# APPLICATION JO2024 : CAHIER DES CHARGES TECHNIQUES

En JAVA

BTS SIO SLAM

FOUSSEYNOU SIDIBE

## Sommaire

1. Contexte ..... 2
1.1 Présentation ..... 2
1.2 Période et Modalité ..... 2
2. Objectifs du projet ..... 2
3. Besoins fonctionnels ..... 2
3.1 Fonctionnalités principales ..... 2
3.2 Sécurité ..... 3
4. Architecture technique ..... 3
4.1 Ressources matérielles et logicielles ..... 3
4.2 Langages utilisés ..... 3
5. Conception et Modélisation ..... 4
5.1 Modèle de données ..... 4
5.2 Interfaces utilisateur ..... 7
5.3 Diagrammes UML ..... 10
6. Développement et Tests ..... 11
6.1 Gestion du projet ..... 11
7. Déploiement ..... 12
8. Livrables ..... 12
9. Conclusion ..... 12

# 1. Contexte

## 1.1 Présentation

Ce projet a été réalisé dans le cadre du **BTS SIO – SLAM** lors des séances d'AP au **Lycée La Tournelle de La Garenne-Colombes**. L'objectif est de développer une **application Java** permettant aux utilisateurs de consulter les **sports, le calendrier des épreuves et les résultats des Jeux Olympiques 2024**.

## 1.2 Période et Modalité

- **Durée** : Du **06 novembre 2023 au 08 mars 2024**
- **Mode de réalisation** : **Travail individuel** avec un suivi pédagogique.

# 2. Objectifs du projet

L'application doit permettre de :

**- Afficher la liste des sports et disciplines olympiques**
**- Gérer le calendrier des épreuves** avec les dates et lieux des compétitions
**- Afficher les résultats des épreuves en temps réel**
**- Créer un espace administrateur** pour gérer les informations des compétitions
**- Stocker et organiser les données via une base de données SQL**

# 3. Besoins fonctionnels

## 3.1 Fonctionnalités principales

- **Consultation des disciplines sportives**
- **Affichage dynamique du calendrier des épreuves**
- **Gestion et mise à jour des résultats des compétitions**
- **Espace administrateur sécurisé** (ajout, modification et suppression des épreuves)
- **Gestion des utilisateurs et droits d'accès**

## 3.2 Sécurité

- **Authentification sécurisée** pour l'espace administrateur
- **Protection contre les injections SQL** via des requêtes préparées
- **Validation des entrées utilisateurs** pour éviter les failles XSS

# 4. Architecture technique

## 4.1 Ressources matérielles et logicielles

- **Matériel** : Ordinateur portable, connexion internet
- **IDE** : **Visual Studio Code**
- **SGBD** : **MySQL** via **MAMP**
- **Outils de conception** : **Mocodo, Visual Paradigm**

## 4.2 Langages utilisés

- **Back-end** : **Java**
- **Front-end** : **HTML5, CSS3, JavaScript**
- **Base de données** : **SQL**

# 5. Conception et Modélisation

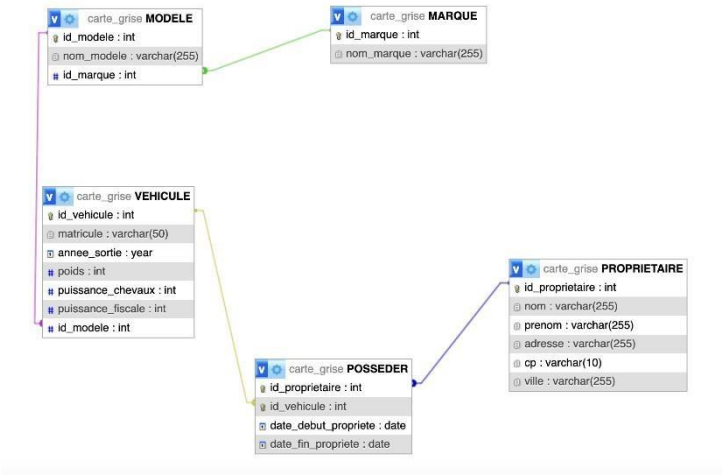
## 5.1 Modèle de données

**Dictionnaire de données :**

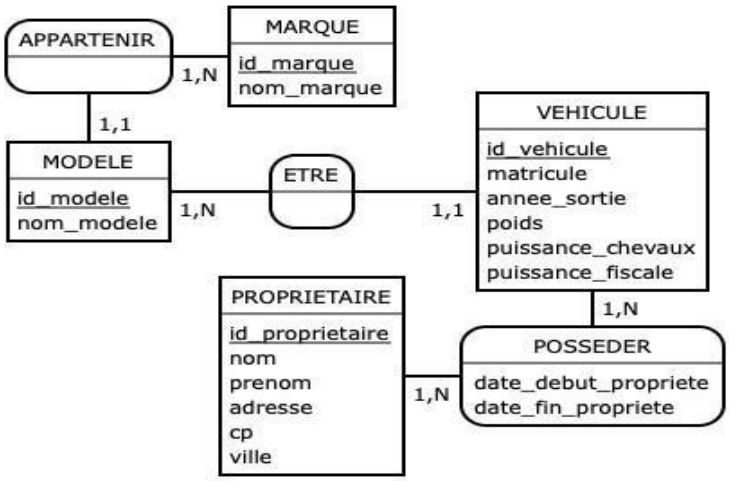
| Table | Attribut | Type de données | Description |
|---|---|---|---|
| MARQUE | id_marque | INT | Identifiant unique de la marque, Clé primaire, Auto-incrémentée |
| MARQUE | nom_marque | VARCHAR(255) | Nom de la marque |
| MODELE | id_modele | INT | Identifiant unique du modèle, Clé primaire, Auto-incrémentée |
| MODELE | nom_modele | VARCHAR(255) | Nom du modèle |
| MODELE | id_marque | INT | Identifiant de la marque associée, Clé étrangère (référence `MARQUE.id_marque`) |
| PROPRIETAIRE | id_proprietaire | INT | Identifiant unique du propriétaire, Clé primaire, Auto-incrémentée |
| 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 postal du propriétaire |
| PROPRIETAIRE | ville | VARCHAR(255) | Ville du propriétaire |
| VEHICULE | id_vehicule | INT | Identifiant unique du véhicule, Clé primaire, Auto-incrémentée |
| VEHICULE | matricule | VARCHAR(50) | Matricule du véhicule |
| VEHICULE | annee_sortie | YEAR | Année de sortie du véhicule |
| VEHICULE | poids | INT | Poids du véhicule |
| VEHICULE | puissance_chevaux | INT | Puissance du véhicule en chevaux |
| VEHICULE | puissance_fiscale | INT | Puissance fiscale du véhicule |
| VEHICULE | id_modele | INT | Identifiant du modèle associé, Clé étrangère (référence `MODELE.id_modele`) |
| POSSEDER | id_proprietaire | INT | Identifiant du propriétaire, Clé étrangère (référence `PROPRIETAIRE.id_proprietaire`) |
| POSSEDER | id_vehicule | INT | Identifiant du véhicule, Clé étrangère (référence `VEHICULE.id_vehicule`) |
| POSSEDER | date_debut_propriete | DATE | Date de début de la propriété |
| POSSEDER | date_fin_propriete | DATE | Date de fin de la propriété (optionnelle) |

- **MPD , MCD & MLD** pour structurer les relations entre les tables

## Modèle Physique de données ( MPD )




## Modèle Conceptuel de données ( MCD )



## Modèle Logiques de données ( MLD ) format 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 référence à MARQUE (id_marque)

- VEHICULE (id_vehicule, matricule, annee_sortie, poids, puissance_chevaux, puissance_fiscale, id_modele)
  Clé primaire : id_vehicule
  Clé étrangère : id_modele en référence à MODELE (id_modele)
- PROPRIETAIRE (id_proprietaire, nom, prenom, adresse, cp, ville)
  Clé primaire : id_proprietair
- POSSEDER (id_proprietaire, id_vehicule, date_debut_propriete, date_fin_propriete)
  Clé primaire : id_proprietaire, id_vehicule, date_debut_propriete
  Clé étrangère : id_proprietaire en référence à PROPRIETAIRE (id_proprietaire)
  Clé étrangère : id_vehicule en référence à VEHICULE (id_vehicule)

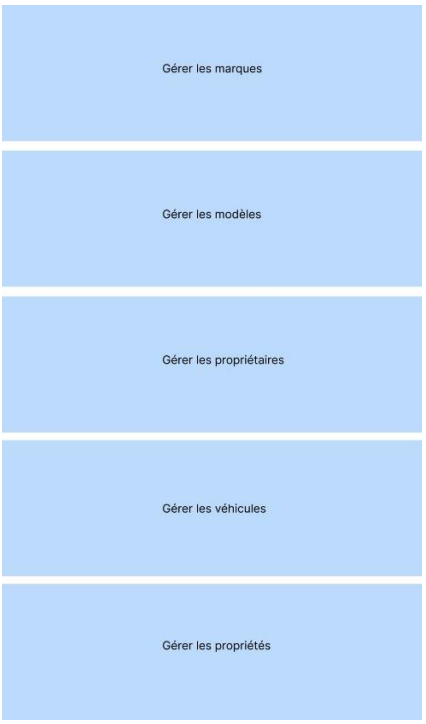
# 5.2 Interfaces utilisateur

- **Wireframes** créés avec le logiciel **Figma.**

## MENU :

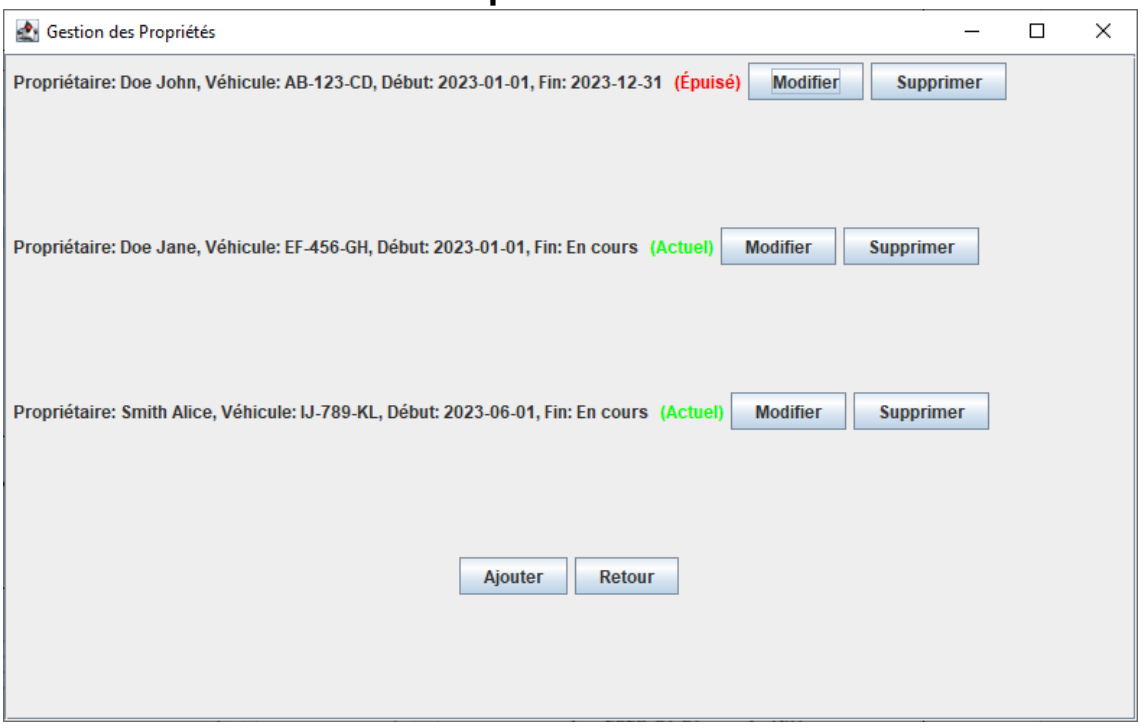
### Maquette :


### Wireframe :

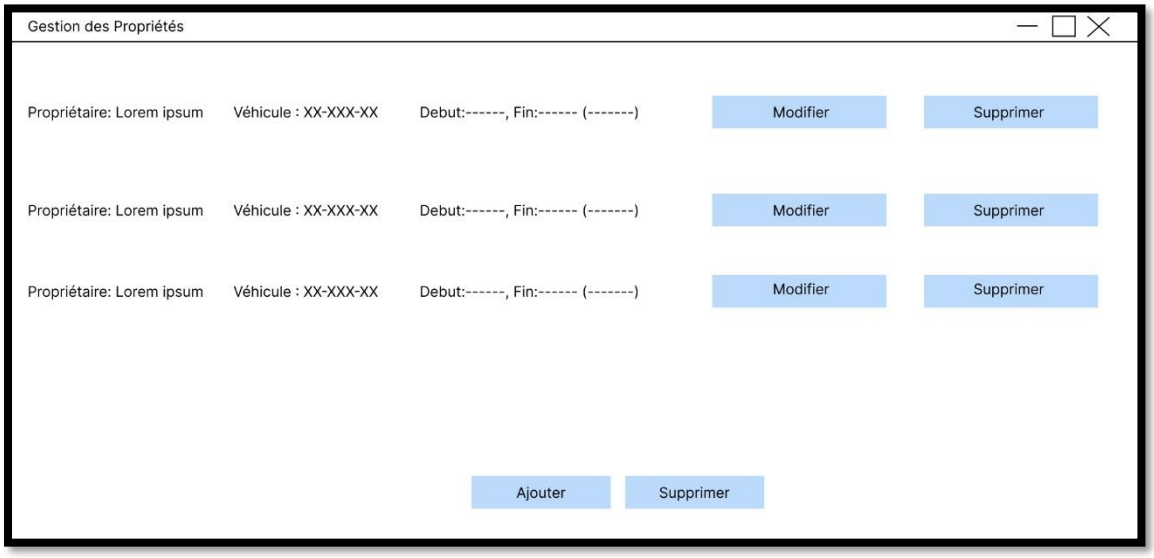


# POSSEDER View :

## Maquette :

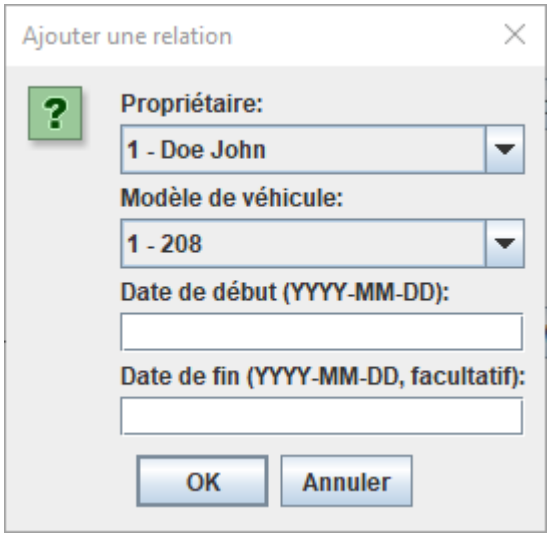



## Wireframe :

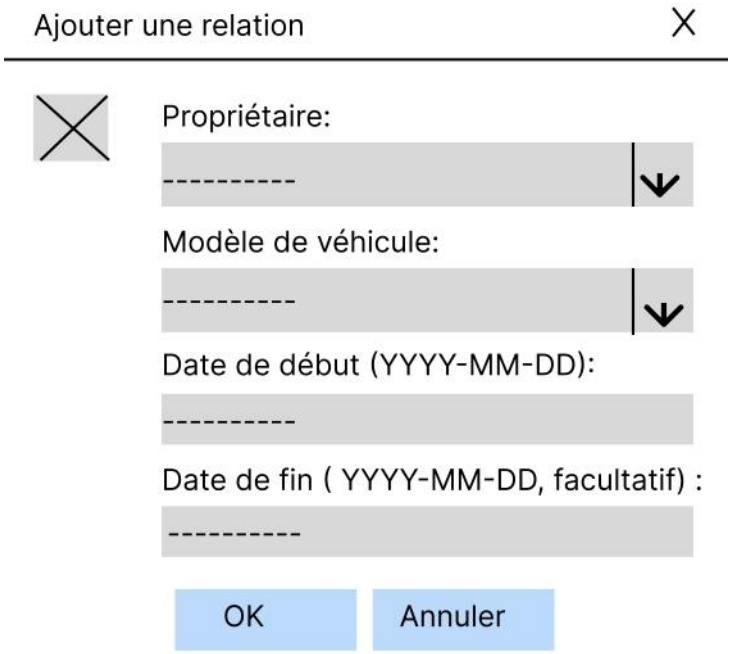


## Ajout POSSEDER :

### Maquette :

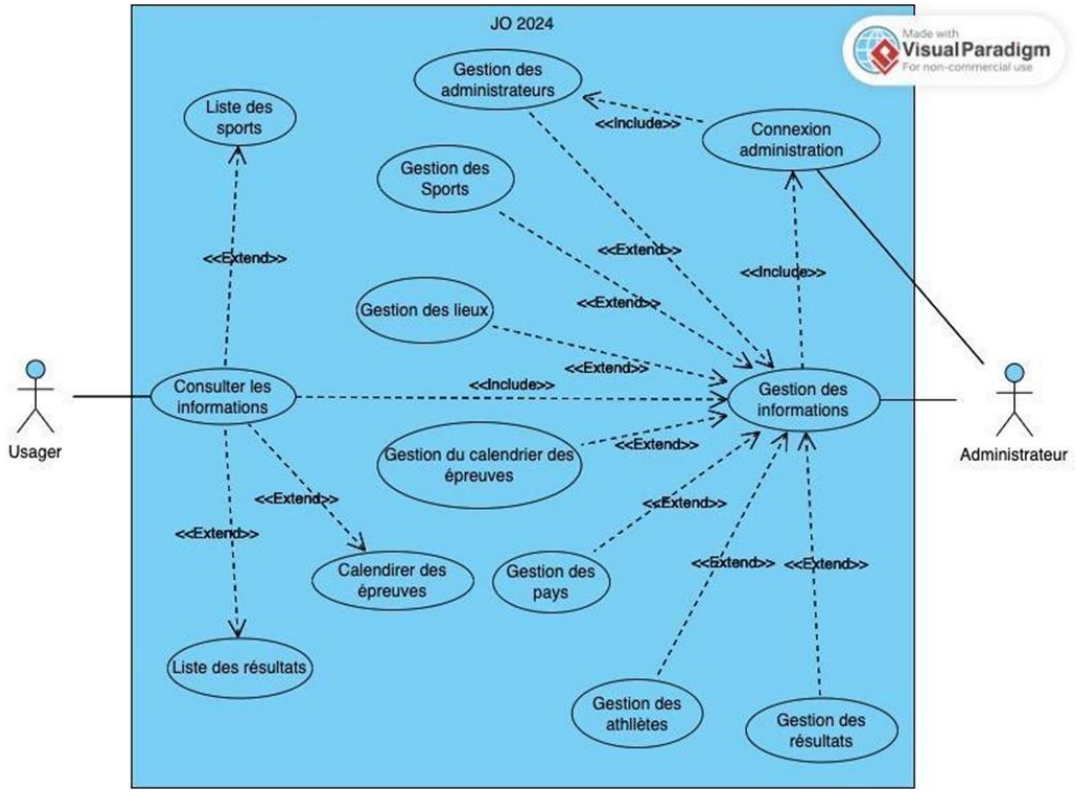



### Wireframe :

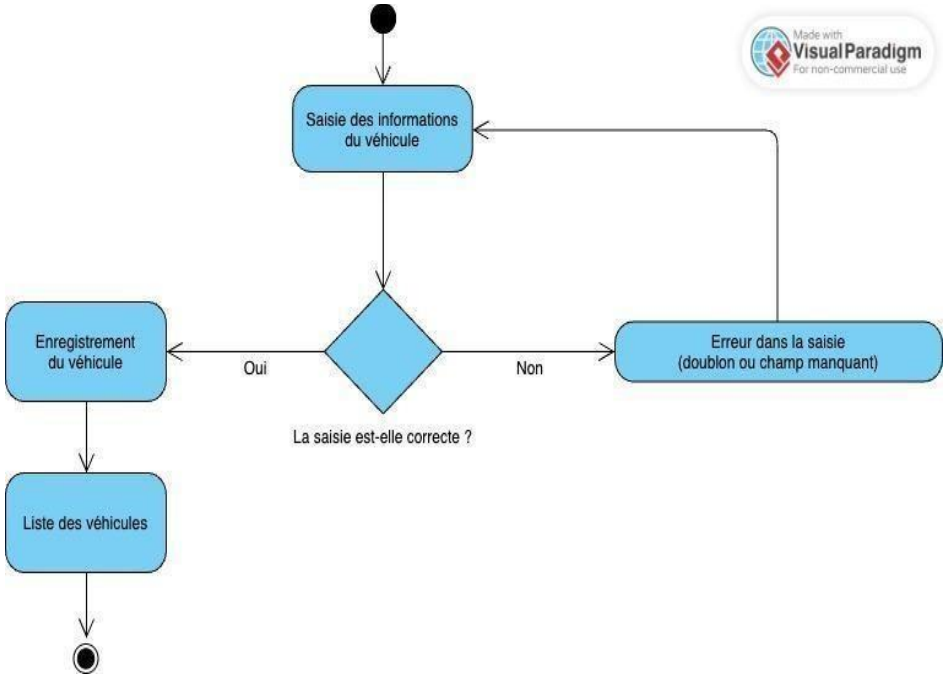


# 5.3 Diagrammes UML

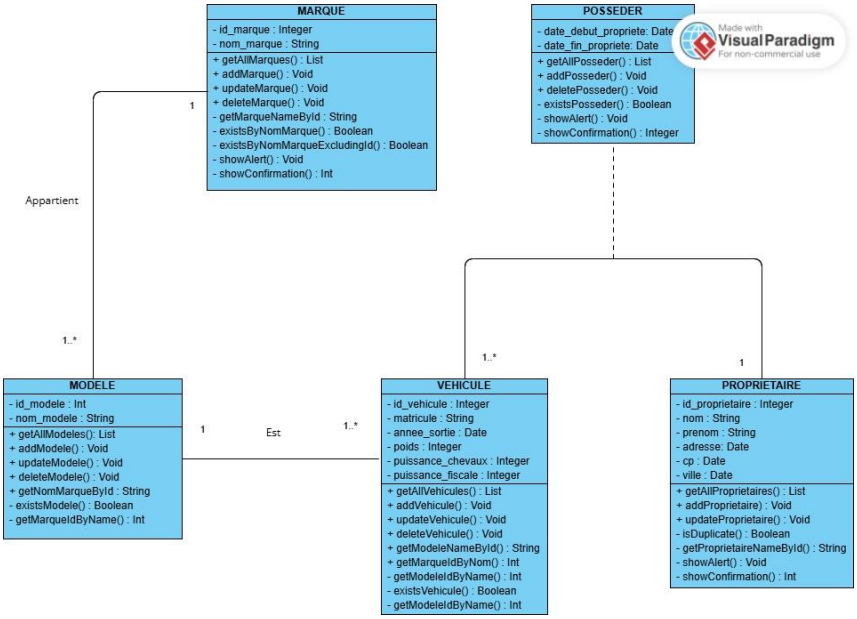
- Diagramme de cas d'utilisation :




- Diagramme d'activités :



- Diagramme de classes :



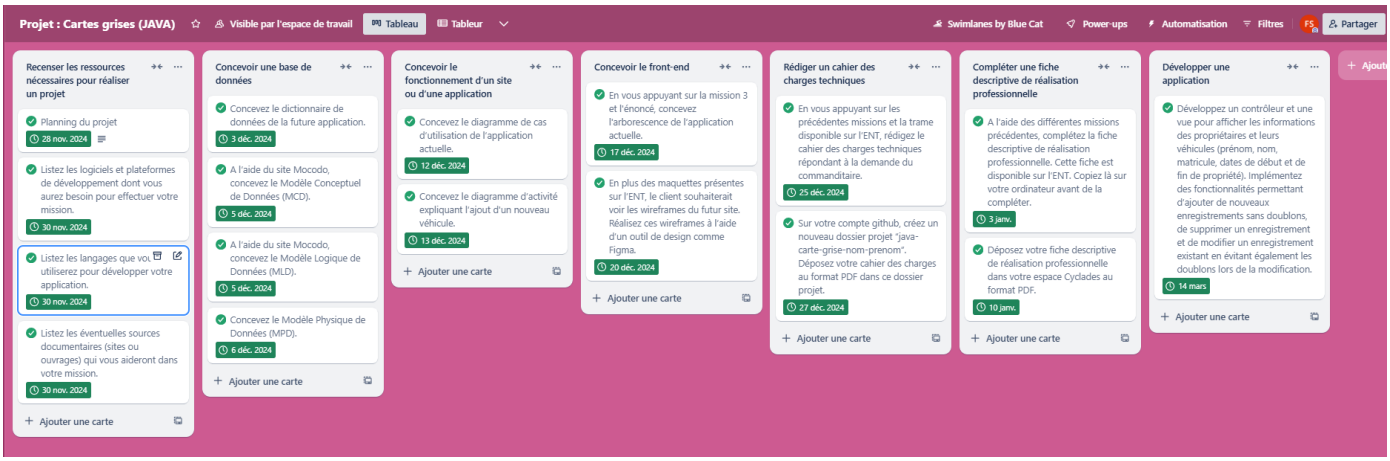

# 6. Développement et Tests

## 6.1 Gestion du projet

- Méthode classique : Modèle V

- Utilisation de GitHub

- Trello :



# 7. Déploiement

- **Hébergement sur un serveur local via MAMP**

# 8. Livrables

- Code source du projet sur **GitHub**
- Base de données SQL exportable
- Documentation technique complète
- Rapport de test détaillé

# 9. Conclusion

Ce projet vise à offrir une **application Java performante et intuitive** pour gérer les informations des **cartes grises de plusieurs modèles de véhicules**. Il met en application les compétences acquises en **conception, développement et gestion des données** dans un cadre professionnel.