

# RÉSUMÉ DU PROJET RANDOMI GO

Le dossier de projet décrit Randomi GO comme une application web 3D adaptant un jeu de cartes existant. Le résumé initial évoque une collaboration de trois étudiants sur neuf semaines hors alternance. Le cahier des charges identifie la gestion des comptes, des amis, des salons de jeu et le suivi des parties. Les mécaniques clés couvrent les points de vie, la pioche, les lancers de dé, le vol et les bonus. L'architecture retenue combine une API REST, un service WebSocket et une base de données relationnelle. Le schéma Merise complet des tables se trouve en annexe, sans détailler toutes les relations ici. La partie Backend utilise Rust, Actix Web et Diesel pour l'API et PostgreSQL pour la persistance. PoloDB a été intégré pour le stockage NoSQL des salons, garantissant souplesse et performance. Le frontend Vanilla JavaScript propose des pages de navigation et de jeu, sans framework tiers. Des custom elements HTML ont été développés pour le profil, la liste d'amis et les salons. Les media queries assurent une mise en page fluide de 320 à 1920 pixels, avec alternatives tactiles. L'absence de hover sur mobile/tablette a conduit à une gestion spécifique en CSS. Des tests unitaires valident la logique Rust, tandis que les tests d'intégration vérifient l'API. Les échanges temps réel sont couverts par un service WebSocket isolé pour chaque salon. Les Server-Sent Events informent dynamiquement l'interface des mises à jour d'état. La conteneurisation s'appuie sur Docker et Docker Compose pour orchestrer base, API et frontend. Le workflow CI/CD sur GitHub Actions exécute lint, compilation, tests et déploiement. Le déploiement en rolling update s'effectue sur DigitalOcean via un script SSH automatisé. Les secrets et variables d'environnement sont centralisés dans un fichier .env à la racine. Les annexes contiennent les schémas de conception, la documentation API et les migrations Diesel.