**Dossier d’Architecture Technique – API Covoiturage**

**1. Introduction**

Ce document explique l'architecture technique de l'API de covoiturage, y compris la structure de la base de données, les relations entre les entités, et les fonctionnalités principales exposées via des routes REST.

**2. Modèle Conceptuel des Données (MCD)**

Le système de covoiturage repose sur plusieurs entités principales :

* **Utilisateur** : Représente les utilisateurs inscrits (passagers et conducteurs).
* **Voiture** : Représente les véhicules associés à un utilisateur conducteur.
* **Ville** : Contient les différentes villes utilisées pour les trajets.
* **Trajet** : Représente un trajet proposé par un conducteur.
* **Réservation** : Permet à un utilisateur de réserver un trajet.
* **Rôle** : Permet de gérer les niveaux d'accès (admin, utilisateur normal).

**3. Modèle Logique des Données (MLD)**

Voici la représentation relationnelle de la base de données :

* **utilisateur** (*id\_utilisateur*, id\_role, id\_ville, nom, prenom, email, mot\_de\_passe)
* **role** (*id\_role*, nom\_role)
* **ville** (*id\_ville*, code\_postale, nom\_commune)
* **voiture** (*id\_voiture*, marque, modele, immatriculation, nb\_places, id\_utilisateur FK)
* **trajet** (*id\_trajet*, id\_utilisateur FK, date\_heure, places\_restantes, detail\_trajet, id\_ville\_depart FK, id\_ville\_arrivee FK)
* **reservation** (*id\_reservation*, id\_utilisateur FK, id\_trajet FK, statut, date\_reservation)

**4. API REST : Routes et Fonctionnalités**

**4.1. Utilisateurs**

* POST /api/utilisateur : Inscription
* GET /api/utilisateurs : Liste des utilisateurs (Admin uniquement)
* PUT /api/utilisateur/{id} : Modification des informations de l'utilisateur
* DELETE /api/utilisateur/{id} : Suppression de l'utilisateur (Admin ou utilisateur lui-même)

**4.2. Voitures**

* POST /api/voiture : Ajout d'une voiture associée à un utilisateur
* DELETE /api/voiture/{id} : Suppression d'une voiture (par son propriétaire uniquement)
* GET /api/voitures : Liste des voitures

**4.3. Villes**

* POST /api/ville : Ajout d'une ville (Admin uniquement)
* GET /api/villes : Liste des villes
* GET /api/ville/recherche : Recherche d'une ville via code\_postal et nom\_commune
* DELETE /api/ville/{id} : Suppression d'une ville (Admin uniquement)

**4.4. Trajets**

* POST /api/trajet : Création d'un trajet
* GET /api/trajets : Liste des trajets
* GET /api/trajets/recherche : Recherche d'un trajet
* PUT /api/trajet/{id} : Modification d'un trajet (Conducteur uniquement)
* DELETE /api/trajet/{id} : Suppression d'un trajet (Conducteur uniquement)

**4.5. Réservations**

* POST /api/reservation : Création d'une réservation
* GET /api/reservation : Liste des réservations
* GET /api/reservation/{id} : Récupération d'une réservation par ID
* PUT /api/reservation/{id}/confirmer : Confirmation d'une réservation (Conducteur uniquement)
* PUT /api/reservation/{id}/annuler : Annulation d'une réservation (Utilisateur uniquement)

**5. Règles de Gestion**

* Un utilisateur peut être conducteur uniquement s'il a une voiture enregistrée.
* Un trajet ne peut être supprimé que par son créateur.
* Une réservation ne peut être annulée que par le passager.
* Un admin peut supprimer n'importe quel utilisateur.
* Un utilisateur ne peut modifier que ses propres informations.

**6. Technologies Utilisées**

* **Backend** : PHP 8.2 avec Symfony
* **Base de données** : MySQL 8.0
* **Authentification** : JWT (LexikJWTAuthenticationBundle)
* **Tests API** : Insomnia
* **ORM** : Doctrine

**7. Conclusion**

Cette API offre une gestion complète des utilisateurs, des trajets et des réservations. Elle implémente des règles de gestion précises pour garantir la cohérence et la sécurité du système.

**8. Schéma MCD et UML**

Schéma MCD

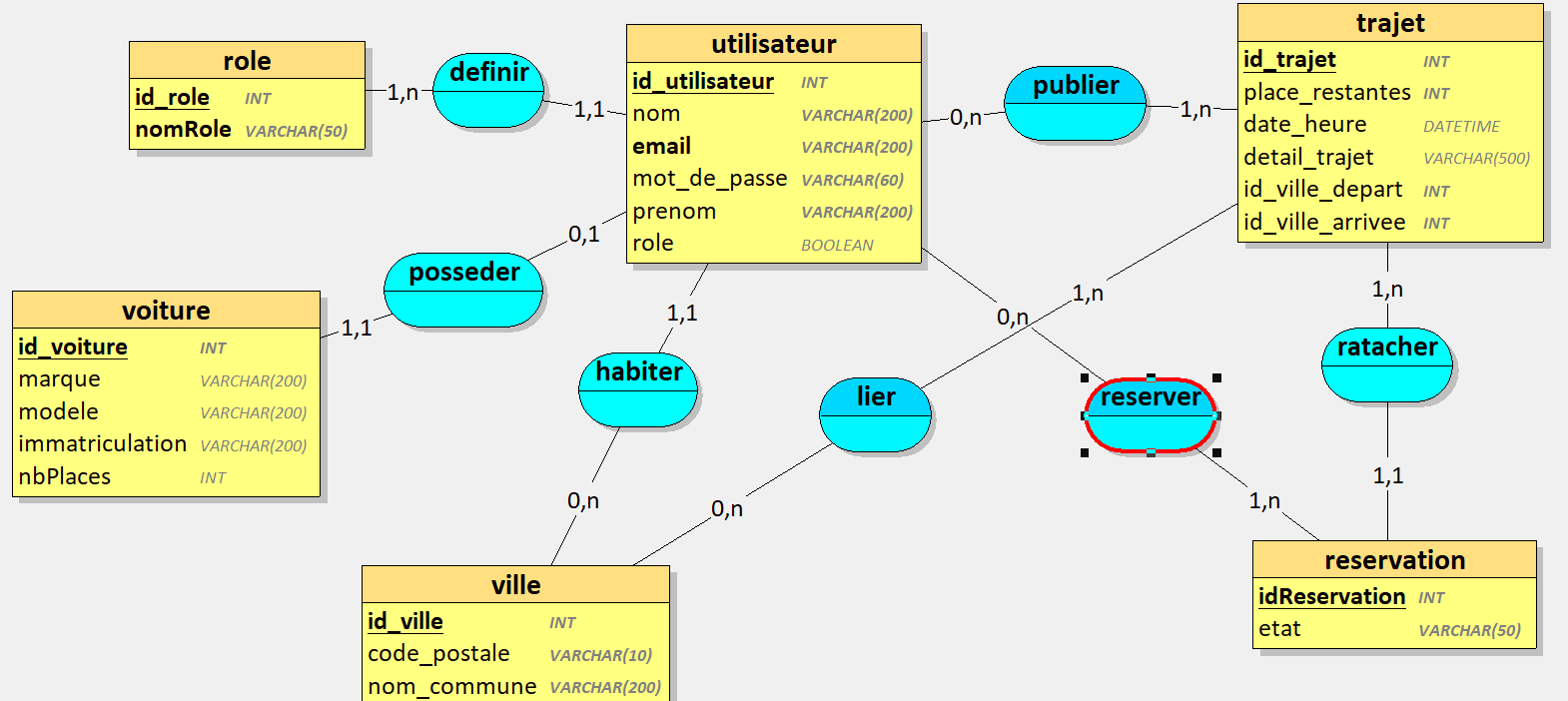


Schéma UML

Le diagramme UML fourni illustre clairement les relations entre les entités du système.

