Bienvenu sur ecoriDE 🍁

EcoRide a pour mission de réduire l’impact environnemental des trajets grâce au covoiturage.

Favre-Bonvin Yoann

# RESUME DU PROJET

**EcoRide** est une jeune startup française fondée avec une ambition claire : **réduire l’impact environnemental des déplacements quotidiens** en promouvant une solution de **covoiturage écologique**. Consciente des enjeux climatiques actuels, l’entreprise souhaite offrir une **alternative durable et économique** à l’usage individuel de la voiture.

La problématique centrale à laquelle EcoRide souhaite répondre est :

* **Comment rendre le covoiturage plus accessible, sécurisé et attractif pour les usagers soucieux de l’environnement ?**

Pour répondre à cette problématique, EcoRide a exprimé le besoin de **développer une application web moderne**, intuitive et éco-conçue, permettant de :

* Rechercher, proposer et réserver des trajets en covoiturage. Offrir une expérience fluide tant pour les passagers que les conducteurs.
* Filtrer les trajets selon des critères environnementaux (véhicule électrique principalement)
* Gérer les profils utilisateurs, les crédits, et la participation aux trajets.
* Mettre en avant une **identité visuelle alignée avec les valeurs écologiques de la marque** (couleurs, icônes, ergonomie).

Le projet est piloté par **José**, directeur technique, qui assure la supervision fonctionnelle et technique. Il souhaite une plateforme robuste, en prenant en compte les besoins fonctionnels détaillés (interfaces, filtres, espace utilisateur, espace administrateur, etc.).

Solution mise en œuvre :

* **Frontend** développé en HTML/CSS/JS avec Bootstrap, en responsive design.
* **Backend** en PHP structuré en **programmation orientée objet (POO)**, séparé en contrôleurs, services et repositories.
* Base **relationnelle (MySQL)** pour les trajets, utilisateurs, avis…
* Base **NoSQL (MongoDB)** pour enregistrer les logs de recherche.

# INTRODUCTION

En introduction, je commencerai par parler un peu de moi. En effet il s’agit de mon 1er projet dans le domaine du développement web puisque je suis en reconversion professionnelle après près de 20 ans dans le commerce.

Pour en revenir à ce projet, ma principale difficulté, au-delà de l’aspect technique puisque je débute dans ce domaine, a été la gestion du projet et plus particulièrement de l’organisation. En effet je suis partie un peu dans tous les sens sans avoir une organisation bien définie ; ce qui a eu pour conséquence une perte de temps dans la conception.

# ANALYSE

* Comment ai-je abordé ce projet ?

Après une lecture complète du sujet, j’ai identifié le nombre d’intervenant (persona) et les cas d’utilisations différents dans ce projet ce qui m’a conduit à la création du diagramme de cas d’utilisation (annexe 1).

* Comment ai-je convertie les besoins « non techniques » en « techniques » ?

Pour répondre à cette question, il m’a fallu réaliser un diagramme de classe (annexe 2), un diagramme de séquence ainsi qu’un schéma MVC. Ces derniers permettent :

* D’analyser et structurer les données : il permet de visualiser les entités (tables) du système (ex. : Utilisateur, Covoiturage, Véhicule) et leurs relations logiques. Cela évite d’oublier des données essentielles ou de créer une base incohérente.
* Faciliter la compréhension du système : C’est un outil de communication entre les développeurs, les chefs de projet, les clients et tous les acteurs techniques. On y voit rapidement : ce qui compose le système, comment les données interagissent et quelles sont les dépendances.
* Préparer la base de données relationnelle : création des tables, définition des clés primaires (PK) et étrangères (FK). (annexe 3)

En résumé ce diagramme est un plan de fondation de l’application : cela permet de voir la logique des données et surtout de repérer d’éventuels oublies.

Une fois ces étapes faites, il est temps de se plonger dans la conception de charte graphique puis par la suite les wireframes et finir par les mockups. Le tout en version desktop et mobile (responsive).

* Comment ai-je défini ma charte graphique ?

Pour répondre à cette question, il faut reprendre la description du projet par le directeur technique. Afin de se différencier de ces concurrents (blablacar par exemple), il souhaite avoir une approche écologique du covoiturage en mettant en avant les trajets écologiques (voitures électriques 🌿 Éco).

Je suis donc partie sur les axes principaux qui définissent le caractère « écologique » dans la société, à savoir : les couleurs et la typographie.

* Pour les couleurs, l’écologie est très souvent associée à la couleur verte (qui rappelle la nature), le beige et le marron : j’ai décidé d’utiliser principalement le vert et le beige pour des soucis de contraste (notamment pour les daltoniens qui peuvent être gênés avec le marron et le vert), j’ai quand même mis quelques touches de marron à certains endroits. L’avantage aussi du beige est que cela permet de moins consommer d’énergie sur les écrans
* Pour la typographie, je suis partie sur le GARAMOND qui permet de réduire de 24% la quantité d’encre (en cas d’impression).

Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.

Charte Graphique

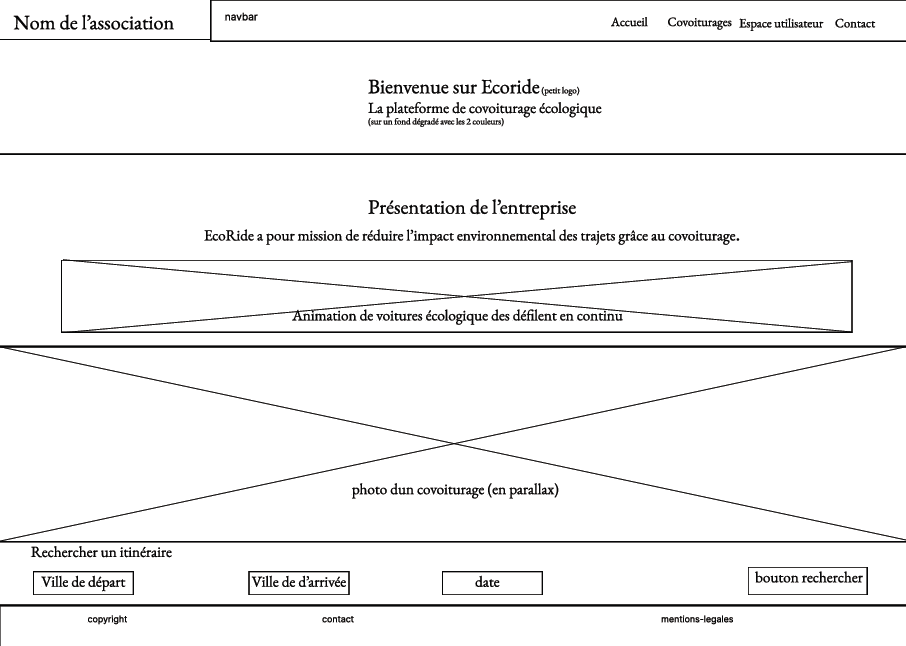
* Partie maquettage au format desktop et responsive (mobile) est à retrouver en intégralité en annexe.

Pour cette partie, il fallait répondre aux cahiers des charges fournis par le directeur technique, à savoir :

* La page d’accueil :
* une navbar avec les éléments suivants « accueil », « covoiturage », « connexion » (que j’ai renommé « espace utilisateur » qui me paraissait plus explicite) et « contact ».
* une présentation rapide de l’entreprise avec des photos.
* Une barre de recherche d’un itinéraire
* Un footer avec le mail et un lien vers les mentions légales.

Une image contenant texte, capture d’écran, habits, Visage humain

Le contenu généré par l’IA peut être incorrect.Une image contenant texte, capture d’écran, Visage humain, personne

Le contenu généré par l’IA peut être incorrect.Une image contenant texte, reçu, ligne, diagramme

Le contenu généré par l’IA peut être incorrect.

mockup responsive accueil

mockup accueil

wireframe accueil

wireframe responsive accueil

# CONCEPTION

Pour la réalisation de ce projet, il a fallu définir l’environnement de travail, à savoir :

* Le poste de travail :
* Visual Studio Code version 1.99.3 : éditeur de code léger et gratuit. Il permet l’utilisation de plusieurs langages (HTML, CSS, JS, PHP, …). Il est équipé d’une interface moderne avec auto-complétion, debug et terminal intégré. Il est léger et rapide et se lie facile avec le repos GitHub. (Voir annexe n°4)
* PHP version 8.X : Langage côté serveur principal, utilisé pour gérer la logique métier. (Voir annexe n°5)
* MySQL version 8.X : Base de données relationnelle pour stocker les utilisateurs et les trajets
* MongoDB version 7.X : Base NoSQL pour enregistrer les recherches des utilisateurs (log / veille)
* JavaScript (Vanilla) version ES6+ : Pour dynamiser le front avec des appels AJAX sans rechargement de page
* HTML / CSS : Pour la structure et le style des pages (Voir annexe n°4)
* Bootstrap 5.X : Framework CSS pour accélérer la création d’une interface responsive (Voir annexe n°4)
* Docker version 24.X : Conteneurisation de l’environnement pour faciliter le déploiement (Voir annexe n°6)
* Composer version 2.X : Gestionnaire de dépendances PHP pour autoloader les classes (PSR-4) (Voir annexe n°7)
* GitHub : Suivi de version du projet et hébergement du code (voir annexe n°8)
* Heroku : Hébergement du site relié au repo GitHub. (Voir annexe n°9)
* Structure logicielle :
* Architecture MVC partielle, avec séparation entre :
* Contrôleurs (gestion des requêtes)
* Services (logique métier)
* Repositories (accès aux données)
* Utilisation de l’**autoload PSR-4** via Composer pour charger dynamiquement les classes PHP.
* Conteneurisation :

L’ensemble du projet est conteneurisé avec Docker :

* Un service php-apache
* Un service mysql avec initialisation automatique de la base
* Un service mongo pour les logs NoSQL

Cela garantit une portabilité du projet sur tout environnement compatible Docker, sans conflits de version ou dépendances.

# DEVELOPPEMENT

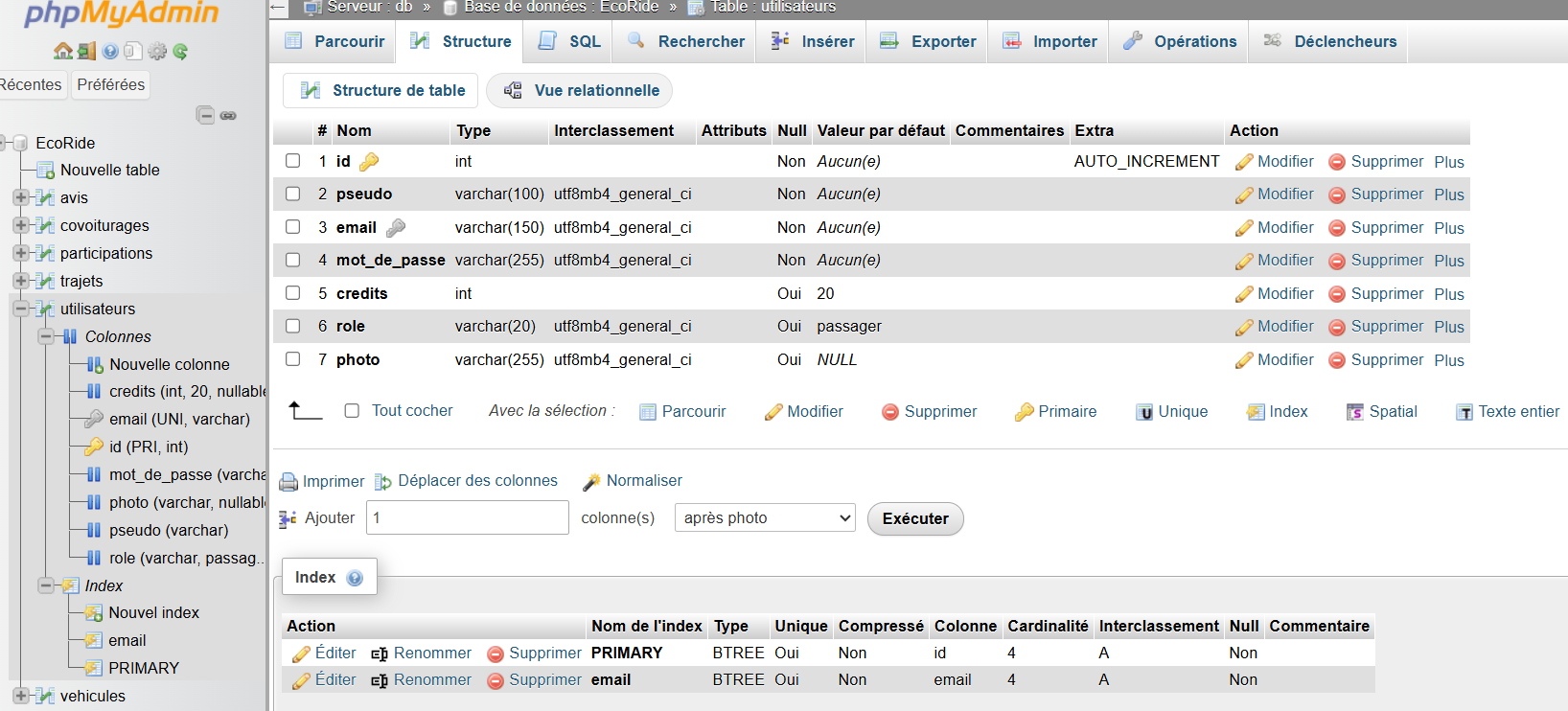
3.1 Données – Modélisation, scripts et jeux de données

* Mise en place de la base de données relationnelle (MySQL)

L’application repose sur une base de données relationnelle MySQL nommée EcoRide, contenant les tables suivantes nécessaire à la gestion de l’application :

* AVIS : enregistre les retours des passagers sur les trajets
* PARTICIPATIONS : permet d’enregistrer les demandes de réservation
* TRAJETS : représente les trajets proposés ou réservés
* UTILISATEURS : stocke les informations des utilisateurs inscrits
* VEHICULES : permet de stocker les différents véhicules
* Scripts de création de base

Ces tables ont été créées automatiquement à l’aide d’un script init.sql, placé dans le dossier Docker docker-entrypoint-initdb.d/. Le script est exécuté automatiquement lors de l’initialisation du conteneur MySQL.



* Jeu de données utilisateur (test)

Pour tester l’application, des utilisateurs fictifs ont été insérés automatiquement via la formule suivante :

INSERT INTO utilisateurs (pseudo, email, mot\_de\_passe, crédit, rôle, photo) VALUES



* Intégration de MongoDB (NoSQL)

Une base MongoDB complémentaire a été intégrée au projet pour **enregistrer les recherches effectuées par les utilisateurs**, sous forme de documents JSON.

Chaque recherche utilisateur est stockée avec l’IP, la date et les termes de recherche :

Une image contenant texte, capture d’écran, logiciel

Le contenu généré par l’IA peut être incorrect.

* Objectifs atteints grâce à cette architecture
* Organisation **claire et normalisée des données relationnelles**
* Enregistrement des **logs non structurés via MongoDB**
* **Complémentarité SQL + NoSQL** dans une même application, via Docker

3.2 Développement Front-end / Back-end

* Front-end – Interface utilisateur

L'interface utilisateur a été développée en **HTML / CSS / JavaScript**, avec l’aide du framework **Bootstrap** pour garantir un rendu responsive sur desktop et mobile. (voir annexe).

Principales caractéristiques :

* **Barre de navigation fixe** avec pages publiques (accueil, inscription, connexion)
* **Formulaire de recherche** dynamique via **JavaScript + AJAX (Fetch)**
* **Affichage** des trajets disponibles
* Design en lien avec la thématique **écoresponsable** (vert, beige, icône voiture électrique)
* Utilisation de la librairie **AOS.js** pour les animations



* Back-end – Programmation orientée objet (POO)

Le projet a été entièrement restructuré en **POO** afin de séparer les responsabilités. L’organisation suit un modèle **Controller – Service – Repository** :

* Controller : Reçoit les requêtes, déclenche les actions à effectuer
* Service : Contient la logique métier (ex : validation, traitement)
* Repository : Gère les requêtes SQL et l’accès aux base de données.

Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.

* Communication AJAX

Les formulaires utilisent **AJAX (Fetch API)** pour interroger le back-end sans recharger la page.  
Le fichier *index.js* intercepte la soumission, envoie les données à *recherche.php*, puis affiche dynamiquement les résultats. (voir annexe)

Cela permet une expérience fluide et interactive, essentielle pour une application moderne.

3.3 Sécurité

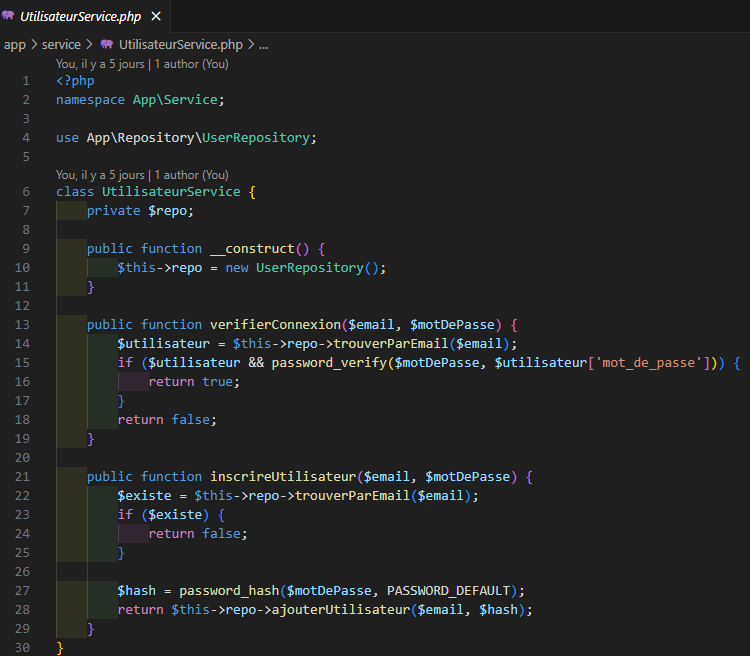
* Authentification sécurisée

L’application gère les connexions via des sessions PHP. Lorsqu’un utilisateur s’authentifie :

* La session est ouverte avec *session\_start()* dès le début du script.
* L’identifiant utilisateur est enregistré dans *$\_SESSION['user\_id'],* ce qui permet de sécuriser l’accès aux pages privées.

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

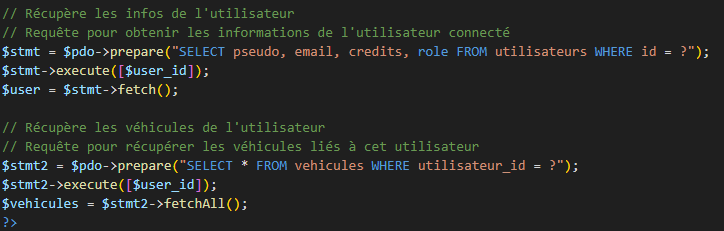
Le contenu généré par l’IA peut être incorrect.

* Les mots de passe sont **hachés avec *password\_hash()*** lors de l'inscription.
* Lors de la connexion, la vérification se fait avec *password\_verify().*

Ces pratiques évitent le stockage de mots de passe en clair et renforcent la sécurité contre les attaques.

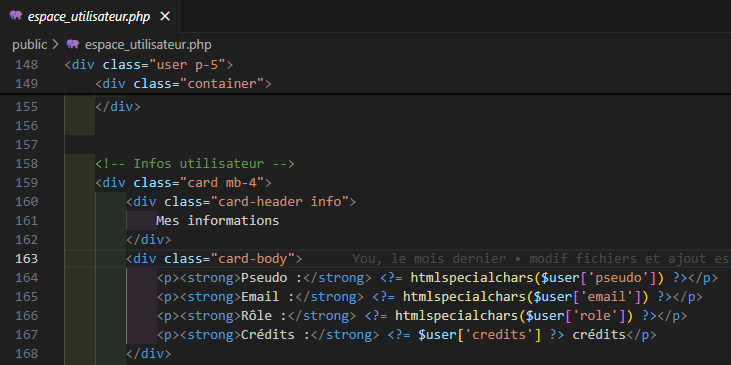
* Protection contre les injections SQL

L’accès à la base de données utilise PDO avec des requêtes préparées :



* Validation des données
* Les formulaires HTML valident les champs côté client *(ex : Required, type=email)*
* Le back-end vérifie également les données avant de les utiliser : il vérifie les champs vides, filtre avec *htmlspecialchars()* pour éviter les attaques XSS, vérifie du format des emails.
* Les mots de passe et configurations ne sont pas exposés directement dans les scripts publics.

Tout ceci contribue à conserver des données propres et évite les attaques les plus fréquentes.



Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Le contenu généré par l’IA peut être incorrect.

* Sécurité MongoDB

Les interactions avec MongoDB sont limitées à : des insertions (*insertOne()*) et des tests de lecture sécurisés (*find()*). Les logs MongoDB ne contiennent aucune donnée sensible.

# ANNEXES

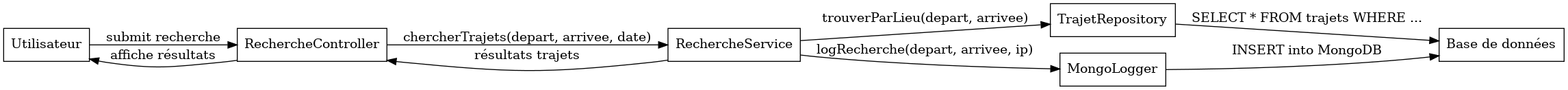
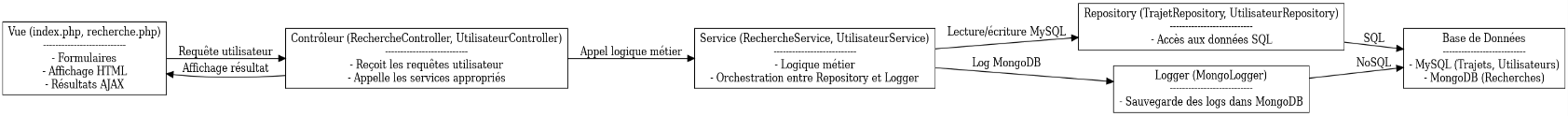
Une image contenant texte, capture d’écran, diagramme, ligne

Le contenu généré par l’IA peut être incorrect.

Annexe 1: Schéma du diagramme d’utilisation

Une image contenant texte, diagramme, cercle, dessin

Le contenu généré par l’IA peut être incorrect.



Annexe 2 : Diagramme de classe

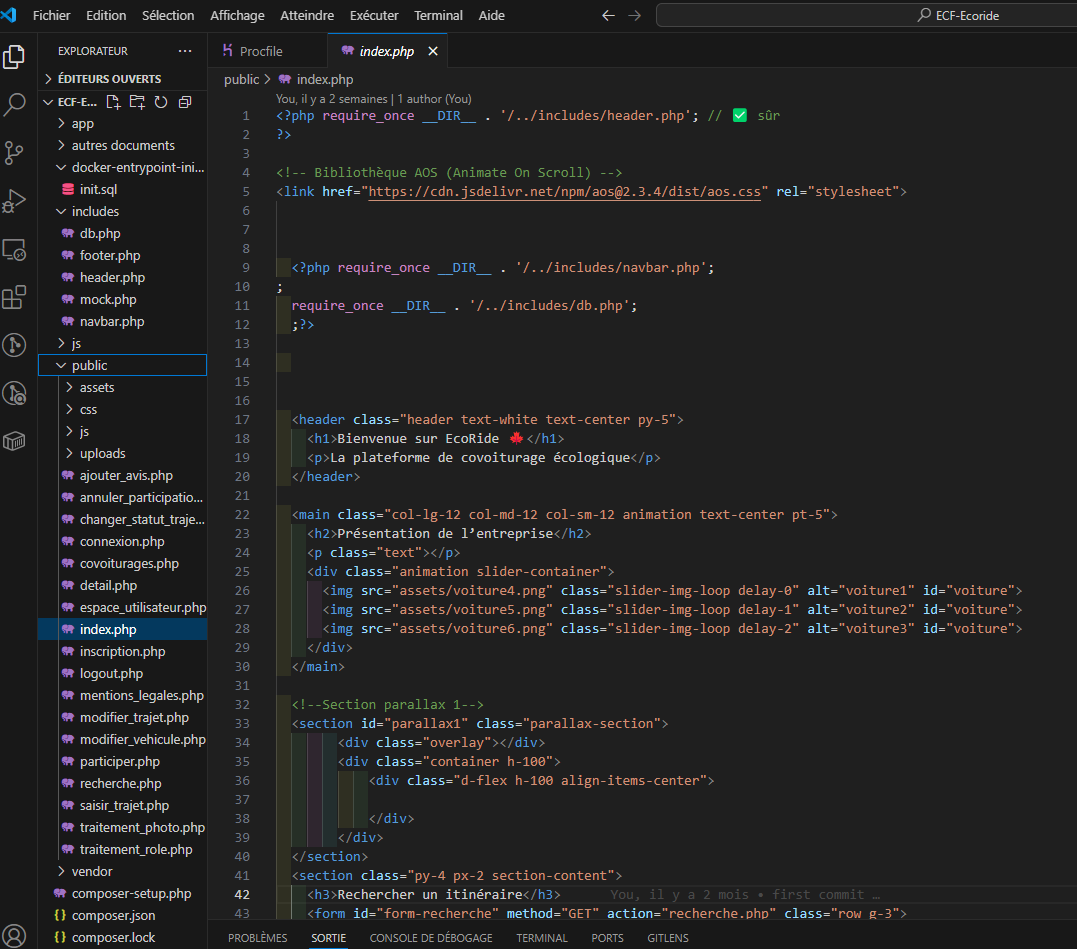
Annexe 2 ter : diagramme MVC

Annexe 2 bis : diagramme de séquence recherche

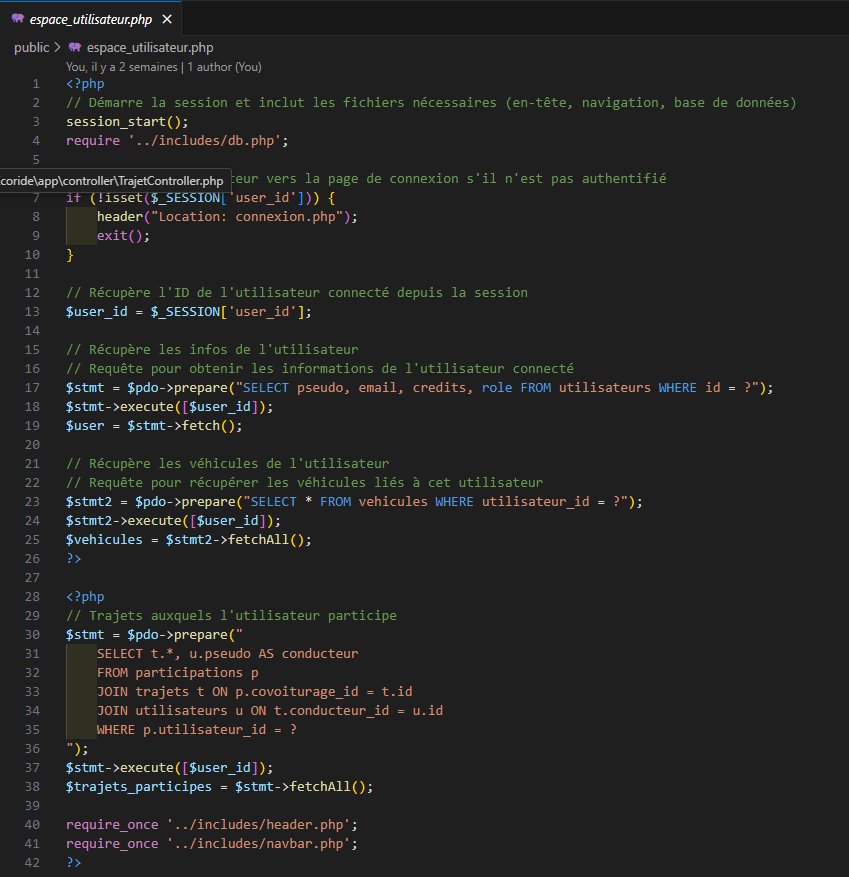
Une image contenant texte, diagramme, Plan, schématique

Le contenu généré par l’IA peut être incorrect.

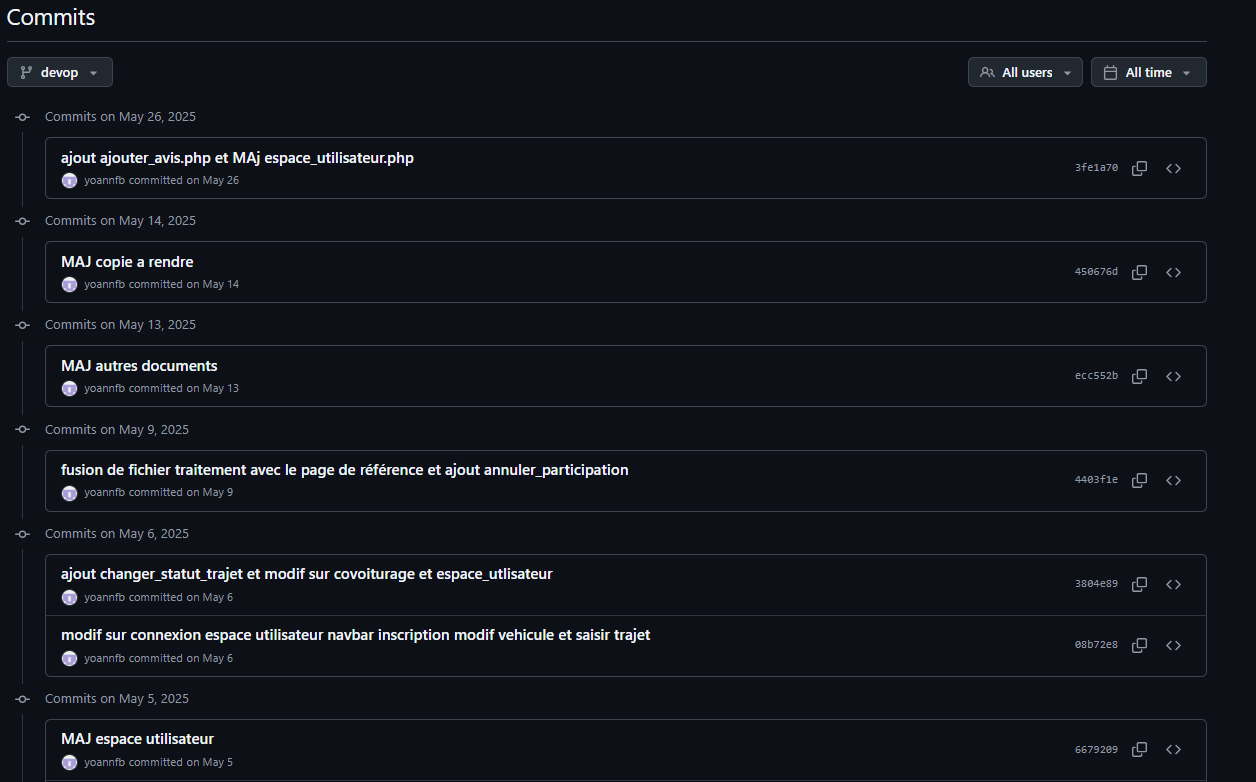
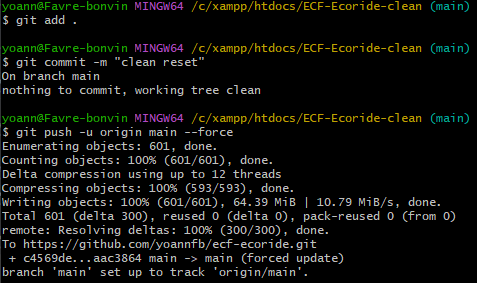
Annexe 3: Schéma MCD de la base de données relationnelle



Annexe 4: structure du projet



Annexe 5: exemple code PHP

Une image contenant capture d’écran, texte, logiciel, Logiciel multimédia

Le contenu généré par l’IA peut être incorrect.

Annexe 8: exemple du repo GitHub

Une image contenant texte, capture d’écran, logiciel, Page web

Le contenu généré par l’IA peut être incorrect.

Annexe 9: Déploiement sur Heroku