

USTHB

30-04-2024/16-05-2024

UNIVERSITE DE SCIENCE ET DE TECHNOLOGIE HOUARI BOUMEDIENNE « USTHB »

Faculté d’Informatique

Spécialité : Informatique Académique

Réalisé par :

BEDDIAR Rawa Tesnim Eldjenna - Team leader

BKHITI Oum El Kheir - Conceptrice

GHARBI Inès - Développeuse

GHARBI Sara - Rédactrice

RAPPORT DU PROJET :

Agence Immobilière

**Résumé du rapport :**

Ce rapport se concentre sur la conception et la création d'une application desktop dédié à la gestion d’une agence immobilière. L'objectif principal de ce projet est d'améliorer la gestion interne de l’agence en fournissant un outil analytique proposant ainsi des fonctionnalités permettant de suivre, analyser et gérer efficacement les clients, les biens, les agents et d'autres aspects importants de l'activité immobilière. Grâce à une interface conviviale, le personnel pourra gérer les informations, accéder aux fonctionnalités nécessaires au besoin.

**Sommaire :**

Introduction au Projet : …………………………………………………………………………4

Analyse des Besoins : …………………………………………………………………………..5

Architecture de l'Application : …………………………………………………………………6

Technologies Utilisées : ………………………………………………………………………..10

Fonctionnalités Implémentées :…………………………………………………………………11

Difficultés Rencontrées : …………………………………………………………………….…13

Conclusion et Perspectives : ……………………………………………………………………14

**Introduction au projet :**

Notre projet consiste à développer une application informatique en langage Java principalement pour aider une agence immobilière à améliorer sa gestion quotidienne. Cette agence s'occupe de la vente et de la location de maisons, villa, Hangar et d'appartements. Nous avons observé que l'agence rencontre des difficultés pour suivre ses clients, gérer ses biens disponibles, rendez-vous et organiser le travail de ses agents...

L'objectif principal de notre projet est de créer une application conviviale qui permettra à l'agence de suivre et de gérer efficacement les clients, les biens immobiliers, les transactions et le travail de ses agents. Nous cherchons à rendre le travail de l'agence plus efficace et à améliorer la qualité du service fourni aux clients.

Nous avons opté pour une application de bureau afin de proposer une solution adaptée spécifiquement aux besoins de notre agence. Au cours du développement, nous avons rencontré des défis, notamment pour gérer les informations dans la base de données et concevoir des interfaces utilisateur, ce sera notre sujet de discussion plus tard.

ANALYSE DES BESOINS :

**Gestion des Rendez-vous**

On a besoin d'un système pour planifier et gérer les rendez-vous avec les clients, les visites de biens et les réunions internes. L'application doit inclure des fonctionnalités de calendrier et de notifications pour garantir une gestion efficace des rendez-vous.

**Gestion des Transactions**

On a besoin d'un système complet pour enregistrer et suivre les transactions immobilières, y compris les ventes et les locations. L'application doit permettre de créer des contrats, d'enregistrer les paiements et de générer des rapports sur les transactions effectuées.

**Gestion des Clients**

On a besoin de suivre et de gérer les informations sur les clients, y compris leurs coordonnées, historique des interactions, préférences et demandes spécifiques. L'application doit permettre d'ajouter de nouveaux clients, de mettre à jour leurs informations et de suivre leurs activités avec l'agence.

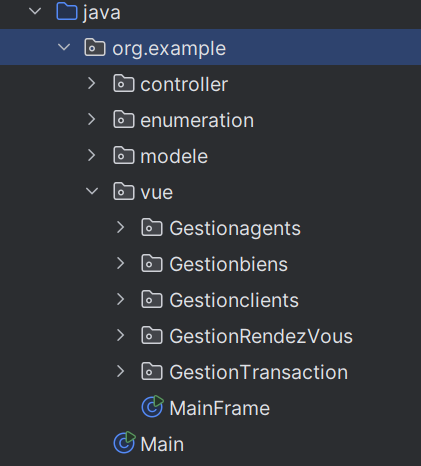
**Gestion des Biens Immobiliers**

On a besoin d’organiser et gérer efficacement les informations sur les biens disponibles à la vente ou à la location. Cela inclut des fonctionnalités pour ajouter de nouveaux biens, mettre à jour les détails existants et consulter les caractéristiques importantes comme le prix, la localisation et les spécifications techniques.

**ARCHITECTURE DE L’APPLICATION :**

L'architecture de l'application suit le modèle MVC (Modèle-Vue-Contrôleur) pour séparer les différentes responsabilités et améliorer la maintenabilité du code.

* **Modèle (Model)** : Contient les classes représentant les données de l'application, telles que **Client**, **Agent**, **Propriété**, etc. Chacune contient les constructeurs, getters et setters.
* **Vue (View)** : Gère l'affichage des données à l'utilisateur. Dans notre cas, il s'agit de l'interface graphique dans la version avec interface.
* **Contrôleur (Controller)** : Gère la logique métier et les interactions entre le modèle et la vue.

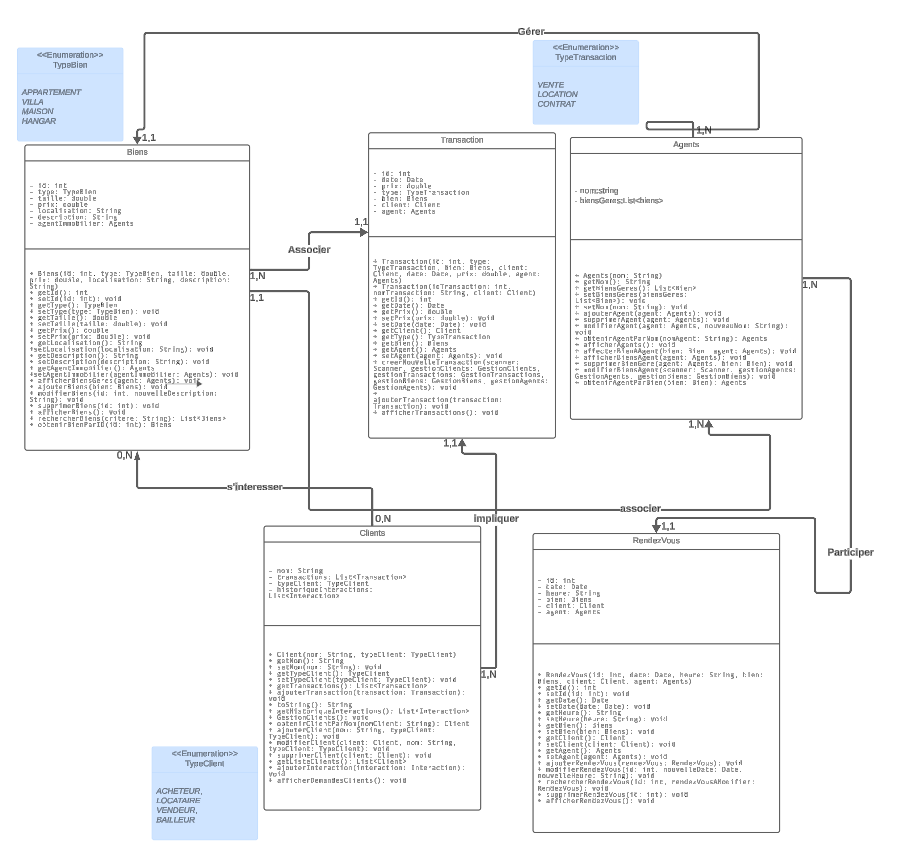
Par contre nous avons deux classes principales pour l'application :

1. **Sans interface graphique** :

Permet de gérer les opérations via la console.

1. **Avec interface graphique** :

Fournit une interface utilisateur plus conviviale.

1. Modélisation des données :

Ci-dessous, nous présentons le diagramme de classe de notre système, qui est une représentation visuelle de la structure et des relations entre les différentes classes du système.

1. Le modèle relationnel définit une structure de données organisée en tables (ou relations), où chaque table représente une entité (comme **agents**, **biens**, **clients**, etc.). Dans le processus de transformation du diagramme de classe en modèle relationnel, chaque classe est convertie en table ou relation. Les attributs de la classe deviennent les attributs de la table, tandis que l'identifiant de l'entité devient la clé primaire de la table.

-Agents (nom) ;

-agent\_biens (id, id\_bien\* , nom\_agent\*);

-biens (id , type, taille, prix, localisation, description, agentImmobilier\*) ;

-clients (nom , TypeClient ) ;

-demandes\_client (id , client\* , typeBienRecherche , budgetMax , description ) ;

-interactions (id , datetime , idClient, description) ;

-rendez\_vous (id , agent\* , bien\* , client\* , heure , date ) ;

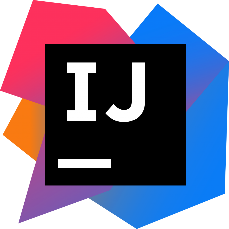
-temp\_biens (id , type , taille , prix , localisation , description ) ;

-transactions ( id , bien\* , client\* , agent\* , date , prix , type ) ;

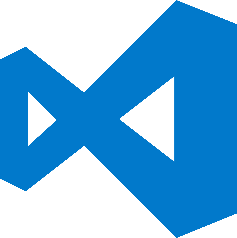
Les clés primaires sont soulignées et les clés étrangères sont désignées par le signe \*

**Technologies Utilisées :**

* **IntelliJ IDEA** : Environnement de développement intégré (IDE) utilisé pour coder l'application en Java.
* **XAMPP** : Utilisé pour le serveur local afin de gérer la base de données.



* **Visual Studio Code** : Utilisé pour écrire les scripts SQL.
* **phpMyAdmin :** Interface web utilisée pour administrer la base de données MySQL.



**Fonctionnalités Implémentées :**

L'application inclut diverses fonctionnalités pour la gestion d'une agence immobilière :

Déjà on résume la classe d’application « main » comme ceci :

Main Loop:

Lire choix\_rendezvous

switch (choix\_rendezvous) {

Idem pour les rendez-vous

default: Choix invalide

}

} while (choix\_rendezvous != 0)

break

case 0: Quitter l'application

default: Choix invalide

}

} while (choix\_principale != 0)

case 3: // Gestion des clients

do {

Afficher le menu des clients

Lire choix\_trois

switch (choix\_trois) {

Idem pour les clients

default: Choix invalide

}

} while (choix\_trois != 0)

break

case 4: // Gestion des agents

do {

Afficher le menu des agents

Lire choix\_deux

switch (choix\_deux) {

Idem pour les agents

default: Choix invalide

}

} while (choix\_deux != 0)

break

case 5: // Gestion des rendez-vous

do {

Afficher le menu des rendez-vous

Lire choix\_rendezvous

switch (choix\_rendezvous) {

Idem pour les rendez-vous

default: Choix invalide

}

} while (choix\_rendezvous != 0)

break

case 0: Quitter l'application

default: Choix invalide

}

} while (choix\_principale != 0)

do {

Afficher le menu principal

Lire choix\_principale

switch (choix\_principale) {

case 1: // Gestion des biens immobiliers

do {

Afficher le menu des biens immobiliers

Lire choix

switch (choix) {

Ici on trouvera differents cas pour traiter les biens

default: Choix invalide

}

} while (choix != 0)

break

case 2: // Gestion des transactions

do {

Afficher le menu des transactions

Lire choixTransaction

switch (choixTransaction) {

Idem pour les transactions

default: Choix invalide

}

} while (choixTransaction != 0)

Break

On a aussi le package d’énumération qui est utilisée pour définir les différents types dans l’agence immobilière que l'application peut gérer. Les valeurs possibles sont :

**TypeTransaction :** *VENTE*,  
*LOCATION*,  
*CONTRAT*

**TypeClient :**

*ACHETEUR*, *LOCATAIRE*, *VENDEUR*, *BAILLEUR*

**TypeBien :**

*APPARTEMENT*, *VILLA*, *MAISON*, *HANGAR*

Si on prend la classe « gestionAgents » dans le package « controller » on trouvera principalement des structures de données comme les listes et les maps. La liste **agentsList** stocke les agents immobiliers pour les opérations d'ajout, de suppression et de recherche, tandis que la map **biensParAgent** associe chaque agent à une liste de biens immobiliers qu'il gère, facilitant ainsi l'affectation, l'affichage et la modification des biens.

Plusieurs fonctions de la classe intègrent des commandes SQL pour interagir avec la base de données. Par exemple, la fonction **ajouterAgent** ajoute un agent à **agentsList** et à la base de données via une commande **INSERT INTO**. De même, **supprimerAgent** enlève un agent de **agentsList**, de **biensParAgent**, et de la base de données avec une commande **DELETE FROM**. La fonction **modifierAgent** permet de modifier le nom d'un agent dans **agentsList**, tandis que **obtenirAgentParNom** recherche et renvoie un agent par son nom dans **agentsList**.

Pour gérer les biens, la fonction **affecterBienAAgent** ajoute un bien à la liste des biens gérés par un agent dans **biensParAgent**, et **affecterBienAgent** ajoute cette association dans la base de données via une commande **INSERT INTO**.

Ces principes de gestion des agents, incluant les opérations CRUD et l'utilisation des listes et maps, s'appliquent de manière similaire aux autres classes de gestion, telles que la gestion des biens et des clients... en adaptant les structures de données et les requêtes SQL appropriées.

**Difficultés Rencontrées :**

***Intégration de la base de données* :** Configurer et intégrer MySQL avec Java a nécessité beaucoup de temps et de recherche.

***Manipulation de GitHub***: La gestion du code source et de la collaboration via GitHub était et reste un challenge surtout qu’on pas une expérience préalable avec cet outil.

***Interface graphique :*** Aussi la création des interfaces graphiques utilisateur intuitive et fonctionnelle a été un défi majeur, surtout au début avec le manque d’expérience.

**Conclusion et Perspectives :**

En conclusion, le développement de ce projet a été une expérience enrichissante malgré les défis rencontrés et ceci est grâce à l'aide précieuse de étudiantes plus âgées qu’on connait, notamment vous Monsieur pour votre projet car chaque étape a été une opportunité d'apprentissage pour nous.