**Document Technique : Plateforme de Gestion de Stocks**

**1. Introduction**

**1.1 Objectif du projet**

L'objectif de ce projet est de développer une plateforme de gestion de stocks destinée aux entrepôts, magasins et autres structures nécessitant un suivi précis de leurs inventaires. La plateforme permettra de :

* Gérer les stocks (ajout, modification, suppression d'articles).
* Générer et gérer des factures.
* Fournir des outils de reporting et d'analyse.
* Être extensible pour ajouter des fonctionnalités futures (intégration de paiements, synchronisation avec des terminaux de caisse, etc.).

**1.2 Public cible**

* Gestionnaires de stocks.
* Responsables d'entrepôts.
* Propriétaires de magasins.
* Équipes logistiques.

**1.3 Technologies principales**

* **Backend** : Java avec Spring Boot.
* **Frontend** : React.
* **Base de données** : PostgreSQL.
* **DevOps** : Docker, GitHub Actions.

**2. Architecture du projet**

**2.1 Choix d'architecture**

Le projet sera basé sur une architecture **monolithique modulaire** avec Spring Boot, ce qui permet une gestion simplifiée au départ tout en restant scalable. Si nécessaire, une migration vers une architecture microservices pourra être envisagée ultérieurement.

**2.2 Diagramme d'architecture**

Voici un schéma simplifié de l'architecture :

Copy

+-------------------+ +-------------------+ +-------------------+

| Frontend | | Backend | | Base de données |

| (React) | <---> | (Spring Boot) | <---> | (PostgreSQL) |

+-------------------+ +-------------------+ +-------------------+

^ ^ ^

| | |

v v v

+-------------------+ +-------------------+ +-------------------+

| Utilisateurs | | Cache (Redis) | | File d'attente |

| (Navigateur) | | | | (RabbitMQ/Kafka) |

+-------------------+ +-------------------+ +-------------------+

**2.3 Composants principaux**

**Backend (Spring Boot)**

* **Spring Boot** : Framework principal pour le développement du backend.
* **Spring Data JPA** : Gestion des entités et accès aux données.
* **Spring Security** : Authentification et autorisation.
* **Spring Batch** : Traitements par lots (exemple : génération de rapports).
* **Swagger** : Documentation de l'API.

**Frontend (React)**

* **React** : Bibliothèque principale pour l'interface utilisateur.
* **Redux Toolkit** : Gestion de l'état global.
* **Axios** : Appels API vers le backend.
* **Material-UI** : Composants UI prêts à l'emploi.
* **React Router** : Gestion de la navigation.
* **Chart.js** : Visualisation des données (tableaux de bord).

**Base de données**

* **PostgreSQL** : Base de données relationnelle pour les stocks, factures, utilisateurs, etc.
* **Redis** : Cache pour optimiser les performances.
* **MongoDB** (optionnel) : Pour des données non structurées ou des logs.

**DevOps**

* **Docker** : Conteneurisation des applications.
* **Docker Compose** : Orchestration des services.
* **GitHub Actions** : Intégration et déploiement continus (CI/CD).
* **Prometheus + Grafana** : Monitoring des performances.

**3. Fonctionnalités principales**

**3.1 Gestion des stocks**

* Ajouter, modifier, supprimer des articles.
* Suivre les quantités en stock.
* Gérer les catégories d'articles.
* Alertes de stock faible.
* Historique des mouvements de stock (entrées/sorties).

**3.2 Gestion des factures**

* Créer, modifier, supprimer des factures.
* Générer des factures au format PDF.
* Gérer les clients et les fournisseurs.
* Suivre les paiements.

**3.3 Authentification et autorisation**

* Connexion/déconnexion des utilisateurs.
* Rôles et permissions (admin, gestionnaire, employé).
* Gestion des profils utilisateurs.

**3.4 Reporting et analytics**

* Tableaux de bord pour visualiser les stocks, les ventes, etc.
* Export des données en CSV ou PDF.
* Rapports personnalisables.

**3.5 Extensions futures**

* Intégration avec des systèmes de paiement (Stripe, PayPal).
* Synchronisation avec des terminaux de caisse.
* API publique pour permettre à d'autres systèmes d'interagir avec la plateforme.

**4. Technologies et outils**

**4.1 Backend (Spring Boot)**

* **Spring Boot** : Framework principal.
* **Spring Data JPA** : Pour l'accès aux données.
* **Spring Security** : Pour la sécurité.
* **Spring Batch** : Pour les traitements par lots.
* **Lombok** : Pour réduire le code boilerplate.
* **MapStruct** : Pour le mapping entre entités et DTOs.
* **Swagger** : Pour documenter l'API.

**4.2 Frontend (React)**

* **React** : Bibliothèque principale.
* **Redux Toolkit** : Pour la gestion de l'état global.
* **Axios** : Pour les appels API.
* **Material-UI** : Pour les composants UI.
* **React Router** : Pour la navigation.
* **Chart.js** : Pour les graphiques et tableaux de bord.

**4.3 Base de données**

* **PostgreSQL** : Base de données relationnelle.
* **Redis** : Pour le caching.
* **MongoDB** (optionnel) : Pour des données non structurées.

**4.4 DevOps**

* **Docker** : Pour la conteneurisation.
* **Docker Compose** : Pour orchestrer les services.
* **GitHub Actions** : Pour la CI/CD.
* **Prometheus + Grafana** : Pour le monitoring.

**5. Étapes de développement**

**5.1 Phase 1 : Initialisation du projet**

* Configurer le projet Spring Boot avec les dépendances nécessaires.
* Configurer le projet React avec Vite ou Create React App.
* Mettre en place Docker et Docker Compose pour l'environnement de développement.

**5.2 Phase 2 : Backend**

* Implémenter l'authentification et l'autorisation.
* Créer les entités et les repositories pour la gestion des stocks.
* Développer les endpoints API pour la gestion des stocks et des factures.
* Configurer Swagger pour documenter l'API.

**5.3 Phase 3 : Frontend**

* Créer les pages de connexion et d'inscription.
* Développer les interfaces pour la gestion des stocks et des factures.
* Intégrer les appels API avec Axios.
* Ajouter des tableaux de bord avec Chart.js.

**5.4 Phase 4 : Tests et déploiement**

* Écrire des tests unitaires et d'intégration (JUnit pour le backend, Jest pour le frontend).
* Configurer GitHub Actions pour la CI/CD.
* Déployer l'application sur un cloud (AWS, Azure, ou Heroku).

**6. Annexes**

**6.1 Glossaire**

* **API** : Interface de programmation d'applications.
* **CI/CD** : Intégration et déploiement continus.
* **JWT** : JSON Web Token, utilisé pour l'authentification.

**6.2 Références**

* [Documentation Spring Boot](https://spring.io/projects/spring-boot)
* [Documentation React](https://reactjs.org/docs/getting-started.html)
* [Documentation PostgreSQL](https://www.postgresql.org/docs/)

**6.3 Contact**

* **Équipe de développement** : RICARD Joris
* **Email** : ricardjoris@gmail.com
* **GitHub** : <https://github.com/jojoricard/WarehousePro>