

Documentation de Gestion de Projet

Projet EcoRide
Plateforme de covoiturage écologique

Décembre 2025

1. Méthodologie de gestion de projet

J'ai adopté une approche agile inspirée de Scrum pour gérer ce projet, tout en l'adaptant au contexte d'un développeur solo. Cette méthodologie m'a permis de découper le travail en itérations courtes et de livrer des fonctionnalités de manière incrémentale.

1.1 Principes appliqués

- **User Stories** : J'ai découpé les besoins en 13 User Stories prioritaires
- **Sprints courts** : Chaque US représente un mini-sprint de développement
- **Livraison continue** : Chaque fonctionnalité est testée et validée avant la suivante
- **Adaptation** : J'ai ajusté mes priorités en fonction des difficultés rencontrées

1.2 Outils utilisés

- **Trello / Notion** : Kanban pour le suivi des tâches
- **Git / GitHub** : Gestion de versions et branches
- **VS Code** : Environnement de développement
- **Symfony CLI** : Serveur de développement local

2. Organisation du Kanban

J'ai organisé mon tableau Kanban avec 5 colonnes pour suivre l'avancement du projet :

Colonne	Description
Backlog	Toutes les fonctionnalités prévues, ordonnées par priorité
À faire (Sprint)	Fonctionnalités prévues pour le sprint en cours
En cours	Fonctionnalité actuellement en développement
À tester	Fonctionnalité développée, en attente de validation
Terminé (Dev)	Fonctionnalité validée, mergée dans la branche develop
Production	Fonctionnalité déployée sur la branche main

3. Stratégie de gestion Git

J'ai appliqué les bonnes pratiques Git avec une stratégie de branches structurée :

3.1 Structure des branches

- **main** : Branche de production, code stable et déployé
- **develop** : Branche de développement, intégration des fonctionnalités
- **feature/xxx** : Branches pour chaque nouvelle fonctionnalité

3.2 Workflow de développement

1. Je crée une branche feature depuis develop : git checkout -b feature/us9-creation-covoiturage
2. Je développe et commite régulièrement avec des messages clairs
3. Je teste la fonctionnalité localement
4. Je merge la feature dans develop : git merge feature/us9-creation-covoiturage
5. Après validation complète, je merge develop dans main

3.3 Convention de commits

J'ai utilisé des messages de commit descriptifs et structurés :

- feat: ajout de la recherche de covoiturages
- fix: correction du calcul des crédits
- style: amélioration du CSS de la page d'accueil
- docs: mise à jour du README

4. Planning de réalisation

J'ai organisé le développement en plusieurs phases, en priorisant les fonctionnalités essentielles :

Phase	User Stories	Durée estimée
Phase 1 - Base	US 1-2 : Accueil, Menu	~8h
Phase 2 - Covoiturages	US 3-5 : Recherche, Filtres, Détail	~12h
Phase 3 - Utilisateurs	US 6-8 : Participation, Inscription, Espace	~15h
Phase 4 - Chauffeurs	US 9-10 : Création trajet, Historique	~10h
Phase 5 - Modération	US 11 : Espace Employé	~8h
Phase 6 - Administration	US 12 : Espace Admin, Stats	~10h
Phase 7 - Finalisation	Tests, Documentation, Déploiement	~7h

5. Difficultés rencontrées et solutions

5.1 Gestion des rôles utilisateurs

Problème : Symfony utilise un système de rôles (ROLE_USER, ROLE_ADMIN) tandis que le métier demande des rôles fonctionnels (Passager, Chauffeur).

Solution : J'ai créé deux champs distincts : 'role' (ENUM métier) et 'roles_system' (JSON pour Symfony). Cela permet de gérer l'autorisation technique tout en affichant des rôles compréhensibles pour l'utilisateur.

5.2 Système de crédits et transactions

Problème : Assurer la cohérence des crédits lors des participations (éviter les doublons, gérer les annulations).

Solution : J'ai implémenté une double vérification avec token CSRF, stockage des crédits utilisés dans la table Participation, et des vérifications côté serveur avant chaque transaction.

5.3 Validation des avis

Problème : Les avis doivent être validés par un employé avant d'être visibles.

Solution : J'ai ajouté un champ 'statut' (en_attente, valide, refuse) dans l'entité Avis, avec un espace employé dédié pour la modération.

6. Bilan et perspectives

6.1 Ce qui a bien fonctionné

- L'architecture MVC de Symfony a facilité l'organisation du code
- Les formulaires Symfony avec validation intégrée ont accéléré le développement
- Bootstrap 5 a permis d'obtenir rapidement une interface responsive
- Le découpage en User Stories a rendu le projet plus gérable

6.2 Points d'amélioration

- Ajouter des tests unitaires et fonctionnels (PHPUnit)
- Implémenter un système de notifications email plus complet
- Ajouter des graphiques interactifs (Chart.js) pour les statistiques
- Optimiser les requêtes SQL pour les grandes volumétries

6.3 Compétences développées

- Maîtrise du framework Symfony 7 et de l'ORM Doctrine
- Gestion de l'authentification et de l'autorisation
- Création d'interfaces responsive avec Bootstrap 5
- Bonnes pratiques Git et organisation de projet

— Fin de la documentation de gestion de projet —