# Cahier des Charges : Projet ShopConnect

## Contexte et objectifs du projet

### 1.1 Contexte

Le projet **ShopConnect** vise à créer une plateforme complète permettant aux commerçants locaux de digitaliser leurs activités.

Les commerçants pourront utiliser cette plateforme pour :

* Créer et personnaliser leur boutique en ligne.
* Gérer les produits et le stock en temps réel.
* Traiter les commandes des clients et suivre les paiements.
* Promouvoir leurs produits grâce à un espace marketing dédié aux promotions.

Les clients de leur côté, auront accès à une interface moderne où ils pourront :

* Parcourir les différentes boutiques en ligne.
* Ajouter des produits à leur panier et passer commande de manière sécurisée.
* Payer via des solutions de paiement modernes telles que Stripe et PayPal.

**ShopConnect** repose sur une architecture moderne et robuste qui garantit à la fois performance, sécurité, et évolutivité, grâce à l’utilisation de technologies bien établies :

#### Backend :

Le backend de l'application sera développé en Java avec Spring Boot. Cette stack technologique permet de gérer les API REST, la sécurité des données, et l'accès à la base de données PostgreSQL, réputée pour sa fiabilité et son évolutivité.

#### Frontend :

Le frontend utilisera Angular avec PrimeNG pour une interface utilisateur réactive et responsive. L'intégration de la bibliothèque ngx-translate assurera la gestion multilingue.

Sécurité : L'application implémentera une authentification basée sur OAuth2 et JWT via Keycloak pour sécuriser les sessions utilisateurs et protéger l'accès aux ressources sensibles.

Paiements : Les transactions sécurisées seront prises en charge grâce à l'intégration des API Stripe et PayPal, garantissant aux clients une expérience de paiement fluide et sécurisée.

### 1.2 Objectifs généraux

* Digitalisation des boutiques locales : Proposer une solution simple pour aider les commerçants à créer et gérer leurs boutiques en ligne.
* Gestion complète des produits et des commandes : Offrir aux commerçants des outils pour gérer facilement leurs stocks, produits et commandes via un tableau de bord intuitif.
* Expérience d’achat optimisée : Permettre aux clients de naviguer de façon fluide sur le site, de gérer leur panier, et de finaliser leurs achats en toute sécurité.
* Personnalisation et accessibilité : L’interface sera multilingue (français/anglais) et totalement responsive, permettant une utilisation sur tous types d’appareils.
* Sécurité des données : La plateforme intégrera des solutions de sécurité robustes (OAuth2, JWT) pour protéger les transactions, les informations personnelles des utilisateurs, et les données des boutiques.

## Analyse des besoins

### 2.1 Acteurs principaux

Commerçants : Les commerçants sont les principaux utilisateurs de la plateforme côté gestion. Ils doivent pouvoir s'inscrire, gérer leurs produits et suivre leurs commandes.

Clients : Les clients utilisent la plateforme pour naviguer entre les boutiques, ajouter des produits au panier, et finaliser des achats.

Administrateurs : Ils supervisent l'ensemble des activités de la plateforme. Ils doivent avoir accès à un espace sécurisé où ils peuvent gérer les utilisateurs, les boutiques, les produits, les promotions et les commandes.

Responsables marketing : Ils ont pour rôle d’encoder des promotions, de les attribuer à des produits spécifiques et de suivre leur impact.

### 2.2 Fonctionnalités détaillées

Inscription et authentification des utilisateurs

Inscription des utilisateurs : Tout utilisateur (client, commerçant) peut s'inscrire via une adresse email ou utiliser un fournisseur d'authentification externe (Google, Facebook).

Connexion sécurisée : Utilisation de OAuth2 ou JWT pour gérer l'authentification et la sécurité des sessions utilisateurs.

Réinitialisation des mots de passe : Système de réinitialisation sécurisé via email pour les utilisateurs.

#### Gestion des boutiques

Création et gestion de boutiques : Chaque commerçant doit pouvoir créer une ou plusieurs boutiques avec des informations telles que le nom, la description, l’adresse, et le numéro de téléphone.

Personnalisation de l'URL : Chaque boutique doit avoir un URL explicite généré automatiquement à partir du nom de la boutique, pour faciliter le référencement (SEO).

Gestion des produits : Les commerçants doivent pouvoir ajouter, modifier, et supprimer des produits via un tableau de bord simple. Les produits auront des attributs comme : nom, description, prix, stock, image, catégorie, etc.

#### Gestion des commandes

Panier d'achat : Les clients peuvent ajouter des produits à un panier, consulter et modifier le contenu de leur panier avant d’effectuer une commande.

Suivi des commandes : Les clients et les commerçants peuvent suivre l’évolution d’une commande : de la création, à l’expédition et la livraison.

Historique des commandes : Les clients et les commerçants peuvent consulter un historique détaillé de leurs transactions passées.

#### Paiements sécurisés

Moyens de paiement : Intégration des API Stripe pour les paiements par carte de crédit et PayPal pour les paiements en ligne.

Suivi des paiements : Le statut des paiements doit être consultable (payé, en attente, échoué). Un identifiant unique de transaction doit être généré pour chaque commande.

#### Espace administrateur

Accès administrateur : L’administrateur a accès à une interface sécurisée permettant de gérer le CRUD des utilisateurs, boutiques, produits, commandes, promotions.

Gestion des promotions : Les responsables marketing peuvent créer des promotions, définir des réductions (pourcentage ou montant fixe) et les attribuer à des produits ou des boutiques spécifiques.

#### Sécurité

Sécurisation des API : Toutes les API seront protégées via OAuth2 ou JWT pour éviter tout accès non autorisé.

Protection contre les attaques : Le site doit être conçu pour résister aux attaques courantes comme les injections SQL, XSS, CSRF.

Authentification à plusieurs niveaux : Gestion de rôles distincts (admin, commerçant, client) avec des accès différenciés.

#### Multilingue

Support multilingue : La plateforme doit être disponible en français et en anglais. Le client pourra choisir la langue de son interface via un sélecteur de langue visible sur chaque page.

Traduction des contenus : Tous les textes des pages doivent être traduisibles via des fichiers de localisation.

#### Responsivité et design

Interface responsive : Le design doit être totalement responsive, capable de s'adapter à tous les types d'appareils (PC, tablettes, smartphones).

Modernité du design : Utilisation de la bibliothèque PrimeNG avec Angular pour offrir des composants visuels modernes et interactifs. Des outils comme Bootstrap ou Tailwind CSS seront utilisés pour assurer un design épuré et professionnel.

### 2.3 Contraintes techniques

#### Backend

Langage : Le backend sera développé en Java 17 ou version supérieure.

Framework : Utilisation de Spring Boot pour le développement rapide d’API REST et la gestion des dépendances via Maven ou Gradle.

Base de données : PostgreSQL sera utilisé pour la gestion des données avec Spring Data JPA pour la gestion des entités (utilisateurs, boutiques, produits, commandes, etc.).

#### Frontend

Framework : Utilisation d’Angular 16 ou version supérieure pour la gestion du frontend, avec TypeScript.

Bibliothèque CSS : Intégration de PrimeNG pour la gestion des composants UI et ngx-translate pour la gestion multilingue.

#### Paiements

Stripe API : Pour les paiements par carte de crédit.

PayPal API : Pour les paiements via PayPal.

#### Sécurité

Keycloak : Pour la gestion des utilisateurs et l’authentification via OAuth2 et JWT.

Spring Security : Utilisé pour sécuriser les routes et protéger les API du backend.

### 2.4 Exigences non fonctionnelles

Performance : Le système doit supporter un grand nombre de connexions simultanées, notamment lors de pics d’activité (comme des périodes de promotion).

Scalabilité : L’architecture doit être capable de grandir facilement en ajoutant des fonctionnalités ou en gérant un nombre croissant d’utilisateurs.

Accessibilité : L’interface doit être simple à utiliser même pour des utilisateurs non techniques. Le respect des normes WCAG doit être pris en compte.

Sécurité : L’utilisation de HTTPS est obligatoire pour sécuriser les communications. Les données sensibles (mots de passe, informations de paiement) doivent être stockées de manière sécurisée (hashing, cryptage).

## Technologies et outils utilisés

### 3.1 Backend (Spring Boot)

Java 17 : Utilisé pour le développement du backend.

Spring Boot : Framework principal pour le développement rapide d’applications backend avec :

Spring Web pour la gestion des API REST.

Spring Data JPA pour la gestion des bases de données via JPA.

Spring Security pour la gestion de la sécurité.

OAuth2 et JWT via Keycloak pour la gestion des sessions.

PostgreSQL : Base de données relationnelle pour stocker les informations des utilisateurs, des boutiques, des produits et des commandes.

### 3.2 Frontend (Angular)

Angular 16+ : Framework JavaScript pour la gestion du frontend, associé à TypeScript pour garantir une meilleure maintenabilité du code.

PrimeNG : Bibliothèque UI pour des composants visuels modernes et responsifs.

ngx-translate : Bibliothèque pour la gestion multilingue.

### 3.3 Sécurité

Keycloak : Pour la gestion des authentifications via OAuth2/JWT et la gestion des rôles utilisateurs.

Spring Security : Intégré pour sécuriser les endpoints du backend.

### 3.4 Paiements

Stripe API : Pour la gestion des paiements par carte de crédit.

PayPal API : Pour la gestion des paiements via PayPal.

## Échéancier du projet

### Phase 1 : Analyse et préparation (13/09/2024 - 22/09/2024)

#### Semaine 1 (13/09/2024 - 22/09/2024)

* Cadrage du projet et prise de contact :

Définitions des grandes lignes du projet.

Identification des besoins des utilisateurs (commerçants, clients, administrateurs).

Finalisation des objectifs fonctionnels (multilingue, paiements sécurisés, espace administrateur, etc.).

* Rédaction du cahier des charges :

Rédaction initiale du cahier des charges : description des fonctionnalités, contraintes techniques, technologies à utiliser.

Relecture et validation du cahier des charges par le professeur.

* Étude et modélisation du MCD :

Définition des tables de la base de données (utilisateurs, produits, commandes, etc.).

Explication des relations entre les tables et conception des cardinalités.

### Phase 2 : Conception technique (23/09/2024 - 29/09/2024)

#### Semaine 2 (23/09/2024 - 29/09/2024)

* Conception de l'architecture technique :

Élaboration de l’architecture backend avec Spring Boot et définition des services REST.

Conception de l'architecture frontend avec Angular et PrimeNG.

* Choix des outils et technologies :

Validation des outils (Spring Boot, Angular, PrimeNG, PostgreSQL, Keycloak).

Mise en place des environnements de développement (installation des outils, IDE, configurations locales).

* Configuration initiale de la base de données :

Création du schéma de la base de données PostgreSQL.

Mise en place des premières tables (utilisateurs, produits, commandes).

### Phase 3 : Développement du backend (30/09/2024 - 20/10/2024)

#### Semaine 3 (30/09/2024 - 06/10/2024)

* Initialisation du projet Spring Boot :

Création du projet Spring Boot avec Maven/Gradle.

Configuration des dépendances principales : Spring Web, Spring Data JPA, Spring Security, PostgreSQL Connector.

* Mise en place du système d'authentification :

Intégration de Keycloak pour gérer l’authentification OAuth2/JWT.

Développement des endpoints pour l’inscription, la connexion, et la gestion des rôles utilisateurs (commerçant, client, administrateur).

#### Semaine 4 (07/10/2024 - 13/10/2024)

* Développement des services backend :

Création des services REST pour la gestion des utilisateurs (CRUD), des boutiques et des produits.

Développement des fonctionnalités CRUD pour les produits (ajout, modification, suppression, récupération).

* Gestion des relations avec JPA :

Mise en place des relations entre les entités : utilisateurs-boutiques, boutiques-produits, commandes-produits.

### Semaine 5 (14/10/2024 - 20/10/2024)

#### Développement des fonctionnalités commandes et paniers :

Création des services et endpoints pour gérer les paniers d’achat (ajout de produits au panier, modification, suppression).

Développement de l'API pour gérer les commandes (passage de commande, suivi du statut).

#### Intégration de la gestion des paiements :

Intégration de l'API Stripe pour gérer les paiements par carte de crédit.

Intégration de l'API PayPal pour les paiements alternatifs.

### Phase 4 : Développement du frontend (21/10/2024 - 10/11/2024)

#### Semaine 6 (21/10/2024 - 27/10/2024)

* Initialisation du projet Angular :

Création du projet Angular avec Angular CLI.

Installation des dépendances Angular nécessaires (PrimeNG, ngx-translate pour la gestion multilingue, Bootstrap ou Tailwind CSS pour le design).

* Mise en place du routage et de la structure du projet :

Création des différentes routes/pages du site : page d’accueil, page produit, page boutique, page panier, page commandes.

Mise en place du routing sécurisé (protections avec AuthGuard pour l’accès aux pages protégées).

#### Semaine 7 (28/10/2024 - 03/11/2024)

* Développement des composants frontend principaux :

Composants pour la gestion des boutiques et produits : affichage des produits, filtres, recherche.

Composants pour la gestion du panier et passage de commande.

Composants pour l’authentification (inscription, connexion via OAuth2 avec Google/Facebook).

* Intégration multilingue :

Configuration de ngx-translate pour permettre le basculement dynamique entre le français et l’anglais sur toutes les pages.

#### Semaine 8 (04/11/2024 - 10/11/2024)

* Finalisation des composants frontend :

Composants de gestion de compte pour les commerçants (ajout/modification de produits, suivi des commandes).

Composants pour les clients (consultation des commandes passées, suivi des livraisons).

* Responsive design :

Finalisation du design responsive pour assurer que toutes les pages s'affichent correctement sur PC, tablettes et mobiles.

### Phase 5 : Intégration et sécurisation (11/11/2024 - 24/11/2024)

#### Semaine 9 (11/11/2024 - 17/11/2024)

* Intégration frontend-backend :

Connexion des services Angular aux API REST du backend.

Test des communications entre le frontend et le backend (authentification, gestion des produits, commandes).

* Sécurisation des API :

Finalisation de l’intégration de Keycloak pour la sécurisation des API backend via OAuth2/JWT.

Test de la gestion des rôles (admin, commerçant, client) et vérification des droits d’accès.

Semaine 10 (18/11/2024 - 24/11/2024)

* Intégration des paiements dans le frontend :

Intégration des systèmes de paiement (Stripe et PayPal) dans le frontend pour la finalisation des commandes.

Test des scénarios de paiement complets (paiement réussi, échec de paiement).

* Tests de bout en bout :

Réalisation de tests de bout en bout pour s'assurer que le parcours utilisateur est fluide et fonctionne correctement (de l'inscription à la finalisation d'une commande).

### Phase 6 : Tests, validations et corrections (25/11/2024 - 08/12/2024)

#### Semaine 11 (25/11/2024 - 01/12/2024)

* Tests unitaires et d'intégration backend :

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

Tests d’intégration des services REST et validation de la sécurité (vérification des accès protégés).

* Tests unitaires frontend :

Rédaction des tests unitaires pour les composants Angular avec Jasmine/Karma.

Vérification de l'affichage des composants sur les différents supports (PC, tablettes, smartphones).

#### Semaine 12 (02/12/2024 - 08/12/2024)

* Résolution des bugs :

Correction des anomalies détectées durant les tests.

Optimisation des performances backend (requêtes SQL, gestion des transactions).

* Tests finaux et validation globale :

Réalisation de tests finaux avec des scénarios complets (création de compte, gestion des boutiques, commandes, paiements).

### Phase 7 : Finalisation et préparation de la soutenance (09/12/2024 - 22/12/2024)

#### Semaine 13 (09/12/2024 - 15/12/2024)

* Documentation :

Rédaction de la documentation technique et utilisateur.

Documentation des endpoints API avec Swagger/OpenAPI.

* Préparation de la présentation :

Création des slides pour la présentation finale.

Préparation de la démonstration en direct (parcours client, gestion commerçant, interface administrateur).

#### Semaine 14 (16/12/2024 - 22/12/2024)

* Préparation à la remise finale et à la présentation :

Révision des derniers détails avant la présentation en janvier 2025.