|  |  |  |
| --- | --- | --- |
| **UNIVERSITE JOSEPH**  **KI-ZERBO**  **(UJKZ)** |  | **INSTITUT BURKINABE DES ARTS ET METIERS**  **(IBAM)** |
|  |  |  |

**Master 1 informatique**

**Module : DEVELOPPEMENT DE BASE DE COMPOSANT ET SERVICES WEB**

**Rapport de Projet :**

**Application RESTful avec Spring Boot, Talend Open Studio et Prométheus**

**GROUPE 2**

**Professeur : M KY OUMAR**

**Membres du groupe 2:**

* ***TRAORE /KABORE Hawa***
* ***YOUGBARE Simon***
* ***SAWADOGO Ernestine***
* ***OUEDRAOGO Latifatou***
* ***KOANDA Moustapha***

*Année Académique 2023-2024*

**SOMMAIRE**

Table des matières

[**I.** **Généralités** 3](#_Toc177466858)

[**1.** **Présentation** 3](#_Toc177466859)

[**2.** **Objectifs** 3](#_Toc177466860)

[**3.** **Tâches** 3](#_Toc177466861)

[**II.** **Outils et Technologies** 3](#_Toc177466862)

[**1.** **Technologies Utilisées** 3](#_Toc177466863)

[**2.** **Environnement de Développement** 4](#_Toc177466864)

[**3.** **Répertoire Git** 4](#_Toc177466865)

[**III.** **Développement de l'application** 4](#_Toc177466866)

[**1.** **Extraction et Transformation de Données avec Talend Open Studio** 4](#_Toc177466867)

[**2.** **Création de l'API RESTful avec Spring Boot** 6](#_Toc177466868)

[**3.** **Monitoring avec Prometheus** 9](#_Toc177466869)

[**4.** **Tests avec Postman** 10](#_Toc177466870)

[**IV.** **Procédure d'installation et de test** 10](#_Toc177466871)

[**1.** **Prérequis** 10](#_Toc177466872)

[**2.** **Installation de l'application** 11](#_Toc177466873)

[**3.** **Monitoring avec Prometheus** 12](#_Toc177466874)

[**4.** **Test avec Postman** 14](#_Toc177466875)

[**Conclusion** 17](#_Toc177466876)

### **Généralités**

### **Présentation**

Le présent projet est réalisé dans le cadre du développement d'une application RESTful utilisant Spring Boot pour exposer des données extraites et transformées par Talend Open Studio à partir d'une base de données MySQL. Le monitoring des métriques est assuré via Prometheus.

Le choix du groupe s’est porté sur le thème « Gestion de la clientèle pour une entreprise informatique. »

### **Objectifs**

L'objectif principal du projet est de créer une application RESTful qui permet :

* L'extraction et la transformation de données de produits à partir d'une base de données MySQL/PostgreSQL via Talend Open Studio.
* L'exposition de ces données via une API RESTful développée avec Spring Boot.
* Le monitoring des performances et des métriques via Prometheus.
* Le test de l'application à l'aide de Postman et l'analyse des métriques dans Prometheus.

### **Tâches**

Utilisez Talend Open Studio pour extraire et transformer des données de produits dans MySQL/PostgreSql.

* Développez une application Spring Boot pour exposer ces données via une API RESTful.
* Intégrez des métriques de Prometheus dans l'application pour le monitoring.
* Testez l'application avec Postman et surveillez les métriques avec Prometheus.

### **Outils et Technologies**

### **Technologies Utilisées**

* **Spring Boot** (version 2.7.18): Pour le développement de l'API RESTful.
* **Talend Open Studio** (TOS\_DI-20211109\_1610-V8.0.1) pour extraire et transformer des données dans MySQL.
* **JDK 11** : pour la version de java compatible avec TOS.
* **MySQL** : Base de données utilisée pour stocker les données de produits.
* **Prometheus** (Prométheus-2.54.0.windows-amd64) pour le monitoring des métriques.
* **Postman** : Outil utilisé pour tester l'API RESTful.

### **Environnement de Développement**

* **IDE** : VS Code
* **Outils de versionnement** : Git avec hébergement sur GitHub
* **Base de données** : MySQL
* **Serveur d'applications** : Tomcat intégré à Spring Boot
* **Outils de tests** : Postman pour tester l'API REST

### **Répertoire Git**

Le projet est disponible publiquement sur un dépôt Git afin de faciliter l'accès, le téléchargement et les tests :

* Lien Git : *https://github.com/Yougsim/Developpement-base-de-composants-et-de-services-web\_Projet-SpringBoot\_Talend.git*

Le répertoire contient :

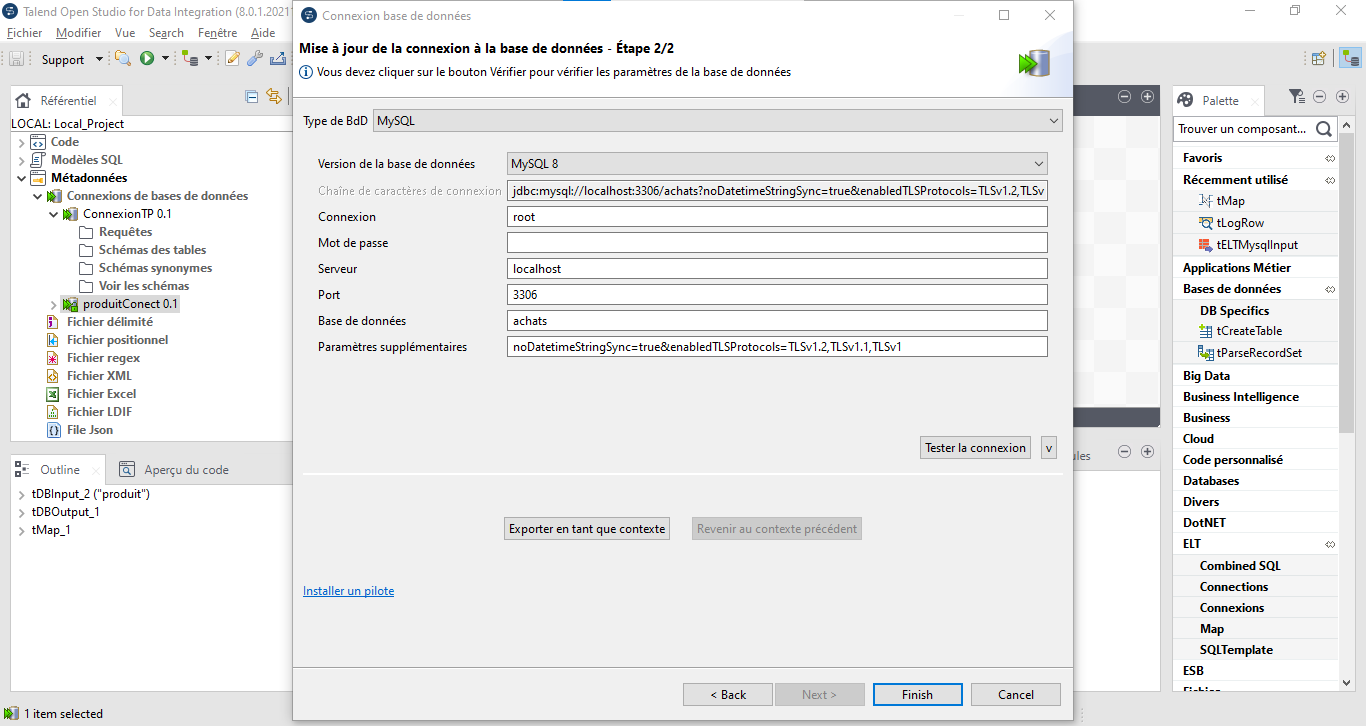
* Le code source de l'application Spring Boot.
* Les fichiers de configuration pour Talend Open Studio.
* Les fichiers de monitoring pour Prometheus.
* La procédure d'installation et de test.
* La présentation PowerPoint du projet.
* Le rapport du projet.

### **Développement de l'application**

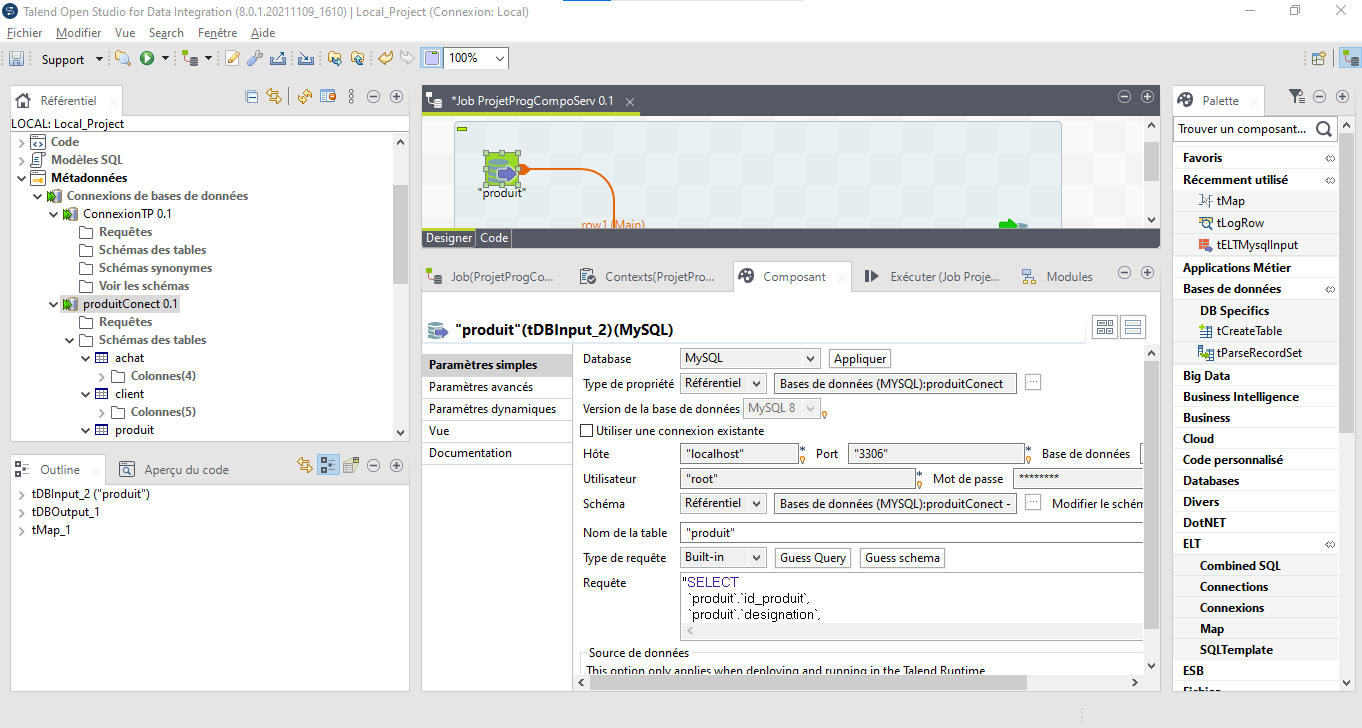
### **Extraction et Transformation de Données avec Talend Open Studio**

L'outil Talend Open Studio a été utilisé pour connecter la base de données MySQL et extraire les informations relatives aux produits. Le flux ETL a permis de transformer les données avant de les insérer dans la base de données cible.

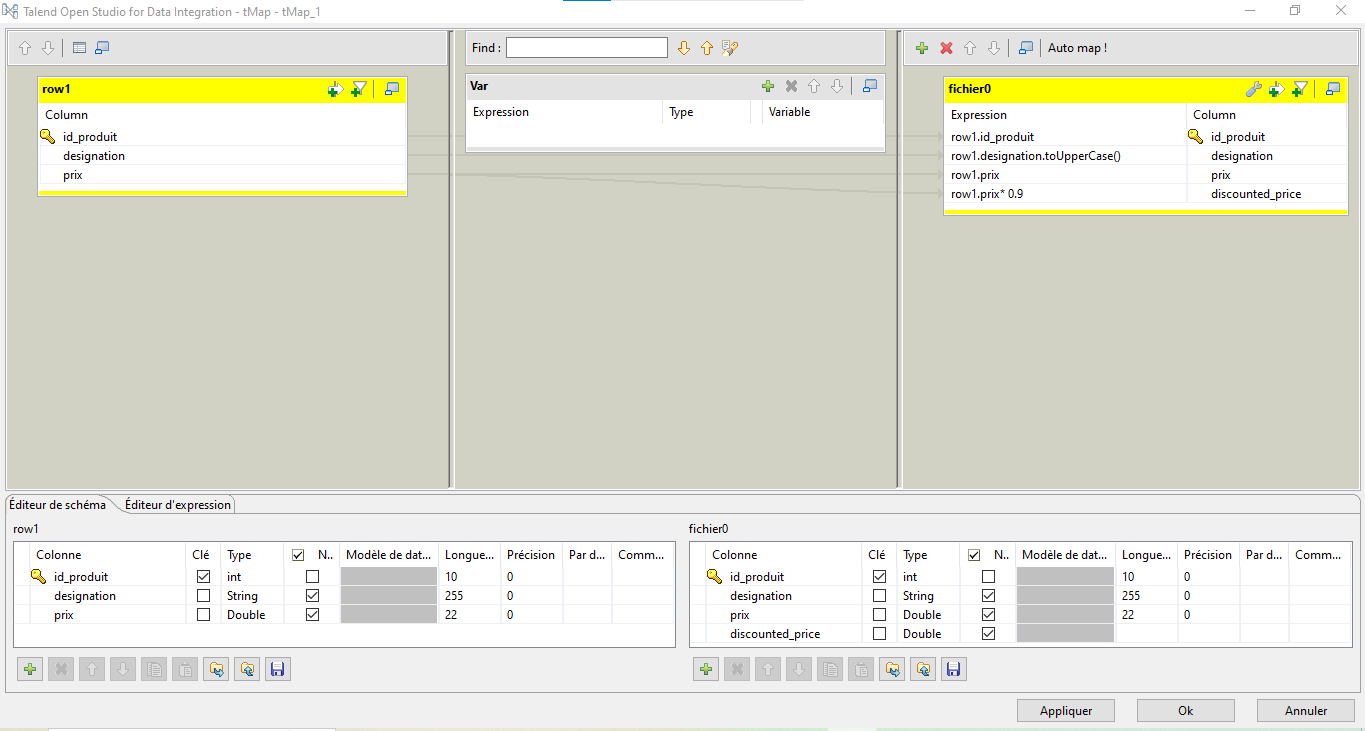
* Connexion à la base de données : cela a été fait avec le sous onglet « connexion de bases de données » de l’onglet « métadonnée » dans lequel on a créé notre connexion « produitconnect 0.1 »



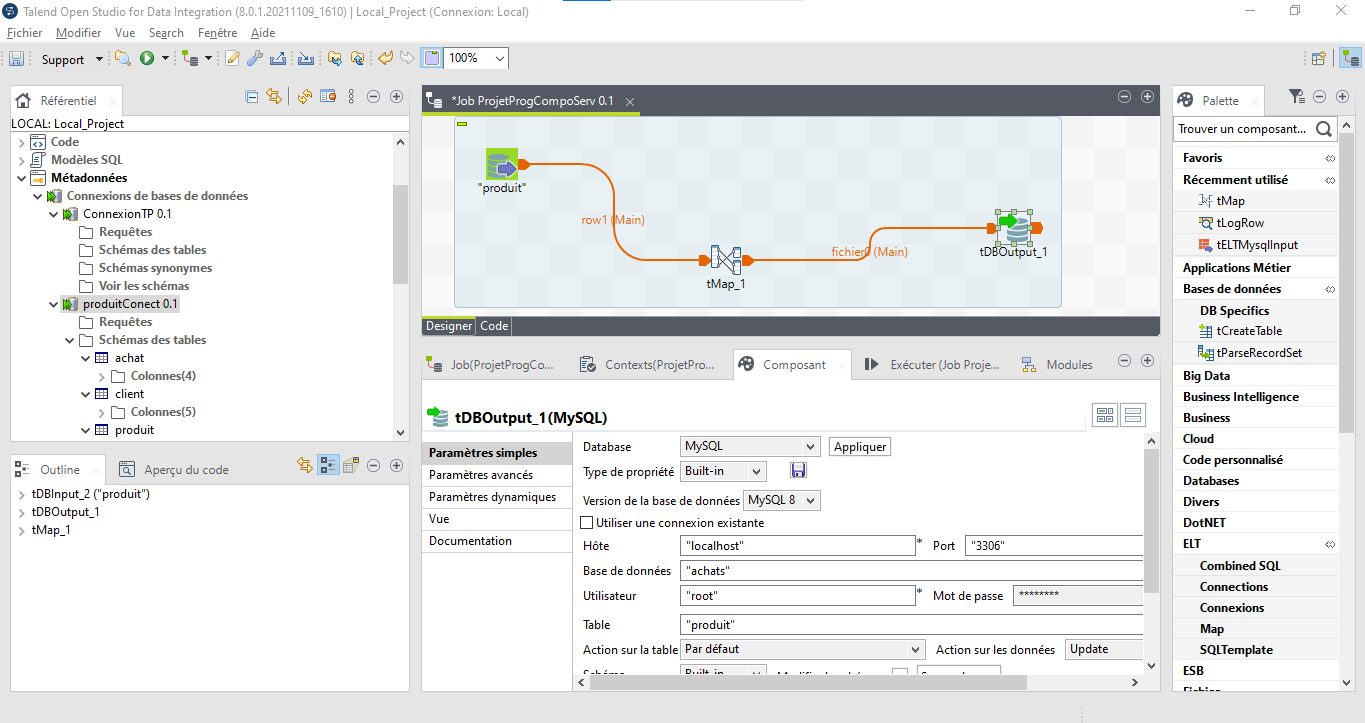
* Extraction des données avec tDBInput :



* Transformation des données avec Tmap dans lequel nous avons ajouté une colonne « discounted\_price » dans notre base de données qui permet de faire une réduction



* Insertion des données dans la base de données avec tDBOutput



### **Création de l'API RESTful avec Spring Boot**

L'API RESTful a été développée avec Spring Boot pour exposer les données de produits transformées via les points d'API suivants :

* **GET** http://localhost:8081/allProduits: Retourne la liste de tous les produits.
* @GetMapping("/allProduits")
* public ResponseEntity<List<Produit>> getAllProduits() {
* try {
* List<Produit> ProduitList = new ArrayList<>();
* produitRepo.findAll().forEach(ProduitList::add);
* if (ProduitList.isEmpty()) {
* return new ResponseEntity<>(HttpStatus.NO\_CONTENT);
* }
* return new ResponseEntity<>(ProduitList, HttpStatus.OK);
* } catch (Exception ex) {
* return new ResponseEntity<>(HttpStatus.INTERNAL\_SERVER\_ERROR);
* }
* }
* **GET** http://localhost:8081/showProduit/{id}Retourne les détails d'un produit spécifique.
* @GetMapping("/showProduit/{id}")
* public ResponseEntity<Produit> showProduit(@PathVariable(value = "id") Integer id) {
* Optional<Produit> optional = produitRepo.findById(id);
* Produit produit = null;
* if (optional.isPresent()) {
* produit = optional.get();
* } else {
* throw new RuntimeException(" product not found for id :: " + id);
* }
* return new ResponseEntity<>(produit, HttpStatus.OK);
* }
* **POST** http://localhost:8081/addProduit: Permet d'ajouter un nouveau produit.
* @PostMapping("/addProduit")
* public ResponseEntity<Produit> addProduit(@RequestBody Produit Produit) {
* Produit ProduitObj = produitRepo.save(Produit);
* return new ResponseEntity<>(ProduitObj, HttpStatus.OK);
* }
* **PUT** http://localhost:8081/produit/{id}: Met à jour les informations d'un produit.
* @PutMapping("/produit/{id}")
* public ResponseEntity<Produit> updateProduit(@PathVariable Integer id, @RequestBody Produit updateProduit) {
* try {
* Produit updated = produService.updateProduit(id, updateProduit);
* return ResponseEntity.ok(updated);
* } catch (Exception e) {
* return ResponseEntity.status(HttpStatus.INTERNAL\_SERVER\_ERROR).build();
* }
* }
* **DELETE** http://localhost:8081/deleteProduit/{id}: Supprime un produit.
* @DeleteMapping("/deleteProduit/{id}")
* public ResponseEntity<Void> deleteProduit(@PathVariable Integer id) {
* produitRepo.deleteById(id);
* return ResponseEntity.noContent().build();
* }

### **Monitoring avec Prometheus**

**Prometheus**, un outil de monitoring open-source, il a été choisi pour ce projet en raison de sa capacité à collecter et à stocker des métriques de manière efficace, ainsi que son intégration native avec Spring Boot. Il sera chargé dans notre application de surveiller les performances du système et collecter les métriques des services exposés via une API RESTful. Ces métriques fournies permettront de suivre l’utilisation des ressources, la latence des requêtes, et d’autres indicateurs essentiels pour assurer la stabilité et la fiabilité de l’application.

* Afin de permettre à Prometheus de collecter des métriques depuis l’application Spring Boot, nous avons utilisé **Micrometer**, qui est une bibliothèque de surveillance intégrée avec Spring Boot Actuator. Les dépendances suivantes ont été ajoutées au fichier pom.xml du projet :

 <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-actuator</artifactId>

        </dependency>

        <dependency>

            <groupId>io.micrometer</groupId>

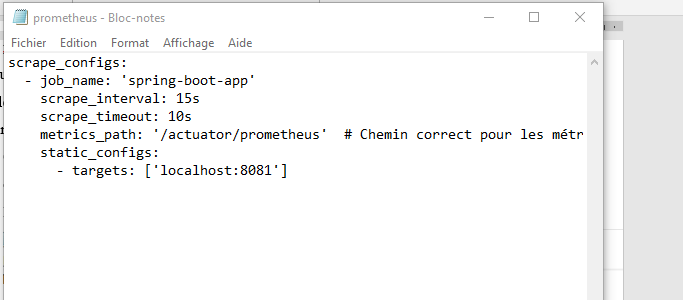
            <artifactId>micrometer-registry-prometheus</artifactId>

        </dependency>

* **Configuration d’Actuator** : Pour exposer les métriques via une API RESTful compatible avec Prometheus, le module **Spring Boot Actuator** a été configuré dans le fichier application.properties comme suit :
* management.endpoints.web.exposure.include=health,info,prometheus
* management.endpoint.prometheus.enabled=true

Cela expose les métriques au format Prometheus à l'URL : /actuator/prometheus.

* **Configuration de Prometheus** : Une fois l’application prête à fournir les métriques, nous avons configuré **Prometheus** pour collecter ces données. Le fichier de configuration prometheus.yml a été ajusté pour scrapper les métriques de l’application sur le port 8081 :



* **Déploiement de Prometheus** : Après avoir téléchargé Prometheus depuis son site officiel, il a été installé et lancé en utilisant la commande suivante depuis le répertoire d'installation :

./prometheus.exe --config.file=prometheus.yml

### **Tests avec Postman**

**Postman** est une application largement utilisée pour effectuer des tests API, permettant de simuler des requêtes HTTP et de valider les réponses des services web. Il offre une interface conviviale pour tester les fonctionnalités de notre application Spring Boot avant son déploiement en production.

Ce qui nous a permis de tester et valider les endpoints de l’API exposée par notre application Spring Boot.

* Envoyer des requêtes HTTP (GET, POST, PUT, DELETE) à l’API,
* Ajout, mise à jour et suppression de produits. Les tests ont permis de valider les performances et la précision des données exposées.

### **Procédure d'installation et de test**

### **Prérequis**

* Java JDK 11+ installé
* MySQL installé et configuré
* Talend Open Studio installé
* Prometheus installé
* Postman pour le test des requêtes API

### **Installation de l'application**

* Clonez le répertoire Git : git clone [https://github.com/Yougsim/Developpement-base-de-composants-et-de-services-web\_Projet-SpringBoot\_Talend.git]
* Télécharger et Installer JDK 11 :

**-**Télécharger JDK11 depuis Oracle ou AdoptOpenJDK.

- Installer JDK11

* Téléchargement du Projet :

-Télécharger le fichier ZIP du projet depuis le dépôt Git.

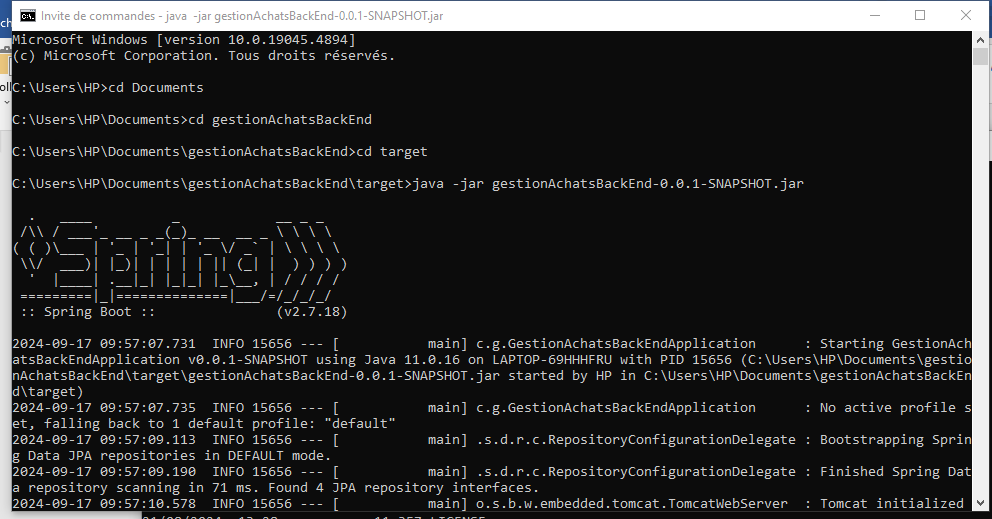
-Dézipper le fichier dans le répertoire de votre choix.

* Lancement de l'Application :

-Ouvrer un terminal ou une invite de commandes.

-Naviguer jusqu'au répertoire gestionAchatsBackEnd\target.

-Exécutez « : java -jar gestionAchatsBackEnd-0.0.1-SNAPSHOT.jar » pour lancer l'application



* Configuration de la Base de Données

-Installer le SGBD MySQL

-Créer une base de données "achats"

Configurez les accès à la base de données MySQL dans le fichier application.properties.

pring.application.name=gestionAchatsBackEnd

#developpement

spring.datasource.url=jdbc:mySql://localhost:3306/achats

spring.datasource.username=root

spring.datasource.password=

#production

#spring.datasource.url=jdbc:mySql://192.168.33.12:3306/achats

#spring.datasource.username=groupe2

#spring.datasource.password=1234

#spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

### **Monitoring avec Prometheus**

Configuration de Prometheus

* Téléchargement et Décompression :

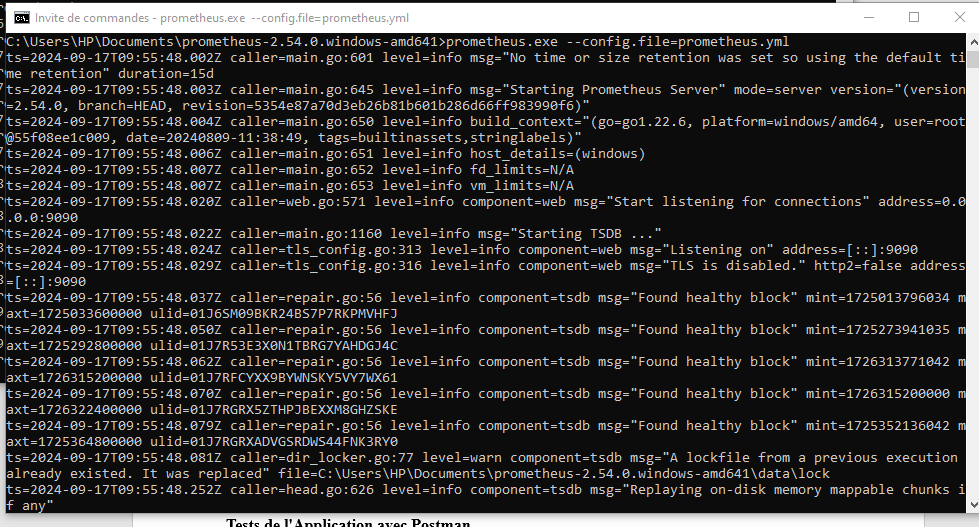
-Télécharger prometheus-2.54.0.windows-amd64.zip depuis le dépôt Git.

-Dézipper le dossier dans un répertoire de votre choix

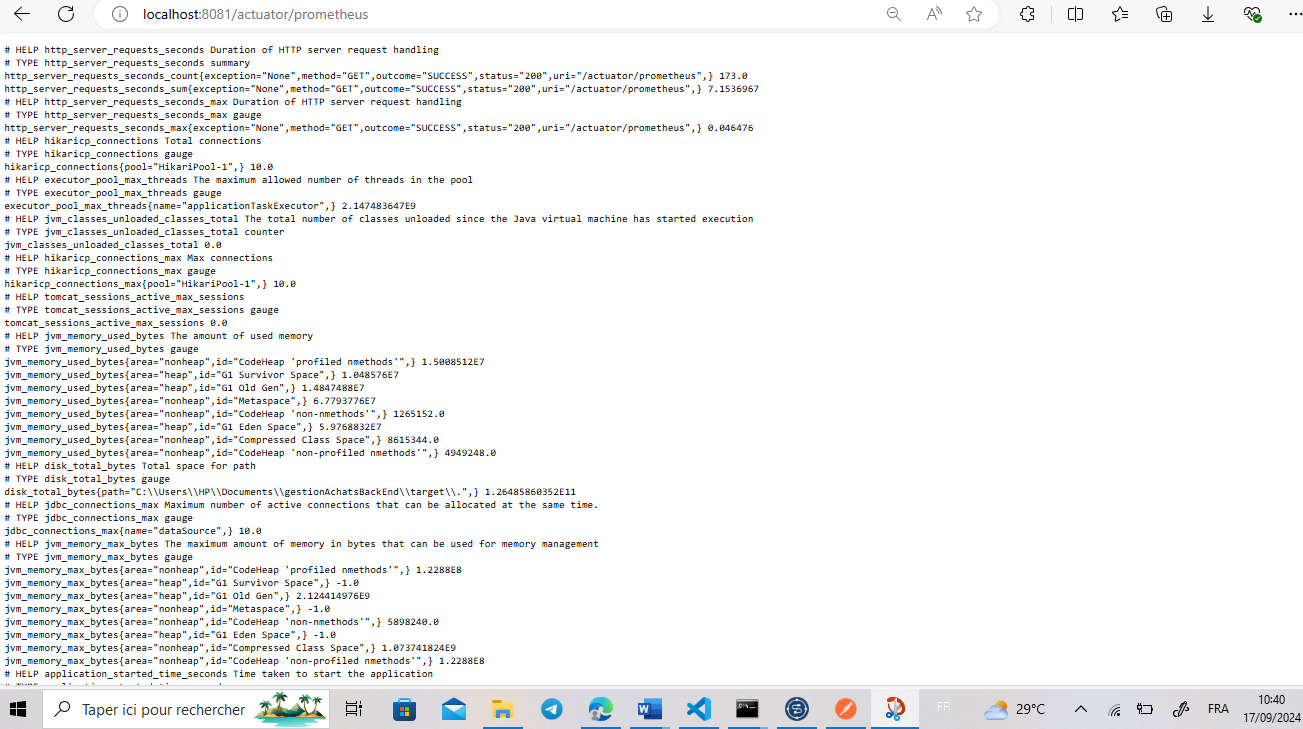
* Lancement de Prometheus :

-Ouvrer un terminal et naviguez jusqu'au répertoire contenant prometheus.yml et prometheus.exe.

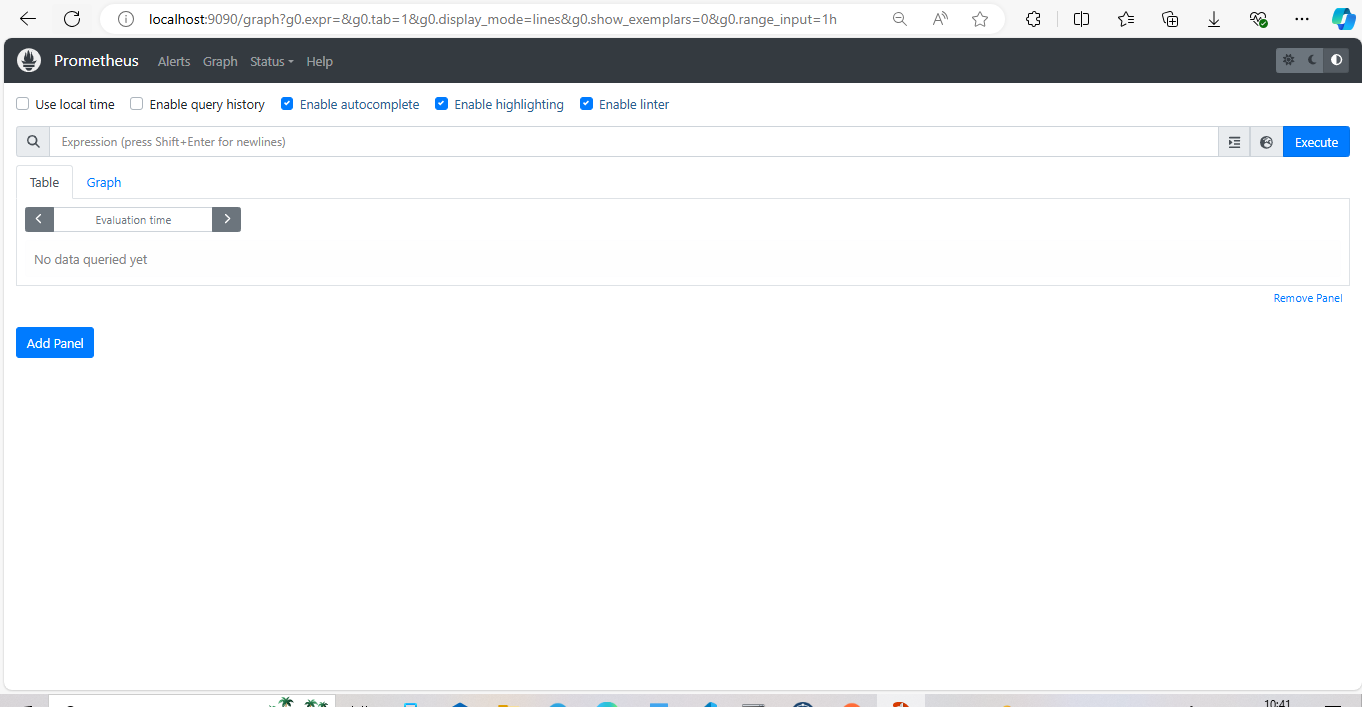
-Exécuter la commande suivante pour démarrer Prometheus : 'prometheus.exe --config.file=prometheus.yml’



-Ouvrer votre navigateur et allez à l'adresse 'localhost:8081/actuator/prometheus' pour vérifier les métriques.

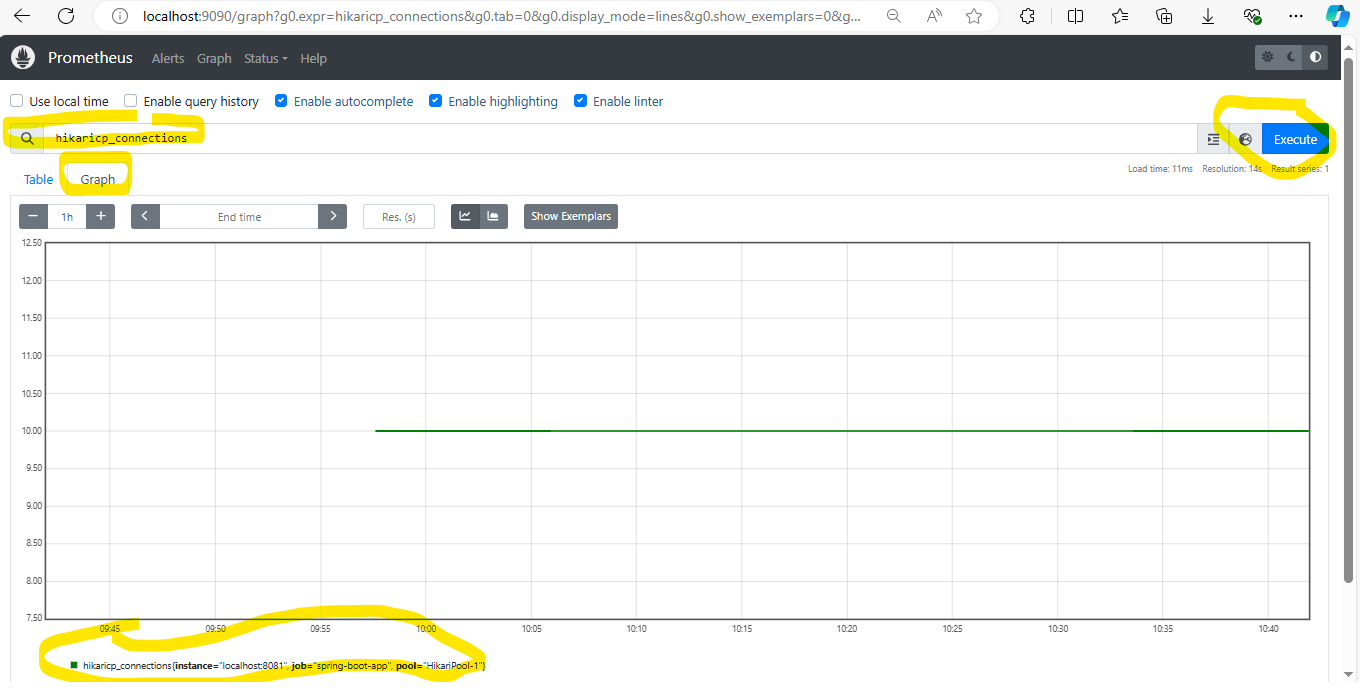


-Accéder à l'interface web de Prometheus en tapant 'http://localhost:9090/' dans votre navigateur.



-Dans l'onglet ‘Graph’, saisir les requêtes dans le champ de requête PromQL

-Cliquer sur Exécute pour voir les requêtes en cours d'exécution sur votre application



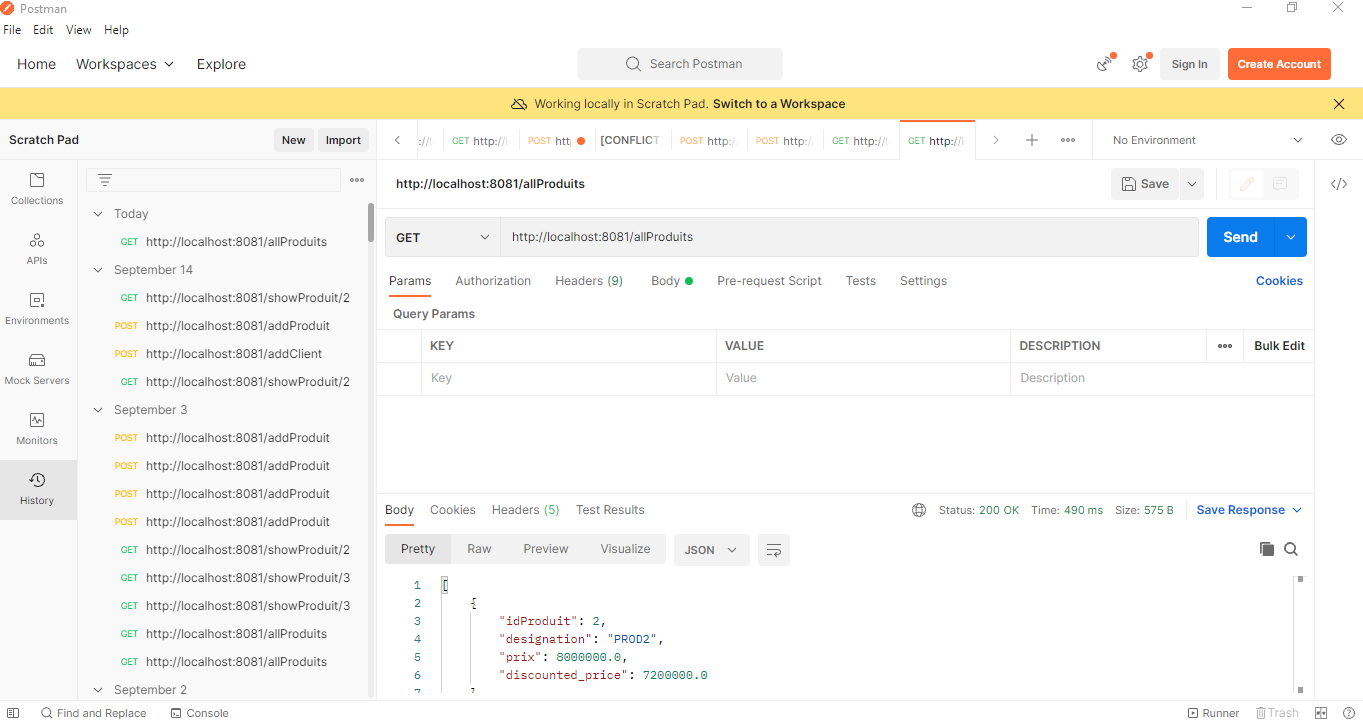
### **Test avec Postman**

**Ouvrez Postman.**

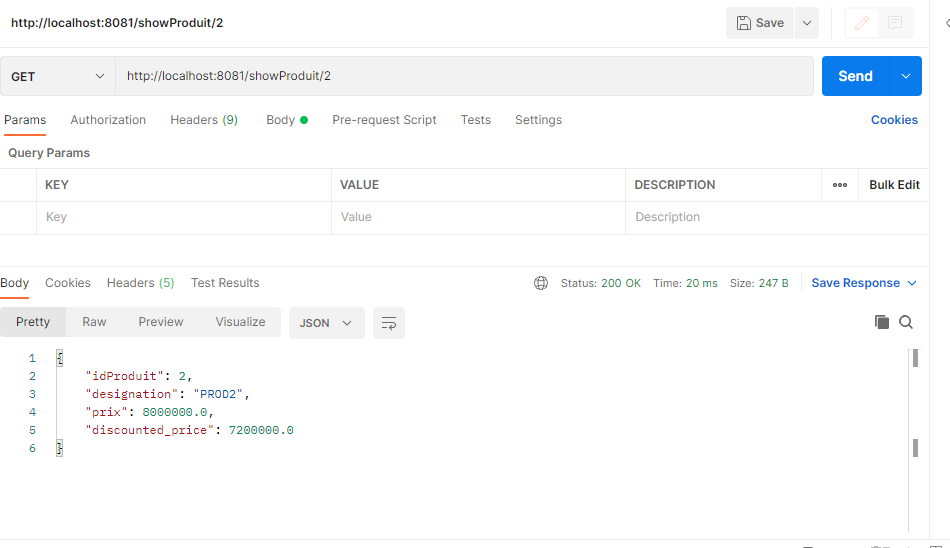
**Tests de l'Application avec Postman**

Test des endpoints de l’API pour tester l'application :

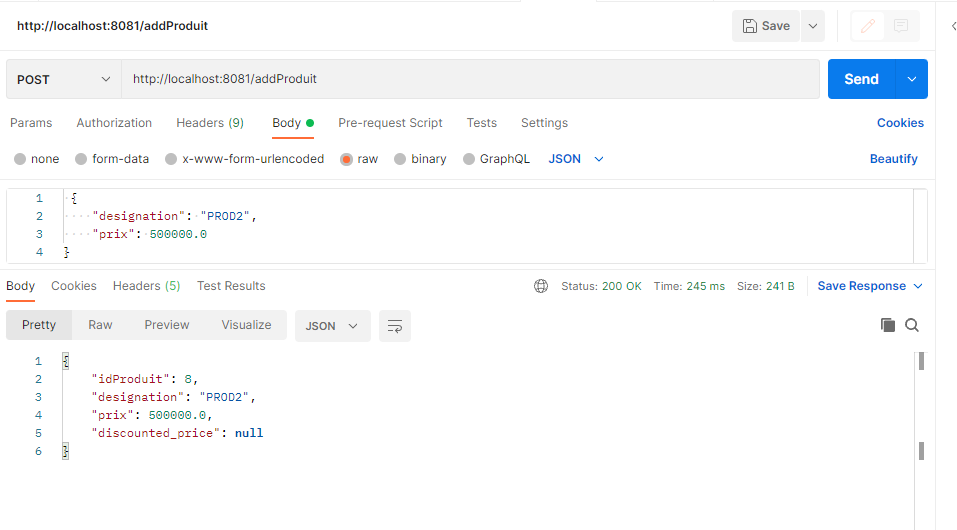
* GET : http://localhost:8081/allProduits (Récupère la liste de tous les produits).



* GET : http://localhost:8081/showProduit/3 (Affiche les détails du produit avec l'ID 3).



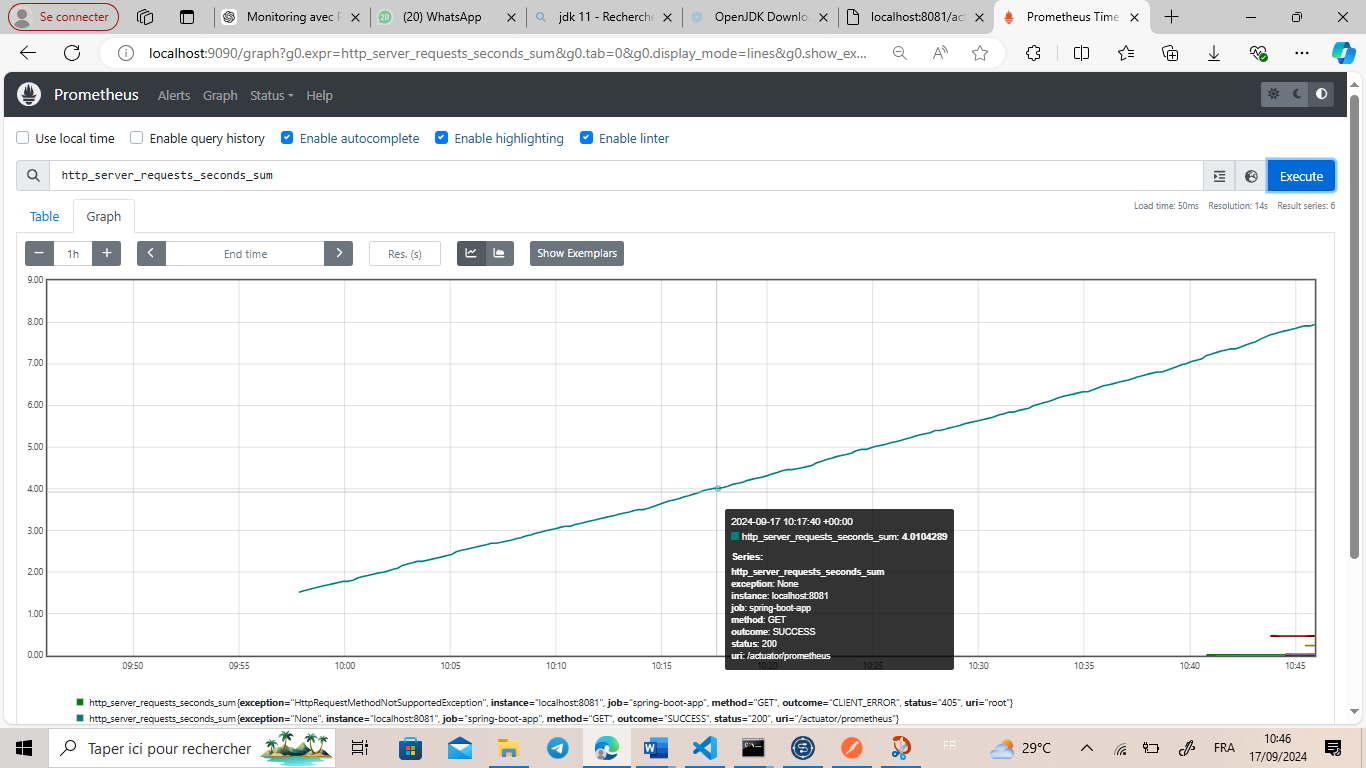
* POST : http://localhost:8081/addProduit (Ajoute un nouveau produit).



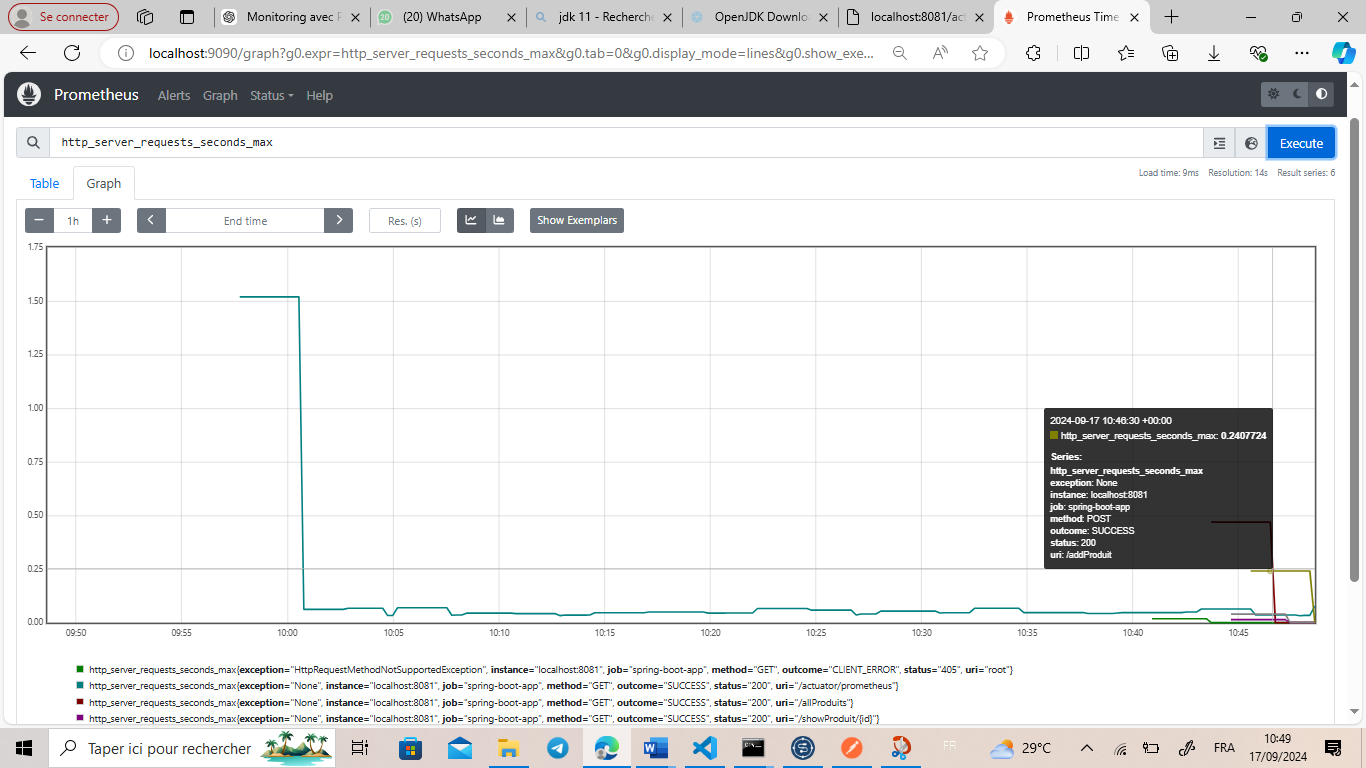
**Requêtes Prometheus :**

Utilisation des requêtes suivantes dans l'interface Prometheus :

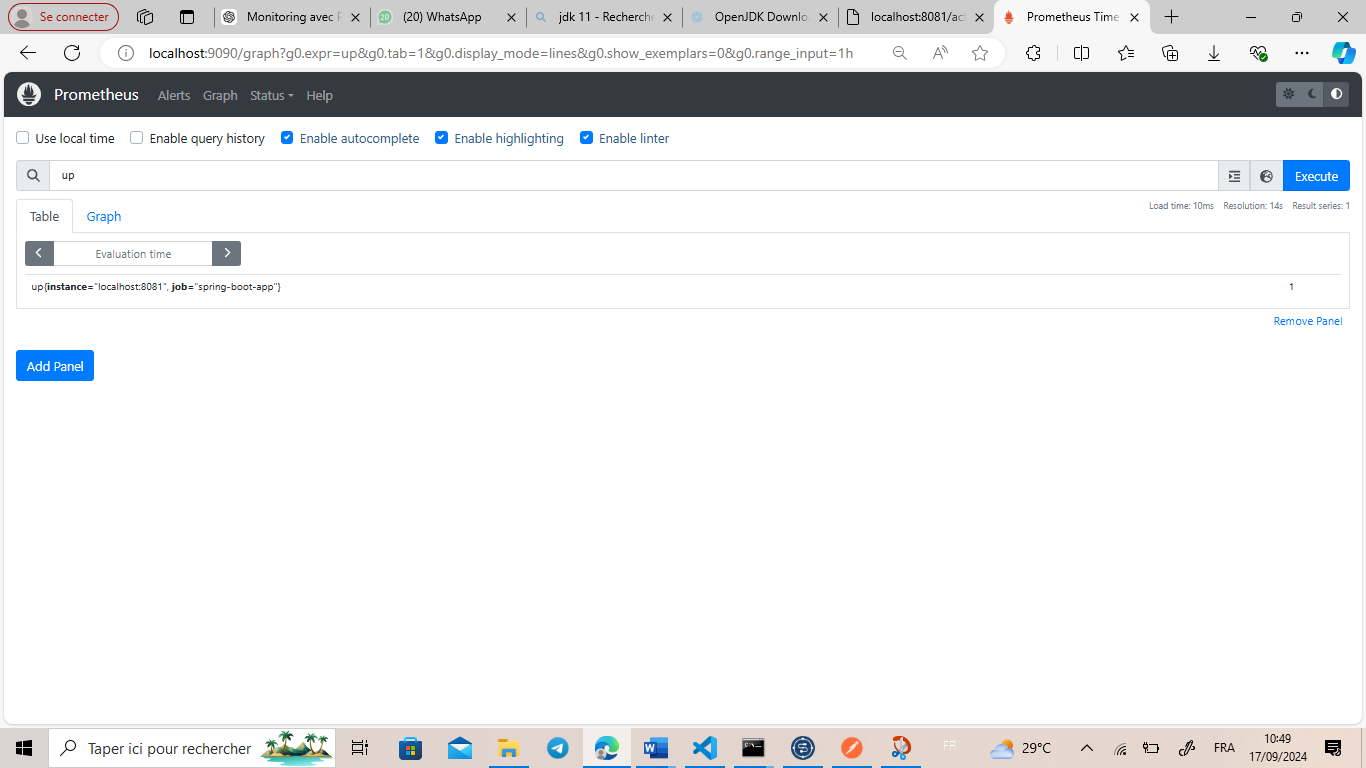
* http\_server\_requests\_seconds\_sum



* http\_server\_requests\_seconds\_max



* up pour verifier que le prometheus marche avec le projet



### **Conclusion**

Le développement de cette application RESTful a permis de mettre en pratique différentes technologies, notamment Spring Boot, Talend Open Studio et Prometheus. Ce projet a permis de renforcer les compétences de l'équipe en intégration de systèmes, en gestion des API, et en monitoring d'applications.