Gr’immo

Une image contenant logo, Graphique, Police, symbole

Description générée automatiquement

Table des matières

[Introduction 3](#_Toc187064688)

[Diagramme de cas d’utilisation 3](#_Toc187064689)

[Base de données 5](#_Toc187064690)

[Modèle Conceptuel de Données (MCD) 5](#_Toc187064691)

[Règle de gestion 5](#_Toc187064692)

[Modèle Logique de Données (MLD) 7](#_Toc187064693)

[Relations : 7](#_Toc187064694)

[Diagramme de classe 8](#_Toc187064695)

[Répartition des tâches au seins de l’équipe 9](#_Toc187064696)

[Développement des fonctionnalité primaires 10](#_Toc187064697)

[Connexion et communication avec le serveur LDAP 10](#_Toc187064698)

[Connexion et communication avec la base de données PostgreSQL 10](#_Toc187064699)

[Problèmes rencontrés 10](#_Toc187064700)

[Solution mise en place 11](#_Toc187064701)

# Introduction

Gr’immo est une agence immobilière située à Grenoble qui prévoit de se développer dans d’autres villes. L’agence nous a donc demandé de développer un client lourd permettant de gérer leur base de données contenant l'ensemble de leurs biens, ainsi qu'un agenda.

# Diagramme de cas d’utilisation

Le diagramme de cas d'utilisation représente les fonctionnalités principales d’un système de gestion immobilière, avec deux types d’utilisateurs identifiés : **l’agent** et **le responsable**. Voici une description des cas d’utilisation associés à chaque acteur :

**1. L’agent**

L’agent, en tant qu’utilisateur du système, dispose des fonctionnalités suivantes :

* **Gérer ses biens** : visualiser la liste des biens qu’il gère personnellement, ajouter de nouveaux biens, modifier ses biens.
* **Gérer son agenda** : organiser et planifier ses rendez-vous, visites ou événements professionnels.
* **Gérer ses clients** : suivre et gérer les informations des clients qui lui sont attribués, tels que leurs coordonnées ou leurs demandes.
* **Se connecter à l’AD (Annuaire LDAP)** : s’authentifier via un serveur LDAP pour accéder au système de manière sécurisée.

**2. Le responsable**

Le responsable dispose d’un rôle élargi avec des privilèges supplémentaires, incluant toutes les fonctionnalités des agents, ainsi que les suivantes :

* **Gérer tous les biens** : accès global à la gestion de l’ensemble des biens du système, indépendamment des agents responsables.
* **Gérer tous les clients** : possibilité de gérer les informations et interactions de tous les clients, sans restriction.
* **Ajouter un agent** : capacité d’ajouter de nouveaux agents au système, leur permettant ainsi d’utiliser la plateforme pour leurs activités.

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

Description générée automatiquement

Figure : Diagramme de cas

# Base de données

## Modèle Conceptuel de Données (MCD)

Pour réaliser ce projet, nous avons dû réaliser un MCD qui nous a permis de créer la structure de la base de données de manière correcte pour répondre aux précisément aux besoins de l’application

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

Description générée automatiquement

Figure : Modèle Conceptuel de Données (MCD)

### Règle de gestion

#### Entité PROPRIÉTAIRE

Un propriétaire peut posséder un ou plusieurs biens.

#### Entité BIENS

Un bien doit appartenir à un et un seul propriétaire.

Un bien peut être loué ou acheté par un ou plusieurs clients.

Un bien peut être administré par un et un seul utilisateur

#### Entité USERS

Un utilisateur peut administrer un ou plusieurs biens.

Un utilisateur peut avoir zéro ou plusieurs événements.

Un utilisateur peut avoir un ou plusieurs clients.

Un utilisateur peut avoir zéro ou plusieurs messages.

Un utilisateur peut superviser un ou plusieurs propriétaires.

#### Entité EVENTS

Un événement doit être ajouté par un et un seul utilisateur.

#### Entité Clients

Un client est géré par un et un seul utilisateur.

Un client peut être associé à un ou plusieurs biens

#### Entité MESSAGES

Un message doit être envoyé par un et un seul utilisateur.

Un message est caractérisé par une description, une date et une heure.

## Modèle Logique de Données (MLD)

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

Description générée automatiquement

Figure : Modèle Logique de Données (MLD)

### Relations :

Entité BIENS :

uid\_user : Clé étrangère faisant référence à la clé primaire uid\_user de l'entité USERS.

uid\_proprio : Clé étrangère faisant référence à la clé primaire uid\_proprio de l'entité PROPRIETAIRE.

Entité SUPERVISER :

uid\_user : Clé étrangère faisant référence à la clé primaire uid\_user de l'entité USERS.

uid\_proprio : Clé étrangère faisant référence à la clé primaire uid\_proprio de l'entité PROPRIETAIRE.

Entité CLIENT :

uid\_user : Clé étrangère faisant référence à la clé primaire uid\_user de l'entité USERS.

Entité AJOUTE :

uid\_user : Clé étrangère faisant référence à la clé primaire uid\_user de l'entité USERS.

id\_event : Clé étrangère faisant référence à la clé primaire id\_event de l'entité EVENT.

Entité MESSAGE : uid\_user : Clé étrangère faisant référence à la clé primaire uid\_user de l'entité USERS.

Entité EVENT : Pas de clé étrangère directe visible, mais liée à AJOUTE via la clé primaire id\_event.

## Diagramme de classe

Une image contenant texte, diagramme, Plan, ligne

Description générée automatiquement

Figure : Diagramme de classe

La classe **PROPRIÉTAIRE** représente les propriétaires de biens immobiliers, avec des informations personnelles comme leur nom, prénom, email et téléphone. Chaque propriétaire possède un ou plusieurs biens.

La classe **BIENS** décrit les propriétés immobilières avec des caractéristiques comme l'adresse, la surface, le prix et le statut (louable ou achetable). Chaque bien est associé à un propriétaire et peut être administré par des utilisateurs.

La classe **USERS** représente les utilisateurs du système, comme les agents ou les responsables, avec des informations telles que leur nom, prénom, téléphone et type (agent ou responsable). Ils peuvent administrer des biens et ajouter des événements.

La classe **CLIENTS** regroupe les personnes intéressées par l'achat ou la location de biens. Chaque client est géré par un utilisateur et peut être associé à un ou plusieurs biens.

La classe **EVENTS** contient les événements liés au système, comme des visites de propriétés, avec des informations comme la date, l'heure et le lieu. Les événements sont ajoutés par des utilisateurs.

La classe **MESSAGES** permet de gérer la communication entre utilisateurs et clients, avec des informations comme la description, la date et l'heure d'envoi.

Les relations entre les classes définissent les interactions : les propriétaires possèdent des biens, les utilisateurs administrent des biens et ajoutent des événements, les clients achètent ou louent des biens, et les messages facilitent la communication.

# Répartition des tâches au seins de l’équipe

Dans le cadre de notre projet de développement d’un logiciel pour une agence immobilière, nous avons adopté une méthodologie collaborative qui nous a permis de répartir les tâches efficacement et de suivre l’avancement du projet de manière structurée. L’équipe était composée de trois membres : Issam Moussi, Lucas Troussier et moi-même.

Pour organiser notre travail, nous avons utilisé le logiciel **Gantt Project**, qui nous a permis de planifier les différentes étapes du projet et de répartir les tâches entre les membres de l’équipe. Cet outil a été essentiel pour établir un planning détaillé et pour définir les responsabilités de chacun tout au long du développement du logiciel.

En complément, nous avons également utilisé la plateforme **Trello** pour le suivi quotidien de l’avancement du projet. Trello nous a permis de centraliser nos tâches sous forme de tableaux, avec des colonnes représentant les différentes étapes du workflow : "To Do", "In Progress", et "Finish". Chaque tâche était associée à un membre de l’équipe, ce qui facilitait la collaboration et assurait une visibilité complète sur l’état d’avancement.

Une image contenant capture d’écran, texte, logiciel, Logiciel multimédia

Description générée automatiquement

Figure : Tâche Trello

Cette approche méthodologique nous a permis de travailler de manière fluide et efficace, en réduisant les risques de confusion ou de chevauchement des tâches. Chacun jouait un rôle précis :

* **Issam Moussi** s’est concentré développement de l’interface utilisateur et la base de données
* **Lucas Troussier** a pris en charge le développement de l’interface utiliateur et la base de données.
* Quant à moi, je me suis occupé du développement back-end et des fonctionnalités.

# Développement des fonctionnalité primaires

Les deux premières étapes réalisées durant le développement du logiciel ont été, le développement d’une fonction nous permettant d’établir une connexion avec la base de données pour ainsi exécuter des requêtes nous permettant d’interagir avec les données présente dans celle-ci. La deuxième fonction a été de créer une fonction nous permettant d’établir une connexion avec le serveur LDAP pour par la suite, pouvoir se connecter avec des compte utilisateur présent sur le serveur.

## Connexion et communication avec le serveur LDAP

## Connexion et communication avec la base de données PostgreSQL

# Problèmes rencontrés

# Solution mise en place

Donc pour la réalisation de se projet nous avons utiliser la langue de programmation python car il y a un implicité d’utilisasation de plus il possèdent un grand nombre de framework et de bibliothèque

Nous avons utilisé certain bibliothèque pour faire ce projet tel que :

* + PYQt5 pour linterface
  + LDAP3 pour la connexion pour ldap
  + Psycopg2 pour la base de données postegre
  + Private
  + Configparser pour la lecture de .env

Table des illustrations

[Figure 1: diagramme de cas d'utilisation 4](#_Toc183442889)

[Figure 2 : diagramme de classe 5](#_Toc183442890)

[Figure 3 : MLD V2 8](#_Toc183442891)

[Figure 4:MLD V2 11](#_Toc183442892)