DOCUMENTATION GESTION DE PROJET ECORIDE

Auteur : Emmanuelle CADILHAC
Client : EcoRide
Date de publication : 03/05/2025
Version V1.0

Table des matières

Auteur : Emmanuelle CADILHAC
Client : EcoRide
Date de publication : 03/05/2025
Version V1.0

I. Introduction

Ce document présente la gestion de projet du développement de la plateforme EcoRide, depuis sa planification initiale jusqu'à son déploiement en production.

Il décrit la méthodologie adoptée, les outils utilisés, le suivi des tâches, les processus de validation, les indicateurs clés et les leçons tirées pour garantir une réalisation fluide et maîtrisée.

II. Méthodologie de gestion

Nous avons opté pour une approche Agile structurant notre travail autour d'un tableau Kanban partagé dans Notion.

Chaque User Story, numérotée de US1 à US16, est représentée par une carte dont le statut reflète son avancement au sein de du workflow Git, allant de la branche de développement à la branche principale.

Cette méthode nous assure une grande visibilité et une adaptation rapide aux changements de priorité.

III. Outils et environnement

La gestion de projet s'effectue dans Notion, où nous maintenons le Kanban, le backlog, la priorisation et le suivi des sprints.

Pour les maquettes wireframes et mockup en desktop et mobile, nous utilisons Figma.

Le développement se fait sous Visual Studio Code en HTML, CSS, JavaScript, PHP, MySQL et MongoDB.

Côté sécurité, les mots de passe sont hachés à l'aide de la fonction password_hash() (algorithme bcrypt par défaut), et toutes les requêtes SQL passent par PDO avec requêtes préparées pour prévenir les injections. De plus, une veille régulière des recommandations OWASP est réalisée.

Le pipeline CI/CD est entièrement géré par GitHub Actions.

Après avoir exécuté l'analyse statique du code avec PHP_CodeSniffer (PSR-12) et PHPStan, puis lancé les tests unitaires PHPUnit avec génération de rapport de couverture, il construit une image Docker multi-stage et publie les artefacts (coverage.xml) ainsi que l'image résultante sur le GitHub Container Registry.

Enfin, grâce à la Heroku CLI intégrée au workflow, le front et l'API sont automatiquement déployés sur des dynos Heroku, qui prennent en charge l'hébergement, la mise à l'échelle et la gestion des certificats SSL en production.

IV. Planification et suivi

Chaque semaine, nous organisons une séance de backlog grooming afin d'ajuster les priorités et d'affiner les User Stories en story points.

Lors du sprint planning, qui a lieu tous les lundis matin, nous sélectionnons dans le backlog les User Stories à déplacer dans la colonne « À faire ».

V. Workflow Git et déploiement

Pour chaque User Story sélectionnée, une branche Git est créée selon le format feature/US#-titre-court.

Dès le premier commit, la carte Notion bascule en statut « In Progress ».

Une fois le développement terminé et tous les tests unitaires validés, nous ouvrons une Pull Request vers la branche develop.

Après revue et approbation, la PR est mergée puis la carte Notion passe en « Merged » pour indiquer son intégration à la branche principale main.

Le déploiement est entièrement automatisé : la branche **develop** est déployée sur un dyno Heroku dédié à l'environnement de **staging** via la Heroku CLI intégrée au pipeline, tandis que la branche **main** est publiée en **production** sur Heroku, garantissant un hébergement scalable, la gestion automatique des certificats SSL et la mise à l'échelle des dynos pour le front comme pour l'API.

VI. Qualité, sécurité et indicateurs

Notre suivi d'avancement s'appuie sur un burndown chart automatisé, obtenu grâce à l'intégration entre Notion et GitHub.

Nous définissons des jalons clairs pour chaque étape majeure, en particulier les releases alpha, beta et production.

À ce jour, notre suite PHPUnit comprend sept à huit tests couvrant les fonctionnalités les plus critiques.

Je prévois d'ajouter progressivement de nouveaux cas de test pour augmenter la couverture au fil des sprints, dans le but d'atteindre un taux encore plus élevé.

Par ailleurs, une revue de sécurité selon les bonnes pratiques OWASP est planifiée à chaque version majeure pour détecter et corriger les vulnérabilités avant tout déploiement.

VII. Gestion des risques et retours d'expérience

Nous avons identifié plusieurs risques techniques, notamment l'instabilité de certaines dépendances PHP, qui a été atténuée grâce à la gestion des versions via Composer et à l'ajout de tests automatisés.

La compatibilité multi-navigateurs a également été un point de vigilance ; nous réalisons des vérifications sur Firefox, Chrome et Safari pour garantir une expérience uniforme.

Enfin, la conformité RGPD et la protection des données utilisateurs ont dicté la mise en place de contrôles stricts sur la transmission et le stockage des informations personnelles.

VIII. Perspectives et évolutions

Pour la prochaine version, nous envisageons d'ajouter la fonctionnalité d'édition du profil utilisateur via un formulaire dédié.

Nous continuerons à enrichir notre suite de tests unitaires et à déployer un système de monitoring en production.

Sur le plan fonctionnel, la roadmap inclut la gestion avancée des préférences personnalisées et l'intégration de notifications intelligentes pour améliorer encore l'expérience des covoitureurs.