

DOSSIER TECHNIQUE & GESTION DE PROJET – ECORIDE

Projet réalisé dans le cadre de l'ECF – Développeur
Web Full Stack

Auteur : **Othmane Lecoeuvre**
Année : 2025/2026

SOMMAIRE

1. Introduction
2. Architecture de l'application
 - 2.1 Architecture générale
 - 2.2 Organisation des fichiers
3. Choix technologiques
4. Modèle Conceptuel de Données (MCD)
5. Diagrammes UML
 - 5.1 Diagramme de Cas d'Utilisation (Use Case)
 - 5.2 Diagramme de Séquence (Participation à un trajet)
 - 5.3 Diagramme de Classes UML
6. Déploiement de l'application
 - 6.1 Étapes :
 - 6.2 Choix de l'hébergement : Fly.io
 - 6.3 Procédure de déploiement sur Fly.io
 1. Installation de Flyctl
 2. Connexion au compte
 3. Initialisation du projet
 4. Déploiement
 - 6.4 Structure des fichiers de déploiement
 1. Dockerfile
 2. fly.toml
 - 6.5 Mise à jour de l'application
7. Conclusion

1. Introduction

Cette documentation technique présente :

- l'architecture logicielle
- les choix technologiques
- le modèle de données
- les diagrammes UML officiels
- l'organisation du code
- les processus de déploiement

Elle constitue la référence technique permettant de comprendre, maintenir et faire évoluer EcoRide.

2. Architecture de l'application

2.1 Architecture générale

EcoRide repose sur une structure claire et modulaire :

Front-end :

- HTML5
- CSS3 (components + responsive)
- JavaScript (modules, fetch API)

Back-end :

- PHP procédural découpé par endpoints
- Réponses **JSON**
- Sécurisation via PDO & requêtes préparées

Base de données relationnelle :

- MySQL

Base NoSQL (prévue) :

- MongoDB (logs & monitoring)

2.2 Organisation des fichiers

Arborescence :

```
ECORIDE_PROJECT/  
| — index.html  
| — details.html  
| — login.html  
| — signup.html  
| — history.html  
| — user-space.html  
| — employee.html  
| — admin.html  
|  
| — assets/  
| — css/  
| — js/  
| — php/  
|   | — database.php  
|   | — get_trips.php  
|   | — get_user_data.php  
|   | — update_user_data.php  
|   | — join_trip.php  
|   | — end_trip.php  
|   | — ...
```

3. Choix technologiques

- **HTML / CSS / JavaScript Vanilla**
→ léger, rapide, aucun framework inutile
- **PHP 8 procédural**
→ simple et efficace pour API
- **MySQL**
→ relationnel, structure claire
- **MongoDB (optionnel)**
→ pour logs futur dashboard admin
- **XAMPP**
→ environnement simple

4. Modèle Conceptuel de Données (MCD)

Entités :

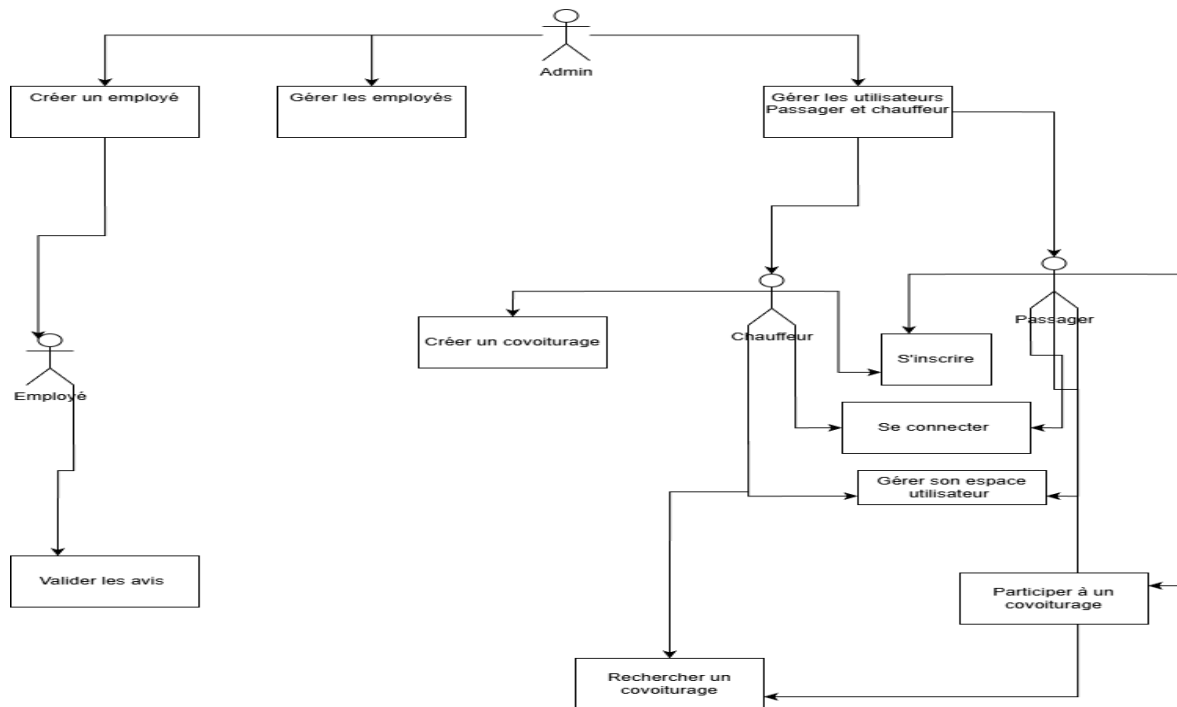
- Utilisateur
- Rôle
- Voiture
- Covoiturage
- Participation
- Avis

Relations :

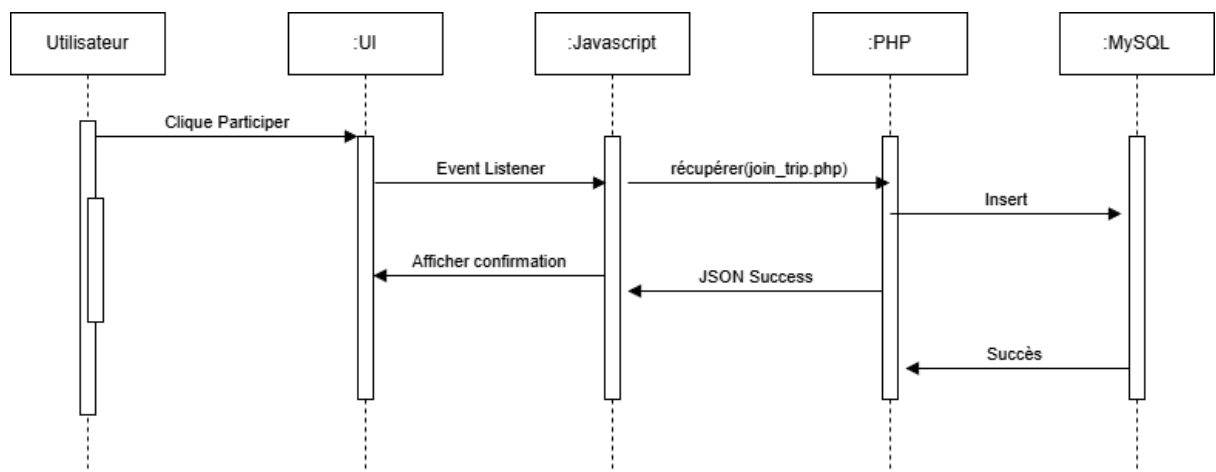
- Utilisateur 1—N Voiture
- Utilisateur 1—N Covoiturage (chauffeur)
- Utilisateur N—N Covoiturage via Participation
- Covoiturage 1—N Avis

5. Diagrammes UML

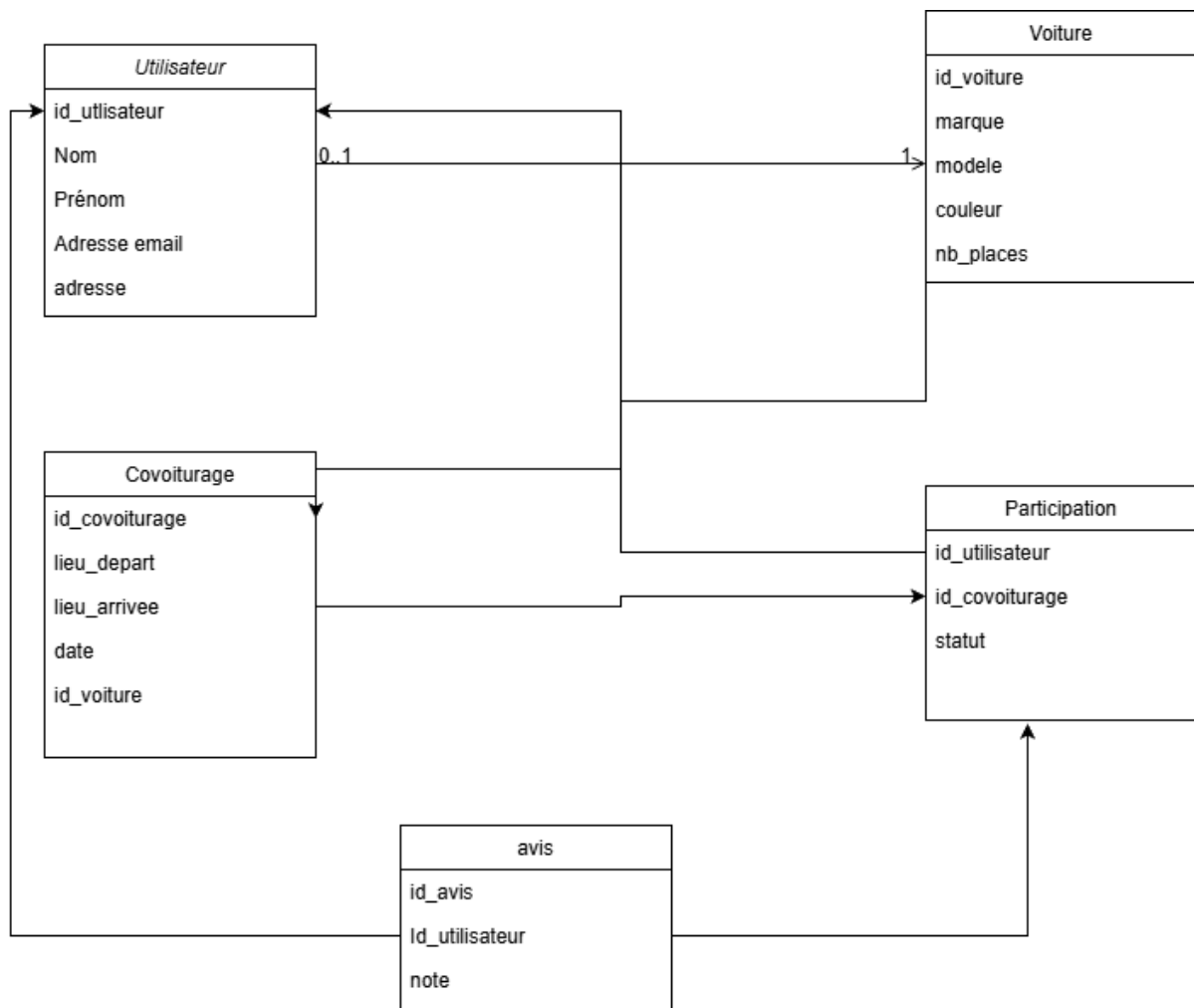
5.1 Diagramme de Cas d'Utilisation (Use Case)



5.2 Diagramme de Séquence (Participation à un trajet)



5.3 Diagramme de Classes UML



6. Déploiement de l'application

6.1 Étapes :

1. Installer XAMPP
2. Démarrer Apache + MySQL
3. Copier le dossier dans `htdocs/`
4. Importer le fichier `ecoride.sql`
5. Configurer `php/database.php`
6. Accéder via :

http://localhost/eecoRide_Project/

6.2 Choix de l'hébergement : Fly.io

Pour le déploiement de l'application EcoRide, le choix s'est porté sur **Fly.io**, une plateforme moderne permettant d'exécuter des applications Web dans des environnements isolés (VM ou conteneurs) à proximité géographique de l'utilisateur.

Les avantages principaux :

- Hébergement gratuit ou très peu coûteux
- Déploiement basé sur Docker
- Support natif de PHP
- Déploiement depuis Git ou via CLI
- Auto-scaling et monitoring intégrés
- Simplicité de configuration (`fly.toml`)

Cela répond parfaitement aux contraintes du projet EcoRide :

simplicité, rapidité de mise en ligne et hébergement adapté à une application PHP+JS.

6.3 Procédure de déploiement sur Fly.io

Le déploiement s'effectue en ligne de commande grâce à l'outil **flyctl**.

1. Installation de Flyctl

```
iwr https://fly.io/install.ps1 -useb | iex #  
PowerShell
```

2. Connexion au compte

```
flyctl auth login
```

3. Initialisation du projet

Depuis la racine du projet :

```
flyctl launch
```

Fly.io génère :

- Un fichier **fly.toml**
- Un **Dockerfile** (ou il détecte automatiquement le site PHP)
- Un nom d'application
- La région de déploiement (Paris : cdg)

4. Déploiement

```
flyctl deploy
```

Après le build et le push de l'image Docker, le service devient accessible via une URL du type :

<https://ecoride.fly.dev>

6.4 Structure des fichiers de déploiement

1. Dockerfile

Exemple minimal pour une application PHP :

```
FROM php:8.2-apache

COPY . /var/www/html/

EXPOSE 8080

CMD ["apache2-foreground"]
```

2. fly.toml

Fly.io génère automatiquement un fichier similaire à :

```
app = "ecoride"

primary_region = "cdg"

[http_service]

  internal_port = 8080

  force_https = true

  auto_start_machines = true

  auto_stop_machines = true
```

6.5 Mise à jour de l'application

En cas de changement dans la branche main :

```
git pull origin main

flyctl deploy
```

7. Conclusion

Cette documentation décrit l'ensemble du fonctionnement technique du projet : architecture, conception, base de données, UML, API PHP, intégration front/back.