République Algérienne Démocratique et Populaire

Ministère de l’Enseignement Supérieur et de la Recherche Scientifique

Université des Sciences et de la Technologie Houari Boumediene



Rapport de projet de TP – Programmation orientée objet en JAVA

|  |
| --- |
| ***Application de gestion d’agence immobilière*** |

Réalisé par : Responsable du TP : M. Abdellahoum Hamza

* Chabri Abdelmalek
* Hosseinat Kaoutar
* Ghazli Yasmine
* Arabi Ghalia Bahria
* Bounekta Adnane

Année : 2023/2024

**Introduction :**

Ce projet a pour objectif de concevoir et développer une application de gestion d’agence immobilière. Cette application vise l’automatisation et le rehaussement de l’efficacité des opérations relatives à la gestion des biens immobiliers, des clients ainsi que des transactions immobilières.

Dans cette optique, une infrastructure de base de données a été développée en plus des classes dédiées pour encadrer les activités essentielles. En parallèle, une interface utilisateur a été conçue afin de faciliter la navigation à travers les fonctionnalités.

1. **Base de données :**

La base de données a été créé à l’aide d’**Oracle Database** et le langage **SQL (Structured Query Language)**, tel qu’après la création et la sauvegarde différentielle de cette dernière, on a créé différentes tables pour stocker toutes les informations essentielles sur :

* Agent\_immobilier (ID Agent, nom, prénom)
* Bien\_immobilier (ID bien, type, size, prix, localisation, description, ID\_agent responsable)
* Client (ID client, nom, prénom, type (acheteur ou vendeur), demande, numéro de transaction)
* Transaction (Numéro de transaction, type (vente ou location), paiement, date, ID client, ID\_bien)
* Rendez-vous (ID rendez-vous, date, ID client)

1. **Classe DataBaseManager :**

Cette classe consolide la majeure partie des actiens de l’application dans différentes méthodes.

* Au début, on a la méthode **getConnection ()** qui assure l’établissement de la connexion à la base de données créé précédemment ; et ceci se fait après avoir vérifié si une connexion est déjà active ou fermé. Ensuite, elle tente d’établir une connexion en utilisant les informations spécifiques telles que l’URL de la base, le nom d’utilisateur et le mot de passe.
* D’autre part, la méthode **closeConnection ()** est responsable de la fermeture de la connexion à la base de données lorsqu’elle n’est plus nécessaire.

Après avoir établi la connexion à la base de données, plusieurs méthodes sont mises en place pour exécuter diverses taches au sein de l’application, parmi ces méthodes :

* **AjouterBienImm (BienImmobilier bien)** : permet l’ajout d’un nouveau bien à la base de données, en utilisant les informations fournies par l’objet « BienImmobilier » passé en paramètre.
* **SupprimerBienImm (Int id)** : permet la suppression d’un bien immobilier après la vérification de son existence à travers son identifiant.
* **ModifierBienImm (BienImmobilier bien)**: elle utilise les nouvelles données fournies par l’objet « BienImmobilier » pour effectuer une modification des informations d’un bien immobilier déjà existant.
* **Exist (int id)** : permet de vérifier si un bien immobilier avec l’identifiant spécifié existe dans la base de données. Elle renvoie **TRUE** si le bien existe et **False** sinon.
* **RechercherBiens (String type, int prix, string localisation)** : permet de rechercher des biens immobiliers dans la base de données en spécifiant certains critères.
* **AjouterClient (Client client)**.
* **ModifierClient (Client client)**.
* **SupprimerClient (Int idClient)**.
* **RecupererClient (Int idClient)** : recherche un client spécifique dans la base de données en fonction de son identifiant. Elle exécute une requête **SELECT** pour récupérer les informations sur le client correspondant à l’ID fourni, puis retourne un objet Client avec les informations récupérées.
* **EnregistrerPaiement (Transactions transaction)** : permet l’enregistrement d’un paiement dans la base de données, en utilisant les informations fournies par l’objet Transactions.
* **PlanifierRendezVous (Client client, int IdAgent, date dateRendezVous)** : permet la planification d’un rendez-vous pour un client avec un agent immobilier à une date spécifiée. Elle enregistre ensuite le rendez-vous dans la base de données et affiche un message de confirmation.
* **EnregistrerRendezVous (Date dateRdv, int IdClient)**: assure l’ajout d’un rendez-vous dans la base de données. Elle établit une connexion, prépare une requête SQL pour insérer les données du rendez-vous.

1. **Classes BienImmobilier, AgentImmobilier, Client, GestionRendezVous, Contrat et Transactions :**

Ces classes ont été conçues pour représenter les principaux acteurs et processus impliqués dans les opérations quotidiennes d’une agence immobilière. Chacune de ces classes contient les attributs essentiels pour décrire et caractériser un acteur (Agent ou client) ou un processus (Rendez-vous ou transactions ou contrat) de manière complète.

Elles fournissent également toutes les **getters et setters** qui permettent de récupérer les valeurs des attributs, les définir ou les mettre à jour.

1. **Interface graphique :**

L’interface graphique a été conçue à l’aide de **IntelliJ IDEA** avec **Java FX** une combinaison qui permet de créer des applications Java dotées d’une interface visuelle interactive. L’intégration de **SceneBuilder** a simplifié la conception visuelle des éléments de l’interface. Dans la fenêtre principale, les boutons ont été disposés comme suit :

Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Description générée automatiquement

On a ajouté un contrôleur (Mouse Controller) en tant que classe Java, responsable de la gestion des actions des boutons. Par exemple, en appuyant sur le bouton « Agent », il redirige l’utilisateur vers une interface affichant la table des agents et les options de modifications.



Dans la classe **MouseController**, nous avons implémenté une méthode pour gérer l’action :

Une image contenant texte, Police, capture d’écran, ligne

Description générée automatiquement

Et dans le Main, on ajoute la méthode qui nous permet de changer la scène vers celle des agents :

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Une fois l’accès à l’une des tables effectuées, nous avons inclus des boutons de retour à la page précédente dans toutes les interfaces :

Une image contenant capture d’écran, texte, diagramme, conception

Description générée automatiquement Une image contenant texte, capture d’écran, affichage, logiciel

Description générée automatiquement

Pour supprimer une ligne de tableau, il suffit d’insérer l’ID correspondant. Par exemple, pour supprimer un client, il suffit d’entrer son identifiant et de cliquer sur « **submit** ».

Une image contenant capture d’écran, texte, diagramme, conception

Description générée automatiquement

Nous avons aussi créé une interface dédiée à l’ajout et la modification de lignes dans une table. Par exemple, pour insérer ou modifier une entrée dans la table des clients, vous devez juste fournir les informations nécessaires pour cette table.

Une image contenant texte, capture d’écran, nombre, conception

Description générée automatiquement