

TP3 – BDA

**TP n° 3 : Premier micro-service, Hibernate, Base de données in memory
(H2)**

Baba SOW

ING – INFO2

2024/2025

I) Création d'un premier microservice :

1. Génération du projet Spring Boot - Maven avec Spring Initializr

The screenshot shows the Spring Initializr web interface at <https://start.spring.io/index.html>. The form is configured with the following settings:

- Project:** ☐ Gradle - Groovy, ☐ Gradle - Kotlin, ☒ Maven
- Language:** ☒ Java, ☐ Kotlin, ☐ Groovy
- Spring Boot:** ☐ 3.5.0 (SNAPSHOT), ☐ 3.5.0 (RC1), ☒ 3.4.6 (SNAPSHOT), ☐ 3.4.5, ☐ 3.3.12 (SNAPSHOT), ☐ 3.3.11
- Project Metadata:**
 - Group:
 - Artifact:
 - Name:
 - Description:
 - Package name:
- Packaging:** ☒ Jar, ☐ War
- Java:** ☐ 24, ☐ 21, ☒ 17

On the right, the **Dependencies** section shows **Spring Web** selected with the **WEB** checkbox. Below it, a description reads: "Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container."

At the bottom, there are three buttons: **GENERATE** (with keyboard shortcut CTRL + G), **EXPLORE** (with keyboard shortcut CTRL + SPACE), and a **...** button.

- jar (pour ne pas à le déployer sur un serveur d'application)
- java 17 (dernière version stable disponible)
- Spring Web dans les dépendances : RESTful pour développer des API Rest

2. Importation dans IntelliJ et premier Run pour tester

The screenshot shows the IntelliJ IDEA Community Edition interface. The **Project** view on the left shows the project structure:

- spring1 (root)
- idea
- mvn
- src
 - main
 - java
 - com.example.spring1
 - Spring1Application
- resources
- test
- target
- .gitattributes
- .gitignore
- HELP.md
- mvnw
- mvnw.cmd

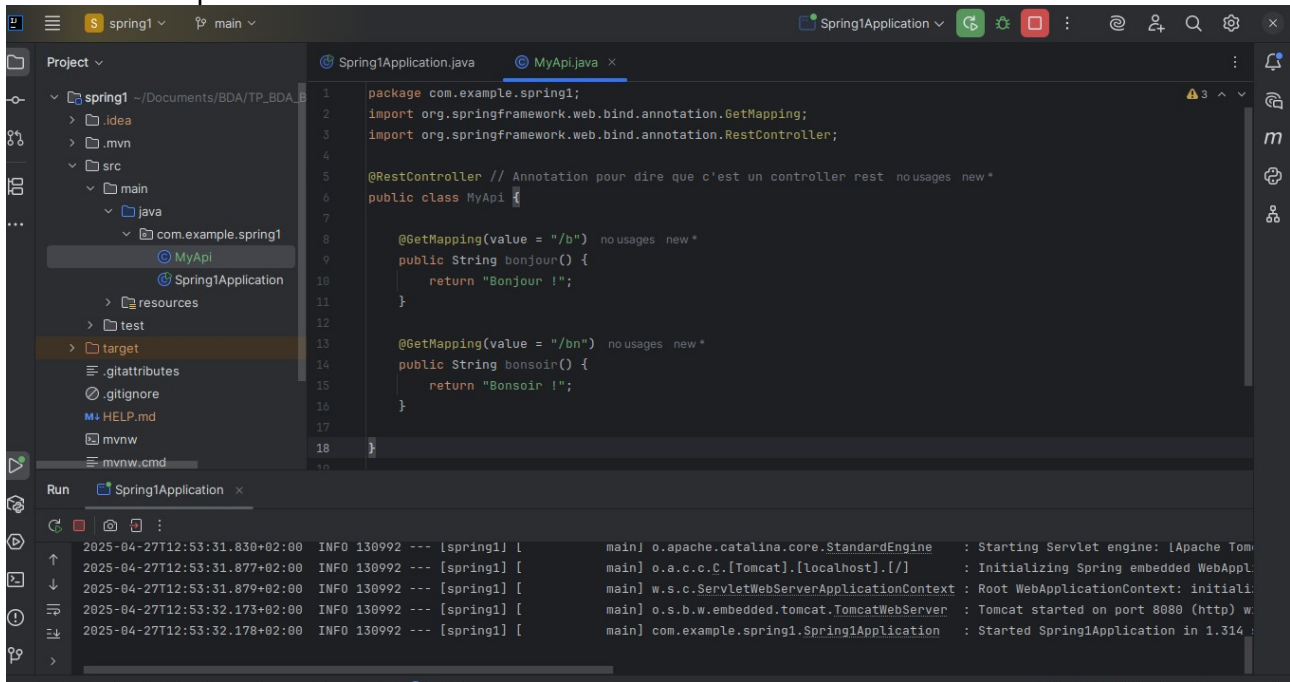
The **Spring1Application.java** file is open in the editor, showing the following code:

```
1 package com.example.spring1;
2
3 > import ...
4
5 @SpringBootApplication & 2baSW
6 public class Spring1Application {
7
8     public static void main(String[] args) { SpringApplication.run(Spring1Application.class, arg
9
10 }
11
12
13
14
```

The **Run** view at the bottom shows the output of the application:

```
2025-04-27T12:26:51.266+02:00 INFO 129332 --- [spring1] [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2025-04-27T12:26:51.266+02:00 INFO 129332 --- [spring1] [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.60]
2025-04-27T12:26:51.291+02:00 INFO 129332 --- [spring1] [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2025-04-27T12:26:51.292+02:00 INFO 129332 --- [spring1] [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed
2025-04-27T12:26:51.517+02:00 INFO 129332 --- [spring1] [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with context path [/]
2025-04-27T12:26:51.532+02:00 INFO 129332 --- [spring1] [main] com.example.spring1.Spring1Application : Started Spring1Application in 1.284 seconds
```

3. Création du premier controller



The screenshot shows an IDE with a project named 'spring1'. The file explorer on the left shows the project structure: 'src/main/java/com.example.spring1'. The main editor displays the code for 'MyApi.java'. The code defines a REST controller with two endpoints: '/b' returning 'Bonjour !' and '/bn' returning 'Bonsoir !'. The bottom panel shows the console output of the application, indicating it started successfully on port 8080.

```
package com.example.spring1;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

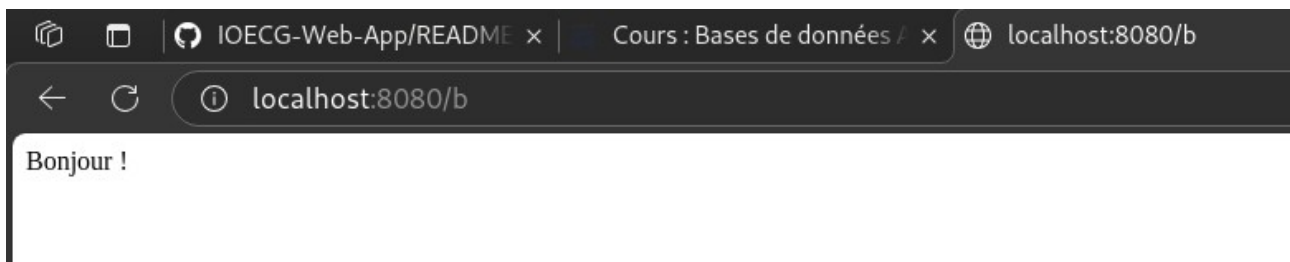
@RestController // Annotation pour dire que c'est un controller rest
public class MyApi {

    @GetMapping(value = "/b")
    public String bonjour() {
        return "Bonjour !";
    }

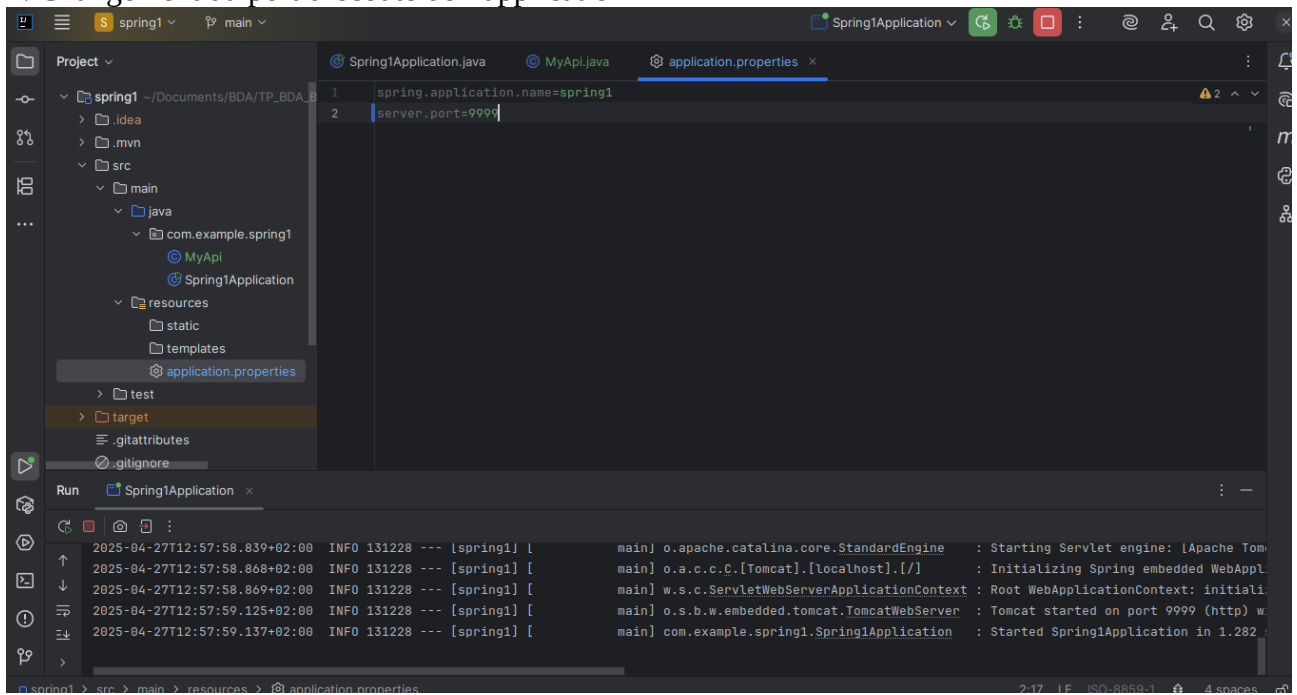
    @GetMapping(value = "/bn")
    public String bonsoir() {
        return "Bonsoir !";
    }
}
```

Run Spring1Application

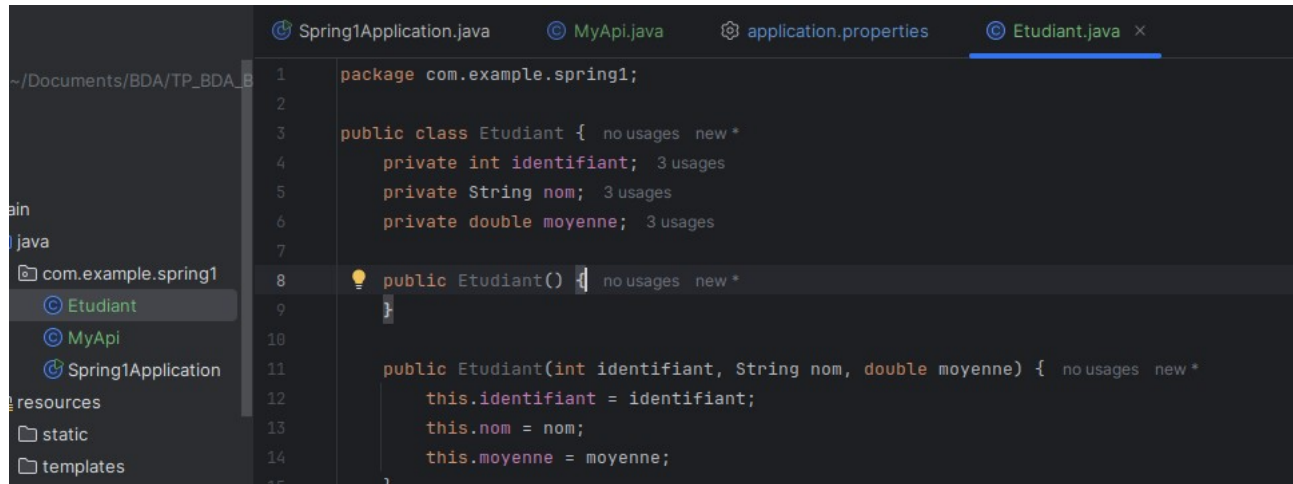
```
2025-04-27T12:53:31.830+02:00 INFO 130992 --- [spring1] [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.53]
2025-04-27T12:53:31.877+02:00 INFO 130992 --- [spring1] [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2025-04-27T12:53:31.879+02:00 INFO 130992 --- [spring1] [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed
2025-04-27T12:53:32.173+02:00 INFO 130992 --- [spring1] [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (http://localhost:8080)
2025-04-27T12:53:32.178+02:00 INFO 130992 --- [spring1] [main] com.example.spring1.Spring1Application : Started Spring1Application in 1.314 seconds
```



4. Changement du port d'écoute de l'application



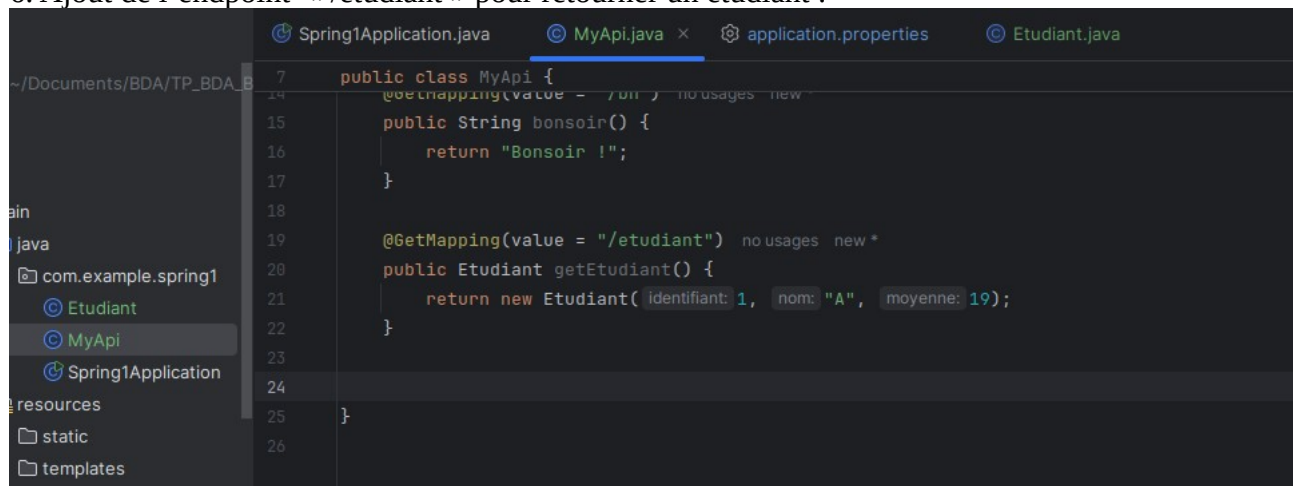
5. Création d'une classe « Etudiant »



The screenshot shows an IDE with the file explorer on the left displaying the project structure: `com.example.spring1` containing `Etudiant`, `MyApi`, and `Spring1Application`. The main editor shows the `Etudiant.java` file with the following code:

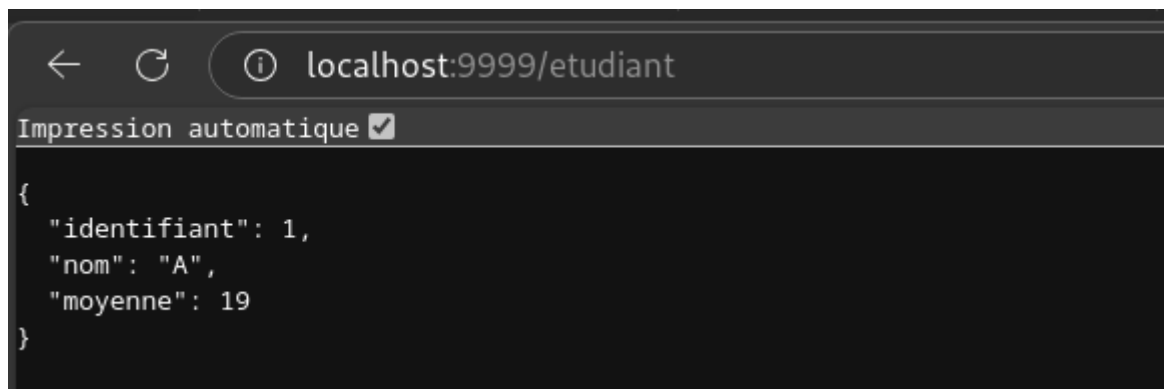
```
1 package com.example.spring1;
2
3 public class Etudiant {
4     private int identifiant;
5     private String nom;
6     private double moyenne;
7
8     public Etudiant() {
9     }
10
11     public Etudiant(int identifiant, String nom, double moyenne) {
12         this.identifiant = identifiant;
13         this.nom = nom;
14         this.moyenne = moyenne;
15     }
16 }
```

6. Ajout de l'endpoint « /etudiant » pour retourner un étudiant :



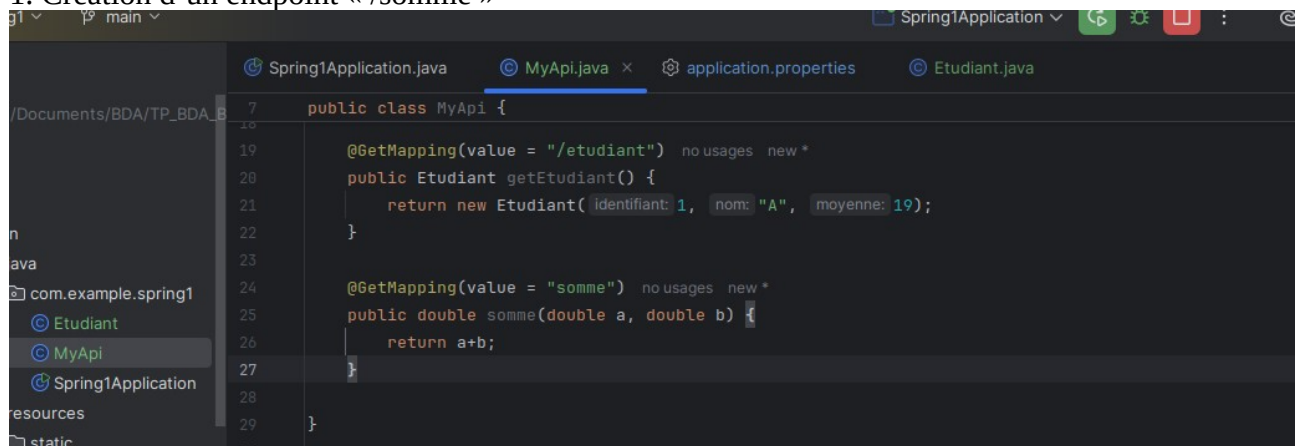
The screenshot shows the `MyApi.java` file in the IDE. The code defines a `MyApi` class with two methods: `bonsoir()` and `getEtudiant()`. The `getEtudiant()` method is annotated with `@GetMapping("/etudiant")` and returns a new `Etudiant` object with specific values.

```
7 public class MyApi {
8     @GetMapping(value = "/bonsoir")
9     public String bonsoir() {
10         return "Bonsoir !";
11     }
12
13     @GetMapping(value = "/etudiant")
14     public Etudiant getEtudiant() {
15         return new Etudiant(1, "A", 19);
16     }
17 }
```



II) Tester les méthodes d'un microservices

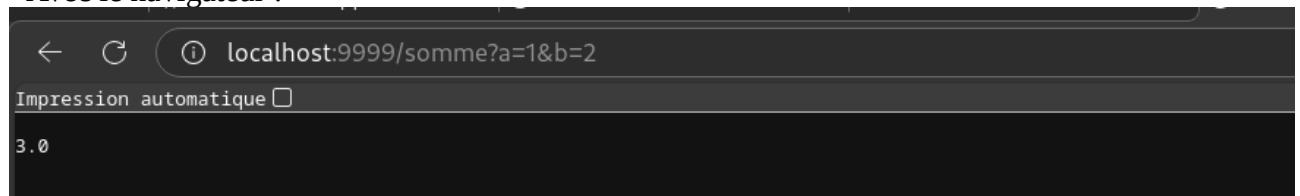
1. Création d'un endpoint « /somme »



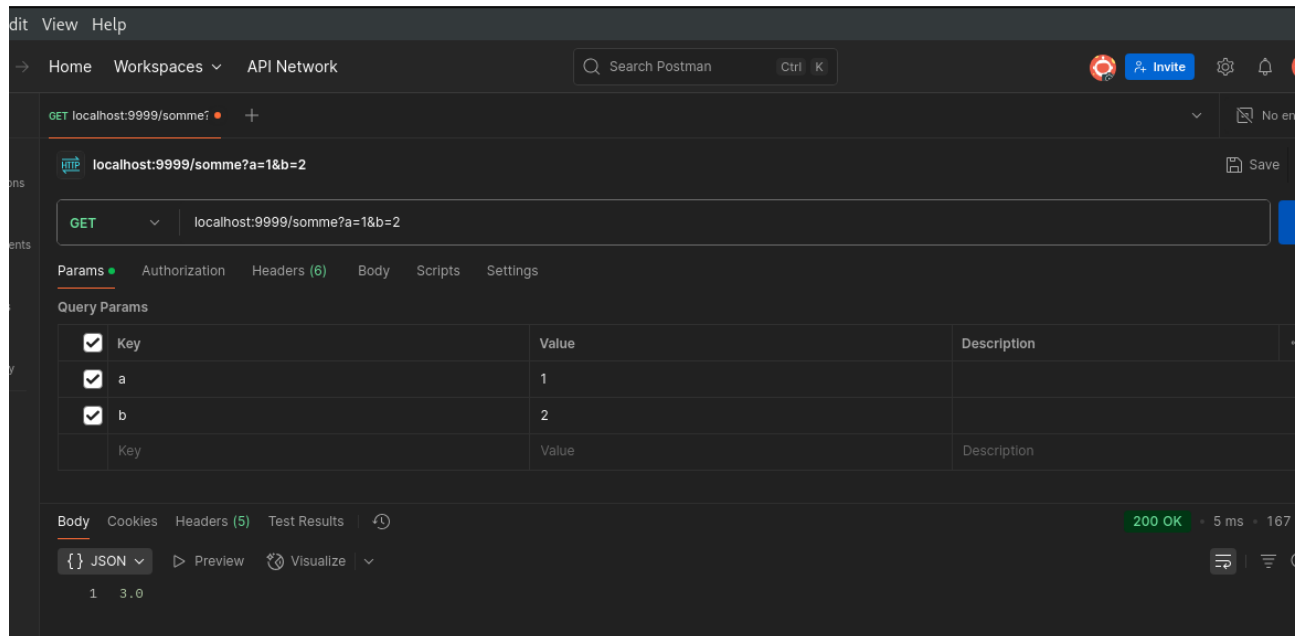
```
public class MyApi {  
    @GetMapping(value = "/etudiant") no usages new *  
    public Etudiant getEtudiant() {  
        return new Etudiant( identifiant: 1, nom: "A", moyenne: 19);  
    }  
  
    @GetMapping(value = "somme") no usages new *  
    public double somme(double a, double b) {  
        return a+b;  
    }  
}
```

2. Test de la méthode somme

- Avec le navigateur :



- Avec Postman :



III. Simulation de la couche de donner, en souvegardant un ensemble d'instance de la classe étudiante dans une arraylist.

1. Méthode GET renvoie d'une ressource

```
Spring1Application.java  MyApi.java  Etudiant.java

7
8 @RestController // Annotation pour dire que c'est un controller rest nousages 2baSW*
9 public class MyApi {
10
11     public static ArrayList<Etudiant> listEtudiant = new ArrayList<>(); 5 usages
12     static {
13         listEtudiant.add(new Etudiant( identifiant: 0, nom: "A", moyenne: 19));
14         listEtudiant.add(new Etudiant( identifiant: 1, nom: "B", moyenne: 17));
15         listEtudiant.add(new Etudiant( identifiant: 2, nom: "C", moyenne: 15));
16         listEtudiant.add(new Etudiant( identifiant: 3, nom: "D", moyenne: 13));
17     }
18
19     @GetMapping(value = "/listEtudiant") no usages new *
20     public ArrayList<Etudiant> getAllEtudiant() {
21         return listEtudiant;
22     }
23 }
```

```
localhost:9999/listEtudiant

GET localhost:9999/listEtudiant

Params Authorization Headers (6) Body Scripts Settings

Body Cookies Headers (5) Test Results 200 OK

[] JSON Preview Visualize

1 [
2   {
3     "identifiant": 0,
4     "nom": "A",
5     "moyenne": 19.0
6   },
7   {
8     "identifiant": 1,
9     "nom": "B",
10    "moyenne": 17.0
11  },
12  {
13    "identifiant": 2,
14    "nom": "C",
15    "moyenne": 15.0
16  },
17  {
18    "identifiant": 3,
19    "nom": "D",
```

```
24
25 @GetMapping(value = "/getEtudiant") no usages new *
26 public Etudiant getEtudiant(int identifiant) {
27     return listEtudiant.get(identifiant);
28 }
29
```

GET localhost:9999/getEtudiant?identifiant=1

Params • Authorization Headers (6) Body Scripts Settings

Query Params

<input checked="" type="checkbox"/>	Key	Value
<input checked="" type="checkbox"/>	identifiant	1
	Key	Value

Body Cookies Headers (5) Test Results ↺

{ } JSON ▾ ▶ Preview 🔄 Visualize ▾

```
1 {
2   "identifiant": 1,
3   "nom": "B",
4   "moyenne": 17.0
5 }
```

2. Méthode POST ajouter une ressource

```
29
30 @PostMapping(value = "/addEtudiant") no usages new *
31 public Etudiant addEtudiant(Etudiant etudiant) {
32     listEtudiant.add(etudiant);
33     return etudiant;
34 }
35
36
```

POST localhost:9999/addEtudiant?identifiant=4&nom="E"&moyenne=18

Params • Authorization Headers (7) Body Scripts Settings

<input checked="" type="checkbox"/>	identifiant	4
<input checked="" type="checkbox"/>	nom	"E"
<input checked="" type="checkbox"/>	moyenne	18
	Key	Value

Body Cookies Headers (5) Test Results ↺

{ } JSON ▾ ▶ Preview 🔄 Visualize ▾

```
1 {
2   "identifiant": 4,
3   "nom": "\"E\"",
4   "moyenne": 18.0
5 }
```


GET localhost:9999/listEtudiant

Params Authorization Headers (6) Body Scripts Settings

Query Params

	Key	Value
	Key	Value

Body Cookies Headers (5) Test Results

{ } JSON Preview Visualize

```

17  {
18      "identifiant": 3,
19      "nom": "D",
20      "moyenne": 13.0
21  },
22  {
23      "identifiant": 4,
24      "nom": "\"E\"",
25      "moyenne": 18.0
26  }
27  ]

```

3. Méthode DELETE pour supprimer une ressource :

```

36
37  @DeleteMapping(value = "/deleteEtudiant") no usages new *
38  public void deleteEtudiant(int identifiant) {
39      listEtudiant.remove(identifiant);
40  }
41

```

DELETE localhost:9999/deleteEtudiant?identifiant=1

Params Authorization Headers (6) Body Scripts Settings

Query Params

<input checked="" type="checkbox"/>	Key	Value
<input checked="" type="checkbox"/>	identifiant	1
	Key	Value

Body Cookies Headers (4) Test Results

Raw Preview Visualize

1

GET localhost:9999/listEtudiant

Params Authorization Headers (6) Body Scripts Settings

Query Params

Key	Value	Description
Key	Value	Description

Body Cookies Headers (5) Test Results 200 OK

{ } JSON Preview Visualize

```
1 [
2   {
3     "identifiant": 0,
4     "nom": "A",
5     "moyenne": 19.0
6   },
7   {
8     "identifiant": 2,
9     "nom": "C",
10    "moyenne": 15.0
11  },
12 ]
```

4. Méthode PUT pour modifier une ressource

```
38
39     @PutMapping(value = "/updateEtudiant") no usages new *
40     public void UpdateEtudiant(int identifiant, String nom) {
41         listEtudiant.get(identifiant).setNom(nom);
42     }
43
```

PUT localhost:9999/updateEtudiant?identifiant=1&nom=Baba

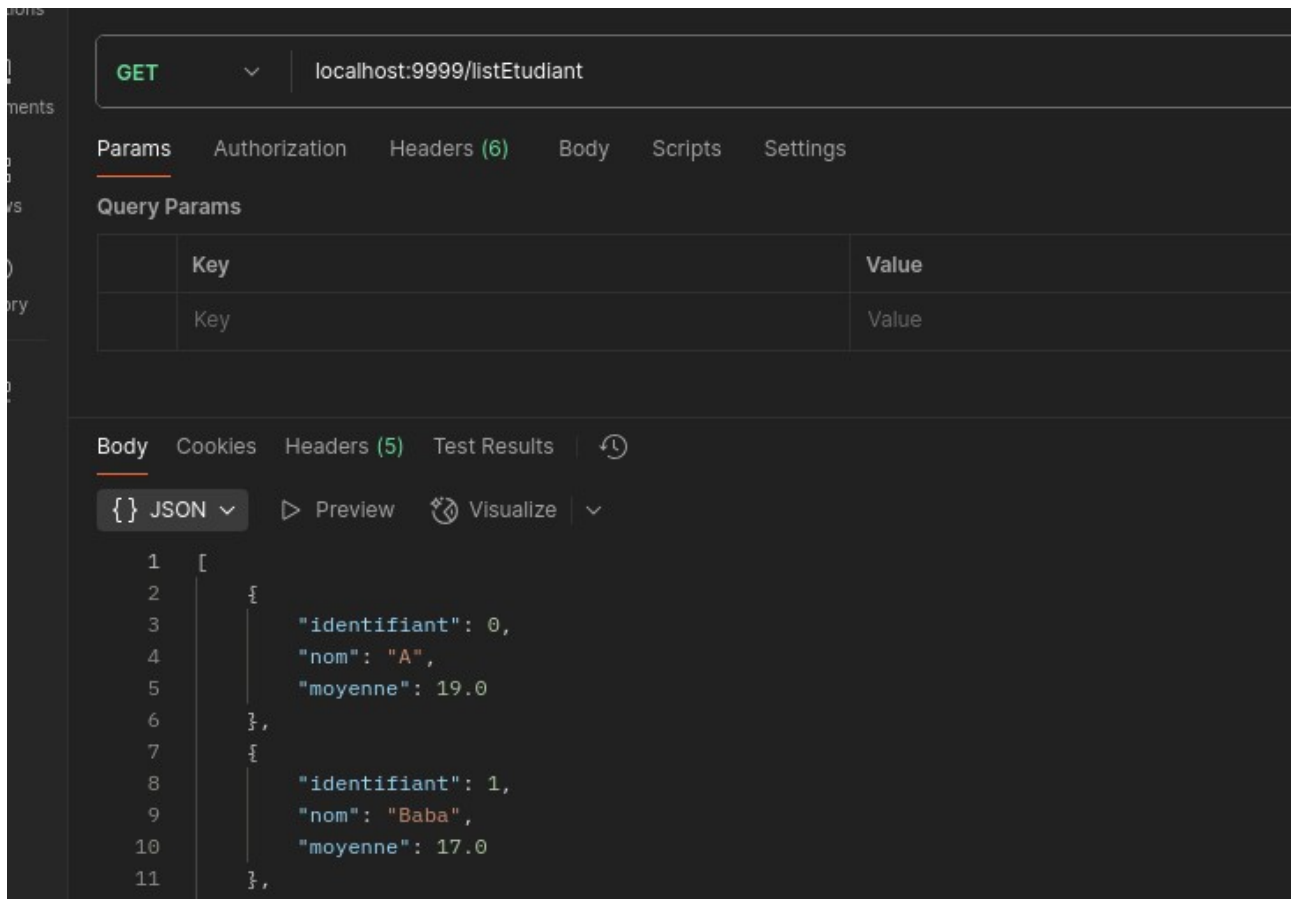
Params Authorization Headers (7) Body Scripts Settings

<input checked="" type="checkbox"/>	Key	Value
<input checked="" type="checkbox"/>	identifiant	1
<input checked="" type="checkbox"/>	nom	Baba
	Key	Value

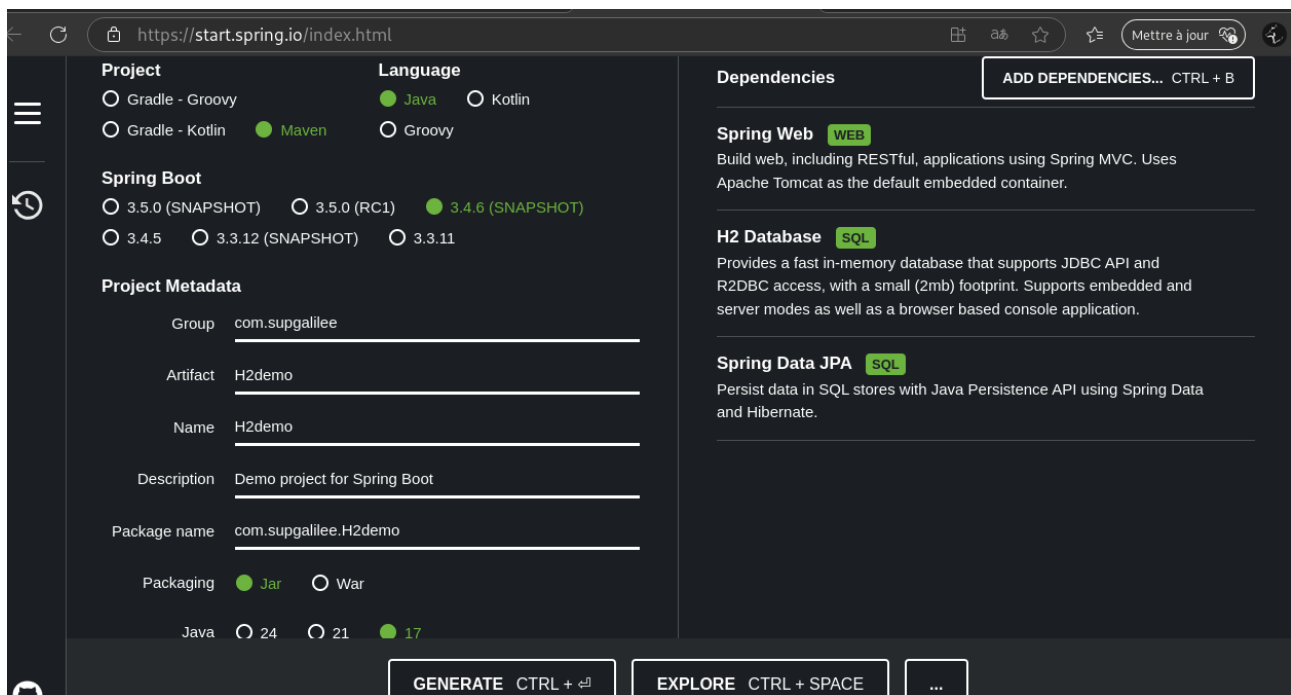
Body Cookies Headers (4) Test Results

Raw Preview Visualize

1



IV) JPA, H2 en mémoire

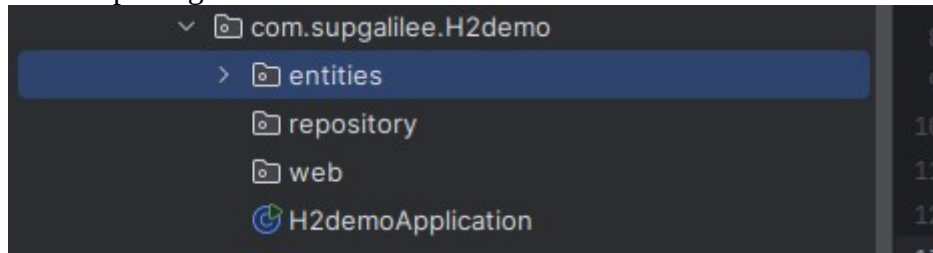


- H2 => parie BD

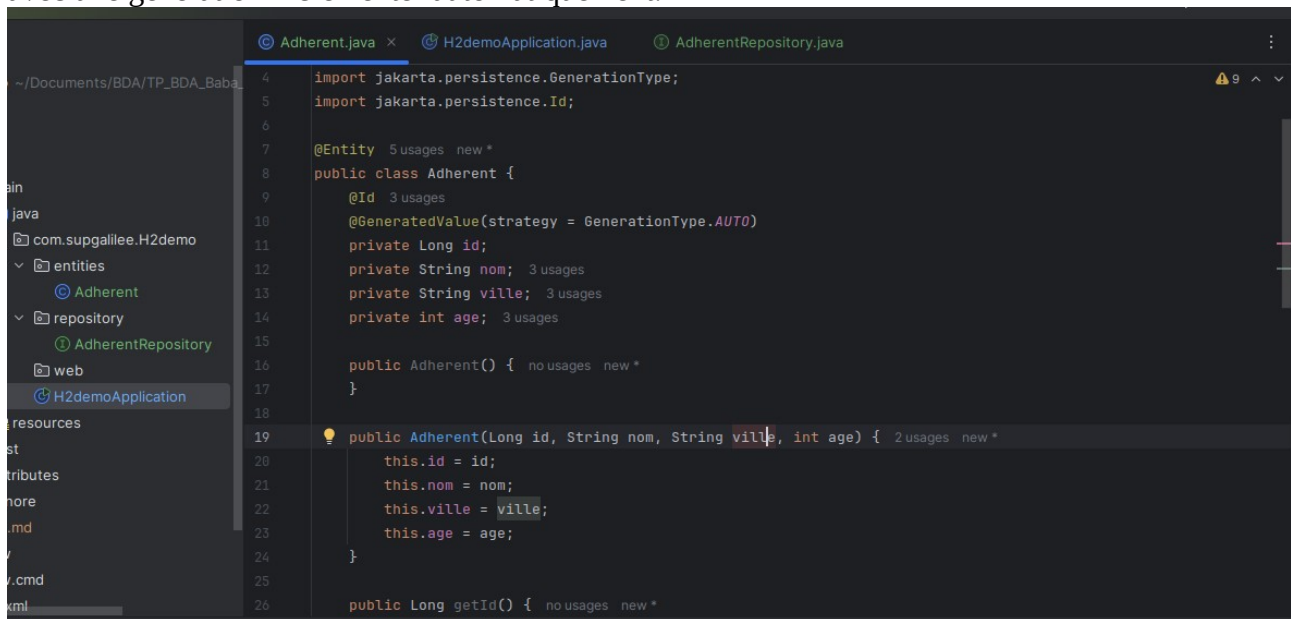
Hibernate => une des implémentation de JPA

- JPA => Spring Data JPA

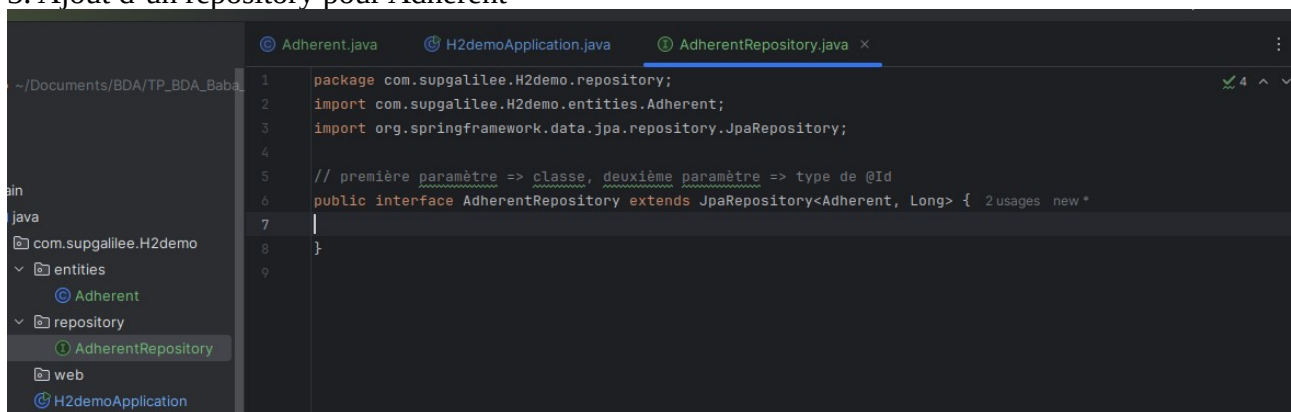
1. Ajout de nouveaux package



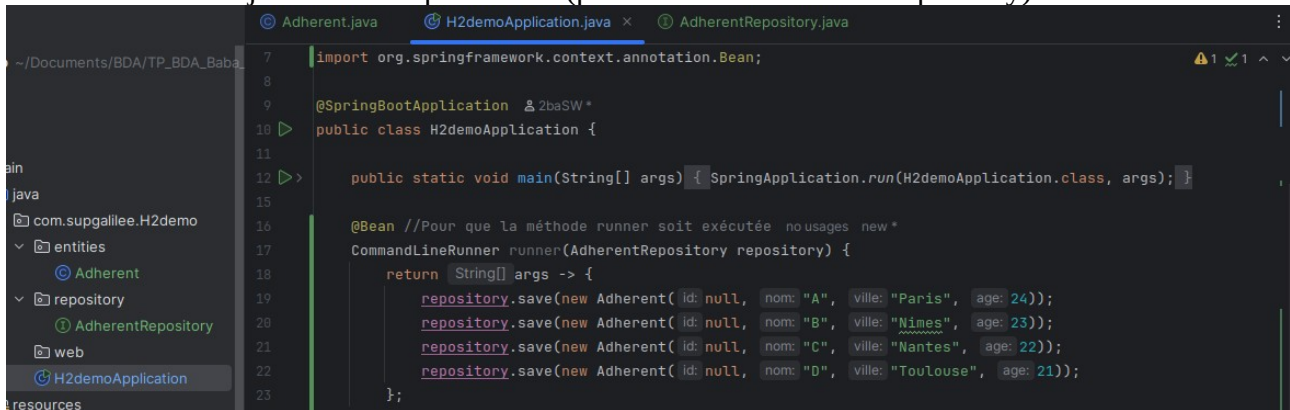
2. Ajout d'une entité Adherent avec en précision @Id pour définir préciser la clé primaire avec une génération incrémenter automatiquement.



3. Ajout d'un repository pour Adherent

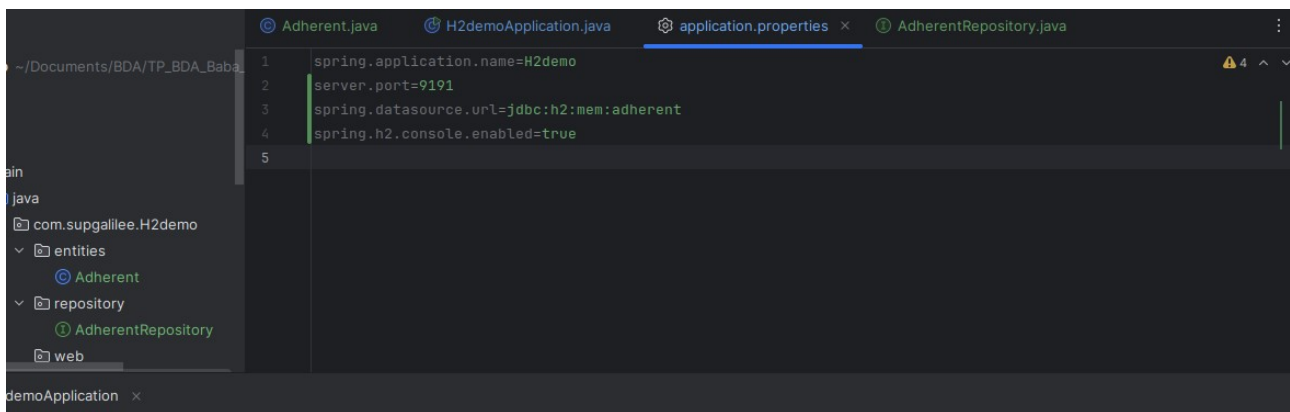


4. Ajout d'une méthode runner dans l'application pour ajouter des (c'est à dire pour tester)
On a utiliser des injections de dépendances (par l'interface AdherentRepository)



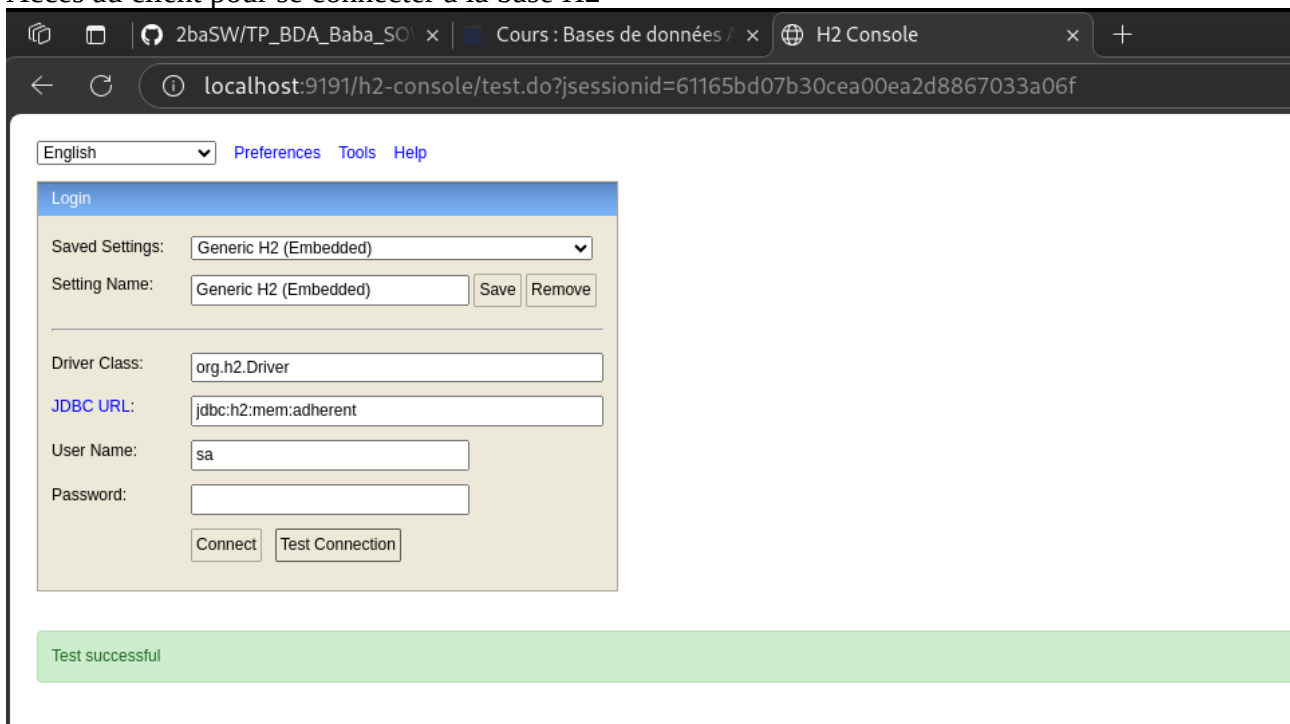
```
7 import org.springframework.context.annotation.Bean;
8
9 @SpringBootApplication
10 public class H2demoApplication {
11
12     public static void main(String[] args) { SpringApplication.run(H2demoApplication.class, args); }
13
14
15
16     @Bean //Pour que la méthode runner soit exécutée nousages new *
17     CommandLineRunner runner(AdherentRepository repository) {
18         return String[] args -> {
19             repository.save(new Adherent( id: null, nom: "A", ville: "Paris", age: 24));
20             repository.save(new Adherent( id: null, nom: "B", ville: "Nîmes", age: 23));
21             repository.save(new Adherent( id: null, nom: "C", ville: "Nantes", age: 22));
22             repository.save(new Adherent( id: null, nom: "D", ville: "Toulouse", age: 21));
23         };
24     }
25 }
```

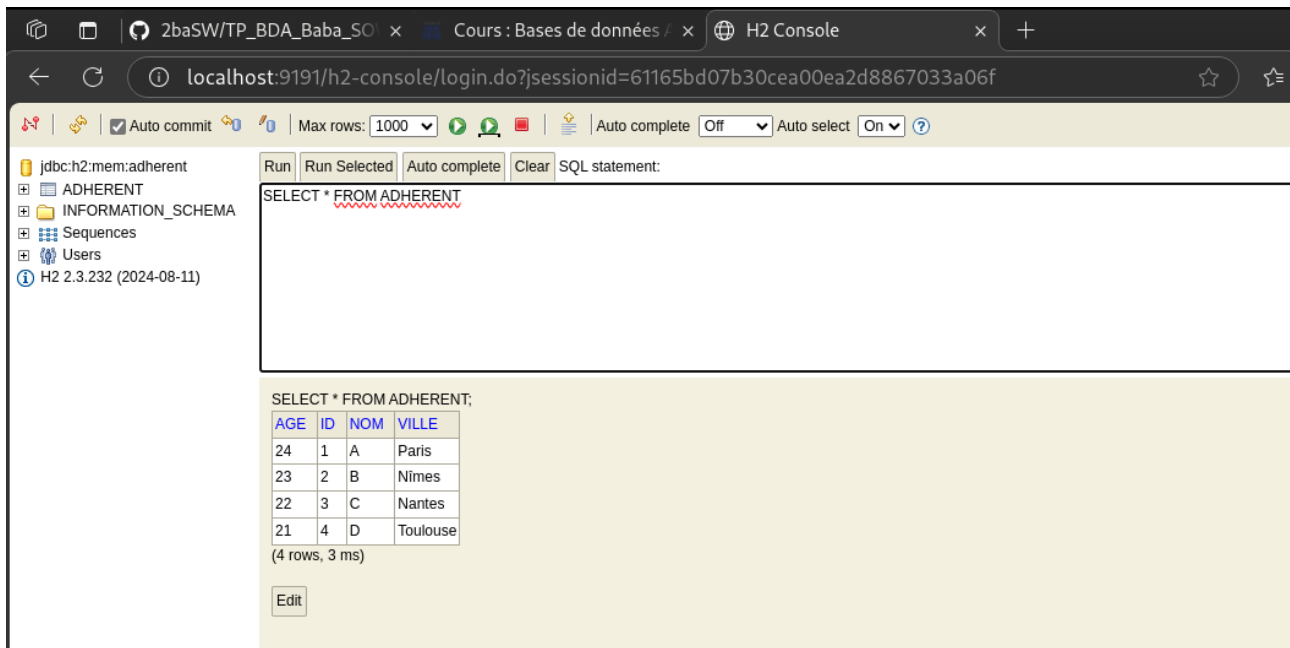
5. Configuration **application.properties**, avec une BD en mémoire nommée adherent, on active aussi la console pour le client:



```
1 spring.application.name=H2demo
2 server.port=9191
3 spring.datasource.url=jdbc:h2:mem:adherent
4 spring.h2.console.enabled=true
5
```

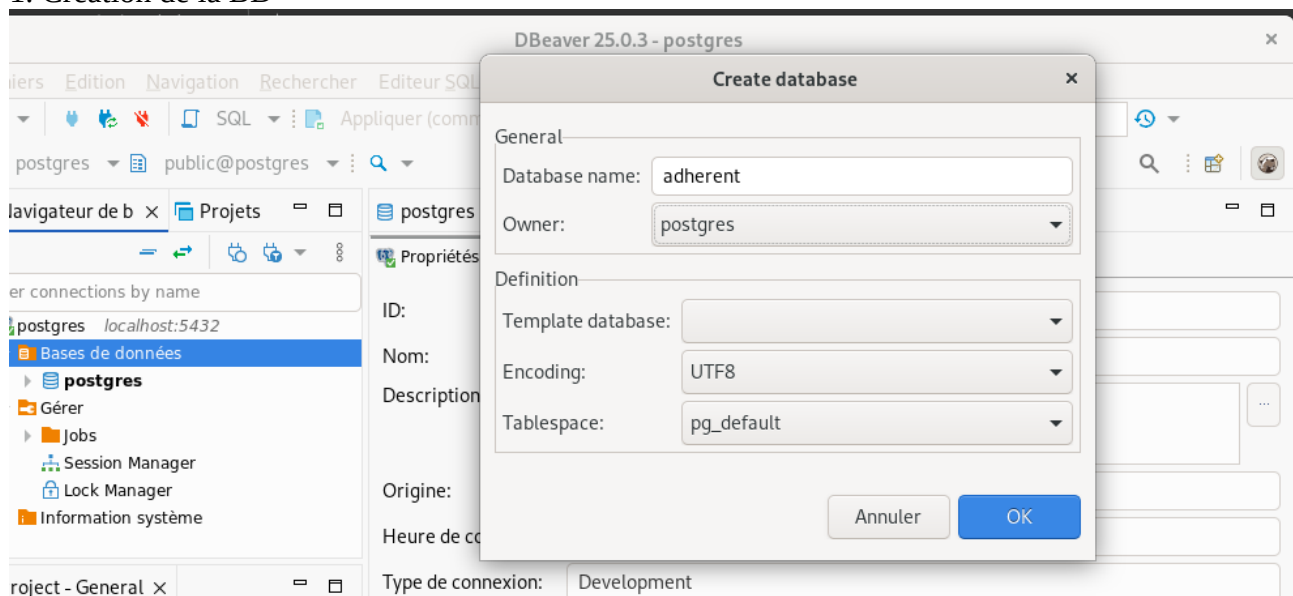
Accès au client pour se connecter à la base H2





V) Connection à un BD de type PostgreSQL

1. Création de la BD



2. Configuration dans application.properties pour se connecter à la BD PostgreSQL adherent

```
Adherent.java  H2demoApplication.java  application.properties  pom.xml (H2demo)  Adh

1  spring.application.name=H2demo
2  server.port=9191
3
4  spring.datasource.url=jdbc:postgresql://localhost:5432/adherent
5  spring.datasource.username=postgres
6  spring.datasource.password=baba
7
8  # Hibernate configuration
9  spring.jpa.hibernate.ddl-auto=create
```

3. Ajout de la dependency pour PostgreSQL dans le pom.xml

```
Adherent.java  H2demoApplication.java  application.properties  pom.xml (H2demo)  Adh

2  <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema
32  <dependencies>
37  <dependency>
38  <groupId>org.springframework.boot</groupId>
39  <artifactId>spring-boot-starter-web</artifactId>
40  </dependency>
41
42  <dependency>
43  <groupId>org.postgresql</groupId>
44  <artifactId>postgresql</artifactId>
45  <scope>runtime</scope>
46  </dependency>
47
48
```

4. Visualisation de la table créée et des données:

DBeaver 25.0.3 - adherent

Fichiers Edition Navigation Rechercher Editeur SQL Base de données Fenêtres Aide

SQL Appliquer (commit) Retour arrière (rollback) Auto adherent public@adherent

Navigateur de bases de données Projets adherent x

Propriétés Données Diagramme

Filter connections by name

adherent localhost:5432

- Bases de données
 - adherent
 - Schémas
 - public
 - Tables
 - adherent

adherent

123 age 123 id A-Z nom A-Z ville

1	24	1	A	Paris
2	23	2	B	Nimes
3	22	3	C	Nantes
4	21	4	D	Toulouse

24