Mini-Projet

Cahier des Charges

5ème Année Génie Logiciel

Année 2024-2025

Présentation

Ce document constitue l'**énoncé du mini-projet** de fin de module. Il décrit les spécifications techniques et fonctionnelles d'une plateforme d'agent IA conversationnel basée sur une architecture microservices moderne.

1 Contexte et Objectifs

Développer une plateforme d'agent IA conversationnel basée sur une architecture microservices. L'agent communique avec des services métier via le **Model Context Protocol (MCP)**, tandis que les services métier communiquent entre eux via **OpenFeign** avec résilience.

1.1 Objectifs Pédagogiques

— Concevoir une architecture microservices complète (Eureka, Gateway)
— Implémenter la communication MCP entre agent et services métier
— Maîtriser OpenFeign avec Circuit Breaker entre services
— Intégrer Spring AI pour créer un agent conversationnel
— Sécuriser avec JWT en mode stateless
— Développer une interface Angular moderne

2 Architecture Système

2.1 Services à Développer

Infrastructure (3 services) :

— Discovery Service (Eureka) — port 8761
— API Gateway — port 8888
— Auth Service — port 8080

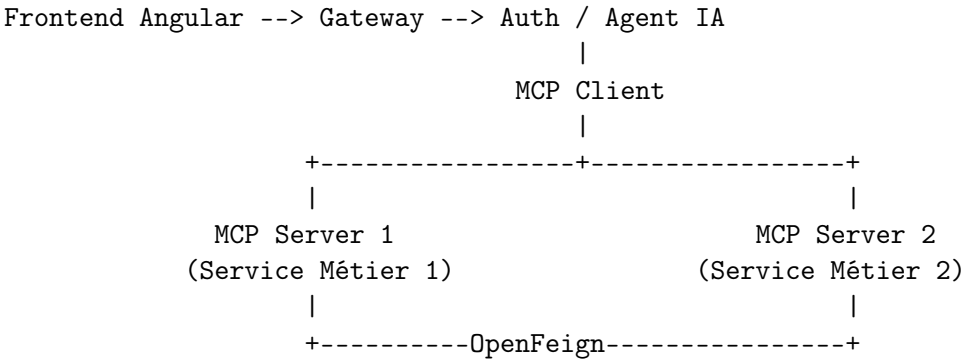
Service Intelligent (1 service) :

— Agent IA Service — port 8081 (MCP Client + Spring AI)

Services Métier (2 services minimum) :

— Service Métier 1 — port 9091 (MCP Server)
— Service Métier 2 — port 9092 (MCP Server)

2.2 Flux de Communication



Principes clés :

— Agent ↔ Services Métier : **MCP** (transport Streamable HTTP)
— Service Métier 1 ↔ Service Métier 2 : **OpenFeign + Circuit Breaker**

3 Technologies Imposées

— **Backend** : Spring Boot 3.x, Spring Cloud (Eureka, Gateway, OpenFeign)
— **IA** : Spring AI 1.1.1+ avec Ollama (llama3.2) ou OpenAI
— **MCP** : spring-ai-starter-mcp-client + spring-ai-starter-mcp-server-webmvc
— **Résilience** : Resilience4j (Circuit Breaker)
— **Sécurité** : Spring Security + JWT
— **BDD** : MySQL
— **Frontend** : Angular 17+

4 Fonctionnalités Requises

4.1 Authentification (Service Auth)

— **Pas d'inscription** — Comptes pré-créés au démarrage (USER + ADMIN)
— Endpoint de connexion : génération de token JWT après validation
— Protection des ressources par token JWT
— Mots de passe hachés avec BCrypt

4.2 Services Métier (MCP Servers)

Chaque service doit :

— Être un microservice Spring Boot complet (Controller, Service, Repository)
— Posséder sa propre base de données
— Exposer des outils via annotation `@McpTool`
— Configurer le transport MCP Streamable HTTP
— S'enregistrer auprès d'Eureka

Exemples de domaines :

— Service Produits + Service Stock
— Service Météo + Service Calculs
— Service Tâches + Service Rappels
— Ou tout autre domaine pertinent

Communication OpenFeign requise : Un service métier doit appeler l'autre via OpenFeign avec Circuit Breaker et méthode de fallback.

4.3 Agent IA (MCP Client)

L'agent doit :

— Se connecter aux services métier via MCP Client
— Découvrir automatiquement les outils disponibles
— Utiliser Spring AI avec mémoire conversationnelle (ChatMemory)
— Implémenter le function calling pour utiliser les outils
— Supporter le streaming des réponses
— Avoir un system prompt définissant son rôle

4.4 Frontend Angular

Interface utilisateur comprenant :

— Page de connexion (gestion du token JWT)
— Interface de chat conversationnel
— Affichage des messages user/agent
— Streaming des réponses
— Design responsive et moderne

5 Exigences Techniques

5.1 Architecture Microservices

— Tous les services doivent s'enregistrer auprès d'Eureka
— Gateway avec routage dynamique basé sur Eureka

5.2 Communication MCP

— Services métier configurés en MCP Servers (transport Streamable HTTP)
— Agent configuré en MCP Client avec connexions vers les services
— Outils MCP bien documentés (nom, description, paramètres)
— Endpoint MCP auto-configuré : `POST /mcp`

5.3 Communication OpenFeign

— Interfaces Feign pour les appels entre services métier
— Circuit Breaker sur chaque appel avec Resilience4j
— Méthodes de fallback pertinentes (messages clairs, pas de null)
— Découverte de services via Eureka (pas d'URL en dur)

5.4 Sécurité JWT

— Authentification stateless avec JWT
— Filtre JWT dans la chaîne de sécurité Spring
— Validation du token sur toutes les routes protégées
— Clé secrète en Base64 dans application.properties
— Token contient : username, rôles, expiration (24h)

5.5 Qualité du Code

— Architecture en couches (Controller, Service, Repository)
— Documentation Swagger/OpenAPI

6 Livrables

6.1 Code Source

— Projet Maven multi-modules (6 modules minimum)
— Projet Angular standalone

6.2 Démonstration Live

Présenter les scénarios suivants :

1. **Connexion** : Se connecter et accéder au chat
2. **Conversation** : Poser des questions simples (vérifier la mémoire)
3. **Utilisation d'outils** : Question nécessitant un outil MCP
4. **OpenFeign** : Montrer la communication entre services métier
5. **Résilience** : Arrêter un service, vérifier le fallback
6. **Dashboard Eureka** : Montrer tous les services enregistrés

Expliquer les choix techniques et répondre aux questions.

Bon courage !