ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR
Membre de
HONORIS UNITED UNIVERSITIES

Projet Microservices & Spring Cloud

• IMANE Chakrellah
• YASSINE Ech-chaoui

23/12/2025

PLAN

1. **Introduction**
2. **Objectif**
3. **Outils**
4. **Architecture**
5. **Démonstation**
6. **Conclusion**

Introduction

- Les applications modernes doivent être scalables, maintenables et faiblement couplées.
- L'architecture microservices permet de découper un monolithe en services indépendants.
- Dans ce projet, on a séparé le domaine en deux services principaux :

  microservice-produits
  microservice-commandes
- Le projet intègre aussi Config Server, Eureka, Gateway, et un Front React.

Objectifs

- Implémenter un **CRUD complet** sur Produits et Commandes sans SQL manuel **(Spring Data JPA).**
- Centraliser la configuration via **Spring Cloud Config + GitHub.**
- Superviser les services avec Spring Boot Actuator **(health + refresh).**
- Mettre en place la découverte des services avec **Eureka.**
- Centraliser l'accès via une **API Gateway** (routing + filters).
- Appliquer la résilience : simulation de timeout + **Hystrix fallback.**
- Documenter l'API avec **Swagger / OpenAPI.**
- Fournir une interface web avec **ReactJS** (gestion produits + commandes).

Outils & technologies utilisées

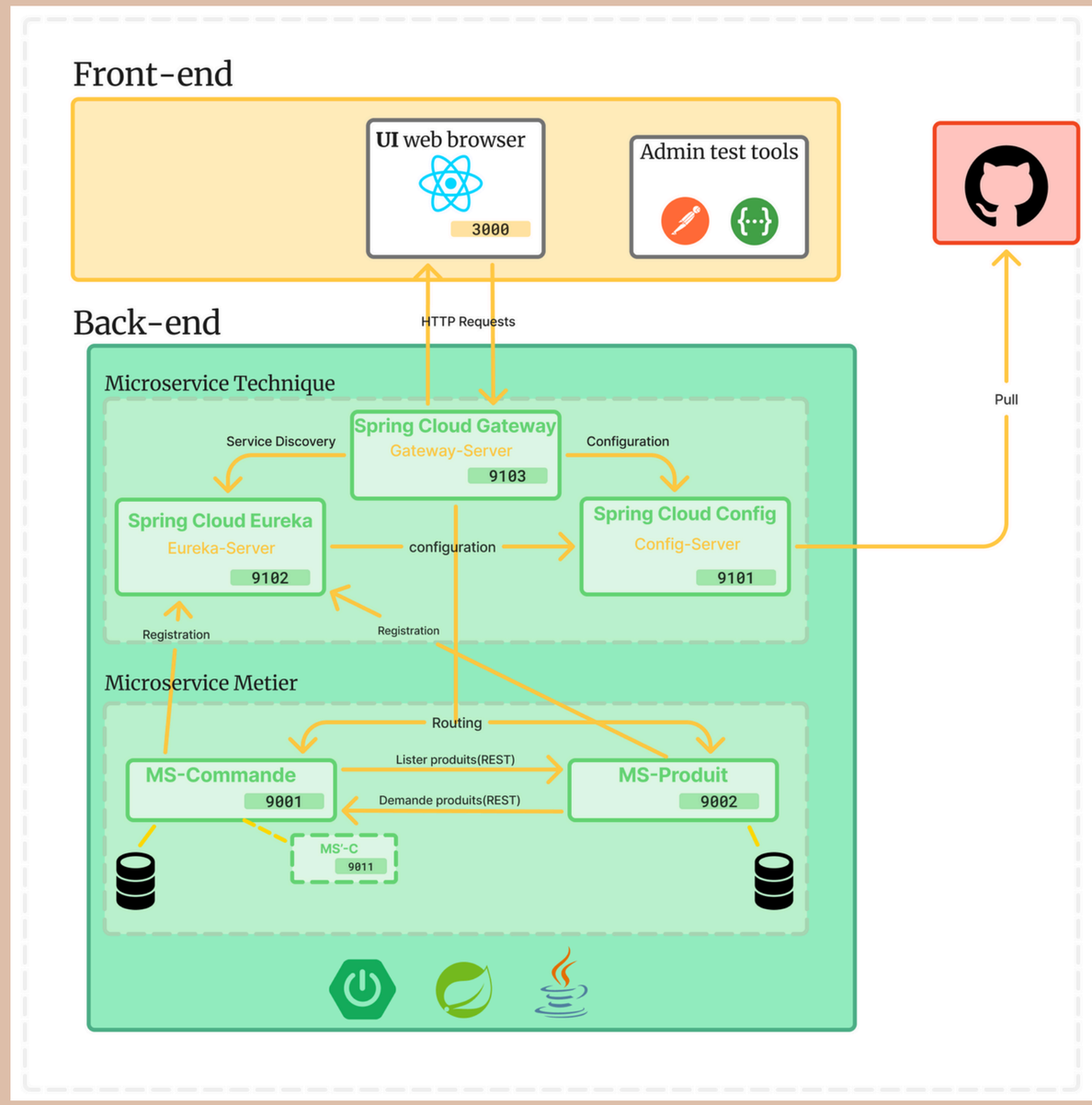
Back-end

- Java 17, Spring Boot
- Spring Data JPA (CRUD sans SQL)
- Spring Cloud Config (config centralisée)
- Eureka Discovery (service registry)
- Spring Cloud Gateway (point d’accès unique)
- Actuator (health, refresh, monitoring)
- Hystrix (fallback sur timeout)
- Swagger/OpenAPI (documentation)
- H2 Database

Font-end

- ReactJS + Axios (consommation API via Gateway)

Architecture



Démo

Conclusion

- Mise en place d’une architecture microservices complète avec Spring Cloud.
- Le projet démontre :

séparation des responsabilités (Produits / Commandes)

configuration centralisée et dynamique

supervision & health check

gateway + discovery + load balancing

résilience via Hystrix

Graziie!