# Journal de Bord de l'Équipe de Développement

## Introduction

Ce document sert de journal de bord pour suivre les choix technologiques et les bonnes pratiques adoptées. Il détaille les outils utilisés et leur rôle dans le processus de développement.

# I. Render : Gestion du rendu et du déploiement

## 1. Utilisation :

Render permet un déploiement rapide et automatique, réduisant les efforts de maintenance. Son intégration avec GitHub assure une mise à jour fluide du code, garantissant une meilleure réactivité et une continuité des mises en production.

## 2. Justification du choix :

Render a été choisi pour sa simplicité d'utilisation et son efficacité dans la gestion des déploiements. Contrairement à d'autres solutions nécessitant une configuration complexe, Render permet un déploiement sans effort manuel grâce à son intégration continue avec GitHub. Il offre également un monitoring en temps réel et une gestion simplifiée des environnements, garantissant une meilleure stabilité et évolutivité des applications.

## 3. Avantages :

- **Automatisation complète** : Chaque push sur le dépôt déclenche automatiquement un nouveau build et un déploiement.
- **Sécurisation avancée** : Gestion intégrée des certificats SSL et des domaines personnalisés.
- **Simplicité et évolutivité** : Support natif des bases de données et des services en arrière-plan sans configuration complexe.
- **Fiabilité** : Uptime élevé avec une infrastructure robuste.

# II. Postman : Gestion des requêtes API

## 1. Utilisation :

Postman facilite la création, le test et la documentation des requêtes API, assurant une meilleure collaboration entre les développeurs.

## 2. Justification du choix :

Postman a été retenu pour sa capacité à fournir un environnement de test robuste et interactif, permettant de simuler les requêtes API avant leur implémentation. Son système de collections facilite le partage et la réutilisation des requêtes entre les membres de l'équipe, garantissant ainsi une uniformité dans le développement des services.

## 3. Avantages :

- **Expérience utilisateur optimisée** : Interface ergonomique.
- **Collaboration efficace** : Partage des collections de requêtes.
- **Automatisation des tests** : Scripts intégrés pour éviter les erreurs en production.
- **Documentation centralisée** : Génération automatique.

# III. Docker : Gestion de la base de données

## 1. Utilisation :

Docker permet de gérer des bases de données en environnement isolé, évitant les conflits et garantissant la cohérence entre les environnements.

## 2. Justification du choix :

Docker a été choisi pour sa capacité à créer des environnements de développement homogènes, évitant les problèmes de compatibilité et simplifiant les déploiements multi-plateformes. Grâce à Docker Compose, il est possible d'orchestrer facilement plusieurs services, assurant ainsi une gestion efficace et évolutive des bases de données.

## 3. Avantages :

- **Isolation et uniformisation** : Conteneurs indépendants.
- **Facilité de mise en place** : Déploiement rapide.
- **Scalabilité et portabilité** : Déplacement facile des services.
- **Consistance et stabilité** : Réduction des écarts entre environnements.

# IV. SonarQube / SonarLint : Tests de vulnérabilité et formatage du code

## 1. Utilisation :

SonarQube et SonarLint analysent le code pour détecter les vulnérabilités et améliorer la qualité du développement.

## 2. Justification du choix :

Ces outils ont été sélectionnés pour assurer une qualité de code optimale en détectant automatiquement les bugs, vulnérabilités et mauvaises pratiques. SonarQube permet une analyse approfondie sur l'ensemble du projet, tandis que SonarLint agit en local pour prévenir les erreurs dès la phase de développement.

## 3. Avantages :

- **Détection proactive des vulnérabilités**.
- **Standardisation et qualité du code**.
- **Analyse continue intégrée aux pipelines CI/CD**.
- **Amélioration de la collaboration**.

# V. Maven, GitHub Workflow, Dockerfile : Gestion des dépendances et pipelines

### 1. Maven (pom.xml) :

- **Utilisation** : Gestion des dépendances et des plugins nécessaires au projet.
- **Justification du choix** : Maven permet une gestion centralisée et automatisée des bibliothèques et des builds, assurant ainsi une uniformité dans l'ensemble du projet. Il facilite également l'intégration avec les outils de CI/CD pour un déploiement sans friction.
- **Avantages** : Centralisation des dépendances, intégration facile avec CI/CD.

### 2. GitHub Workflow (pipelineci.cd.yml) :

- **Utilisation** : Orchestration des étapes du pipeline CI/CD.

- **Justification du choix** : GitHub Workflow a été adopté pour automatiser les processus de développement, allant des tests unitaires aux déploiements en production. Il garantit une intégration continue fluide et un suivi structuré de chaque étape du cycle de développement.
- **Avantages** : Sécurisation du pipeline, intégration fluide avec les services cloud.

### 3. Dockerfile (ressources/Dockerfile) :

- **Utilisation** : Création d'images conteneurisées pour un déploiement reproductible.
- **Justification du choix** : L'utilisation d'un Dockerfile permet de garantir que l’application s’exécute de manière cohérente sur n’importe quel environnement, réduisant ainsi les erreurs dues aux écarts de configuration.
- **Avantages** : Standardisation des environnements.