Cahier des charges techniques

Projet Carte Grise

Rédigé par De Oliveira Lucas

Sommaire

1. Contexte du projet

1.1. Présentation du projet

Votre agence web a été sélectionnée par le comité d'organisation des jeux olympiques de Los Angeles 2028 pour développer une application web permettant aux organisateurs, aux médias et aux spectateurs de consulter des informations sur les sports, les calendriers des épreuves et les résultats des JO 2028.

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 05/12/2024.

2. Besoins fonctionnels

Le site web devra avoir une partie accessible au public et une partie privée permettant de gérer les données.

Les données seront stockées dans une base de données relationnelle pour faciliter la gestion et la mise à jour des informations. Ces données peuvent être gérées directement via le site web à travers un espace administrateur.

3. Ressources nécessaires à la réalisation du projet

3.1. Ressources matérielles

- Ordinateur portable ou fixe : clavier, souris, écran, unité centrale.
- Connexion Internet : par câble ou Wi-Fi.

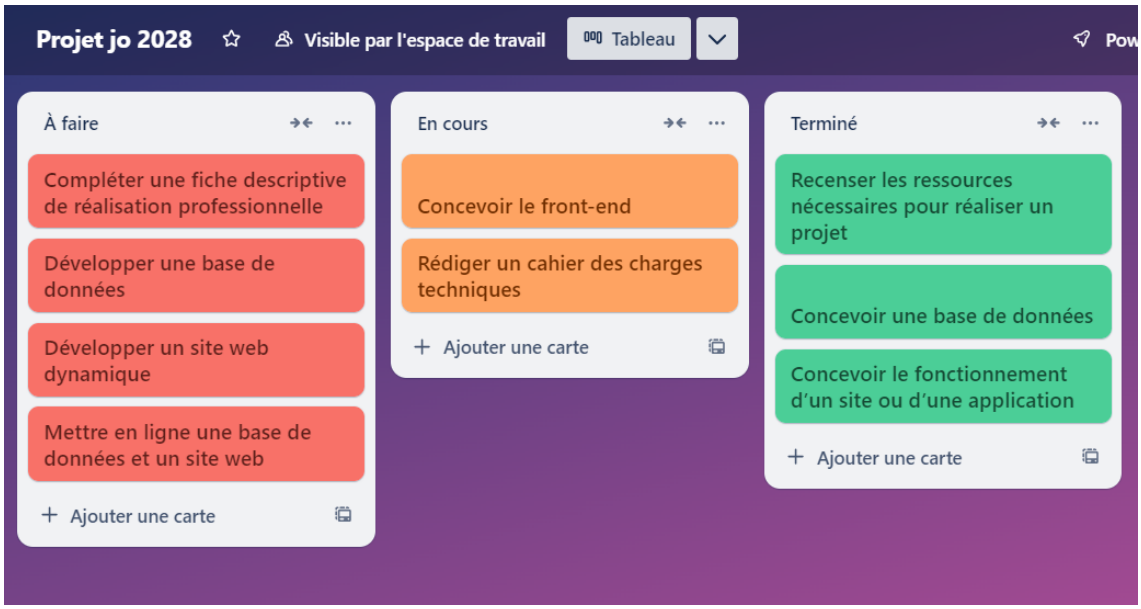
3.2. Ressources logicielles

- IDE : Visual Studio Code
- Plateforme de développement collaboratif : GitHub
- Serveur web : Apache (contenu dans MAMP)
- Système de Gestion de Base de Données relationnelle : MySQL (contenu dans MAMP)
- Outil de gestion de projet : Trello
- Conception UML et Arborescence : Visual Paradigm Online
- Maquettage : Figma

- Conception de la base de données : Mocodo

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.



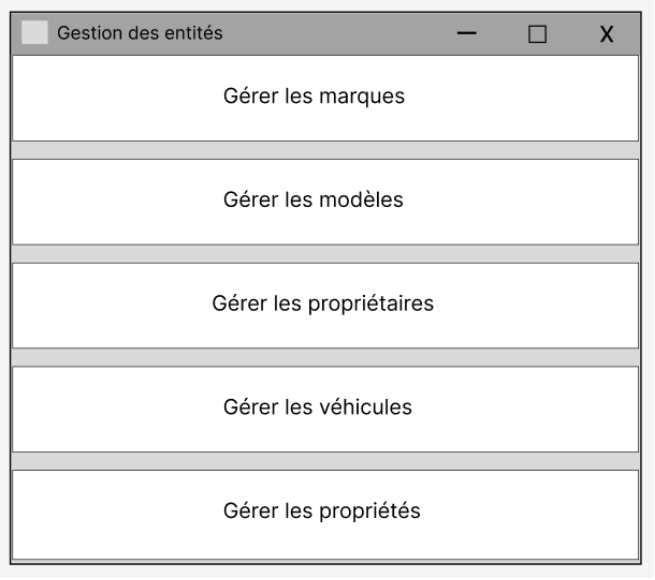
Nous travaillons également sur GitHub, plateforme de développement collaboratif.

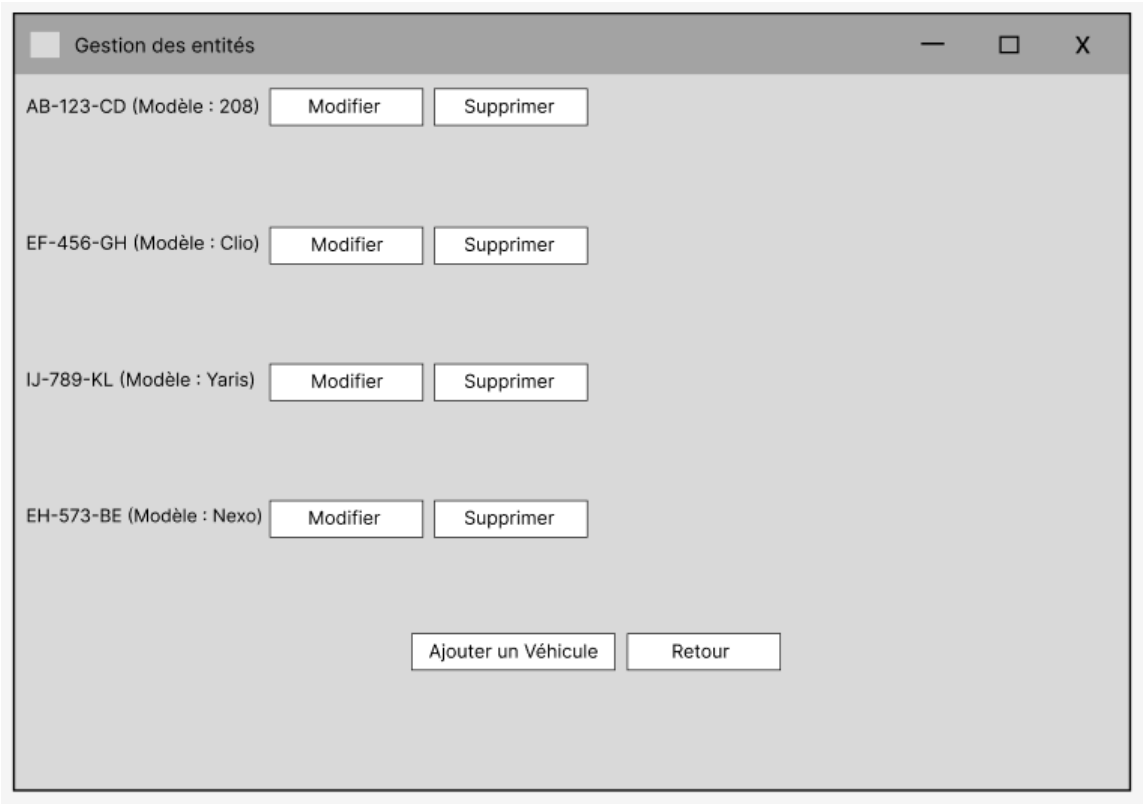
5. Conception du projet

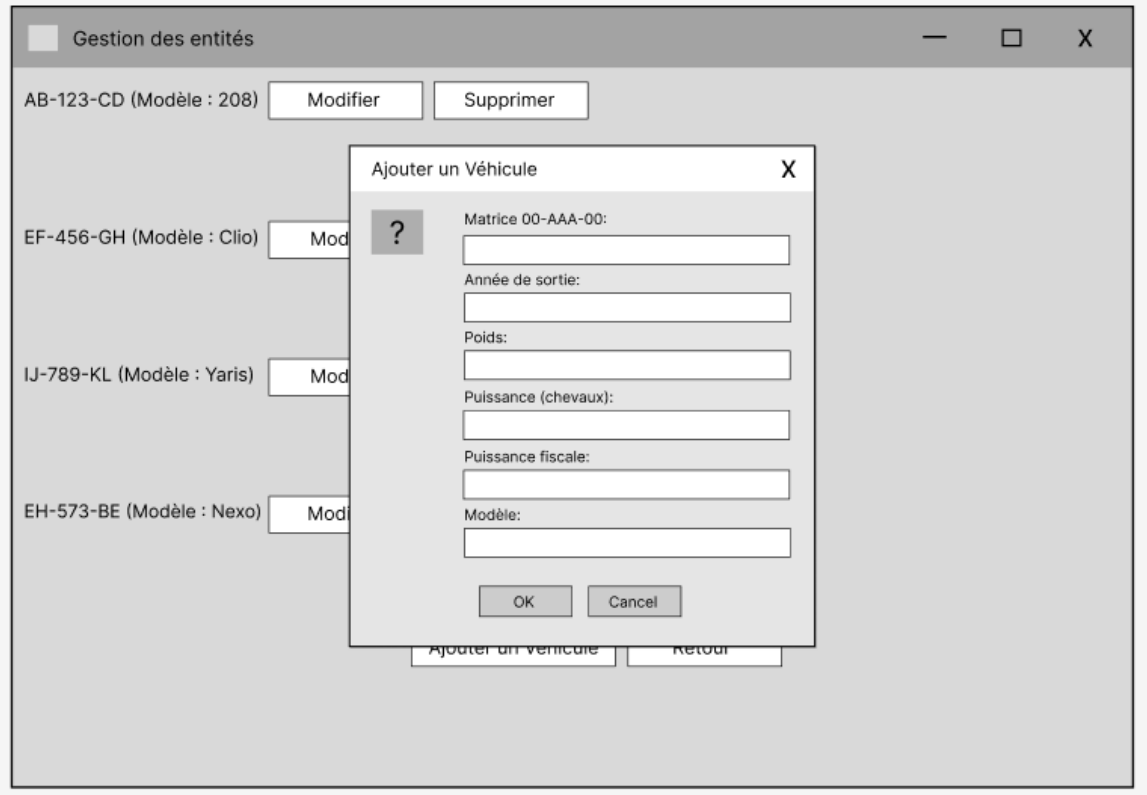
5.1. Le front-end

5.1.1. Wireframes

Sur ordinateur

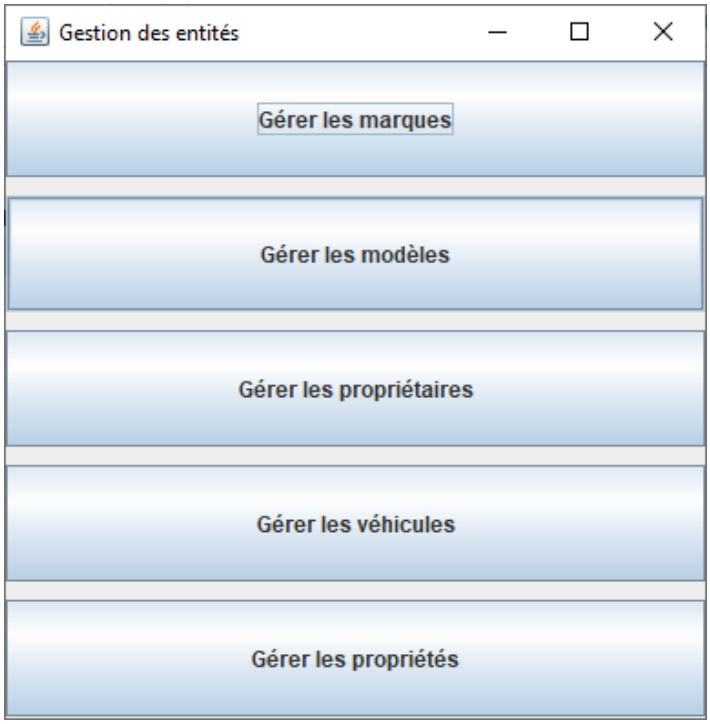


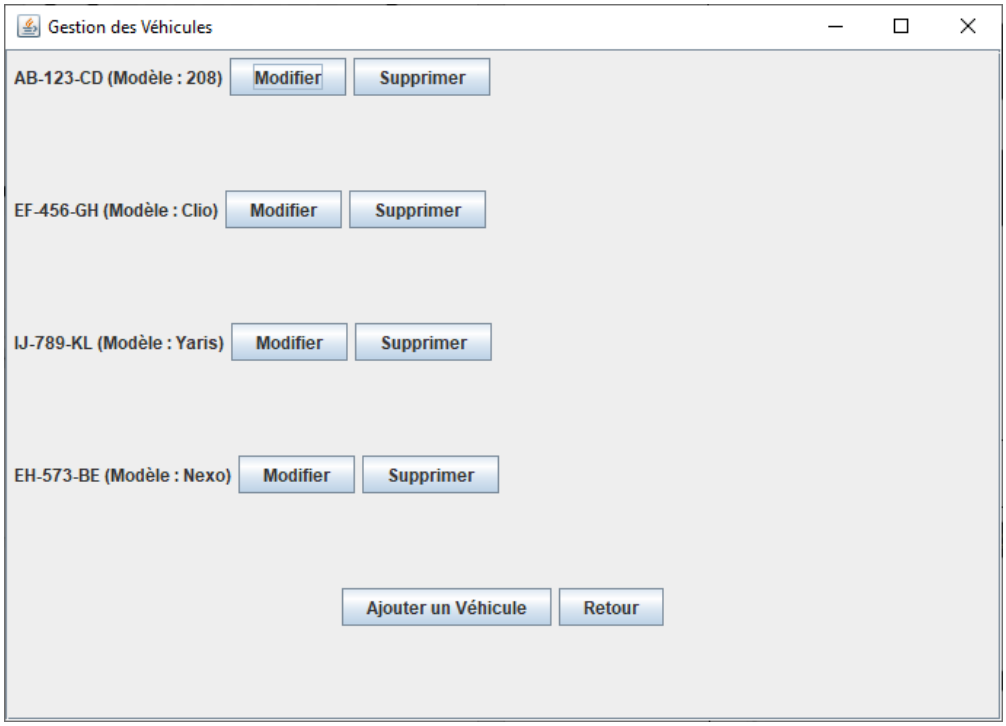

Gestion des entités
AB-123-CD (Modèle : 208)
Modifier
Supprimer
EF-456-GH (Modèle : Clio)
Modifier
Supprimer
IJ-789-KL (Modèle : Yaris)
Modifier
Supprimer
EH-573-BE (Modèle : Nexo)
Modifier
Supprimer
Ajouter un Véhicule
Retour

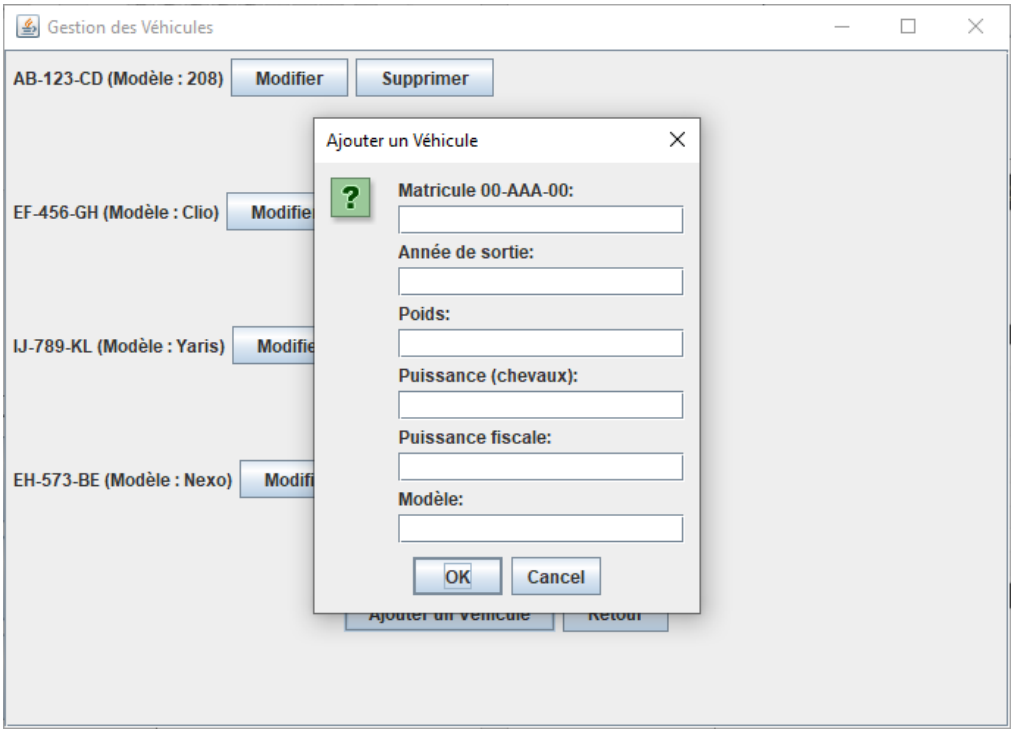

Gestion des entités
AB-123-CD (Modèle : 208)
Modifier
Supprimer
EF-456-GH (Modèle : Clio)
IJ-789-KL (Modèle : Yaris)
EH-573-BE (Modèle : Nexo)
Ajouter un Véhicule
?
Matrice 00-AAA-00:
Année de sortie:
Poids:
Puissance (chevaux):
Puissance fiscale:
Modèle:
OK
Cancel

5.1.2. Maquettes

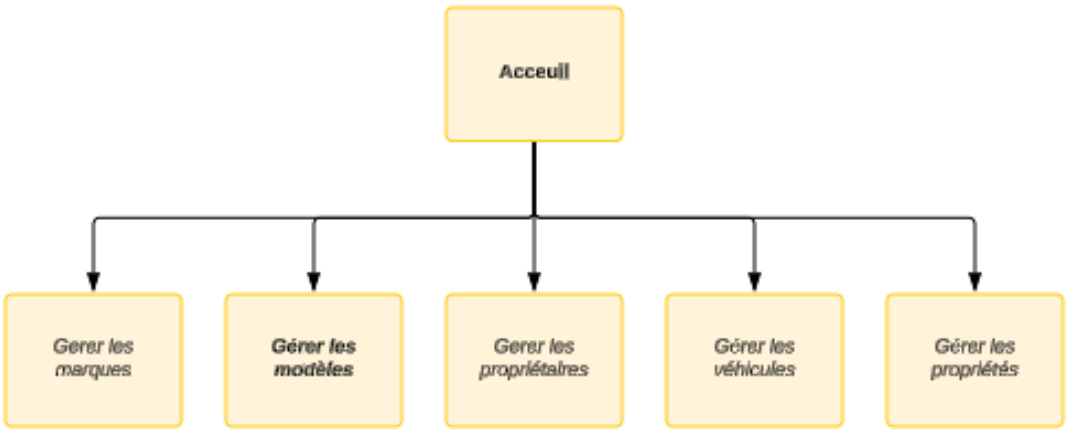
Sur ordinateur





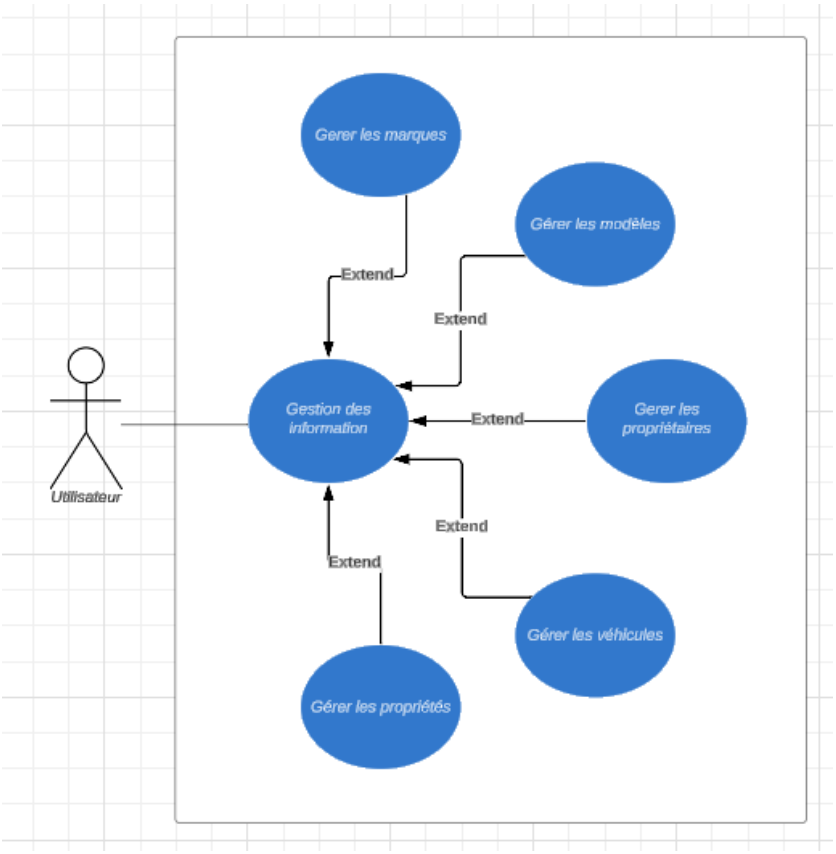


5.1.3. Arborescences

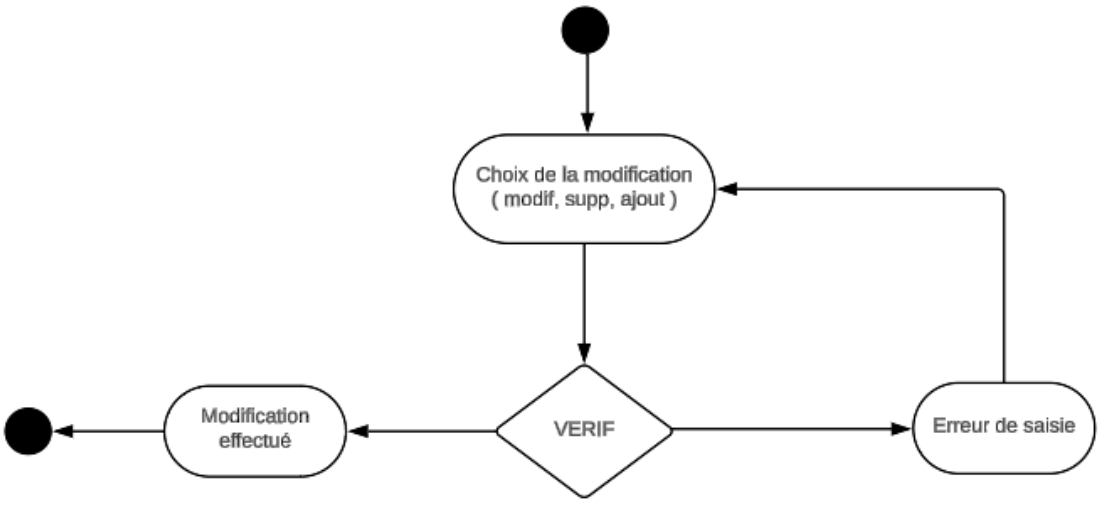


5.2. Le back-end

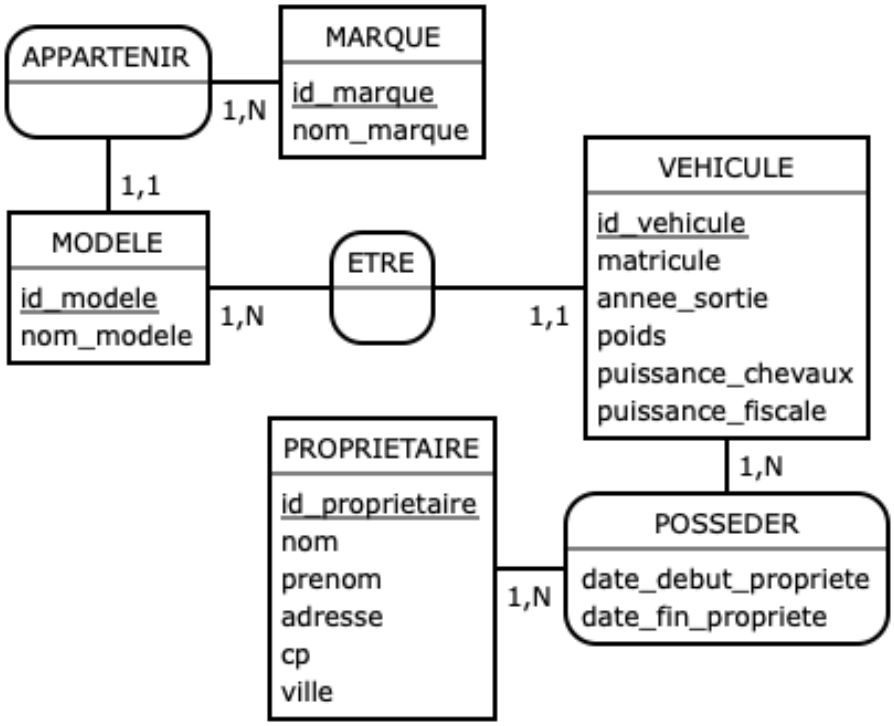
5.2.1. Diagramme de cas d'utilisation



5.2.2. Diagramme d'activités



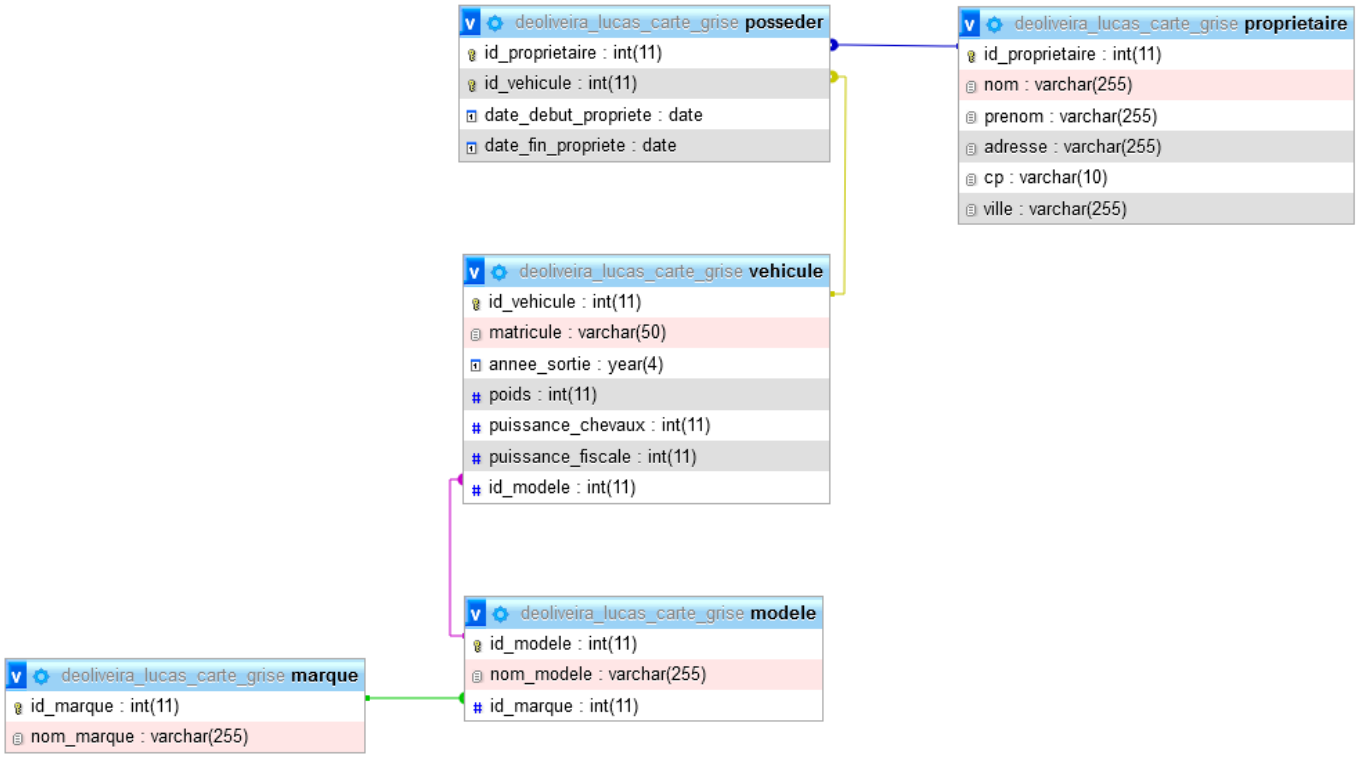
5.2.3. Modèles Conceptuel de Données (MCD)



5.2.4. Modèle Logique de Données (MLD)

- ATHLETE (id_athlete, nom_athlete, prenom_athlete, #id_pays, #id_genre)
- EPREUVE (id_epreuve, nom_epreuve, date_epreuve, heure_epreuve, #id_lieu, #id_sport)
- GENRE (id_genre, nom_genre)
- LIEU (id_lieu, nom_lieu, adresse_lieu, cp_lieu, ville_lieu)
- PARTICIPER (#id_athlete, #id_epreuve, resultat)
- PAYS (id_pays, nom_pays)
- SPORT (id_sport, nom_sport)
- UTILISATEUR (id_utilisateur, nom_utilisateur, prenom_utilisateur, login, password)

5.2.5. Modèle Physique de Données (MPD)



6. Technologies utilisées

6.1. Langages de développement Web

- Front-end : HTML5, CSS, JavaScript Vanilla
- Back-end : PHP 8.1

6.2. Base de données

- Langage base de données : SQL

7. Sécurité

7.1. Login et protection des pages administrateurs

Les données devront être sauvegardées dans une base de données relationnelles :

- Les mots de passe des administrateurs soient cryptés en base de données.
- Les pages réservées aux administrateurs ne doivent pas être accessibles si l'utilisateur n'est pas connecté.

7.2. Cryptage des mots de passe avec Bcrypt

Bcrypt est un algorithme de hachage de mot de passe fort qui est résistant aux attaques par force brute. Il est plus lent que d'autres algorithmes, tels que SHA-256, ce qui rend plus difficile pour les attaquants de deviner les mots de passe.
Dans notre application, cela va permettre d'assurer un environnement sécurisé pour tous les utilisateurs.

```
bcrypt . genSalt ( saltRounds ,  function ( err ,  salt )  {
    bcrypt . hash ( myPlaintextPassword ,  salt ,  function ( err ,  hash )  {
        // Stockez le hachage dans votre base de données de mots de passe.
    } ) ;
} ) ;
```

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

Le Cross-site Scripting ou attaque XSS est une vulnérabilité par injection de code. Pour détecter et exploiter ce type de menace, il est indispensable qu'un hacker injecte du code malveillant à partir des paramètres d'accès côté client.
Dans notre application, cela va permettre de protéger les données des utilisateurs et assurer une confiance.

```
function escape(s) {
  return '<script>console.log("'+s+'");</script>';
}
```

7.4. Protection contre les injections SQL

Une injection SQL se produit lorsqu'un utilisateur malveillant communique une entrée qui modifie la requête SQL envoyée par l'application web à la base de données. Cela lui permet alors d'exécuter d'autres requêtes SQL non souhaitées directement sur la base de données.

Pour ce faire, l'attaquant doit injecter du code en dehors des limites de l'entrée utilisateur attendue, afin qu'il ne soit pas exécuté comme une entrée standard. Dans le cas le plus simple, il suffit d'injecter un guillemet simple ou double pour échapper aux limites de la saisie utilisateur et ainsi insérer des données directement dans la requête SQL.