entreprise.

Conclusion Générale

« *La pérennité d'une société n'est pas forcément liée à son gigantisme mais avant tout à son pragmatisme et à ses hommes*. »

C'est sur ces bases, que **DISCOVERY Informatique** a conquis de nombreux clients prestigieux à l'échelle nationale et internationale et a bâti sa notoriété sur leur totale satisfaction. Au-delà de l'expertise, de la technologie et du savoir-faire dans le domaine de l'intégration des ERP, base indispensable de son métier.

Forte d'une équipe dirigeante pérenne, de collaborateurs assidus, d'une stratégie de développement claire, **DISCOVERY Informatique** aborde l'avenir avec réalisme autour de projets ambitieux : recrutement de nouvelles compétences, rapprochement auprès de grands partenaires « métier » et de nouveaux éditeurs, mener des projets d'intégration réussis, déploiement à l'international, continuité en matière de recherches et développement.

C'est dans cette perspective que nous avons opté à l'informatisation de la tâche de recrutement et d'autre part la gestion des employées . Et on a développé cette application intégrable dans le logiciel libre SugarCRM, pour assurer la gestion de la relation avec le client. Notre projet permet au responsable de recrutement de Discovery, de déceler en quelques simples click les profils candidats adéquats aux critères exigés pour un poste donné ou un éventuel recrutement, aussi il peut affecter et gérer son groupe , à partir de l'interface SugarCRM.

Ces nouveaux fonctionnalités de SugarCRM, développée par ce projet offre à ses utilisateurs un moteur de recherche bien évolué et développé, la possibilité d'entrer des critères précis de recherche, pour obtenir des résultats ciblés. Par ailleurs, il permet d'informatiser la fiche d' intervention, pour qu'elle soit plus pratique, sécuriser, et fiable et une interface Timesheet destinées ou employées pour gérer le déroulement des projets et la gestion des affectations.

Ce travail nous a donné l'opportunité de s'initier à la vie professionnelle dans un milieu réel et avoir un début d'expérience significatif. Ce stage nous a appris comment dès le matin on peut prendre le parti de la gaieté, comment réussir de bonnes relations pour assurer un travail de groupe, comment compter sur soi pour résoudre les problèmes au cas où ils se présentent, comment être méticuleuses dans notre travail, comment être attentives aux indications de nos supérieurs, comment être bien organisées pour accomplir dans les meilleurs délais, et meilleures conditions les tâches qui nous sont confiées.

Cette application devenue une référence, dans les sites web dynamiques, nous a permis de découvrir le monde professionnel, le plaisir du développement des applications, la créativité, le monde passionnant de l'informatique dans la réalité des entreprises.

Durant cette période de stage, nous avons appris de nouveaux langages de programmation PHP5, Ajax, JavaScript… Nous avons appris à maîtriser le développement sur le logiciel SugarCRM, la manipulation de la base de données avec MySQL.

Nous avons aussi rencontré quelques problèmes au début du projet, faute de documentation sur le logiciel, manque d'expérience et d'expertise dans la manipulation des logiciels libres. Il reste beaucoup d'améliorations et extensions qui peuvent être apportées à ce module.

En conséquence de tout ce qui précède, nous espérons avoir été au niveau de la tâche qui nous a été affectée.

Références bibliographiq

Ouvrages

Auteur : Julien Maquet & Xavier Rigal

Guide Administrateur

Guide pour débutants SugarCRM

Sugar Developer Guide Version 5.2, 2008

Cours

Auteur : Guillare Rossolini

PHP 5

Auteur : Bruno Mermet

Architecture MVC

Auteur : Fleur-Anne Blain

Qu'est ce qu'un CRM ?

Références Netographiq

[1]: http://developers.sugarcrm.com/documentation.php

[2]: http://www.sugardev.net/search/node/sugarcrm

[3] : http://www.3acrm.com/logiciel-crm/solutions/sugarcrm/presentation.html

[4] : http://www.sugarforge.org/content/open-source/

[5] : http://www.commentcamrche.com

[6] : http://www. Developpez.com

[7] : http://www. wikipedia.com

[8] : http://www.siteduzero.com/

Annexe A : Vue d'ensemble différents répertoires et fic SugarCRM

- *Aperçu des modules du Framework :*

Tous les modules, modules de base ou les modules que vous créez et installez par l'intermédiaire du moduleLoader, sont mis dans le dossier : <sugar root> / modules.

Généralement les modules sont créés lorsque vous avez à représenter un objet en sugar, tels que le module 'Contact', et l'objet a des points de données qui doivent être stockés dans la base de données, ainsi qu'avoir une interface utilisateur pour l'utilisateur, pour créer, modifier et supprimer des enregistrements pour l'objet.

Voyons un module Framework et comment il s'intègre dans l'ensemble de la plate-forme de sugar. Dans la section Aperçu de la Plate-forme, nous montrons un exemple d'un appel qui serait typique d'une « DetailView action » à l'intérieur d'un module. Il existe cinq grandes actions pour un module :

- **ListView :** Ce contrôleur d'action existe pour fournir à l'utilisateur le formulaire de la recherche et la recherche de résultats d'un module. À partir de cet écran, l'utilisateur peut prendre des mesures telles que la suppression, l'exportation et la mise à jour de plusieurs enregistrements de masse ou d'exercice, dans un enregistrement spécifique pour afficher et modifier les détails. L'utilisateur est tombé dans ce point de vue, par défaut quand il clique sur un des onglets en haut de l'application. Les fichiers de chaque module décrivent le contenu de la liste de recherche et de vue.

- **DetailView :** A DetailView offre une lecture-seule de vue détaillée d'un objet particulier. Typiquement, l'utilisateur aura accès à un enregistrement DetailView par l'intermédiaire d'un ListView. L'utilisateur peut consulter les détails de l'objet lui-même et, en dessous des détails, il va voir la liste des articles connexes qui sont dénommés «SubPanels» dans Sugar.

Les SubPanels agissent en tant que mini-ListView d'objets qui sont liés à l'objet parent consultée avec le DetailView action. Par exemple, les tâches assignées à un projet ou aux

Contacts pour une Opportunity, apparaîtra dans SubPanels ci-dessous le projet ou les oppotunity.
<module> / métadonnées / detailviewdefs.php : définit un module de la page layout DetailView.
<module> / métadonnées / subpaneldefs.php : définit les SubPanels qui sont visibles dans le cadre du module de la page DetailView.

- **EditView :** L'action EditView est consultée chaque fois qu'un utilisateur crée un nouvel enregistrement ou à l'édition des détails d'un existant. Il est possible d'accéder directement à l'EditView à partir de la ListView.

<Module> / metadata / editviewdefs.php : définit un module de EditView à partir de la page layout.

- **Save :** Ce contrôleur de l'action est traité à chaque fois que l'utilisateur clique sur le bouton «Save» de l'EditView d'un enregistrement.

- **Delete :** Cette action est traitée à chaque fois que l'utilisateur clique sur le bouton «Supprimer» de la DetailView d'un enregistrement ou, comme un cas particulier, à partir d'un point figurant dans un SubPanel.

Ces actions sont conduites par le framework de l'interface utilisateur, ainsi que le framework repose sur les fichiers métadonnées dans les modules demandés.

• <Module> / metadata / listviewdefs.php décrit la présentation de la ListView.
• <Module> / metadata / searchdefs.php formulaire de recherche décrit les onglets au-dessus de la ListView.
• <Module> / metadata / editviewdefs.php décrit la présentation de la EditView.
• <Module> / metadata / detailviewdefs.php décrit la présentation de la DetailView.

Outre les fichiers de l'action décrite ci-dessus vous avez probablement remarqué certains autres fichiers situés dans le dossier :

- **forms.php:** Ce fichier contient deux fonctions pour rendre spécifiques le JavaScript, pour validation ou autres actions que vous pouvez accomplir au cours des (edits/saves) modifications/sauvegarde. Par défaut, vous pouvez les laisser vides et ils retournent : return";

- **menu.php:** Ce fichier est responsable de l'affichage des raccourcis de menu, vous avez l'habitude de les voir sur le côté gauche de l'écran. Par défaut, vous avez généralement un lien pour créer un nouvel enregistrement, et un lien vers ListView pour la recherche.

- **Popup.php:** Ce fichier agit entant que couverture (wrapper)de la classe Popup central situé sous le dossier utils. Il est appelé, si jamais un autre module veut obtenir une popup-list de documents à partir d'un module connexe. La classe Popup centrale utilise ensuite le Popup_picker.html et le fichier <MODULE_NAME>/Métadonnées/popupdefs.php pour rendre la popup.

- **Metadata:** Comme de plus en plus de métadonnées et l'extension qui a été construite dans la plate-forme de sugar, des spécifiques modules de fichiers de métadonnées ont été ajoutés à ce dossier. Certains des plus importants dossiers en ce répertoire comprennent:

Additionaldetails.php : définit le contenu de la popup s'affichant dans la ListView.

Listviewdefs.php : définit les colonnes affichées dans la page du module ListView.

Popupdefs.php : définit les champs de recherche et la liste des colonnes pour un module de popup.

SearchFields.php : définit la recherche de base et recherche avancée des Formulaire vues dans les pages ListView.

Studio.php : définit la manière dont l'outil Studio interagit avec un module de fichiers de métadonnées.

Maintenant, regardons les différents modules dans le dossier module et comme exemple «pfe_candidats » pour compléter la vue d'ensemble de la structure du module aussi les fichiers dans le sous-dossier metadata.

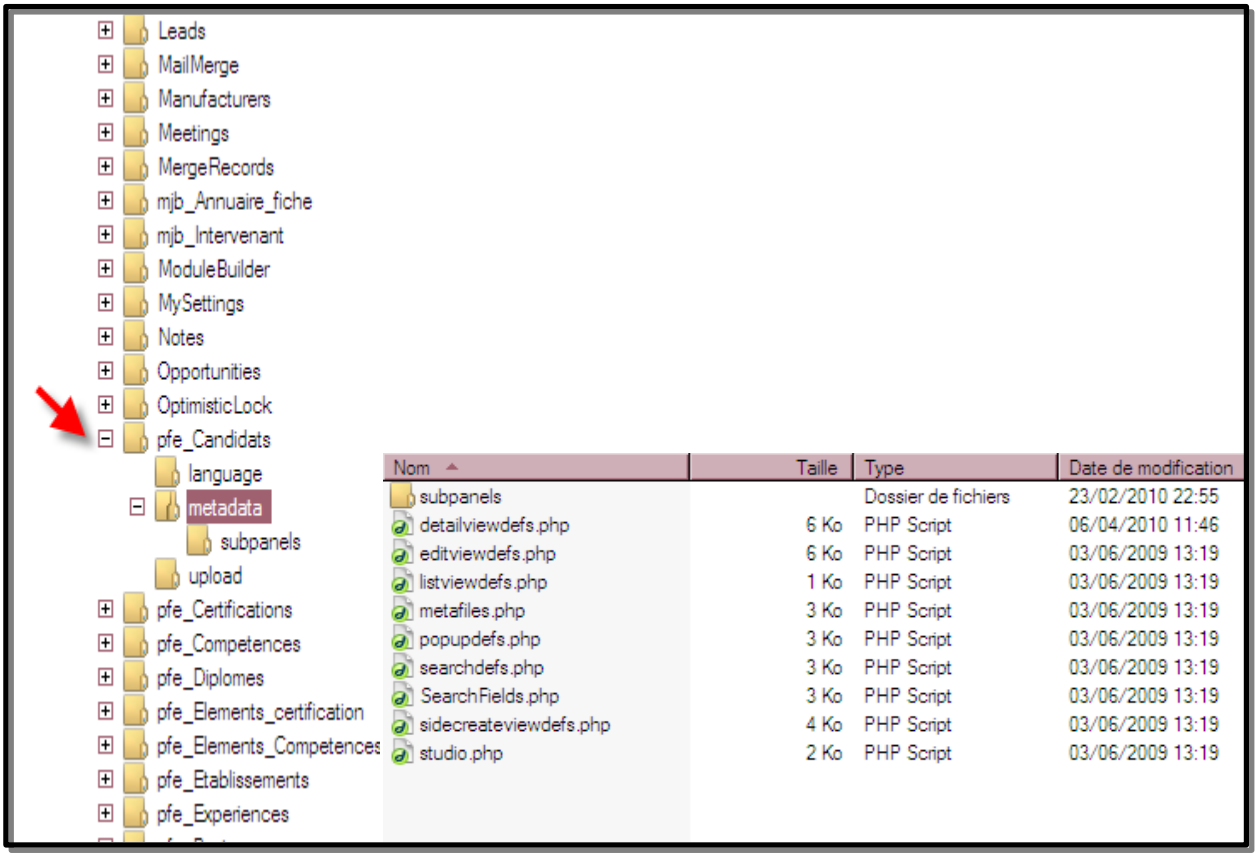


Figure 36. Fichiers sous le dossier metadata

Annexe B : Le développement du m
recherche des candidats

```
<body>
<table cellspacing="0" cellpadding="0" border="3" width="100%" class="listview">
<tbody><tr height="20">
<td nowrap="" width="100%" class="listViewThS1" scope="col">
<div align="left" width="100%" style="white-space: nowrap;">
<a class="listViewThLinkS1
"href="index.php?module=pfe_Recrutements&amp;action=index&amp;lvso=DESC&amp;pfe_
Recrutements2_pfe_Recrutements_ORDER_BY=name">Name</a>
</div>
</td> </tr>
    <?php
require_once ('Connections/connexion_recrutement.php');
mysql_select_db ($database_connexion_recrutement, $connexion_recrutement);
$and = '';$where = '';
$from = ' FROM pfe_candidats';
    if(isset ($_POST['elem_comp']) && $_POST['elem_comp'] != "")
{
// Controle sur le champ elements de competences ///
    $where .= $and."
pfe_elements_competences.id=pfe_candida_competences.pfe_elements_competences_idb AND
pfe_candidats.id=pfe_candida_competences.pfe_candidats_ida AND
pfe_elements_competences.libelle_element_competence='".$_POST['elem_comp']."'";
    $from .= ", pfe_elements_competences, pfe_candida_competences";
```

```
$and = " AND ";
}
// Controle sur le champ elements de certification ///
if(isset ($_POST['elem_cert']) && $_POST['elem_cert'] != "")
{
$where .= $and."
pfe_elements_certification.id=pfe_candidaertification.pfe_elements_certification_idb AND
pfe_candidats.id=pfe_candidaertification.pfe_candidats_ida AND
pfe_elements_certification.libelle_cert='".$_POST['elem_cert']."'";
$from .= ",pfe_elements_certification ,pfe_candidaertification ";
$and = " AND ";
}
// Controle sur le champ diplome ///
if(isset ($_POST['diplome']) && $_POST['diplome'] != -1)
{
$where .= $and." pfe_diplomes.id=pfe_candidapfe_diplomes.pfe_diplomes_idb AND
pfe_candidats.id=pfe_candidapfe_diplomes.pfe_candidats_ida AND
pfe_diplomes.libelle_diplome='".$_POST['diplome']."'";
$from .= ", pfe_diplomes, pfe_candidapfe_diplomes";
$and = " AND ";
}
if(isset ($_POST['mots']))
{
$where .= $and." 'pfe_candidats.adresse' LIKE '%".$_POST['mots']."%'";
$and = " AND ";
}
if(isset ($_POST['permis']) && $_POST['permis'] != "")
```

```
{  if( $_POST['permis'] == "O")
   {
        $where .= $and."pfe_candidats.permis_de_conduire = 'on'";
        $and = " AND ";
     }
   else
     {
        $where .= $and."pfe_candidats.permis_de_conduire = 'off'";
        $and = " AND ";
     }
}
if(isset ($_POST['niveau_exp']) && $_POST['niveau_exp'] != -1)
{
     $where .= $and." pfe_candidats.type_experience = '".$_POST['niveau_exp']."'";
     $and = " AND ";
}
$query = "SELECT distinct pfe_candidats.id , pfe_candidats.name , pfe_candidats.prenom";
if($from != "")
{
     $query .= $from;
}
if($where != "")
{
     $query .= ' WHERE ' . $where;
}
$res=mysql_query($query);
```

```
$table = '<table width="100%">';
while($row=mysql_fetch_assoc($res))
{
    $table .= '<tr>';

    $table .= '    <td nowrap="" bgcolor="#ffffff" width="1%" class="oddListRowS1">
                        <input type="checkbox" value='.$row[id].' name="mass[]"
class="checkbox" onclick="sListView.check_item('.this.', document.MassUpdate)"/>
    </td>';
      $table .= '
<td bgcolor="#ffffff" align="left" valign="top" class="oddListRowS1" scope="row">
    <a
href="http://localhost/sugarcrm/index.php?module=pfe_Candidats&offset=1&stamp=124023887
5095607600&return_module=pfe_Candidats
&action=DetailView&record='.$row['id'].'">'.$row['name']." ".$row['prenom'].'</a></t>';
    $table .= '  </tr>';
}
$table .= '</table>';
echo $table;
?> </tbody></table>          </td>
    </tr></tbody></table>
</body>
</html>
```