
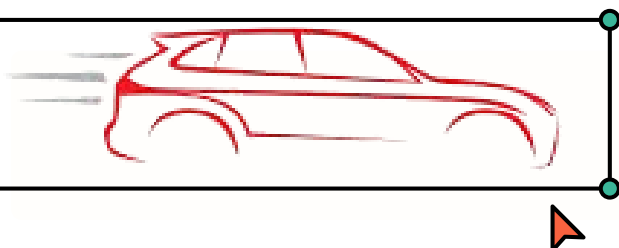
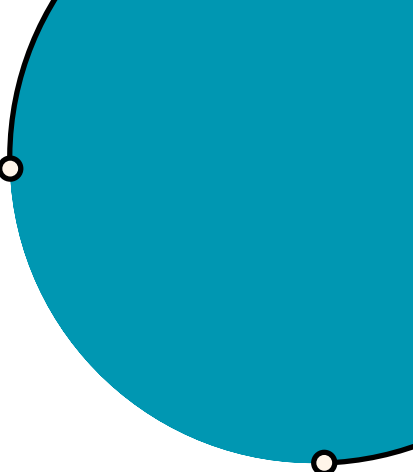


PROJET ENCADRÉ PAR  
MR GHARBI

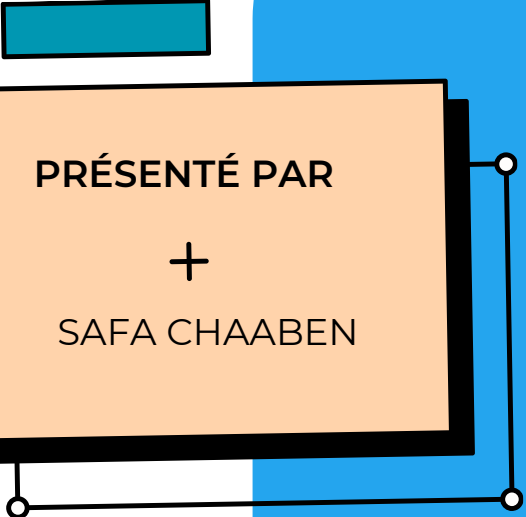
# GESTION DE CARTES GRISES



AVEC JAVA



PRÉSENTÉ PAR  
+  
SAFA CHAABEN



# Cahier des charges techniques

## Sommaire

### I. Contexte du projet

#### I.I. Présentation du projet

#### I.II. Date de rendu du projet

### II. Besoins fonctionnels

### III. Ressources nécessaires à la réalisation du projet

#### III.I. Ressources matérielles

#### III.II. Ressources logicielles

### IV. Gestion du projet

### V. Conception du projet

#### V.I. Le front-end

##### V.I.I. Wireframes

##### V.I.II. Maquettes

##### V.I.III. Arborescences

#### V.II. Le back-end

##### V.II.I. Diagramme de cas d'utilisation

##### V.II.II. Diagramme d'activités

##### V.II.III. Modèles Conceptuel de Données (MCD)

##### V.II.IV. Modèle Logique de Données (MLD)

##### V.II.V. Modèle Physique de Données (MPD)

### VI. Technologies utilisées

#### VI.I. Langages de développement Web

#### VI.II. Base de données

### VII. Sécurité

#### VII.I. Login et protection des pages administrateurs

#### VII.II. Cryptage des mots de passe avec Bcrypt

#### VII.III. Protection contre les attaques XSS (Cross-Site Scripting)

#### VII.IV. Protection contre les injections SQL

# I. Contexte du projet

## I.I. Présentation du projet

On occupe actuellement le poste de concepteur et développeur au sein de la Direction des systèmes d'information de la préfecture du département. La responsable du service des cartes grises souhaite faire évoluer l'application métiers. Cependant, aucun document de conception n'est disponible. Notre mission consiste donc à élaborer des documents de conception de l'application actuelle afin de faciliter la réflexion autour de son évolution.

## I.II. Date de rendu du projet

Le projet doit être rendu au plus tard le 6 mars 2025.

# II. Besoins fonctionnels

Il s'agit d'une application de gestion de cartes grises :

### I – Technologies :

- Back-end : Java
- Front-end : Javascript (optionnel)
- Basse de données : Relationnelle

### II – Sécurité :

- Pas d'authentification requise (Installation locale)
- Protection contre les attaques XSS et injections SQL

# III. Ressources nécessaires à la réalisation du projet

## III.I. Ressources matérielles

Les ressources matérielles nécessaires pour réaliser le projet sont :

- Pc portable, connexion internet (Wi-Fi)

## III.II Ressources logicielles

Les ressources logicielles nécessaires pour réaliser le projet sont :

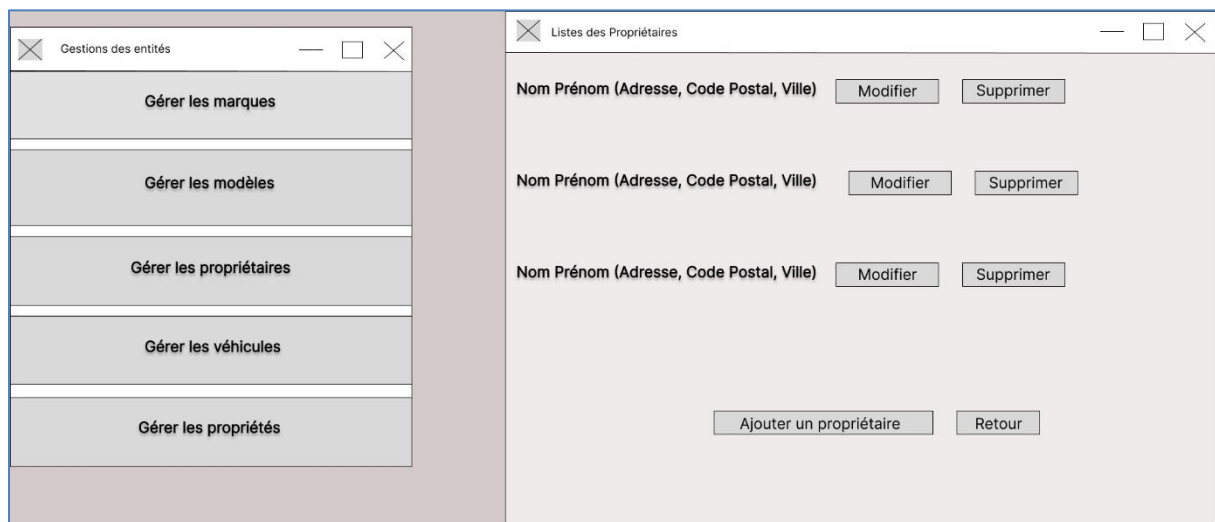
- Un IDE : Visual Studio Code
- Une plateforme de développement collaborative : Github
- MAMP :
  - Serveur web Apache
  - SGBDR : MySQL

## IV. Gestion du projet

## V. Conception du projet

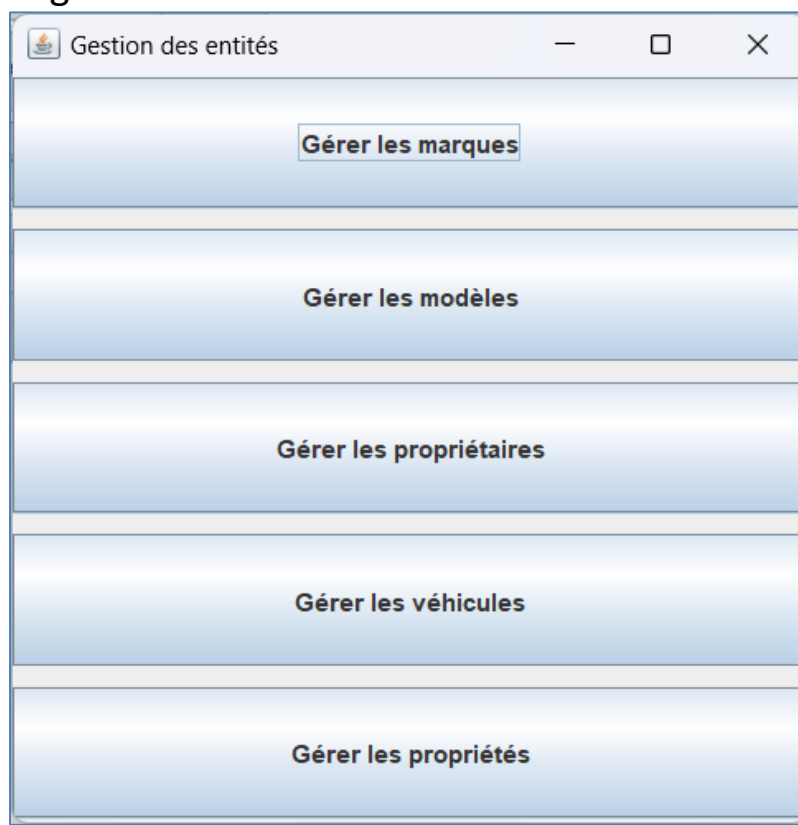
### V.I. Les Wireframes

Afin de réaliser les Wireframes, on a utilisé l'outil Figma.



### V.II Les Maquettes

Page d'accueil :



## Liste des propriétaires :

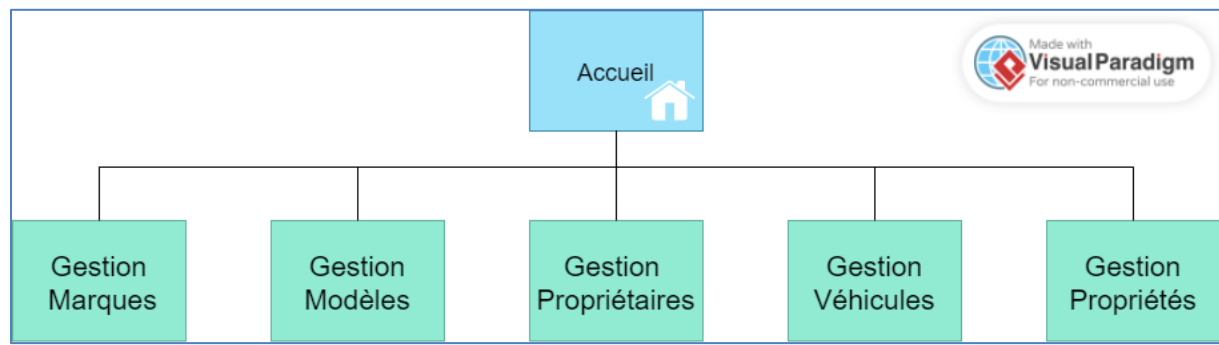
Liste des Propriétaires

Doe John (123 Rue de Paris, 75000 Paris)

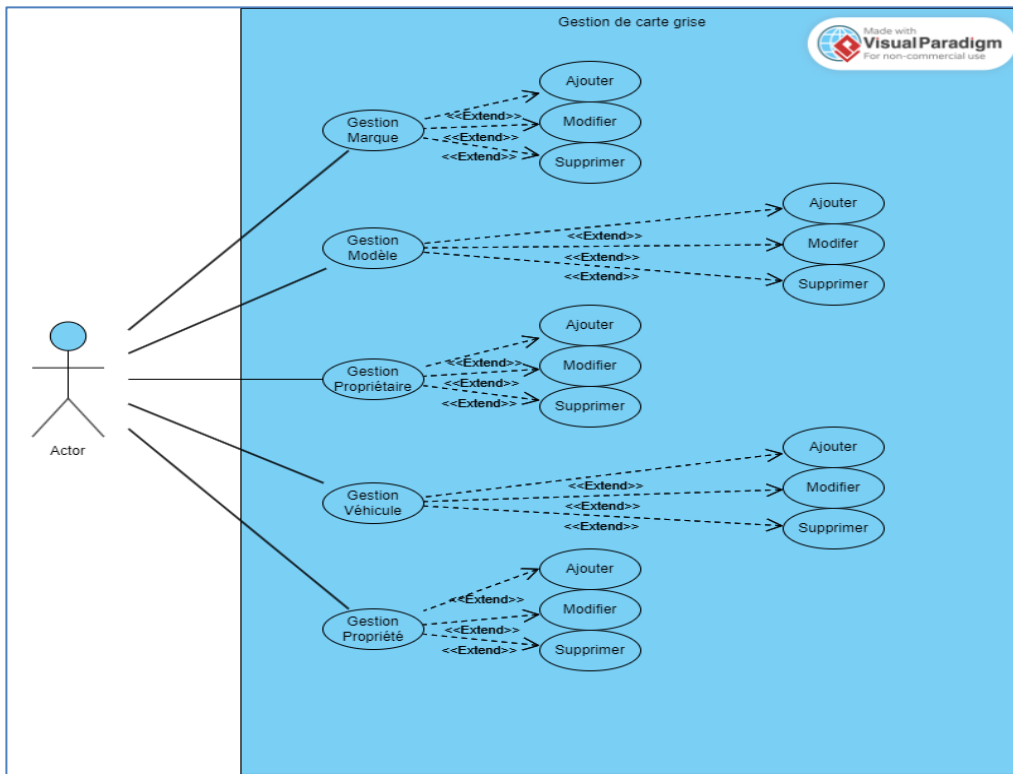
Doe Jane (456 Avenue des Champs, 75008 Paris)

Smith Alice (789 Boulevard Haussmann, 75009 Paris)

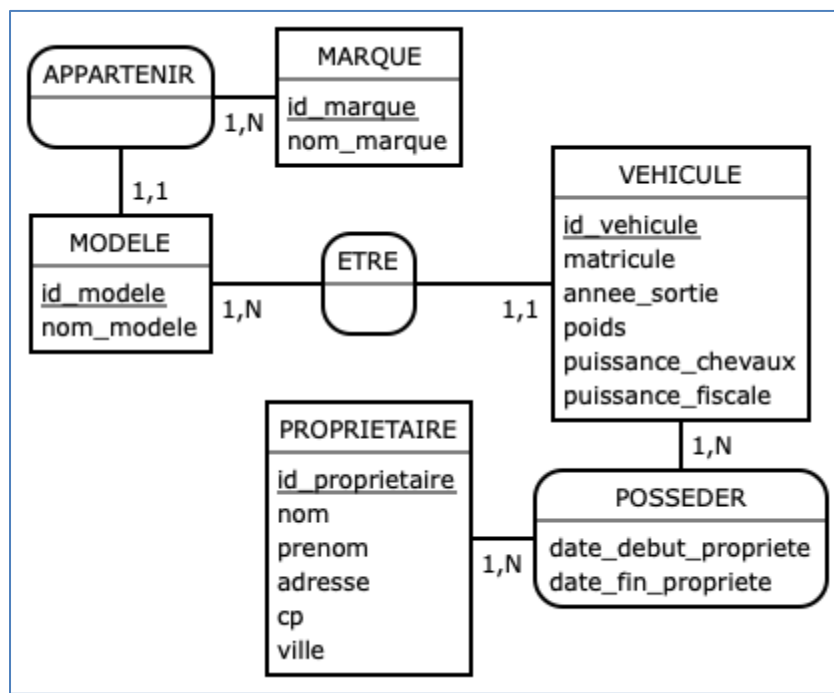
## V.III Arborescence de l'application



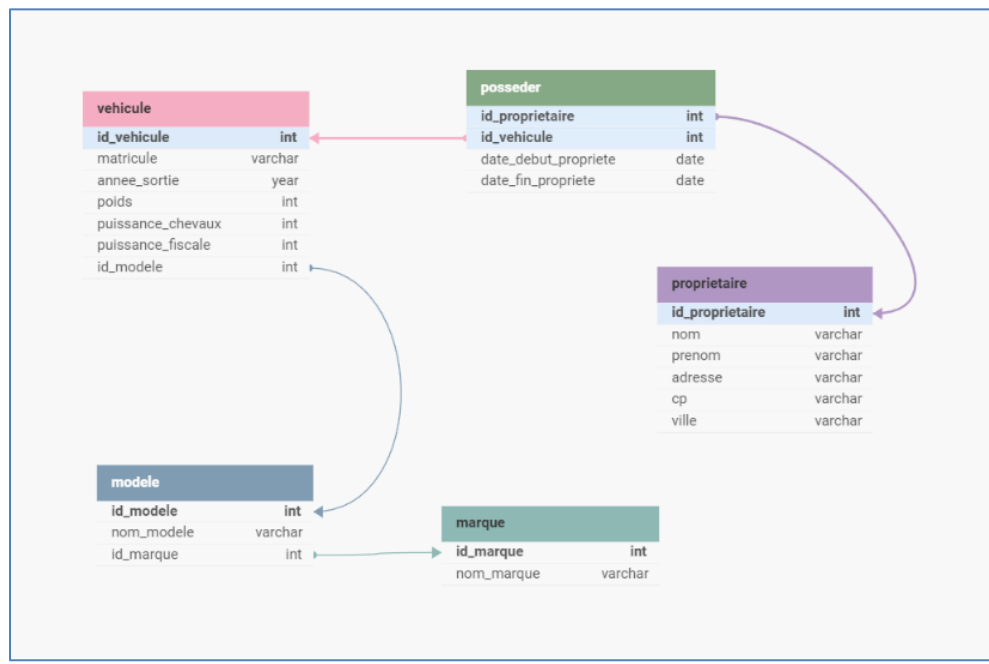
## V.IIIII Diagramme de cas d'utilisation



## V.IV MCD



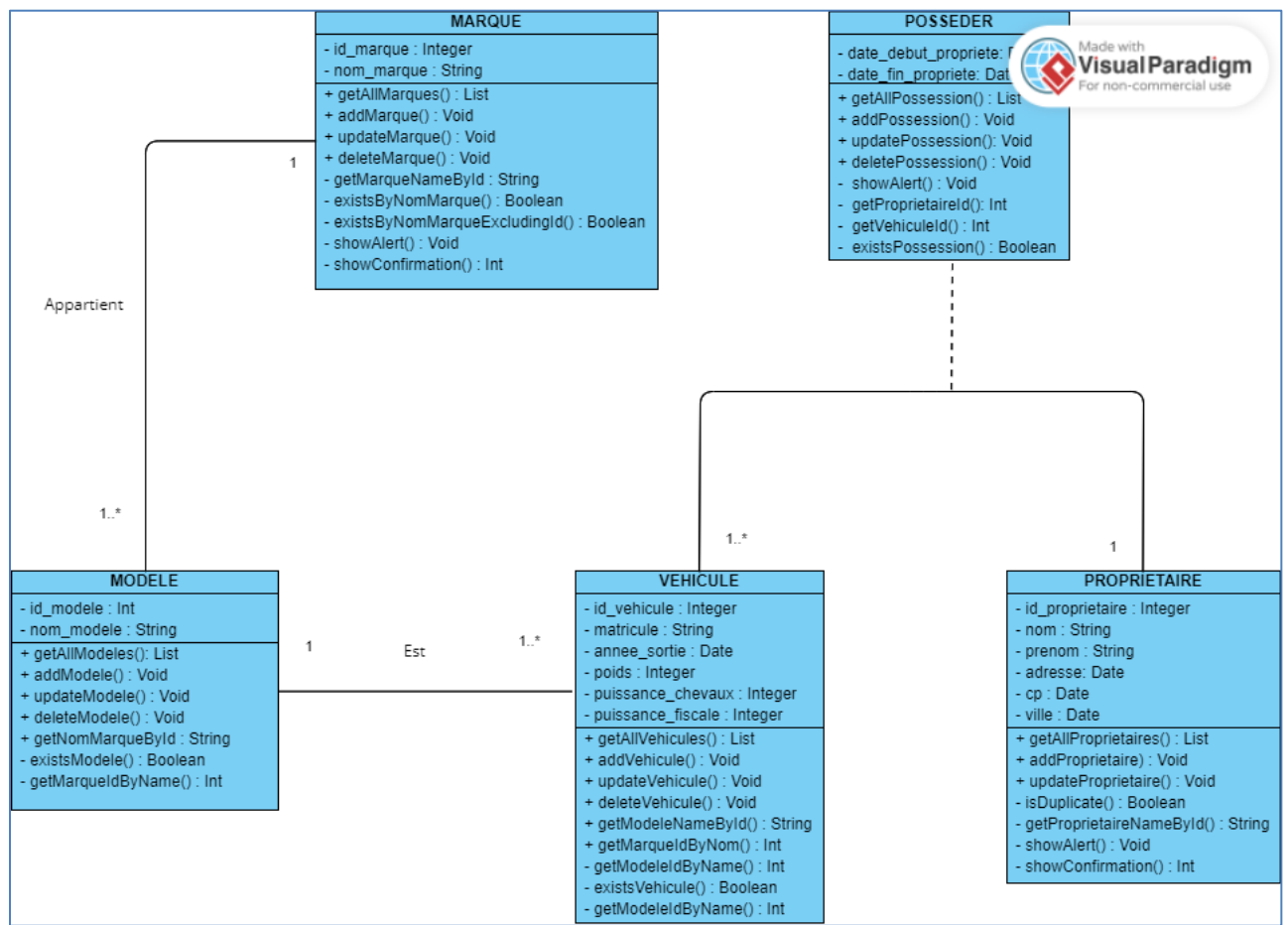
## V.IV MPD



## V.IV MLD (format BTS)

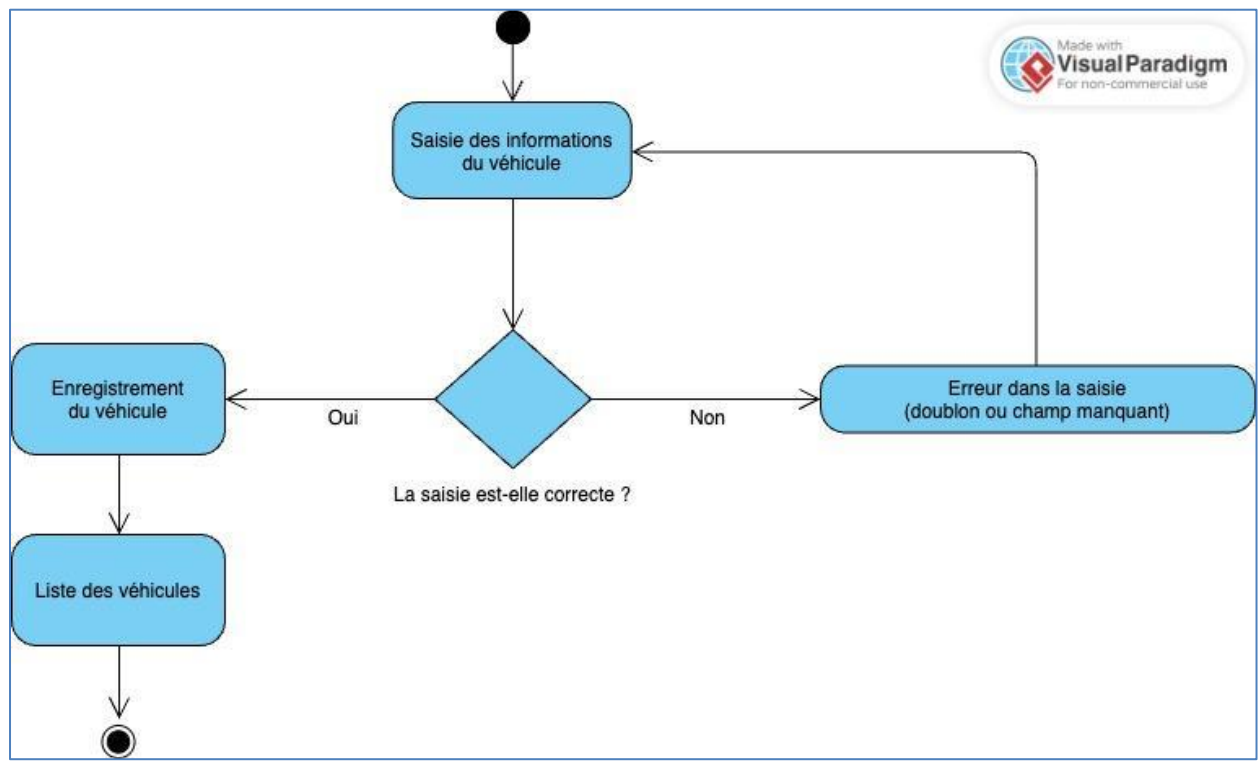
- MARQUE ( id\_marque, nom\_marque )
- MODELE ( id\_modele, nom\_modele, #id\_marque )
- POSSEDER ( #id\_vehicule, #id\_proprietaire, date\_debut\_propriete, date\_fin\_propriete )
- PROPRIETAIRE ( id\_proprietaire, nom, prenom, adresse, cp, ville )
- VEHICULE ( id\_vehicule, matricule, annee\_sortie, poids, puissance\_chevaux, puissance\_fiscale, #id\_modele )

## V.V Diagramme de Classe



## V.V Diagramme d'activité





## I. Technologies utilisées

### I.I. Langages de développement

**HTML :** Java

**Outil :** Visual Studio Code pour le développement et l'édition de code.

### I.II. Base de données

**Modélisation :**

- **Mocodo** pour le modèle conceptuel des données.
- **Visual Paradigm** pour la modélisation UML.

**Base de données :**

- **MAMP** pour l'environnement de développement local (Serveur PHP/MySQL).
- **MySQL** comme SGBR pour la gestion des données.

## II. Sécurité

### II.I Protection contre les injections SQL

Toutes les requêtes SQL utilisent des PreparedStatement pour éviter l'injection SQL.

```
// Insérer un nouveau propriétaire
try (PreparedStatement ps = conn.prepareStatement(
    sql:"INSERT INTO PROPRIETAIRE (nom, prenom, adresse, cp, ville) VALUES (?, ?, ?, ?, ?)")) {
    ps.setString(parameterIndex:1, nom);
    ps.setString(parameterIndex:2, prenom);
    ps.setString(parameterIndex:3, adresse);
    ps.setString(parameterIndex:4, cp);
    ps.setString(parameterIndex:5, ville);
    ps.executeUpdate();
    showAlert(title:"Succès", "Le propriétaire '" + nom + " " + prenom + "' a été ajouté avec succès !");
}
```

Les « ? » empêchent la concaténation de chaînes et donc l'injection SQL.

### II.II Vérification des doublons

Avant d'ajouter ou modifier un élément, on vérifie s'il n'est pas déjà présent en base de données. Cela empêche d'ajouter des doublons involontaires dans la base.

```
// Vérifie si un enregistrement est un doublon
private boolean isDuplicate(Connection conn, String nom, String prenom, String adresse, String cp, String ville) {
    String query = "SELECT COUNT(*) FROM PROPRIETAIRE WHERE nom = ? AND prenom = ? AND adresse = ? AND cp = ? AND ville = ?";
    try (PreparedStatement ps = conn.prepareStatement(query)) {
        ps.setString(parameterIndex:1, nom);
        ps.setString(parameterIndex:2, prenom);
        ps.setString(parameterIndex:3, adresse);
        ps.setString(parameterIndex:4, cp);
        ps.setString(parameterIndex:5, ville);

        try (ResultSet rs = ps.executeQuery()) {
            if (rs.next()) {
                return rs.getInt(columnIndex:1) > 0; // Retourne true si un doublon existe
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return false;
}
```

### II.III Gestion des erreurs SQL

Chaque requête SQL est entourée d'un try-catch pour éviter un crash en cas d'erreur.

```
try (Connection conn = DatabaseConnection.getConnection();
    Statement stmt = conn.createStatement();
    ResultSet rs = stmt.executeQuery(sql:"SELECT * FROM PROPRIETAIRE")) {

    while (rs.next()) {
        proprietaires.add(new Proprietaire(
            rs.getInt(columnLabel:"id_proprietaire"),
            rs.getString(columnLabel:"nom"),
            rs.getString(columnLabel:"prenom"),
            rs.getString(columnLabel:"adresse"),
            rs.getString(columnLabel:"cp"),
            rs.getString(columnLabel:"ville")));
    }
} catch (SQLException e) {
    e.printStackTrace();
}
```

### II.IV Confirmation avant suppression

Avant de supprimer un propriétaire, on demande confirmation à l'utilisateur.

```
// Afficher une boîte de confirmation
private int showConfirmation(String title, String message) {
    return JOptionPane.showConfirmDialog(parentComponent:null,
        message, title, JOptionPane.YES_NO_OPTION, JOptionPane.WARNING_MESSAGE);
}
```