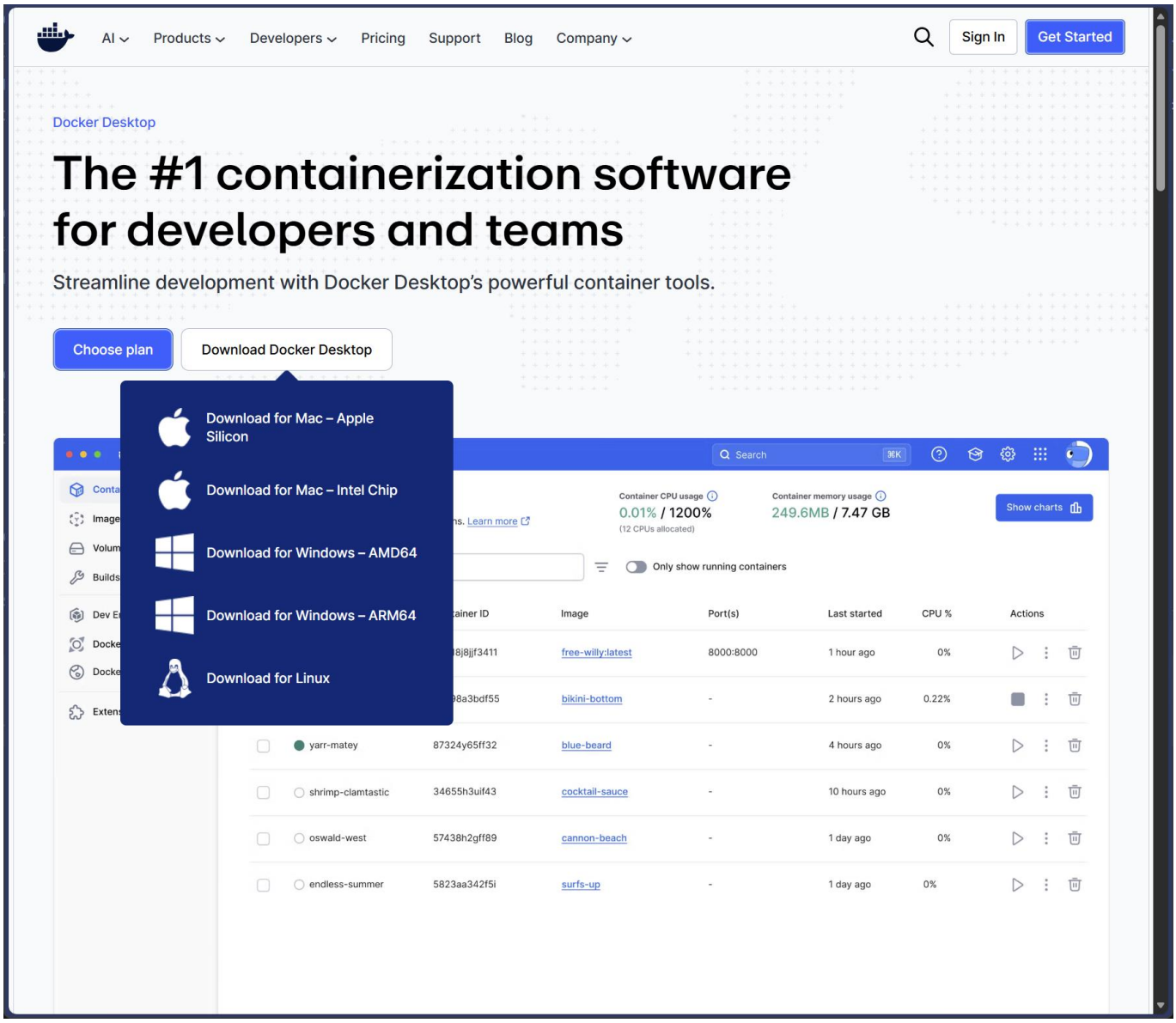
NFP 121 - Programmation avancée
Projet

Installation et configuration des outils

Docker desktop pour Windows

Télécharger et installer Docker desktop pour Windows via l'URL :

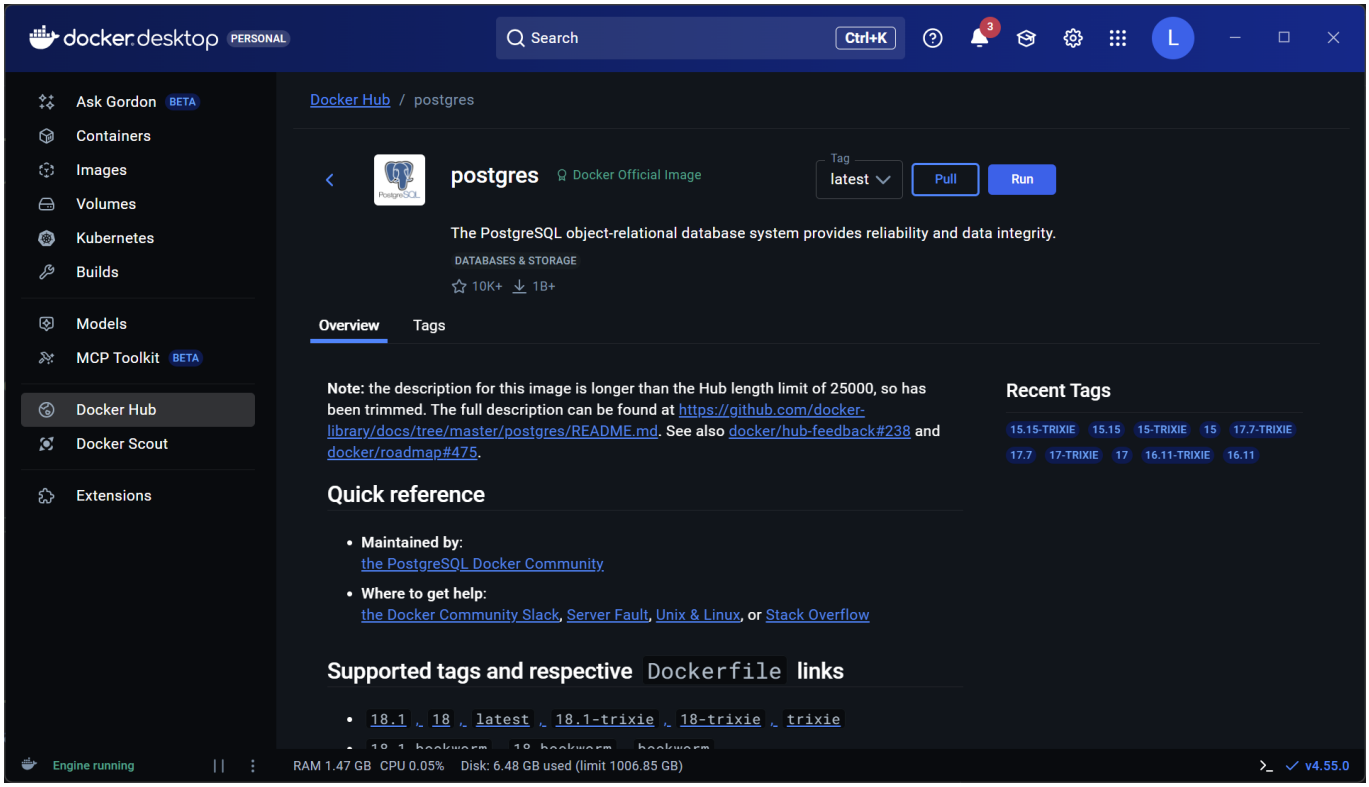
[Docker Desktop: The #1 Containerization Tool for Developers | Docker]



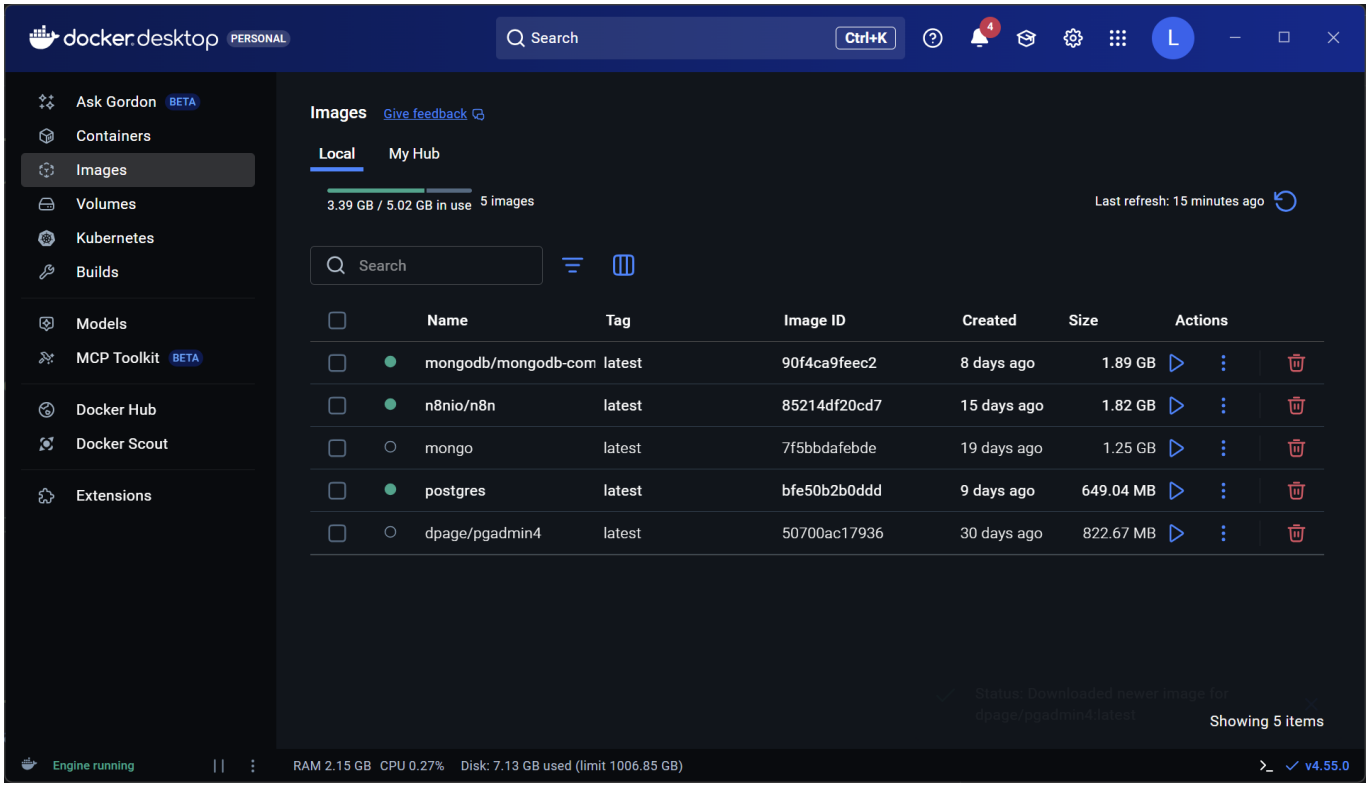
PostgreSQL sous docker

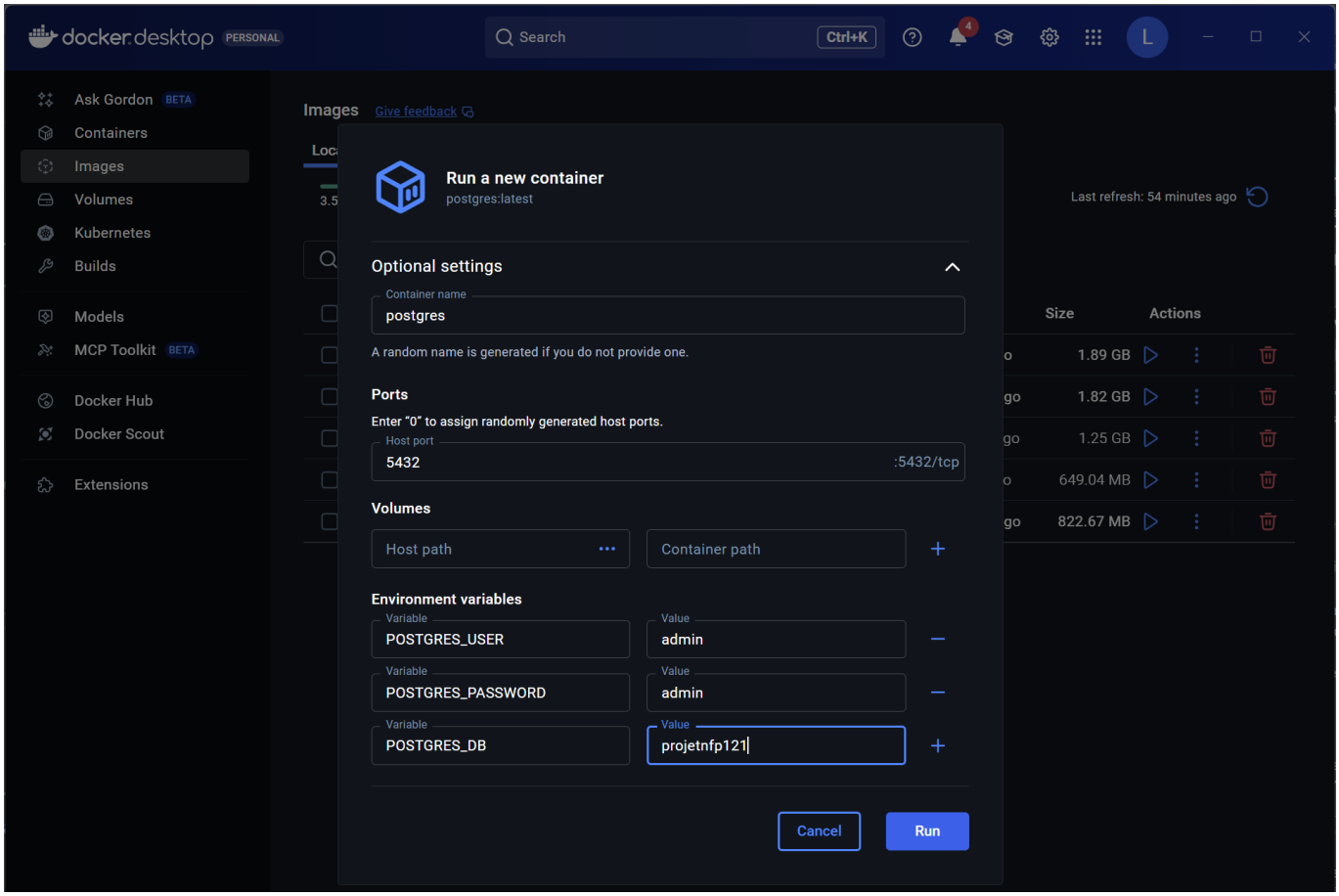
Dans Docker desktop, télécharger (Pull) l'image docker de **postgres** via l'url :

[postgres - Official Image | Docker Hub]



Exécuter l'image docker de **postgres** (crée et exécute un container)

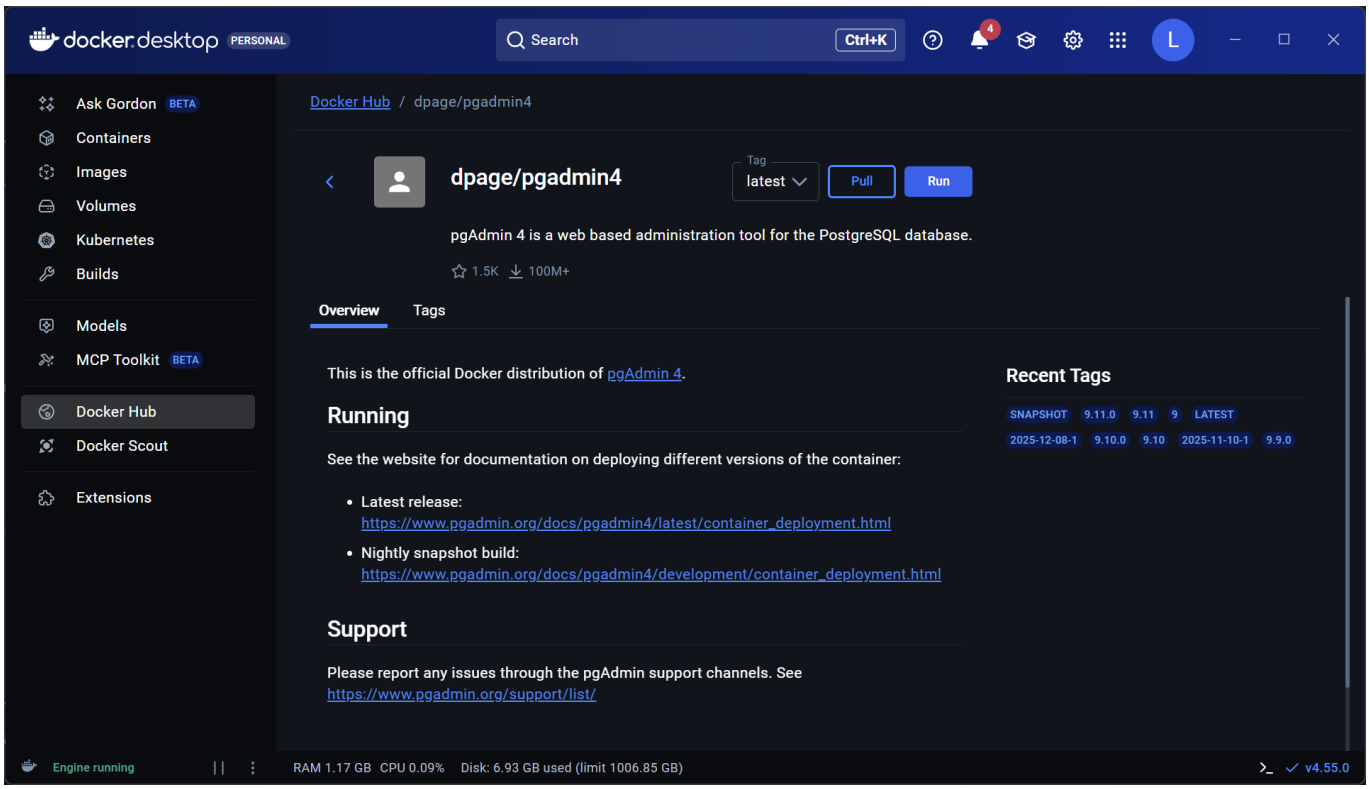




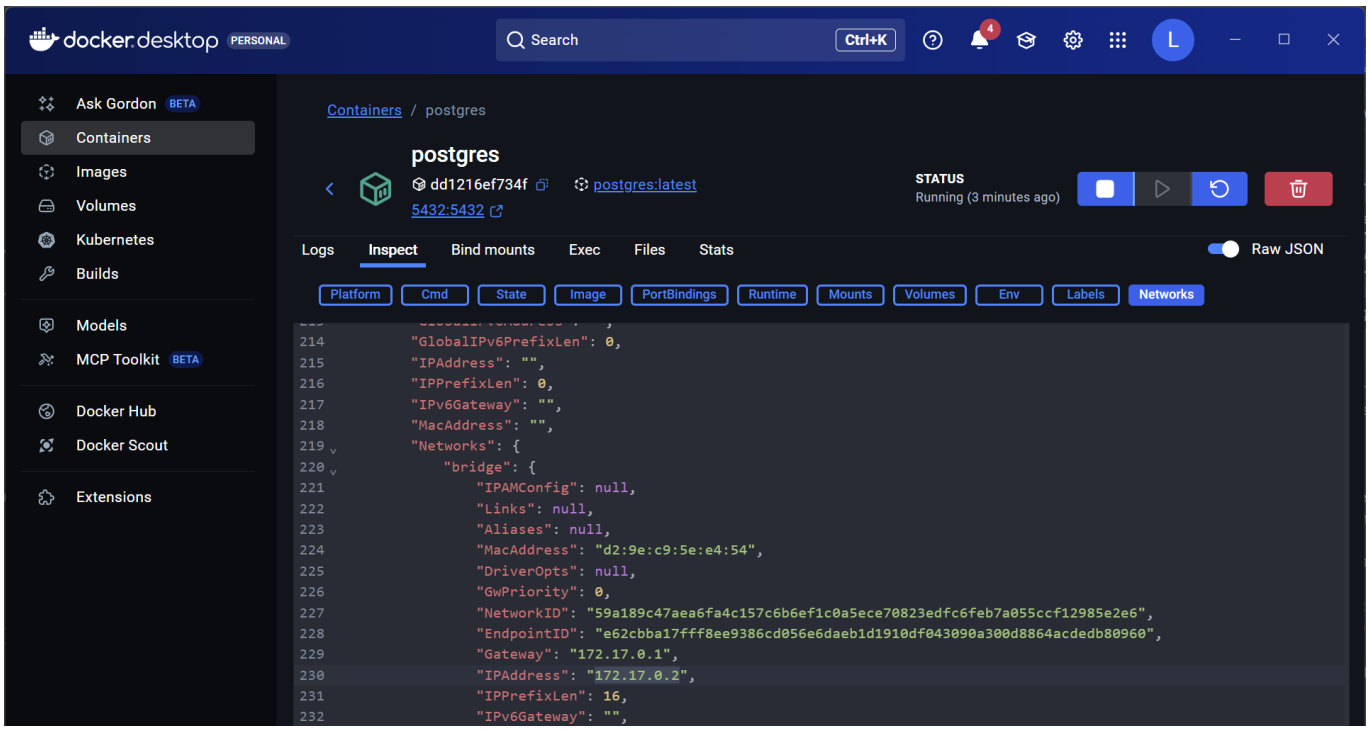
PGAdmin4 sous docker

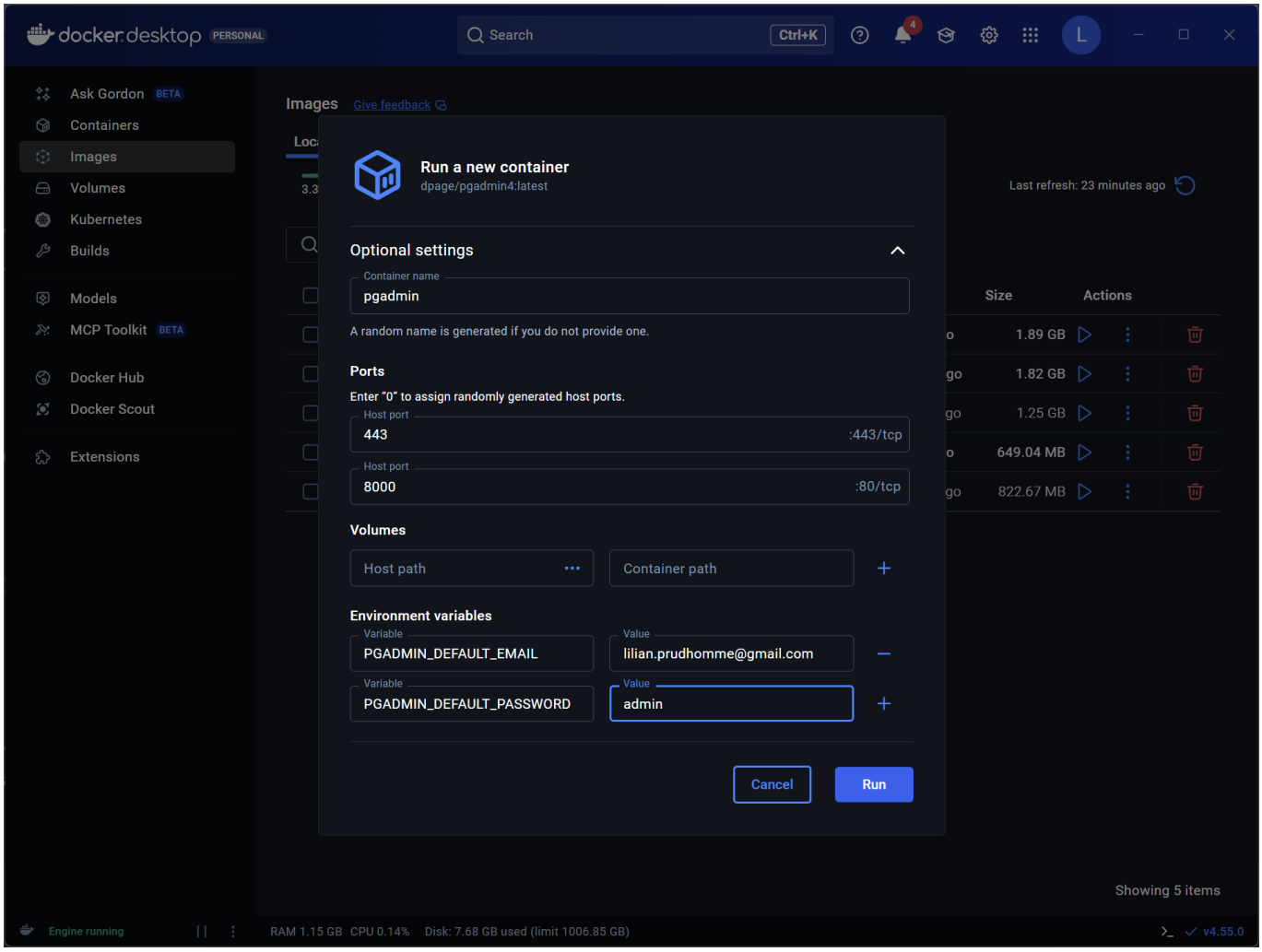
Dans Docker desktop, télécharger (Pull) l'image docker de **pgadmin4** via l'url :

[dpage/pgadmin4 - Docker Image]



Exécuter l'image docker de **pgadmin4** (crée et exécute un container)

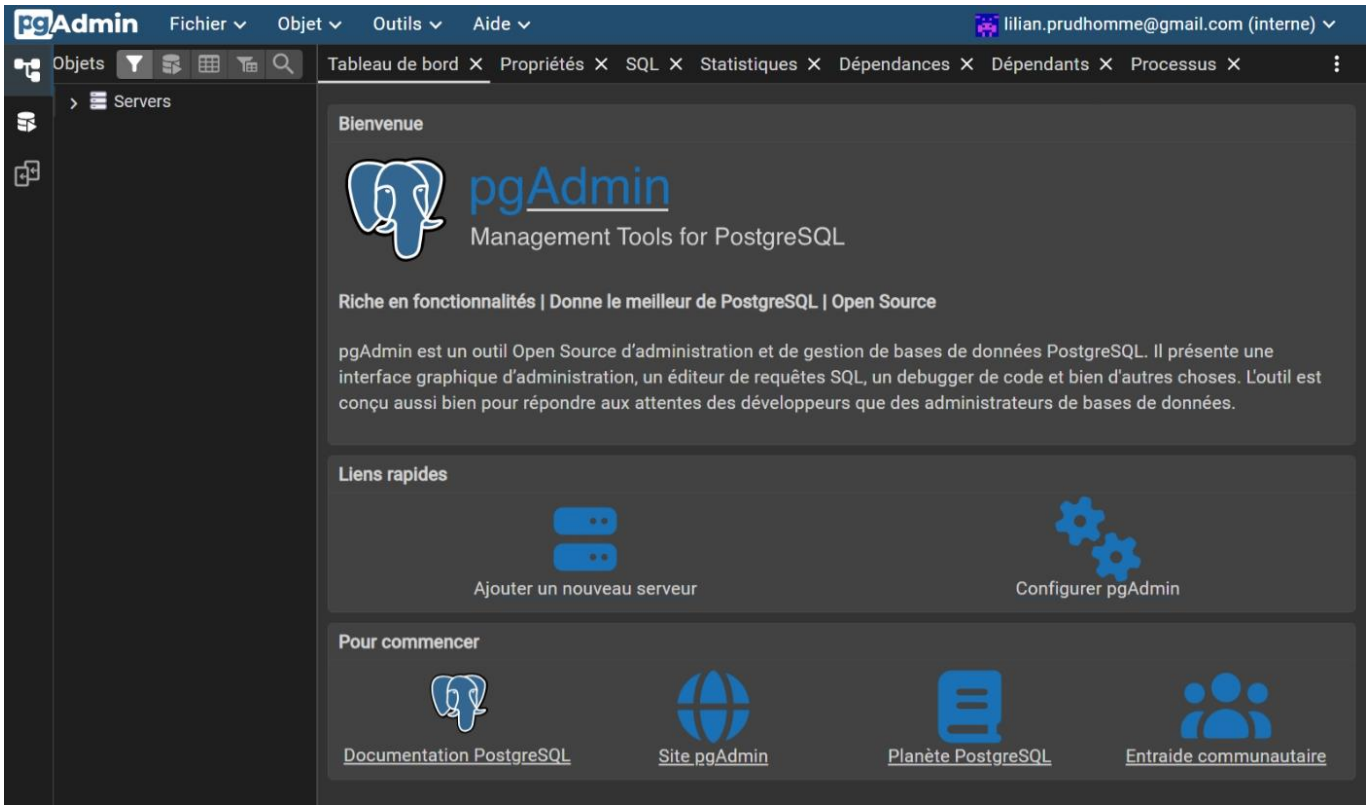


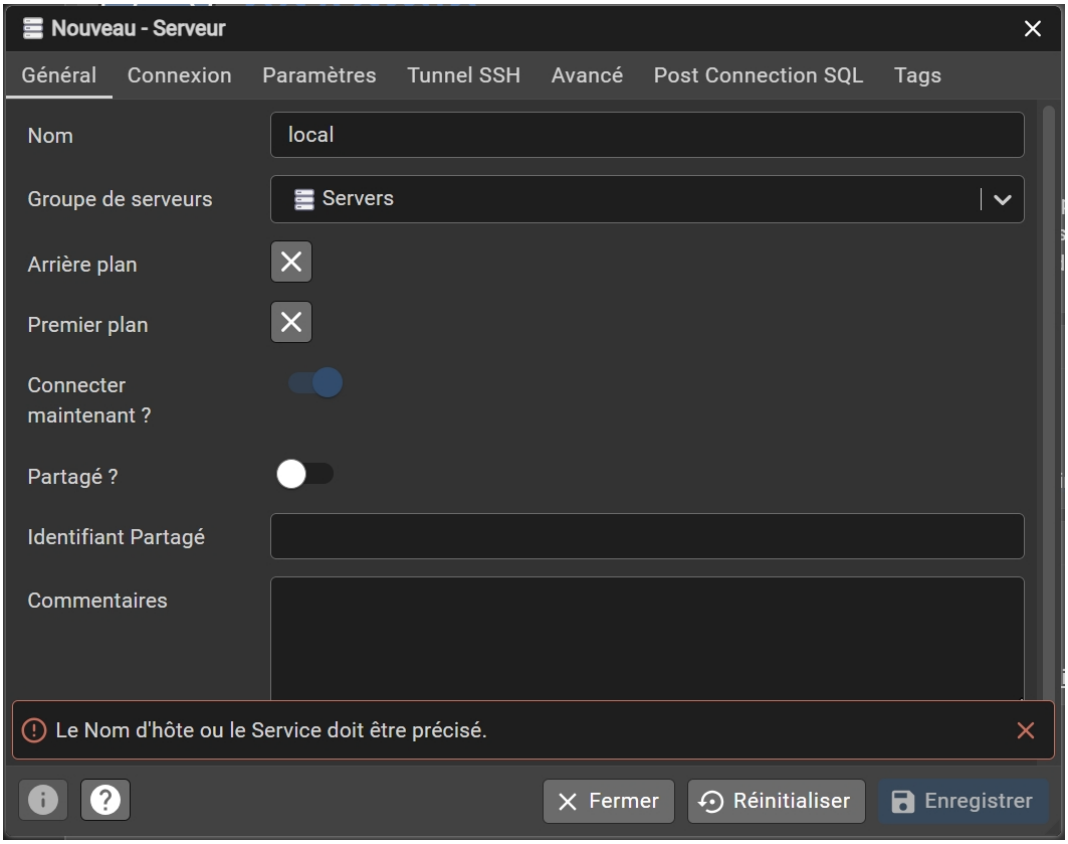


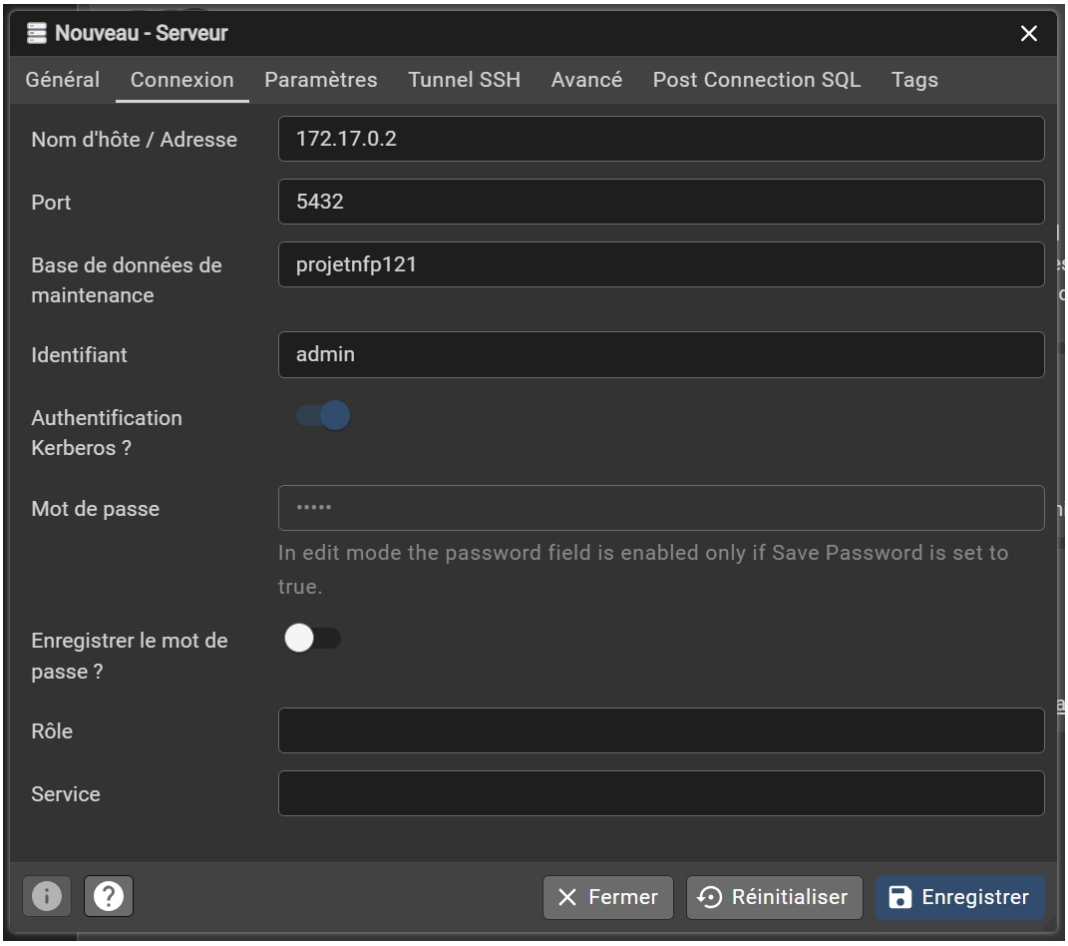
Configurer **pgadmin4** via l'url :

http://localhost:8000/

Ajouter le serveur **postgres** de docker :

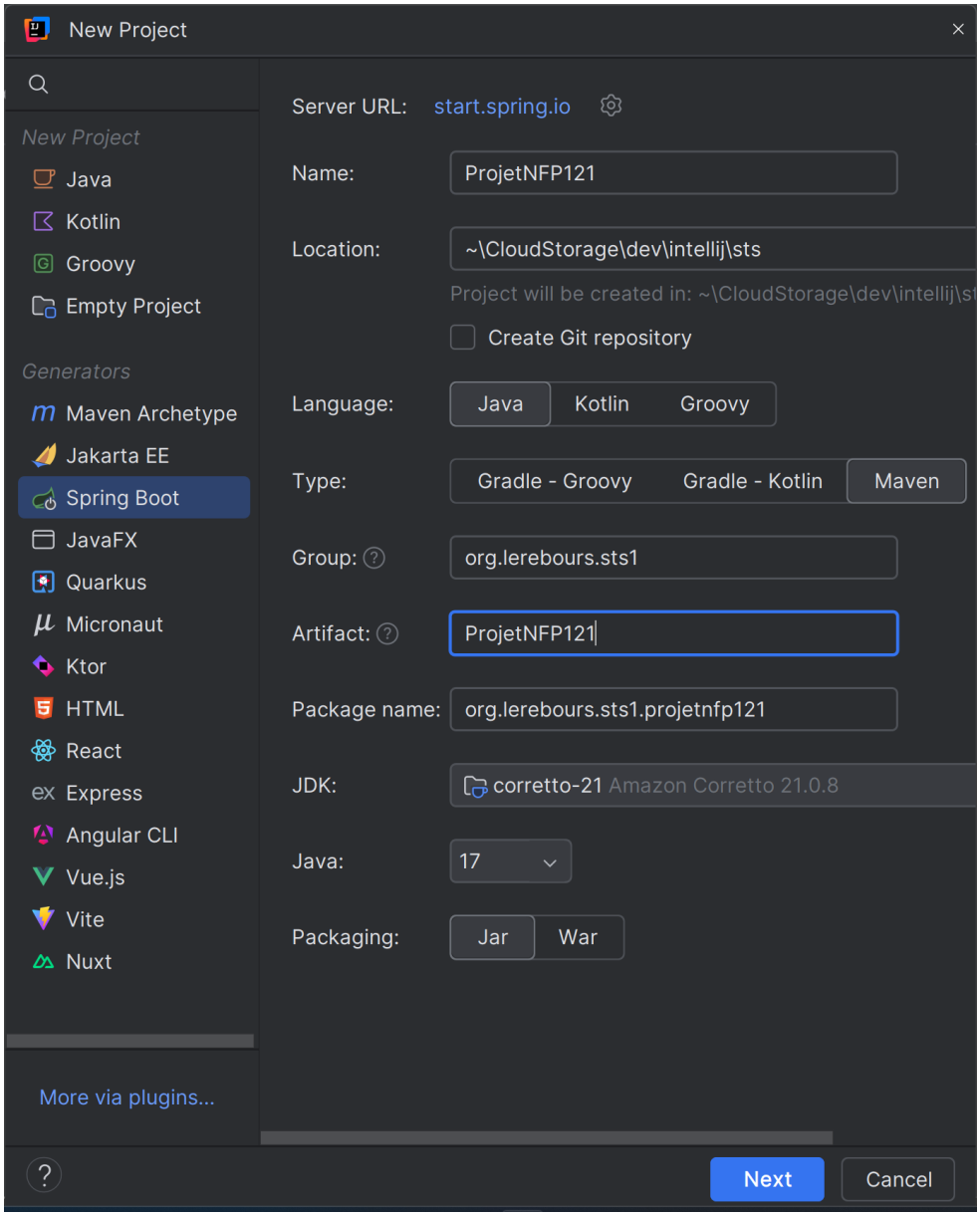


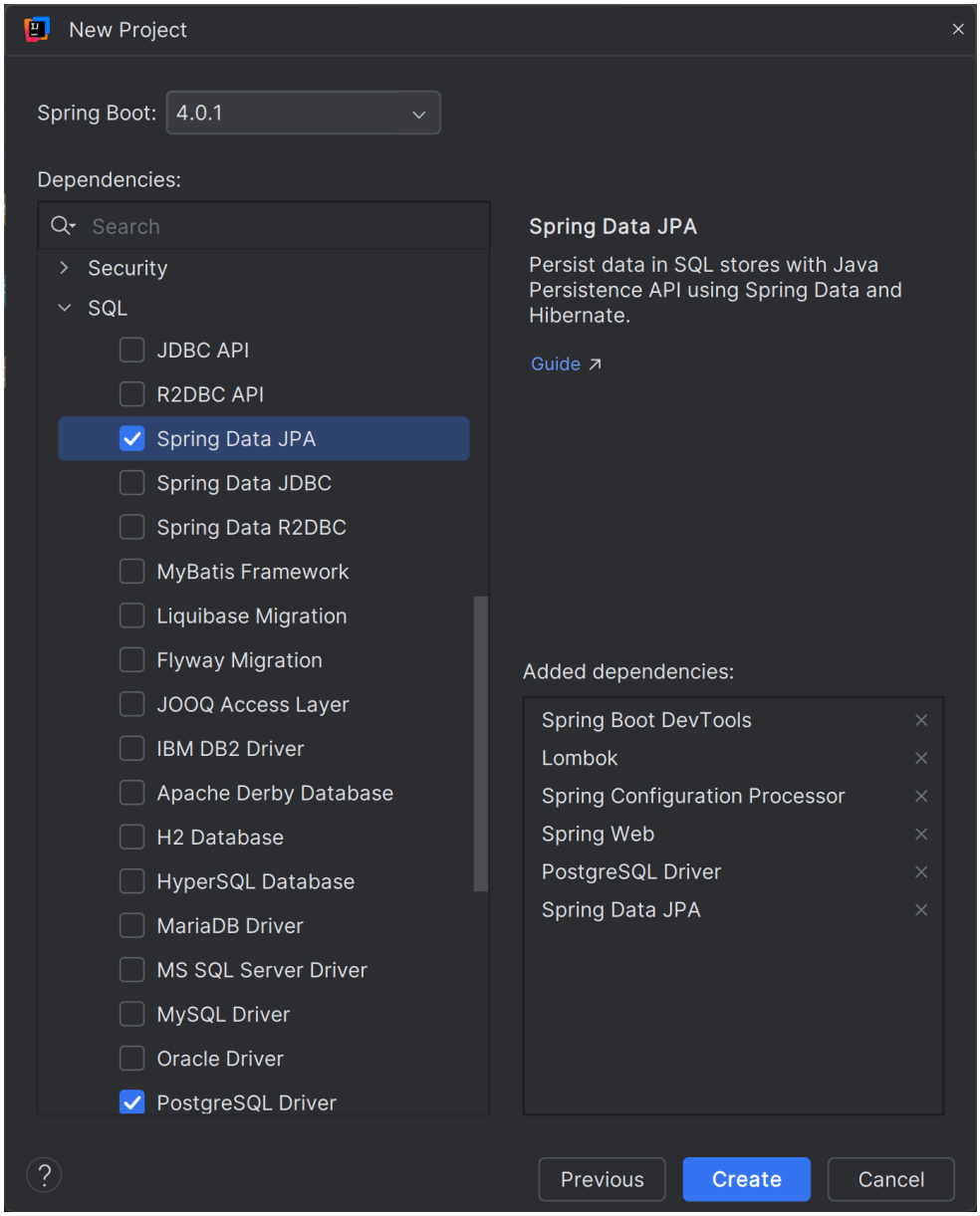

Nouveau - Serveur
Général
Connexion
Paramètres
Tunnel SSH
Avancé
Post Connection SQL
Tags
Nom
local
Groupe de serveurs
Servers
Arrière plan
Premier plan
Connecter maintenant ?
Partagé ?
Identifiant Partagé
Commentaires
Le Nom d'hôte ou le Service doit être précisé.
Fermer
Réinitialiser
Enregistrer


Nouveau - Serveur
Général
Connexion
Paramètres
Tunnel SSH
Avancé
Post Connection SQL
Tags
Nom d'hôte / Adresse
172.17.0.2
Port
5432
Base de données de maintenance
projetnfp121
Identifiant
admin
Authentification Kerberos ?
Mot de passe
In edit mode the password field is enabled only if Save Password is set to true.
Enregistrer le mot de passe ?
Rôle
Service
Fermer
Réinitialiser
Enregistrer

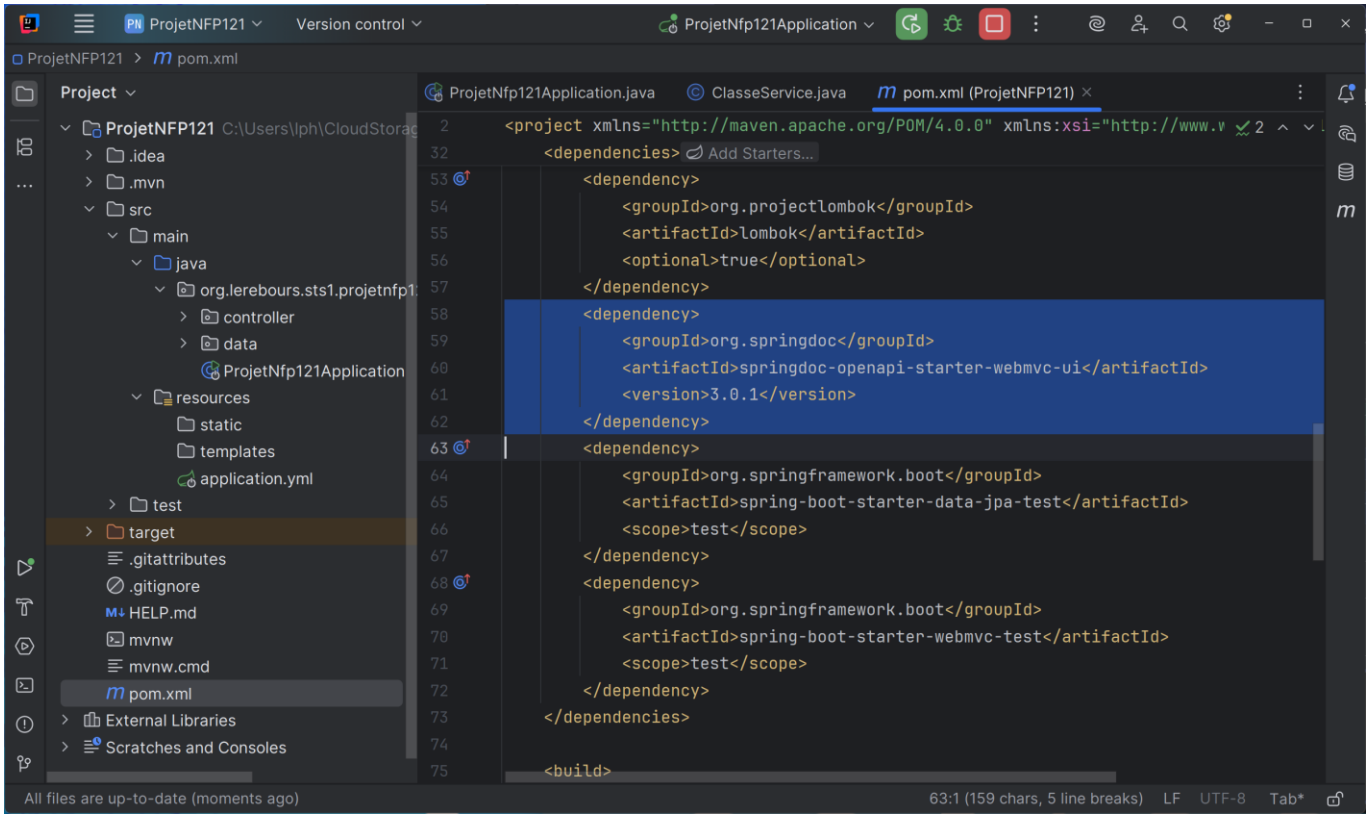
Création et configuration du projet Java SpringBoot (avec swagger)

Dans IntelliJ IDEA, créer un nouveau projet **Spring Boot**





Dans le fichier pom.xml (maven), ajouter la dépendance pour swagger :


ProjetNFP121
Version control
ProjetNfp121Application
ProjetNFP121 > pom.xml
Project
ProjetNFP121 C:\Users\lph\CloudStorag
.idea
.mvn
src
main
java
org.lerebours.sts1.projetnfp1
controller
data
ProjetNfp121Application
resources
static
templates
application.yml
test
target
.gitattributes
.gitignore
HELP.md
mvnw
mvnw.cmd
pom.xml
External Libraries
Scratches and Consoles
ProjetNfp121Application.java
ClasseService.java
pom.xml (ProjetNFP121)
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w
<dependencies> Add Starters...
<dependency>
<groupId>org.projectlombok</groupId>
<artifactId>lombok</artifactId>
<optional>true</optional>
</dependency>
<dependency>
<groupId>org.springdoc</groupId>
<artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
<version>3.0.1</version>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-data-jpa-test</artifactId>
<scope>test</scope>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-webmvc-test</artifactId>
<scope>test</scope>
</dependency>
</dependencies>
<build>
All files are up-to-date (moments ago)
63:1 (159 chars, 5 line breaks)
LF
UTF-8
Tab*

Implémenter le projet

Configuration initiale

Modifier le fichier **src/main/java/org/lerebours/sts1/projetnfp121/ProjetNfp121Application.java**

```java
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class ProjetNfp121Application {

    public static void main(String[] args) {
        System.out.println("Démarrage du Back-end !!!!!!!!!!");
        SpringApplication.run(ProjetNfp121Application.class, args);
    }

}
```

Créer le fichier **src/main/resources/application.yml** (remplace le fichier **application.properties**)

```yaml
spring:
  application:
    name: ProjetNFP121

  datasource:
    url: jdbc:postgresql://127.0.1.2:5432/projetnfp121
    username: admin
    password: admin
    driver-class-name: org.postgresql.Driver

  jpa:
    hibernate:
      ddl-auto: validate    # dev: update / create-drop, prod: validate/none
    properties:
      hibernate:
        format_sql: true
    open-in-view: false

logging:
  level:
    org.hibernate.SQL: debug
    org.hibernate.orm.jdbc.bind: trace    # Boot 3 / Hibernate 6 (paramètres SQL)
```

Implémentation des @Entity (objet Java ←→ DB)

Créer le package **data** et créer dedans la classe **Classe**

```java
package org.lerebours.sts1.projetnfp121.data;

import jakarta.persistence.*;
import lombok.Data;

@Data
@Entity
public class Classe {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(nullable = false)
    private String denomination;
}
```

Dans le package **data**, créer la classe **Etudiant**

```
package org.lerebours.sts1.projetnfp121.data;

import jakarta.persistence.*;
import lombok.Data;

@Data
@Entity
public class Etudiant {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(nullable = false)
    private String nom;

    @Column(nullable = false)
    private String prenom;

//    private String photo;

    @ManyToOne(fetch = FetchType. EAGER
)  // ne charge la classe que si elle est utilisée
    //@JoinColumn(name = "classe_id")     // facultatif si le champ de la clé étrangère
se nomme <champ>_id
    private Classe classe;
}
```

Implémentation des Repository (gestion des @Entity)

Dans le package **data**, créer l'interface **ClasseRepository**

```
package org.lerebours.sts1.projetnfp121.data;

import org.springframework.data.jpa.repository.JpaRepository;

public interface ClasseRepository extends JpaRepository<Classe, Long> {
}
```

Dans le package **data**, créer l'interface **EtudiantRepository**

```
package org.lerebours.sts1.projetnfp121.data;

import org.springframework.data.jpa.repository.JpaRepository;

import java.util.List;

public interface EtudiantRepository extends JpaRepository<Etudiant, Long> {
    List<Etudiant> getEtudiantsByClasse_Id(Long classeId);
}
```

Implémentation des @RestController (API Rest)

Créer le package **controller** et créer dedans la classe **ClasseService**

```
package org.lerebours.sts1.projetnfp121.controller;

import org.lerebours.sts1.projetnfp121.data.Classe;
import org.lerebours.sts1.projetnfp121.data.ClasseRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/classes")
class ClasseService {

    @Autowired private ClasseRepository classeRepository;

    @GetMapping("/")
    List<Classe> getAllClasses() {
        return classeRepository.findAll();
    }

    @GetMapping("/{id}")
    Classe getClasseById(@PathVariable Long id) {
        return classeRepository.findById(id).orElse(null);
    }

    @PostMapping("/add" )
    void addClasse(@RequestBody Classe classe) {
        classeRepository.save(classe);
    }

}
```

Dans le package **controller,** créer la classe **EtudiantService**

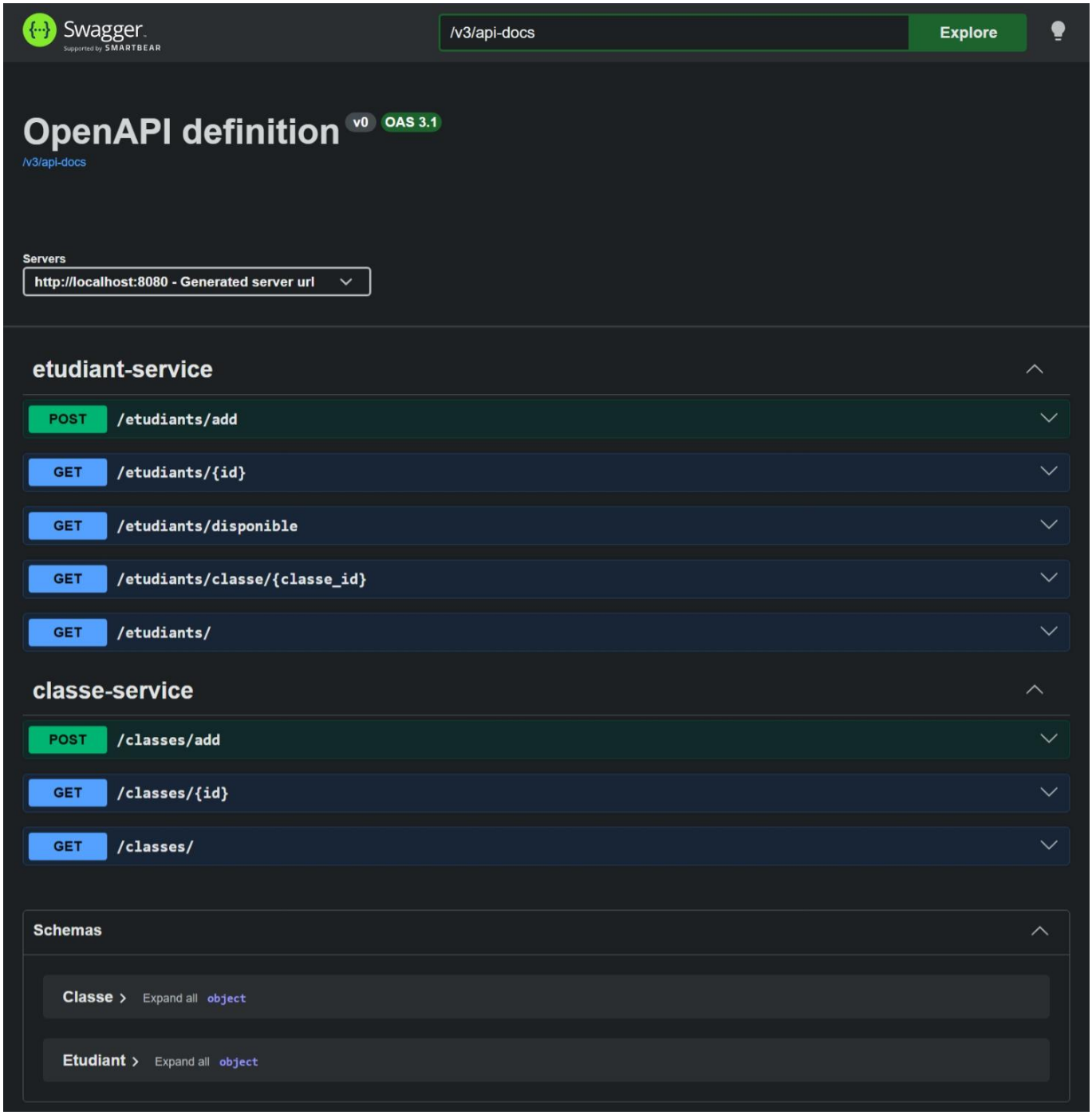
```
package org.lerebours.sts1.projetnfp121.controller;

import org.lerebours.sts1.projetnfp121.data.Etudiant;
import org.lerebours.sts1.projetnfp121.data.EtudiantRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
import java.util.List;

@RestController
@RequestMapping("/etudiants")
class EtudiantService {

    @Autowired private EtudiantRepository etudiantRepository;

    @GetMapping("/")
    List<Etudiant> getAllEtudiants() {
        return etudiantRepository.findAll();
    }

    @GetMapping("/{id}")
    Etudiant getEtudiantById(@PathVariable Long id) {
        return etudiantRepository.findById(id).orElse(null);
    }

    @GetMapping("/classe/{classe_id}")
    List<Etudiant> getEtudiantsByClasse(@PathVariable Long classe_id) {
        return etudiantRepository.getEtudiantsByClasse_Id(classe_id);
```

```
    }

    @GetMapping("/disponible")
    List<Etudiant> getEtudiantsDisponible() {
        return getEtudiantsByClasse(null);
    }

    @PostMapping("/add" )
    void addEtudiant(@RequestBody Etudiant etudiant) {
        etudiantRepository.save(etudiant);
    }


}
```

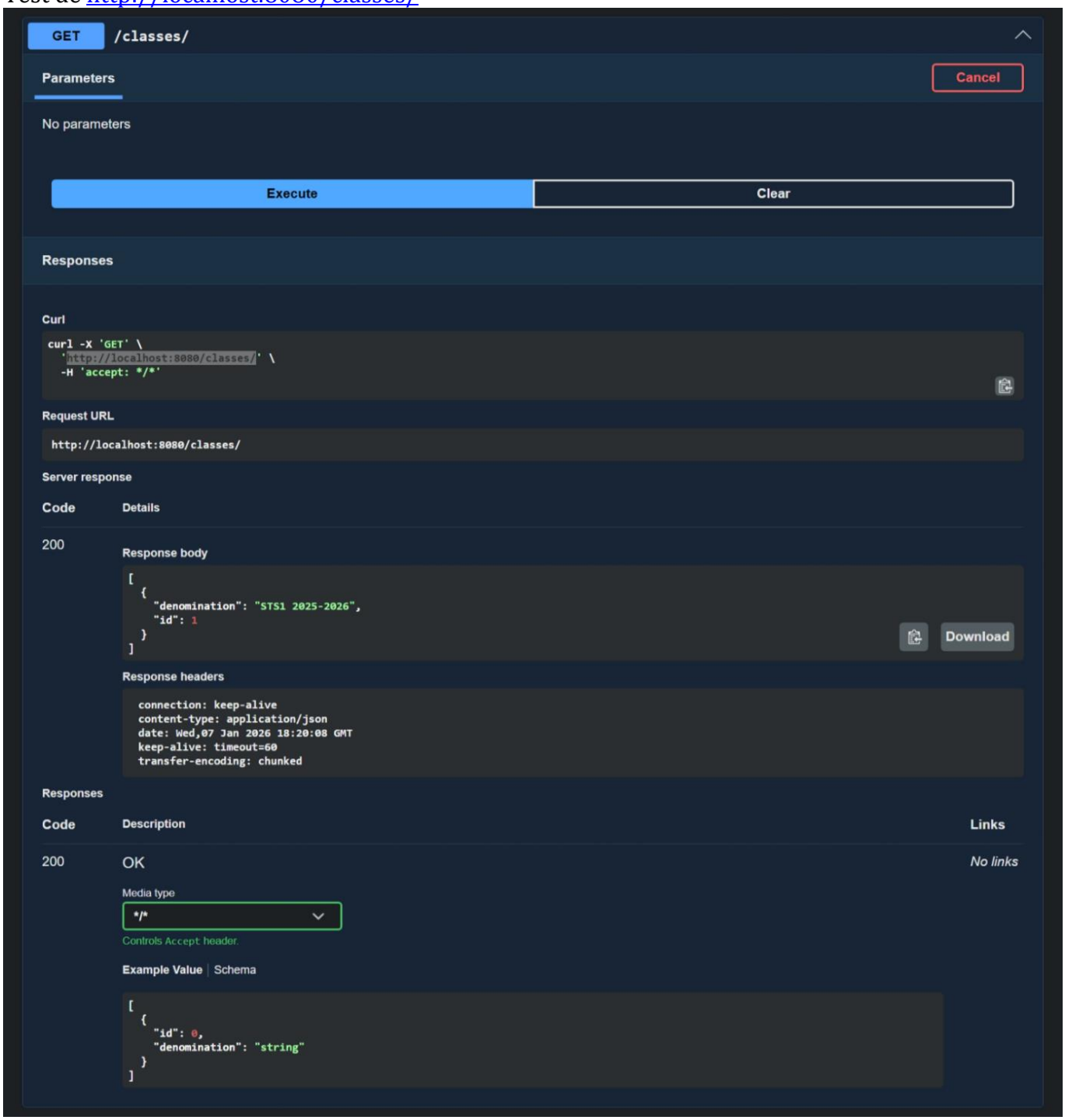
Exécuter l'application

Tester les API du Back-end via swagger UI

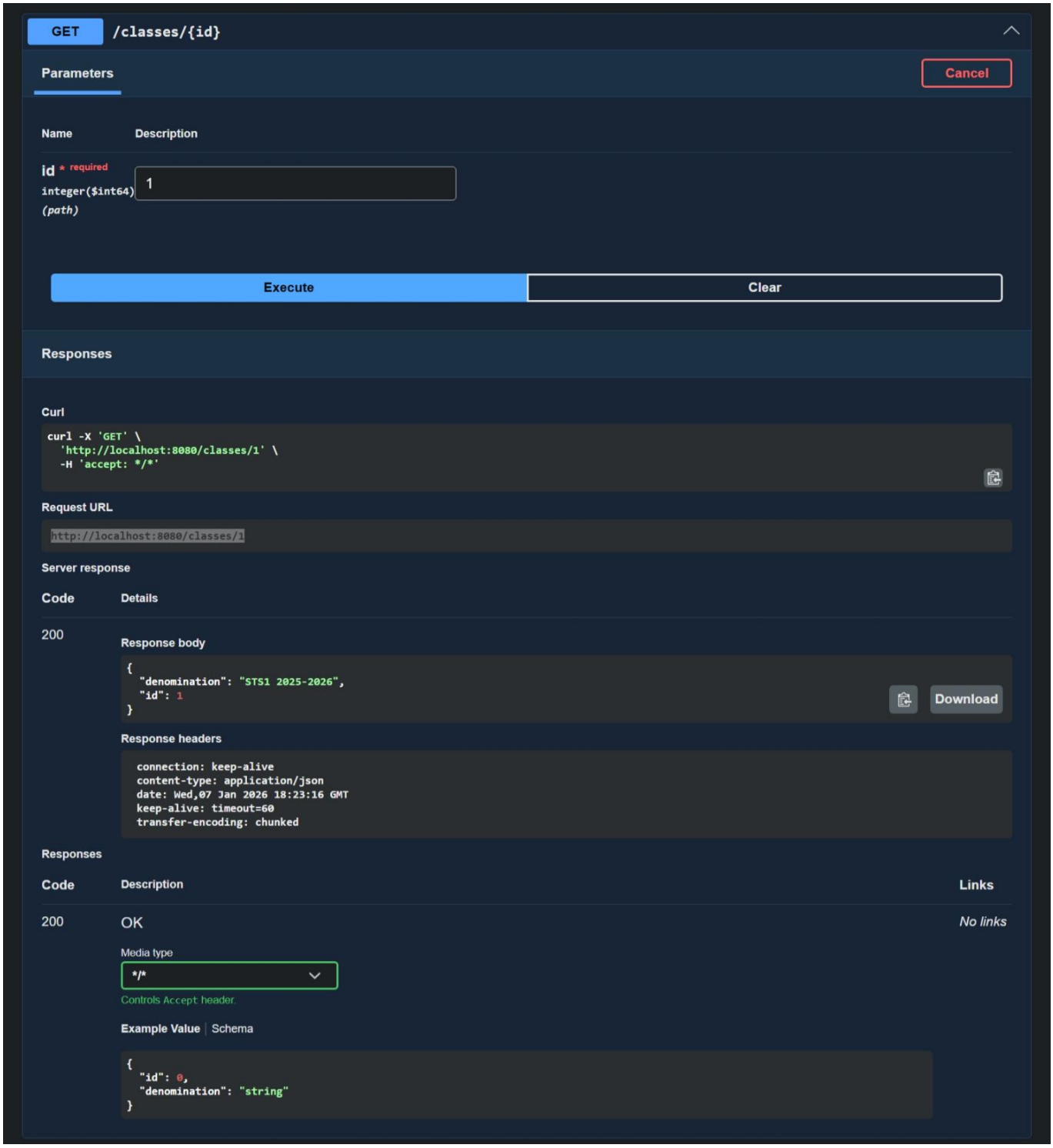
Dans votre navigateur, ouvrir l'url : http://localhost:8080/swagger-ui/index.html



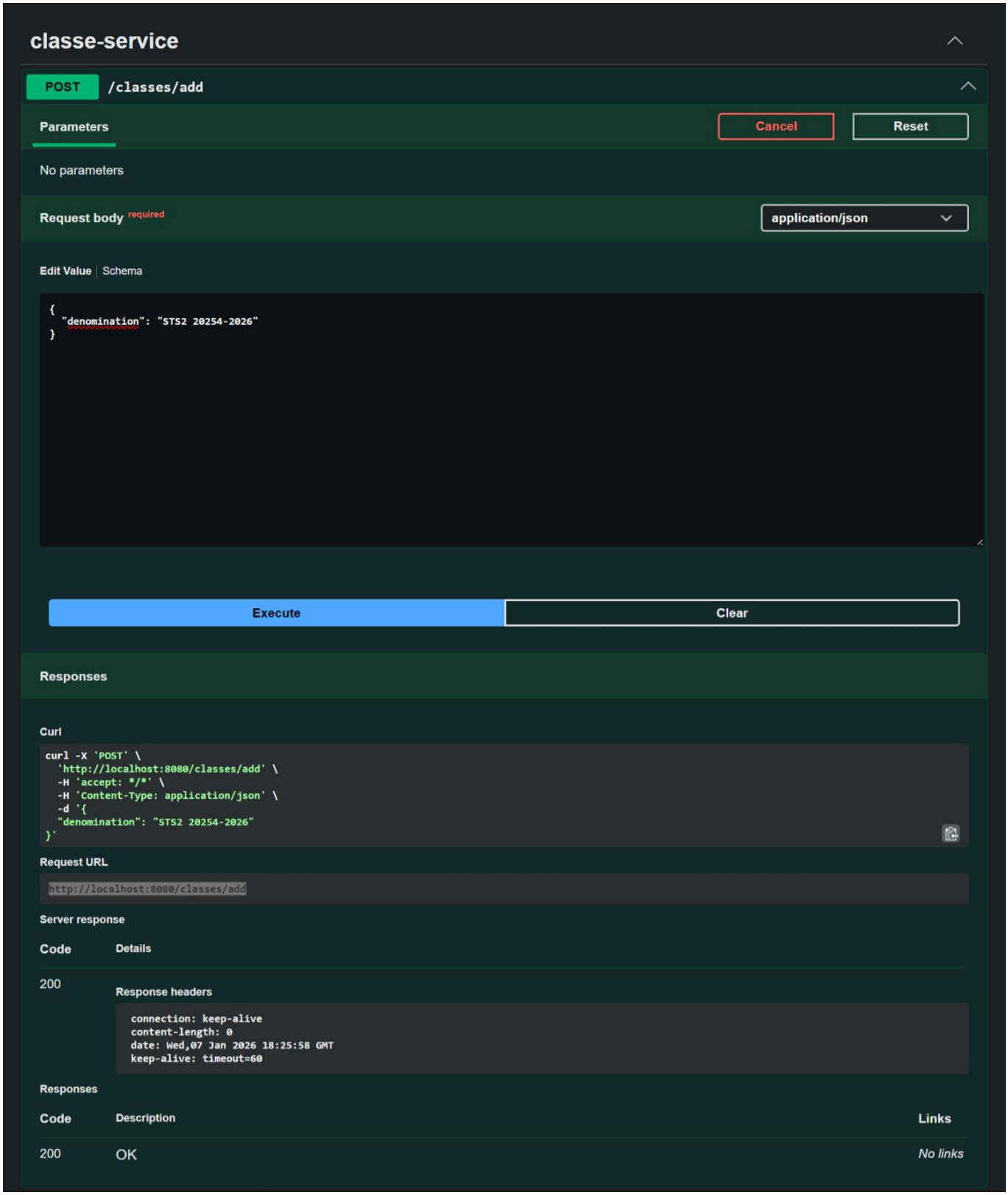
Test de http://localhost:8080/classes/



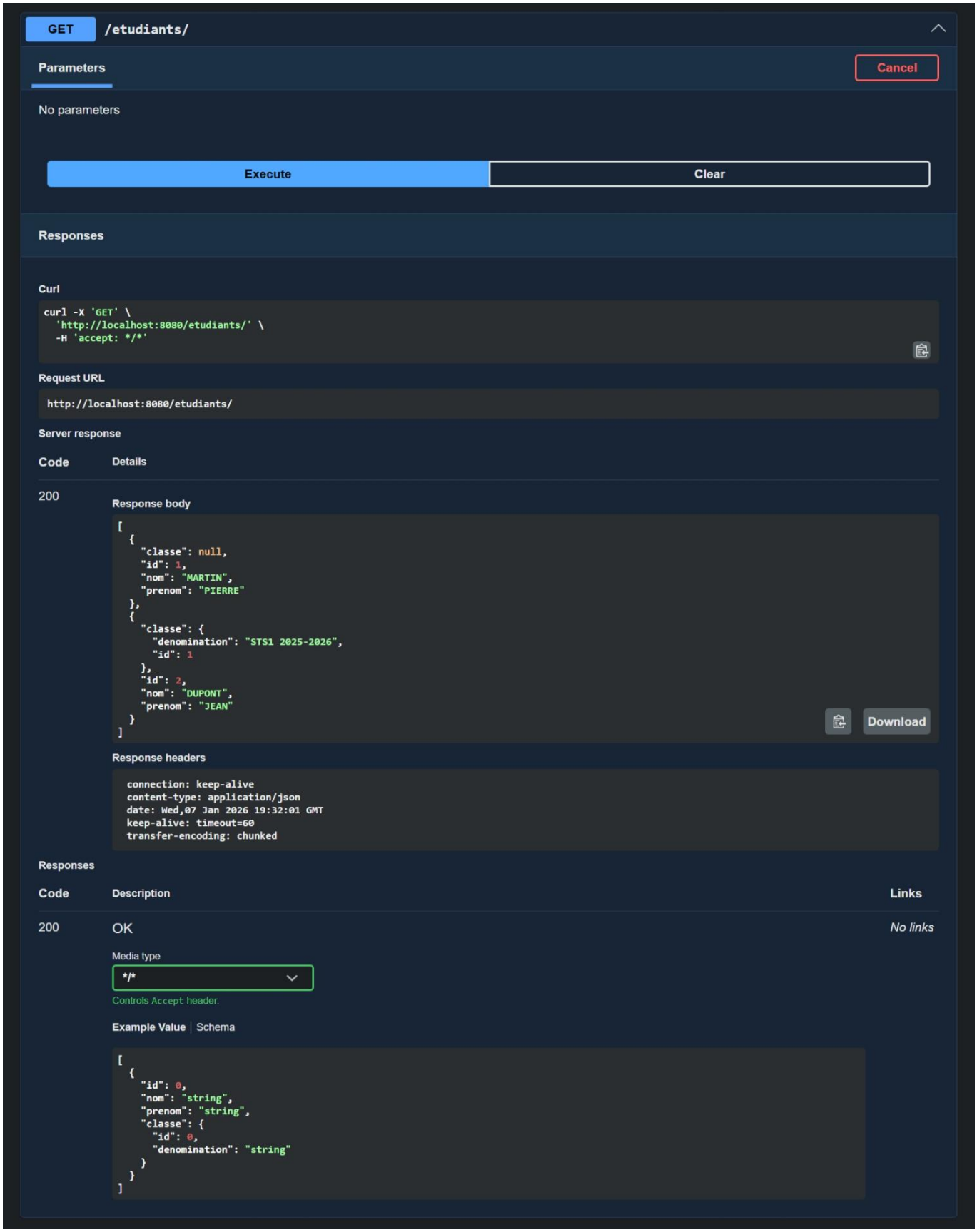
Test de http://localhost:8080/classes/1



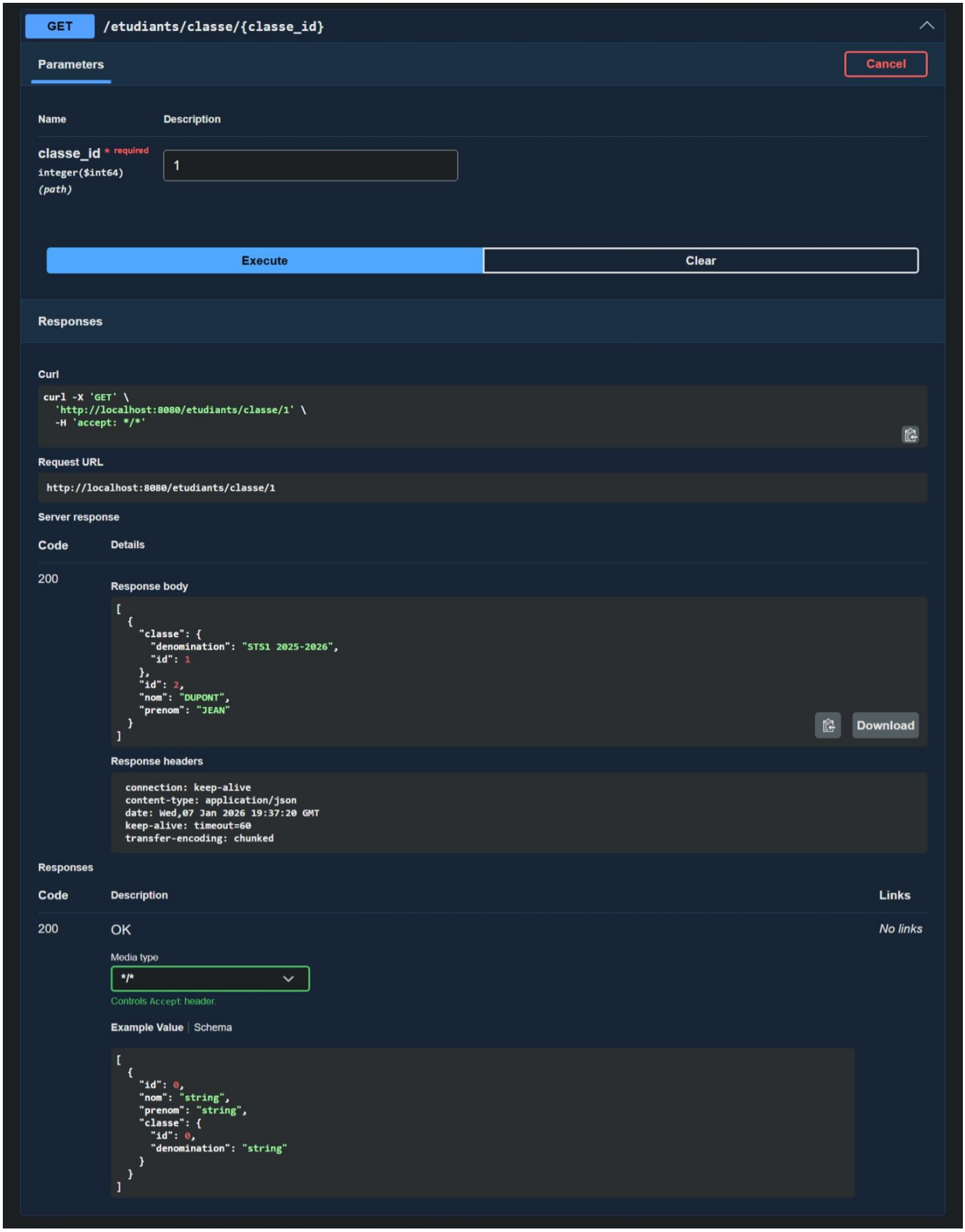
Test de http://localhost:8080/classes/add



Test de http://localhost:8080/etudiants/



Test de http://localhost:8080/etudiants/classe/1



A vous de « jouer » …