EcoRide Documentation Technique

Chaigneau Johan
Titre Professionel DWWM Septembre 2025

Table des matières:

1. Présentation du projet

EcoRide est une application web de covoiturage écologique. Elle permet aux utilisateurs de proposer ou de réserver des trajets entre particuliers, avec une priorité donnée aux véhicules électriques ou hybrides .

Objectif Principal:

- Proposer une solution alternative de transport responsable
- Favoriser l'entraide local et les trajets du quotidien
- Mettre en place une stack technique moderne (HTL,CSS,JS,PHP,Symfony,Docker)

Public visé :

- Utilisateurs réguliers de trajets domicile/travail
- Employés d'entreprise soucieuse de l'écologie
- Administrateur de flotte de véhicule

2. Choix Techniques

Technologie utilisées :

Élément	Choix	Justification
Frontend	HTML/CSS/JS + Bootstrap	Technologie que je maîtrise bien et qui garantissent un rendu responsive rapide.
Backend	PHP 8.3/Symfony 7	Framework sécurisé,structurant et largement utilisé en entreprise (ex:BlaBlaCar).
BDD principal	MariDB	Relationnelle, légère,compatible MYSQL. Utilisée via Dbeaver pour sa clarté.
BDD Complémentaire	MongoDB En local unquement	Adaptée aux données non relationnelles
Serveur Web	Nginx (en découverte)	Choisi dans un contexte Docker.
Conteneurisation	Docker/Docker Compose	Pour isoler l'environnement et simuler une architecture en production
IDE	VS Code	Léger,puissant, extensions adaptées au projet
Versionning	GIT/GitHub	Utilisation du flux main/dev

Note personnelle : Le choix de Symfony est aussi motivé par sa capacité à gérer nativement la sécurité (authentification,rôle,CSRF, validation des formulaires),Malgré une courbe d'apprentissages plus longue que du php natif , il permet d'adopter de bonnes pratiques professionnelles dès le départ. J'ai profité de ce projet pour découvrir Nginx avec Docker, bien que je sois plus à l'aise avec Apache.

3. Environnement de Travail

J’ai réalisé le développement d’EcoRide principalement sur mon ordinateur avec **Visual Studio Code** (VSC), que je trouve pratique grâce à ses extensions pour PHP, Docker et Symfony.
J’ai mis en place l’environnement avec **Docker** et **Docker Compose** pour lancer facilement tous les services nécessaires : le serveur Nginx, Symfony avec PHP, et la base MariaDB.

Pour gérer les dépendances PHP et installer Symfony, j’ai utilisé **Composer**.
Pour tout ce qui est CSS, JavaScript et Bootstrap, j’ai utilisé **Webpack Encore**, même si ce n’est pas la partie que je maîtrise le mieux pour l’instant.
Pour manipuler la base de données, j’ai préféré passer par **DBeaver**, qui m’a vraiment facilité la vie pour voir et modifier mes tables.

J’ai choisi ces outils parce qu’ils permettent d’avoir un environnement stable,facile à relancer, et assez proche de ce qu’on attend en entreprise, même si certains aspects (Webpack Encore ou Docker) restent encore nouveau pour moi .

4. Modèle de données

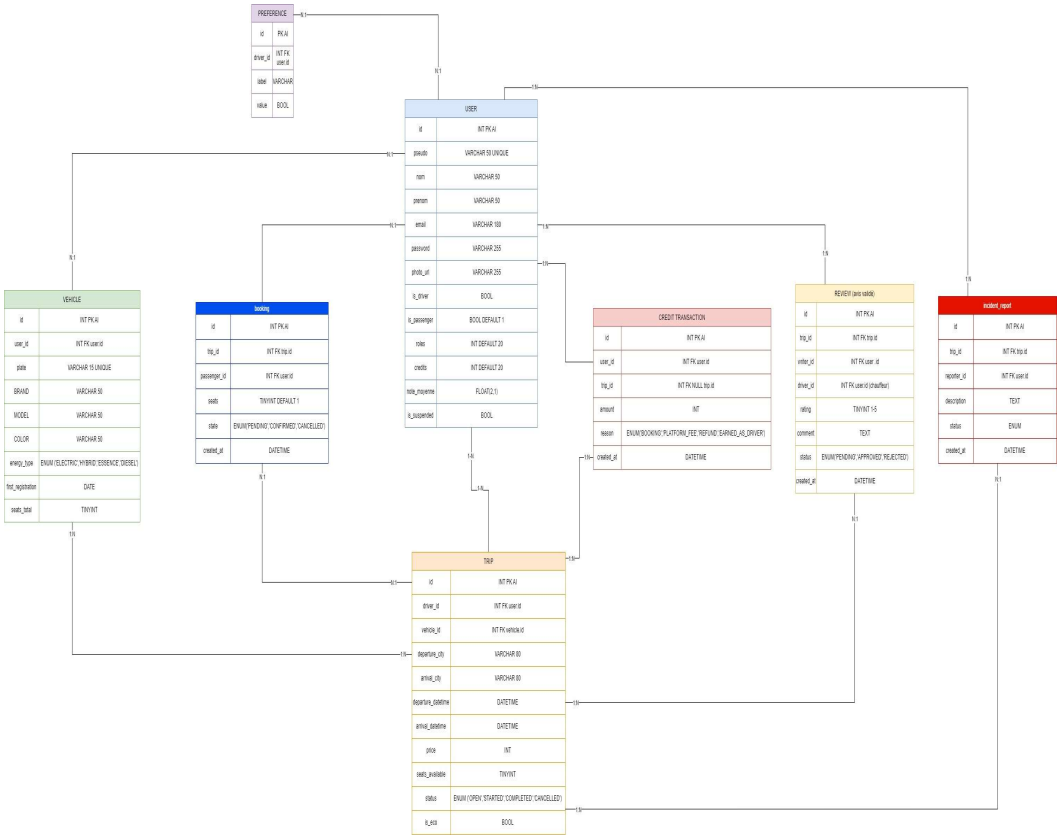

Schéma de base de données disponible dans le fichier doc au format .JPG.

Le modèle conceptuelle de données de l'application Ecoride , décrit les principales entité de l'application . Elle répond aux besoins de gestion du covoiturage , des utilisateurs,des véhicules, des trajets, des réservations, des avis et des transactions.

- User Contient les informations personnelles, d'identification et de rôle (passager, chauffeur, employé…).
  Champs clés : email, mot de passe, rôles, prénom, nom, téléphone, date de naissance, crédit, statut (chauffeur/passager), photo de profil, suspension.
- Préférences Permet à chaque utilisateur d'enregistrer des préférences (ex : fumeur, animal, etc.).
  Champs clés : utilisateur lié, libellé, valeur.
- Véhicle Représente un véhicule déclaré par un utilisateur chauffeur (modèle, marque, couleur, plaque, énergie, nombre de places).
  Champs clés : marque, modèle, couleur, plaque d'immatriculation, type d'énergie, date première immatriculation, propriétaire, nombre de places

- Trip Représente une offre de covoiturage, associée à un chauffeur et un véhicule.
  Champs clés : chauffeur, véhicule, ville de départ/arrivée, date/heure de départ et d'arrivée, durée, prix, nombre de places restantes, aspect écologique, statut.
- Booking Permet à un utilisateur de réserver une ou plusieurs places sur un trajet.
  Champs clés : passager, trajet, nombre de places, état, date de création, statut du retour/avis.
- Reviews Permet de noter et commenter un trajet, en liant l'avis au passager, au chauffeur et au trajet.
  Champs clés : trajet, auteur, chauffeur, note, commentaire, statut, date.
- Crédit transaction Gère les mouvements de crédits (achat, dépense, gain, etc.) liés aux utilisateurs et trajets.
  Champs clés : utilisateur, trajet lié, type d'opération, montant, description, date.
- Incident repport Permet aux utilisateurs de signaler un problème sur un trajet, pour gestion par un employé.
  Champs clés : trajet concerné, auteur du signalement, description, statut, date.

5. Diagrammes UML

Diagramme de cas d'utilisation:

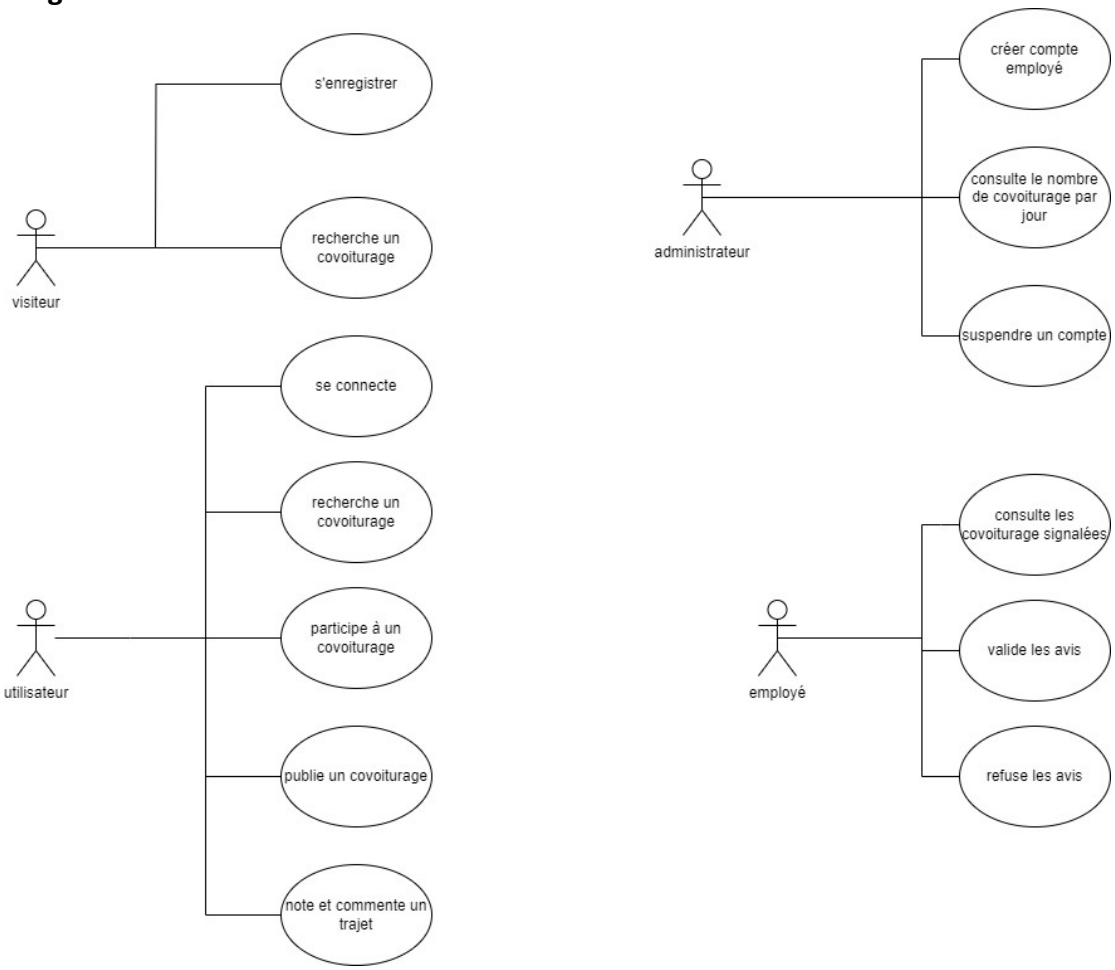

Diagramme de séquence:

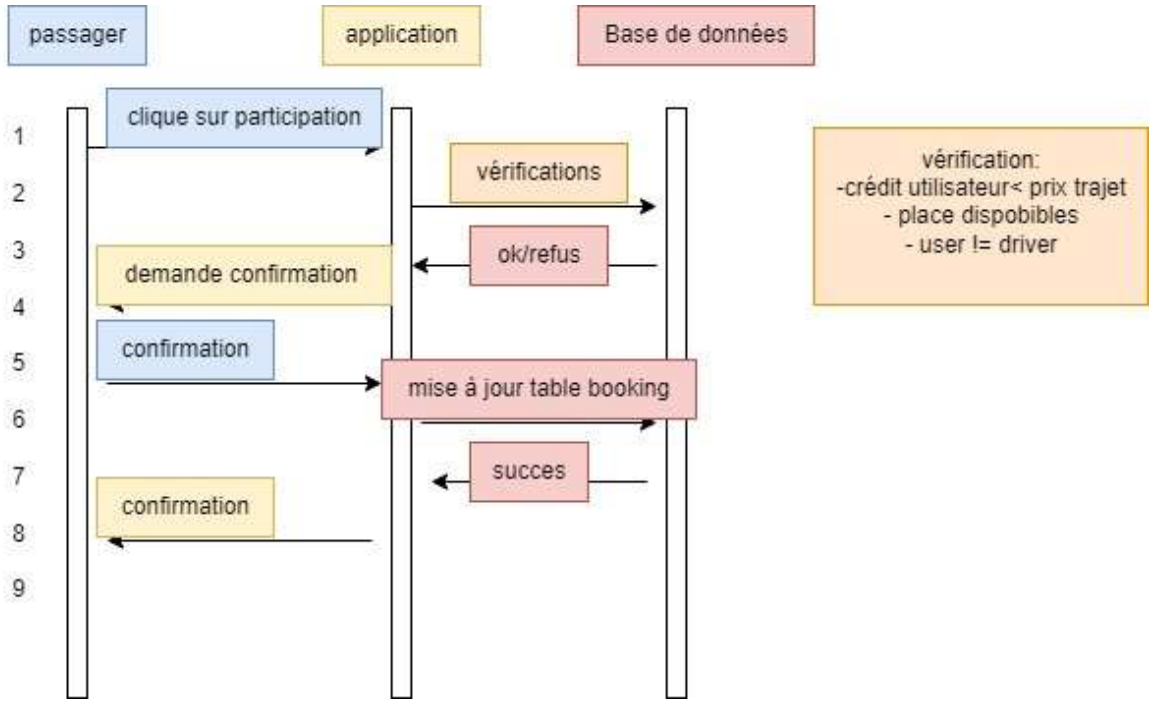

Diagramme de séquences montrant la réservation d'un trajet.

6. Déploiement

1. Choix de la plateforme de deploiement

J'ai choisi de deployer mon application symfony sur fly.io.
Mon choix à été motivé par:

- ✓ Compatibilité Docker
- ✓ Simplicité de mise en oeuvre
- ✓ Base de données ecterne possible
- ✓ Gratuit pour les petits projet

J'ai utiliser railway pour déployer ma base de données relationnel:

- ✓ Simple d'utilisation
- ✓ Compatible fly.io en ce qui concerne la BDD relationnel.
- ✓ Mise en place rapide

2. Préparation de l'environnement de production

Avant de déployé j'ai procédé à plusieurs étapes:

- ✓ Nettoyage du code:

Suppression des fichiers temporaire ,images de testes , assets inutiles et données de développement.

Passage en mode APP_ENV=prod , APP_DEBUG=0

Mise à jour des dépendance:
« composer install --no-dev --optimize-autoloader»

Compilation Web Pack Encore pour la production:
«npm run build»

3. Configuration de la base de données

- ✓ Création d'un compte sur railway
- ✓ Création de la base de données et récupération du lien de connexion

4. Conteneurisation de l'application

J'ai modifié mon Dockerfile local pour l'optimiser pour la production

- ✓ Installation de toutes les extensions PHP nécessaires à EcoRide (pdo_mysql, intl, gd, zip, mongodb, etc)
- ✓ Préparation de la compilation des assets
- ✓ Inclut Composer, permettant d'installer les dépendances Symfony même en mode production.
- ✓ Gérer la création des dossiers nécessaires (var/cache, var/log, public/uploads)
- ✓ Copie de la configuration personnalisée de Nginx, adaptée à Symfony(rooting,requêtes,sécurité,gestion des fichier statiques).»
- ✓ Ajoute un scrypte d'entré (entrypoint.sh) qui lance Nginx et PHP-FPM, et permet d'éxecuter des scripts d'initialisation (ex:migration auto,nettoyage cache...).

5. Déploiement sur Fly.io

Avant de procéder au déploiement de l'application, j'ai mis à jour les secrets via la CLI Fly.io :
flyctl secrets set DATABASE_URL="…"
flyctl secrets set APP_DEBUG=0
flyctl secrets set APP_ENV=prod

Procédure de déploiement :

- Construction du conteneur optimisé :
  docker build -t ecoride-app .
- Déploiement de l'application :
  fly deploy
- Ouverture du terminal du conteneur déployé :
  fly ssh console
- Migration de la base de données via Doctrine :
  php bin/console doctrine:migrations:migrate --no-interaction

6. Problèmes rencontrés & solutions

Premier test : impossible de se connecter en tant qu'utilisateur.
Après recherche dans la documentation Symfony (lien), j'ai compris que les requêtes ne passaient pas jusqu'à mon application.
J'ai donc utilisé la méthode `setTrustedProxies()` pour permettre à mon application de recevoir les requêtes à travers le proxy inverse de Fly.io.

Deuxième test : accès à la page admin impossible.
J'ai découvert que ma base de données NoSQL (MongoDB, initialement sur Atlas) ne se connectait pas à Symfony. Problème : Atlas fonctionne en IPv4, Fly.io en IPv6 : la connexion était impossible. J'ai donc migré MongoDB sur Railway pour rassembler mes bases de données sur le même service.

Troisième test : MongoDB ne se connectait toujours pas.
J'ai vérifié la connexion via Railway, mais l'application était rejetée lors de la connexion. Par manque de temps, j'ai décidé d'adapter mon code pour qu'il fonctionne uniquement avec la base de données relationnelle.