

RAPPORT

Application java de gestion d'une agence immobiliere.

22 MAI

ACAD C

CHECK

<https://github.com/lynlyna04/AGENCE-IMMOBILIERE-OOP>



TEAM MEMBERS

Ouabel Lyna
Ghoul Safia
Djeghlaf Lydia
Boukhemkhem Lina

SOMMAIRE

INTRODUCTION.....	1
CONCEPTION.....	2
FONCTIONALITES.....	7
EXEMPLE DE CODE.....	8
GUI.....	9
FONCTIONALITES CLES	10
AVANTAGES DE L'APPLI ET TECH.....	11
EXEMPLE D'INTERFACE.....	12
CONCLUSION.....	13



INTRODUCTION

Dans le contexte actuel de numérisation croissante et de compétitivité accrue dans le secteur immobilier, l'optimisation des processus de gestion est devenue une nécessité pour les agences immobilières souhaitant rester performantes et réactives. Ce rapport présente le développement d'une application de gestion d'agence immobilière, conçue pour automatiser et optimiser les différentes opérations liées à la gestion des biens immobiliers, des clients et des transactions.

L'objectif principal de cette application est de fournir une solution intégrée et conviviale qui simplifie la gestion quotidienne, améliore l'efficacité opérationnelle et renforce la satisfaction client. En automatisant les tâches répétitives et en centralisant les informations, l'application permet aux agents immobiliers de se concentrer davantage sur les activités à forte valeur ajoutée, telles que le service à la clientèle et la conclusion de transactions.

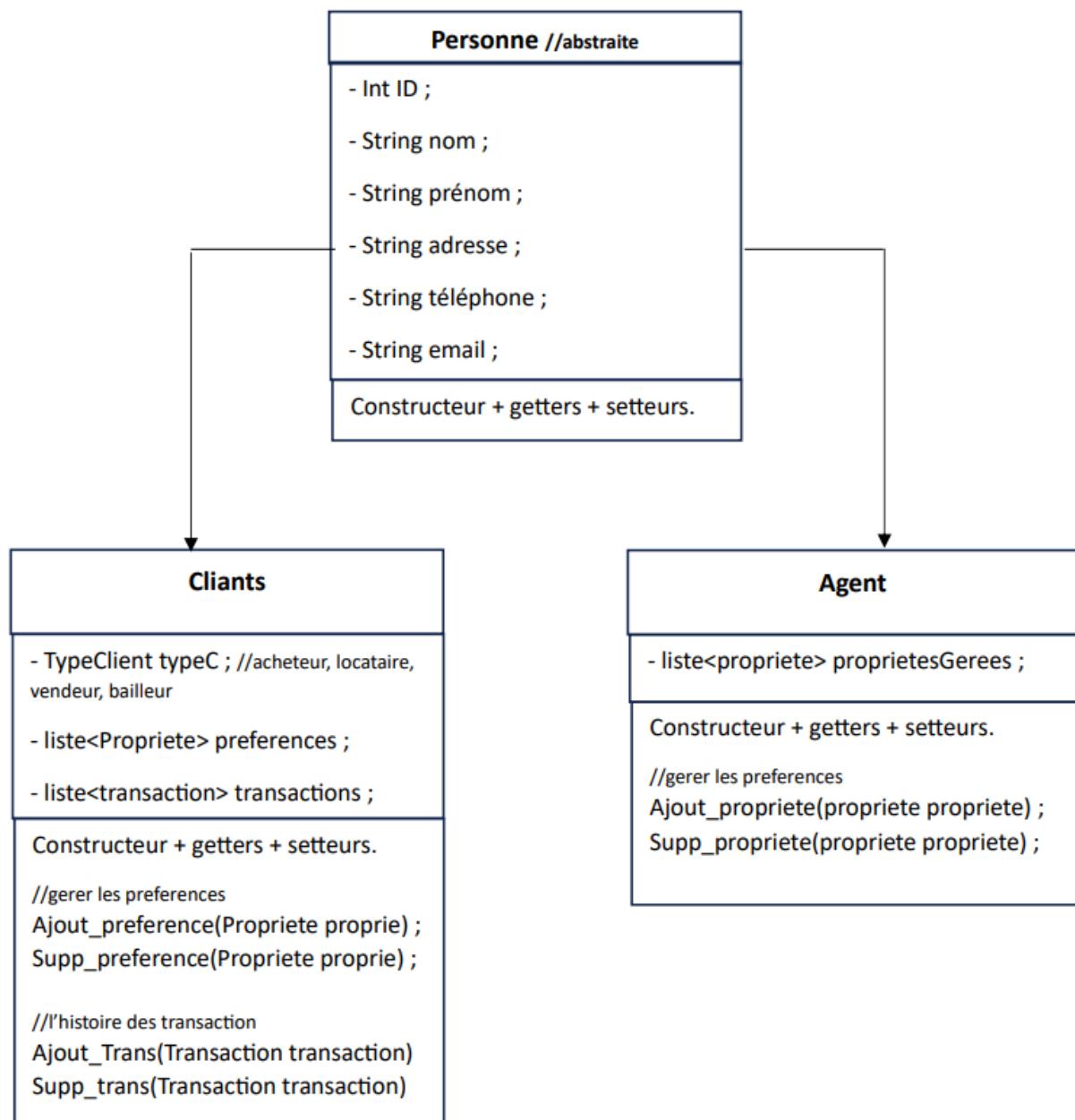
Ce rapport détaille les étapes clés du développement de l'application, des besoins initiaux à la mise en œuvre finale, en passant par la conception, le développement et les tests. Il examine également les bénéfices attendus en termes de gain de temps, de réduction des erreurs humaines et d'amélioration de la gestion des relations avec les clients. Grâce à cette solution technologique innovante, les agences immobilières peuvent anticiper une meilleure gestion de leurs activités et une plus grande compétitivité sur le marché immobilier.



CONCEPTION

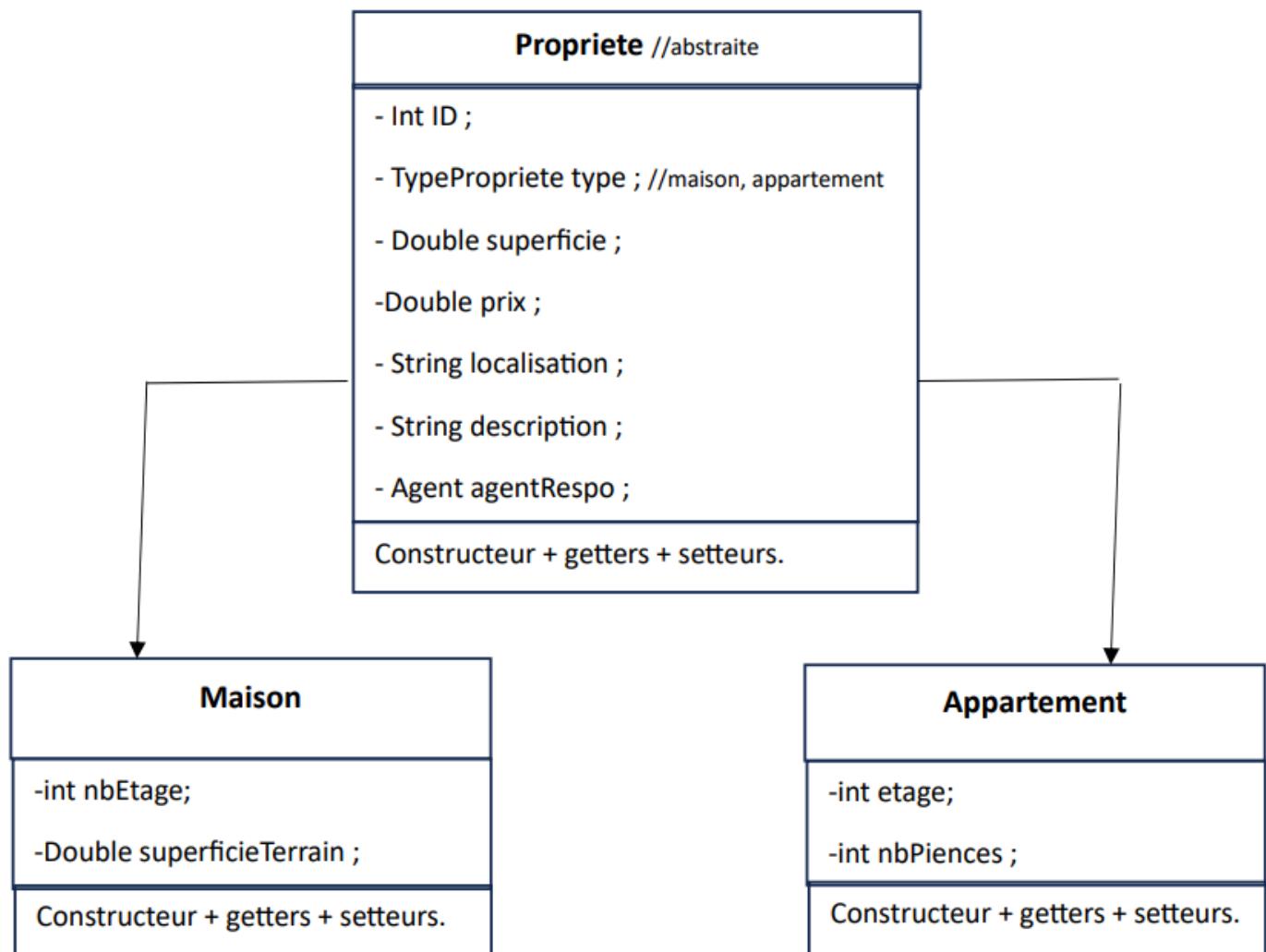
Les énumérations définissent des types spécifiques pour les propriétés, transactions, clients et rendez-vous, assurant une meilleure lisibilité et maintenance du code.

```
enum TypePropriete { APPARTEMENT, MAISON }
enum TypeTransaction { VENTE, LOCATION }
enum TypeClient { ACHETEUR, LOCATAIRE, VENDEUR, BAILLEUR }
enum TypeRDV { PLANIFIE, REALISE, ANNULE }
```

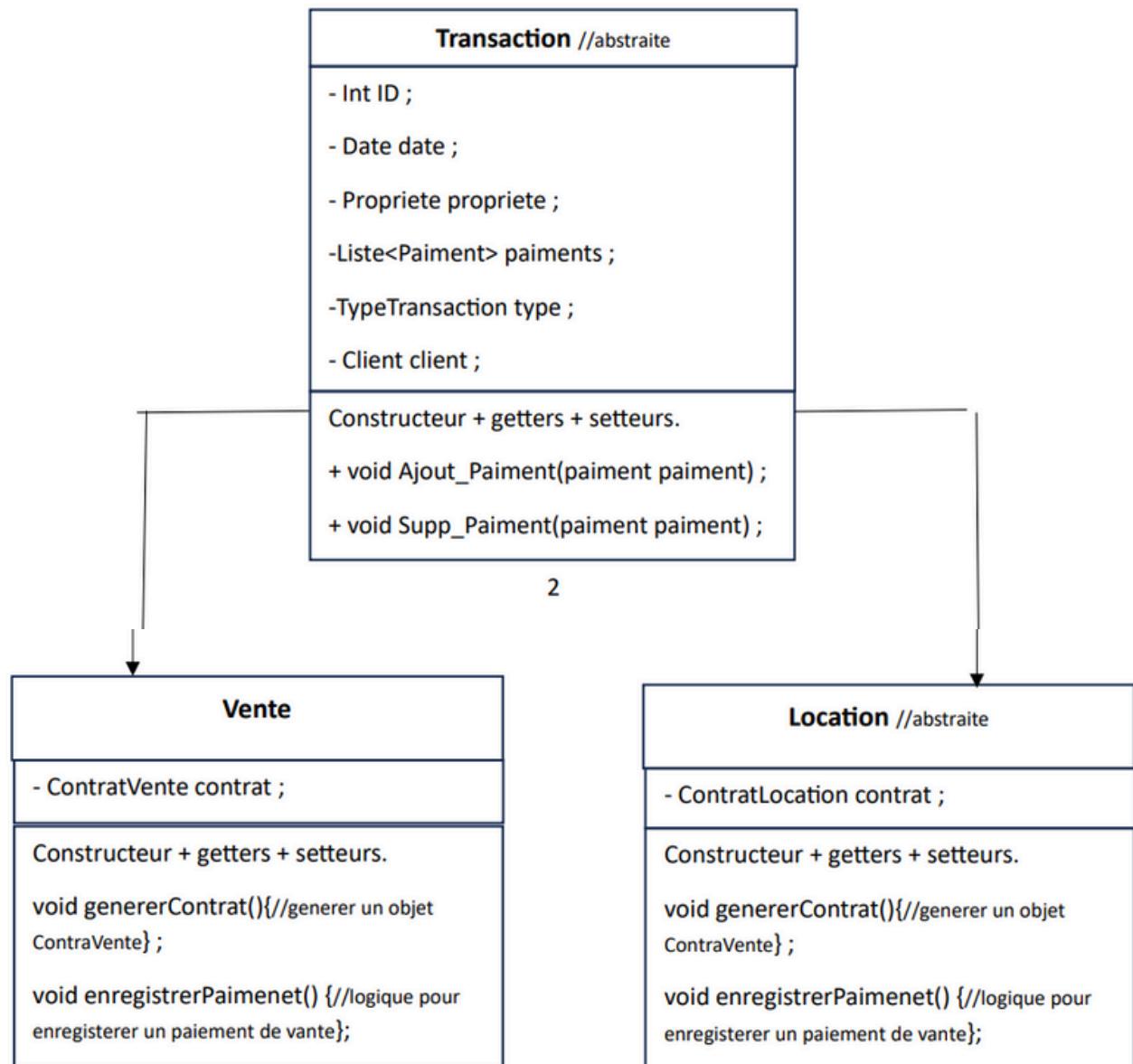


La méthode ajout_preference dans la classe Client concerne les préférences du client en termes de propriétés recherchées, tandis que la méthode ajout_propriete dans la classe Agent concerne l'attribution des propriétés à gérer par l'agent.

La classe Propriete représente une propriété générale avec des attributs communs à toutes les propriétés. Les classes Maison et Appartement héritent de cette classe et ajoutent des attributs spécifiques.

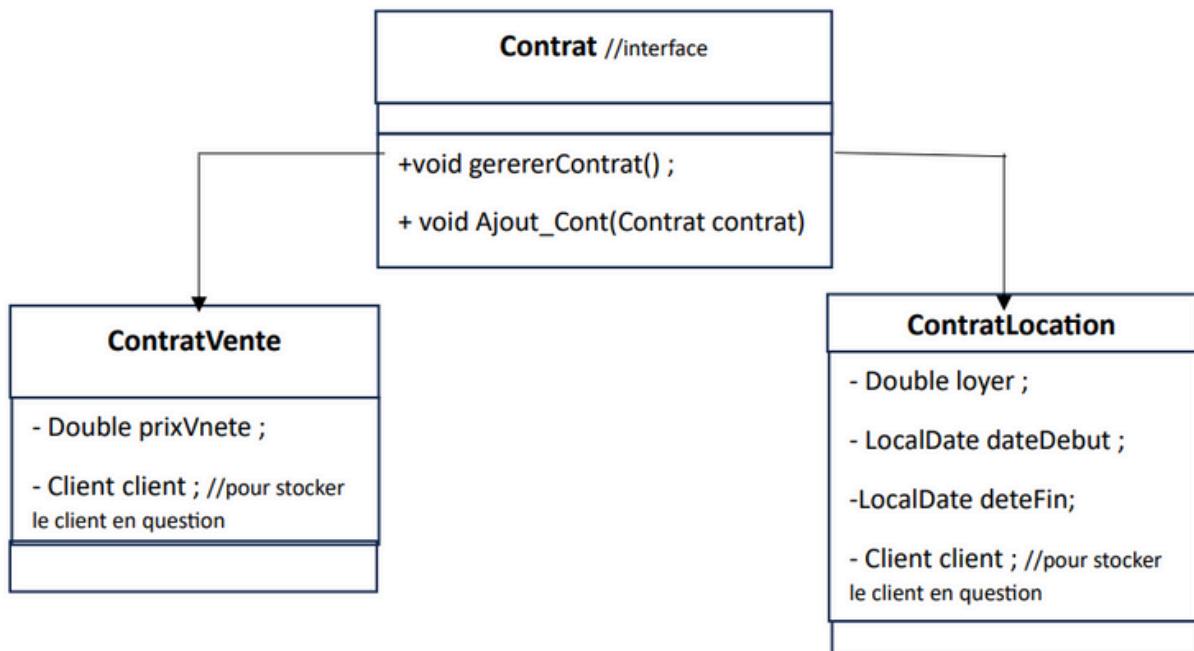


Cette classe représente une transaction générale avec des attributs communs, tandis que les classes Vente et Location ajoutent des spécificités.



L'interface **Contrat** définit les méthodes pour gérer les contrats.





Cette classe permet de définir les critères de recherche pour les propriétés.

Cette classe gère les paiements liés aux transactions.

CritererecherchePropriete
-Double Prixmin;
-Double Prixmax;
-TypePropriete type ;
-String localisation;
Constructeur + getters + setteurs.

Paiement
-Int ID;
-Double montant;
-bool paye;
-Transaction transaction;
Constructeur + getters + setteurs.
+ void marq_payer();



<p>RDV</p> <ul style="list-style-type: none"> -Int ID; -LocalDateTime dateHeure; -Propriete propriete; -Client client; -Agent agent -TypeRDV typeR ; <p>Constructeur + getters + setteurs.</p> <ul style="list-style-type: none"> + void rdv_planifier(LocalDateTime){} ; +void rdv_annuler(); +void rdv_realiser() ; 	<p>ServicePropriete</p> <ul style="list-style-type: none"> + List<Propriete> getToutesLesProprietes(){}; +Propriete getProprieteParId(int id){}; +void ajouterPropriete(Propriete propriete) {}; +void modifierPropriete(Propriete propriete) {}; +void supprimerPropriete(Propriete propriete) {}; +List<Propriete> rechercherProprietes(CritereRecherchePropriete criteres) {};
<p>ServiceClient</p> <ul style="list-style-type: none"> +List<Client> getTousLesClients() {}; +Client getClientParId(int id) {}; +void ajouterClient(Client client) {}; +void modifierClient(Client client) {}; +void supprimerClient(Client client) {}; 	<p>ServiceTransaction</p> <ul style="list-style-type: none"> +List<Transaction> getToutesLesTransactions() {}; +Transaction getTransactionParId(int id) {}; +void ajouterTransaction(Transaction transaction) {}; +void modifierTransaction(Transaction transaction) {}; +void supprimerTransaction(Transaction transaction) {}; +void ajouterPaiement(Paiement paiement) {}; +void modifierPaiement(Paiement paiement) {}; +void supprimerPaiement(Paiement paiement) {};
<p>ServiceRDV</p> <pre style="font-family: monospace; margin: 0;">//retourne la liste de tous les rdv +List<RendezVous> getTousRDV() {}; //prend un ID et retourne son rdv +RendezVous getRDVParID(int id) {}; +void ajouterRDV(RendezVous RDV) {}; +void modifierRDV(RendezVous RDV) {}; +void supprimerRDV(RendezVous RDV){}; //ces methodes retournent l'historique des rdv d'une perso + List<RendezVous> getRDVPropriete (Propriete propriete) + List<RendezVous> getRDVClient (Client client) + List<RendezVous> getRDVAgent(Agent agent)</pre>	



FONCTIONALITES

1. Insertion:

- Ajout de Nouveaux Enregistrements : Permet l'ajout de nouveaux biens immobiliers, clients, transactions, paiements, et rendez-vous dans le système.
- Expansion des Données : Facilite la mise à jour continue de la base de données avec des informations nouvelles et pertinentes.
- Gestion de l'Inventaire: Ajoute de nouvelles propriétés à la liste des biens disponibles pour la vente ou la location.

Exemple :

- ajouterClient(Client client) : Ajoute un nouveau client, permettant ainsi de gérer de nouveaux acheteurs ou locataires potentiels.

2. Suppression

- Nettoyage des Données: Permet de supprimer des enregistrements obsolètes ou incorrects pour maintenir la base de données propre et à jour.
- Gestion de l'Inventaire : Retire les propriétés qui ne sont plus disponibles ou qui ont été vendues/achetées.
- Conformité : Supprime les informations des clients qui ont quitté l'agence, respectant ainsi les normes de confidentialité et de gestion des données personnelles.

Exemple :

- supprimerPropriete(Propriete proprieté) : Enlève une propriété de la liste, assurant que seules les propriétés disponibles sont affichées.

3. Modification

- Mise à Jour des Informations : Permet la mise à jour des informations des propriétés, clients, transactions, paiements, et rendez-vous pour refléter les changements.
- Correction des Erreurs : Facilite la correction des erreurs dans les enregistrements existants sans avoir à supprimer et réinsérer les données.



Exemple:

-**modifierClient(Client client)** : Met à jour les informations du client, telles que son adresse ou ses préférences en matière de propriété.

4. Affichage

-**Consultation des Données** : Permet aux utilisateurs de consulter les données stockées dans le système de manière claire et organisée.

-**Aide à la Décision** : Facilite la prise de décision pour les agents immobiliers et les clients en leur fournissant des informations pertinentes et à jour.

-**Transparence** : Assure la transparence dans les opérations de l'agence immobilière en fournissant un accès facile aux informations sur les propriétés et les transactions.

Exemple :

-**getToutesLesProprietes()** : Affiche toutes les propriétés disponibles, aidant les agents et les clients à voir l'ensemble des options disponibles.

EXEMPLE DE CODE

```
switch (reponse1.toLowerCase()) {
    case "personnes":
        do {
            System.out.println("Voulez-vous effectuer une modification dans le service personnes ? (Oui/Non)");
            reponse2 = scanner.nextLine();
            if (reponse2.equalsIgnoreCase("Oui")) {
                System.out.println("Sur quel type de propriétés voulez-vous effectuer votre modification ? (Client/Agent)");
                reponse3 = scanner.nextLine();
                switch (reponse3.toLowerCase()) {
                    case "client":
                        System.out.println("Quelle modification voulez-vous apporter au service ? (Ajout/Suppression/Modification)");
                        reponse4 = scanner.nextLine();
                        switch (reponse4.toLowerCase()) {
                            case "ajout":
                                System.out.println("Entrez les détails du client :");
                                Client nouveauClient = Client.createClientFromInput();
                                serviceClient.ajouterClient(nouveauClient);
                                break;
                            case "suppression":
                                System.out.println("Entrez l'ID du client à supprimer : ");
                                int idSuppressionClient = scanner.nextInt();
                                scanner.nextLine();

```

traitement des données dans le main avec des scanners et des system.out.println

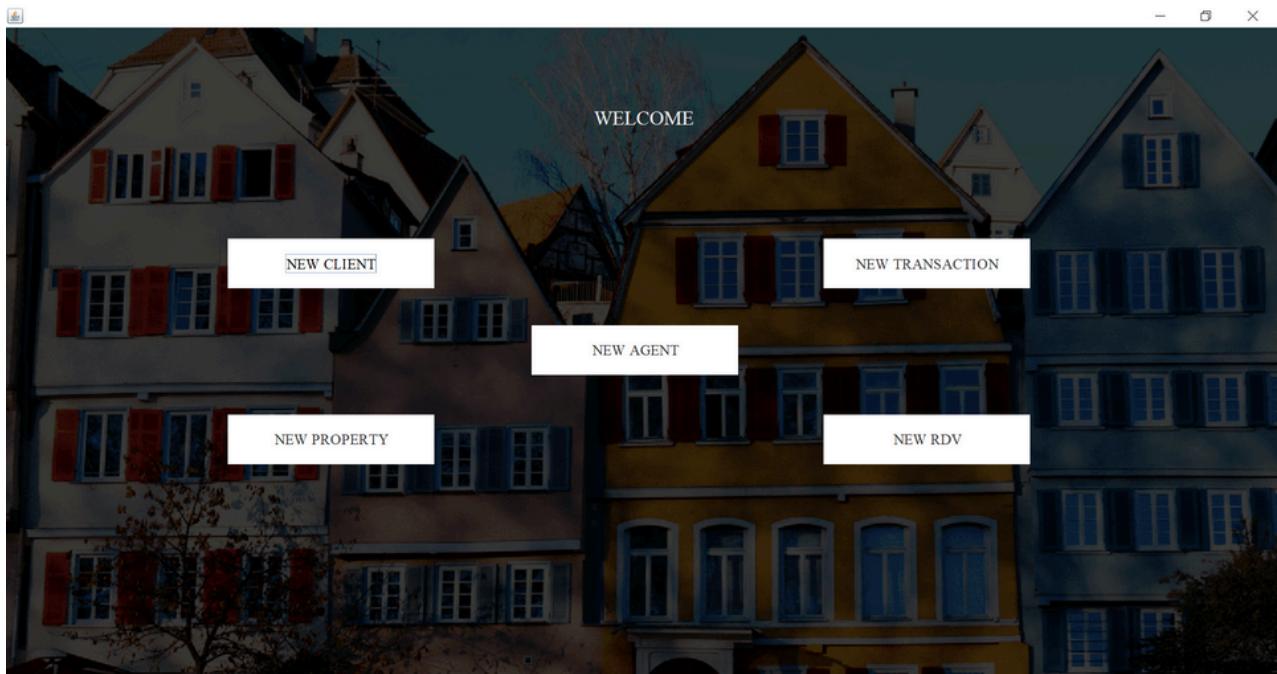


GUI

Description Générale

L'application est conçue en Java, avec une interface graphique (GUI) créée à l'aide de WindowBuilder dans Eclipse. Elle se connecte à une base de données Oracle pour la gestion des informations. L'interface utilisateur est divisée en plusieurs sections, chacune dédiée à la gestion d'un aspect spécifique de l'agence immobilière :

1. **Clients** : Gestion des informations personnelles des clients.
2. **Agents** : Gestion des données des agents immobiliers.
3. **Propriétés** : Suivi des propriétés disponibles et leurs caractéristiques.
4. **Rendez-vous** : Organisation et suivi des rendez-vous entre les clients et les agents.
5. **Transactions** : Enregistrement et gestion des transactions immobilières.



FONCTIONNALITÉS CLÉS

1. Insertion des Données : Permet aux utilisateurs de saisir de nouvelles informations pour les clients, les agents, les propriétés, les rendez-vous et les transactions via des formulaires intuitifs.
2. Modification des Données : Facilite la mise à jour des informations existantes avec des formulaires pré-remplis et des boutons de validation.
3. Suppression des Données : Permet de supprimer des enregistrements sélectionnés via des options de radio boutons et des confirmations de suppression.
4. Affichage des Données : Présente les données sous forme de tableau pour une visualisation claire et ordonnée, facilitant ainsi la recherche et la consultation.

The screenshot displays a software application window titled "CLIENTS". At the top right, there is a search bar labeled "Nom" with a dropdown arrow and a "Search" button. Below the title, there is a table with columns: ID, NOM, PRENOM, ADRESSE, TELEPHONE, EMAIL, and TYPE_CLIENT. The table contains the following data:

ID	NOM	PRENOM	ADRESSE	TELEPHONE	EMAIL	TYPE_CLIENT
4	ouabel	aminaaa	dz	9876542	amy@gmail.com	bailleur
5	ghoul	safia	alger	0223344556	safie@gmail.com	VENDEUR
7	ouabel	walid	oran	2222222	hehe@gmail.com	vendeur
8	lynda	ouabel	algeRR	000000000	lyna@gmail.com	locataire

On the left side of the screen, there are input fields for "NOM", "PRENOM", "EMAIL", "ADRESSE", "TELEPHONE", and "TYPE". The "TYPE" field includes radio buttons for "ACHETEUR", "LOCATAIRE", "VENDEUR", and "BAILLEUR". At the bottom left are buttons for "AJOUTER", "AFFICHER CLIENTS", and "New client". On the right side, there are buttons for "Delete", "Modify", and "Save".



TECHNOLOGIES UTILISÉES



Langage de Programmation
Java



Oracle data base pour le
backend



Interface Graphique :
Swing avec WindowBuilder
dans Eclipse

AVANTAGES DE L'APPLICATION

L'application offre une solution centralisée et automatisée pour la gestion des données immobilières, réduisant ainsi les erreurs humaines et augmentant l'efficacité opérationnelle. Grâce à son interface utilisateur conviviale, elle permet aux agents immobiliers de se concentrer davantage sur leurs interactions avec les clients et sur la conclusion des transactions, plutôt que sur la gestion administrative des données.



QUELQUE INTERFACES

AGENTS Nom Search

NOM :

PRENOM :

EMAIL :

ADRESSE :

TELEPHONE :

AJOUTER AFFICHER AGENTS New agent Delete Modify Save

ID	NOM	PRENOM	ADRESSE	TELEPHONE	EMAIL
1	ssdfds	fdsgfdgfd	dz	345678987654	gvfdgsdss

AGENTS Nom Search

NOM :

PRENOM :

EMAIL :

ADRESSE :

TELEPHONE :

AJOUTER AFFICHER AGENTS New agent Message X

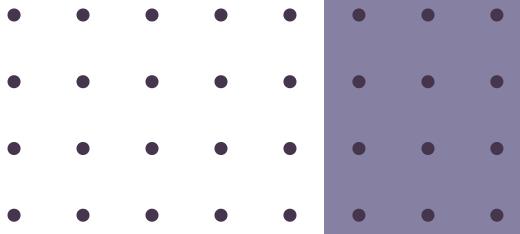
Agent details updated successfully.

OK

Delete Modify Save

ID	NOM	PRENOM	ADRESSE	TELEPHONE	EMAIL
1	ssdfds	fdsgfdgfd	dz	345678987654	gvfdgsdss





CONCLUSION

Le développement de cette application de gestion d'agence immobilière marque une avancée majeure dans l'optimisation des processus internes, en automatisant et centralisant la gestion des biens, des clients, des transactions et des rendez-vous. En intégrant des fonctionnalités essentielles telles que l'insertion, la suppression, la modification et l'affichage des données, l'application améliore l'efficacité opérationnelle et la précision des informations. Elle permet aux agents de se concentrer sur des activités à forte valeur ajoutée, comme la négociation et la conclusion de ventes ou de locations, tout en offrant un meilleur service à la clientèle. Cette solution technologique positionne les agences immobilières pour rester compétitives dans un marché en évolution rapide, en augmentant leur performance et leur satisfaction client.

