Projets J2EE : Architecture MicroService

1. Contexte du projet

Le projet consiste à concevoir et développer une application distribuée basée sur une architecture microservices, conforme aux standards modernes du cloud (Spring Cloud). L'objectif est de permettre de maîtriser la conception, le découpage fonctionnel, la communication inter-services et les bonnes pratiques DevOps.

L'application doit reposer sur :

- 2 microservices métier
- 1 microservice interface (Front API)
- Une architecture Spring Cloud complète **:**
  - Spring Cloud Config Server
  - Eureka Server (Service Discovery)
  - Spring Cloud Gateway (Routing & Security)
  - Actuator (Monitoring / Observability)
  - Résilience
  - OpenFeign (appel inter-services recommandé)

2. Architecture globale attendue

2.1. Microservices

Microservice 1 : Métier A

- Fournit un ensemble d'opérations métier (CRUD + logique métier).
- Contenir une base de données dédiée (MySQL/PostgreSQL…)**.**

Microservice 2 : Métier B

- Service indépendant du premier, avec sa propre base de données.
- Interaction possible avec Microservice A via OpenFeign**.**

Microservice Interface (Front-End Backend)

- Agrège les données des deux microservices métier.
- Fournit une API simplifiée destinée au front (Angular, React, Mobile…).
- Peut contenir une couche sécurité (Spring Security / JWT).

2.2. Services d'infrastructure (Spring Cloud)

A. Spring Cloud Config Server

- Centralisation des fichiers de configuration.
- Dépôt Git obligatoire.
- Configurations externalisées pour tous les microservices.

B. Eureka Server

- Enregistrement automatique des microservices.
- Load balancing (Ribbon/Feign).

C. Spring Cloud API Gateway

- Point d'entrée unique pour l'application.
- Routage dynamique basé sur Eureka.
- Possibilité d'ajouter des filtres :
    - Logging
    - Authentification (JWT)
    - Rate Limiting (optionnel)

D. Actuator

- Monitoring : health, env, metrics…
- Prometheus/Grafana optionnel pour les groupes avancés.

3. Suggestions de projets

Voici des suggestions ouvertes, mais Vous pouvez proposer vos propres projets.

3.1. *Plateforme de réservation (Hôtels / Salles / Terrains)*

- MS 1 : Gestion des ressources (chambres/salles)
- MS 2 : Gestion des réservations
- MS Interface : API de consultation + paiement simulé

3.2. *Application e-commerce simplifiée*

- MS 1 : Produits & catalogue
- MS 2 : Commandes & panier
- MS Interface : API client + auth JWT

3.3. *Système de gestion médicale*

- MS 1 : Patients / Dossiers
- MS 2 : Rendez-vous / Médecins
- MS Interface : Dashboard

3.4. Plateforme de livraison

- MS 1 : Restaurants / Menu
- MS 2 : Commandes / Livreurs
- MS Interface : Orchestration des commandes

3.5. Application éducative

- MS 1 : Étudiants / Profils
- MS 2 : Cours / Examens
- MS Interface : Portail d'accès

3.6. Gestion immobilière (recommandé vu ton métier)

- MS 1 : Biens immobiliers
- MS 2 : Clients & réservations
- MS Interface : API de recherche avancée

4. Bonus pour majoration

- Dockerisation complète des services
- Orchestration avec Docker Compose
- Observabilité via Prometheus + Grafana
- CI/CD GitHub Actions ou GitLab CI

Remarque :

1- L'ensemble du suivi du projet doit être versionné et régulièrement envoyé sur un dépôt Git, afin d'assurer un suivi dans le temps et de conserver l'historique des modifications

2- Quel que soit le projet choisi, il doit impérativement contenir une variable de configuration similaire à celle-ci :

La configuration du microservice-commandes inclut une propriété personnalisée `mes-config-ms.commandes-last`, utilisée pour afficher les dernières commandes reçues. Par exemple, la valeur `mes-config-ms.commandes-last = 10` permet d'afficher les commandes enregistrées au cours des 10 derniers jours.