Année universitaire : 2017-2018

1ère année cycle d'ingénieur, filière informatique

Rapport de projet de programmation orientée objet. :

Conception et réalisation de projet de gestion d'une agence d'immobilière.

Auteurs :
M. KEFI LASSAD
M. Ouni HENI

Classe :
1 ING-INFO C

17 Avril 2018

Table des matières

Table des figures

Introduction

Un agent immobilier est un intermédiaire dans les transactions portant sur des biens immobiliers : ventes et locations. Cet agent peut être un travailleur indépendant ou une entreprise, l'agence immobilière, employant des négociateurs.
Il est généralement en contact avec ses confrères de la même branche, avec les mairies et communautés urbaines, les collectivités, les institutions, les avocats et les tribunaux.
Il peut être varié. Les agents immobiliers ne sont pas limités par la loi à un domaine. L'expérience et leurs préférences ou modalité de rémunération les fait souvent se spécialiser ;
Certains répondent aux besoins des particuliers : logements, garages, terrains ; dans une diversité de situations urbaines ou rurales, de biens récents ou anciens, communs ou prestigieux.
10% des agents immobiliers avec peu de mixité sont spécialisés en immobilier d'entreprise : bureaux, entrepôts, terrains industriels, commerces, activités spécifiques.
Tout au long de ce rapport, nous allons examiner les différentes étapes de réalisation de notre projet.
Le présent rapport est composé de 4 parties essentielles :

- Le premier chapitre intitulé « **Présentation générale du projet** » est réservé à la présentation de contexte général du projet, ainsi qu'une description des différentes solutions proposées.
- Le deuxième chapitre intitulé « **Analyse et conception** » comporte une identification des différents acteurs et cas d'utilisations du système à construire et réservé à la conception en précisant la structure interne des éléments et leurs relations les uns par rapport aux autres.
- Le troisième chapitre intitulé « **Réalisation** »» rapporte une description des différents outils de développement utilisés pour la réalisation du projet avec une exposition de quelques interfaces de l'application

Finalement, une conclusion générale du projet en présentant les perspectives pour une éventuelles extension de ce travail.

Chapitre 1

Contexte du projet

1.1 Introduction :

Dans ce chapitre, nous allons commencer par définir quelques notions de base, indispensables pour la compréhension du cadre générale du projet. Ainsi, nous allons spécifier d'une façon détaillée les besoins fonctionnels et les besoins non fonctionnels de notre application.

1.2 Cadre de projet :

Le projet a été proposé dans le cadre d'un projet de programmation orienté objet. Le projet affecté est l'élaboration une application de gestion d'une agence d'immobilière.

1.3 Travail demandé :

On souhaite faire, en C++, une solution de gestion d'une agence immobilière.
Cette agence met en relation les personnes qui sont à la recherche d'un logement à acheter ou à louer et celles qui en ont de disponibles soit à la vente soit à la location.
L'agence peut aussi gérer les problèmes quotidiens (des réclamations / des suggestions) des propriétaires et des locataires en leur servant d'intermédiaire. Dans le cadre de la gestion locative, la mission de l'agence est de : rechercher un locataire, rechercher un logement, établir les contrats, les quittances de loyer, etc. En contre partie du temps gagné par les clients, pour une vente ou une location d'un bien immobilier (maison, dépôt, appartement, terrain, immeuble, etc.), l'agence perçoit un pourcentage sur la valeur du bien ou du loyer à la signature de l'acte notarié ou du contrat de location.
Définir les classes et les méthodes pour informatiser ce système ; tel que : la gestion des

biens immobiliers, des clients, des employés, tout en comptabilisant les dépenses et les bénéfices de l'agence ainsi que ses frais.

1.4 Solution proposée

Afin de trouver une solution pour la gestion de l'agence d'immobilière, nous avons choisi de développer une application pour des raisons de sécurité afin de ne pas divulguer des informations secrètes à des sous-traitants, des raisons technique pour faciliter la résolution des problèmes rencontrés. L'application doit être souple, paramétrable et conviviale, en d'autres termes, elle permet de répondre aux exigences suivantes :

— Les informations doivent être à jour, fiables, cohérentes et uniques.
— Automatiser la gestion de l'agence.
— Informatiser la demande/annulation des achats ou des locations.
— Informatiser la recherche les logements.
— Informatiser l'établissement des contrats.
— Informatiser la gestions les clients/des employées.
— Définir des droits d'accès (Agence, Client, Propriétaire).

1.5 Spécification des besoins :

Cette étape est caractérisée par la production d'un document des spécifications décrivant les principales fonctionnalités du système à développer. Les spécifications sont déterminées à partir **des besoins fonctionnels** qui présentent les différentes fonctions du système, **des besoins non fonctionnels** sont les contraintes que le système doit respecter

1.5.1 Les besoins fonctionnels :

Les besoins fonctionnels présentent les actions que le système doit exécuter et pour ceci, le système à réaliser doit satisfaire les exigences de la totalité des utilisateurs. Le système doit permettre :

- Afficher les logements, leurs états, leurs disponibilités et leurs propriétaires.
- gestion des logements.
- gestions des utilisateurs.
- Élaborer des contrats entre les locataires et les propriétaires.
- Gestion des réclamations/suggestions.
- Gestion de profil.
- Passer des commandes des achats/locations.
- Classifier les utilisateurs et les affecter des privilèges.

Nous distinguons les trois acteurs suivants :

- **L'agence :**
  Cet acteur a le rôle de :
  — Gérer les utilisateurs.
  — Gérer les immobiliers.
  — Gérer les employées.
  — Gérer les réclamations/suggestions.
  — Élaborer des contrats.
- **Le propriétaire :**
  Cet acteur peut :
  — Ajouter un logement.
  — Gérer son profil.
  — Consulter ses contrats.
  — Réclamer ou suggérer.
- **Le propriétaire :**
  Cet acteur peut :
  — Consulter un logement.
  — Gérer son profil.
  — Consulter des contrats.
  — Réclamer ou suggérer.
  — Louer/Acheter un logement.

1.5.2 Les besoins non fonctionnels

En plus des besoins fonctionnels,le futur système doit répondre aux contraintes suivantes :

— **La rapidité de traitement :** vu le nombre important des transactions quotidiennes, il est impérativement nécessaire que la durée d'exécution des traitements s'approche le plus possible du temps réel.
— **La performance :** un logiciel doit être avant tout performant et capable de répondre à toutes les exigences des usagers d'une manière optimale.
— **La convivialité :** le futur logiciel doit être facile à utiliser. En effet, les interfaces utilisateurs doivent être conviviales c'est-à-dire simples et ergonomiques.
— **La sécurité :** l'application doit être protégée contre des accès non autorisés en renforçant la sécurité et la confidentialité des données.

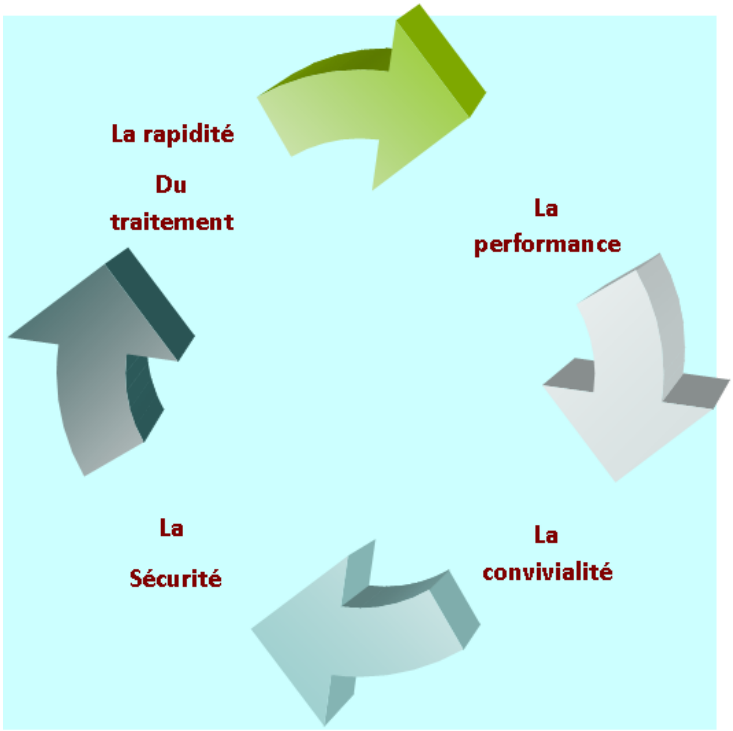

FIGURE 1.1 – Les besoins non fonctionnels.

1.6 Conclusion :

Dans cette section, on représente une analyse du projet qui est l'origine de toute activité de développement. Elle permet, principalement, la résolution de la complexité du problème tout en précisant les besoins fonctionnels et non fonctionnels de l'application. Ces besoins seront mis en relief, par la suite, en se basant sur les diagrammes des cas d'utilisations qui servent, essentiellement, à définir d'une manière assez rigoureux les processus de l'application.

Chapitre 2

Analyse et conception

2.1 Introduction

Dans ce chapitre, nous abordons la phase d'analyse et spécification des besoins. Ainsi, nous présentons les acteurs de notre application. Nous utilisons le logiciel Star UML comme moyen simple et compréhensible afin de décrire les principaux cas d'utilisation, scénario, intervenants.

2.2 Identification des acteurs :

2.2.1 Définition du acteur :

— l'acteur est l'agent qui interagit avec le cas d'utilisation, il peut être un être humain ou un autre système externe, il envoi des événements pour accueillir de la part du système.
— l'acteur n'identifie pas l'élément physique mais le rôle joué par celui-ci.

2.2.2 Les acteurs du système :

Nous distinguons trois acteurs en interaction avec notre système dont chacun à un profil particulier en termes de fonctionnalités.

Les acteurs principaux sont :

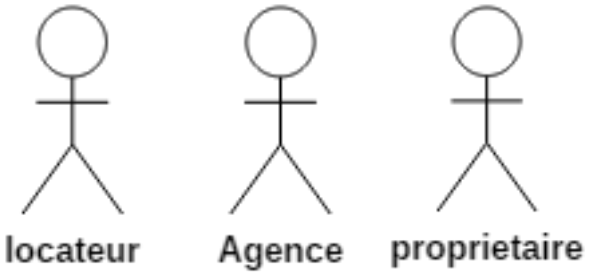

FIGURE 2.1 – Les acteurs de l'application.

2.3 Spécification des besoins

Il existe plusieurs langage de modélisation et description des besoins tel que :
Business Process Modelling Notation, Energy Systems Languages, IDEF, Unified modelling language (UML) . Afin de bien modéliser les besoins de notre projet nous allons utilisés le langage de modélisation UML. Choix de modélisation de langage de conception UML : UML (langage de modélisation unifiée) est un langage de modélisation graphique. Il est utilisé en développement logiciel et en conception orientée objet. UML est utilisée pour spécifier, visualiser, modifier et construire les documents nécessaires au bon développement d'un logiciel orienté objet.
UML propose **trois types de diagrammes** :

— **Les diagrammes statiques :**
  - Diagramme de classe : il représente les intervenants dans le système.
  - Diagramme d'objets : il représente les instances de classe utilisées dans le système.
  - Diagramme de composants : il permet de montrer les composants du système d'un point de vue physique.
  - Diagramme de déploiement : il sert à représenter les éléments matériels et la manière dont les composants du système sont répartis sur ces éléments matériels et interagissant entre eux.

— **Les diagrammes dynamiques :**

- Diagramme de séquence : il représente les séquences du déroulement des traitements et des interactions entre le système et les acteurs.
- Diagramme de communication : représentation simplifiée d'un diagramme de séquence se concentrant sur les échanges de message entre les objets.
- Diagramme de collaboration : il montre les interactions entre objet.

— **Les diagrammes comportementaux :**

- Diagramme états transition : permet de décrire sous forme de machine à état finis le comportement du système ou de ses composants.
- Diagramme d'activité : permet de décrire sous forme de machine à états finis le comportement du système ou de ses composants.
- Diagramme de cas d'utilisation : il permet d'identifier les possibilités d'interaction entre le système et les acteurs (intervenants extérieurs au système), c'est-à-dire toutes les fonctionnalités que doit fournir le système.

Au cours la conception de notre interface, nous avons établis les diagrammes suivants :

- **Diagramme de cas d'utilisation.**
- **Diagramme de classe.**

2.4 Diagramme de cas d'utilisation :

2.4.1 Définition :

Un diagramme de cas d'utilisation est un diagramme qui permet les interactions des acteurs externes avec le système et la manière dont ils vont l'utiliser, Ce diagramme permet de mettre en place et de comprendre les besoins du client. Il est composé par :

— Des acteurs : ce sont des personnes ou un système extérieur à l'application et qui interagissent avec elle, *exemple* : Un acteur externe : client.

— Les cas d'utilisation de systèmes : sont les fonctionnalités du système à construire, ils peuvent êtres en relation avec les acteurs et avec d'autres cas d'utilisations, *exemple :* S'authentifier : le client doit saisir son login et son mot de passe pour accéder à son compte.

— Le système : représente l'application et les cas d'utilisations.

— Les relations : sont généralement orientés, elles permettent de relire les acteurs par les cas d'utilisations.

2.4.2 Le diagramme de cas pour chaque acteur :

Ce diagramme illustre les différents acteurs (l'agence, le locataire et le propriétaire) avec leurs principaux tâches.

Ce diagramme représente le diagramme de cas d'utilisation de l'**agence** ; Ce dia-

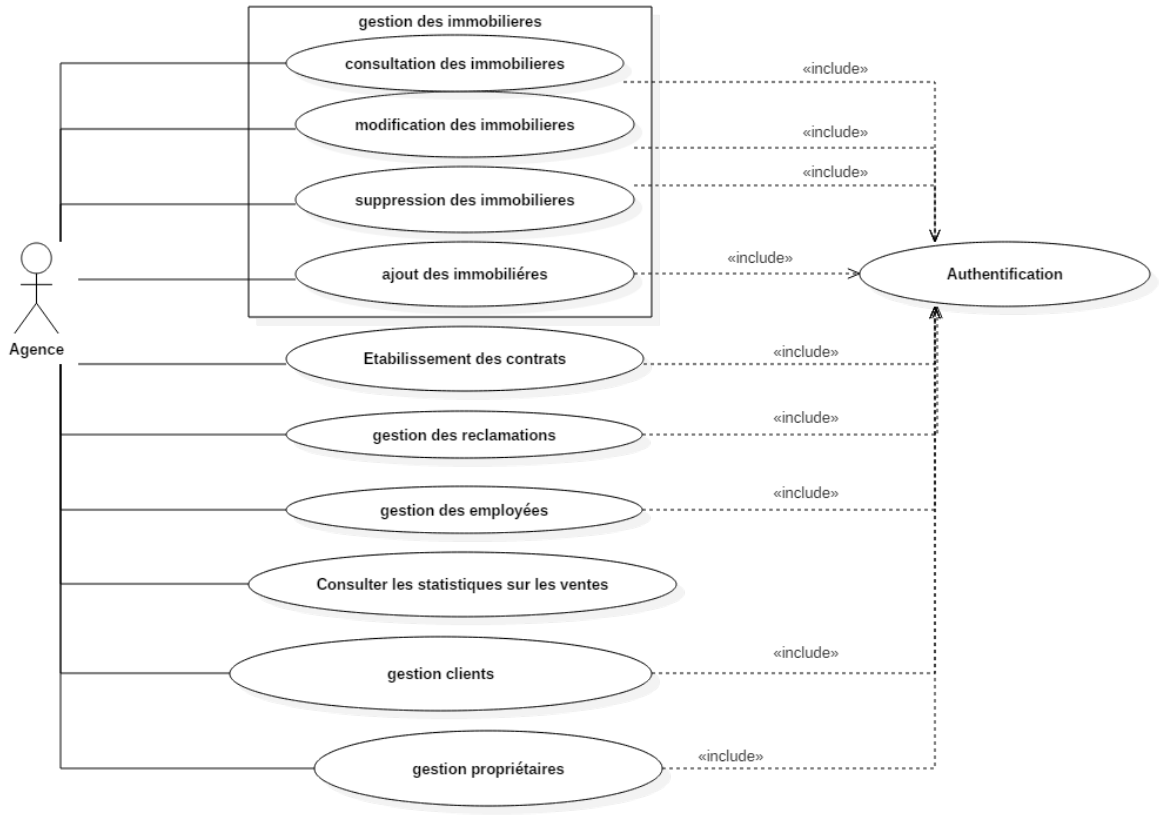

FIGURE 2.2 – Diagramme de cas d'utilisation général de l'**agence**

gramme représente le diagramme de cas d'utilisation de le**propriétaire** ;

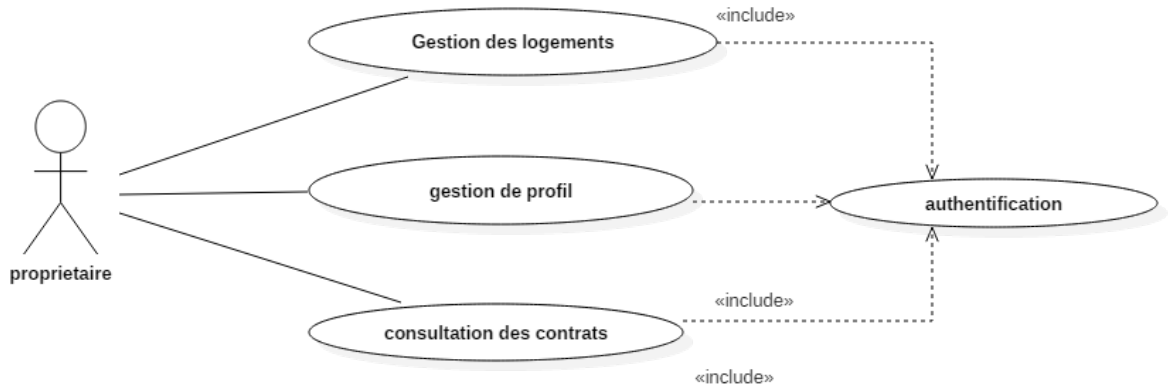

FIGURE 2.3 – Diagramme de cas d'utilisation général de l'**propriétaire**

Ce diagramme représente le diagramme de cas d'utilisation de le**locataire** ;

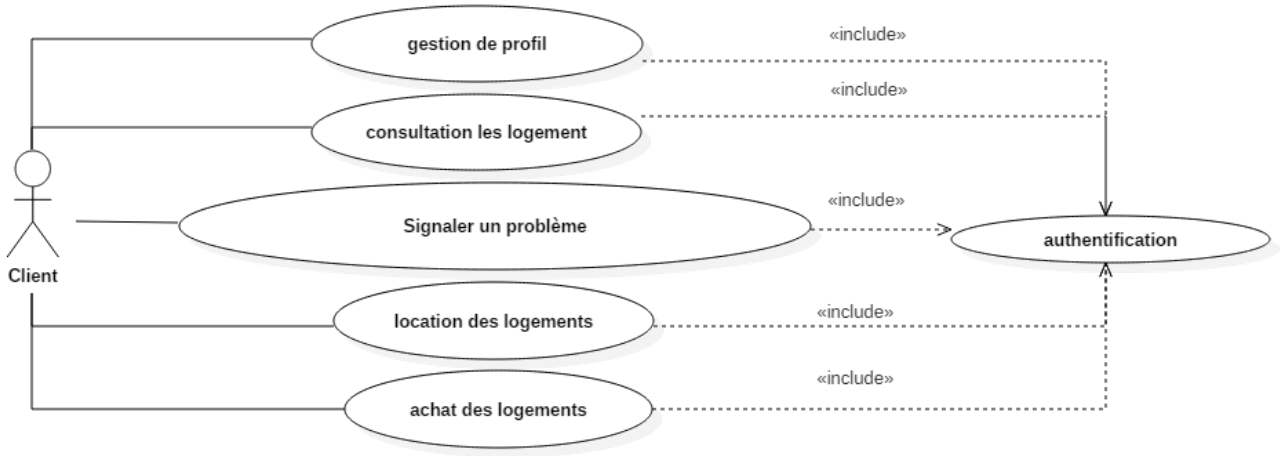

FIGURE 2.4 – Diagramme de cas d'utilisation général de le **locataire**

2.4.3 La description détaillée du cas d'utilisation général :

Dans cette partie, nous allons détailler tous les cas d'utilisation en présentant une description pour chaque cas, et ce, afin de mettre en évidence les interactions entre les acteurs et le système.

1. Description textuelle du cas « s'authentifier » :

Cas d'utilisation	**S'authentifier**
Acteur(s)	Utilisateur ou Administrateur
Résumé	L'acteur introduit un login et mot de passe pour accéder au système.
Pré condition	L'acteur doit le type de connexion.
Scénario	l'acteur saisie le type de connexion et le système les vérifie

TABLE 2.1 – Cas d'utilisation "S'authentifier détaillé

2. Description textuelle du cas « gestion de profil » :

Cas d'utilisation	**Gestion de profil**
Acteur(s)	Utilisateur
Résumé	L'acteur peut mettre à jour son profil.
Pré condition	S'authentifier
Scénario	le système met à jour et affiche les nouveaux informations saisies.

TABLE 2.2 – Cas d'utilisation "gestion de profil" détaillé

3. Description textuelle du cas « Consulter une immobilière » :

Cas d'utilisation	**Consulter une immobilière**
Acteur(s)	Utilisateur
Résumé	L'acteur peut consulter les immobilières.
Pré condition	S'authentifier et la disponibilité des immobilières.
Scénario	L'acteur consulte les immobilières et voit leurs informations.

TABLE 2.3 – Cas d'utilisation "Demander un matériel" détaillé

4. Description textuelle du cas « gestion des immobilières » :
   Le mot **gestion** englobe la modification, l'ajout et la suppression.

Cas d'utilisation	**gestion des immobilières**
Acteur(s)	Agence
Résumé	L'agence peut gérer les immobilières.
Pré condition	S'authentifier et l'existence d'immobilière.
Scénario	L'agence consulte les immobilières et aussi choisir l'immobilière à mettre à jour ou a supprimer ou bien ajouter une nouvelle immobilière.

TABLE 2.4 – Cas d'utilisation "gestion des immobilières" détaillé

5. Description textuelle du cas « Gestion des réclamations » :
   Le mot **gestion** englobe la modification, l'ajout et la suppression.

Cas d'utilisation	**Gestion des réclamations/suggestions**
Acteur(s)	Agence
Résumé	L'agence peut gérer la liste des réclamations/suggestions.
Pré condition	S'authentifier et l'existence des réclamations/suggestions
Scénario	L'agence consulte les réclamations/suggestions et aussi choisir la réclamation/suggestion à mettre à jour ou a supprimer

TABLE 2.5 – Cas d'utilisation "Gestion des réclamations/suggestions" détaillé

6. Description textuelle du cas « gestion des employées » :

Cas d'utilisation	**Gestion des employées**
Acteur(s)	Agence
Résumé	L'acteur peut gérer les employées.
Pré condition	S'authentifier et l'existence des employées.
Scénario	L'agence consulte les employées et aussi choisir l'employée à mettre à jour ou a supprimer ou bien ajouter une nouveau employée.

TABLE 2.6 – Cas d'utilisation "Gestion des employées" détaillé

7. Description textuelle du cas « Élaborer un contrat » :

Cas d'utilisation	**Élaborer un contrat**
Acteur(s)	Agence
Résumé	L'agence assure l'établissement d'un contrat entre le client et le propriétaire.
Pré condition	S'authentifier
Post-condition	Le propriétaire et le client concerné auront une mise à jour.
Scénario	le système associe pour chaque immobilière un client et son propriétaire .

TABLE 2.7 – Cas d'utilisation "Élaborer un contrat" détaillé

8. Description textuelle du cas « Consulter les statistiques » :

Cas d'utilisation	Consulter les statistiques
Acteur(s)	Agence
Résumé	L'agence peut consulter ces propres chiffres d'affaires.
Pré condition	S'authentifier
Scénario	le système affiche les chiffres d'affaires.

TABLE 2.8 – Cas d'utilisation "Consulter les statistiques" détaillé

2.5 Diagramme de classe :

2.5.1 Définition :

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.
Il représente les classes intervenants dans le système. Le diagramme de classe est une représentation statique des éléments qui composent un système et de leurs relations.

2.5.2 Le diagramme de classe général :

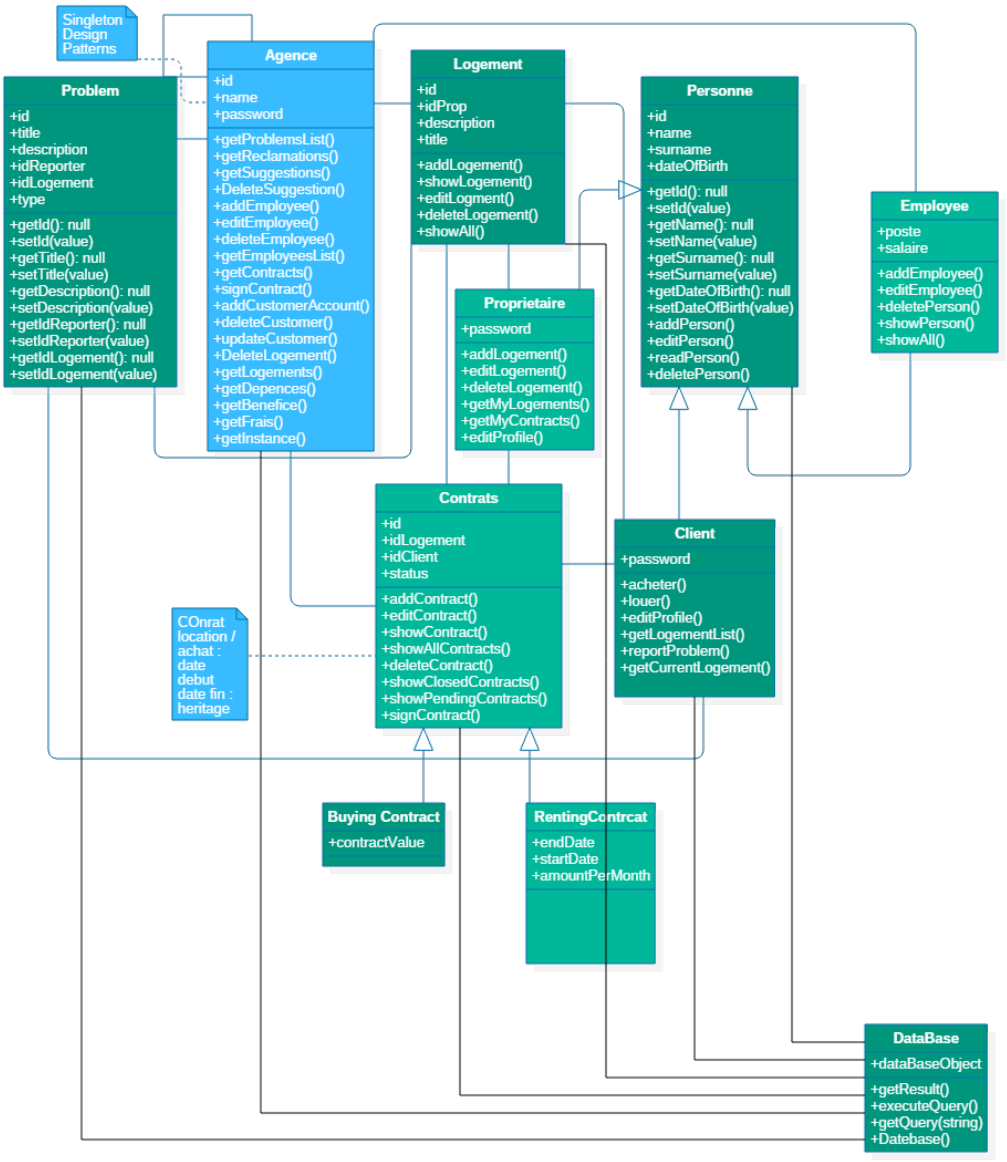

FIGURE 2.5 – Diagramme de classe général

2.6 Conclusion

Dans ce chapitre, nous avons présenté une conception détaillée du travail à partir de la modélisation graphique qui nous a permit de raisonner sur des solutions à partir de modèle organisé autours d'idées qui se rapporte à notre spécification des besoins. Le modèle nous a facilité la compréhension du problème et la modélisation nous a permit de mieux appréhender les besoins.

Chapitre 3

Réalisation

3.1 Introduction

Dans ce chapitre, nous allons décrire l'environnement de travail logiciel et matériel permettant la réalisation de notre application. Par la suite, nous présenterons le travail réalisé et les résultats obtenus.

3.2 Environnement de travail :

Nous présentons dans cette partie l'environnement matériel et logiciel utilisé dans le développement du notre application web.

3.2.1 Environnement matériel :

— Lenovo ideapad Y510P

FIGURE 3.1 – Logo Lenovo

1. **Processeur :** Intel Inside Core i7.
2. **RAM :** 8.00 Go .
3. **Système d'exploitation** Windows 10 64bits

3.2.2 Environnement logiciel

Nous avons utilisé les logiciels suivants :

— **XAMPP :** est un ensemble des logiciels qui permet de mettre en place un serveur web local, serveur FTP, etc.

FIGURE 3.2 – Logo XAMPP

— **phpMyAdmin :** est une application web de gestion graphique pour les systèmes de gestion des bases des données. Il s'agit de l'une des plus célèbres interfaces pour gérer une base de données MySQL.

FIGURE 3.3 – Logo phpMyAdmin

— **Code : :Blocks :** est un IDE gratuit C, C ++ et Fortran construit pour répondre aux besoins les plus exigeants de ses utilisateurs. Il est conçu pour être très extensible et entièrement configurable.
Un IDE avec toutes les fonctionnalités dont vous avez besoin, ayant un aspect cohérent, une sensation et un fonctionnement sur toutes les plates-formes.

FIGURE 3.4 – Logo Code : :Blocks

— **GitHub :** GitHub (exploité sous le nom de GitHub, Inc.) est un service web d'hébergement et de gestion de développement de logiciels, utilisant le logiciel de gestion de versions Git.
Le site assure également un contrôle d'accès et des fonctionnalités destinées à la collaboration comme le suivi des bugs, les demandes de fonctionnalités, la gestion de tâches et un wiki pour chaque projet.

FIGURE 3.5 – Logo GitHub

3.3 Modèles réalisés :

3.4 Menu principal :

Le menu principal permet de choisir le type de connexion.

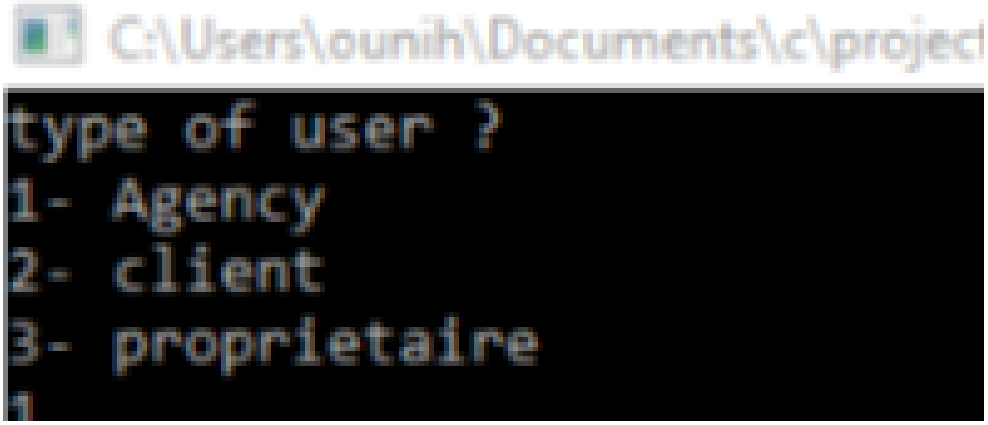

FIGURE 3.6 – Menu principal

3.5 Menu de l'agence :

Ce menu permet à l'agence de naviguer dans le système.

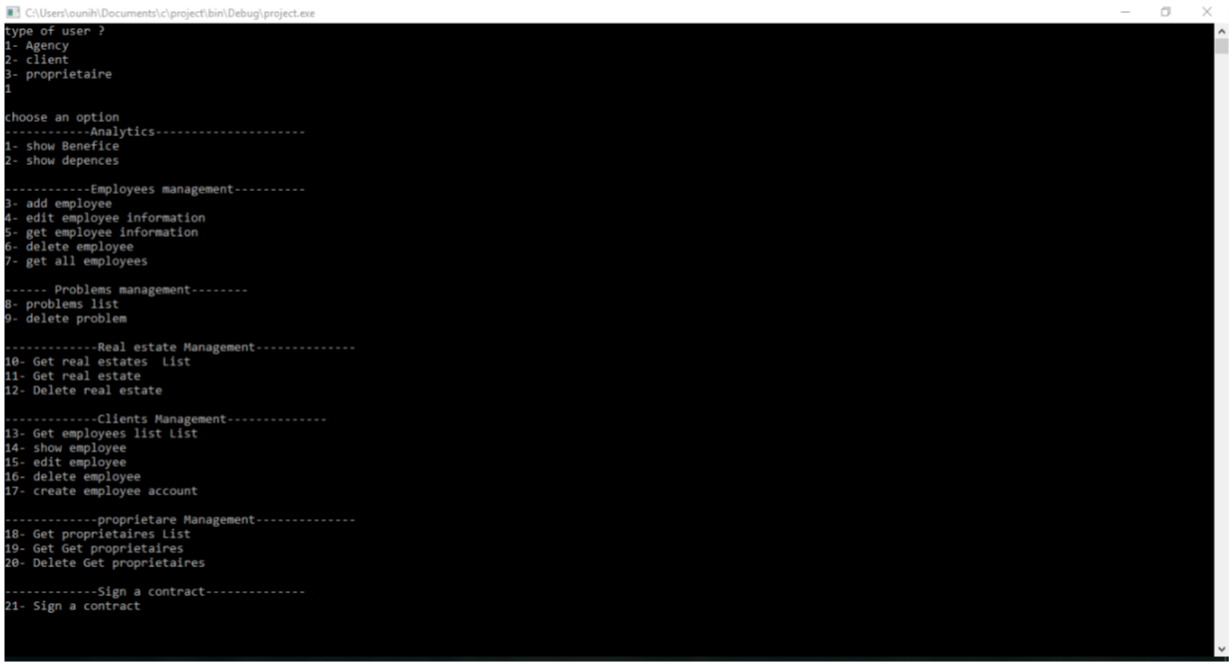

FIGURE 3.7 – Menu de l'agence

3.6 Menu du propriétaire :

Ce menu permet à le propriétaire choisir la fonction à exécuter.

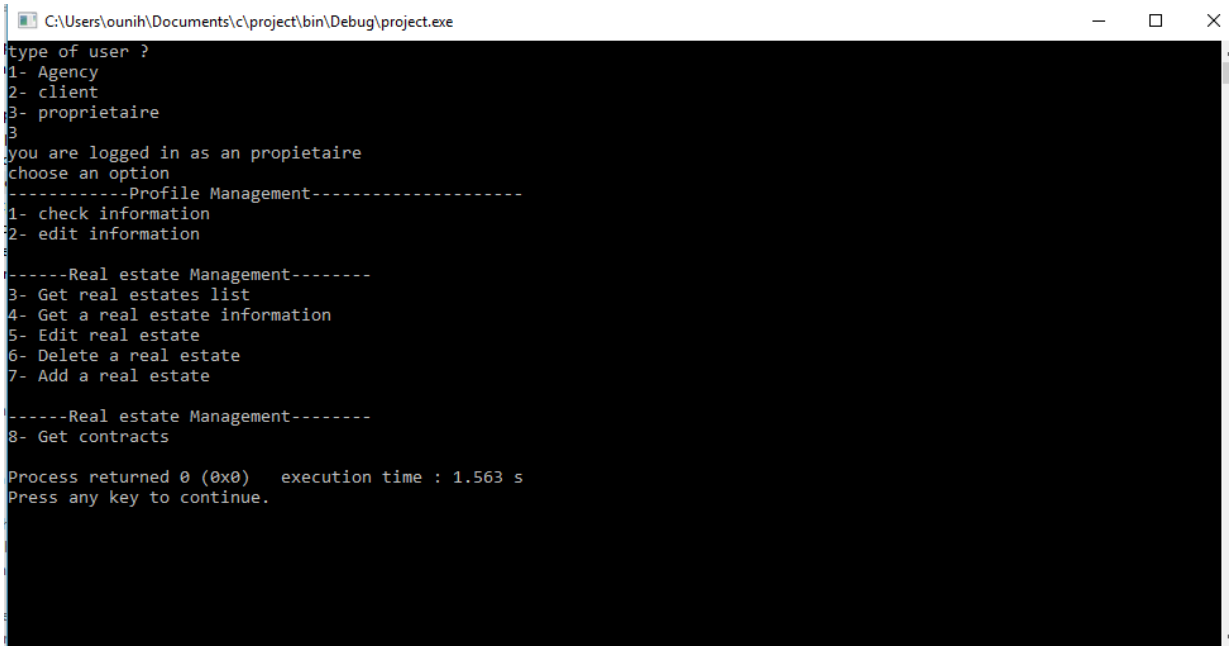

FIGURE 3.8 – Menu du propriétaire

3.7 Menu du client :

Ce menu permet à le client choisir la fonction à exécuter.

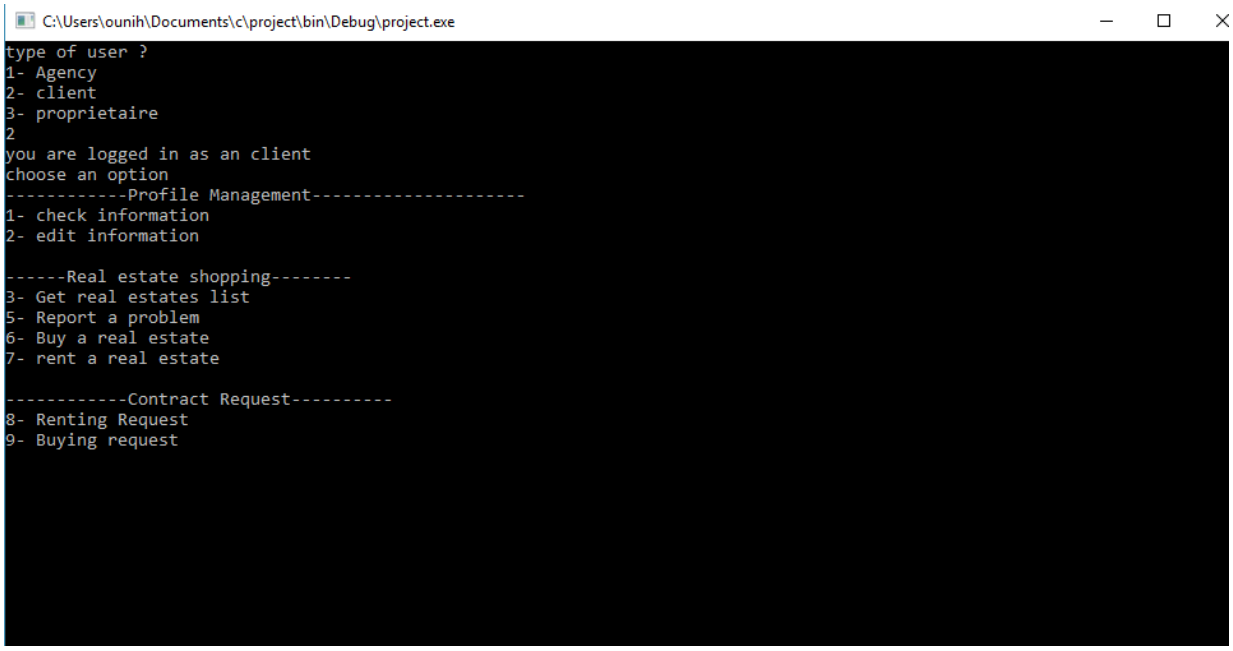

FIGURE 3.9 – Menu du client

3.8 Conclusion :

Dans ce dernier chapitre, nous avons traité les détails de la réalisation de notre application C++. De plus, nous avons aussi montré de différentes interfaces.

Conclusion et perspectives

Lors le développement de ce projet, nous avons pu mettre en pratique nos connaissances théoriques acquises durant notre formation en programmation orientée objet, de plus, nous nous sommes confronté aux difficultés réelles du monde du travail et du management d'équipes.
De ma point de vue, ce projet peut être améliorer en développant des interfaces graphiques.