Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université de Jijel
Faculté des Sciences exactes et de l'informatique
Département d'informatique

– Module –
Environnements et Programmation Dédiés

Master 1 : IA

Enseignant du module : Dr. Hemza FICEL

Contact: hemza.ficel@univ-jijel.dz

TP 5 - Recherche sémantique avec Milvus, Spring Boot et Serveur MCP

Objectifs

À la fin de ce TP, vous serez capable de :

- Comprendre pourquoi une base de données vectorielle est nécessaire pour les systèmes d'IA modernes
- Implémenter une recherche sémantique (au lieu d'une recherche SQL classique)
- Comprendre le rôle d'un serveur MCP (Model Context Protocol) pour connecter une IA à des services externes

Travail demandé

L’objectif de ce TP est de mettre en place un système de question–réponse intelligent basé sur :

- une base de données vectorielle (Milvus),
- un backend Spring Boot,
- des embeddings générés automatiquement (Google gemini),
- un serveur MCP permettant à un LLM d’interroger dynamiquement les données.

Architecture

Utilisateur

|

LLM (Claude Desktop ou autre)

|

Serveur MCP

|

Spring Boot API

|

Milvus (BD vectorielle)

Parie 1

1. Installer et lancer une base de données vectorielle (Milvus 2.6.X) à l'aide de Docker

2. Vérifier l'état de votre base de données (http://localhost:9091/webui/)

milvus

Home Collections Query Data Tasks Slow Requests Configurations Tools

Your Cluster is running well!

System Information

Attribute	Value	Description
Git Commit	3439c3e	Git commit SHA that the current build of the system is based on
Deploy Mode	STANDALONE	The mode in which the system is deployed
Build Version	2.6.7	The version of the system that was built
Build Time	Wed Dec 3 07:22:45 UTC 2025	The exact time when the system was built
Go Version	go version go1.24.9 linux/amd64	The version of the Golang that was used to build the system

Parie 2

1. Créer un nouveau projet Spring Boot 3.5.X
2. Vous pouvez utiliser Spring Initializr ou votre IDE.
3. Choisir Maven comme système de build.
4. Définir Java 17 ou supérieur.
5. Ajouter les dépendances suivantes dans le pom.xml :

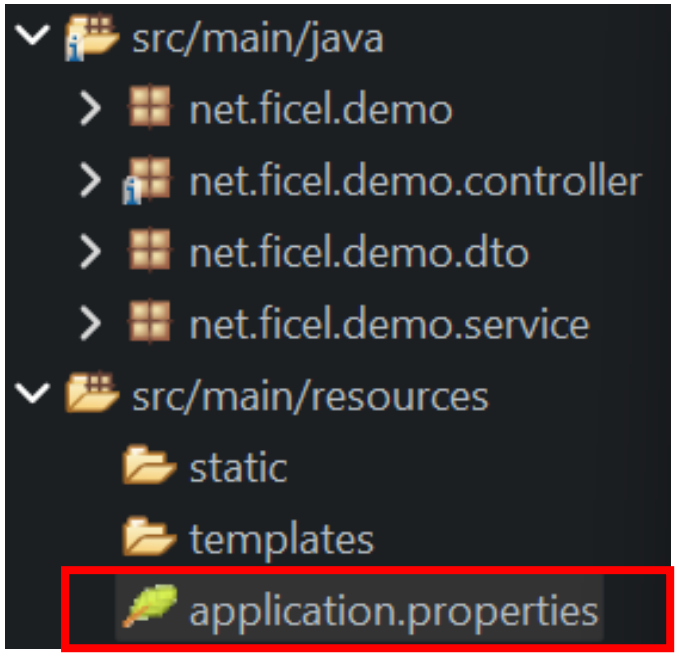
```xml
<dependencies>
    <!-- Pour créer des API REST -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <!-- Pour l'indexation et la recherche vectorielle -->
    <dependency>
        <groupId>org.springframework.ai</groupId>
        <artifactId>spring-ai-advisors-vector-store</artifactId>
    </dependency>

    <!-- Connecteur Milvus pour stocker des vecteurs -->
    <dependency>
        <groupId>org.springframework.ai</groupId>
        <artifactId>spring-ai-starter-vector-store-milvus</artifactId>
    </dependency>

    <!-- Génération d'embeddings via Google GenAI -->
    <dependency>
        <groupId>org.springframework.ai</groupId>
        <artifactId>spring-ai-starter-model-google-genai-embedding</artifactId>
    </dependency>

    <!-- Swagger / OpenAPI pour la documentation de l'API -->
    <dependency>
        <groupId>org.springdoc</groupId>
        <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
        <version>2.8.14</version>
    </dependency>
</dependencies>
```

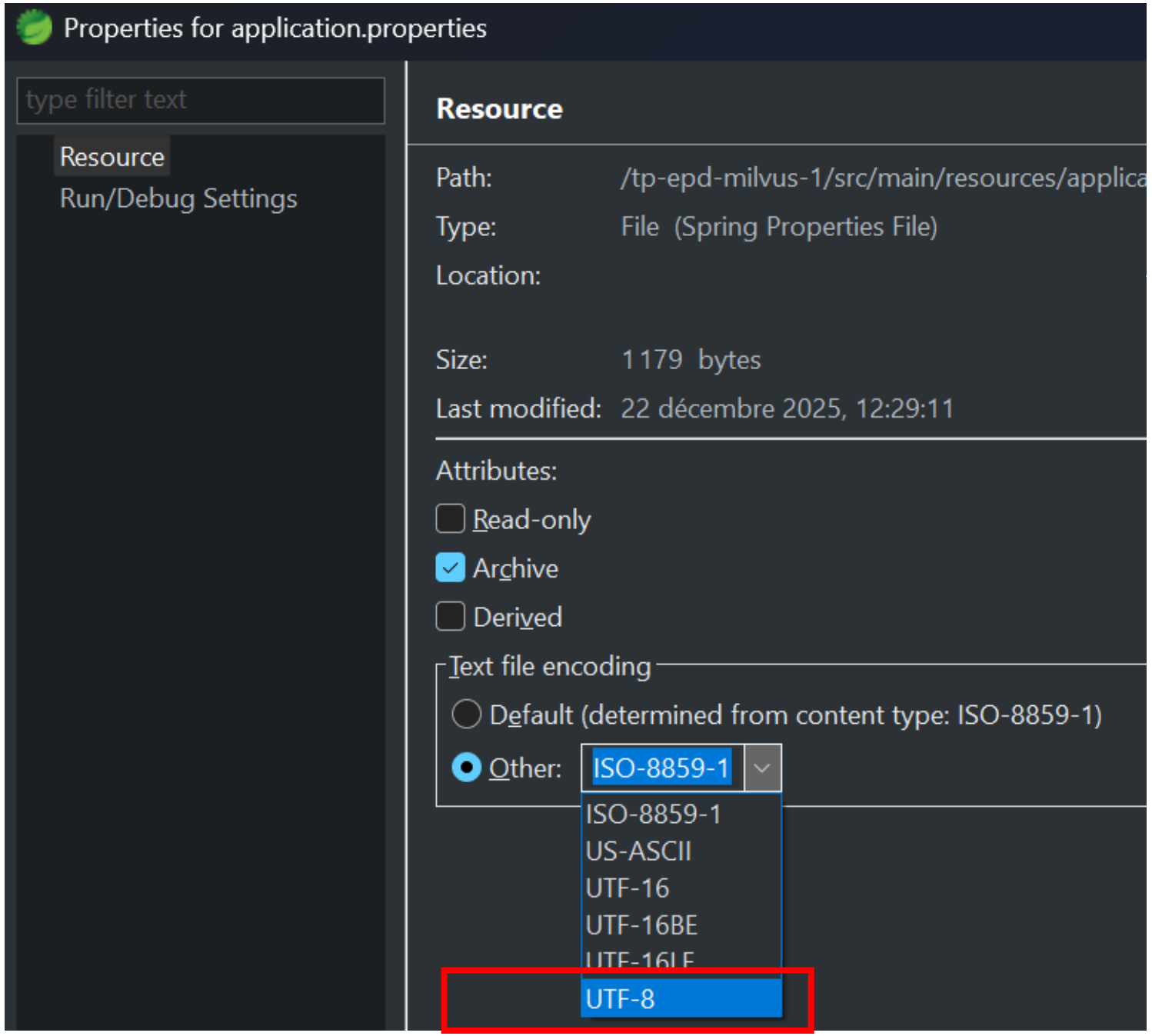
Parie 2

1. Configurer Spring AI pour utiliser le modèle d'embeddings Gemini (Google) comme moteur de représentation vectorielle.
2. Configurer votre projet Spring Boot afin que : les embeddings soient générés automatiquement, et qu'ils soient stockés dans Milvus (Spring AI se charge de transformer le texte en vecteurs, et de stocker ces vecteurs dans Milvus).

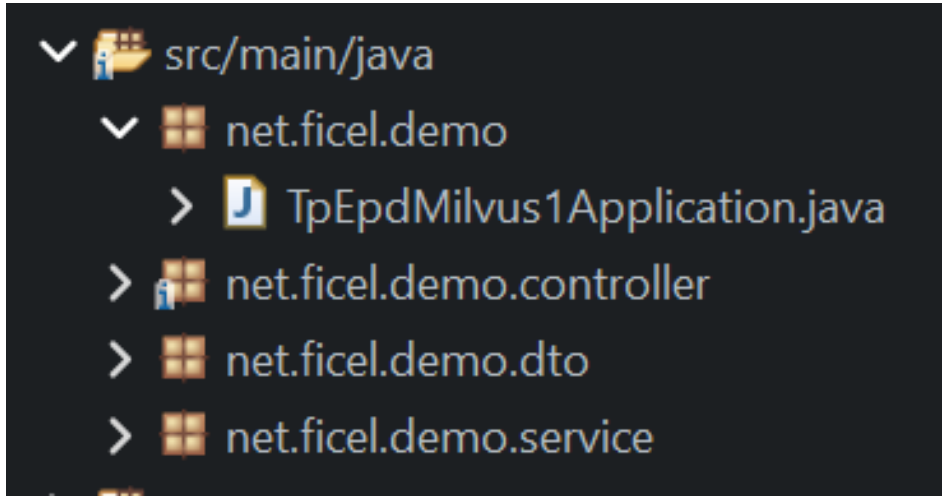
application.properties

```
spring.application.name=pdf-rag-search

# --- Google Gemini Configuration (API Google AI Studio) ---
# API Key
spring.ai.google.genai.embedding.api-key=your-API-key
# Le modèle text-embedding-004 est le plus performant pour le français
spring.ai.google.genai.embedding.text.options.model=text-embedding-004

# Configuration Milvus
spring.ai.vectorstore.milvus.client.host=localhost
spring.ai.vectorstore.milvus.client.port=19530

# Nom de la collection qui sera creer automatiquement
spring.ai.vectorstore.milvus.collection-name=my_documents
# Dimension pour embeddings
spring.ai.vectorstore.milvus.embedding-dimension=768
# Initialisation automatique du schéma
spring.ai.vectorstore.milvus.initialize-schema=true
```


src/main/java
net.ficel.demo
net.ficel.demo.controller
net.ficel.demo.dto
net.ficel.demo.service
src/main/resources
static
templates
application.properties


Properties for application.properties
type filter text
Resource
Run/Debug Settings
Resource
Path: /tp-epd-milvus-1/src/main/resources/applica
Type: File (Spring Properties File)
Location:
Size: 1 179 bytes
Last modified: 22 décembre 2025, 12:29:11
Attributes:
Read-only
Archive
Derived
Text file encoding
Default (determined from content type: ISO-8859-1)
Other: ISO-8859-1
ISO-8859-1
US-ASCII
UTF-16
UTF-16BE
UTF-16LE
UTF-8

Implémenter votre logique métier


src/main/java
net.ficel.demo
TpEpdMilvus1Application.java
net.ficel.demo.controller
net.ficel.demo.dto
net.ficel.demo.service

Package DTO

```java
package net.ficel.demo.dto;

public record QuestionRequest(String question) {}
```

```java
package net.ficel.demo.dto;

import java.util.Map;

public record SearchResultDto(
        String id,
        String text,
        Map<String, Object> metadata,
        Double score
) {}
```

```java
package net.ficel.demo.dto;

import org.springframework.ai.document.Document;

public final class DocumentMapper {

    private DocumentMapper() {}

    public static SearchResultDto toDto(Document doc) {
        return new SearchResultDto(
                doc.getId(),
                doc.getText(),
                doc.getMetadata(),
                doc.getScore()
        );
    }
}
```

Package Service

```java
package net.ficel.demo.service;

import org.springframework.ai.document.Document;
import org.springframework.ai.vectorstore.VectorStore;
import org.springframework.stereotype.Service;
import java.util.List;
import java.util.Map;

@Service
public class DocumentIndexerService {
	private final VectorStore vectorStore;

	public DocumentIndexerService(VectorStore vectorStore) {
		this.vectorStore = vectorStore;
	}

	public void ingestText(List<String> contents) {

		try {
			List<Document> documents = contents.stream()
					.filter(text -> text != null && !text.isBlank())
					.map(text -> new Document(text, Map.of("category", "info"))).toList();

			// Cette ligne fait tout : Embedding via Google + Stockage dans Milvus
			vectorStore.add(documents);
			System.out.println(documents.size() + " passages indexés avec succès.");
		} catch (Exception e) {
			throw new RuntimeException("Erreur lors de l'indexation des passages : " + e.getMessage(), e);
		}

	}

}
```

```java
package net.ficel.demo.service;

import java.util.List;
import org.springframework.ai.document.Document;
import org.springframework.ai.vectorstore.SearchRequest;
import org.springframework.ai.vectorstore.VectorStore;
import org.springframework.stereotype.Service;
import net.ficel.demo.dto.DocumentMapper;
import net.ficel.demo.dto.SearchResultDto;

@Service
public class SemanticSearchService {
	private final VectorStore vectorStore;

	public SemanticSearchService2(VectorStore vectorStore) {
		this.vectorStore = vectorStore;
	}

	public List<SearchResultDto> search(String query) {

		int queryLength = query.split("\\s+").length;
		double similarityThreshold = 0.1;
		 int topK = 4;
		SearchRequest searchRequest = SearchRequest.builder()
				.query(query)
				.topK(topK)
				.similarityThreshold(similarityThreshold).build();

		// 2. Effectuer la recherche vectorielle
		List<Document> results = vectorStore.similaritySearch(searchRequest);
		return results.stream().map(DocumentMapper::toDto).toList();
	}
}
```

Package Controller

```java
package net.ficel.demo.controller;

import java.util.List;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;

import net.ficel.demo.service.DocumentIndexerService;

@RestController
@RequestMapping("/api/v1/documents")
public class DocumentController {

	private final DocumentIndexerService indexerService;

	public DocumentController(DocumentIndexerService indexerService) {
		this.indexerService = indexerService;
	}

	@GetMapping("/")
	public String sayHello() {
		return "Bienvenue dans Document Controller";
	}

	@PostMapping("/texts")
	public ResponseEntity<String> ingestText(@RequestBody List<String> contents) {
		indexerService.ingestText(contents);
		return ResponseEntity.ok("Textes indexés avec succès");
	}
}
```

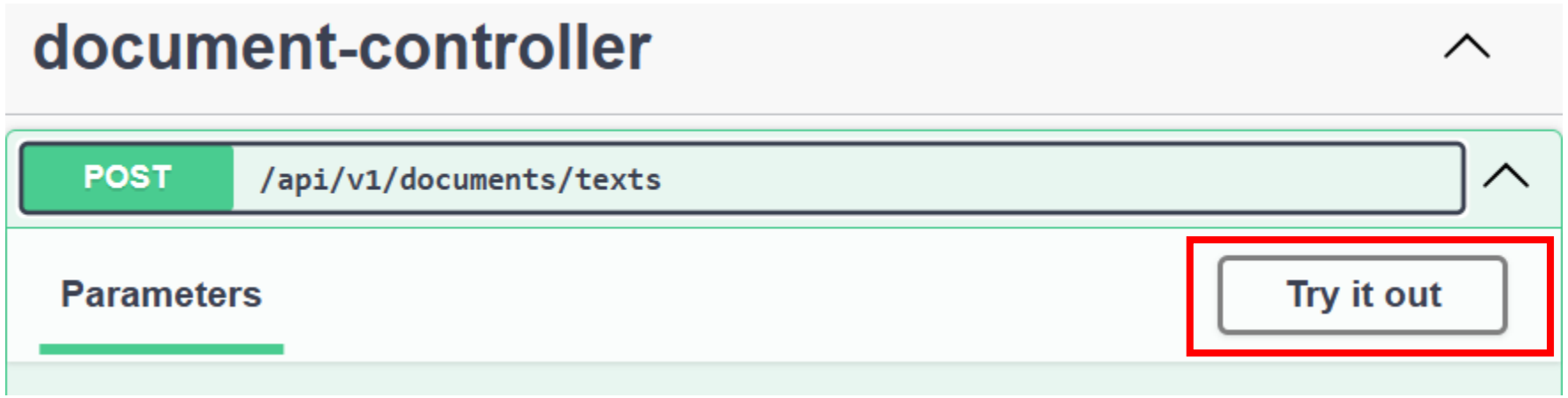
Créer L’endpoint REST « POST /api/v1/questions » dans le Controller « **`SearchController »`** pour effectuer une recherche vectorielle dans Milvus, récupérer les passages les plus proches et retourner les résultats au format JSON.

```java
package net.ficel.demo.controller;

import org.springframework.web.bind.annotation.*;

import net.ficel.demo.dto.QuestionRequest;
import net.ficel.demo.dto.SearchResultDto;
import net.ficel.demo.service.SemanticSearchService;

import java.util.List;

@RestController
@RequestMapping("/api/v1")
public class SearchController {

	private final SemanticSearchService semanticSearchService;

	public SearchController(SemanticSearchService semanticSearchService) {
		this.semanticSearchService = semanticSearchService;
	}

	@PostMapping("/questions")
	public List<SearchResultDto> askQuestion(@RequestBody QuestionRequest request) {

		return semanticSearchService.search(request.question());

	}
}
```

Parie 3

1. Tester la génération des embeddings Gemini et leur indexation dans Milvus via l'endpoint d'ingestion.
2. Utilisez l'outil « swagger » pour les testes. Vous pourrez accéder à la documentation interactive de ton API via l'URL par défaut :

http://localhost:8080/swagger-ui/index.html


document-controller
POST
/api/v1/documents/texts
Parameters
Try it out

Fichier JSON d’ingestion

À envoyer à ton endpoint : **POST /api/v1/documents/texts**

```
[
  "Le service doit être redémarré après toute modification du fichier de configuration.",
  "Les sauvegardes automatiques sont exécutées chaque nuit à 02:00 sur le serveur principal.",
  "Un redémarrage à chaud est possible uniquement lorsque le système est en mode maintenance.",
  "En cas d’échec de démarrage, vérifier les logs situés dans le répertoire /var/log/app.",
  "La modification des paramètres de sécurité nécessite des droits administrateur.",
  "Le mode maintenance empêche les utilisateurs externes d’accéder à l’application.",
  "Les fichiers de configuration sont chargés uniquement au démarrage de l’application.",
  "Une sauvegarde manuelle peut être déclenchée via l’interface d’administration.",
  "Après une mise à jour du système, un redémarrage complet est fortement recommandé.",
  "Les erreurs critiques sont enregistrées avec un niveau de log ERROR."
]
```

document-controller

POST /api/v1/documents/texts

Parameters

Cancel Reset

No parameters

Request body required

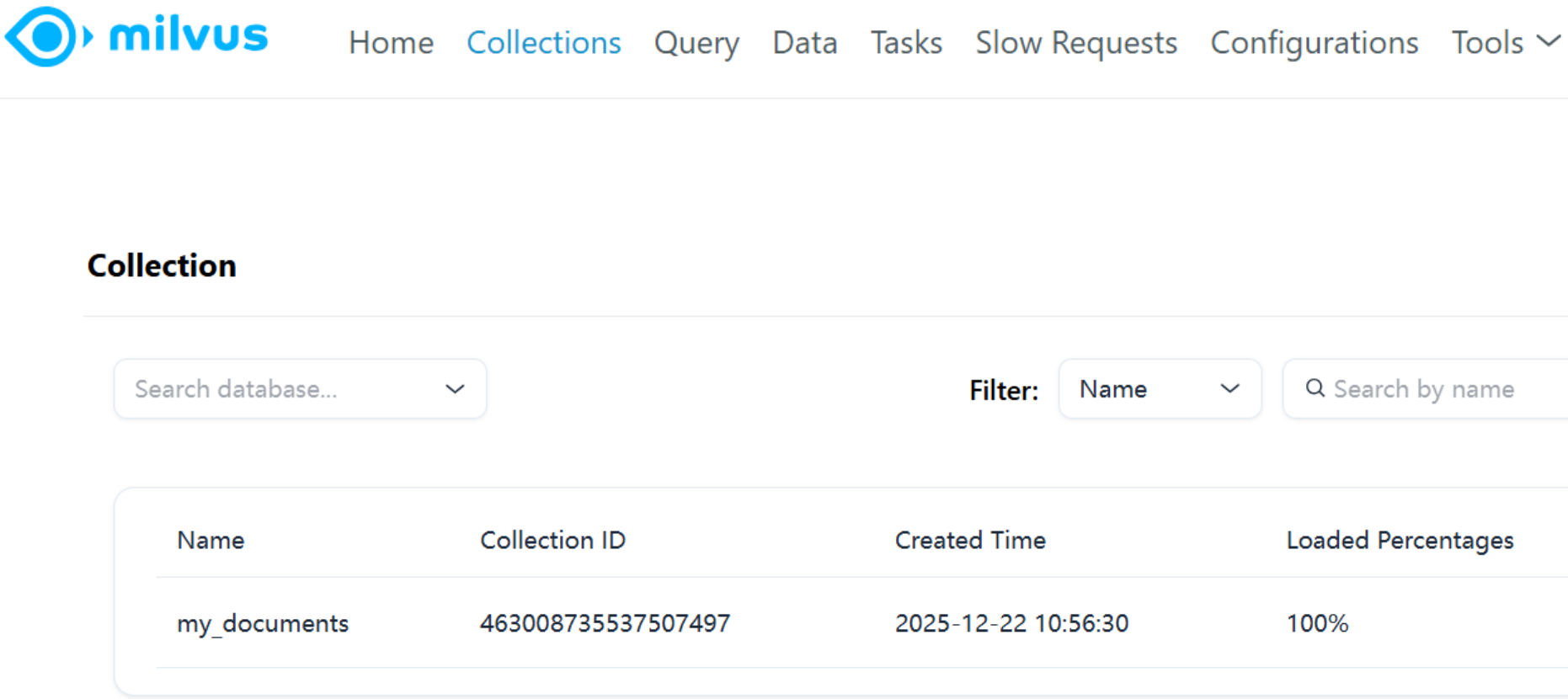
application/json

Edit Value | Schema


```
[
  "Le service doit être redémarré après toute modification du fichier de configuration.",
  "Les sauvegardes automatiques sont exécutées chaque nuit à 02:00 sur le serveur principal.",
  "Un redémarrage à chaud est possible uniquement lorsque le système est en mode maintenance.",
  "En cas d'échec de démarrage, vérifier les logs situés dans le répertoire /var/log/app.",
  "La modification des paramètres de sécurité nécessite des droits administrateur.",
  "Le mode maintenance empêche les utilisateurs externes d'accéder à l'application.",
  "Les fichiers de configuration sont chargés uniquement au démarrage de l'application.",
  "Une sauvegarde manuelle peut être déclenchée via l'interface d'administration.",
  "Après une mise à jour du système, un redémarrage complet est fortement recommandé.",
  "Les erreurs critiques sont enregistrées avec un niveau de log ERROR."
]
```

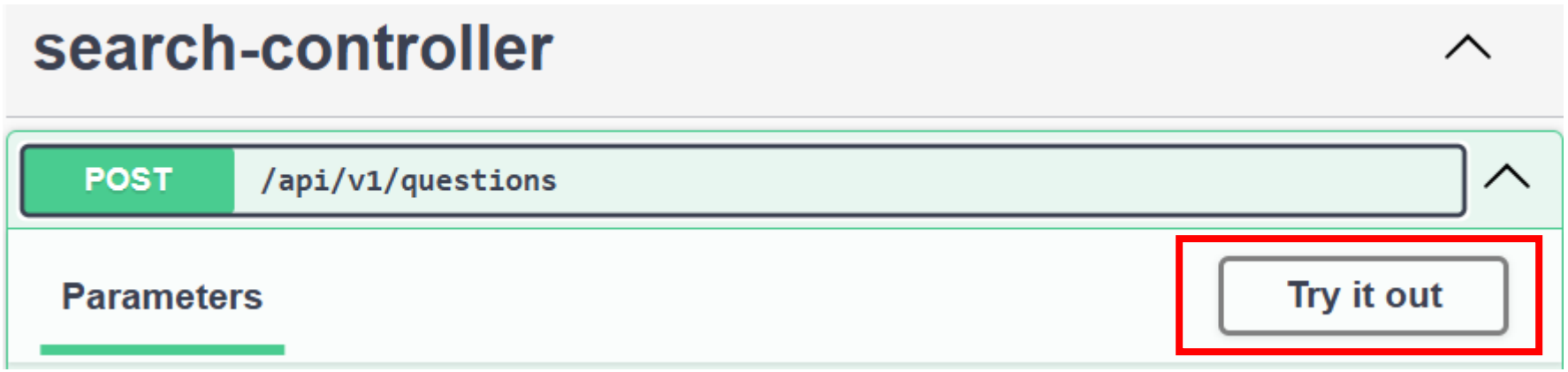

Execute

http://localhost:9091/webui/



🔍 Exemples de recherches sémantiques (à tester)

À envoyer à ton endpoint **: POST /api/v1/questions**


search-controller
POST
/api/v1/questions
Parameters
Try it out

search-controller

POST /api/v1/questions

Parameters | Cancel

No parameters

Request body required | application/json

Edit Value | Schema

```
{
  "question": "string"
}
```

Execute

Recherche 1 – reformulation naturelle

```
{
  "question": " Faut-il relancer l’application après un changement ? "
}
```

Résultats attendus (top)

☑ *"Le service doit être redémarré après toute modification du fichier de configuration."*

📌 **Aucun mot commun exact avec “relancer”**

➡ La recherche keyword échouerait

➡ La recherche sémantique fonctionne

Recherche 2 – intention utilisateur

```
{
  "question": " Comment éviter que les utilisateurs accèdent au système ?"
}
```

Résultats attendus (top)

☑ *"Le mode maintenance empêche les utilisateurs externes d'accéder à l'application."*

📌 Le mot *empêcher ≠ éviter*

📌 Le sens est reconnu

Recherche 3 – diagnostic / support

```
{
 "question": " Où trouver les informations en cas de problème au démarrage ?"
}
```

Résultats attendus (top)

☑ *"En cas d'échec de démarrage, vérifier les logs situés dans le répertoire /var/log/app."*

☑ *"Les erreurs critiques sont enregistrées avec un niveau de log ERROR."*

Recherche 4 – recommandation implicite

```
{
 "question": "Que faire après une mise à jour ?"
}
```

Résultats attendus (top)

☑ *"Après une mise à jour du système, un redémarrage complet est fortement recommandé."*

Recherche 5 – action utilisateur indirecte

```
{
 "question": "Est-ce que je peux lancer une sauvegarde moi-même ?"
}
```

Résultats attendus (top)

☑ *"Une sauvegarde manuelle peut être déclenchée via l'interface d'administration."*

📌 Aucun mot commun :

lancer ≠ déclencher

moi-même ≠ manuelle ➡ **C'est exactement là que la sémantique brille**

Partie 4

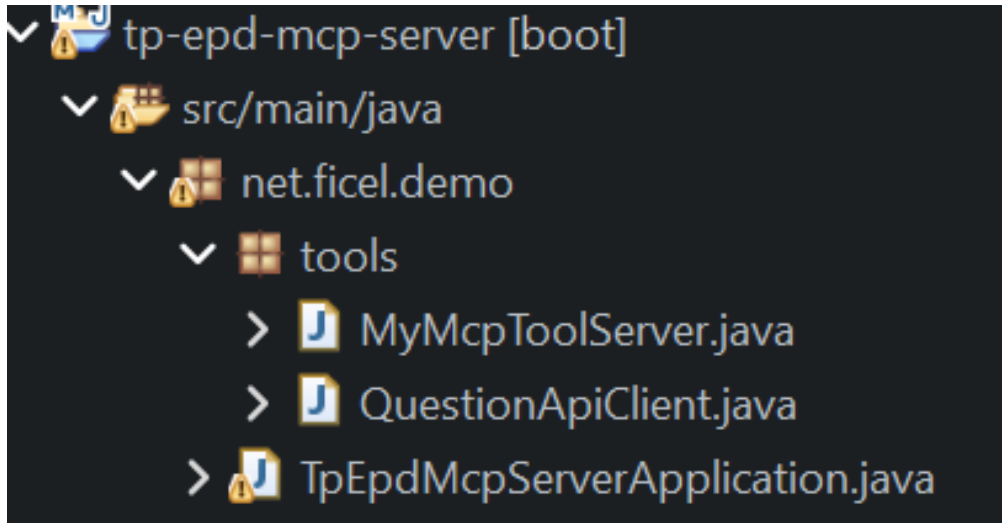
- Créer un nouveau projet Spring Boot qui agit comme un client pour votre API REST **et un serveur MCP**.
- Ce serveur doit servir d'intermédiaire entre : un LLM et votre API Spring Boot de recherche sémantique.
- Ajouter les dépendances suivantes dans le pom.xml :

```xml
<dependencies>

    <!--
        Spring AI – MCP Server (Model Context Protocol)
        transforme ton app Spring en serveur MCP consommable par un LLM
    -->
    <dependency>
        <groupId>org.springframework.ai</groupId>
        <artifactId>spring-ai-starter-mcp-server</artifactId>
    </dependency>

    <!--
        Spring Boot Web
        fournit le serveur HTTP et l'infrastructure REST
     -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

</dependencies>
```

application.properties

```
spring.application.name=tp-epd-mcp-server

# Configuration du serveur MCP
# Desactive le serveur Web pour plus de legerete (Transport Stdio)
spring.main.web-application-type=none
server.port=8877

# Configuration Identitée MCP
spring.ai.mcp.server.name=epd-demo-mcp
spring.ai.mcp.server.version=0.0.1

# Nettoyage de la console pour ne pas corrompre le flux Stdio
spring.main.banner-mode=off
logging.pattern.console=
# Redirige les logs vers stderr pour pouvoir debugger sans casser le protocole
logging.level.root=INFO

# URL de l'API cible (par défaut localhost:8080 si non défini)
api.questions.url=http://localhost:8080
```

Implémenter votre logique métier


tp-epd-mcp-server [boot]
src/main/java
net.ficel.demo
tools
MyMcpToolServer.java
QuestionApiClient.java
TpEpdMcpServerApplication.java

Package tools

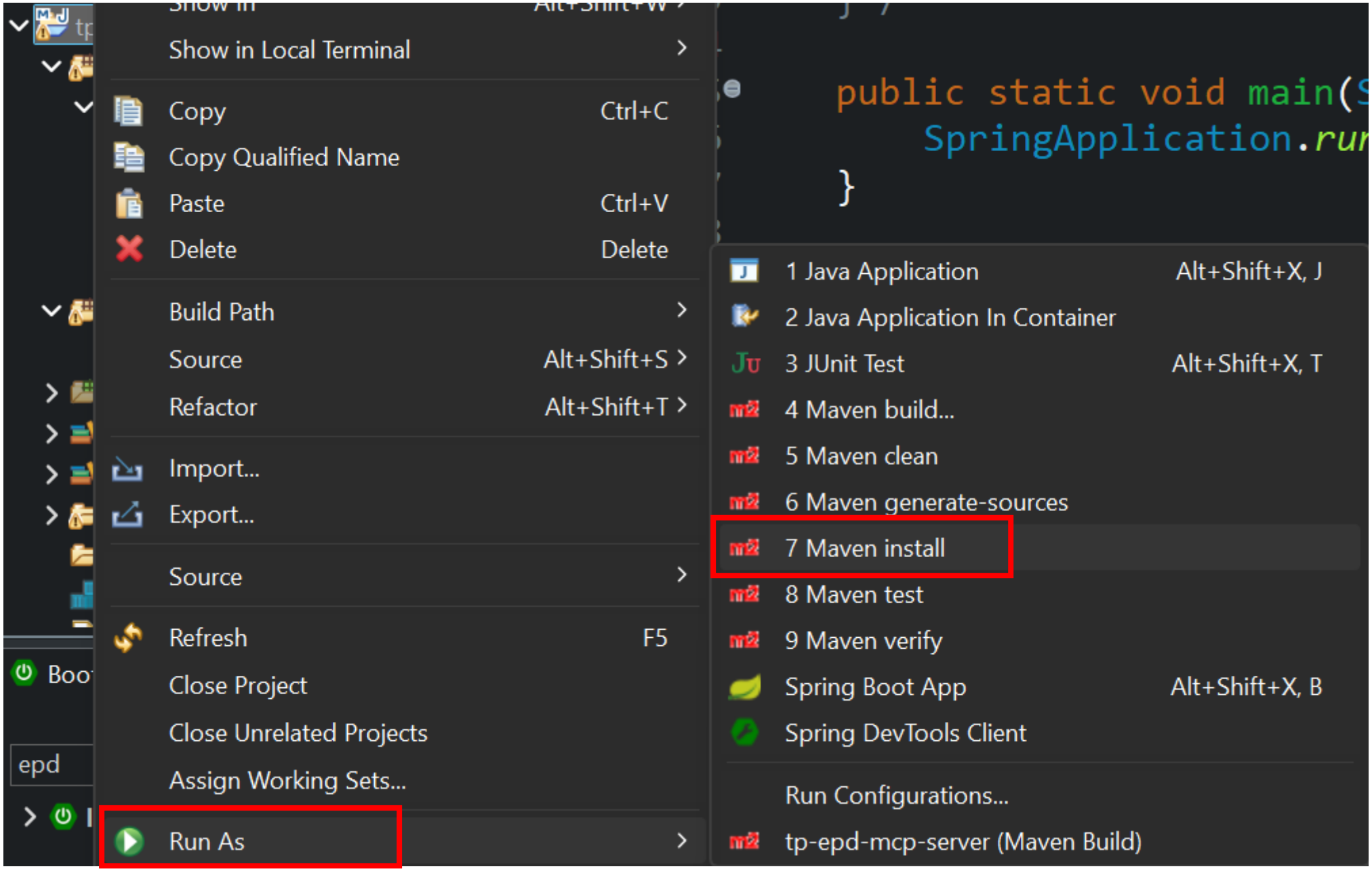
```java
package net.ficel.demo.tools;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.http.HttpStatusCode;
import org.springframework.stereotype.Service;
import org.springframework.web.client.RestClient;

@Service
public class QuestionApiClient {
    private final RestClient restClient;

    public QuestionApiClient(@Value("${api.questions.url:http://localhost:8080}") String baseUrl) {
        this.restClient = RestClient.builder()
                .baseUrl(baseUrl)
                .build();
    }

    public String postQuestion(String content) {
        return restClient.post()
                .uri("/api/v1/questions")
                .body(new QuestionRequest(content))
                .retrieve()
                .onStatus(HttpStatusCode::is4xxClientError, (request, response) -> {
                    throw new RuntimeException("Client Error: " + response.getStatusCode());
                })
                .body(String.class);
    }

    // Record simple pour le JSON
    public record QuestionRequest(String question) {}
}
```

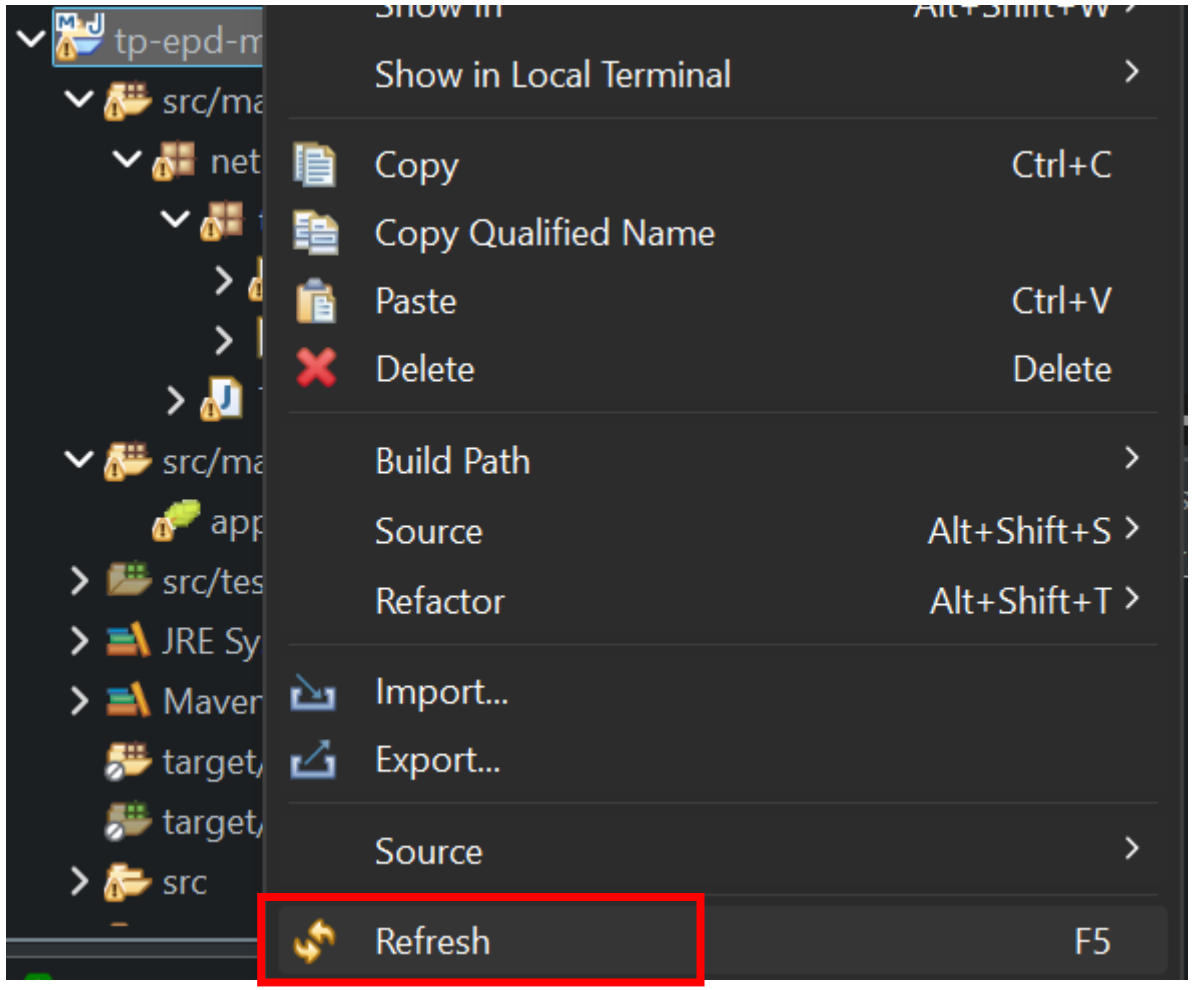
```java
package net.ficel.demo.tools;

import org.springframework.ai.tool.annotation.Tool;
import org.springframework.stereotype.Component;

@Component
public class MyMcpToolServer {
    private final QuestionApiClient apiClient;

    public MyMcpToolServer(QuestionApiClient apiClient) {
        this.apiClient = apiClient;
    }

    @Tool(description = "Get server name")
      public String getServerName() {
            return "Demo d'un serveur MCP pour le TP EPD";
      }

    @Tool(description = "Permet de poser une question à l'API interne sur le port 8080")
    public String askApiQuestion(String question) {
        try {
            return apiClient.postQuestion(question);
        } catch (Exception e) {
            return "Erreur lors de l'appel à l'API : " + e.getMessage();
        }
    }

}
```

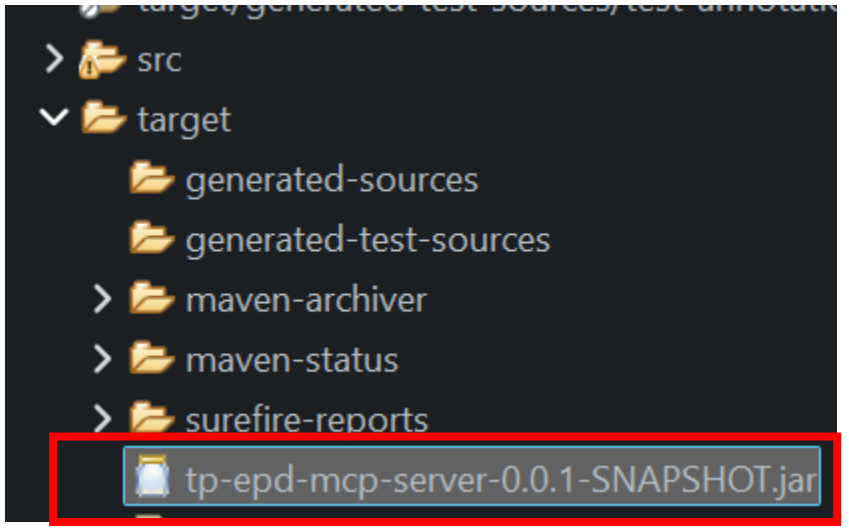
La classe principale

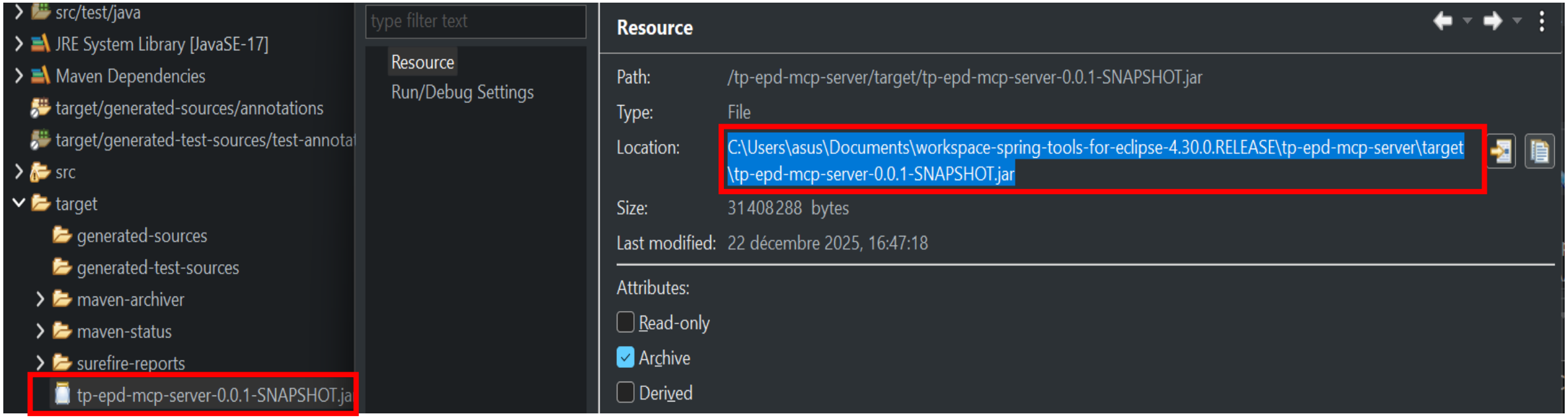
```java
package net.ficel.demo;

import java.util.List;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;

import org.springframework.ai.support.ToolCallbacks;
import org.springframework.ai.tool.ToolCallback;


import net.ficel.demo.tools.MyMcpToolServer;

@SpringBootApplication
public class TpEpdMcpServerApplication {

	public static void main(String[] args) {
		SpringApplication.run(TpEpdMcpServerApplication.class, args);
	}

	@Bean
    public List<ToolCallback> danTools(MyMcpToolServer mcpToolServer) {
        return List.of(ToolCallbacks.from(mcpToolServer));
    }

}
```

Compiler et récupérer le lien du Jar de votre serveur MCP


Show in Local Terminal
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Build Path
Source Alt+Shift+S
Refactor Alt+Shift+T
Import...
Export...
Source
Refresh F5
Close Project
Close Unrelated Projects
Assign Working Sets...
Run As
public static void main(
SpringApplication.ru
}
1 Java Application Alt+Shift+X, J
2 Java Application In Container
3 JUnit Test Alt+Shift+X, T
4 Maven build...
5 Maven clean
6 Maven generate-sources
7 Maven install
8 Maven test
9 Maven verify
Spring Boot App Alt+Shift+X, B
Spring DevTools Client
Run Configurations...
tp-epd-mcp-server (Maven Build)
epd

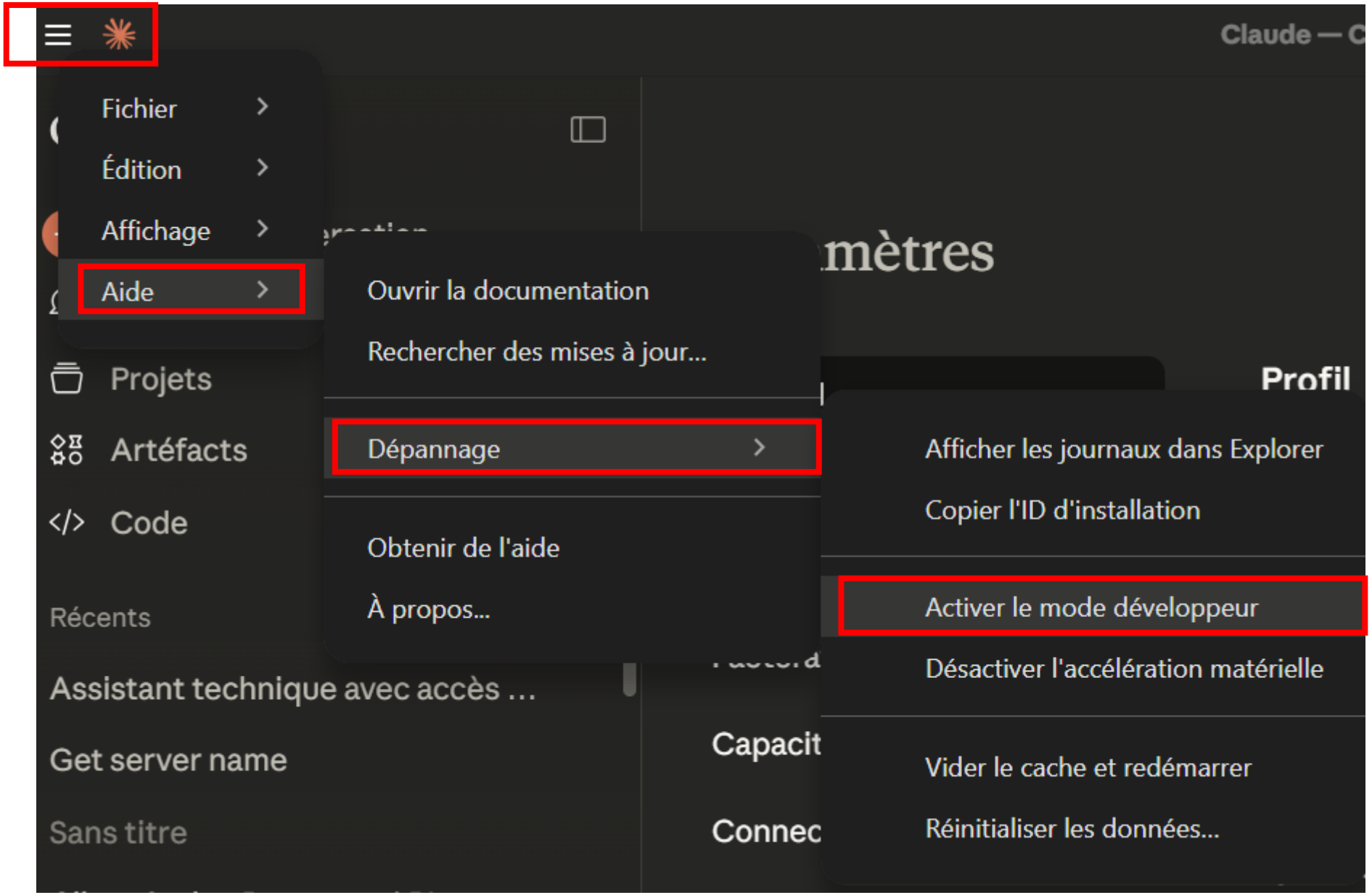

tp-epd-m
src/ma
net
src/ma
app
src/tes
JRE Sy
Maver
target/
target/
src
Show in Local Terminal
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Build Path
Source Alt+Shift+S
Refactor Alt+Shift+T
Import...
Export...
Source
Refresh F5

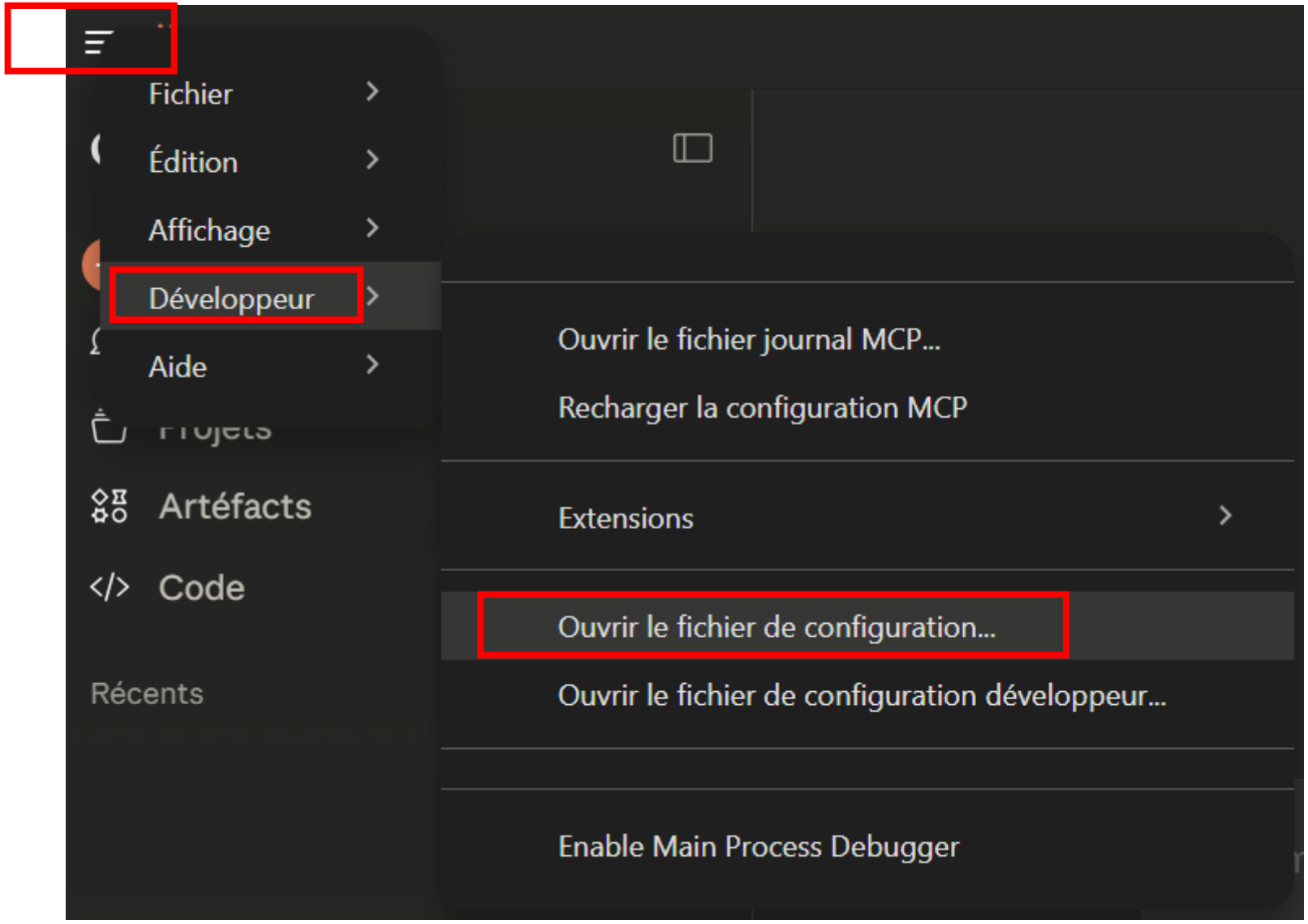

src
target
generated-sources
generated-test-sources
maven-archiver
maven-status
surefire-reports
tp-epd-mcp-server-0.0.1-SNAPSHOT.jar


src/test/java
JRE System Library [JavaSE-17]
Maven Dependencies
target/generated-sources/annotations
target/generated-test-sources/test-annotat
src
target
generated-sources
generated-test-sources
maven-archiver
maven-status
surefire-reports
tp-epd-mcp-server-0.0.1-SNAPSHOT.ja
type filter text
Resource
Run/Debug Settings
Resource
Path: /tp-epd-mcp-server/target/tp-epd-mcp-server-0.0.1-SNAPSHOT.jar
Type: File
Location: C:\Users\asus\Documents\workspace-spring-tools-for-eclipse-4.30.0.RELEASE\tp-epd-mcp-server\target\tp-epd-mcp-server-0.0.1-SNAPSHOT.jar
Size: 31408288 bytes
Last modified: 22 décembre 2025, 16:47:18
Attributes:
Read-only
Archive
Derived

Download Claude for desktop

[Download Claude | Claude]

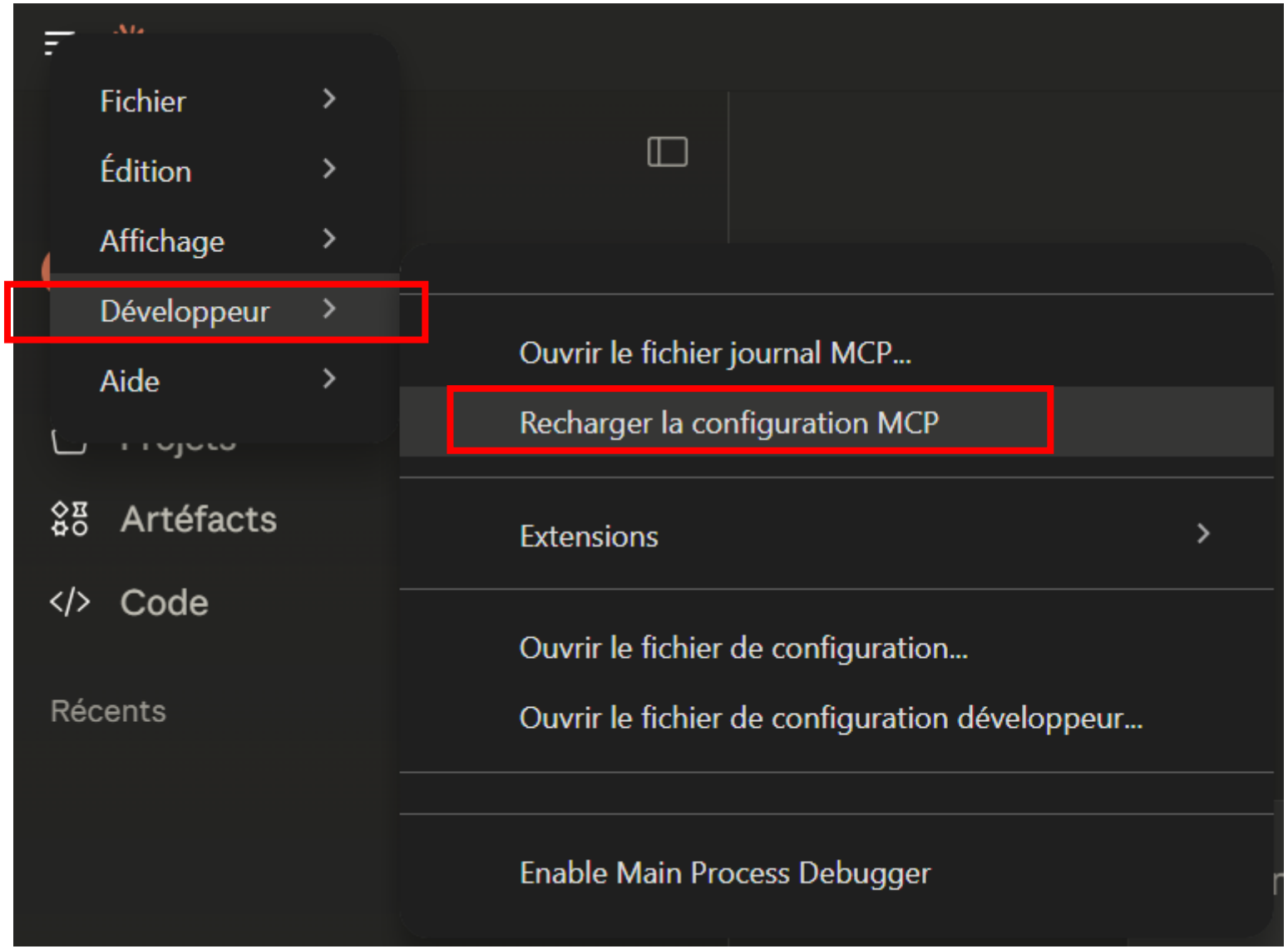

Claude — C
Fichier
Édition
Affichage
Aide
Ouvrir la documentation
Rechercher des mises à jour...
Dépannage
Obtenir de l'aide
À propos...
Afficher les journaux dans Explorer
Copier l'ID d'installation
Activer le mode développeur
Désactiver l'accélération matérielle
Vider le cache et redémarrer
Réinitialiser les données...
mètres
Profil
Projets
Artéfacts
Code
Récents
Assistant technique avec accès ...
Get server name
Sans titre
Capacit
Connec


Fichier
Édition
Affichage
Développeur
Aide
Artéfacts
Code
Récents
Ouvrir le fichier journal MCP...
Recharger la configuration MCP
Extensions
Ouvrir le fichier de configuration...
Ouvrir le fichier de configuration développeur...
Enable Main Process Debugger

Déclarer votre serveur MCP comme outil en ajoutant la configuration suivante à `"mcpServers"` dans le fichier de configuration JSON de Claude

Déclarer votre serveur MCP comme outil auprès de Claude Desktop

```
{
  "mcpServers": {
    "spring-ai-mcp-demo": {
      "command": "java",
      "args": [
        "-Dspring.ai.mcp.server.transport=STDIO",
        "-jar",
        "C:\\Users\\asus\\Documents\\workspace-spring-tools-for-eclipse-4.30.0.RELEASE\\tp-epd-mcp-server\\target\\tp-epd-mcp-server-0.0.1-SNAPSHOT.jar"
      ]
    }
  }

, ...
}
```


Fichier
Édition
Affichage
Développeur
Aide
Artéfacts
Code
Récents
Ouvrir le fichier journal MCP...
Recharger la configuration MCP
Extensions
Ouvrir le fichier de configuration...
Ouvrir le fichier de configuration développeur...
Enable Main Process Debugger

Partie 5

- Tester l'appel du serveur MCP par un LLM via un le prompt suivant :

```
Tu es un assistant technique disposant d'un accès à notre infrastructure interne via un serveur MCP.

Utilise uniquement les informations fournies dans ce contexte pour répondre aux questions. Si l'information n'est pas disponible dans le contexte, indique clairement que tu ne peux pas répondre basé sur les documents fournis.
```

Tu es un assistant technique disposant d'un accès à notre infrastructure interne via un serveur MCP.

Utilise uniquement les informations fournies dans ce contexte pour répondre aux questions. Si l'information n'est pas disponible dans le contexte, indique clairement que tu ne peux pas répondre basé sur les documents fournis.

Bonjour ! Je suis prêt à vous assister avec votre infrastructure interne via le serveur MCP.

Je comprends que je dois :

- Utiliser uniquement les outils MCP disponibles pour accéder aux informations de votre système
- Répondre uniquement en me basant sur les données retournées par ces outils
- Indiquer clairement quand je ne peux pas trouver l'information demandée

Tester les exemples de recherches sémantiques précédents

Recherche 4 – recommandation implicite

```
{
 "question": "Que faire après une mise à jour ?"
}
```

Résultats attendus (top)

☑ *"Après une mise à jour du système, un redémarrage complet est fortement recommandé."*

Que faire après une mise à jour ?

S askApiQuestion

D'après les informations disponibles dans votre système :

Après une mise à jour du système, un redémarrage complet est fortement recommandé.

Informations complémentaires pertinentes :

- Le service doit être redémarré après toute modification du fichier de configuration
- Un redémarrage à chaud est possible uniquement lorsque le système est en mode maintenance

S askApiQuestion

Requête

```
{
  `question`: `Que faire après une mise à jour ?`
}
```

Réponse


```
[{"id":"8d53f5c9-dbe8-4713-87e0-044f383a2cd2","text":"Après une mise à jour du système, un redémarrage complet est fortement recommandé.","metadata":{"category":"info","distance":0.2372267246246338},"score":0.7627732753753662},
```