DOCUMENTATION TECHNIQUE

Titre du projet : Application Web EcoRide
Nom du Titre Professionnel : Développeur Web et Web Mobile
RIGAUT William
Date de réalisation : 19/04/2025
Organisme de formation : Studi

1. Introduction / Réflexions Technologiques Initiales

L'objectif de ce projet était de développer une application web complète de covoiturage écologique, nommée EcoRide, conformément au cahier des charges fourni pour l'Évaluation en Cours de Formation (ECF) du titre professionnel Développeur Web et Web Mobile.

Dès le départ, l'orientation principale était de construire une application web dynamique nécessitant une partie frontend (interface utilisateur) et une partie backend (logique serveur et gestion des données). Il était clair qu'une base de données relationnelle serait nécessaire pour stocker les informations structurées (utilisateurs, trajets, réservations). L'énoncé imposait également l'utilisation d'une base de données NoSQL et le déploiement final de l'application.

Mon approche initiale s'est portée sur des technologies courantes et adaptées au développement web moderne, en particulier dans l'écosystème JavaScript, compte tenu de sa flexibilité pour le développement full-stack.

---

2. Justification des Choix Techniques Finaux

Plusieurs technologies ont été sélectionnées pour la réalisation de ce projet :

- **Backend : Node.js avec le framework Express.js**
    - *Pourquoi ?* Node.js permet d'utiliser JavaScript côté serveur, créant un environnement de développement cohérent avec le frontend. Express.js est un framework minimaliste, populaire et très bien documenté pour Node.js, idéal pour construire rapidement des API RESTful nécessaires à l'application. Sa nature asynchrone est adaptée à la gestion des requêtes web.

- **Frontend : HTML5, CSS3, JavaScript (Vanilla)**
    - *Pourquoi ?* Pour l'interface utilisateur, l'utilisation des standards du web (HTML, CSS, JS) sans framework frontend complexe (comme React, Vue ou Angular) a été jugée suffisante pour répondre aux exigences des Users

Stories. Cela permet de se concentrer sur les fondamentaux du développement web.

- **Base de Données Relationnelle : MySQL**
    - *Pourquoi ?* MySQL est un système de gestion de base de données relationnelle open-source, très utilisé, fiable et performant pour les données structurées comme les utilisateurs, véhicules, trajets, participations et avis. Sa syntaxe SQL est standard et il s'intègre bien avec Node.js via des pilotes comme `mysql2`.

- **Base de Données Non Relationnelle : MongoDB**
    - *Pourquoi ?* L'utilisation d'une base NoSQL étant requise, MongoDB (base de données orientée document) a été choisi car il est très populaire dans l'écosystème Node.js et se manipule facilement avec des objets JavaScript/JSON.
    - *Utilisation :* MongoDB a été utilisé pour **enregistrer les logs d'application**. Ce choix se justifie car les logs peuvent être nombreux, arriver fréquemment, et leur structure peut varier (erreurs, succès de connexion, actions admin...). MongoDB gère bien ce type de données sans nécessiter de schéma fixe et est performant pour les écritures.

- **Sécurité Mots de Passe : Bcrypt**
    - *Pourquoi ?* La bibliothèque `bcrypt` est le standard actuel pour hacher les mots de passe de manière sécurisée. Elle intègre automatiquement un salage pour chaque mot de passe, le rendant résistant aux attaques par tables arc-en-ciel.

- **Gestion des Sessions : express-session**
    - *Pourquoi ?* `express-session` est le middleware standard et simple pour gérer les sessions utilisateur avec Express.js. Dans ce projet, les sessions sont stockées [par défaut en mémoire / dans MongoDB
    -
- **Graphiques : Chart.js**
    - *Pourquoi ?* Pour afficher les graphiques dans l'espace admin (US 13), Chart.js a été choisi car c'est une bibliothèque JavaScript gratuite, facile à

utiliser via un CDN, bien documentée, et offrant les types de graphiques (barres) nécessaires.

---

3. Configuration de l'Environnement de Travail

Le développement a été réalisé sur l'environnement suivant :

- **Système d'exploitation :** Windows 10
- **Éditeur de code :** Visual Studio Code
- **Node.js :** Version 22.14.0
- **npm :** Version 10.9.2
- **Base de données SQL :** MySQL Version 8.3.0
- **Base de données NoSQL :** MongoDB Version 8.0.8
- **Navigateur(s) de test :** Google Chrome
- **Modules Node.js clés `package.json` :** express, mysql2, mongodb, bcrypt, express-session.
- **Instructions d'installation locale :**

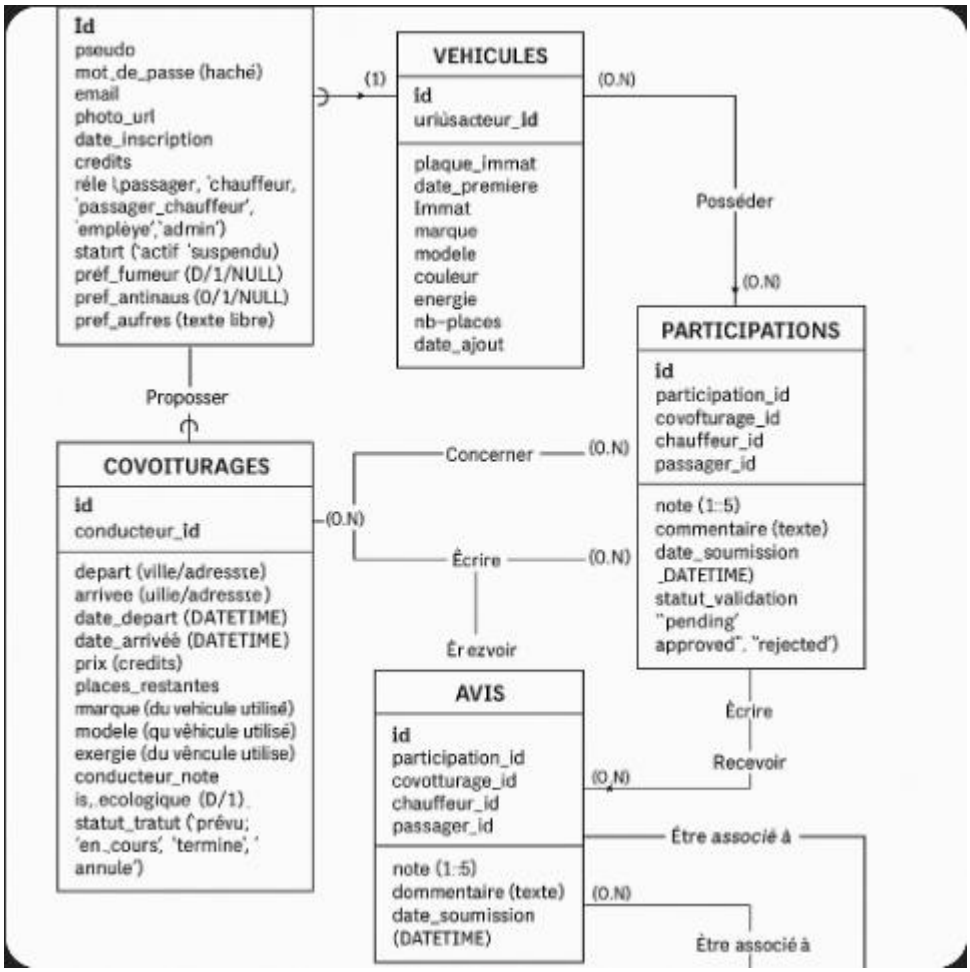
    1. Cloner le dépôt : `https://github.com/bibou911/EcoRide.git`
    2. Aller dans le dossier du projet : `cd ProjetEcoRide`
    3. Installer les dépendances Node.js : `npm install`
    4. S'assurer qu'un serveur MySQL est lancé et accessible.
    5. Créer une base de données MySQL nommée `ecoride`.
    6. Importer la structure des tables et éventuellement les données initiales en exécutant le fichier `ecoride.sql` fourni dans le dépôt.
    7. S'assurer qu'un serveur MongoDB est lancé et accessible sur l'adresse par défaut (`mongodb://localhost:27017`). La base `ecoride_logs` et la collection `logs` seront créées automatiquement.
    8. Lancer le serveur Node.js : `node server.js`
    9. Accéder à l'application dans un navigateur : `http://localhost:4000`

---

4. Modèle Conceptuel de Données (MCD) / Diagramme de Classe (SQL)

Ce diagramme présente la structure de la base de données relationnelle MySQL utilisée pour stocker les données principales de l'application (utilisateurs, trajets,

véhicules,participations, avis). Il montre les tables, leurs colonnes, les types de données, les clés primaires (PK), les clés étrangères (FK) et les relations (cardinalités) entre elles.

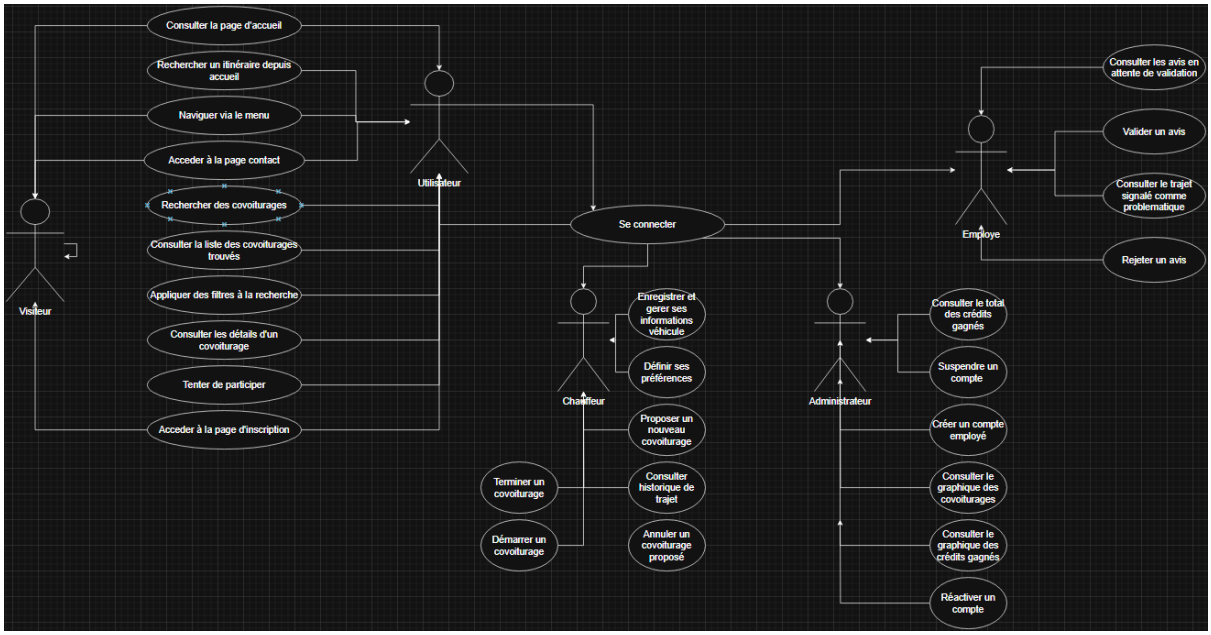


---

5. Diagramme d'Utilisation (Use Case)

Ce diagramme identifie les différents acteurs interagissant avec l'application EcoRide et les principales actions (cas d'utilisation) qu'ils peuvent réaliser.

- **Acteurs :** Visiteur, Utilisateur (Passager/Chauffeur), Employé, Administrateur.
- **Cas d'utilisation principaux :** Rechercher un trajet, Consulter les détails, S'inscrire, Se connecter, Gérer son profil/véhicules/préférences, Proposer un trajet, Participer à un trajet, Annuler une participation/un trajet, Démarrer/Terminer un trajet, Laisser un avis, Valider/Rejeter un avis, Consulter les trajets problématiques, Créer un compte employé, Suspendre un compte, Consulter les statistiques/graphiques.

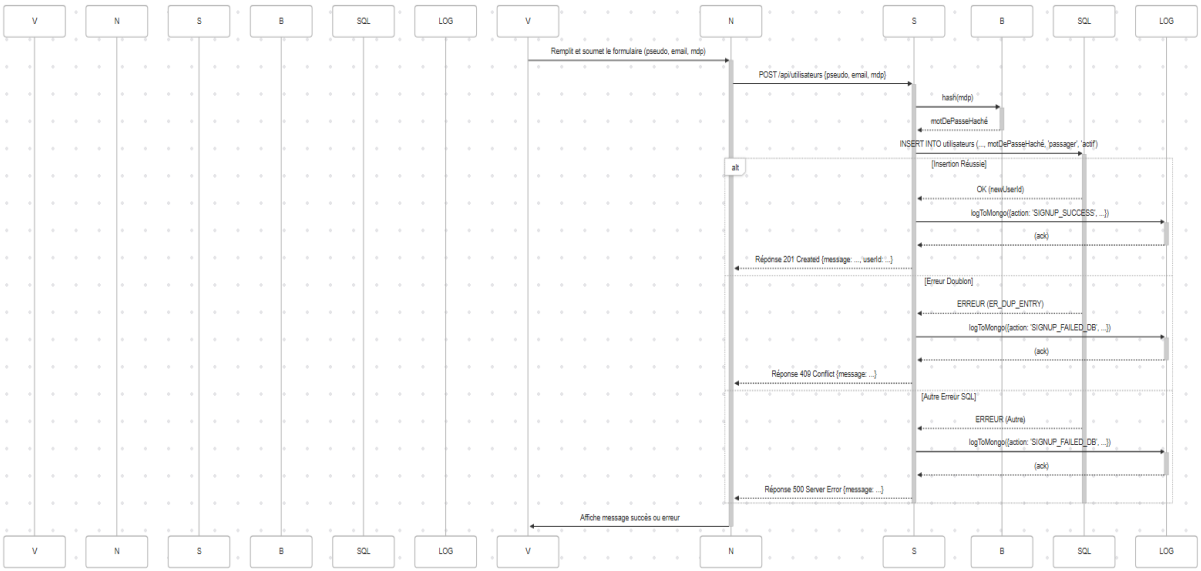


---

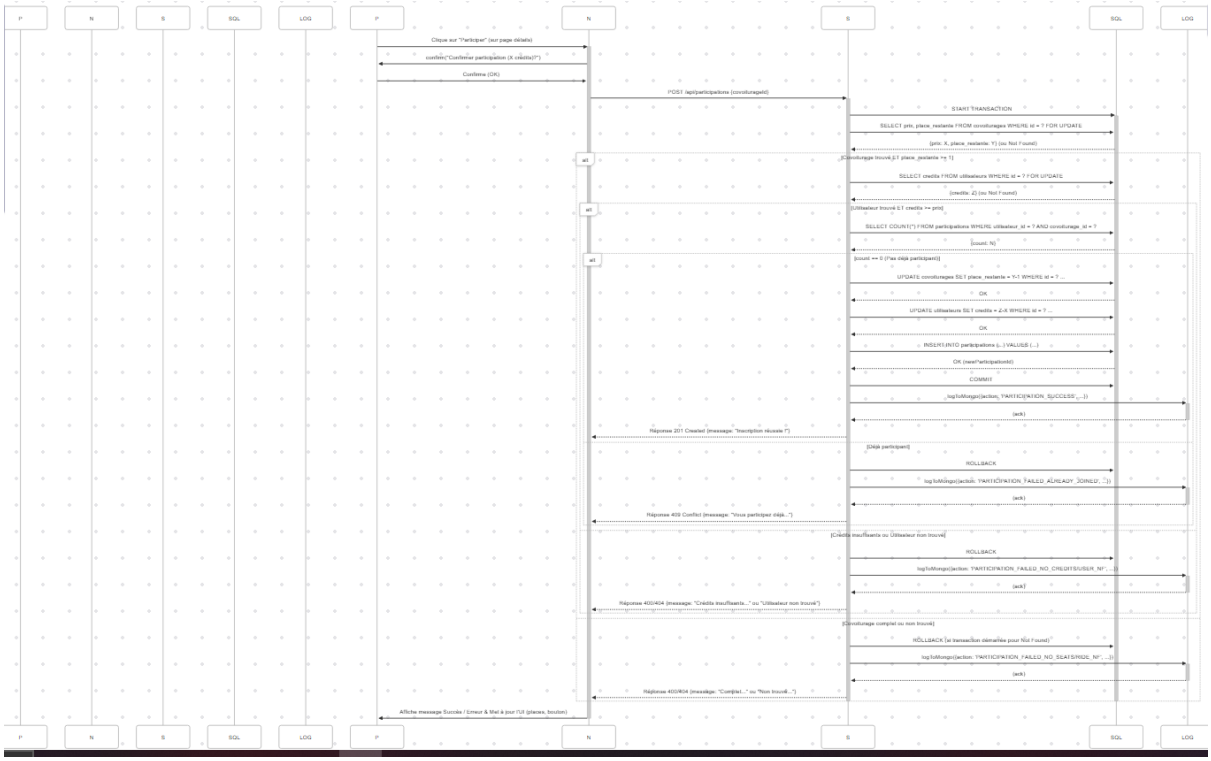
6. Diagramme de Séquence

Ces diagrammes illustrent les interactions détaillées entre les composants pour des scénarios spécifiques.

- **Cas d'usage 1 : Inscription d'un nouvel utilisateur**
    - Ce diagramme montre les étapes : l'utilisateur remplit le formulaire, le JS envoie les données à l'API `/api/utilisateurs`, le serveur valide, appelle bcrypt pour hacher le mot de passe, insère l'utilisateur dans MySQL, et renvoie une réponse au JS qui affiche un message.



- **Cas d'usage 2 : Participation à un covoiturage**
    - Ce diagramme montre les étapes : l'utilisateur clique sur "Participer", le JS demande confirmation, appelle l'API `/api/participations`, le serveur vérifie l'utilisateur, les places, les crédits (avec `SELECT ... FOR UPDATE`), met à jour les tables `covoiturages` et `utilisateurs`, insère dans `participations` (tout ça dans une transaction), commit, et renvoie une réponse au JS qui met à jour l'interface.



---

7. Documentation du Déploiement

[Section à remplir une fois le déploiement effectué]

- **Plateforme Choisie :** Render
- **Justification du Choix :** Offre un niveau gratuit ("Free tier") généreux pour les services web et certaines bases de données.
- Facile à utiliser, connexion directe avec GitHub pour les déploiements automatiques.
- Détection automatique des applications Node.js.
- Gestion simplifiée des variables d'environnement.
- Propose des bases de données managées (comme PostgreSQL, qui peut parfois remplacer MySQL, ou Redis).
- **Prérequis :** Avoir un compte sur Render.com.
- Avoir le code du projet sur un dépôt Git (comme GitHub, GitLab).
- **Configuration :**

    - Variables d'environnement Les variables d'environnement nécessaires (`MONGO_URI`, `MYSQL_HOST`, `MYSQL_USER`, `MYSQL_PASSWORD`, `MYSQL_DATABASE`, `PORT`, `SESSION_SECRET`) n'ont pas été mises dans le fichier `.env` (qui n'est pas envoyé sur Git). Elles ont été configurées directement dans l'interface de Render, dans la section "Environment" du service web. Les valeurs pour `MONGO_URI` et les variables `MYSQL_*` doivent pointer vers les bases de données **en ligne** (MongoDB Atlas et la base MySQL cloud), pas vers `localhost`.
    - Fichiers de configuration spécifiques : Aucun fichier de configuration spécifique à Render (ex: `render.yaml`) n'a été nécessaire, la détection automatique du service Node.js a suffi."
    - Commandes de build La commande de build par défaut (`npm install`) a été utilisée. Aucune commande de build supplémentaire n'était requise."
- **Étapes du Déploiement :** Connexion au tableau de bord Render.

Cliquer sur "New +" > "Web Service".

Connecter le dépôt GitHub contenant le projet EcoRide.

Configurer les paramètres du service :

- Nom du service (ex: ecoride-app).
- Région (ex: Frankfurt).
- Branche Git à déployer (ex: `main`).
- Runtime : Node.
- Commande de Start : `node server.js` (Très important de vérifier que c'est bien cette commande qui est indiquée).
- Plan : (Choisir le plan Free).

Ajouter les variables d'environnement (voir section précédente).

Cliquer sur "Create Web Service"

Attendre la fin du build et du déploiement.

- **Gestion des Bases de Données :**J’ai utilisé Atlas pour MongoBD et pour MYSQL j’ai utilisé railway
- **Lien vers l'application déployée :** [https://ecoride-app.onrender.com]