

APPLICATION DE GESTION D'AGENCE IMMOBILIERE

Réalisé par :

- Deghbar Djouhra. 222231368818
- Aouali Melissa Inel. 222231620408
- Louriachi Meriem. 212131059635
- Abbaci Hadjer 212132004487

Responsable du module :

Mme Bouziane.

Enseignants de TP :

Mr H. Abdellahoum.

Mr M.Ait Mehdi.

- Numero de Groupe : 24

I-Introduction :

Le rapport ci-dessous constitue une présentation générale du Projet, intitulé "Application de Gestion d'agence immobilière". Ce projet vise à développer une application en Java pour gérer efficacement les biens immobiliers, les clients et les transactions. Cette dernière a pour objectif d'automatiser les processus de gestion, d'améliorer la productivité et de faciliter le travail de l'agence immobilière. Les composantes principales du rapport incluent :

1-Description du projet : Présentation des objectifs et des fonctionnalités de l'application.

2-Conception technique : Détails sur l'architecture du système et les choix technologiques, notamment l'utilisation de Java et MySQL.

3-Développement : Explication des étapes de développement et des outils utilisés, comme le langage Java pour le back-end et JavaSwing pour l'interface utilisateur.

4-Tests et validation : Stratégies de test et résultats obtenus pour assurer la qualité et la fiabilité de l'application.

5-Conclusion et perspectives : Bilan du projet et perspectives d'amélioration futures.

II -Objectif Du Projet :

-Automatisation des Processus : Réduire les tâches manuelles en automatisant la gestion des biens immobiliers, des clients et des transactions.

-Optimisation de la Productivité : Améliorer l'efficacité des agents immobiliers en centralisant les informations et en facilitant l'accès aux données pertinentes.

-Gestion Complète des Propriétés : Permettre l'enregistrement, la modification et la consultation détaillée des propriétés disponibles à la vente ou à la location.

-Suivi des Transactions : Suivre et enregistrer les transactions immobilières.

-Gestion des Clients : Gérer les informations des clients.

-Interface Conviviale : Concevoir une interface utilisateur intuitive et conviviale pour faciliter l'utilisation de l'application par les agents immobiliers.

III -Description Générale sur L’application « «FULL HOUSE » :

Ce projet consiste en la conception et le développement d'une application de gestion d'agence immobilière. L'objectif principal est de fournir un outil complet et intuitif permettant de gérer efficacement les biens immobiliers, les clients et les transactions. L'application vise à automatiser et à optimiser les processus de gestion pour améliorer la productivité et offrir une meilleure expérience aux utilisateurs.

La gestion d'une agence immobilière implique de nombreuses tâches complexes, telles que l'enregistrement et la mise à jour des propriétés, la gestion des informations clients et le suivi des transactions. Notre application répondra à ces besoins en proposant des fonctionnalités variées et adaptées aux exigences du secteur immobilier.

En utilisant les technologies modernes telles que Java pour le développement backend, MySQL pour la gestion des bases de données et JavaSwing pour l'interface utilisateur, cette application assurera une performance optimale et une grande convivialité. En outre, l'intégration avec des outils comme NetBeans et sceneBuilder facilitera la conception et la personnalisation de l'interface, rendant l'application accessible et agréable à utiliser pour les agents immobiliers.

IV -Description des différentes parties de l’application «FULL HOUSE »:

L'application Full House est développée en utilisant Java et JavaSwing sans oublier MYSQL ,myPhpAdmin et xampp server pour manipuler les tables de notre base de données.

Les différentes classes de l'application contiennent des méthodes et des constructeurs qui font partie à la classe elle même, ainsi que d'autres intégrés à l'interface implémentée avec JavaSwing. De plus, l'utilisation des d'exceptions a joué un rôle très important dans notre projet pour assurer la fiabilité de l'application et guider les utilisateurs lors de leur utilisation avec des message erreur, de sucée... .

1-Classe Home:

Description : La classe Home gère les fonctionnalités de connexion pour les administrateurs et les agents dans l'application, c'est la première interface qui interagit avec l'utilisateur dès qu'on exécute l'application.

Principales Méthodes et Fonctionnalités :

Constructeur Home() : Initialise la classe en appelant initComponents().

initComponents() : Initialise les composants tels que les labels, les champs de texte et les boutons.

userActionPerformed() : Gère l'événement de pression de la touche "Entrée" dans le champ de texte "Username".

2-Classe Admin :

Description Cette classe permet à un administrateur d'accéder aux différentes fonctionnalités de gestion de l'application immobilière, telles que la gestion des acheteurs, vendeurs, agents immobiliers, transactions et la liste des biens immobiliers, l'Admin a l'accès à tout le système de gestion .

3-Classe Agent:

La classe "Agent" permet de gérer les agents dans l'agence immobilière que ce soit leur insertion, suppression, modification ... seul l'admin a accès à cette classe.

1-Principales Méthodes et Fonctionnalités :

Constructeur agent() : Le constructeur initialise la classe et appelle la méthode displayTable() pour afficher les données des agents dans un tableau.

Méthode displayTable() : Cette méthode récupère les données des agents à partir de la base de données MySQL "full-house-java-app" et les affiche dans un tableau. Elle utilise la bibliothèque DbUtils pour convertir les résultats de la requête en un modèle de tableau.

2-Les different Operations effectuees sur les Agents:

Ajouter un agent (jButton1ActionPerformed()) : Cette fonctionnalité permet d'ajouter un nouvel agent à la base de données en utilisant les champs ID, nom, contact, email et montant des transactions (deal).

Supprimer un agent (jButton2ActionPerformed()) : Permet de supprimer un agent de la base de données en fonction de son ID.

Mettre à jour les attribus d'un agent (jButton3ActionPerformed()) : Cette fonctionnalité permet de mettre à jour les détails d'un agent, y compris son nom, contact, email et montant des transactions.

Rechercher un agent (jButton6ActionPerformed()) : Permet de rechercher un agent dans la base de données en fonction de son ID et d'afficher ses détails dans les champs correspondants.

Actualiser (jButton4ActionPerformed()) : Réinitialise tous les champs et actualise le tableau des agents.

3-Connexion à la base de données :

La classe se connecte à la base de données MySQL en utilisant le pilote JDBC "com.mysql.cj.jdbc.Driver" et les informations de connexion ("jdbc:mysql://localhost:3306/full-house-java-app", "root", "").

4-Shema de la Table Agent:

agentID (Clé primaire, identifiant unique de l'agent).

Agent Name (Nom de l'agent).

contact_no (Numéro de contact de l'agent).

email (Adresse email de l'agent).

deals_amount (Nombre des transaction effecutuées par l'agent X).

4-Classe Seller:

1-Principales Méthodes et Fonctionnalités :

Constructeur seller(): Le constructeur initialise la classe et appelle la méthode displayTable() pour afficher les données des vendeurs dans un tableau.

Méthode displayTable(): Cette méthode récupère les données des vendeurs à partir de la base de données MySQL "full-house-java-app" et les affiche dans un tableau en utilisant la bibliothèque DbUtils pour convertir les résultats de la requête en un modèle de tableau.

2- Les différentes Opérations effectuées sur les Vendeurs :

Ajouter un vendeur (jButton1ActionPerformed()): Cette fonctionnalité permet d'ajouter un nouveau vendeur à la base de données en utilisant les champs ID, nom, contact et email.

Supprimer un vendeur (jButton2ActionPerformed()): Permet de supprimer un vendeur de la base de données en fonction de son ID.

Mettre à jour les attributs d'un vendeur (jButton3ActionPerformed()): Cette fonctionnalité permet de mettre à jour les détails d'un vendeur, y compris son nom, contact et email.

Rechercher un vendeur (jButton6ActionPerformed()): Permet de rechercher un vendeur dans la base de données en fonction de son ID et d'afficher ses détails dans les champs correspondants.

Actualiser (jButton4ActionPerformed()): Réinitialise tous les champs et actualise le tableau des vendeurs.

3- Connexion à la base de données :

La classe se connecte à la base de données MySQL en utilisant le pilote JDBC "com.mysql.cj.jdbc.Driver" et les informations de connexion ("jdbc:mysql://localhost:3306/full-house-java-app", "root", "").

4- Schéma de la Table "Seller" :

sellerID (Clé primaire, identifiant unique du vendeur).

S_name (Nom du vendeur).

contact_no (Numéro de contact du vendeur).

email (Adresse email du vendeur).

5-Classe Buyer:

La classe "Buyer" gère les informations des acheteurs dans l'agence immobilière, permettant des opérations telles que l'ajout, la suppression, la modification, etc. Seul l'administrateur a accès à cette classe.

1-Principales Méthodes et Fonctionnalités :

Constructeur buyer() : Initialise la classe et appelle la méthode displayTable() pour afficher les données des acheteurs dans un tableau.

Méthode displayTable() : Récupère les données des acheteurs depuis la base de données MySQL "full-house-java-app" et les affiche dans un tableau. Utilise la bibliothèque DbUtils pour convertir les résultats de la requête en un modèle de tableau.

2-Opérations sur les Acheteurs :

Ajouter un acheteur (jButton1ActionPerformed()) : Permet d'ajouter un nouvel acheteur à la base de données en utilisant les champs ID, nom, contact, et email.

Supprimer un acheteur (jButton2ActionPerformed()) : Supprime un acheteur de la base de données en fonction de son ID.

Mettre à jour les informations d'un acheteur (jButton3ActionPerformed()) : Permet de modifier les détails d'un acheteur tels que son nom, contact, et email.

Rechercher un acheteur (jButton6ActionPerformed()) : Permet de rechercher un acheteur dans la base de données en utilisant son ID et affiche ses détails.

Actualiser (jButton4ActionPerformed()) : Réinitialise tous les champs et actualise le tableau des acheteurs.

3-Connexion à la base de données :

La classe se connecte à la base de données MySQL en utilisant le pilote JDBC "com.mysql.cj.jdbc.Driver" et les informations de connexion ("jdbc:mysql://localhost:3306/full-house-java-app", "root", "").

4-Schéma de la Table Buyer :

buyerID (Clé primaire, identifiant unique de l'acheteur).

B_name (Nom de l'acheteur).

contact_no (Numéro de contact de l'acheteur).

email (Adresse email de l'acheteur).

6-Classe Transactions:

La classe "Transaction" permet de gérer les transactions dans l'agence immobilière, que ce soit leur insertion, suppression, modification...

1-Principales Méthodes et Fonctionnalités :

Constructeur transaction() : Le constructeur initialise la classe et appelle la méthode displayTable() pour afficher les données des transactions dans un tableau.

Méthode displayTable() : Cette méthode récupère les données des transactions à partir de la base de données MySQL "full-house-java-app" et les affiche dans un tableau. Elle utilise la bibliothèque DbUtils pour convertir les résultats de la requête en un modèle de tableau.

2-Les différentes opérations effectuées sur les transactions :

Ajouter une transaction (jButton1ActionPerformed()) : Cette fonctionnalité permet d'ajouter une nouvelle transaction à la base de données en utilisant les champs ID, ID de l'agent, ID du bien, date de vente/location, prix et ID de l'acheteur/locataire.

Supprimer une transaction (jButton2ActionPerformed()) : Permet de supprimer une transaction de la base de données en fonction de son ID.

Mettre à jour les attributs d'une transaction (jButton3ActionPerformed()):

Cette fonctionnalité permet de mettre à jour les détails d'une transaction, y compris l'ID de l'agent, ID du bien, date de vente/location, prix et ID de l'acheteur/locataire.

Rechercher une transaction (jButton6ActionPerformed()) : Permet de rechercher une transaction dans la base de données en fonction de son ID et d'afficher ses détails dans les champs correspondants.

Actualiser (jButton4ActionPerformed()) : Réinitialise tous les champs et actualise le tableau des transactions.

3-Connexion à la base de données :

La classe se connecte à la base de données MySQL en utilisant le pilote JDBC "com.mysql.cj.jdbc.Driver" et les informations de connexion ("jdbc:mysql://localhost:3306/full-house-java-app", "root", "").

4-Schéma de la Table Transaction :

transactionID (Clé primaire, identifiant unique de la transaction) .

agentID (Identifiant de l'agent qui est une clé étrangère de la table Agent) .

propertyID (Identifiant du bien qui est une clé étrangère de la table Property) .

sell_rent_date (Date de vente/location) .

actual_price (Prix de vente/location) .

buyerID/tenantID (Identifiant de l'acheteur/locataire qui est une clé étrangère de la table buyer).

7-Classe PropertyList:

Cette classe permet d'interagir avec une base de données pour ajouter, supprimer, mettre à jour et afficher des propriétés immobilières.

1-Principales Méthodes et Fonctionnalités :

Constructeur PropertyList() : Initialise la fenêtre de la liste des propriétés et appelle la méthode displayTable() pour afficher les données des propriétés dans un tableau.

Méthode displayTable() : Récupère les données des propriétés à partir de la base de données MySQL "full-house-java-app" et les affiche dans un tableau. Utilise la bibliothèque DbUtils pour convertir les résultats de la requête en un modèle de tableau.

2-Les different Operations effectuees sur les Propriétés:

Ajouter une propriété (jButton1ActionPerformed()) : Ajoute une nouvelle propriété à la base de données en utilisant les champs PropertyID, Cost, Area, BHK, Category, Street, City_State, Pincode, AgentID et SellerID.

Supprimer une propriété (jButton2ActionPerformed()) : Supprime une propriété de la base de données en fonction de son PropertyID.

Mettre à jour les attributs d'une propriété (jButton3ActionPerformed()) : Met à jour les détails d'une propriété, y compris son Cost, Area, BHK, Category, Street, City_State, Pincode, AgentID et SellerID.

Rechercher une propriété (jButton6ActionPerformed()) : Recherche une propriété dans la base de données en fonction de son PropertyID et affiche ses détails dans les champs correspondants.

Actualiser (jButton4ActionPerformed()) : Réinitialise tous les champs et actualise le tableau des propriétés.

3-Connexion à la base de données :

La classe se connecte à la base de données MySQL en utilisant le pilote JDBC

"com.mysql.cj.jdbc.Driver" et les informations de connexion ("jdbc:mysql://localhost:3306/full-house-java-app", "root", "").

4-Shema de la Table "Property":

PropertyID (Clé primaire, identifiant unique de la propriété)

Cost (Coût de la propriété)

Area (Surface de la propriété)

BHK (Nombre de chambres à coucher)

Category (Catégorie de la propriété : Résidentielle/Commerciale/Indisctuelle/Terrain)

Street (Rue où se trouve la propriété)

City_State (Ville et état où se trouve la propriété/ville-wilaya)

Pincode (Code postal de la propriété)

AgentID (Identifiant de l'agent responsable de la propriété qui est une clé étrangère de la table Agent)

SellerID (Identifiant du vendeur de la propriété qui es tune cle etrangere de la table Seller)

8-Classe Contact:

Description : Cette classe fournit un moyen d'accéder aux informations de contact de l'entreprise tout comme le numero de telephone , localization, boite electronique.

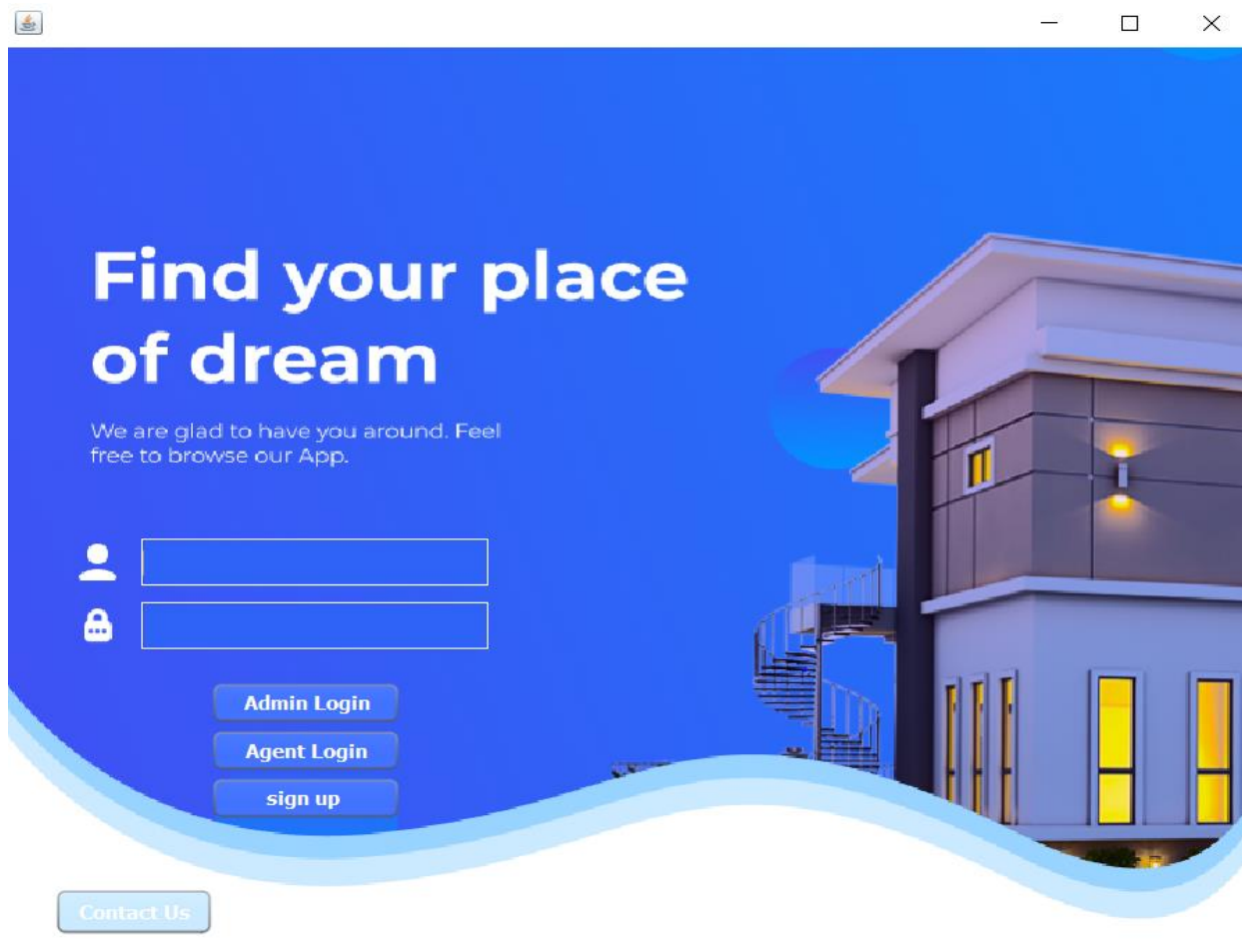
Principales Méthodes:

Constructeur Contact() : Initialise les composants tels que les labels pour l'adresse, le numéro de contact et l'e-mail.

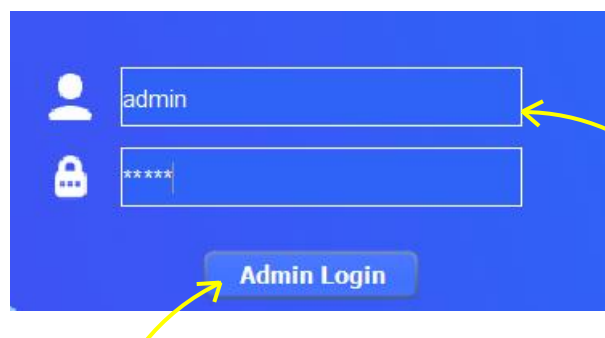
initComponents() : Initialise tous les composants en définissant leurs propriétés.

V- Captures et Guide pour utiliser “Full House real estate management System” App:

1- Menu d'accueil:



2- Se connecter en tant que Administrateur:



Se Connecter en tant que Admin

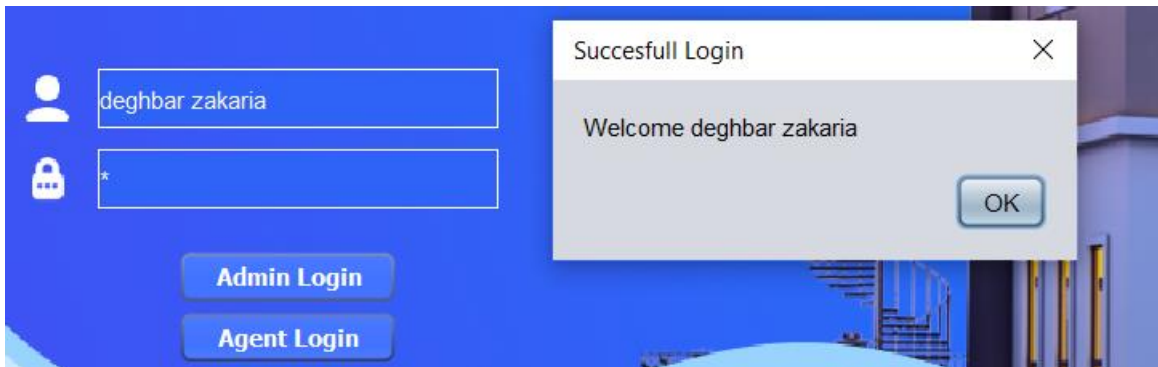
Username: Admin

Password: Admin

3- Le menu de l'Administrateur:



4- Se connecter en tant qu'Agent:




5- Le menu de l'Agent:



6- Inscription d'un Nouvel Agent (cas d'oublier le mot de passe) :

A screenshot of a web application window titled "AGENT SIGNUP". The window has a blue gradient background. At the top, there is a title bar with standard window controls (minimize, maximize, close). Below the title, the main heading "AGENT SIGNUP" is displayed in white. The form consists of five input fields on the left, each with a label: "Username/Agent name", "UserID/password", "Contact", "Email", and "Deals Amount". To the right of these fields is a large white icon of a person with a plus sign. At the bottom, there are three buttons: "Clear", "ADD", and "Close".

7-Gestion des Agents (seul l'Admin a accès) :



Agent Details

agentID	Agent_Name	contact_no	email	deals_amount
1	Deghbar Djouhara	555529223	djouharadgb@gmail.com	1
2	Deghbar Hiba	555523366	hibadgb@gmail.com	4
3	Louriachi Meriem	777725588	louriachimeriem@gmail.co	5
4	Deghbar Selma	612457893	selmadeghbar@gmail.com	7
5	degbar zakaria	792551145	zakdegbar@gmail.com	5
6	Deghbar Farida	777859932	fardgb@gmail.com	5
7	Ait mokhtar ouafa	756884412	aitouafa@gmail.com	5
8	Louriachi Hiba	7775699	hlour@gmail.com	1

ID

Name

Contact

Email

Deals Amount

ADD

DELETE


Search

Update

Refresh

Back

8-Gestion des vendeurs/ locataires :



Seller Details

sellerID	S_name	contact	email
2	Bouguerra souad	555529555	soubg@gmail.com
3	Deghbar Ahcene	12345674	ahcenedgb@gmail.com
4	abacci Hadjer	555521144	abhadjer@gmail.com
5	Melissa aouali	598774423	aouimelissa@gmail.com

ID

Name

Contact

Email

ADD

DELETE


Update

Search

Refresh

Back

9-Gestion des Acheteurs :



Buyer Details

buyerID	B_name	contact	email
1	Louriachi Amina	777758841	louramina@gmail.com
2	Deghbar Rihana	555521144	rihdbg@gmail.com
3	Deghbar Asma	777758841	asma@gmail.com

ID

Name

Contact

Email

ADD

DELETE


Search

Update

Refresh

Back

10-Gestion des Propriétés :



Property List

propertyID	cost	area	bhk	category	street	city_state	pincode	agentID	sellerID
1	45000000.00	2000	8	residential	177 rue Ami...	Alger-Algerie	1123	5	2

PropertyID

Cost

area

BHK

category

Street

city&state

pincode

agentID

sellerID

ADD

DELETE


Search

Update

Refresh

Back

11-Gestion des Transactions :



Transaction ID
propertyID
Buyer ID
Sell Rent Date (yyyy-mm-dd)
Actual price
Agent ID

Transactions Details

transactionID	propertyID	buyerID	sell_rent_date	actual_price	agentID
1	1	1	2024-05-06	4500000.00	5

ADD

Search

Update

DELETE

Refresh

Back

12-Comment contacter notre Agence FULL HOUSE:

contact FULL HOUSE
Agency

 fullHouseAgency@fha.dz

 +213 3698710

 190 AV Mustapha Ali Biar- Alger

Back





VI -Conclusion :

Ce projet de gestion d'agence immobilière a permis de développer une application complète et fonctionnelle pour la gestion des opérations immobilières. En utilisant Java pour le développement backend, Java Swing pour l'interface utilisateur et MySQL avec XAMPP serveur et myPhpAdmin pour la gestion de la base de données, nous avons réussi à créer une solution intégrée et efficace.

Les fonctionnalités clés incluent la gestion des comptes administrateurs, des agents, des acheteurs, des vendeurs, des propriétés et des transactions. Cette application simplifie et automatise les processus administratifs, améliorant ainsi l'efficacité opérationnelle de l'agence immobilière.

En conclusion, ce projet a non seulement renforcé nos compétences techniques en développement d'applications, mais il a également démontré l'importance de l'intégration harmonieuse entre différents composants logiciels pour créer des solutions robustes et fiables.

Remarque : Vous pouvez consulter notre repository sur GitHub pour plus de détails sur notre application « **FULL HOUSE MANAGEMENT SYSTEM APP** », il contient un fichier TXT README ainsi qu'une vidéo, ça va vous aider à bien comprendre notre projet !

Lien vers repository Github :

<https://github.com/djouharadgb/Full-House-real-estate-management-system--JAVA-App.git>