

**Dossier de conception**

>> Diginamic.fr

|  |  |  |  |
| --- | --- | --- | --- |
| **Révision** | **Rédacteurs** | **Date** | **Objet** |
| 1 | R. BONNAMY | 17/02/2021 | Création du document |
| 2 | M.BASIER | 23/01/2025 | Renseignement du document |
| 3 | J.BROU | 27/01/2025 | Ajout des diagrammes |

# Introduction

## Objet du document

Ce document a pour objectif de présenter l’essentiel des questions techniques liées à la mise en place de l’application OuiMouve.

Ce document présente :

* Le diagramme de classes
* Le modèle physique de données
* L'architecture du projet
* La stratégie de test

# Sommaire

### Table des matières

[1 Introduction 2](#_Toc182817810)

[1.1 Objet du document 2](#_Toc182817811)

[2 Sommaire 3](#_Toc182817812)

[2.1.1 Table des matières 3](#_Toc182817813)

[3 Architecture logicielle 4](#_Toc182817814)

[3.1 Produits et versions 4](#_Toc182817815)

[3.1.1 Langages, frameworks et librairies spécifiques 4](#_Toc182817816)

[3.1.2 Serveur de base de données 4](#_Toc182817817)

[4 Focus Technique 5](#_Toc182817818)

[4.1 Diagramme de classes métier (ou MCD) 5](#_Toc182817819)

[4.2 Diagramme entités relations (ou MPD) 5](#_Toc182817820)

[4.3 Règles de développement coté back 5](#_Toc182817821)

[5 TESTS et intégration 6](#_Toc182817822)

[5.1 Stratégie de tests 6](#_Toc182817823)

[5.2 Indicateurs de qualité de code 6](#_Toc182817824)

# Architecture logicielle

## Produits et versions

### Langages, frameworks et librairies spécifiques

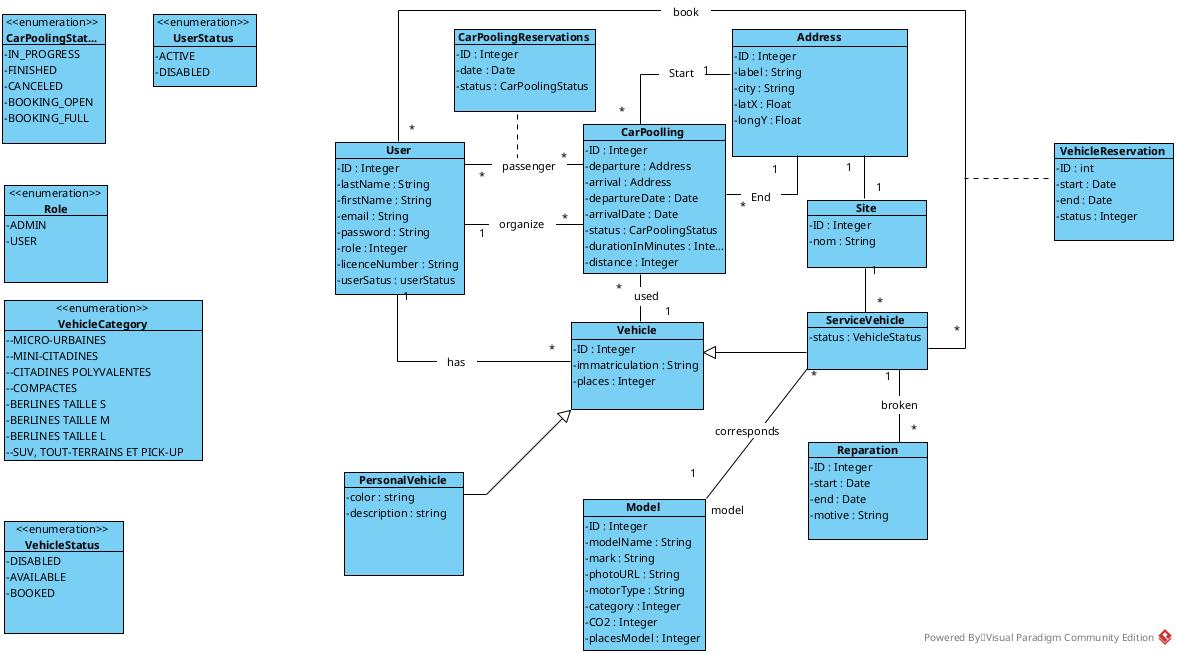
|  |  |
| --- | --- |
| **Nom** | **Version** |
| Langage Java | 21 |
| Spring Boot | 3.4.2 |
| JPA | 2.1 |
| Hibernate | 2.0 |
| Angular | 19 |
| OpenStreetMap (base de données géographiques) | 0.6 |
| Open Source Routing Machine (calcul d’iténéraire distance/durée) | 1 |
| Photon (autocomplétion d’adresse) | 1 |

### Serveur de base de données

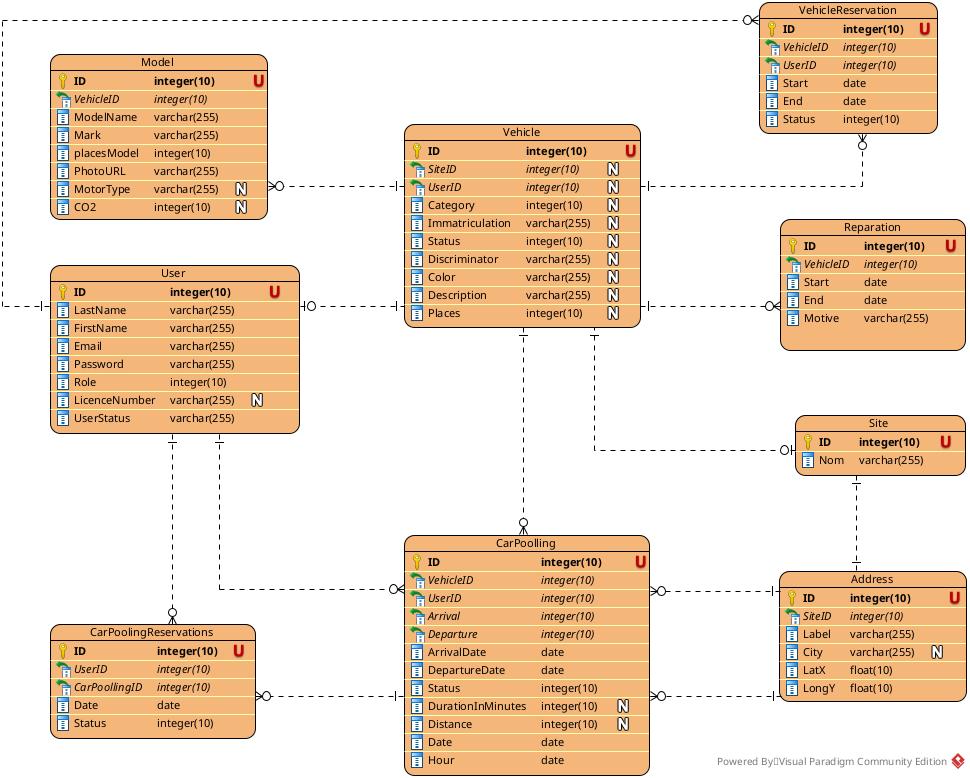
|  |  |
| --- | --- |
| **Nom** | **Version** |
| MariaDB | 3.5.1 |

# Focus Technique

## Diagramme de classes métier (ou MCD)



## Diagramme entités relations (ou MPD)



## Règles de développement coté back

Règles de développement :

* 100% de la Javadoc doit être renseigner sur les classes et les méthodes publique
* Indentation du code : 2 espaces
* Les noms de fonctions/variables doivent être descriptifs et respecter la norme camelCase.
* Pas plus de 2 imbrications dans les méthodes.
* Une méthode ne doit pas faire plus de 50 lignes et ne doit faire qu’une seule chose.
* Le nombre de paramètres ne doit pas dépasser 7.

**Découpage en couches** :

* Couche contrôleur qui va réaliser les requêtes api
* Couche de services qui va réaliser les contrôles métier.
* Couche repository qui sert d’interface entre la base de données et la couche service
* Couche DTO : le contrôleur renvoie vers le front des instances de DTO.
* Classes utilitaires/bean

# TESTS et intégration

## Stratégie de tests

Il y aura des tests unitaires sur l’ensemble des développements.

Les tests doivent suivre

Les tests automatisés seront faits sur les apis (CRUD), et appels base de données (CRUD).

Des tests d’intégrations seront rédigés dans Squash Tm.

## Indicateurs de qualité de code

Taux de documentation des méthodes et classes.

Conformité de l'indentation (2 espaces).

Conformité des noms de fonctions/variables à la norme camelCase.

Nombre d'imbrications dans les méthodes (pas plus de 2).

Nombre de lignes par méthode (pas plus de 50 lignes).

Nombre de paramètres par méthode (pas plus de 7).

Respect de l'architecture en couches (contrôleur, service, repository, DTO).

Taux de duplication de code.

Taux de couverture de code par les tests unitaires.

Nombre de violations des règles de codage.

## Test d’intégrations

Avancement de la description des cas de tests dans squashTm:

Cas de tests organisation :

Cas de tests Généraux :



Exigences :

