

---

# Cahier des Charges Techniques

---



---

PETHIYAGODA HIRUSHA  
BTS SIO 2

---

# SOMMAIRE

1.	
Contexte du projet .....	3
1.1 Présentation du projet.....	3
1.2 Date de rendu du projet .....	3
2. Besoins fonctionnels.....	3
3. Ressources nécessaires à la réalisation du projet .....	3
3.1 Ressources matérielle .....	3
3.2 Ressources logicielles .....	4
4. Gestion du projet .....	4
5. Conception du projet.....	5
5.1. Le Front-End.....	5
5.1.1. Wireframes .....	5
5.1.2. Maquettes .....	6
5.1.3. Arborescence .....	7
5.2 Le Back-End.....	8
5.2.1. Diagramme de cas d'utilisation .....	8
5.2.2. Diagramme d'activité .....	9
5.2.3 Dictionnaire des données.....	10
5.2.4. Modèle Conceptuel des Données (MCD).....	11
5.2.5. Modèle Logique des Données (MLD) Version BTS .....	11
5.2.6. Modèle Physique des Données (MPD) .....	12
6. Technologies utilisées.....	13
6.1. Langages de développement Applicatif.....	13
6.2. Base de données .....	13
7. Sécurité.....	13
7.1. Protections utilisées .....	13
7.2. Protection contre les injections SQL.....	13

---

# Contexte du projet

## 1.1 Présentation du projet

Vous occupez actuellement le poste de concepteur et développeur au sein de la Direction des systèmes d'information de la préfecture de votre département. La responsable du service des cartes grises souhaiterait faire évoluer leur application métiers. Cependant, aucun document de conception n'est disponible. Votre travail consiste donc à travailler sur l'élaboration de documents de conception de l'application actuelle en vue de faciliter la réflexion autour de son évolution.

## 1.2 Date de rendu du projet

Le projet doit être rendu au plus tard le 06/03/2025.

# 2. Besoins fonctionnels

Les données seront stockées dans une base de données relationnelle pour faciliter la gestion et la mise à jour des informations.

# 3. Ressources nécessaires à la réalisation du projet

## 3.1 Ressources matérielle

Les ressources matérielles dont nous avons besoins sont :

- PC Fixe
- Connexion Internet (par câble ou wifi)
- Ecran
- Clavier
- Souris
- Cahier de brouillon
- Stylo

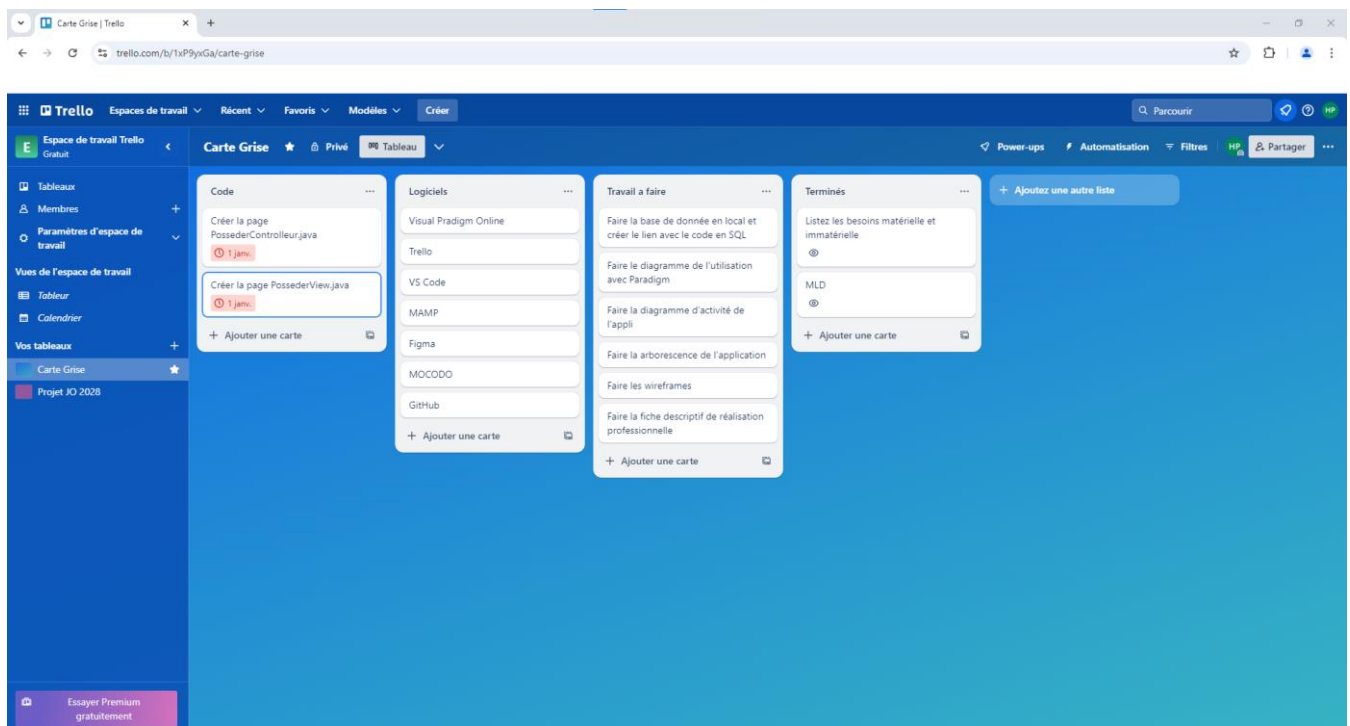
## 3.2 Ressources logicielles

Les ressources logicielles dont nous avons besoins sont :

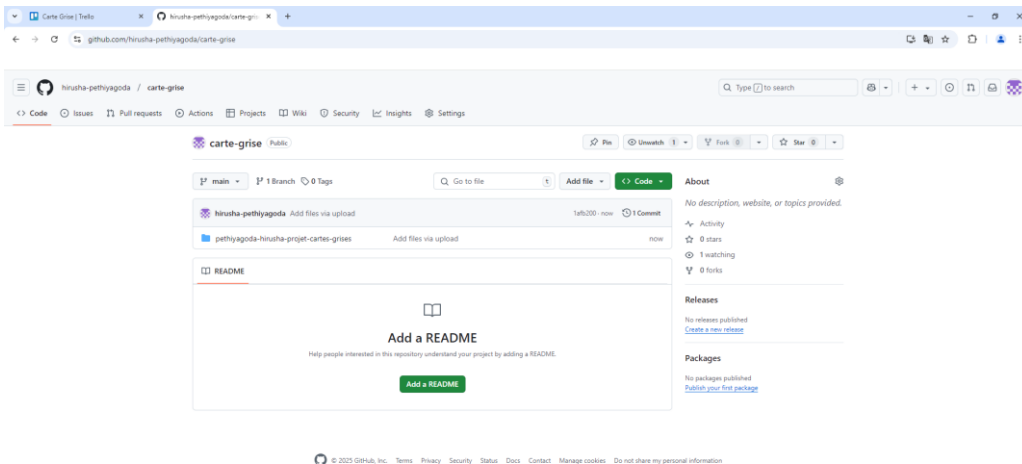
- IDE (Environnement de Développement) : Visual Studio Code
- Plateforme de développement collaboratif : GitHub
- Conception de Base des Données : MOCODO
- Outil de gestion des projets : Trello
- Conception UML et arborescence : Visual Paradigm Online
- Maquettage : Figma
- MAMP
  - Serveur Web (Contenu dans MAMP) : Apache
  - SGBDR (Contenu dans MAMP) : MySQL

## 4. Gestion du projet

Pour réaliser le projet, nous utiliserons la méthode Agile Kanban. Nous utiliserons également l'outil de gestion de projet en ligne Trello.



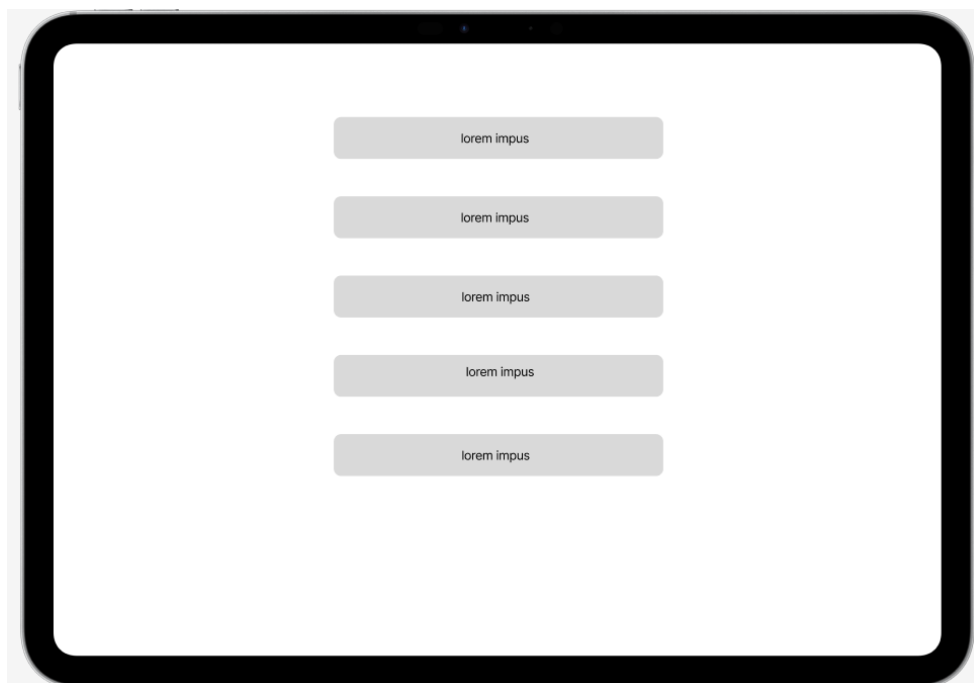
Nous travaillons également sur GitHub, plateforme de développement collaboratif.

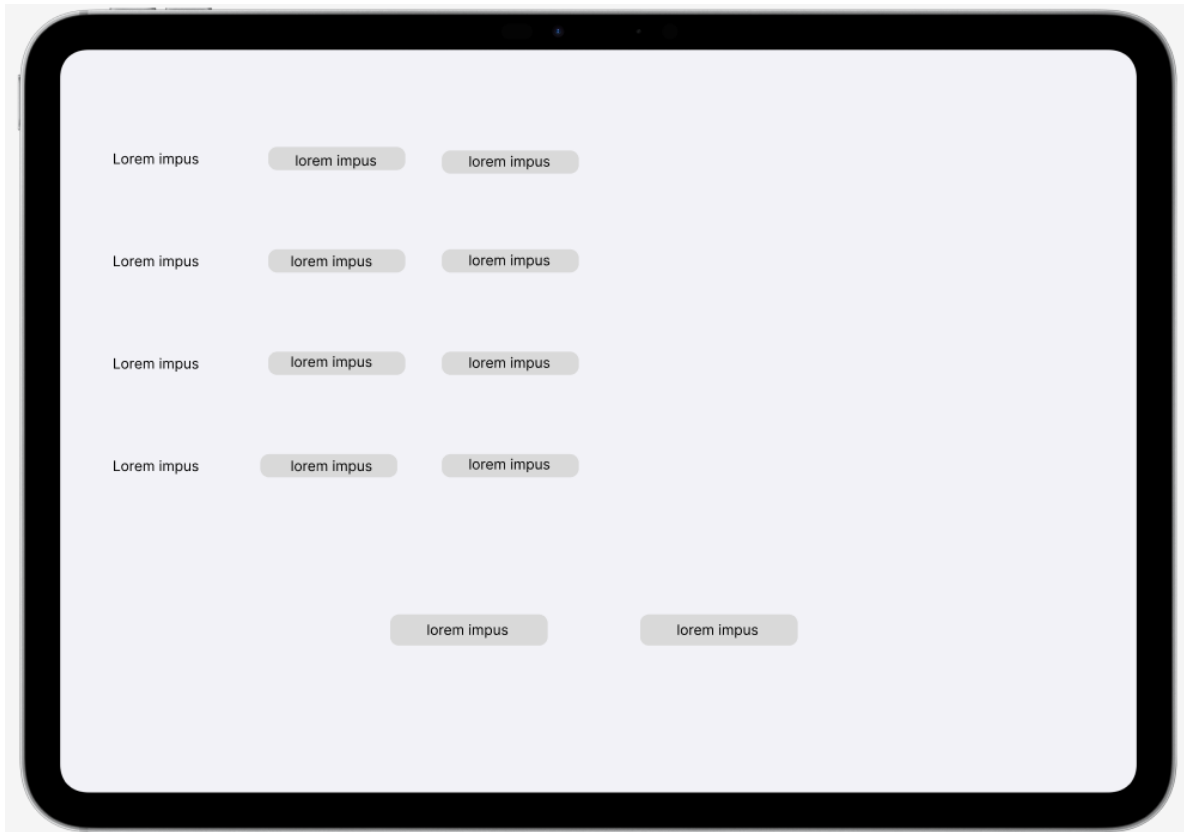


## 5. Conception du projet

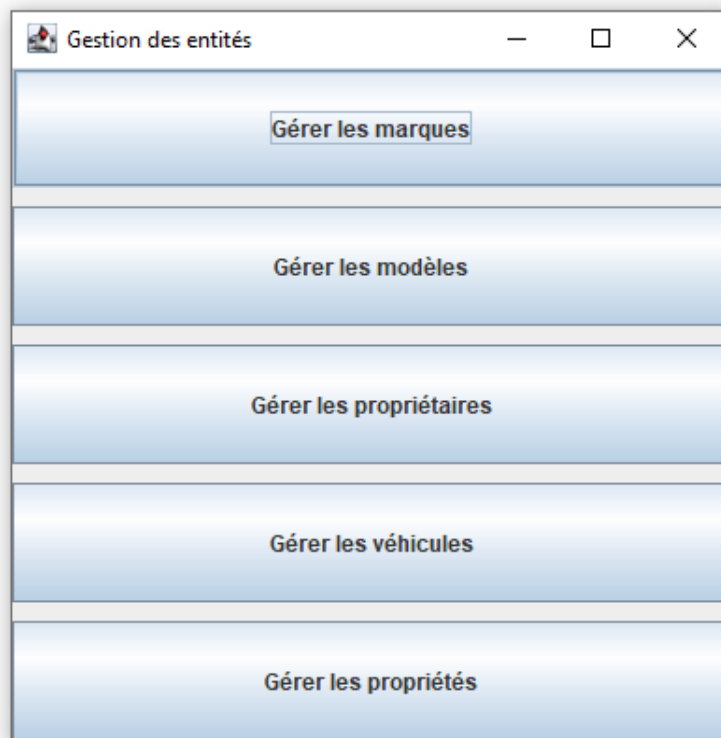
### 5.1. Le Front-End

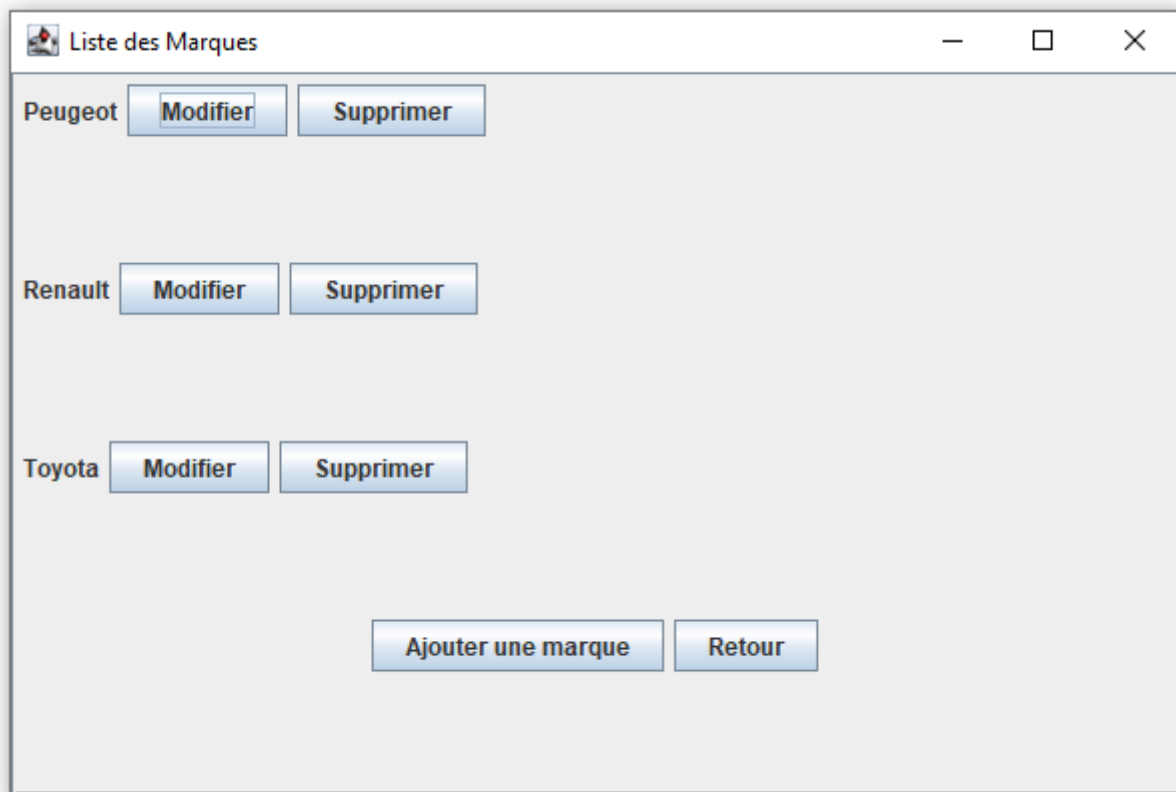
#### 5.1.1. Wireframes



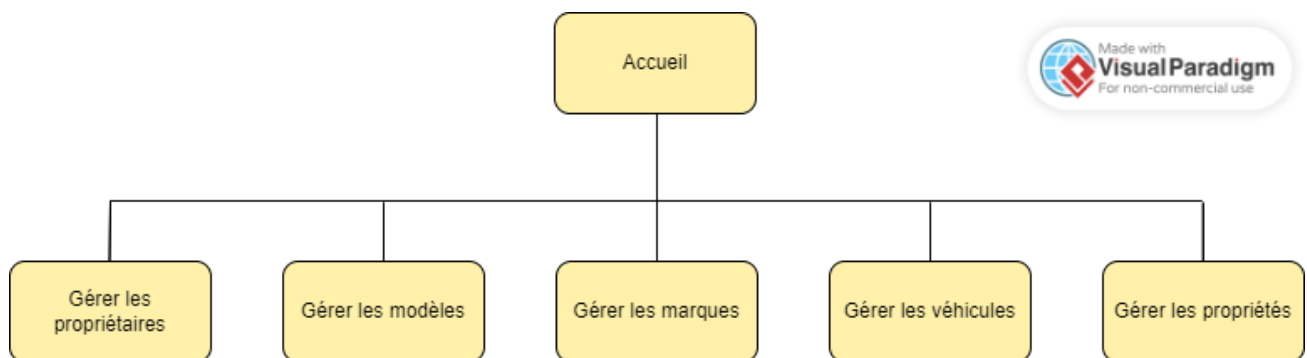


### 5.1.2. Maquettes



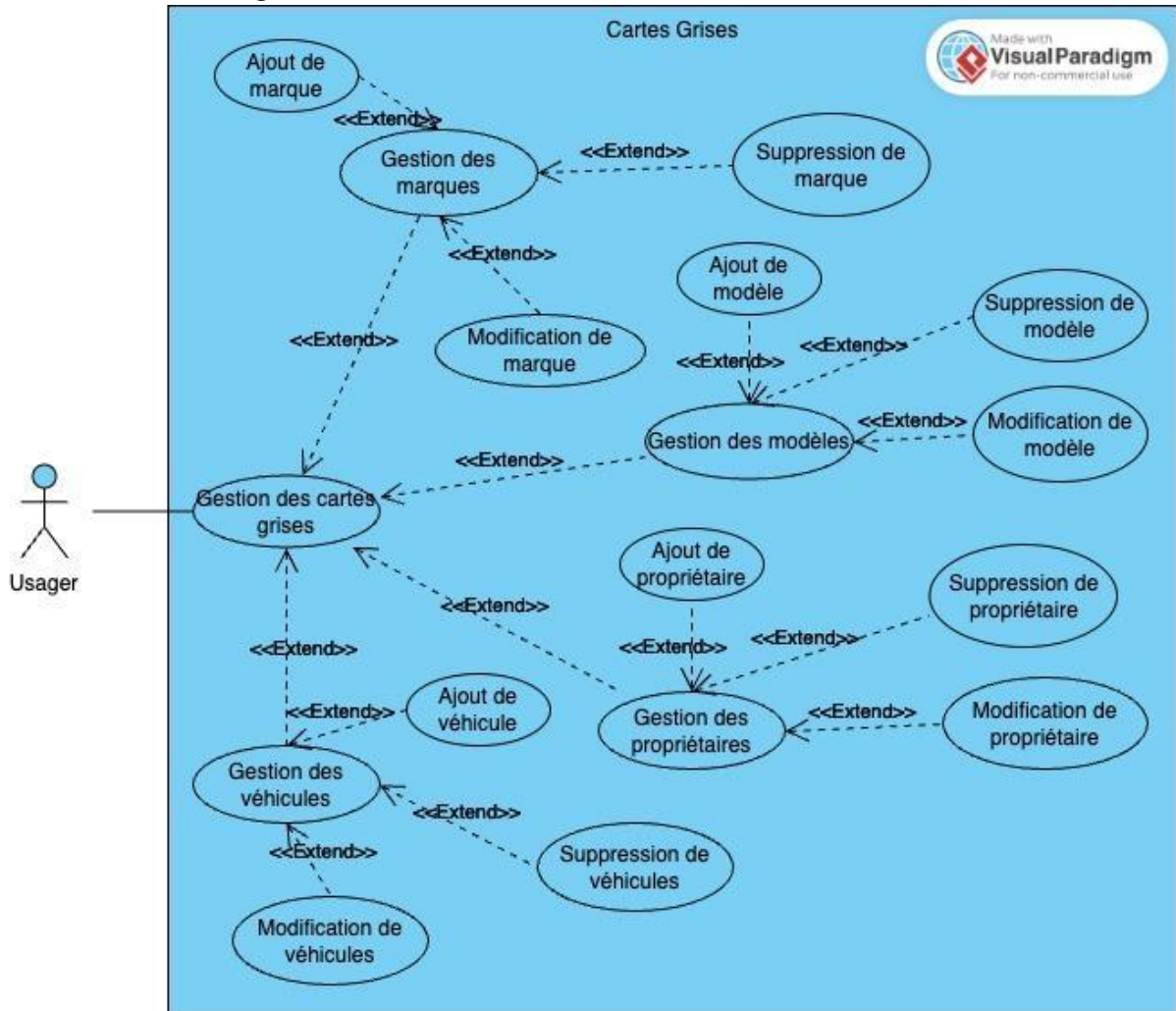


### 5.1.3. Arborescence



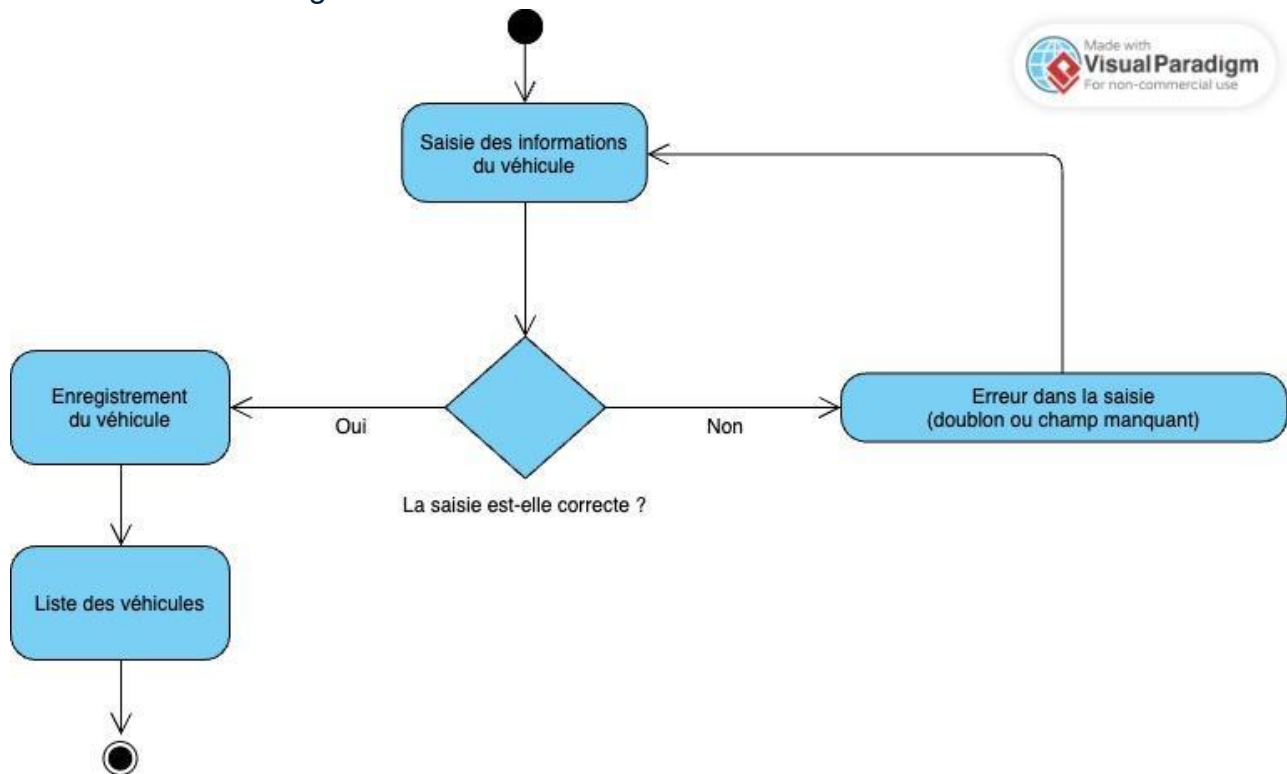
## 5.2 Le Back-End

### 5.2.1. Diagramme de cas d'utilisation





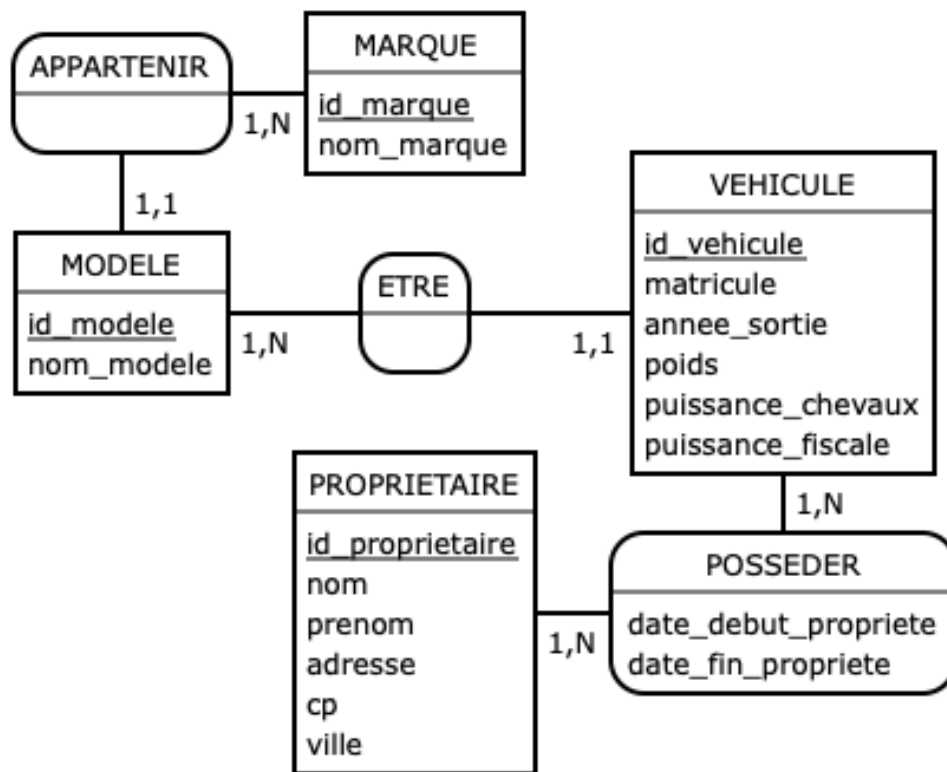
### 5.2.2. Diagramme d'activité



### 5.2.3 Dictionnaire des données

Table	Attribut	Type	Description
Marque	id_marque	INTEGER(4)	Identifiant unique de la marque
Marque	nom_marque	VARCHAR(255)	Nom de la marque
Modele	id_modele	INTEGER(4)	Identifiant unique du modèle
Modele	nom_modele	VARCHAR(255)	Nom du modèle
Modele	id_marque	INTEGER(4)	Identifiant unique de la marque (Clé étrangère)
Vehicule	id_vehicule	INTEGER(4)	Identifiant unique du véhicule
Vehicule	matricule	VARCHAR(50)	Immatriculation du véhicule
Vehicule	annee_sortie	YEAR(4)	Année de sortie du véhicule
Vehicule	poids	INTEGER(11)	Poids du véhicule
Vehicule	puissance_chevaux	INTEGER(11)	Puissance chevaux du véhicule
Vehicule	puissance_fiscale	INTEGER(11)	Puissance fiscale du véhicule
Vehicule	id_modele	INTEGER(4)	Identifiant unique du modèle (Clé étrangère)
Proprietaire	id_proprietaire	INTEGER(4)	Identifiant unique du propriétaire
Proprietaire	nom	VARCHAR(255)	Nom du propriétaire
Proprietaire	prenom	VARCHAR(255)	Prénom du propriétaire
Proprietaire	adresse	VARCHAR(255)	Adresse du propriétaire
Proprietaire	cp	VARCHAR(10)	Code Postale du propriétaire
Proprietaire	ville	VARCHAR(255)	Ville du propriétaire
Posseder	id_vehicule	INTEGER(4)	Identifiant unique du véhicule (Clé étrangère)
Posseder	id_proprietaire	INTEGER(4)	Identifiant unique du propriétaire (Clé étrangère)
Posseder	date_debut_propriete	DATE	Date début du propriété
Posseder	date_fin_propriete	DATE	Date fin du propriété

#### 5.2.4. Modèle Conceptuel des Données (MCD)



#### 5.2.5. Modèle Logique des Données (MLD) Version BTS

**Marque** (id\_marque, nom\_marque)

Clé primaire : id\_marque

**Modele** (id\_modele, nom\_modele, id\_marque)

Clé primaire : id\_modele

Clé étrangère : id\_marque en reference à id\_marque de MARQUE

**Vehicule** (id\_vehicule, matricule, annee\_sortie, poids, puissance\_chevaux, puissance\_fiscale, id\_modele)

Clé primaire : id\_vehicule

Clé étrangère : id\_modele en reference à id\_modele de MODELE

**Proprietaire** (id\_proprietaire, nom, prenom, adresse, cp, ville)

Clé primaire : id\_proprietaire

**Posseder** (id\_vehicule, id\_proprietaire, date\_debut\_propriete, date\_fin\_propriete)

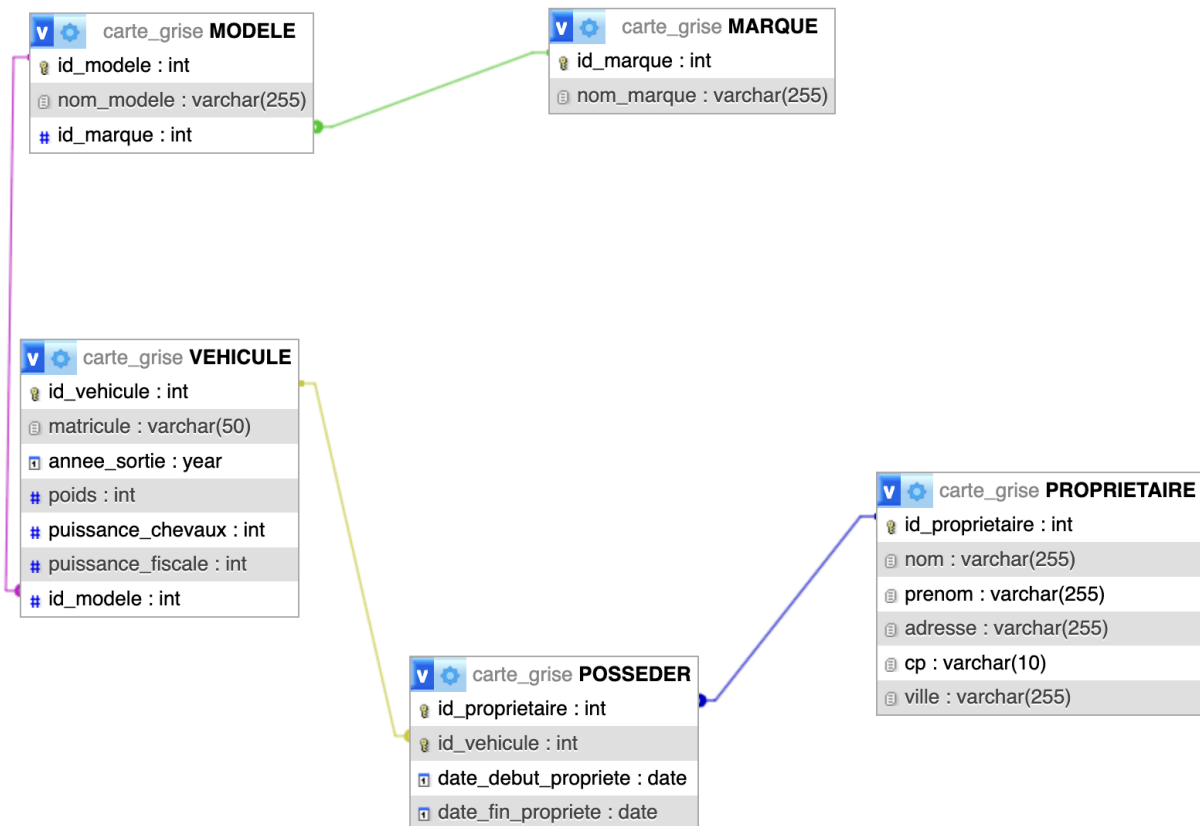
Clé primaire : id\_proprietaire en reference à id\_proprietaire de PROPRIETAIRE

id\_vehicule en reference à id\_vehicule de VEHICULE

Clé étrangère : id\_proprietaire en reference à id\_proprietaire de PROPRIETAIRE

id\_vehicule en reference à id\_vehicule de VEHICULE

### 5.2.6. Modèle Physique des Données (MPD)



---

## 6. Technologies utilisées

### 6.1. Langages de développement Applicatif

Afin de réaliser le site web nous allons utiliser des langages de programmations suivants :

- Java

### 6.2. Base de données

Nous allons utiliser le langage de programmation SQL afin de créer et gérer notre base de données en passant par le Système de Gestion de Base de Données Relationnel (MySQL) de MAMP.

## 7. Sécurité

### 7.1. Protections utilisées

- Utilisation des requêtes préparées (PreparedStatement) afin d'empêcher l'exécution de commandes SQL malveillantes.
- Vérification des doublons afin de garantir l'intégrité des données en évitant les doublons.

### 7.2. Protection contre les injections SQL

Une injection SQL, parfois abrégée en SQLi, est un type de vulnérabilité dans lequel un pirate utilise un morceau de code SQL (« Structured Query Language », langage de requête structuré) pour manipuler une base de données et accéder à des informations potentiellement importantes. C'est l'un des types d'attaques les plus répandus et menaçants, car il peut potentiellement être utilisé pour nuire à n'importe quelle application Web ou n'importe quel site Web qui utilise une base de données SQL (soit la plupart).

Pour se protéger des injections SQL, nous allons donc utiliser les requêtes préparées **PreparedStatement** , **try-catch**.

```

public void addMarque(String nomMarque) {
    try (Connection conn = DatabaseConnection.getConnection()) {
        if (existsByNomMarque(conn, nomMarque)) {
            showAlert(title:"Erreur", "La marque '" + nomMarque + "' existe déjà !");
            return;
        }
        try (PreparedStatement ps = conn.prepareStatement(sql:"INSERT INTO MARQUE (nom_marque) VALUES (?)")) {
            ps.setString(parameterIndex:1, nomMarque);
            ps.executeUpdate();
            showAlert(title:"Succès", "La marque '" + nomMarque + "' a été ajoutée avec succès !");
        }
    } catch (SQLException e) {
        e.printStackTrace();
        showAlert(title:"Erreur", message:"Une erreur s'est produite lors de l'ajout de la marque.");
    }
}

public void updateMarque(int idMarque, String newNom) {
    try (Connection conn = DatabaseConnection.getConnection()) {
        if (existsByNomMarqueExcludingId(conn, newNom, idMarque)) {
            showAlert(title:"Erreur", "Une autre marque porte déjà le nom '" + newNom + "' !");
            return;
        }
        try (PreparedStatement ps = conn.prepareStatement(sql:"UPDATE MARQUE SET nom_marque = ? WHERE id_marque = ?")) {
            ps.setString(parameterIndex:1, newNom);
            ps.setInt(parameterIndex:2, idMarque);
            ps.executeUpdate();
            showAlert(title:"Succès", message:"La marque a été mise à jour avec succès !");
        }
    } catch (SQLException e) {
        e.printStackTrace();
        showAlert(title:"Erreur", message:"Une erreur s'est produite lors de la mise à jour de la marque.");
    }
}
}

```