# Shop Connect – étapes de développement

## Étape 1 : Analyse et planification

### Cahier des charges :

Objectif : Clarifier et formaliser les exigences du projet en un document précis.

Détails : Décrire les fonctionnalités principales (inscription, gestion de produits, commandes, etc.).

Définir les rôles (commerçant, client, administrateur).

Inclure les choix de technologie (Spring Boot, Angular, PostgreSQL, PayPal, OAuth).

Prendre en compte les contraintes de sécurité et la gestion des langues.

### Document d'analyse :

Objectif : Décrire les aspects techniques et fonctionnels.

Détails : Modéliser les cas d’utilisation avec des diagrammes UML (use cases, séquence).

Définir le modèle de données (tables pour les produits, utilisateurs, commandes, promotions, etc.).

Décrire les endpoints de l’API (utilise une approche API First avec Swagger/OpenAPI).

### Choix techniques :

Décrire pourquoi tu choisis Java, Spring Boot pour le backend, Angular avec TypeScript pour le frontend.

Justifier l'utilisation de PostgreSQL pour la base de données.

Détailler les raisons de l’intégration de Keycloak pour la sécurité (OAuth ou JWT).

Choisir la passerelle de paiement (API Stripe pour cartes de crédit, PayPal).

## Étape 2 : Initialisation des projets et configuration

### Backend - Spring Boot :

Configurer Spring Boot avec Maven/Gradle.

#### Installer les dépendances :

Spring Web (pour les API REST),

Spring Data JPA (pour la gestion des données),

Spring Security (pour OAuth2/JWT),

Spring Boot DevTools (pour le développement rapide),

PostgreSQL Connector (pour la base de données).

Configurer la base de données dans le fichier application.properties (ou application.yml).

### Frontend - Angular :

Initialiser un projet Angular avec la CLI Angular.

Installer PrimeNG pour la bibliothèque CSS, ainsi que Bootstrap ou Tailwind pour d’autres styles.

Configurer le routing pour les différentes pages : page d’accueil, page de boutique, page de gestion des produits, page de paiement, etc.

#### Gestion du versionnement :

Créer un dépôt Git (GitHub ou GitLab).

Mettre en place un workflow Git avec le client (Mr Martiel) avec des branches pour les fonctionnalités spécifiques.

## Étape 3 : Développement du backend avec Spring Boot

### Modélisation des entités JPA :

Créer les entités principales pour les tables de la base de données : User, Product, Order, Promotion.

Gérer les relations entre les entités (exemple : un utilisateur peut avoir plusieurs commandes, une commande peut contenir plusieurs produits).

### Création des repositories :

Utiliser Spring Data JPA pour générer automatiquement les opérations CRUD sur les entités.

Créer les services pour gérer la logique métier (par exemple, UserService, ProductService).

Créer les contrôleurs REST avec des endpoints clairs (par exemple, /api/products pour gérer les produits).

### Sécurisation de l’application :

Configurer OAuth2 avec Spring Security pour permettre l’authentification via Google ou Facebook.

Gérer l’authentification des API avec OAuth ou JWT (en utilisant Keycloak par exemple).

### Swagger/OpenAPI :

Documenter l’API avec Swagger en ajoutant la dépendance Springdoc pour générer automatiquement la documentation des endpoints.

### Gestion des paiements :

Intégrer l’API PayPal ou Stripe pour permettre les paiements.

Créer un service qui communique avec l’API de paiement pour traiter les commandes.

## Étape 4 : Développement du frontend avec Angular

### Création des services Angular :

Créer des services Angular pour communiquer avec les endpoints REST du backend (UserService, ProductService, OrderService).

### Création des composants Angular :

Créer des composants pour les pages principales : liste des produits, détails des produits, panier, page de paiement, page d’administration.

Utiliser PrimeNG pour des éléments visuels modernes et responsives.

### Routing :

Configurer le routage pour les différentes pages et utiliser des "guards" pour protéger les pages accessibles uniquement aux utilisateurs connectés (par exemple, la page d'administration).

### Multilingue :

Utiliser ngx-translate pour gérer les traductions et permettre aux utilisateurs de choisir entre le français et l’anglais.

Étape 5 : Test, intégration continue et déploiement

### Définir les tests :

Rédiger des tests unitaires pour les services backend avec JUnit et Mockito.

Rédiger des tests pour les composants Angular avec Jasmine/Karma.

Créer des tests d'intégration pour vérifier le bon fonctionnement de l’API REST.

### Intégration continue :

Mettre en place une intégration continue avec GitHub Actions ou GitLab CI pour automatiser les tests à chaque commit.

S’assurer que la base de données PostgreSQL est correctement configurée sur le serveur distant.

Gérer la configuration de production pour Spring Boot et Angular.

## Étape 6 : Présentation et soutenance

### Préparation de la démo :

Préparer une démonstration fluide montrant toutes les fonctionnalités principales : inscription, navigation des produits, commande, et paiement.

### Rapport :

Rédiger le rapport final en détaillant chaque partie du projet : l’analyse, les choix techniques, l’implémentation, les tests, et les outils utilisés.