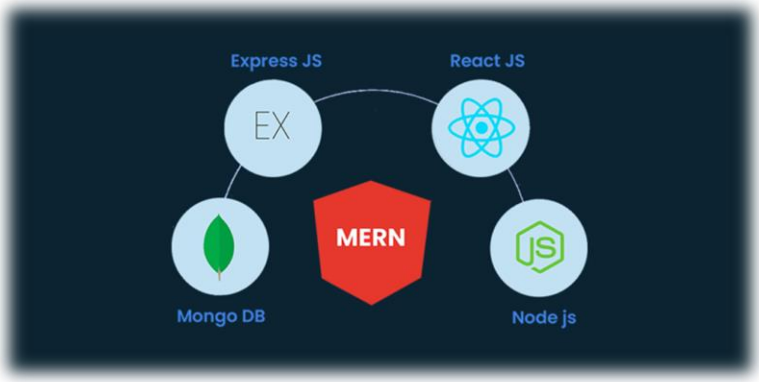
Documentation Technique

1. Présentation de l'Application

L'application est une solution de gestion de produits et de commandes pour une chaîne de distribution. Elle permet aux utilisateurs de naviguer dans un catalogue de produits mis à jour via l'API Open Food Facts, de passer des commandes et de générer des factures de manière automatisée. L'authentification sécurisée et la gestion des rôles permettent une séparation claire des responsabilités entre les vendeurs, acheteurs et administrateurs.

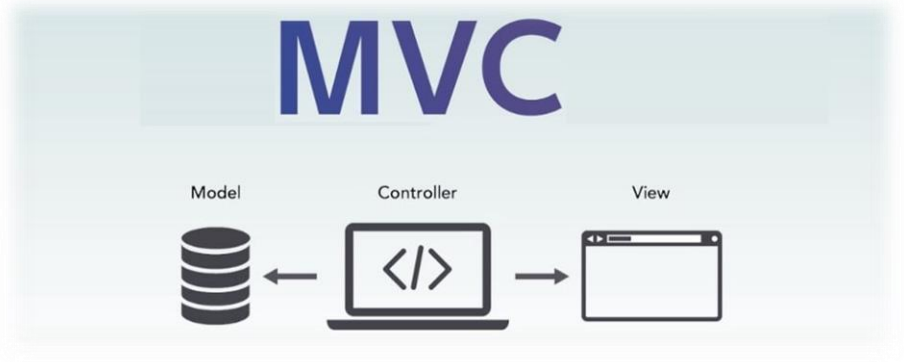
2. Architecture de l'Application

Technologies Utilisées (MERN STACK)



- **MongoDB** : Base de données NoSQL pour le stockage des utilisateurs, produits, commandes et factures.
- **Express.js** : Framework backend facilitant la gestion des API REST.
- **React.js** : Interface utilisateur dynamique et réactive.
- **Node.js** : Serveur backend permettant d'exécuter JavaScript côté serveur.

Modèle d'Architecture : MVC (Model-View-Controller)



- **Model** : Gestion des différentes collections MongoDB (User, Product, Order, Invoice).
- **View :** Interface utilisateur développée en React.js.
- **Controller :** Ensemble des fonctions qui gèrent la logique métier et les interactions avec la base de données.

Collections MongoDB

1. **User**
    a. Rôles : user / admin
    b. Statut : acheteur où vendeur
    c. Authentification sécurisée avec JWT, CSRF, et Google OAuth.
2. **Product**
    a. Script d'intégration avec Open Food Facts API pour mise à jour automatique des données produit (valeurs nutritionnelles, catégories, etc.).
    b. Mises à jour automatisées à raison d'un produit par seconde tout les soirs à 3h du matin.
3. **Order**
    a. Gestion des commandes et des statuts associés (en attente, validée, expédiée).
4. **Invoice**
    a. Génération de factures liées aux commandes et utilisateurs.
    b. Un utilisateur peut avoir plusieurs commandes, et chaque commande a une facture unique.

Diagramme d'Architecture

- Base de données avec 4 collections principales.
- Collection intermédiaire pour le panier (relation entre utilisateurs et produits).
- Intégration de l'API Open Food Facts pour enrichir les informations des produits.
- Gestion des factures liées aux commandes et aux utilisateurs.

3. Développement Front-End

- Framework : **React.js**
- Communication avec l'API REST via **Axios**.
- Interface utilisateur ergonomique avec **Material UI / Tailwind CSS**.
- Sécurisation des sessions et gestion des tokens.
- Fonctionnalités principales :
    - Affichage des produits avec filtres et recherche avancée.
    - Ajout au panier et gestion des commandes.
    - Interface administrateur pour la gestion des produits et utilisateurs.

4. Développement Back-End

- Développement de l'API REST avec **Node.js & Express**.
- Base de données **MongoDB**, accessible via **Mongoose**.
- Sécurisation avec **JWT, bcrypt** pour le hachage des mots de passe.
- Tests unitaires et d'intégration avec **Jest**.
- Gestion des transactions et paiements intégrée avec **PayPal API** (préparation pour futur développement).

Documentation API

- **Swagger** utilisé pour générer la documentation interactive de l'API.
- Endpoints REST :
    - /users : Gestion des utilisateurs (CRUD).

- /products : Gestion des produits et mise à jour depuis Open Food Facts.
- /orders : Gestion des commandes.
- /invoices : Génération et récupération des factures.
- /auth : Authentification et gestion des tokens JWT.

5. Intégration Open Food Facts API

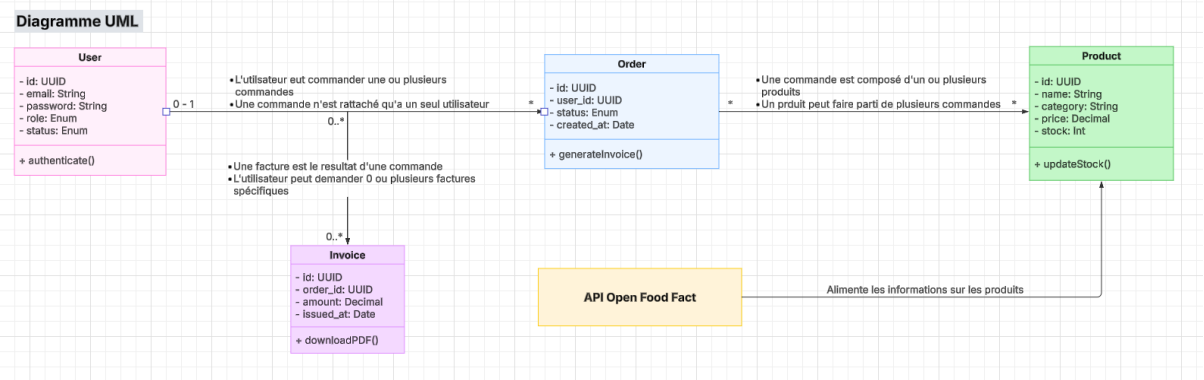
- Récupération et mise à jour des informations produits.
- Traitement asynchrone des requêtes pour éviter les surcharges.
- Sécurisation des appels API avec des clés d'accès.

6. Tests Fonctionnels

- **Jest & Supertest** pour les tests d'API.
- Vérifications automatisées des fonctionnalités :
  - Authentification et gestion des utilisateurs.
  - Création et mise à jour des produits.
  - Passage de commandes et génération des factures.

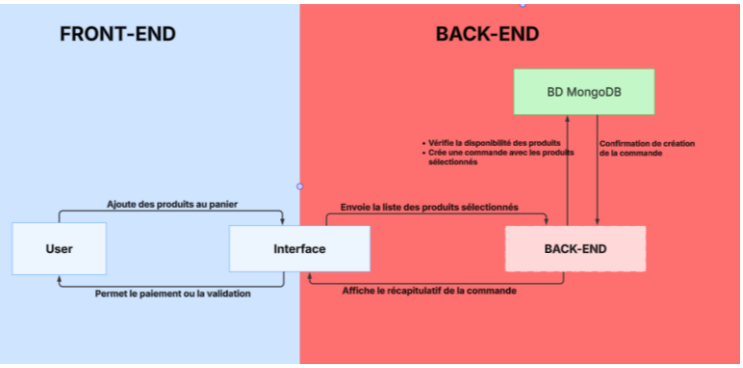
7. Diagrammes

Diagramme UML

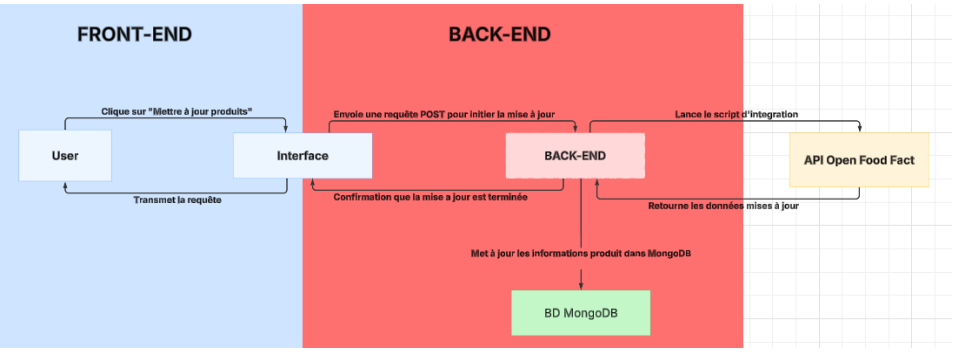


Diagrammes de Séquence

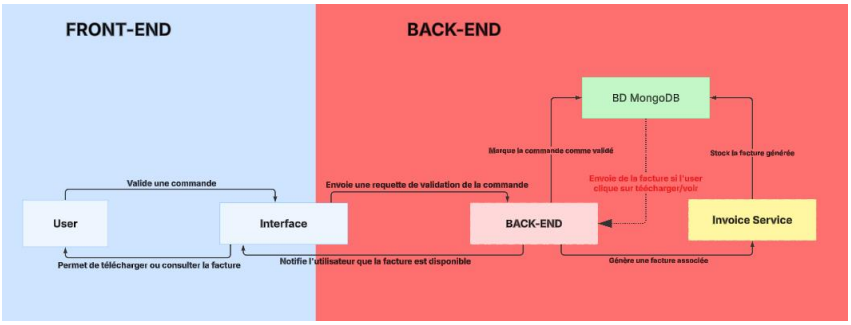
1. L'utilisateur passe une comme



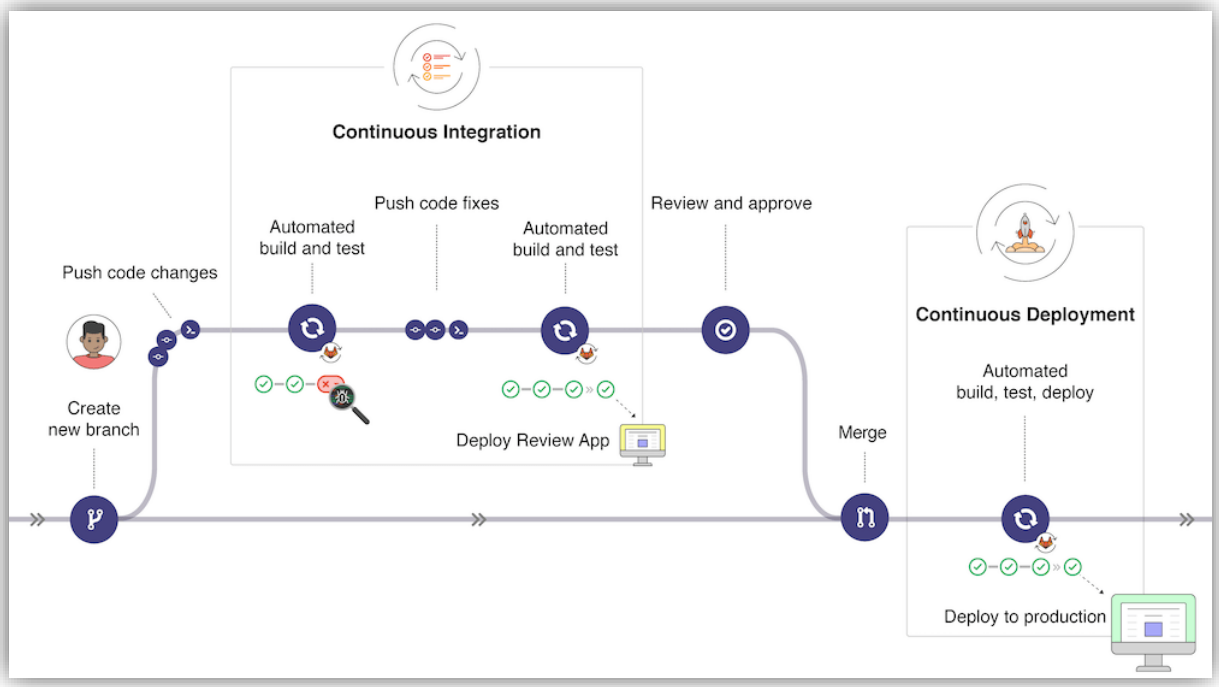
2. Mise à jour des produits



3. L'utilisateur génère une facture



8. CI/CD & Déploiement



- **Pipeline CI/CD** avec **GitLab CI/CD** :
  - Dockerisation pour les environnements **dev** et **prod**.
  - Déploiement automatisé sur **Heroku/AWS**.
  - Tests unitaires intégrés dans le pipeline avant toute mise en production.

9. Conclusion

Cette documentation couvre les aspects techniques essentiels du projet, de son architecture aux tests et au déploiement. L'utilisation de la stack MERN combinée à des bonnes pratiques de développement assure une application évolutive et sécurisée