Mini-Projet : Développement d'une application micro-services sécurisée

Spring Boot – React – Keycloak

1. Contexte du projet

Ce mini-projet a pour objectif de concevoir et développer une **application web moderne basée sur une architecture micro-services sécurisée**.
L'application permettra la **gestion des produits et des commandes** d'une entreprise, tout en respectant les **standards industriels** en matière de **sécurité, modularité, conteneurisation et DevSecOps**.

2. Architecture générale attendue

L'architecture de l'application devra être composée des éléments suivants :

- Un **frontend web développé en React**
- Un **API Gateway** servant de point d'entrée unique
- Deux **micro-services Spring Boot indépendants** :
  - Micro-service **Produit**
  - Micro-service **Commande**
- Un **serveur d'authentification Keycloak**
- Une **base de données distincte pour chaque micro-service**

L'accès direct aux micro-services depuis le frontend est **strictement interdit**. Toutes les requêtes doivent obligatoirement transiter par l'API Gateway.

3. Frontend Web (React)

Le frontend React devra fournir une interface utilisateur **sécurisée** et **adaptée aux rôles**.

- Authentification via **Keycloak (OAuth2 / OpenID Connect)**
- Gestion de la session à l'aide de **tokens JWT**
- Affichage du catalogue des produits
- Création et consultation des commandes
- Adaptation de l'interface selon le rôle de l'utilisateur
- Communication exclusive avec l'API Gateway
- Gestion des erreurs d'accès (**401**, **403**)

4. Micro-service Produit (Spring Boot)

Le micro-service Produit sera responsable de la **gestion du catalogue des produits**.

- Ajouter un produit (**ADMIN**)
- Modifier un produit (**ADMIN**)
- Supprimer un produit (**ADMIN**)
- Lister les produits (**ADMIN, CLIENT**)
- Consulter un produit par identifiant (**ADMIN, CLIENT**)

Chaque produit devra contenir au minimum les attributs suivants :

- Identifiant
- Nom
- Description
- Prix
- quantité en stock

5. Micro-service Commande (Spring Boot)

Le micro-service Commande sera chargé de la **gestion des commandes clients**.

- Créer une commande (**CLIENT**)
- Consulter ses propres commandes (**CLIENT**)
- Lister toutes les commandes (**ADMIN**)
- Calculer automatiquement le montant total d'une commande
- Vérifier la disponibilité des produits avant validation

Une commande devra contenir au minimum :

- Identifiant
- date de commande
- statut
- montant total
- liste des produits commandés (idProduit, quantité, prix)

6. Communication inter-services

La communication entre les micro-services devra respecter les règles suivantes :

- Communication REST entre le micro-service Commande et le micro-service Produit
- Propagation du **token JWT** lors des appels inter-services
- Gestion propre des erreurs métiers (produit inexistant, stock insuffisant, etc.)

7. Sécurité avec Keycloak (obligatoire)

La sécurité constitue un **axe central du projet** et doit être implémentée à l'aide de Keycloak.

- Keycloak joue le rôle de **serveur d'authentification et d'autorisation**
- Authentification basée sur **OAuth2 / OpenID Connect**
- Sécurisation des APIs par **JWT**
- Gestion des rôles :
  – **ADMIN**
  – **CLIENT**

Les règles d'autorisation doivent être appliquées :

- Au niveau de l'API Gateway
- Au niveau de chaque micro-service

8. API Gateway (Spring Cloud Gateway)

L'API Gateway aura les responsabilités suivantes :

- Point d'entrée unique pour le frontend React
- Validation des tokens JWT
- Routage des requêtes vers les micro-services
- Centralisation des règles de sécurité
- Aucune logique métier n'est autorisée dans l'API Gateway

9. Gestion des données

La gestion des données devra respecter les principes micro-services.

- Une base de données distincte par micro-service
- Aucun partage de base entre Produit et Commande
- Choix libre du SGBD (PostgreSQL recommandé)
- Comptes d'accès distincts et sécurisés

10. Conteneurisation (Docker)

La conteneurisation est obligatoire pour l'ensemble de la plateforme.

• Un **Dockerfile** pour chaque composant :
  – frontend React
  – API Gateway
  – micro-service Produit
  – micro-service Commande

• Un **Docker Compose** permettant de lancer :
  – tous les services
  – Keycloak
  – les bases de données

11. DevSecOps (obligatoire)

Le projet devra intégrer une **démarche DevSecOps complète**.

• Analyse statique du code (ex. SonarQube)
• Analyse des dépendances (OWASP Dependency-Check)
• Scan des images Docker (Trivy)
• Correction des vulnérabilités détectées

12. Journalisation et traçabilité

Une attention particulière devra être portée à la journalisation et à la traçabilité.

• Logs d'accès aux APIs
• Logs d'erreurs applicatives
• Identification de l'utilisateur dans les logs
• Suivi basique de l'état des services

13. Livrables attendus

Les livrables attendus pour ce mini-projet sont :

- Code source complet versionné (Git)
- Diagramme d'architecture globale
- Diagramme de séquence du processus de commande
- Fichier Docker Compose fonctionnel
- Documentation technique (README)
- Captures des outils de sécurité utilisés
- Démonstration finale (optionnelle)

14. Extensions (bonus)

Pour les groupes avancés, les extensions suivantes peuvent être proposées :

- Déploiement Kubernetes
- Sécurisation inter-services par mTLS
- Circuit Breaker
- Tests automatisés
- Monitoring avancé