Cahier des charges techniques

Sommaire

1. Contexte du projet
    1.1. Présentation du projet
    1.2. Date de rendu du projet
2. Besoins fonctionnels
3. Ressources nécessaires à la réalisation du projet
    3.1. Ressources matérielles
    3.2. Ressources logicielles
4. Gestion du projet
5. Conception du projet
    5.1. Le front-end
        5.1.1. Wireframes
        5.1.2. Maquettes
        5.1.3. Arborescences
    5.2. Le back-end
        5.2.1. Diagramme de cas d'utilisation
        5.2.2. Diagramme d'activités
        5.2.3. Modèles Conceptuel de Données (MCD)
        5.2.4. Modèle Logique de Données (MLD)
        5.2.5. Modèle Physique de Données (MPD)
6. Technologies utilisées
    6.1. Langages de développement Web
    6.2. Base de données
7. Sécurité
    7.1. Login et protection des pages administrateurs
    7.2. Hachages des mots de passe avec Bcrypt
    7.3. Protection contre les attaques XSS (Cross-Site Scripting)
    7.4. Protection contre les injections SQL

1. Contexte du projet

1.1. Présentation du projet

Votre agence web a été sélectionnée par le comité d’organisation des jeux olympiques Je travaille pour la Direction des Systèmes d'Information (DSI) de la Préfecture. Le service des immatriculations souhaite moderniser sa gestion des cartes grises. Ma mission est de reprendre l'application existante pour y ajouter un module de gestion de l'historique des véhicules (qui a possédé quel véhicule et quand), tout en sécurisant les données

Votre équipe et vous-même avez pour mission de proposer une solution qui répondra à la demande du client.

1.2. Date de rendu du projet

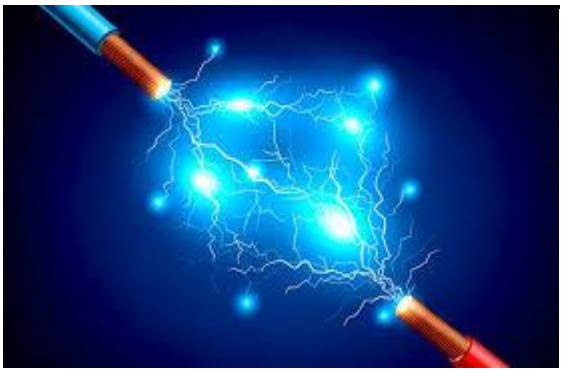
Le projet doit être rendu au plus tard le 29/12/2025

2. Besoins fonctionnels

L'application est un **logiciel de bureau (Client Lourd)** développé en Java. Elle est destinée à un usage interne par les agents de la préfecture (Intranet). Elle doit permettre de gérer les propriétaires, les véhicules et leurs associations (historique de possession) via une interface graphique sécurisée.

3. Ressources nécessaires à la réalisation du projet

3.1. Ressources matérielles



3.2. Ressources logicielles

Visual Studio Code

Visual Paradigm Online

MySQL®

php

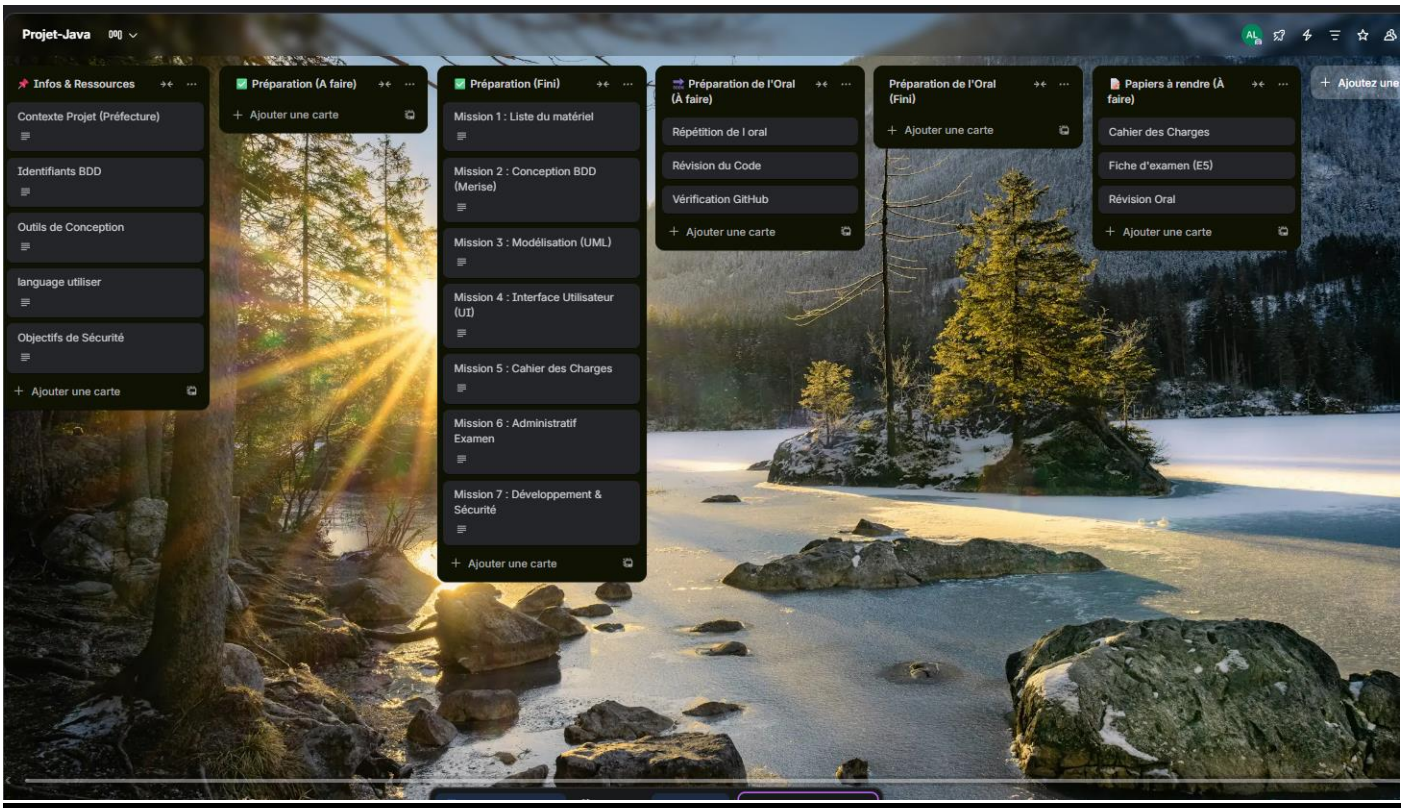
MAMP

MOCODO

Modélisation Conceptuelle de Données. Nickel. Ni souris.

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.

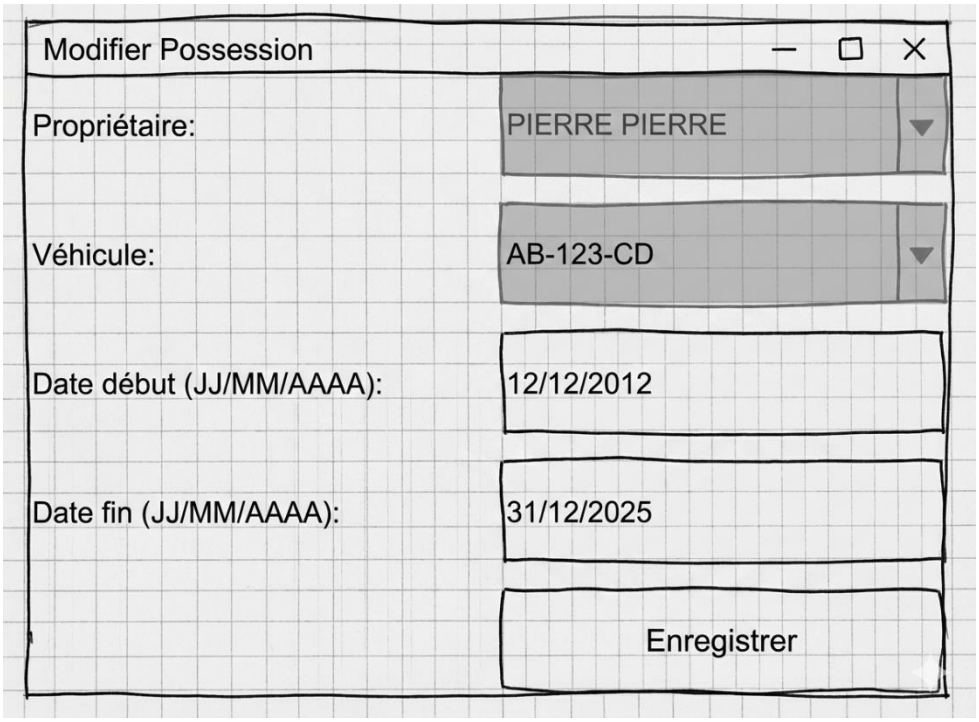


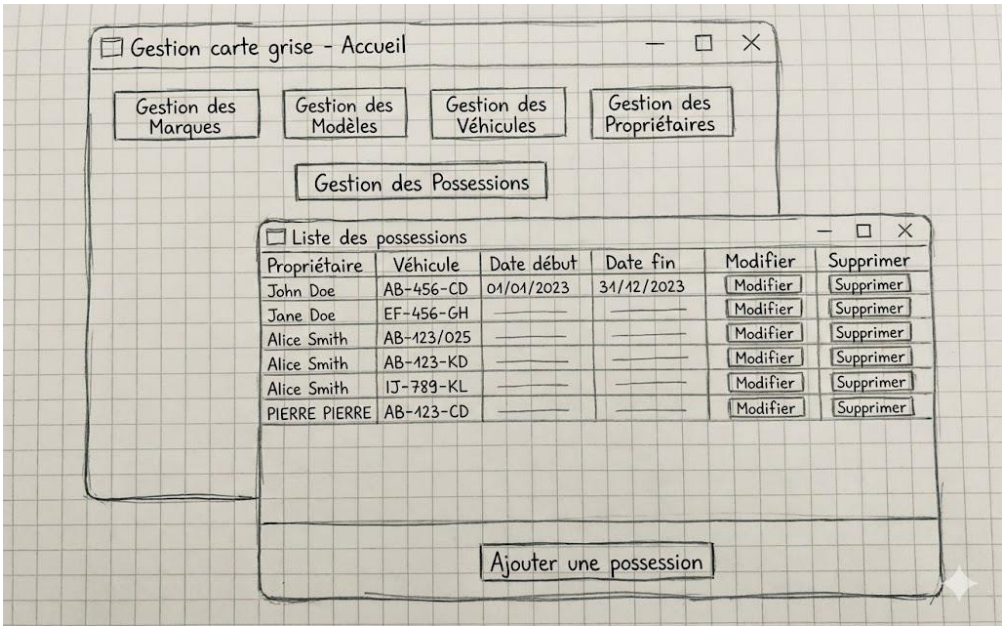
Nous travaillons également sur GitHub, plateforme de développement collaboratif.

5. Conception du projet

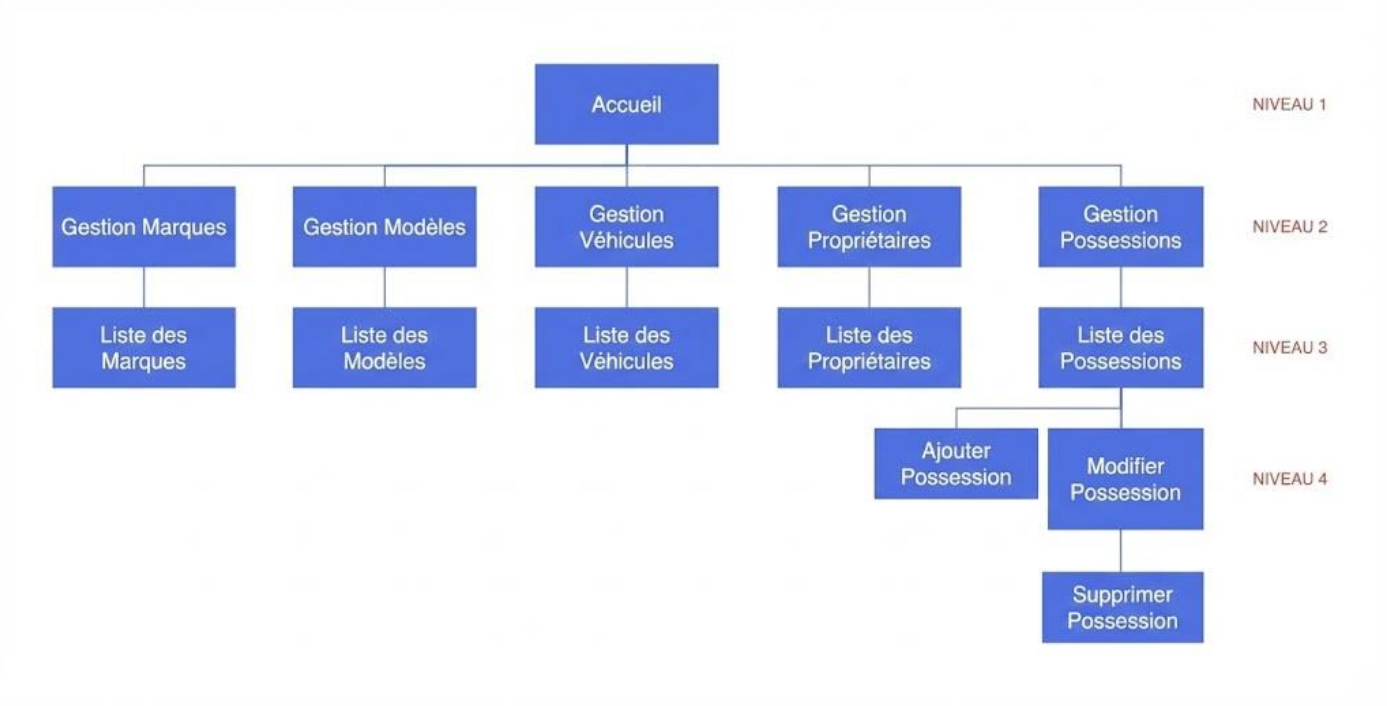
5.1. Le front-end

5.1.1. Maquette



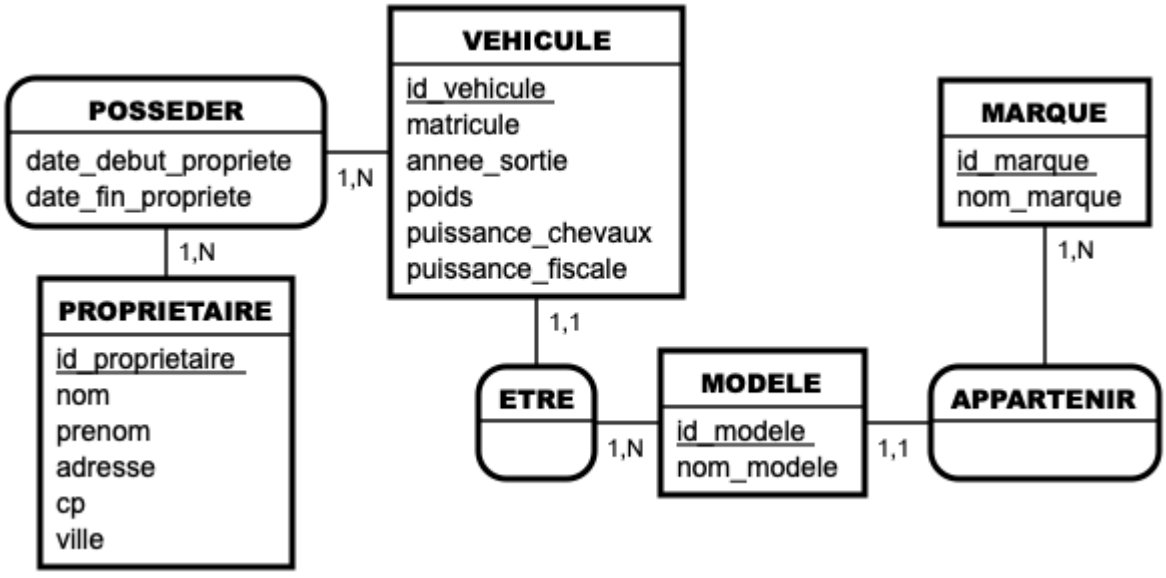


5.1.3. Arborescences

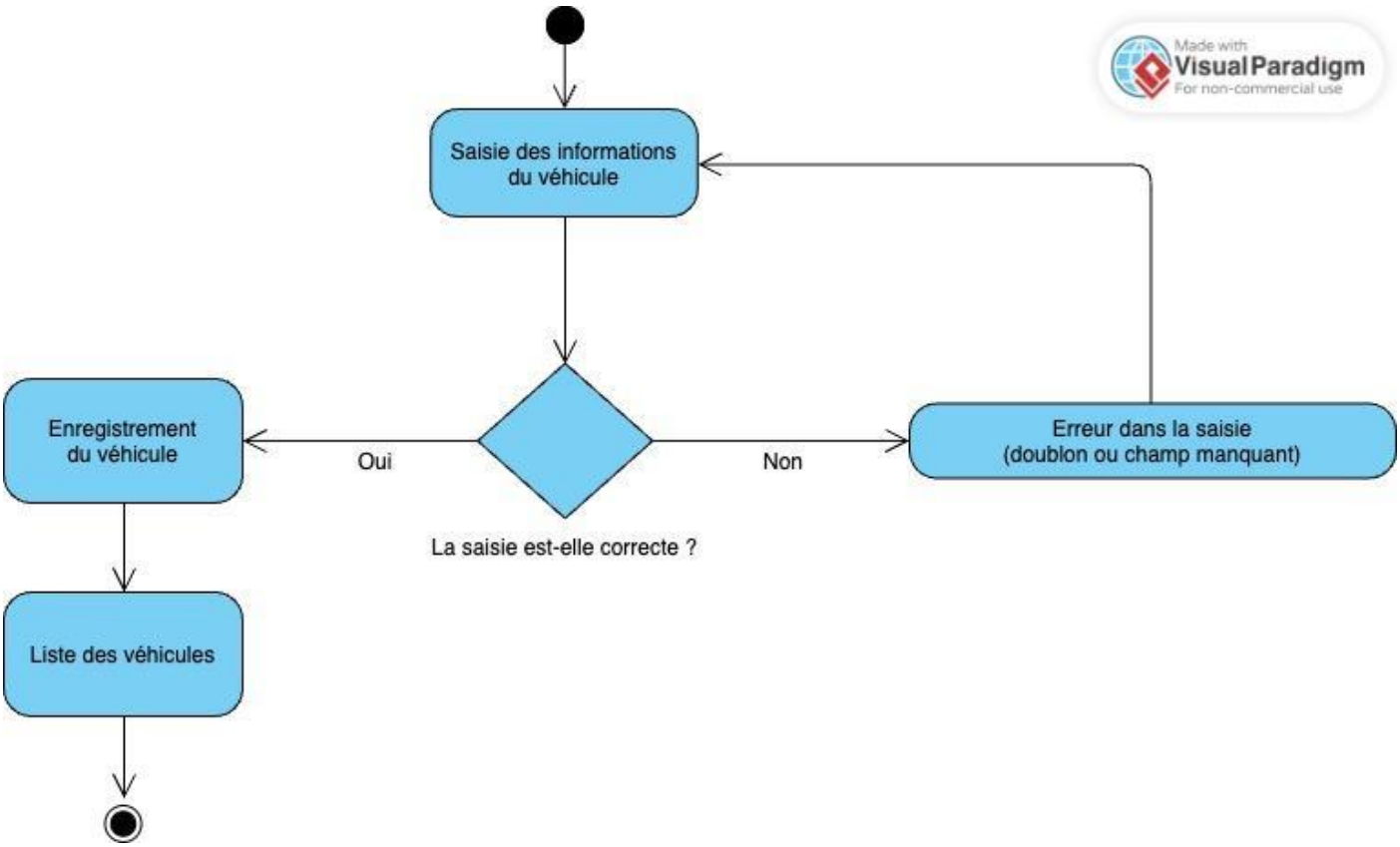


5.2. Le back-end

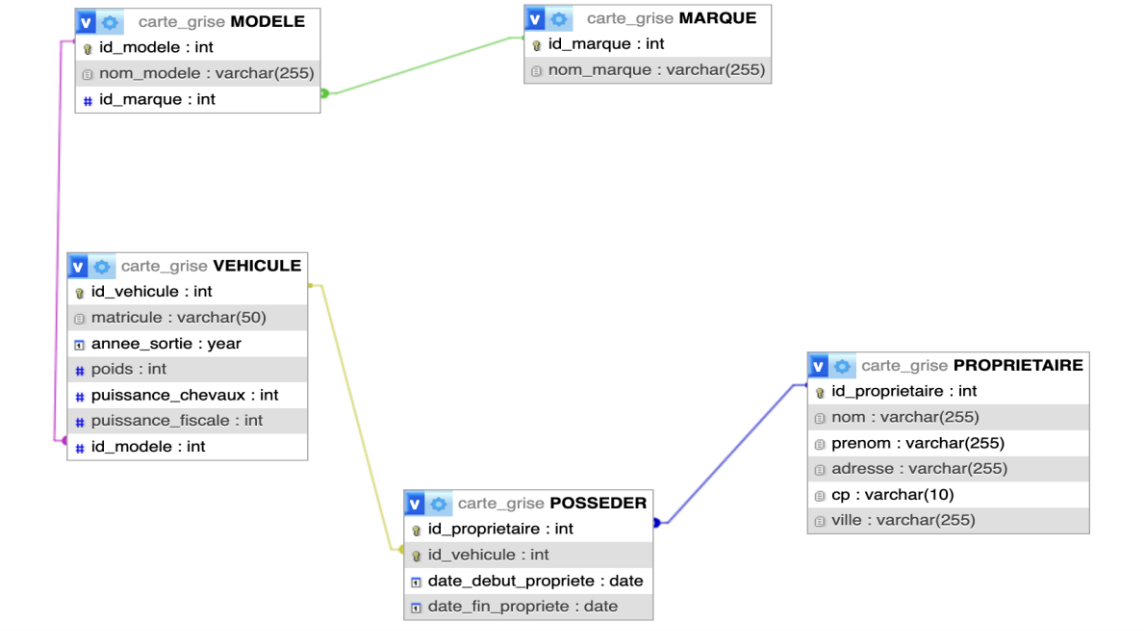
5.2.1. Diagramme de cas d'utilisation



5.2.2. Diagramme d’activités



5.2.3. Modèles Conceptuel de Données (MCD)



5.2.4. Modèle Logique de Données (MLD)

PROPRIETAIRE (<ins>id_proprietaire</ins>, nom, prenom, adresse, cp, ville)

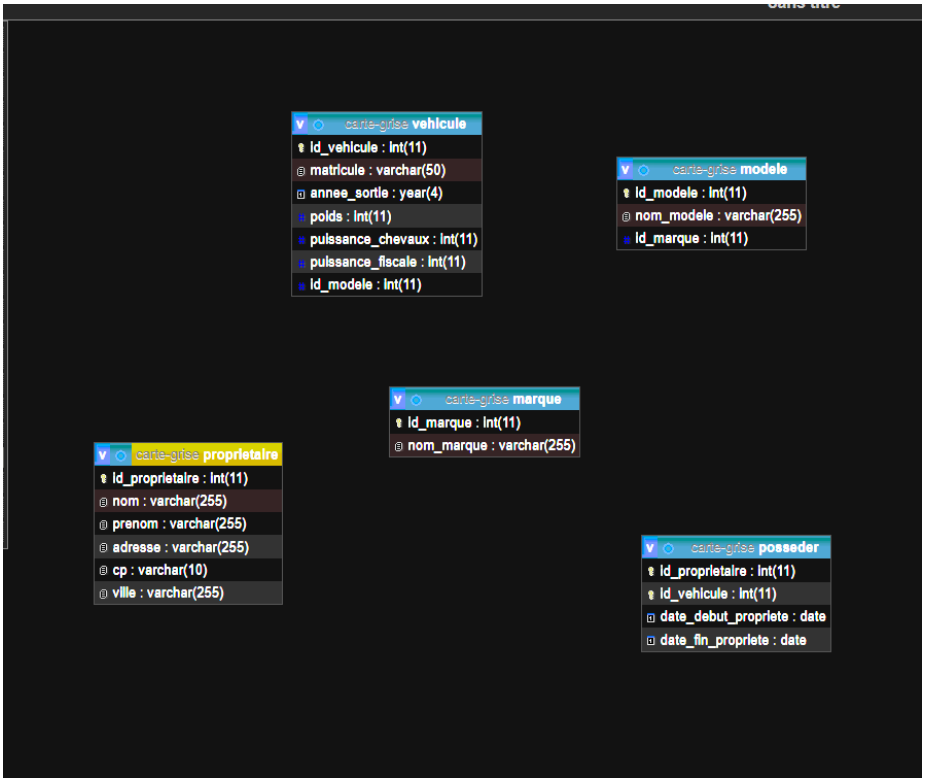
MARQUE (<ins>id_marque</ins>, nom_marque)

MODELE (<ins>id_modele</ins>, nom_modele, #id_marque)

VEHICULE (<ins>id_vehicule</ins>, matricule, annee_sortie, poids, puissance_chevaux, puissance_fiscale, #id_modele)

POSSEDER (<ins>#id_proprietaire, #id_vehicule</ins>, date_debut_propriete, date_fin_propriete)

5.2.5. Modèle Physique de Données (MPD)



6. Technologies utilisées

6.1. Langages de développement Web

6.2. Base de données

7. Sécurité

7.1. Login et protection des pages administrateurs

L'authentification vérifie l'identité de l'utilisateur (généralement via un identifiant et un mot de passe).

L'autorisation (ou protection des pages) vérifie si cet utilisateur a le droit d'accéder à une zone spécifique (ex : seul le rôle "Admin" peut voir la page de suppression). Si l'utilisateur n'est pas connecté, il est redirigé vers la page de login.

L'application actuelle est conçue pour un usage interne sécurisé (Intranet Préfecture), l'accès n'est donc pas restreint par un login dans cette version 1. Pour la version 2, une fenêtre d'authentification sera mise en place pour vérifier les droits de l'agent avant d'ouvrir le menu principal.

7.2. Hachages des mots de passe avec Bcrypt

Le hachage est une technique de cryptographie qui transforme le mot de passe en clair (ex: "loulou12") en une suite de caractères illisible et unique (le "hash"). C'est irréversible : on ne peut pas retrouver le mot de passe original à partir du hash. Bcrypt est un algorithme spécifique de hachage, très lent et sécurisé, conçu pour empêcher les pirates de deviner les mots de passe par force brute même s'ils volent la base de données.

Non applicable pour le moment car l'application ne stocke pas encore d'utilisateurs. Dans le futur, nous utiliserons la bibliothèque **BCrypt for Java** (org.mindrot.jbcrypt) pour hacher les mots de passe avant l'insertion en base de données, afin qu'ils ne soient jamais lisibles en clair.

7.3. Protection contre les attaques XSS (Cross-Site Scripting)

La faille XSS est une vulnérabilité qui touche surtout les sites Web. Elle permet à un pirate d'injecter du code malveillant (souvent du JavaScript) dans une page vue par d'autres utilisateurs. Cela arrive quand l'application affiche ce que l'utilisateur a tapé sans le "nettoyer". Le pirate peut s'en servir pour voler des cookies de session ou rediriger l'utilisateur vers un site piégé.

Cette faille concerne principalement les navigateurs Web (exécution de scripts JavaScript malveillants). Notre application étant un **logiciel de bureau (Java Swing)** et non un site web,

elle n'est pas sensible aux failles XSS classiques car elle n'interprète pas le code HTML/JS dans ses champs de saisie

7.4. Protection contre les injections SQL

L'injection SQL est une technique de piratage où l'attaquant insère des morceaux de code SQL dans un formulaire de saisie (comme un champ de recherche ou de login). Si le code n'est pas protégé, la base de données exécute ce code malveillant. Cela permet au pirate de contourner l'authentification, de voler des données sensibles, ou même de supprimer toutes les tables de la base.

Pour me protéger, j'ai systématiquement utilisé des **Requêtes Préparées** au lieu de concaténer les chaînes de caractères. Cette méthode force Java à traiter les données saisies comme du simple texte et non comme du code exécutable.