Documentation Technique

Projet EcoRide

Plateforme de covoiturage écologique

Décembre 2025

Développeur Web Full Stack

1. Introduction

J'ai développé EcoRide, une plateforme de covoiturage écologique répondant aux besoins de la startup EcoRide. L'objectif principal est de réduire l'impact environnemental des déplacements en encourageant le covoiturage, avec une attention particulière portée aux véhicules électriques.

L'application permet aux utilisateurs de rechercher des covoiturages, de proposer des trajets en tant que chauffeur, et de gérer leur compte avec un système de crédits. Un espace employé permet la modération des avis, tandis qu'un espace administrateur offre une vue statistique complète de la plateforme.

1.1 Objectifs du projet

- Créer une plateforme de covoiturage intuitive et responsive
- Mettre en avant les trajets écologiques (véhicules électriques)
- Implémenter un système de crédits pour les transactions
- Assurer la sécurité des données utilisateurs
- Proposer des espaces dédiés pour chaque type d'utilisateur

2. Réflexions initiales technologiques

J'ai choisi une stack technique moderne et robuste pour ce projet, en privilégiant des technologies éprouvées et bien documentées.

2.1 Choix du framework back-end : Symfony

J'ai opté pour Symfony 7 comme framework PHP pour plusieurs raisons :

- **Architecture MVC** : séparation claire des responsabilités (Modèle, Vue, Contrôleur) facilitant la maintenance
- **Sécurité intégrée** : système d'authentification robuste, protection CSRF, pare-feu configurable
- **ORM Doctrine** : gestion simplifiée de la base de données via des entités PHP
- **Système de formulaires** : validation côté serveur avec contraintes déclaratives
- **Moteur Twig** : templates sécurisés avec échappement automatique des données
- **Large communauté** : documentation complète et bundles disponibles

2.2 Choix du framework front-end : Bootstrap 5

J'ai utilisé Bootstrap 5 pour le développement front-end :

- Système de grille responsive intégré
- Composants UI prêts à l'emploi (modals, tabs, accordéons)
- Personnalisation facile via CSS custom properties
- Compatibilité navigateurs optimale

2.3 Base de données : MySQL/MariaDB

J'ai choisi MySQL/MariaDB comme système de gestion de base de données relationnelle pour sa fiabilité, ses performances et sa compatibilité native avec Doctrine ORM.

2.4 Gestion des assets : Webpack Encore

Webpack Encore permet de compiler et optimiser les assets CSS et JavaScript, facilitant l'intégration de la charte graphique personnalisée et la minification pour la production.

3. Architecture de l'application

3.1 Organisation des Contrôleurs

J'ai organisé les contrôleurs par domaine fonctionnel pour une meilleure lisibilité :

Contrôleur	Responsabilité
HomeController	Page d'accueil, présentation, contact
CovoiturageController	Recherche, détail, création, modification, participation
SecurityController	Inscription, connexion, déconnexion
EspaceUtilisateurController	Profil, véhicules, préférences, historique, avis
EmployeController	Dashboard employé, validation avis, signalement problèmes
AdminController	Gestion utilisateurs, employés, statistiques
ResetPasswordController	Réinitialisation du mot de passe par email

3.2 Organisation des Repositories

Les Repositories encapsulent la logique d'accès aux données. J'ai créé des méthodes métier spécifiques :

- **UtilisateurRepository** : findByFilters(), countByRole() pour la recherche et les statistiques
- **CovoiturageRepository** : findBySearch(), findCovoituragesPasses(), getTopChauffeurs()
- **ParticipationRepository** : calculerCreditsPlateforme(), getCreditsParMois()
- **AvisRepository** : findAvisEnAttente(), countAvisValidesAujourdHui()

3.3 Organisation des Templates Twig

J'ai structuré les templates avec un système d'héritage et de partials :

- **base.html.twig** : layout principal avec navbar et footer
- **partials/** : composants réutilisables (_header.html.twig, _footer.html.twig)
- **espace_utilisateur/** : templates avec sous-includes pour chaque onglet
- **employe/ et admin/** : espaces dédiés avec leurs propres layouts

4. Modèle Conceptuel de Données

J'ai conçu le modèle de données suivant pour répondre aux besoins fonctionnels de l'application.

4.1 Entités principales

Entité	Attributs principaux
Utilisateur	id, pseudo, email, password, credits, role (ENUM), photo, is_suspended, roles_system (JSON)
Vehicule	id, marque, modele, immatriculation, energie (ENUM), couleur, places_disponibles
Covoiturage	id, ville_depart, ville_arrivee, date_depart, prix, places_restantes, ecologique
Participation	id, utilisateur_id, covoiturage_id, confirme, credits_utilises
Avis	id, utilisateur_id, covoiturage_id, note, commentaire, statut (ENUM)
Preferences	id, utilisateur_id, accepte_fumeurs, accepte_animaux, preferences_personnalisees

4.2 Relations entre entités

- Utilisateur 1:N Vehicule — Un utilisateur peut posséder plusieurs véhicules
- Utilisateur 1:N Covoiturage — Un chauffeur peut créer plusieurs covoiturages
- Utilisateur N:M Covoiturage (via Participation) — Relation passager-trajet
- Covoiturage 1:N Avis — Un covoiturage peut avoir plusieurs avis
- Utilisateur 1:1 Preferences — Un chauffeur a une fiche de préférences

5. Sécurité de l'application

J'ai mis en place plusieurs mesures de sécurité pour protéger l'application et les données utilisateurs.

5.1 Authentification

- Hashage des mots de passe avec bcrypt (auto-detection Symfony)
- Validation stricte : 8 caractères min, majuscule, minuscule, chiffre, caractère spécial
- Système Remember Me avec cookie sécurisé (durée : 1 semaine)
- Réinitialisation de mot de passe par email avec token (durée : 1 heure)

5.2 Autorisation

- Système de rôles hiérarchique : ROLE_ADMIN > ROLE_EMPLOYE > ROLE_USER
- Contrôle d'accès par route via security.yaml (access_control)
- Attribut #[IsGranted] sur les contrôleurs et méthodes
- Vérification de propriété des ressources (véhicules, covoiturages)

5.3 Protection CSRF et données

- Token CSRF sur tous les formulaires sensibles
- Validation côté serveur avec isCsrfTokenValid()
- Échappement automatique des sorties par Twig
- Requêtes préparées via Doctrine (protection injection SQL)
- Upload sécurisé : validation type MIME, renommage unique, limite de taille

6. Fonctionnalités implémentées

US	Description	Statut
1	Page d'accueil avec présentation et recherche	✓ Complétée
2	Menu de navigation responsive	✓ Complétée
3	Vue des covoiturages avec recherche	✓ Complétée
4	Filtres avancés (écologique, prix, note)	✓ Complétée
5	Vue détaillée d'un covoiturage	✓ Complétée
6	Participation avec double confirmation	✓ Complétée
7	Création de compte avec 20 crédits	✓ Complétée
8	Espace utilisateur complet	✓ Complétée
9	Création de covoiturage (chauffeur)	✓ Complétée
10	Historique et système de notation	✓ Complétée
11	Espace employé (validation avis)	✓ Complétée
12	Espace administrateur (stats, gestion)	✓ Complétée

6.1 Fonctionnalités bonus

- Système de réinitialisation de mot de passe par email
- Upload et gestion de photos de profil
- Statistiques avancées dans l'espace administrateur
- Interface responsive avec Bootstrap 5
- Charte graphique personnalisée avec thème écologique

7. Documentation du déploiement

7.1 Prérequis serveur

- PHP 8.2+ avec extensions : pdo_mysql, intl, mbstring, zip, gd
- MySQL 8.0 ou MariaDB 10.6+
- Node.js 18+ et npm
- Composer 2.x
- Serveur web : Apache ou Nginx

7.2 Étapes de déploiement

1. Cloner le dépôt : git clone https://github.com/username/ecoride.git
2. Installer les dépendances : composer install --no-dev --optimize-autoloader
3. Configurer .env.local : APP_ENV=prod, DATABASE_URL, MAILER_DSN
4. Créer la BDD : php bin/console doctrine:database:create
5. Exécuter les migrations : php bin/console doctrine:migrations:migrate
6. Compiler les assets : npm install && npm run build
7. Vider le cache : php bin/console cache:clear --env=prod
8. Configurer les permissions : chown -R www-data:www-data var/ public/uploads/

— Fin de la documentation technique —