

NFP 121 - Programmation avancée

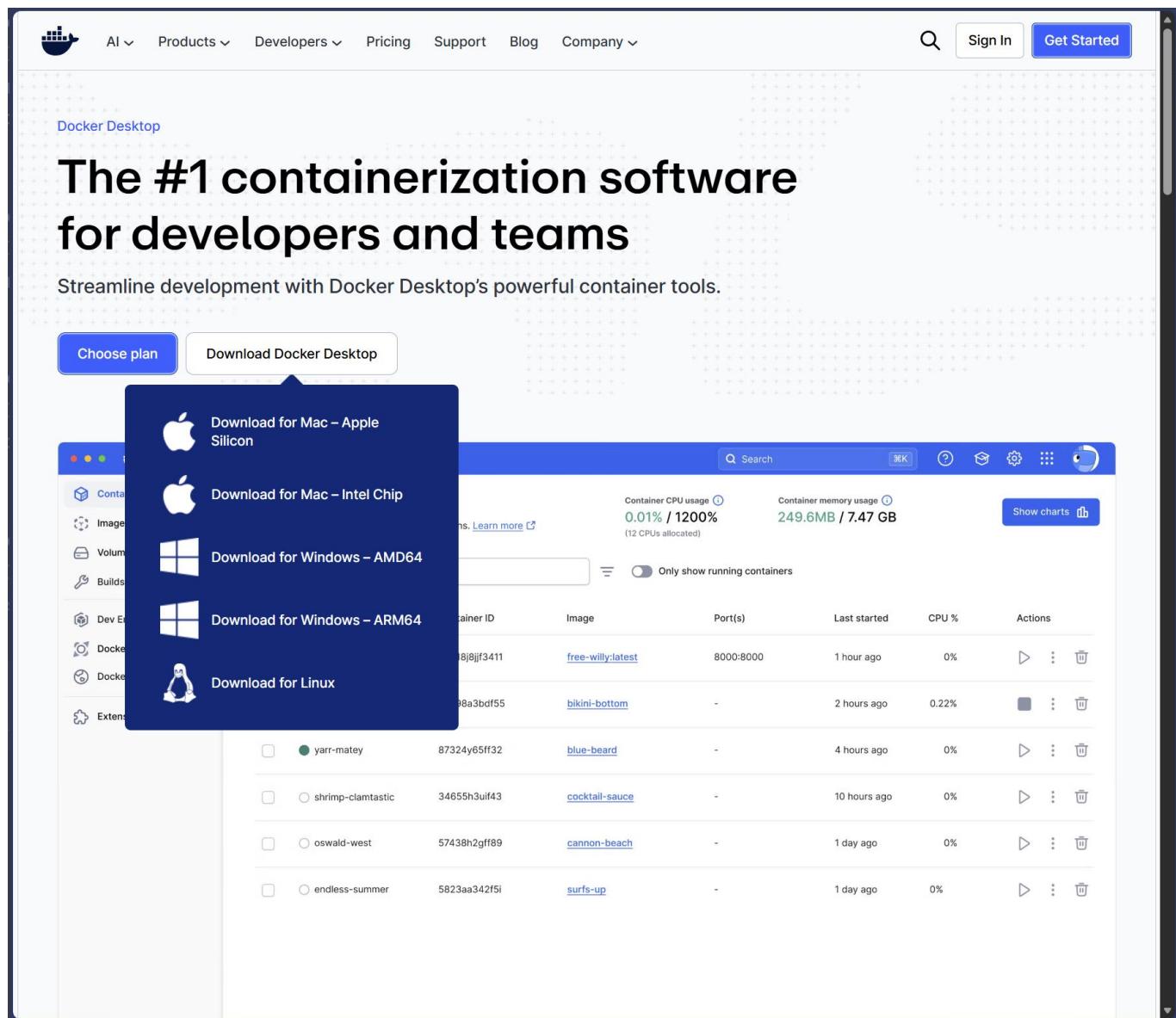
Projet

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

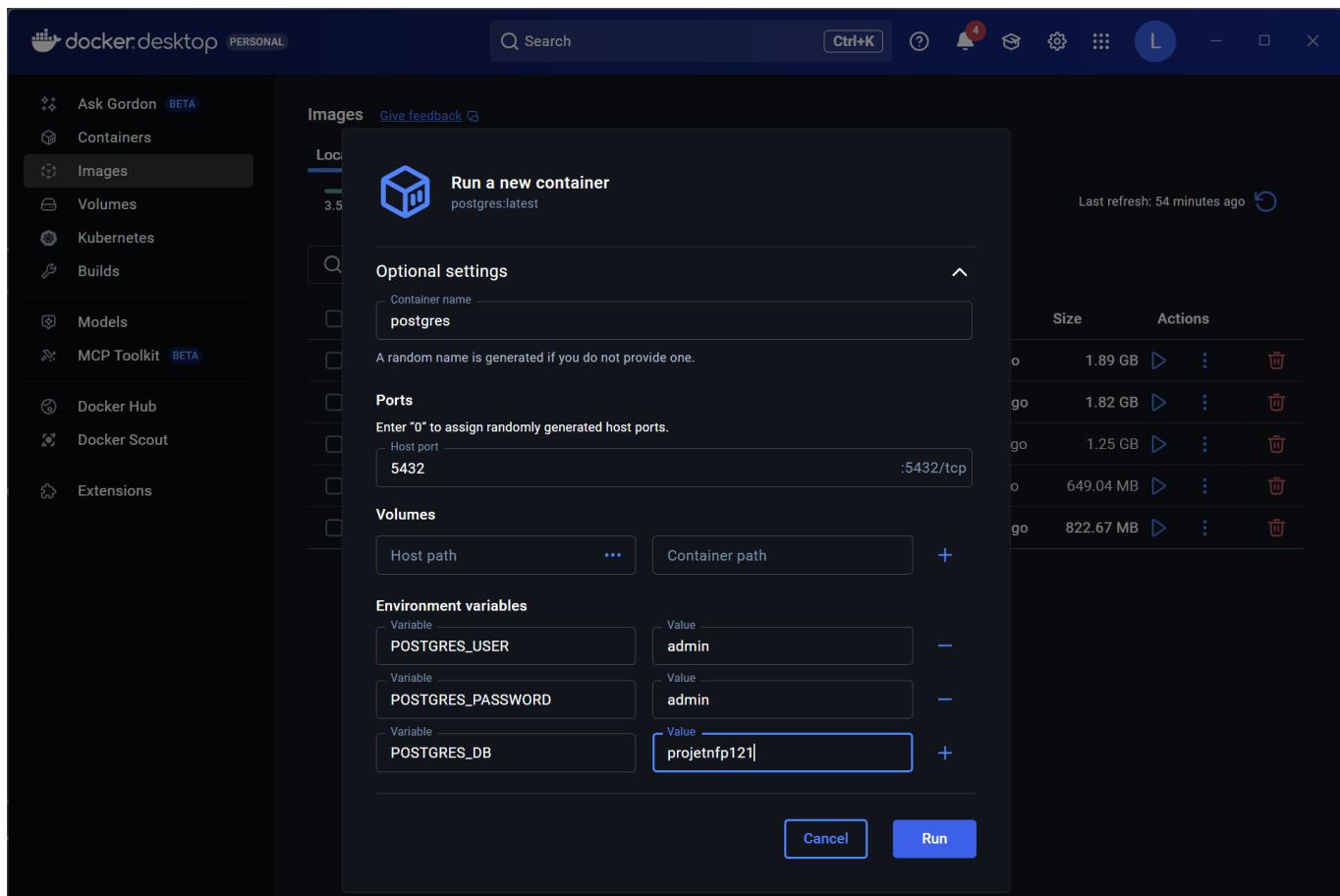
Dans Docker desktop, télécharger (Pull) l'image docker de **postgres** via l'url :

[postgres - Official Image | Docker Hub](#)

The screenshot shows the Docker Desktop interface. On the left, a sidebar lists various tools: Ask Gordon (Beta), Containers, Images, Volumes, Kubernetes, Builds, Models, MCP Toolkit (Beta), Docker Hub (selected), Docker Scout, and Extensions. The main area displays the Docker Hub page for the 'postgres' image. At the top, there's a search bar, a 'Ctrl+K' button, and a notification icon with 3 messages. Below the search bar, the 'Docker Hub / postgres' page is shown, featuring the PostgreSQL logo, the image name 'postgres', its status as a 'Docker Official Image', and a 'Tag latest' dropdown with a 'Pull' button. A brief description of PostgreSQL is provided, along with database storage statistics (10K+ stars, 1B+ pulls). The 'Overview' tab is selected, and the 'Tags' tab is also present. To the right, a 'Recent Tags' section lists several versions: 15.15-TRIXIE, 15.15, 15-TRIXIE, 15, 17.7-TRIXIE, 17.7, 17-TRIXIE, 17, 16.11-TRIXIE, and 16.11. At the bottom of the Docker Hub page, supported tags like 18.1, 18, latest, 18.1-trixie, 18-trixie, and trixie are listed. The Docker Desktop status bar at the bottom indicates 'Engine running', resource usage (RAM 1.47 GB, CPU 0.05%), and disk usage (Disk: 6.48 GB used / limit 1006.85 GB). The version 'v4.55.0' is also visible.

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

The screenshot shows the Docker Desktop interface. The sidebar on the left is identical to the previous screenshot, with 'Images' selected. The main area now displays the 'Images' page under the 'Local' tab. It shows a summary: '3.39 GB / 5.02 GB in use' for 5 images, with a note that it was last refreshed 15 minutes ago. A search bar and filter icons are available. A table lists the images: 'mongodb/mongodb-com' (latest, 90f4ca9feec2, 8 days ago, 1.89 GB), 'n8nio/n8n' (latest, 85214df20cd7, 15 days ago, 1.82 GB), 'mongo' (latest, 7f5bbdafabde, 19 days ago, 1.25 GB), 'postgres' (latest, bfe50b2b0ddd, 9 days ago, 649.04 MB), and 'dpage/pgadmin4' (latest, 50700ac17936, 30 days ago, 822.67 MB). Each row has a 'More' button and a trash icon. A status message at the bottom right says 'Status: Downloaded newer image for dpage/pgadmin4:latest'. The Docker Desktop status bar at the bottom indicates 'Engine running', resource usage (RAM 2.15 GB, CPU 0.27%), and disk usage (Disk: 7.13 GB used / limit 1006.85 GB). The version 'v4.55.0' is also visible.



PGAdmin4 sous docker

Dans Docker desktop, télécharger (Pull) l'image docker de **pgadmin4** via l'url :
[dpage/pgadmin4 - Docker Image](https://hub.docker.com/r/dpage/pgadmin4)

dpage/pgadmin4 / dpage/pgadmin4

Tag: latest **Pull** **Run**

pgAdmin 4 is a web based administration tool for the PostgreSQL database.

1.5K 100M+

Overview **Tags**

This is the official Docker distribution of pgAdmin 4.

Running

See the website for documentation on deploying different versions of the container:

- Latest release: https://www.pgadmin.org/docs/pgadmin4/latest/container_deployment.html
- Nightly snapshot build: https://www.pgadmin.org/docs/pgadmin4/development/container_deployment.html

Support

Please report any issues through the pgAdmin support channels. See <https://www.pgadmin.org/support/list/>

Recent Tags

SNAPSHOT	9.11.0	9.11	9	LATEST
2025-12-08-1	9.10.0	9.10	2025-11-10-1	9.9.0

Engine running | RAM: 1.17 GB CPU: 0.09% Disk: 6.93 GB used (limit 1006.85 GB) v4.55.0

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

The screenshot shows the Docker Desktop interface. On the left sidebar, 'Containers' is selected. In the main area, a container named 'postgres' is selected, showing its details. The 'Logs' tab is active, displaying log entries. The 'Inspect' tab is also visible, showing various configuration details. A large portion of the screen is occupied by the 'Inspect' tab's content, which includes a JSON dump of the container's configuration.

```
214     "GlobalIPv6PrefixLen": 0,
215     "IPAddress": "",
216     "IPPrefixLen": 0,
217     "IPv6Gateway": "",
218     "MacAddress": "",
219     "Networks": {
220       "bridge": {
221         "IPAMConfig": null,
222         "Links": null,
223         "Aliases": null,
224         "MacAddress": "d2:9e:c9:5e:e4:54",
225         "DriverOpts": null,
226         "GwPriority": 0,
227         "NetworkID": "59a189c47aea6fa4c157c6b6ef1c0a5ece70823edfc6feb7a055ccf12985e2e6",
228         "EndpointID": "e62cbb17fff8ee9386cd056e6daeb1d1910df043090a300d8864acdedb80960",
229         "Gateway": "172.17.0.1",
230         "IPAddress": "172.17.0.2",
231         "IPPrefixLen": 16,
232         "IPv6Gateway": ""}
```

The screenshot shows the Docker Desktop interface with the 'Images' tab selected in the sidebar. A modal dialog titled 'Run a new container' is open, showing settings for a container based on the 'dpave/pgadmin4:latest' image. The 'Optional settings' section includes a 'Container name' field set to 'pgadmin'. Under 'Ports', 'Host port 443' is mapped to 'Container port :443/tcp' and 'Host port 8000' is mapped to 'Container port :80/tcp'. The 'Volumes' section shows a mapping from 'Host path' to 'Container path'. The 'Environment variables' section contains two entries: 'PGADMIN_DEFAULT_EMAIL' with value 'lilian.prudhomme@gmail.com' and 'PGADMIN_DEFAULT_PASSWORD' with value 'admin'. At the bottom of the dialog are 'Cancel' and 'Run' buttons. To the right of the dialog, a list of existing containers is shown, each with a size, actions (Edit and Delete), and a refresh icon.

Size	Actions
1.89 GB	⋮ ⚡
1.82 GB	⋮ ⚡
1.25 GB	⋮ ⚡
649.04 MB	⋮ ⚡
822.67 MB	⋮ ⚡

Configurer pgadmin4 via l'url :

<http://localhost:8000/>



Ajouter le serveur postgres de docker :

A screenshot of the pgAdmin4 dashboard. The top navigation bar includes "Fichier", "Objet", "Outils", and "Aide". The user "lilian.prudhomme@gmail.com (interne)" is logged in. The left sidebar has a "Servers" section. The main content area is titled "Bienvenue" and features the pgAdmin logo and the text "Management Tools for PostgreSQL". It highlights "Riche en fonctionnalités | Donne le meilleur de PostgreSQL | Open Source" and describes pgAdmin as an Open Source administration and management tool for PostgreSQL. Below this is a "Liens rapides" section with "Ajouter un nouveau serveur" and "Configurer pgAdmin" buttons. The "Pour commencer" section includes links to "Documentation PostgreSQL", "Site pgAdmin", "Planète PostgreSQL", and "Entraide communautaire".

Nouveau - Serveur

Général Connexion Paramètres Tunnel SSH Avancé Post Connection SQL Tags

Nom	local
Groupe de serveurs	Servers
Arrière plan	X
Premier plan	X
Connecter maintenant ?	<input checked="" type="checkbox"/>
Partagé ?	<input type="checkbox"/>
Identifiant Partagé	
Commentaires	

! Le Nom d'hôte ou le Service doit être précisé.

i **?** **×** Fermer **⟳ Réinitialiser** **💾 Enregistrer**

Nouveau - Serveur

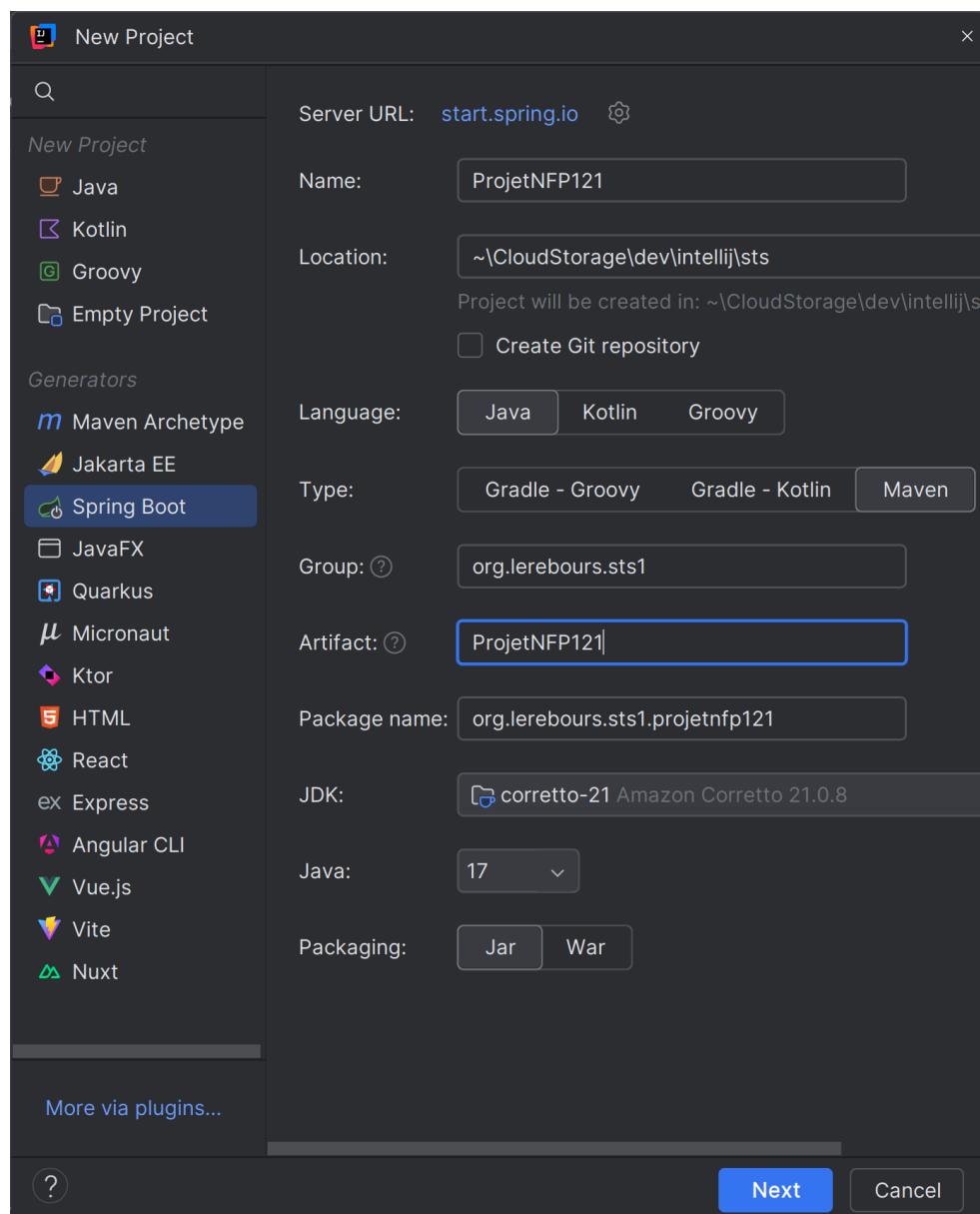
Général Connexion Paramètres Tunnel SSH Avancé Post Connection SQL Tags

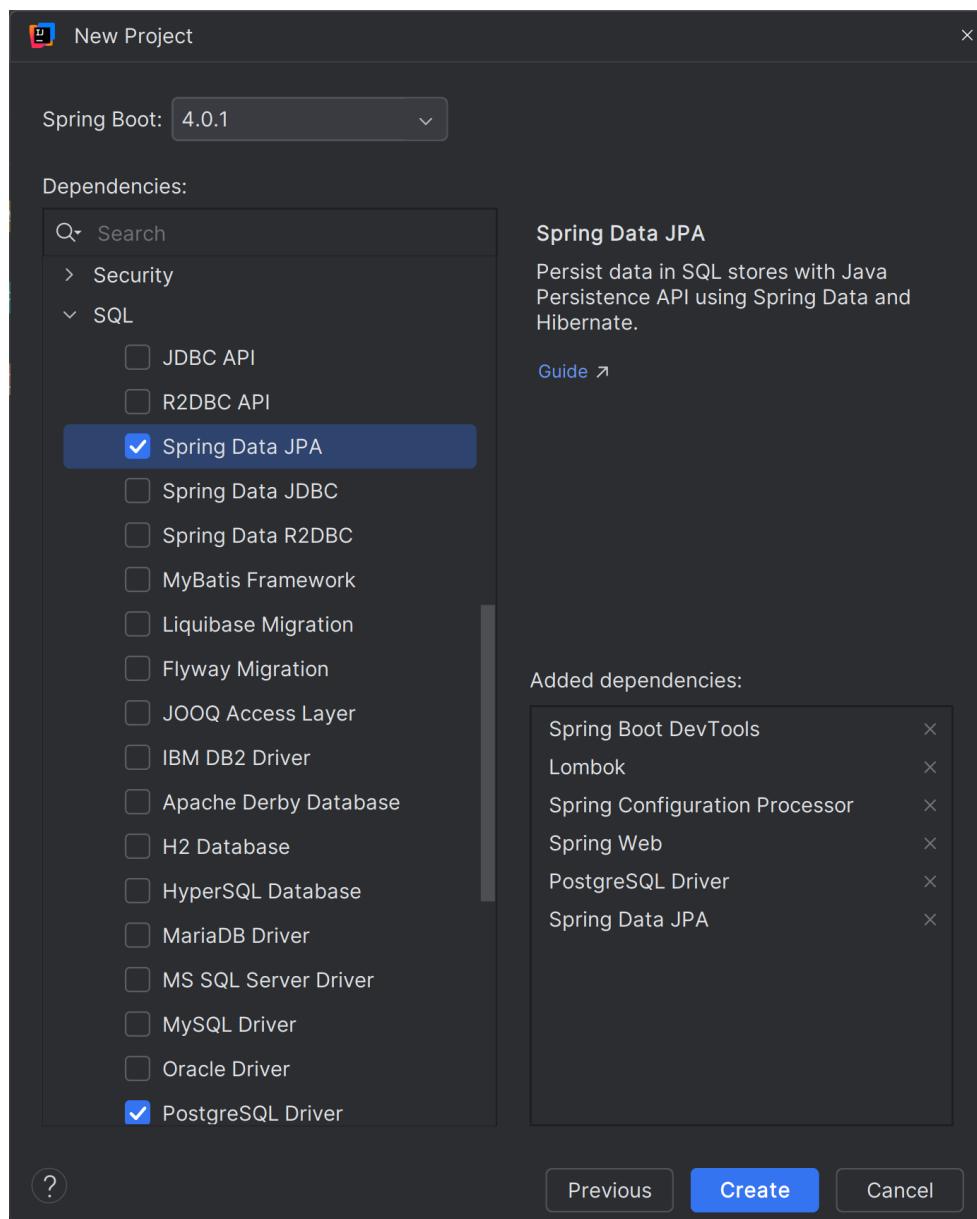
Nom d'hôte / Adresse	172.17.0.2
Port	5432
Base de données de maintenance	projetcnfp121
Identifiant	admin
Authentification Kerberos ?	<input checked="" type="checkbox"/>
Mot de passe
In edit mode the password field is enabled only if Save Password is set to true.	
Enregistrer le mot de passe ?	<input type="checkbox"/>
Rôle	
Service	

i **?** **×** 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 :

The screenshot shows a Java development environment with the following details:

- Project:** ProjetNfp121
- File:** pom.xml (ProjetNfp121)
- Content:** Maven configuration for the project.

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.lerebours.sts1.projetnfp121</groupId>
  <artifactId>ProjetNfp121Application</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>ProjetNfp121Application</name>
  <description>Spring Boot application for managing classes and students</description>
  <packaging>jar</packaging>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.6.3</version>
    <relativePath/>
  </parent>
  <dependencies>
    <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>
    <plugins>
      <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
        <version>2.6.3</version>
        <configuration>
          <excludes>
            <exclude>org.springframework.boot:spring-boot-devtools</exclude>
          </excludes>
        </configuration>
      </plugin>
    </plugins>
  </build>

```

The project structure on the left includes .idea, .mvn, src (main, test), target, .gitattributes, .gitignore, HELP.md, mvnw, mvnw.cmd, pom.xml, External Libraries, and Scratches and Consoles.

Implémenter le projet

Configuration initiale

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

```
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**

```
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>

The screenshot shows the Swagger UI interface for a REST API. At the top, there's a navigation bar with the Swagger logo, a search bar containing '/v3/api-docs', a 'Explore' button, and a lightbulb icon. Below the header, the title 'OpenAPI definition' is displayed, along with 'v0' and 'OAS 3.1' status indicators. A 'Servers' section shows a dropdown menu set to 'http://localhost:8080 - Generated server url'. The main content area is divided into sections for different services:

- etudiant-service**: Contains five API endpoints:
 - POST /etudiants/add**
 - GET /etudiants/{id}**
 - GET /etudiants/disponible**
 - GET /etudiants/classe/{classe_id}**
 - GET /etudiants/**
- classe-service**: Contains three API endpoints:
 - POST /classes/add**
 - GET /classes/{id}**
 - GET /classes/**
- Schemas**: Shows two schema definitions:
 - Classe** > Expand all `object`
 - Etudiant** > Expand all `object`

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

The screenshot shows a REST API testing interface with the following details:

Method: GET
Path: /classes/
Parameters: No parameters
Buttons: Execute, Clear, Cancel
Responses:

Curl:

```
curl -X 'GET' \
'http://localhost:8080/classes/' \
-H 'accept: */*'
```

Request URL: http://localhost:8080/classes/

Server response:

Code	Details	Links
200	Response body [{ "denomination": "STS1 2025-2026", "id": 1 }] Response headers connection: keep-alive content-type: application/json date: Wed, 07 Jan 2026 18:20:08 GMT keep-alive: timeout=60 transfer-encoding: chunked	Copy Download

Responses:

Code	Description	Links
200	OK	No links

Media type: */*
Controls Accept header.

[Example Value](#) | [Schema](#)

```
[  
  {  
    "id": 0,  
    "denomination": "string"  
  }  
]
```

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

GET /classes/{id}

Cancel

Parameters

Name	Description
id * required integer(\$int64) (path)	1

Execute Clear

Responses

Curl

```
curl -X 'GET' \
'http://localhost:8080/classes/1' \
-H 'accept: */'
```

Request URL

```
http://localhost:8080/classes/1
```

Server response

Code Details

200 Response body

```
{
  "denomination": "STS1 2025-2026",
  "id": 1
}
```

Download

Response headers

```
connection: keep-alive
content-type: application/json
date: Wed, 07 Jan 2026 18:23:16 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

Responses

Code	Description	Links
200	OK	No links

Media type

```
*/*
```

Controls Accept header.

Example Value | Schema

```
{
  "id": 0,
  "denomination": "string"
}
```

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

classe-service

POST /classes/add

Parameters

No parameters

Request body required

application/json

Edit Value | Schema

```
{ "denomination": "STS2 20254-2026" }
```

Execute Clear

Responses

Curl

```
curl -X 'POST' \
'http://localhost:8080/classes/add' \
-H 'accept: */*' \
-H 'Content-Type: application/json' \
-d '{
  "denomination": "STS2 20254-2026"
}'
```

Request URL

http://localhost:8080/classes/add

Server response

Code Details

200 Response headers

```
connection: keep-alive
content-length: 0
date: Wed,07 Jan 2026 18:25:58 GMT
keep-alive: timeout=60
```

Responses

Code Description Links

200 OK No links

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

GET /etudiants/

Parameters

No parameters

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:8080/etudiants/' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:8080/etudiants/
```

Server response

Code Details

200 Response body

```
[  
  {  
    "classe": null,  
    "id": 1,  
    "nom": "MARTIN",  
    "prenom": "PIERRE"  
  },  
  {  
    "classe": {  
      "denomination": "STS1 2025-2026",  
      "id": 1  
    },  
    "id": 2,  
    "nom": "DUPONT",  
    "prenom": "JEAN"  
  }]
```

Response headers

```
connection: keep-alive  
content-type: application/json  
date: Wed,07 Jan 2026 19:32:01 GMT  
keep-alive: timeout=60  
transfer-encoding: chunked
```

Responses

Code	Description	Links
200	OK	No links

Media type

/

Controls Accept header.

Example Value | Schema

```
[  
  {  
    "id": 0,  
    "nom": "string",  
    "prenom": "string",  
    "classe": {  
      "id": 0,  
      "denomination": "string"  
    }  
  }]
```

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

The screenshot shows a REST API testing interface with the following details:

- Method:** GET
- Endpoint:** /etudiants/classe/{classe_id}
- Parameters:** classe_id (required, integer, path) = 1
- Buttons:** Execute, Clear
- Responses:**
 - Curl:** curl -X 'GET' '\http://localhost:8080/etudiants/classe/1' \-H 'accept: */*'
 - Request URL:** http://localhost:8080/etudiants/classe/1
 - Server response:**
 - Code:** 200
 - Details:** Response body
 - Content:** [{ "classe": { "denomination": "STS1 2025-2026", "id": 1 }, "id": 2, "nom": "DUPONT", "prenom": "JEAN" }]
 - Actions:** Copy, Download
 - Response headers:**
 - connection: keep-alive
 - content-type: application/json
 - date: Wed, 07 Jan 2026 19:37:20 GMT
 - keep-alive: timeout=60
 - transfer-encoding: chunked
- Responses:**
 - Code:** 200
 - Description:** OK
 - Links:** No links
 - Media type:** */*
 - Controls:** Accept header.
 - Example Value | Schema:**
 - Content:** [{ "id": 0, "nom": "string", "prenom": "string", "classe": { "id": 0, "denomination": "string" } }]

A vous de « jouer » ...