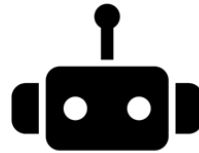




**PUCP**

## SESIÓN DE LABORATORIO 2

### Soap y Rest



*HORARIO 10M1*

Empezaremos a las 8:10 p.m

Gracias!

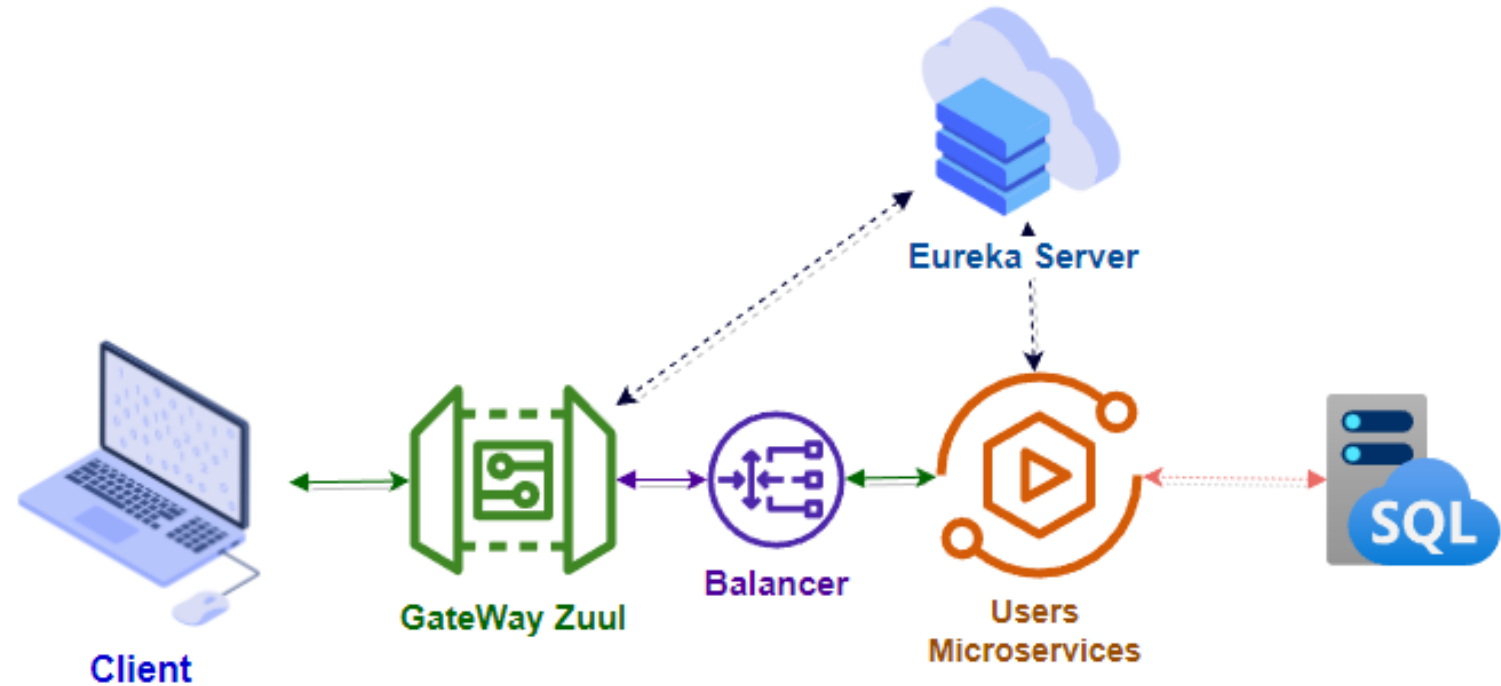


# Arquitectura

## API Gateway

## Spring Cloud

Allows a single and centralized access to our microservices



# Spring Data JPA

Estas son las siglas de Java Persistence API (JPA).

Se utiliza para persistir datos entre un objeto Java y una base de datos relacional.

JPA actúa como puente entre los modelos de dominio orientados a objetos y los sistemas de bases de datos relacionales.

Como JPA es sólo una especificación, no realiza ninguna operación por sí misma.



# Spring Data JPA

Spring Data JPA forma parte de la familia Spring Data y facilita la implementación de repositorios basados en JPA.

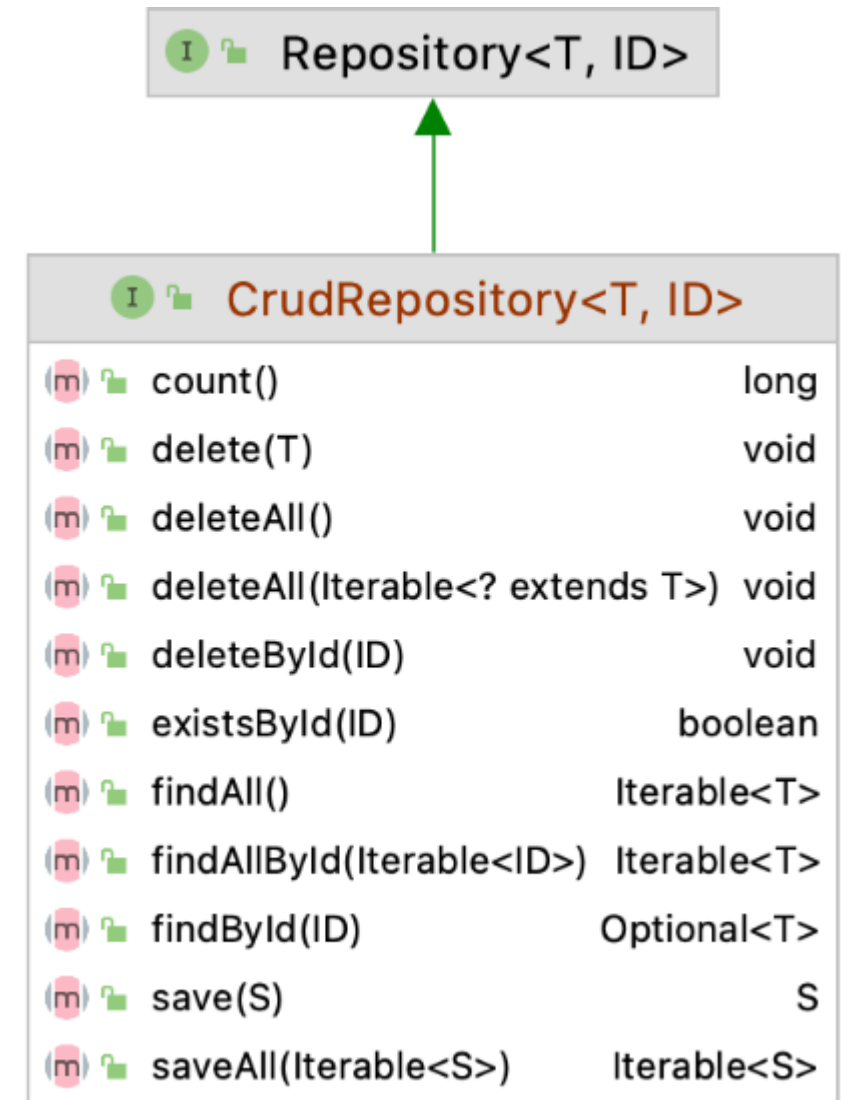
Este módulo trata del soporte mejorado para las capas de acceso a datos basadas en JPA facilitando la construcción de aplicaciones potenciadas por Spring que utilizan tecnologías de acceso a datos reduciendo el esfuerzo con la ayuda de interfaces Spring Data como CrudRepository



# CrudRepository

CrudRepository es una interfaz de Spring Data para operaciones CRUD genéricas en un repositorio y proporciona varios métodos para interactuar con una base de datos como Create, Read, Update y delete.

Es necesario recordar que CrudRepository es sólo una interfaz base y funciona como una extensión de nuestra interfaz Repository.



# Maria DB Configuration



[Download](#) | [Documentation](#) | [Contribute](#) | [Server Fest](#) | [Events](#) | [Sponsor](#) | [Blog](#) | [Planet MariaDB Server](#) | [About](#)

[Sign up for the MariaDB Day in Brussels 1.02.2025!](#)

[Download MariaDB Server](#)

[REST API](#)

[Release Schedule](#)

[Reporting Bugs](#)

[MariaDB Server Statistics](#)

[Download MariaDB Server Documentation](#)

**View all releases for:**

[MariaDB Server](#)

[Connector/C](#)

[Connector/J](#)

[Connector/ODBC](#)

[Connector/Python](#)

[Connector/Node.js](#)

**MariaDB Server Version**

MariaDB Server 11.4.4

Display older releases: ☐

See the MariaDB 11.4.4 Release Notes and Changes and Improvements in MariaDB 11.4.

**Operating System**

Windows

**Architecture**

x86\_64

**Package Type**

MSI Package

Download

**Mirror**

Insacom - Valparaíso

Release date: 2024-11-04

File name: mariadb-11.4.4-winx64.msi

File size: 74.5 MB

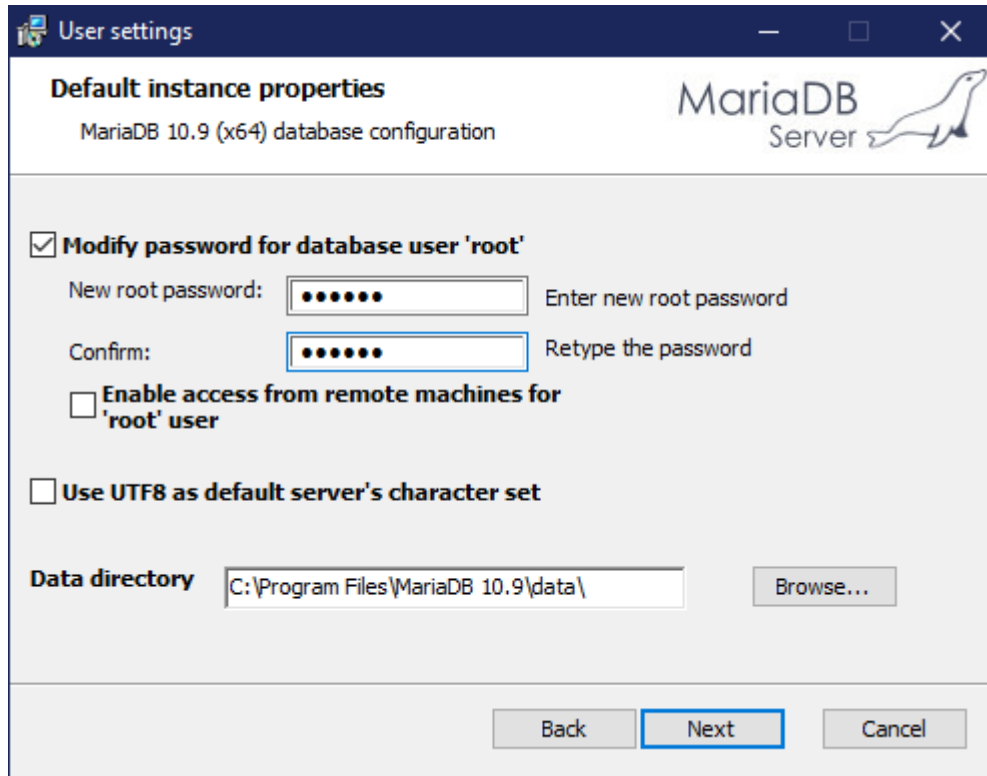
• [Download galera-26.4.20/](#)

[Display signature and checksums](#)

Remember to give us a star on GitHub

☆ Star 5,820

# Maria DB Configuration



**User settings**

Default instance properties  
MariaDB 10.9 (x64) database configuration

☒ **Modify password for database user 'root'**

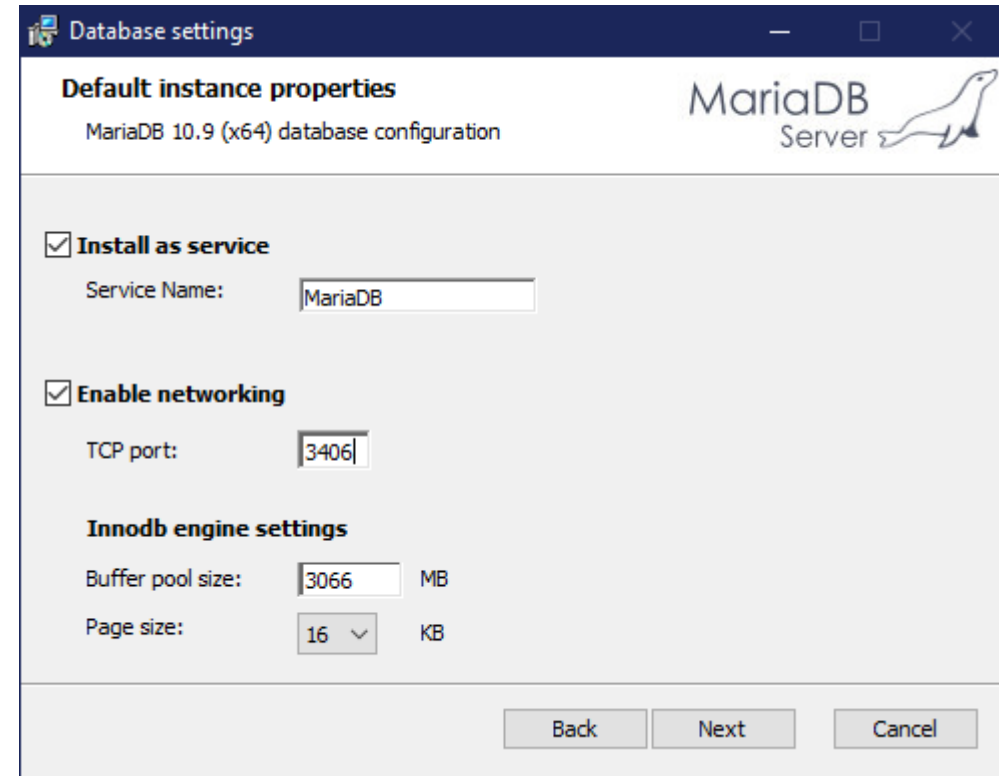
New root password:  Enter new root password

Confirm:  Retype the password

☐ **Enable access from remote machines for 'root' user**

☐ **Use UTF8 as default server's character set**

**Data directory**



**Database settings**

Default instance properties  
MariaDB 10.9 (x64) database configuration

☒ **Install as service**

Service Name:

☒ **Enable networking**

TCP port:

**InnoDB engine settings**

Buffer pool size:  MB

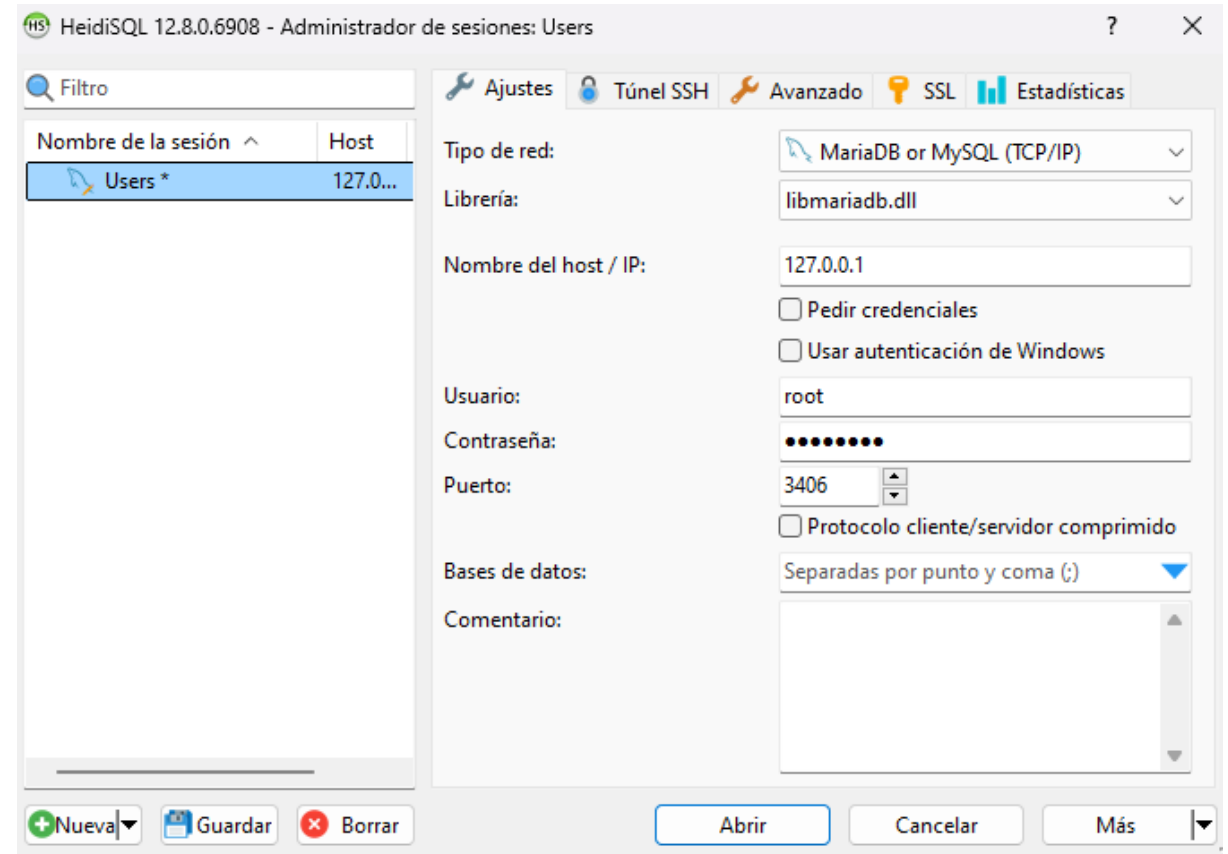
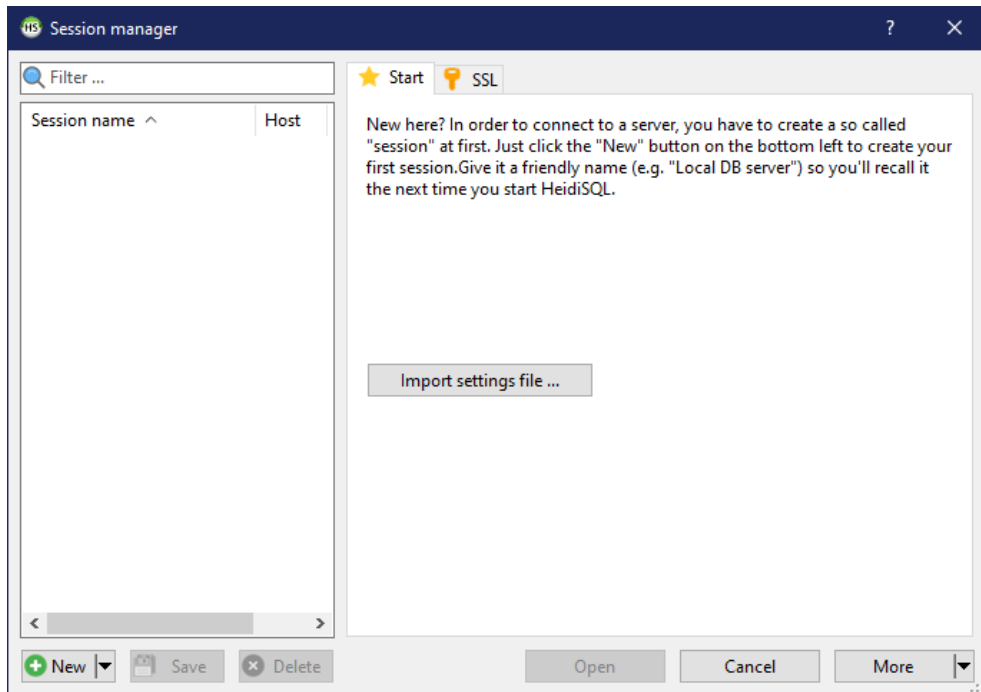
Page size:  KB



# Maria DB Configuracion (HeidiSQL)

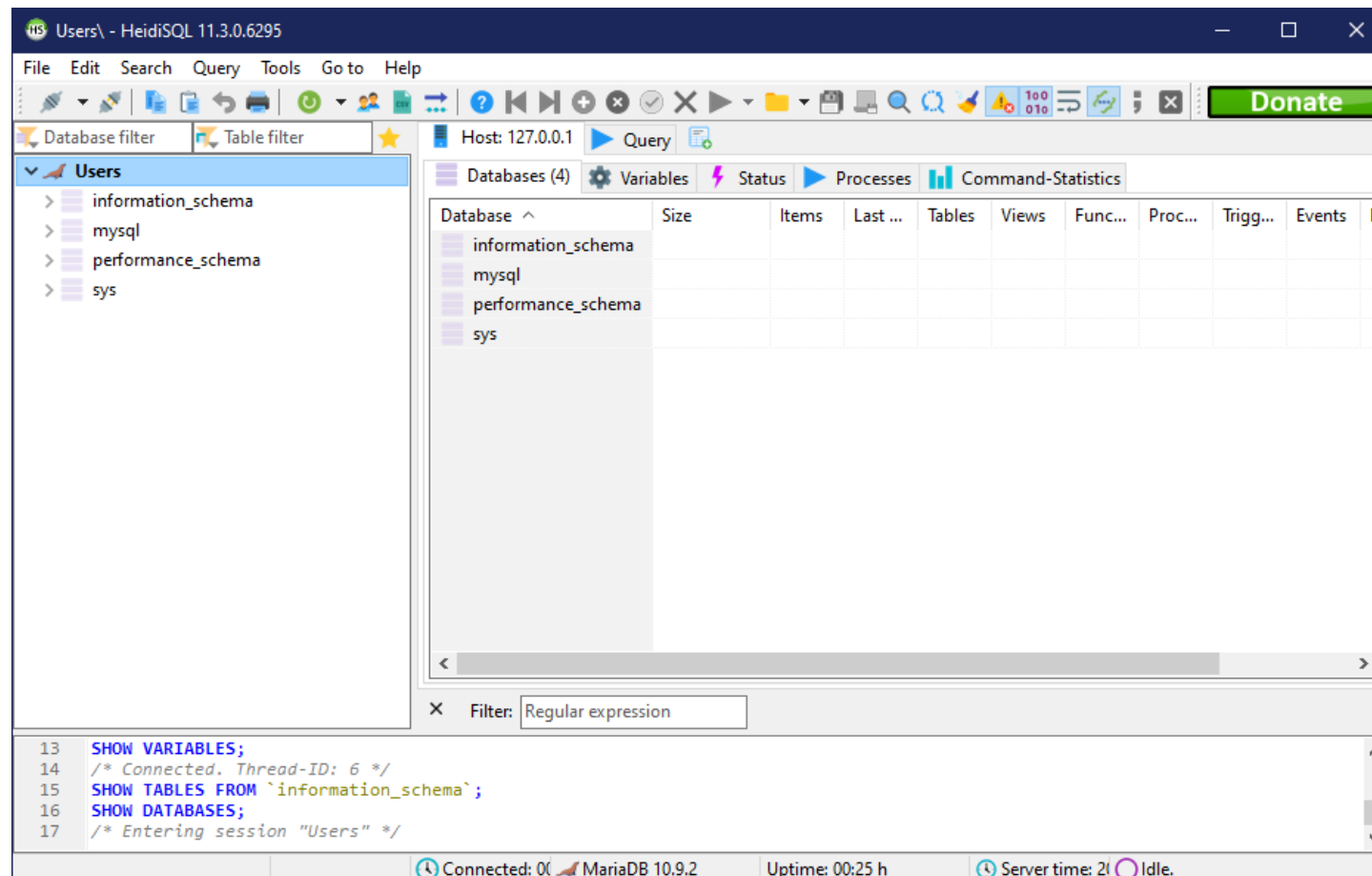


Haga clic en «Nuevo».  
Rellene el campo «Contraseña»  
Haga clic en «Abrir».

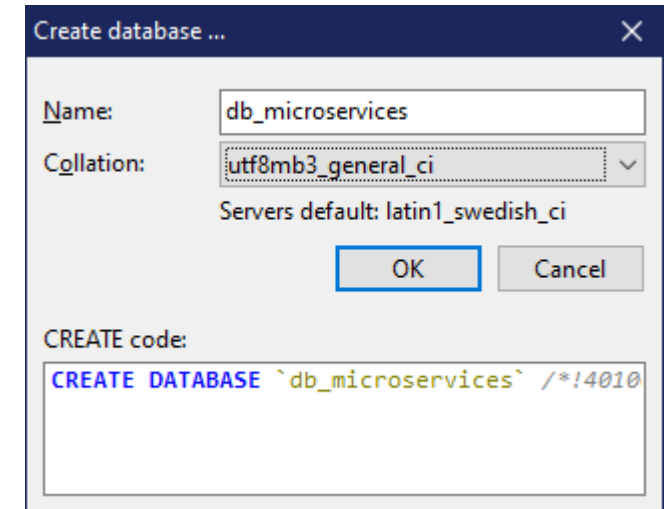
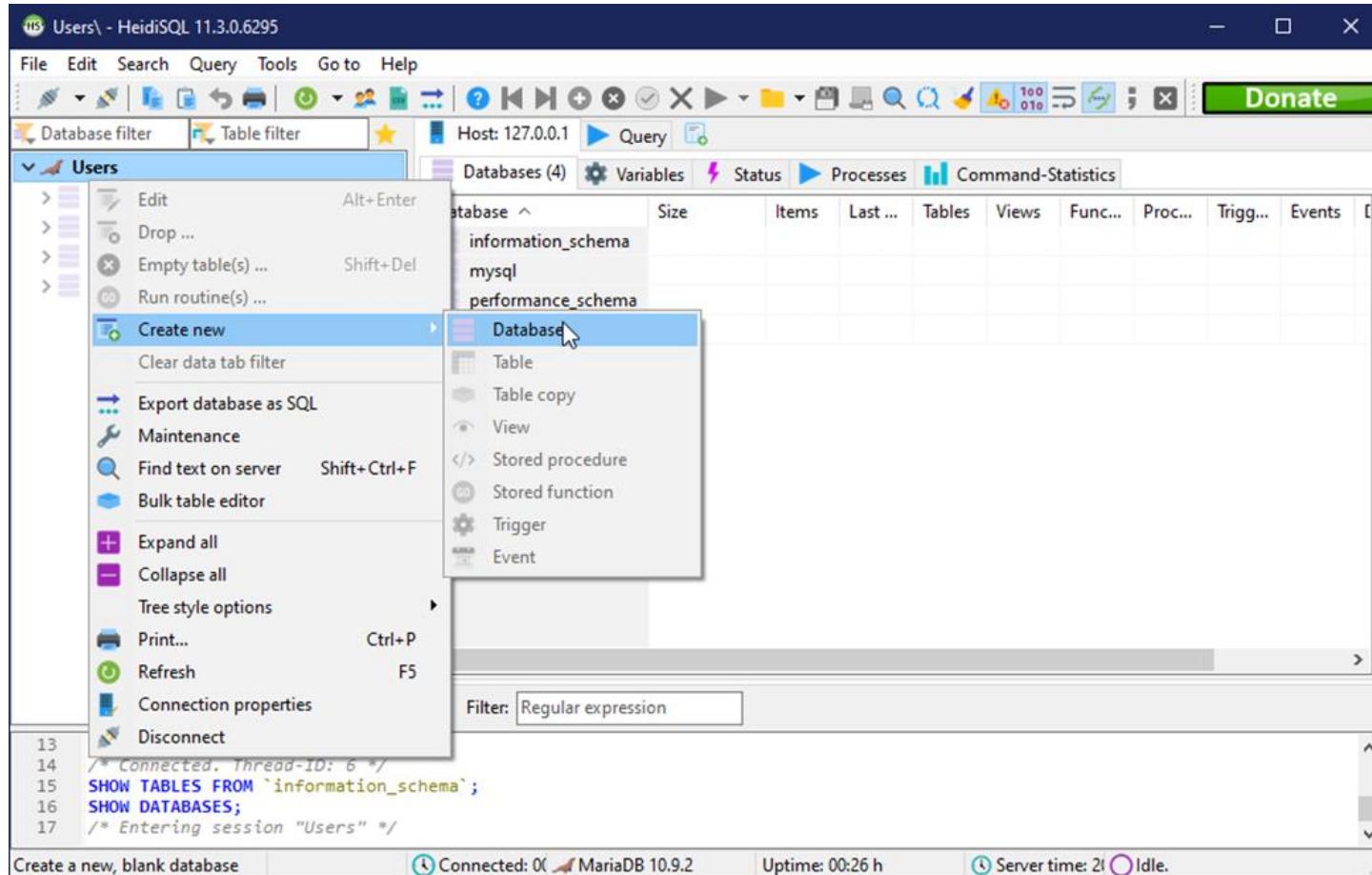




# Maria DB Configuracion (HeidiSQL)



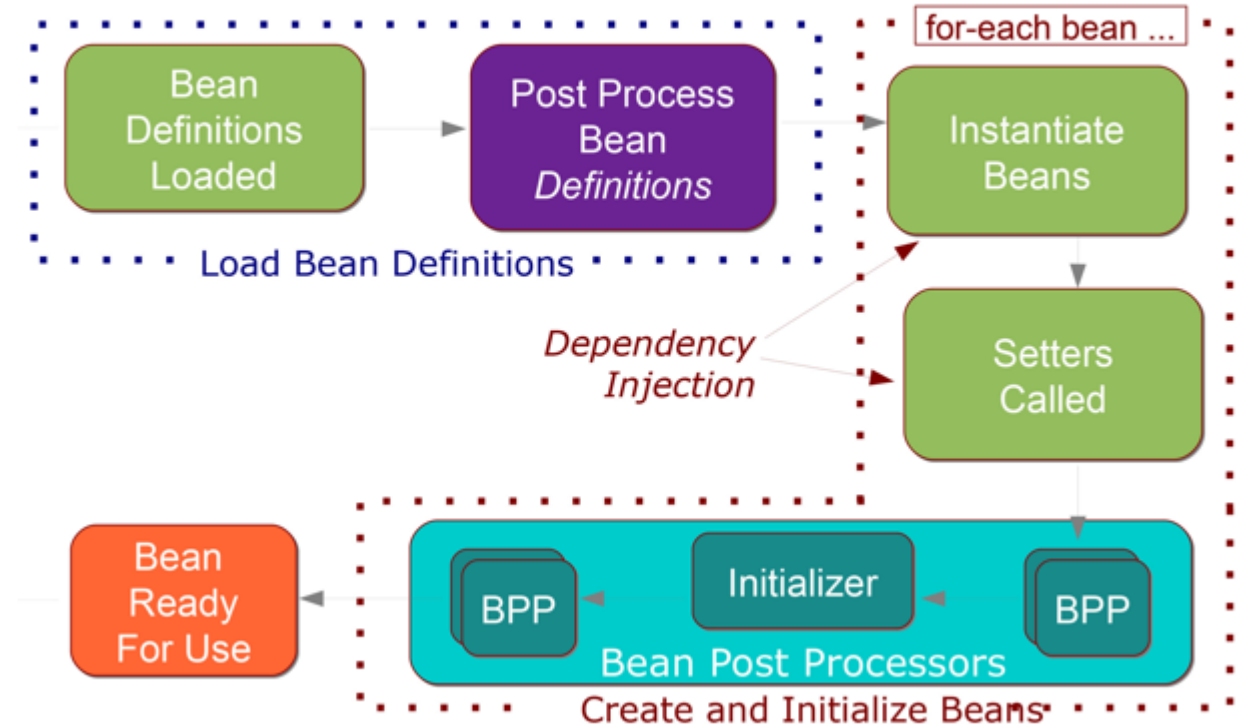
# Maria DB Configuracion (HeidiSQL)



# Bean en Spring

Son los objetos que forman la columna vertebral de la aplicación y que son instanciados, ensamblados y gestionados por un Spring ApplicationContext

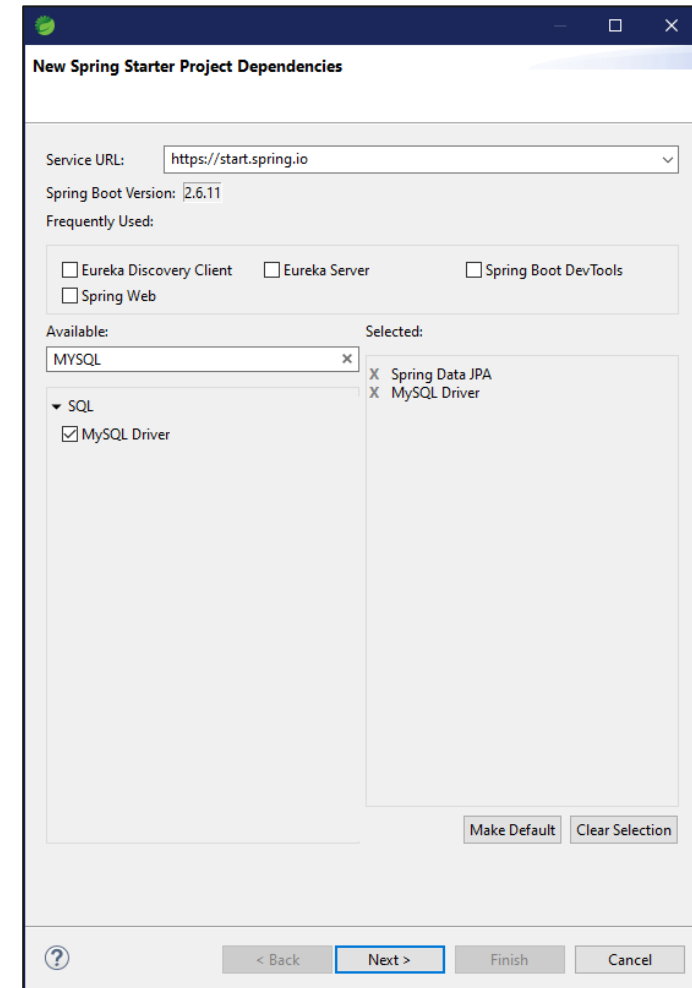
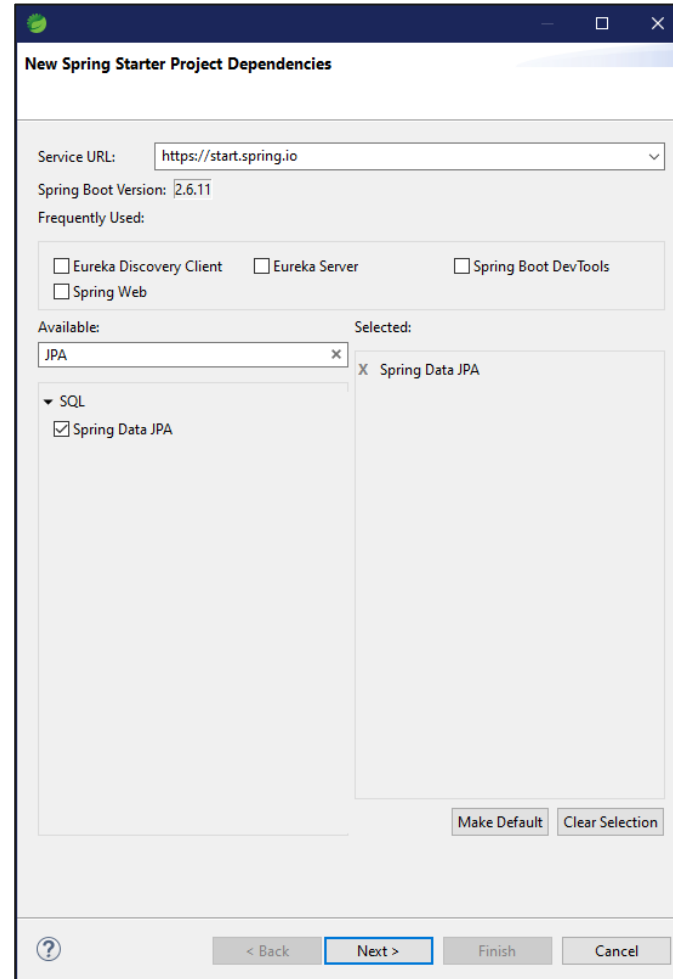
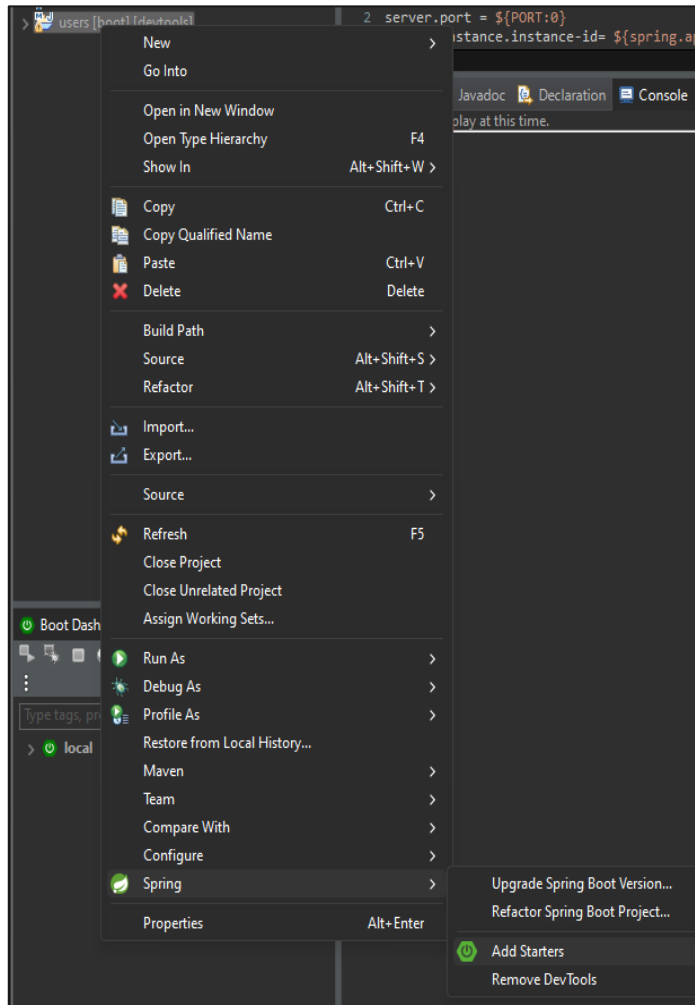
## Bean Initialization Steps



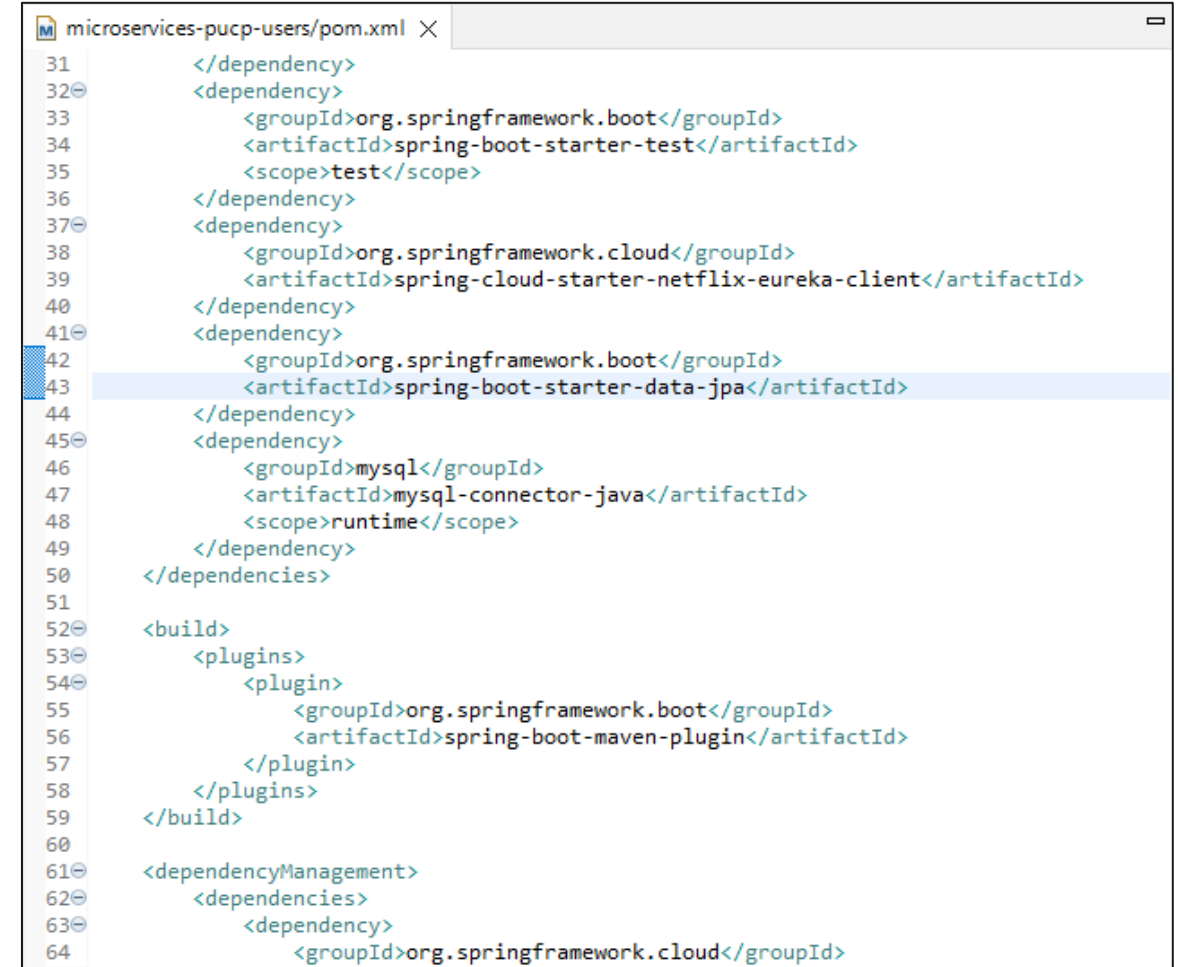
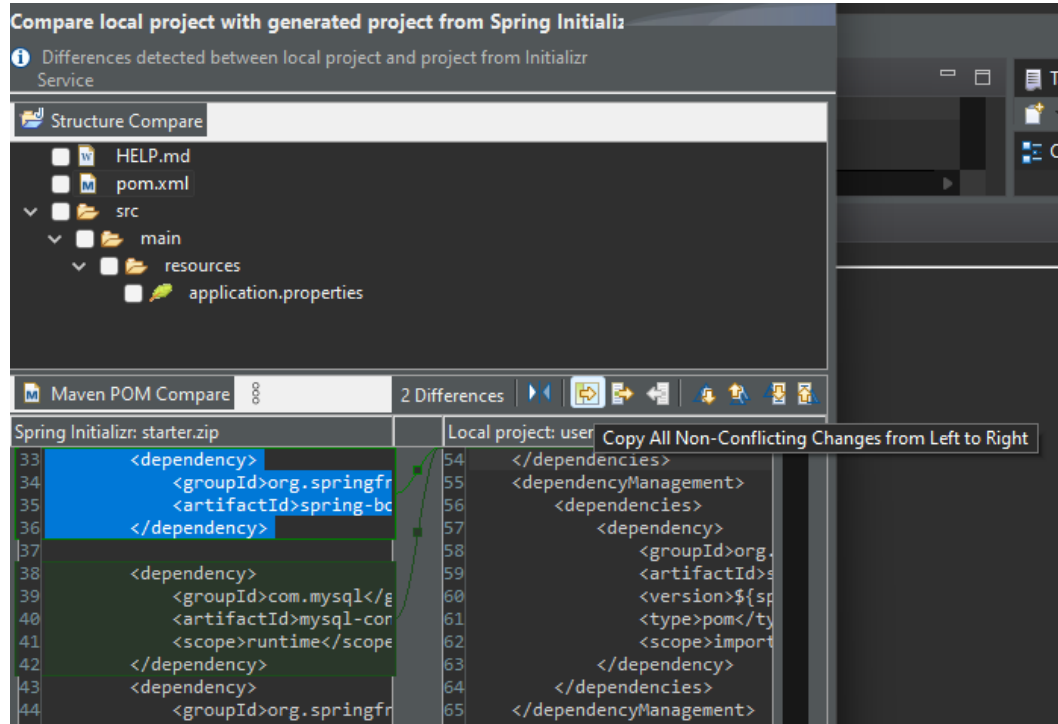
# Continuemos con la implementacion



# Adicionando dependencias para Data



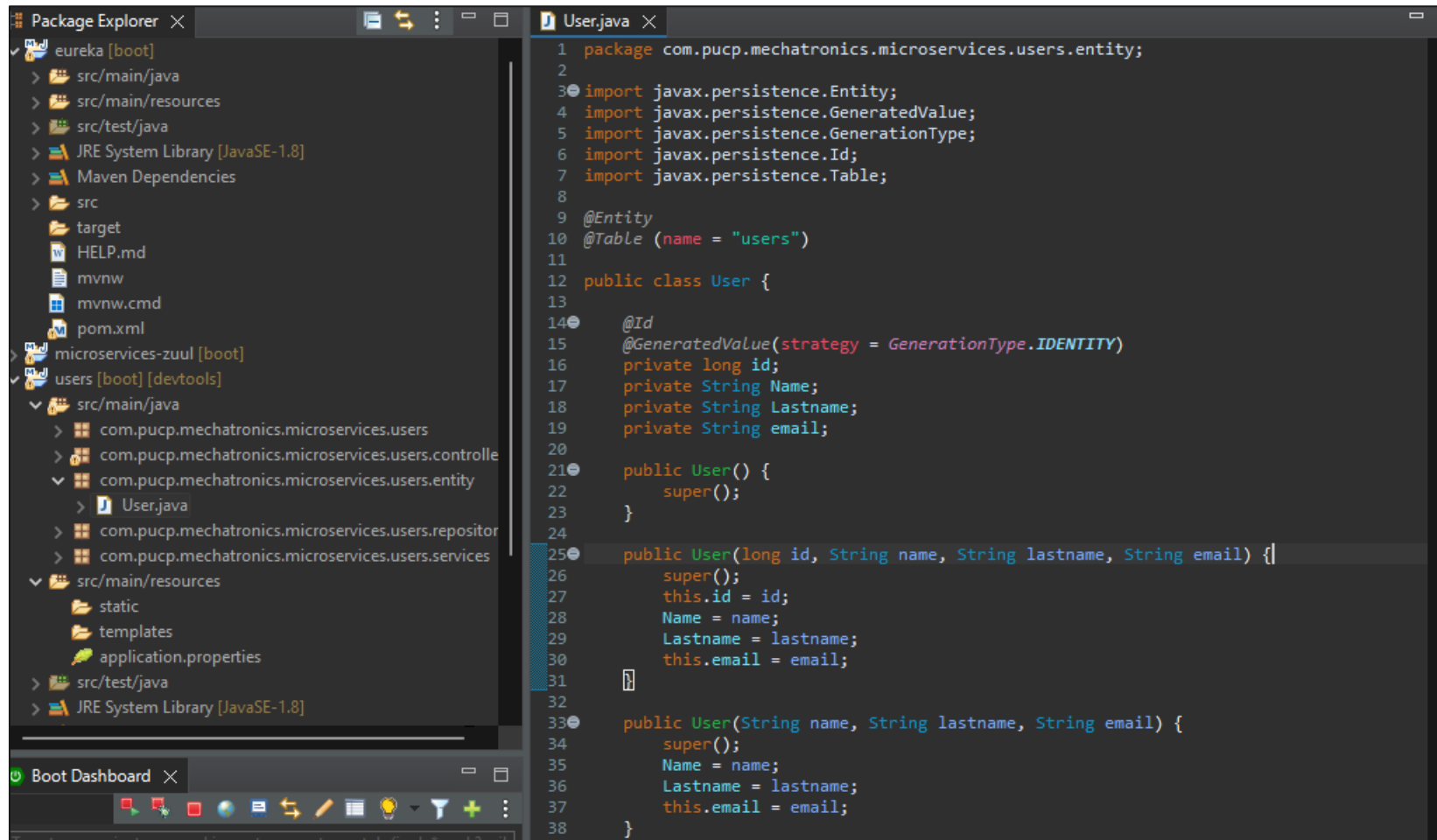
# Adicionando dependencias para Data



```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>2.7.10</version> Upgrade to the Latest Patch
  <relativePath/> <!-- lookup parent from repository -->
</parent>
<properties>
  <java.version>1.8</java.version>
  <spring-cloud.version>2021.0.6</spring-cloud.version>
</properties>
```



# Adicionando la entidad

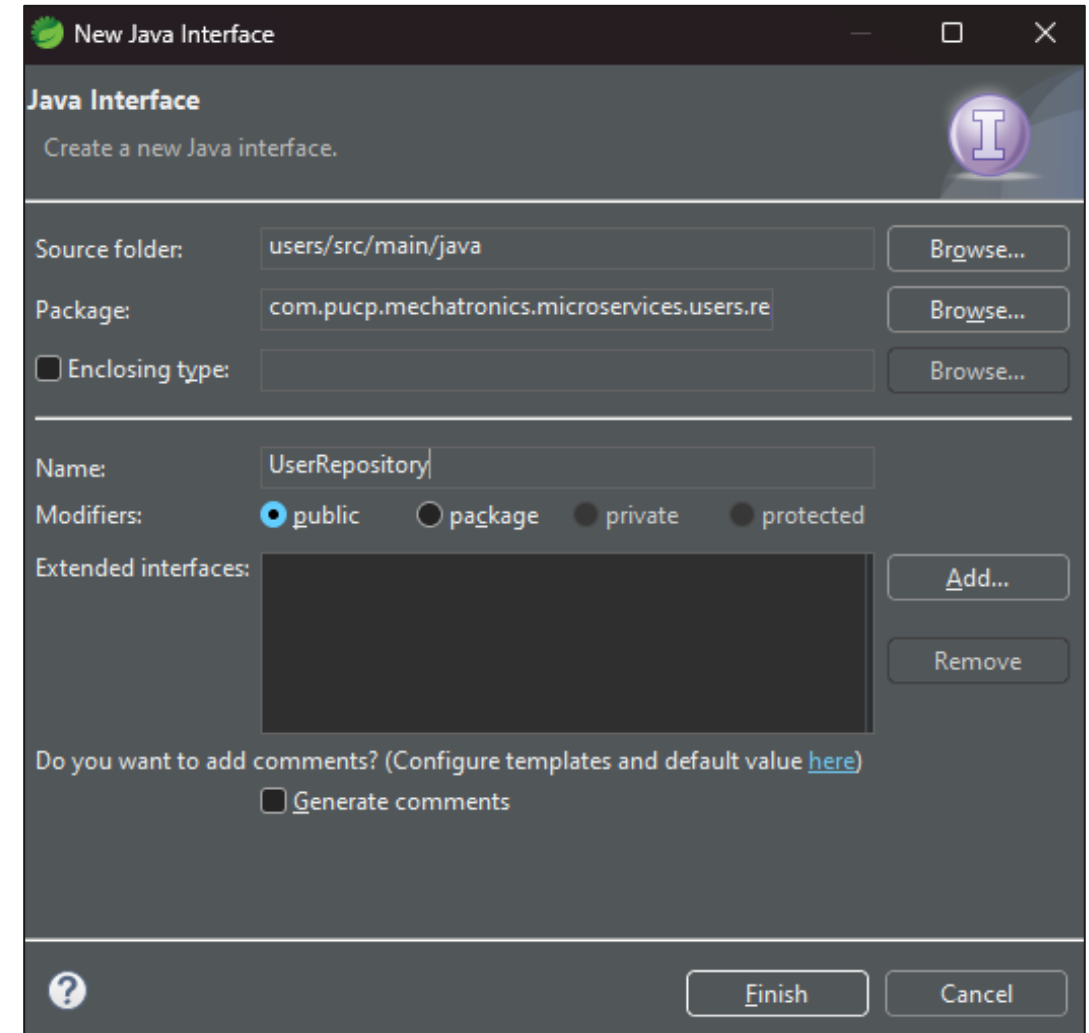
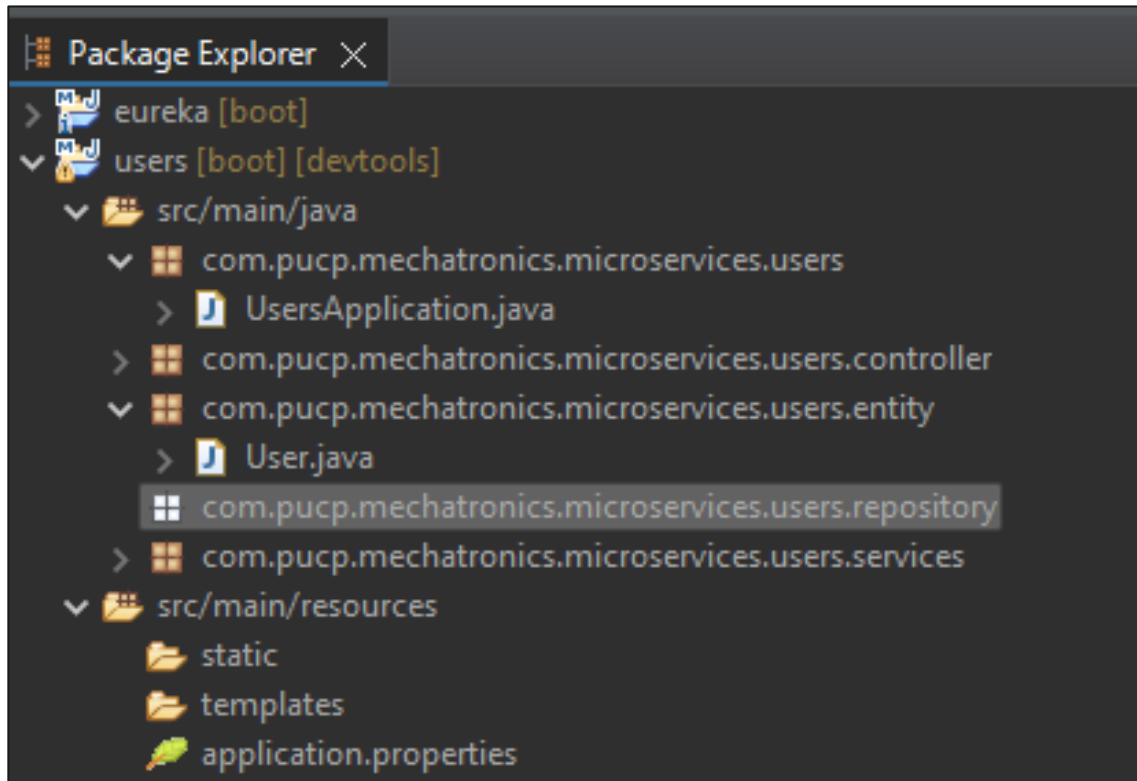


The screenshot shows an IDE with two main panels. The left panel is the Package Explorer, showing a project structure with a package named `com.pucp.mechatronics.microservices.users.entity` containing a file `User.java`. The right panel shows the code of `User.java`. The code defines a `User` entity with a primary key `id` and attributes `name`, `lastname`, and `email`. It includes imports for JPA annotations and implements the `Entity` interface.

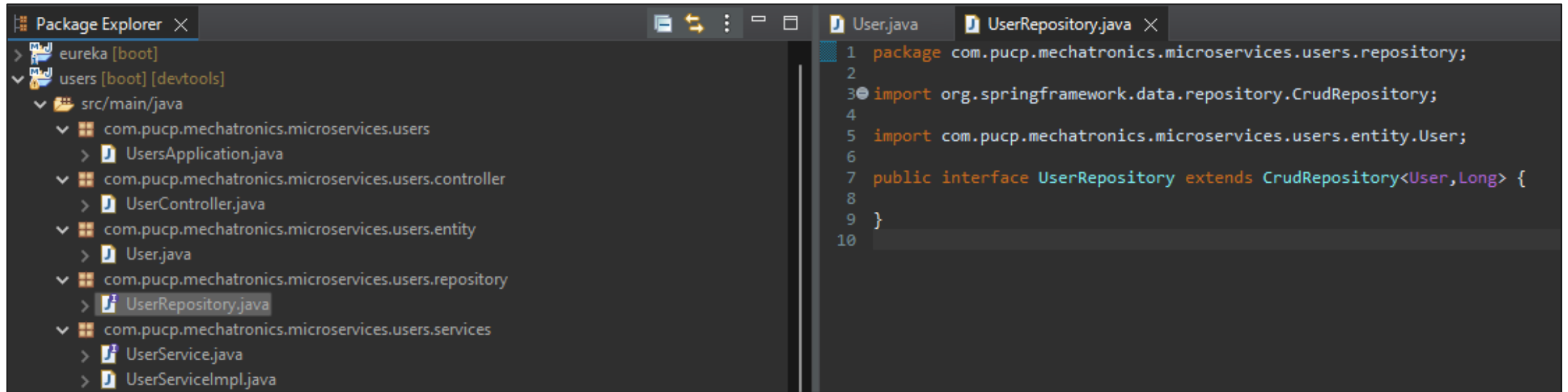
```
1 package com.pucp.mechatronics.microservices.users.entity;
2
3 import javax.persistence.Entity;
4 import javax.persistence.GeneratedValue;
5 import javax.persistence.GenerationType;
6 import javax.persistence.Id;
7 import javax.persistence.Table;
8
9 @Entity
10 @Table (name = "users")
11
12 public class User {
13
14     @Id
15     @GeneratedValue(strategy = GenerationType.IDENTITY)
16     private long id;
17     private String Name;
18     private String Lastname;
19     private String email;
20
21     public User() {
22         super();
23     }
24
25     public User(long id, String name, String lastname, String email) {
26         super();
27         this.id = id;
28         Name = name;
29         Lastname = lastname;
30         this.email = email;
31     }
32
33     public User(String name, String lastname, String email) {
34         super();
35         Name = name;
36         Lastname = lastname;
37         this.email = email;
38     }
39 }
```



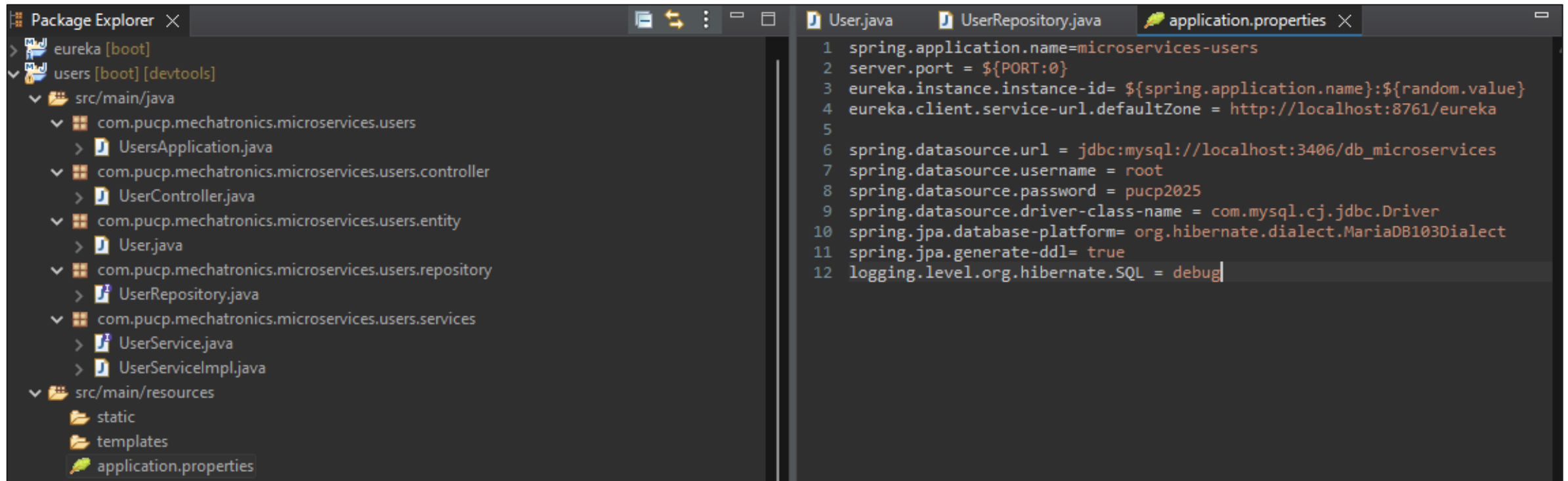
# Creando el repositorio Entidad



# Creando el repositorio Entidad



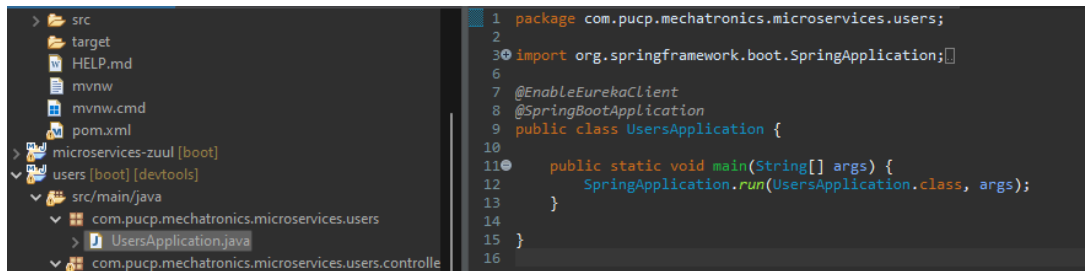
# Configurando las propiedades de nuestro microservicio



The screenshot shows an IDE with two main panels. The left panel, labeled 'Package Explorer', displays the project structure. It includes a 'users' module with a 'src/main/java' directory containing several packages: 'com.pucp.mechatronics.microservices.users', 'com.pucp.mechatronics.microservices.users.controller', 'com.pucp.mechatronics.microservices.users.entity', 'com.pucp.mechatronics.microservices.users.repository', and 'com.pucp.mechatronics.microservices.users.services'. Each package contains a corresponding Java file (e.g., 'User.java', 'UserController.java', 'UserRepository.java', 'UserService.java', 'UserServiceImpl.java'). The 'src/main/resources' directory contains 'static', 'templates', and 'application.properties' files. The right panel shows the 'application.properties' file with the following configuration:

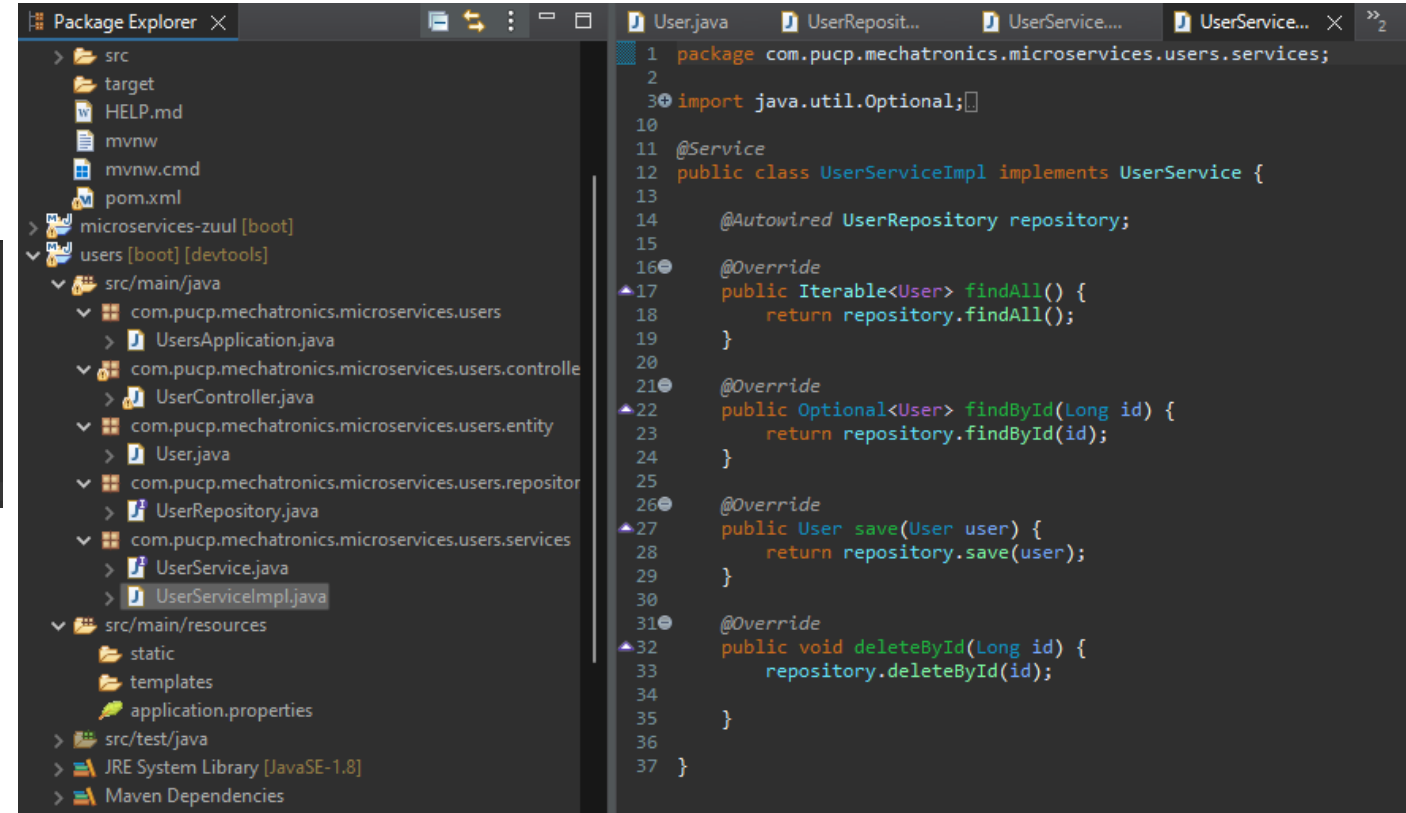
```
1 spring.application.name=microservices-users
2 server.port = ${PORT:0}
3 eureka.instance.instance-id= ${spring.application.name}:${random.value}
4 eureka.client.service-url.defaultZone = http://localhost:8761/eureka
5
6 spring.datasource.url = jdbc:mysql://localhost:3406/db_microservices
7 spring.datasource.username = root
8 spring.datasource.password = pucp2025
9 spring.datasource.driver-class-name = com.mysql.cj.jdbc.Driver
10 spring.jpa.database-platform= org.hibernate.dialect.MariaDB103Dialect
11 spring.jpa.generate-ddl= true
12 logging.level.org.hibernate.SQL = debug
```

# Configurando nuestro microservicio



This screenshot shows the left-hand side of an IDE. On the left, the 'Package Explorer' displays the project structure: a 'src' folder containing 'main' and 'test' subfolders. The 'main' folder contains a 'com.pucp.mechatronics.microservices.users' package, which includes 'UsersApplication.java'. On the right, the code editor shows the content of 'UsersApplication.java'.

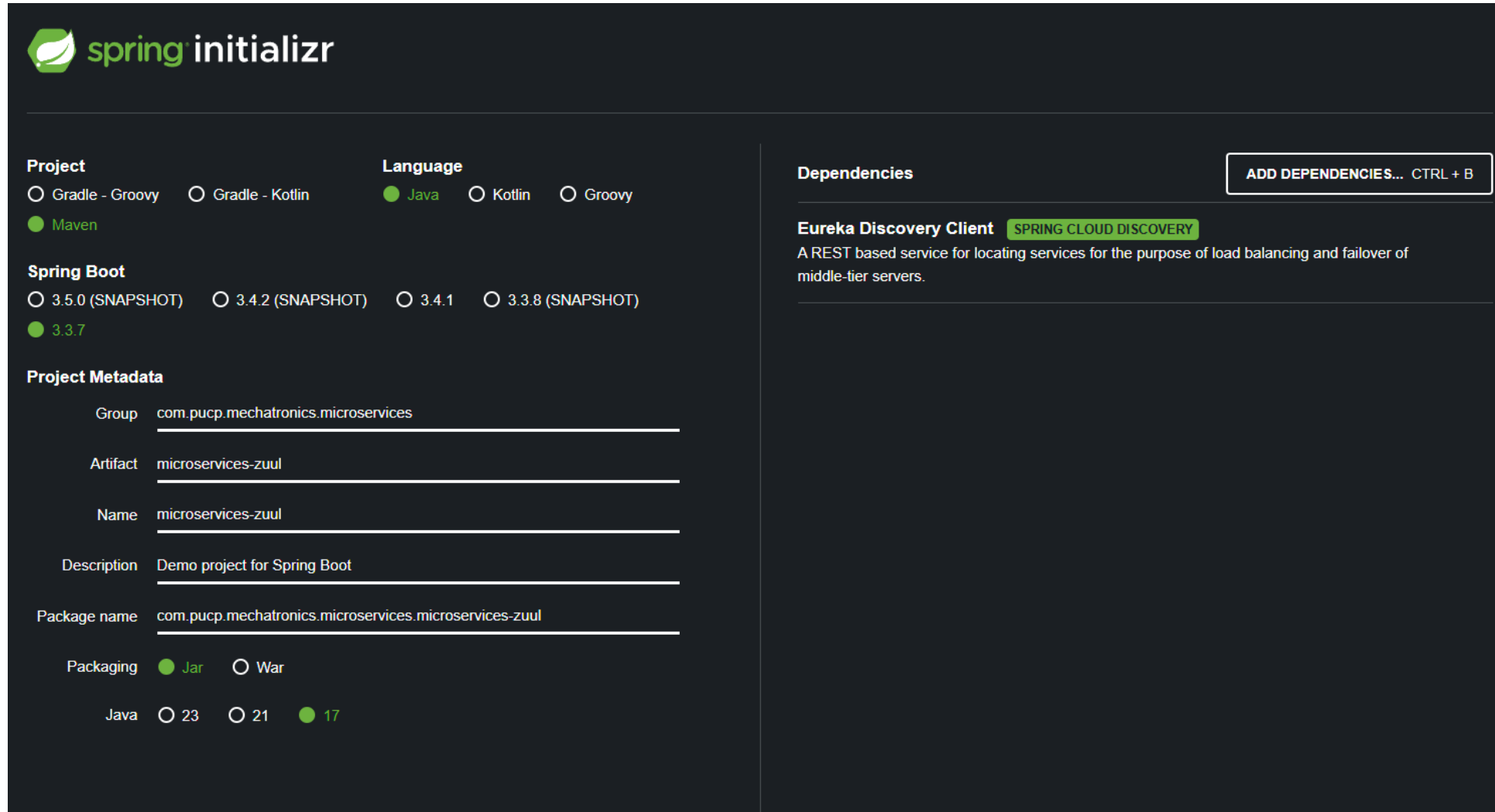
```
1 package com.pucp.mechatronics.microservices.users;
2
3 import org.springframework.boot.SpringApplication;
4
5
6
7 @EnableEurekaClient
8 @SpringBootApplication
9 public class UsersApplication {
10
11     public static void main(String[] args) {
12         SpringApplication.run(UsersApplication.class, args);
13     }
14 }
15
16
```



This screenshot shows the right-hand side of an IDE. On the left, the 'Package Explorer' displays the project structure, showing the 'com.pucp.mechatronics.microservices.users.services' package containing 'UserServiceImpl.java'. On the right, the code editor shows the content of 'UserServiceImpl.java'.

```
1 package com.pucp.mechatronics.microservices.users.services;
2
3 import java.util.Optional;
4
5
6
7 @Service
8 public class UserServiceImpl implements UserService {
9
10     @Autowired UserRepository repository;
11
12     @Override
13     public Iterable<User> findAll() {
14         return repository.findAll();
15     }
16
17     @Override
18     public Optional<User> findById(Long id) {
19         return repository.findById(id);
20     }
21
22     @Override
23     public User save(User user) {
24         return repository.save(user);
25     }
26
27     @Override
28     public void deleteById(Long id) {
29         repository.deleteById(id);
30     }
31 }
32
33
34
35
36
37
```

# Configurando Zuul



The image shows the Spring Initializr web interface for configuring a new project. The interface is dark-themed and includes sections for Project, Language, Spring Boot, Project Metadata, and Dependencies.

**Project**

- ☐ Gradle - Groovy
- ☐ Gradle - Kotlin
- ☒ Maven

**Language**

- ☒ Java
- ☐ Kotlin
- ☐ Groovy

**Spring Boot**

- ☐ 3.5.0 (SNAPSHOT)
- ☐ 3.4.2 (SNAPSHOT)
- ☐ 3.4.1
- ☐ 3.3.8 (SNAPSHOT)
- ☒ 3.3.7

**Project Metadata**

Group:

Artifact:

Name:

Description:

Package name:

Packaging: ☒ Jar ☐ War

Java: ☐ 23 ☐ 21 ☒ 17

**Dependencies**

[ADD DEPENDENCIES... CTRL + B](#)

**Eureka Discovery Client** SPRING CLOUD DISCOVERY

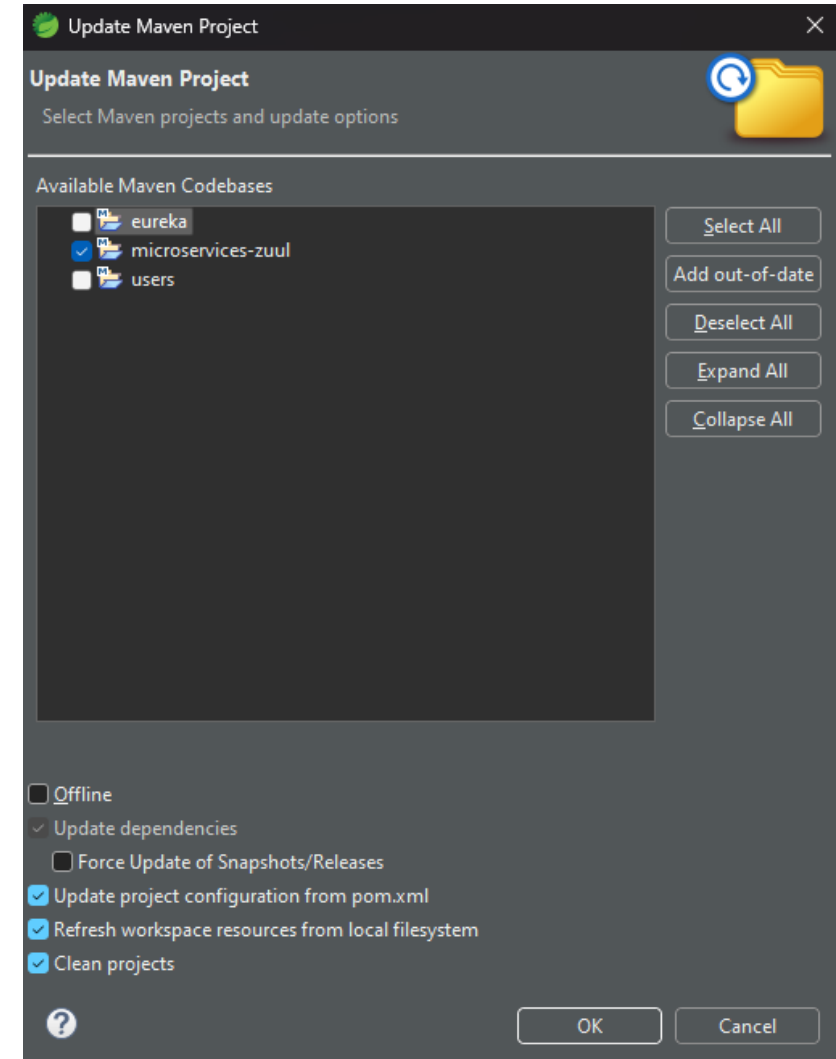
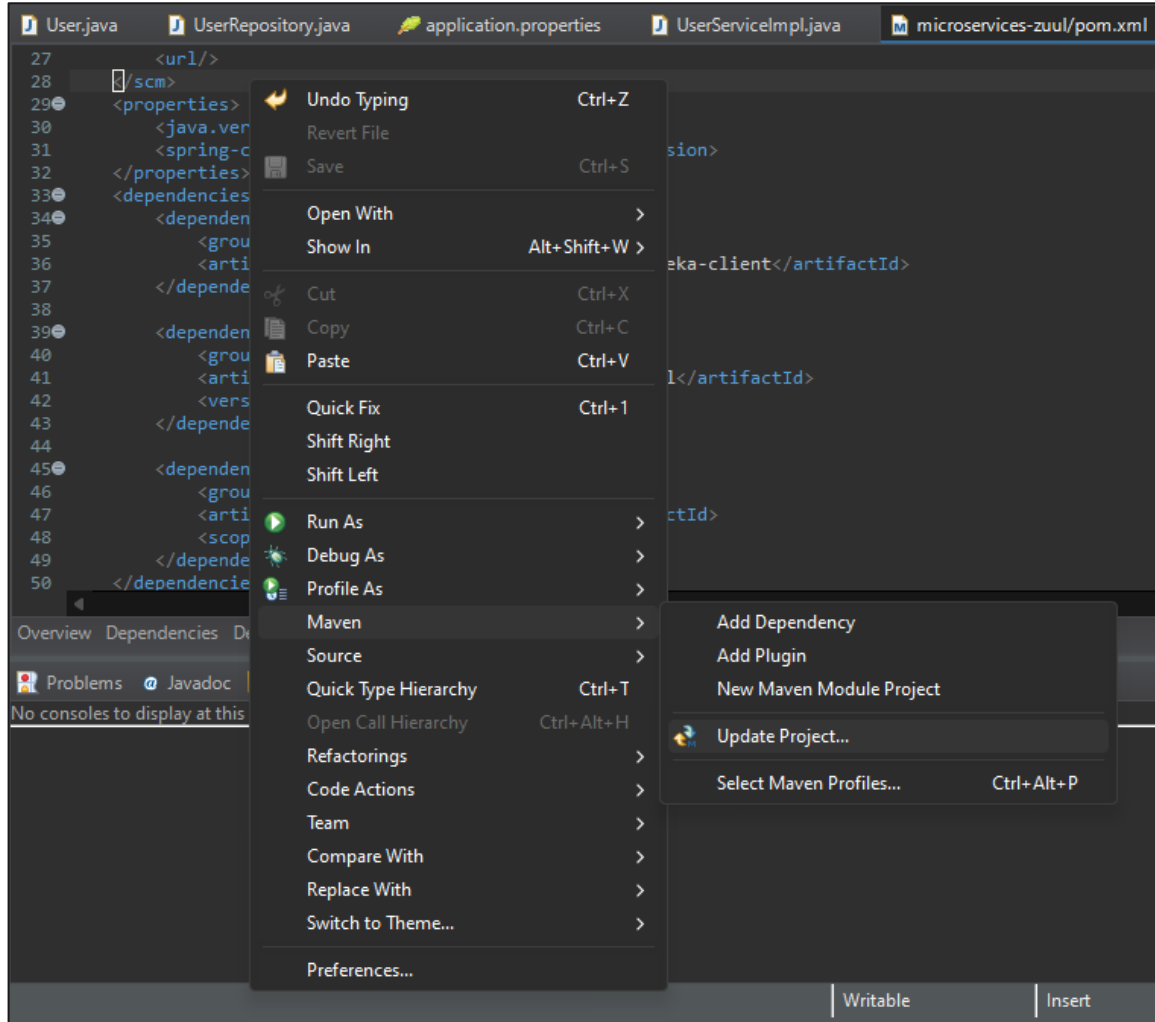
A REST based service for locating services for the purpose of load balancing and failover of middle-tier servers.

# Configurando Zuul

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/
3   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd
4   <modelVersion>4.0.0</modelVersion>
5   <parent>
6     <groupId>org.springframework.boot</groupId>
7     <artifactId>spring-boot-starter-parent</artifactId>
8     <version>2.3.4.RELEASE</version>
9     <relativePath/> <!-- lookup parent from repository -->
10  </parent>
11  <groupId>com.concept.pucp.microservices.app.zuul</groupId>
12  <artifactId>microservices-zuul</artifactId>
13  <version>0.0.1-SNAPSHOT</version>
14  <name>microservices-zuul</name>
15  <description>Zuul Microservices</description>
16  <properties>
17    <java.version>1.8</java.version>
18    <spring-cloud.version>Hoxton.SR9</spring-cloud.version>
19  </properties>
20  <dependencies>
21
22    <dependency>
23      <groupId>org.springframework.cloud</groupId>
24      <artifactId>spring-cloud-starter-netflix-zuul</artifactId>
25      <version>2.2.6.RELEASE</version>
26    </dependency>
27
28    <dependency>
29      <groupId>org.springframework.cloud</groupId>
30      <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
31    </dependency>
32
33    <dependency>
34      <groupId>org.springframework.boot</groupId>
35      <artifactId>spring-boot-starter-test</artifactId>
36      <scope>test</scope>
37  </dependencies>
38 </project>
```

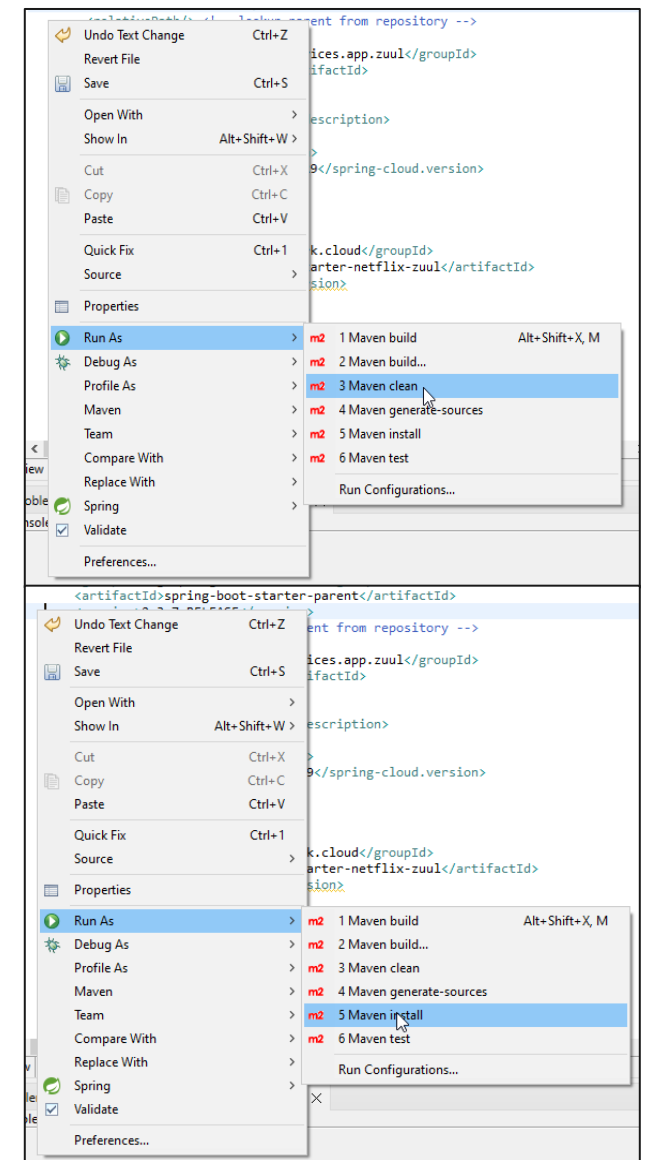
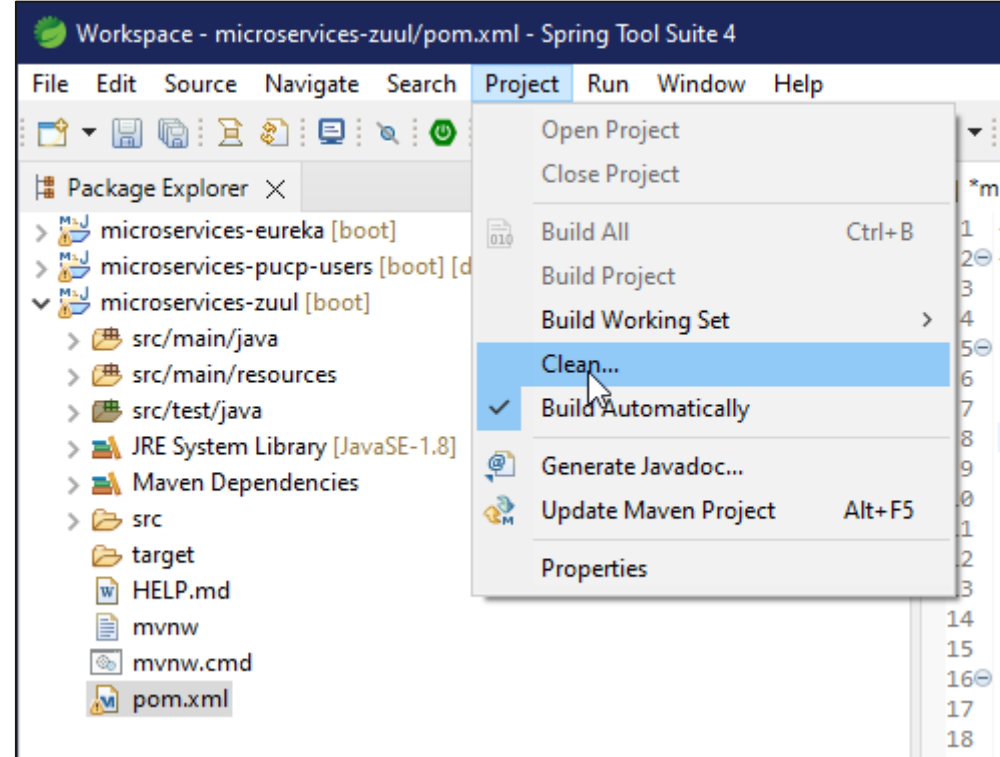
```
<dependency>
<groupId>org.springframework.cloud</groupId>
<artifactId>spring-cloud-starter-netflix-
zuul</artifactId>
<version>2.2.6.RELEASE</version>
</dependency>
```

# Configurando Zuul



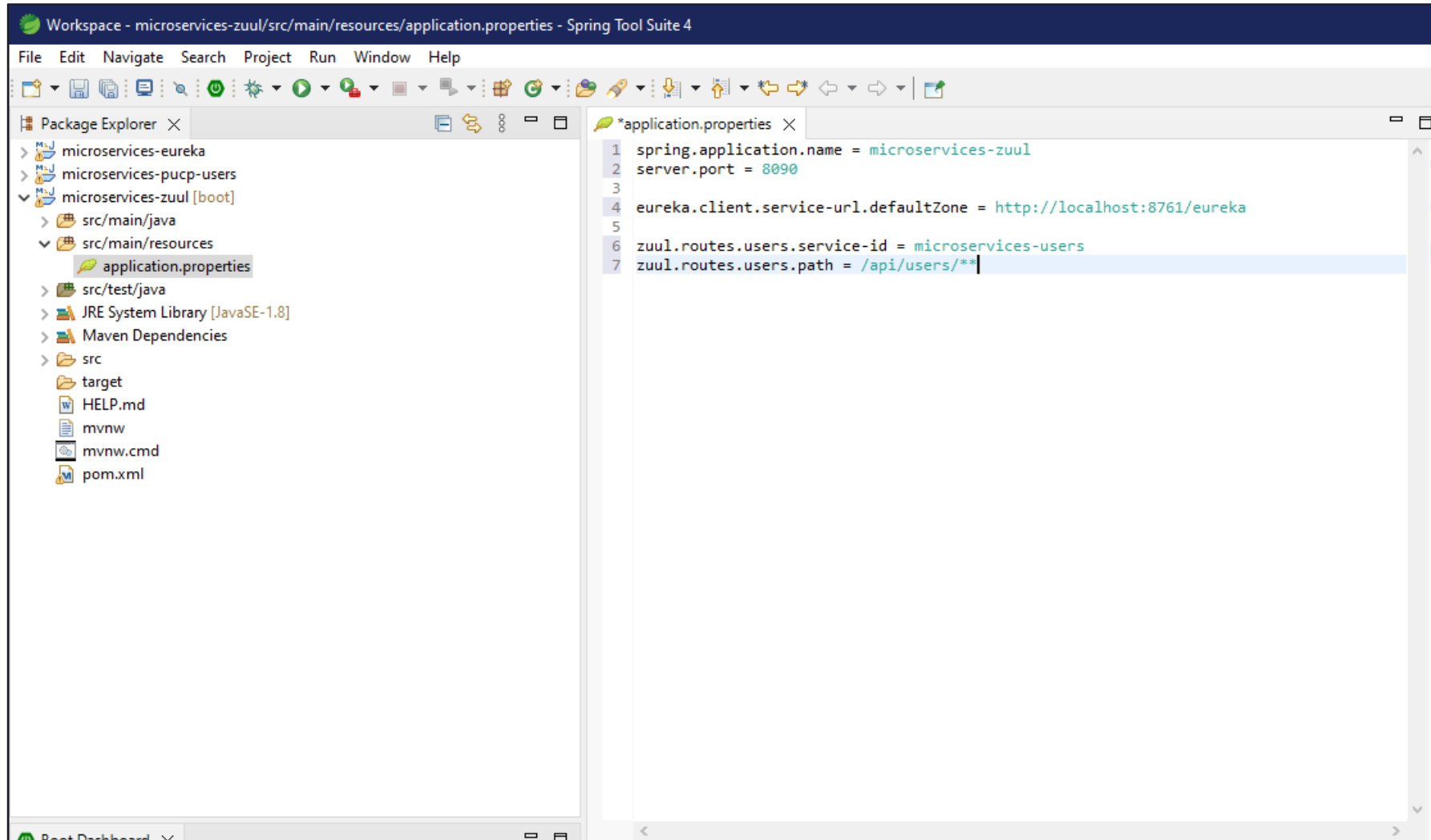
# Solo en caso de que falle

- En caso de que el pom muestre un error que indica que falta eureka-client.jar, limpie el proyecto (Proyecto -> Limpiar...) y reinicie el IDE.
- Si el problema persiste- Clic derecho sobre el proyecto -> Ejecutar como... -> Maven Clean- Click derecho sobre el proyecto -> Ejecutar como... -> Maven Install





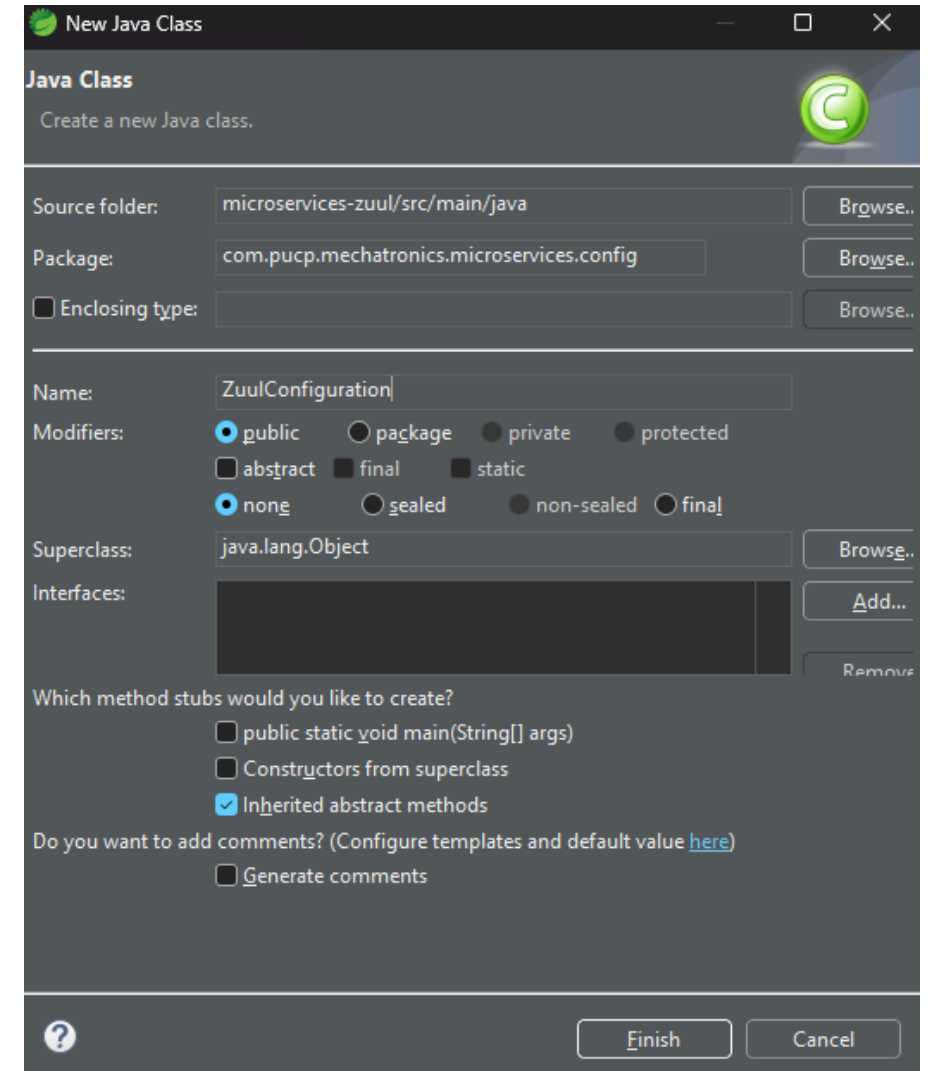
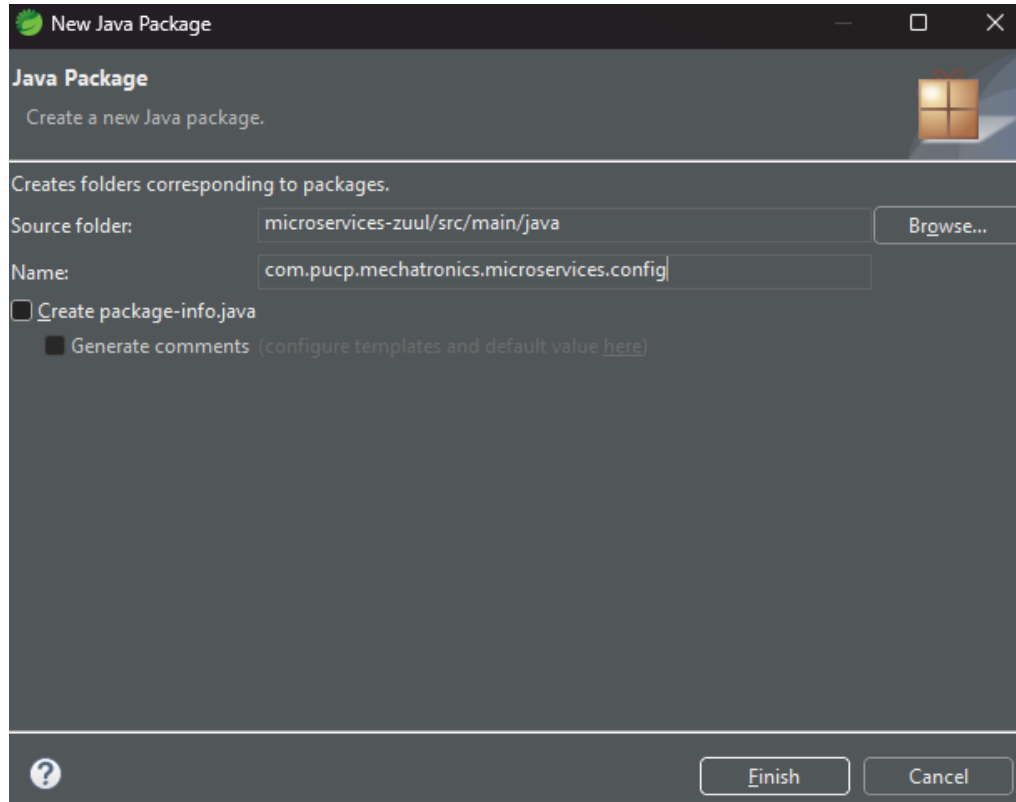
# Propiedades de Zuul



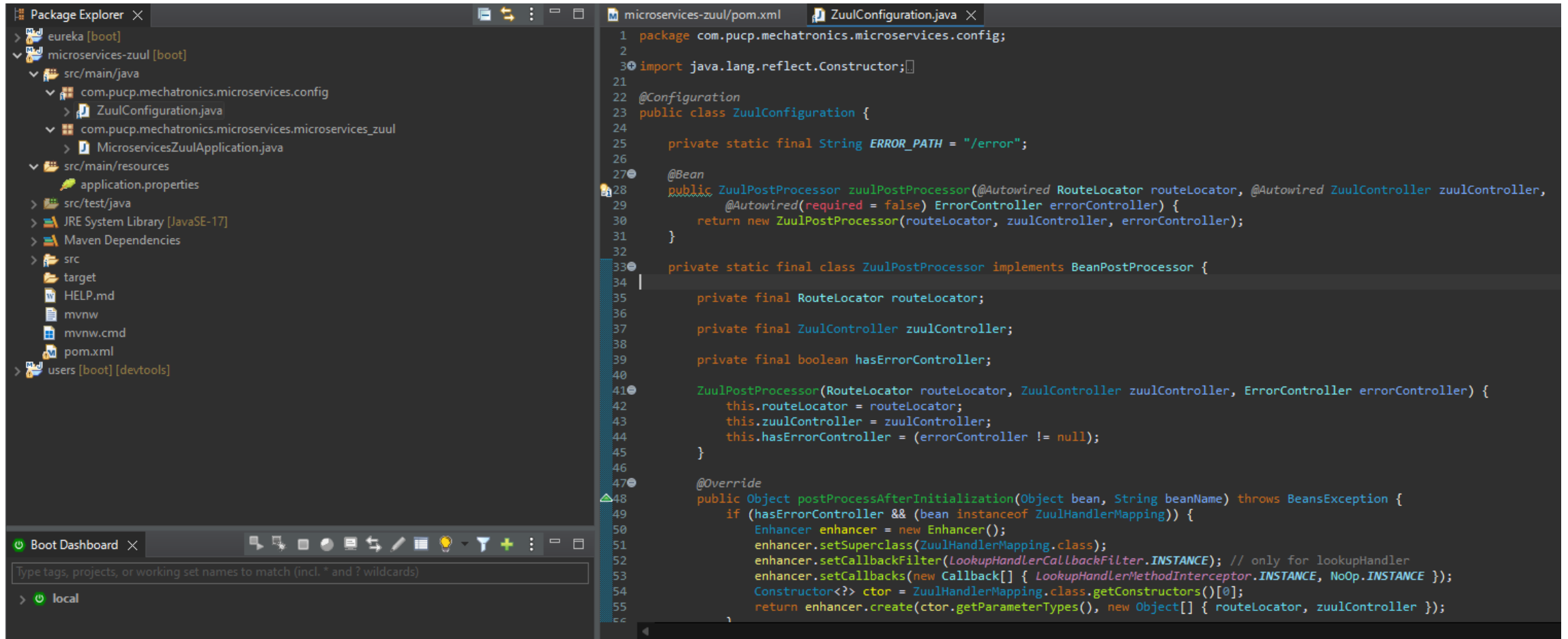
The screenshot shows the Spring Tool Suite 4 IDE. The Package Explorer on the left displays the project structure for 'microservices-zuul', with 'src/main/resources/application.properties' selected. The main editor window shows the content of this file:

```
1 spring.application.name = microservices-zuul
2 server.port = 8090
3
4 eureka.client.service-url.defaultZone = http://localhost:8761/eureka
5
6 zuul.routes.users.service-id = microservices-users
7 zuul.routes.users.path = /api/users/**
```

# Adicionando Clases en Zuul



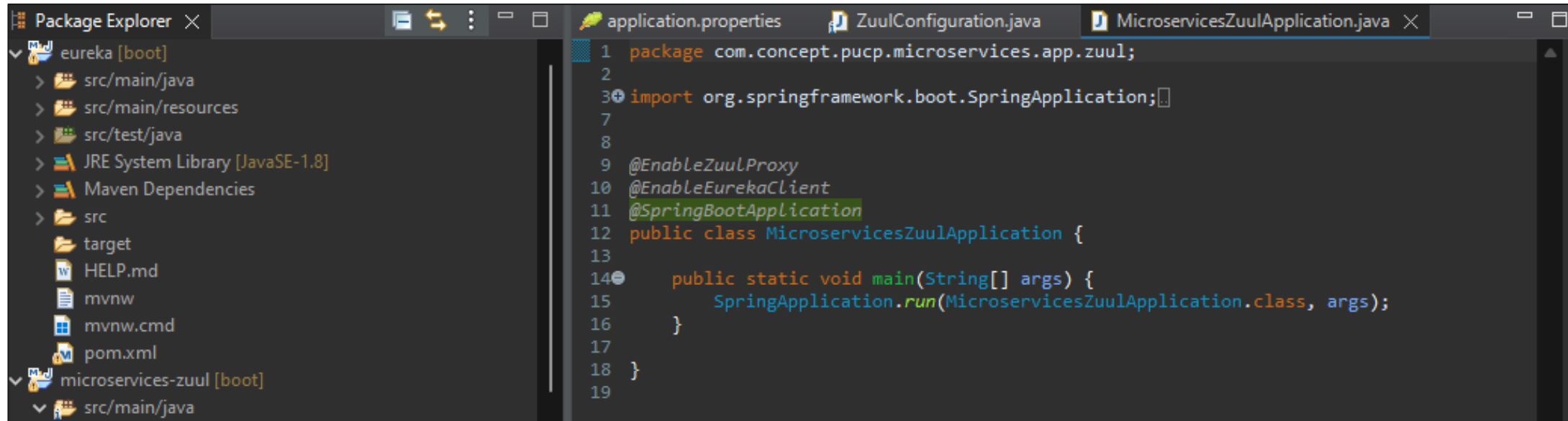
# Añadiendo Clases en Zuul



The screenshot shows an IDE with the Package Explorer on the left and the editor on the right. The Package Explorer displays the project structure for 'microservices-zuul'. The editor shows the 'ZuulConfiguration.java' file, which contains the following code:

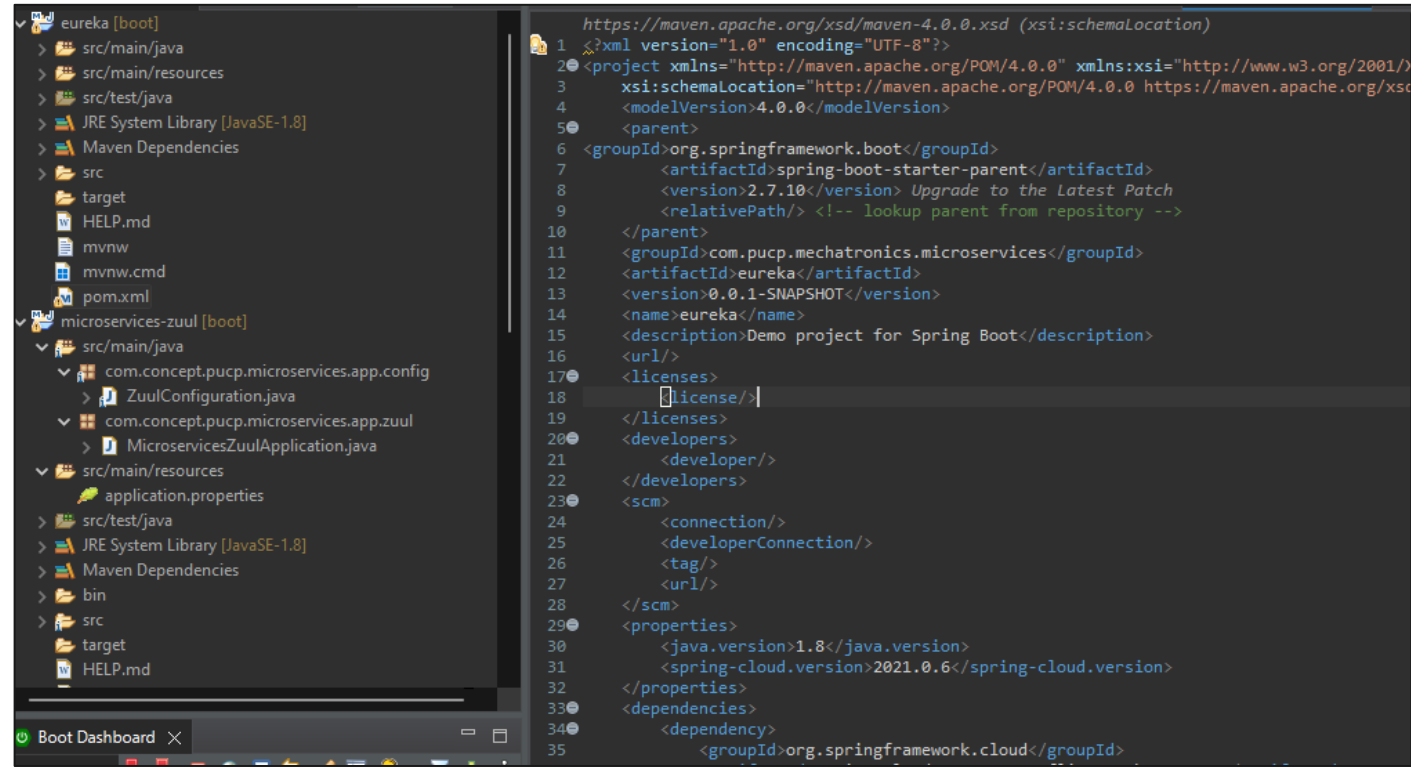
```
1 package com.pucp.mechatronics.microservices.config;
2
3 import java.lang.reflect.Constructor;
4
21
22 @Configuration
23 public class ZuulConfiguration {
24
25     private static final String ERROR_PATH = "/error";
26
27     @Bean
28     public ZuulPostProcessor zuulPostProcessor(@Autowired RouteLocator routeLocator, @Autowired ZuulController zuulController,
29         @Autowired(required = false) ErrorController errorController) {
30         return new ZuulPostProcessor(routeLocator, zuulController, errorController);
31     }
32
33     private static final class ZuulPostProcessor implements BeanPostProcessor {
34
35         private final RouteLocator routeLocator;
36
37         private final ZuulController zuulController;
38
39         private final boolean hasErrorController;
40
41         ZuulPostProcessor(RouteLocator routeLocator, ZuulController zuulController, ErrorController errorController) {
42             this.routeLocator = routeLocator;
43             this.zuulController = zuulController;
44             this.hasErrorController = (errorController != null);
45         }
46
47         @Override
48         public Object postProcessAfterInitialization(Object bean, String beanName) throws BeansException {
49             if (hasErrorController && (bean instanceof ZuulHandlerMapping)) {
50                 Enhancer enhancer = new Enhancer();
51                 enhancer.setSuperclass(ZuulHandlerMapping.class);
52                 enhancer.setCallbackFilter(LookupHandlerCallbackFilter.INSTANCE); // only for lookupHandler
53                 enhancer.setCallbacks(new Callback[] { LookupHandlerMethodInterceptor.INSTANCE, NoOp.INSTANCE });
54                 Constructor<?> ctor = ZuulHandlerMapping.class.getConstructors()[0];
55                 return enhancer.create(ctor.getParameterTypes(), new Object[] { routeLocator, zuulController });
56             }
57         }
58     }
59 }
```

# Configuración adicional en SpringApplication



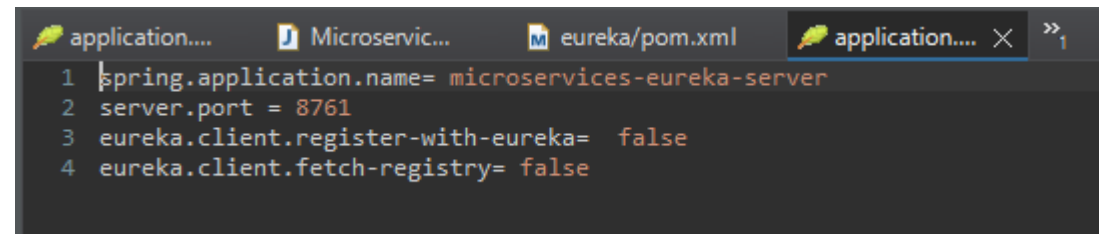
```
1 package com.concept.pucp.microservices.app.zuul;
2
3 import org.springframework.boot.SpringApplication;
4
5
6
7
8
9 @EnableZuulProxy
10 @EnableEurekaClient
11 @SpringBootApplication
12 public class MicroservicesZuulApplication {
13
14     public static void main(String[] args) {
15         SpringApplication.run(MicroservicesZuulApplication.class, args);
16     }
17
18 }
19
```

# Modificaciones en Eureka



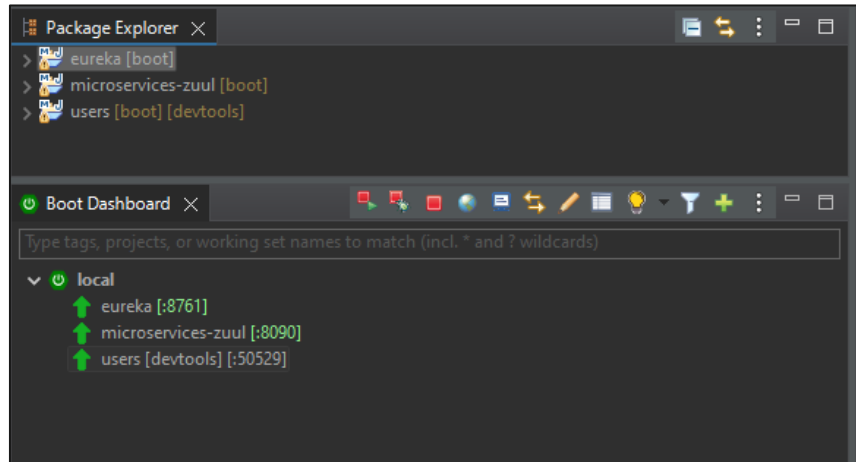
```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/
3   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd
4   <modelVersion>4.0.0</modelVersion>
5   <parent>
6     <groupId>org.springframework.boot</groupId>
7     <artifactId>spring-boot-starter-parent</artifactId>
8     <version>2.7.10</version> Upgrade to the Latest Patch
9     <relativePath/> <!-- lookup parent from repository -->
10  </parent>
11  <groupId>com.pucp.mechatronics.microservices</groupId>
12  <artifactId>eureka</artifactId>
13  <version>0.0.1-SNAPSHOT</version>
14  <name>eureka</name>
15  <description>Demo project for Spring Boot</description>
16  <url/>
17  <licenses>
18    <license/>
19  </licenses>
20  <developers>
21    <developer/>
22  </developers>
23  <scm>
24    <connection/>
25    <developerConnection/>
26    <tag/>
27    <url/>
28  </scm>
29  <properties>
30    <java.version>1.8</java.version>
31    <spring-cloud.version>2021.0.6</spring-cloud.version>
32  </properties>
33  <dependencies>
34    <dependency>
35      <groupId>org.springframework.cloud</groupId>
```

```
1 package com.pucp.mechatronics.microservices.eureka;
2
3 import org.springframework.boot.SpringApplication;
4
5
6
7 @EnableEurekaServer
8 @SpringBootApplication
9 public class EurekaApplication {
10
11     public static void main(String[] args) {
12         SpringApplication.run(EurekaApplication.class, args);
13     }
14
15 }
16
```



```
1 spring.application.name= microservices-eureka-server
2 server.port = 8761
3 eureka.client.register-with-eureka= false
4 eureka.client.fetch-registry= false
```

# Probando nuestro proyecto



The screenshot shows the Spring Eureka web interface in a browser. The page has a dark header with the 'spring Eureka' logo and a 'Toggle navigation' button. The main content area is divided into three sections: 'System Status', 'DS Replicas', and 'General Info'.

### System Status

Environment	test	Current time	2022-09-06T23:50:39 -0500
Data center	default	Uptime	00:01
		Lease expiration enabled	false
		Renews threshold	5
		Renews (last min)	2

### DS Replicas

[localhost](#)

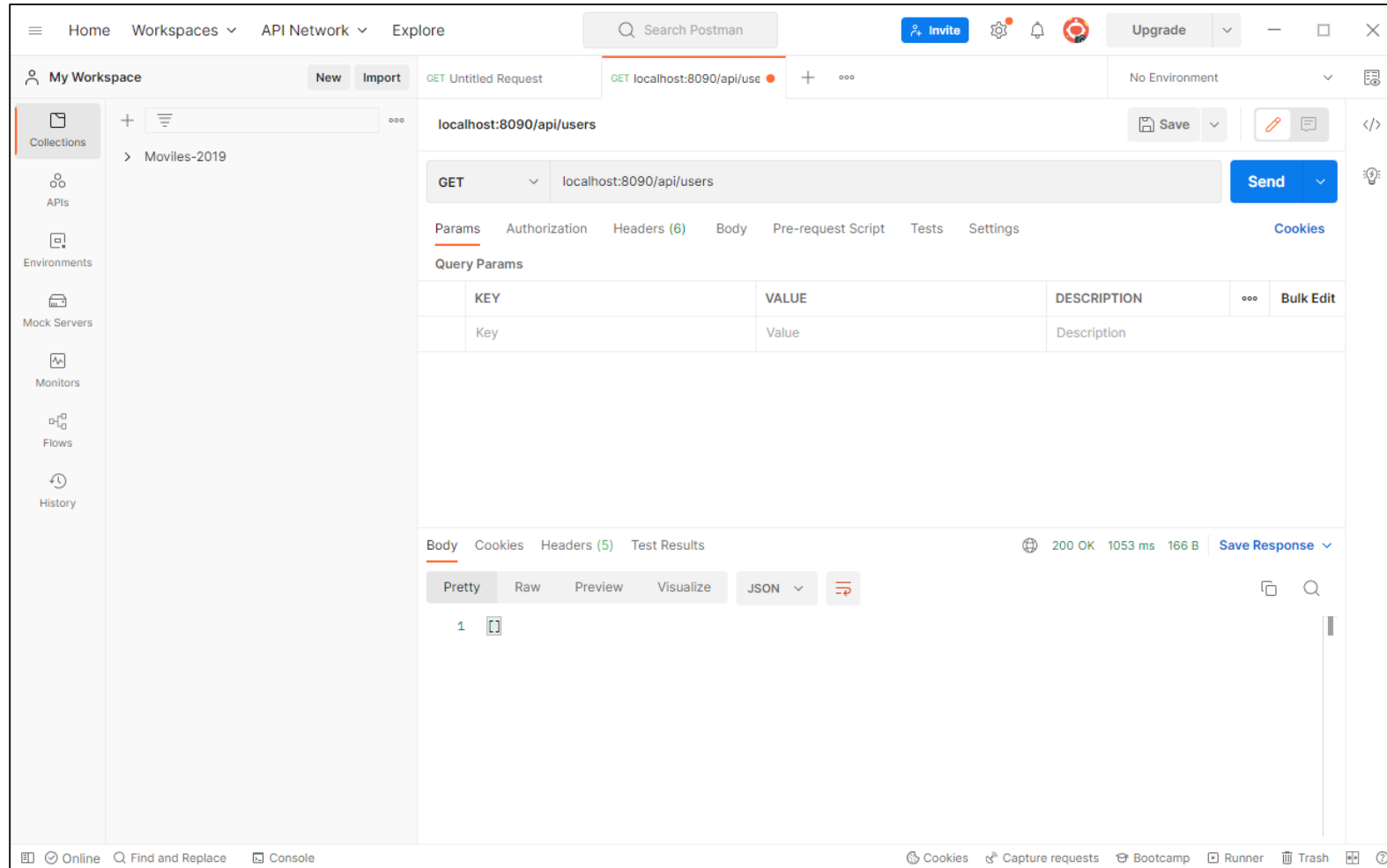
### Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
MICROSERVICES-USERS	n/a (1)	(1)	UP (1) - <a href="#">microservices-users.d957405f482228a6c5bb54ad92810eed</a>
MICROSERVICES-ZUUL	n/a (1)	(1)	UP (1) - <a href="#">DESKTOP-M7HUQSV:microservices-zuul:8090</a>

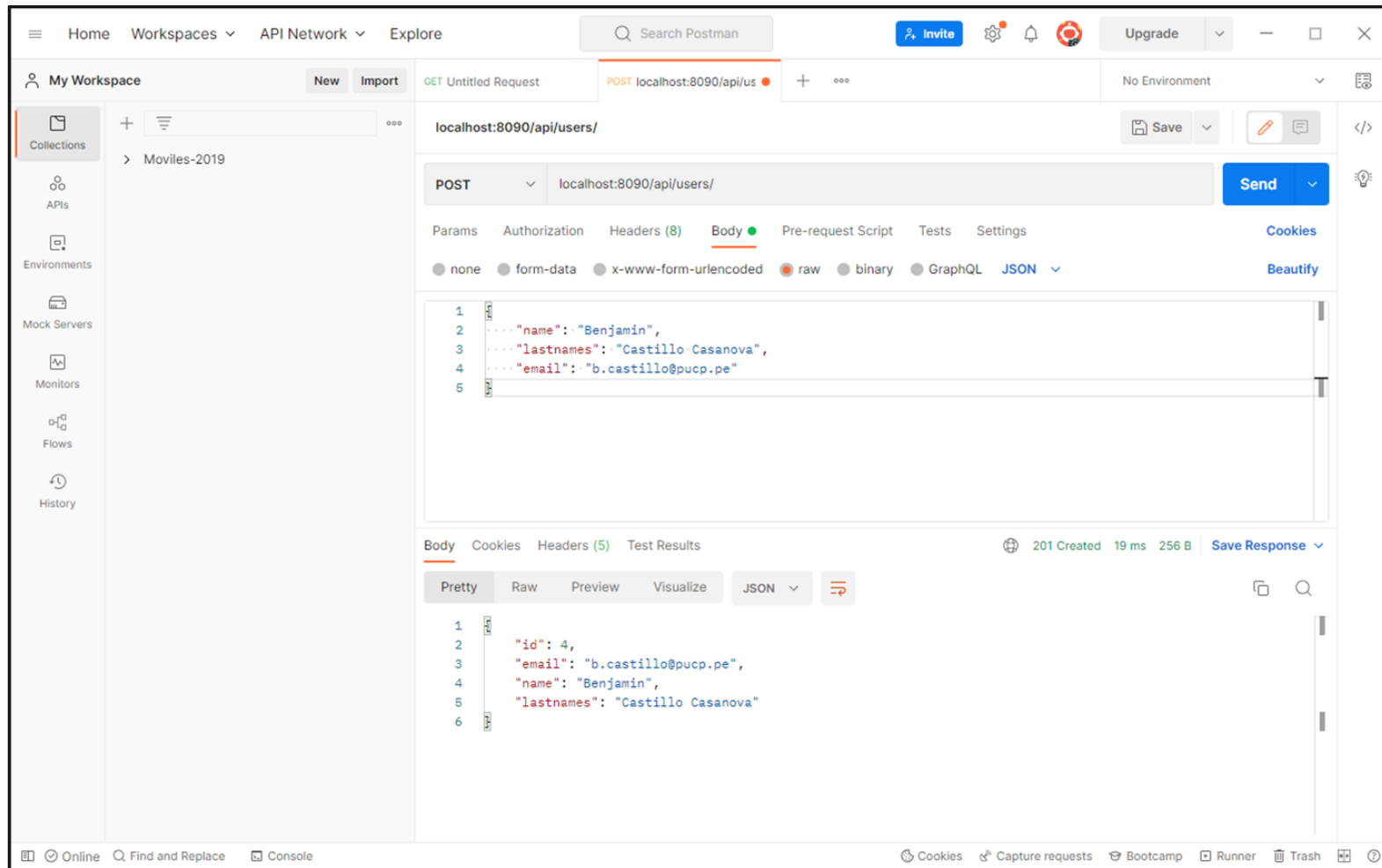
### General Info

Name	Value
total-avail-memory	136mb
num-of-cpus	8
current-memory-usage	45mb (33%)
server-uptime	00:01

# Probando nuestro proyecto

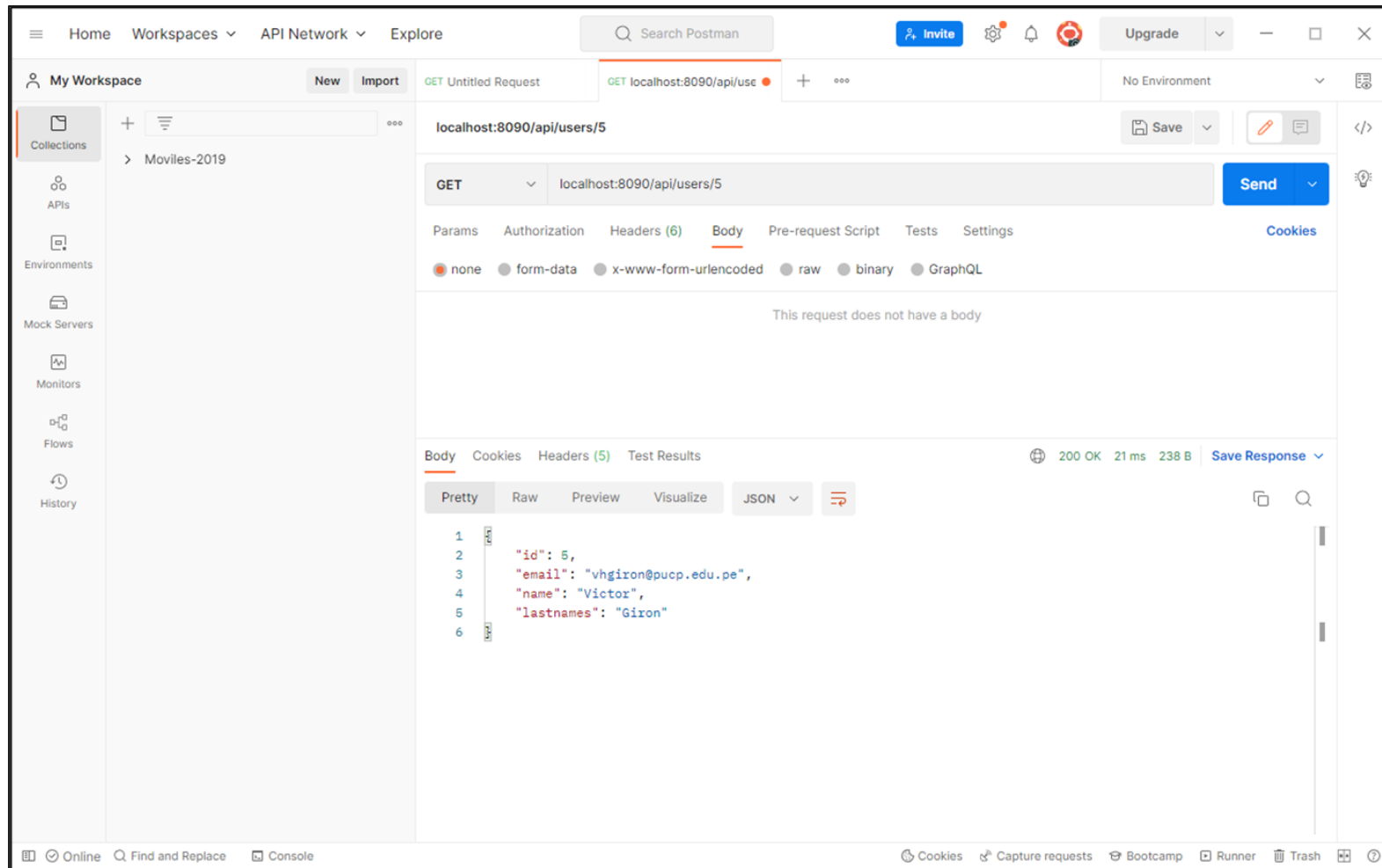


# Probando nuestro proyecto

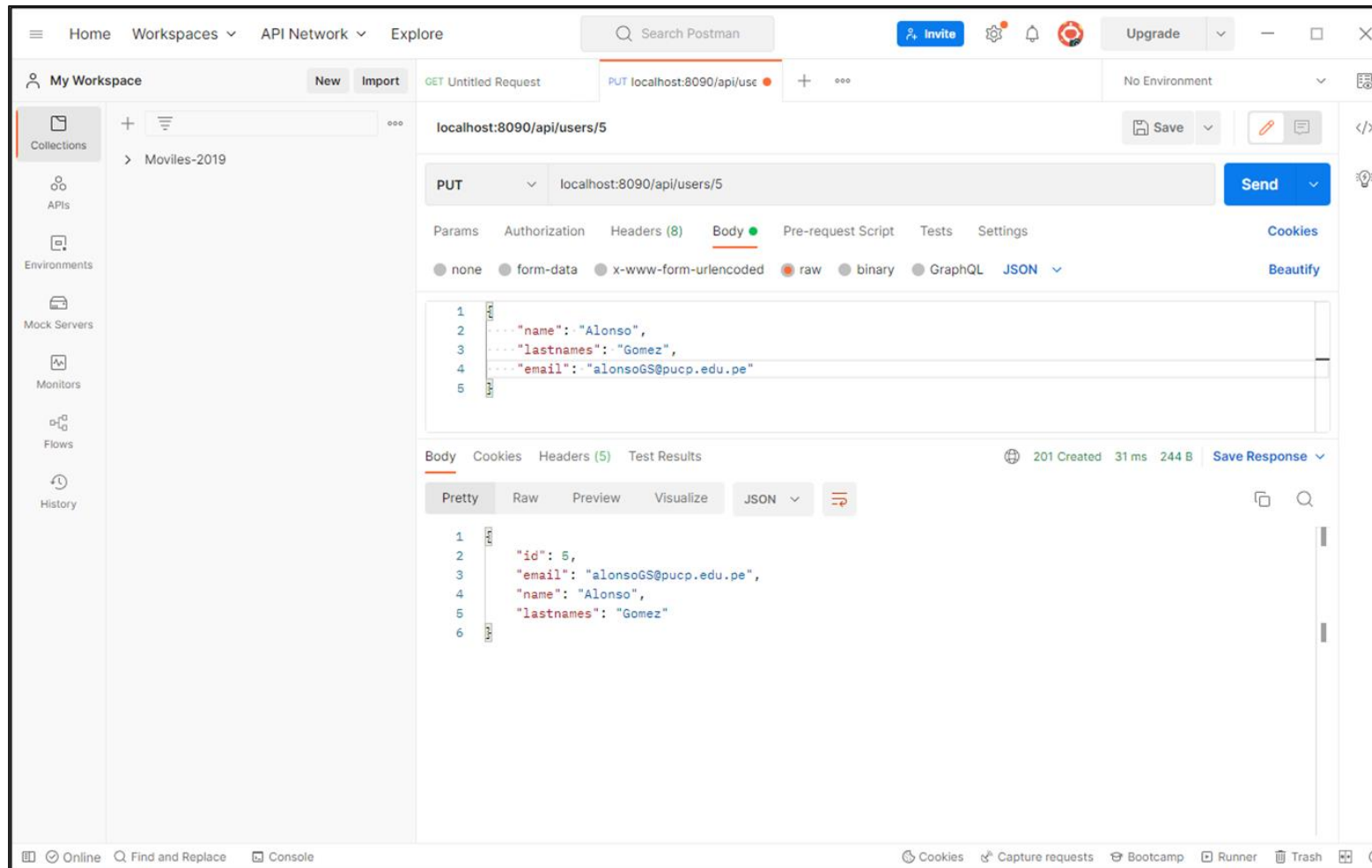




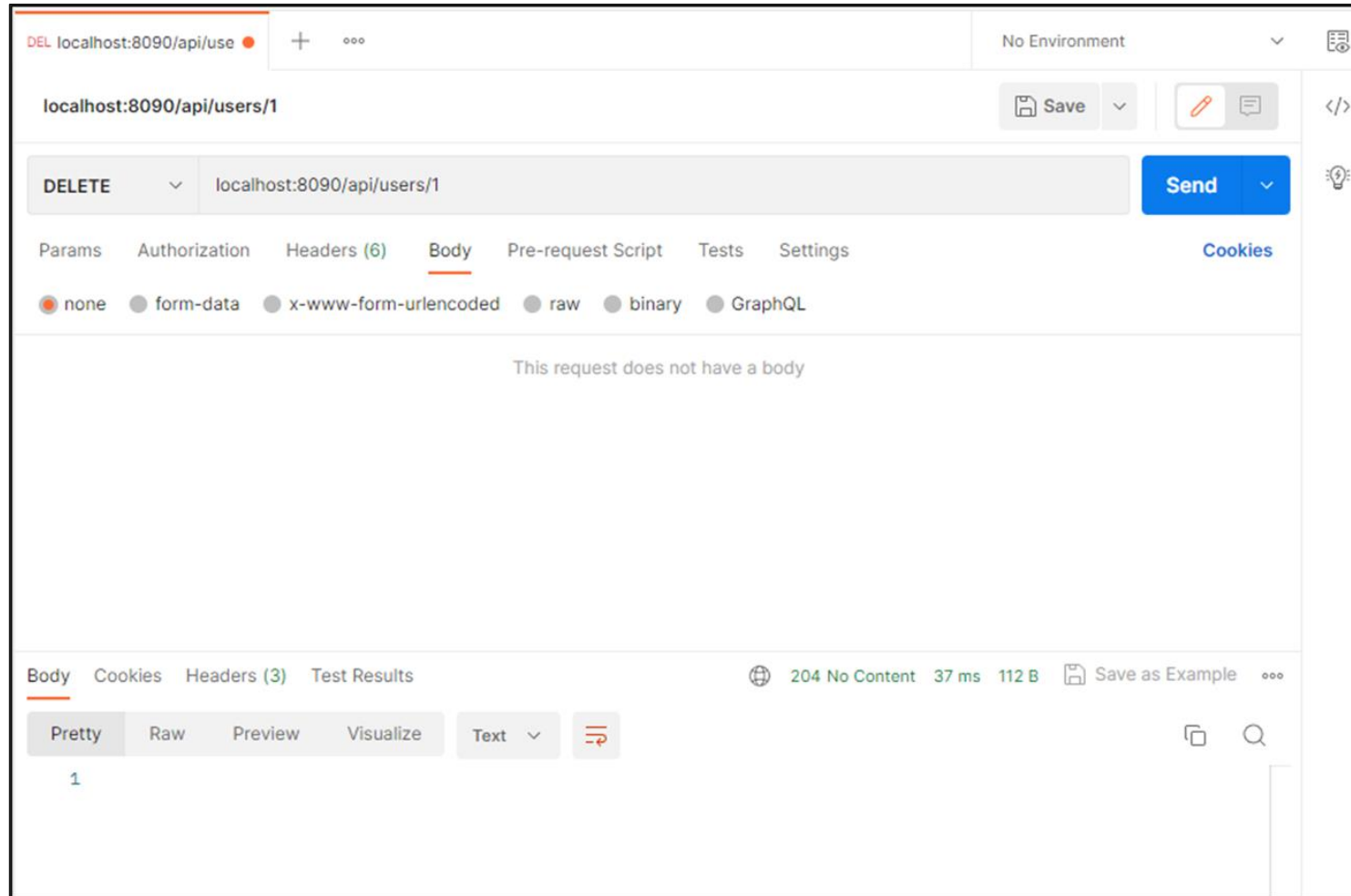
# Probando nuestro proyecto



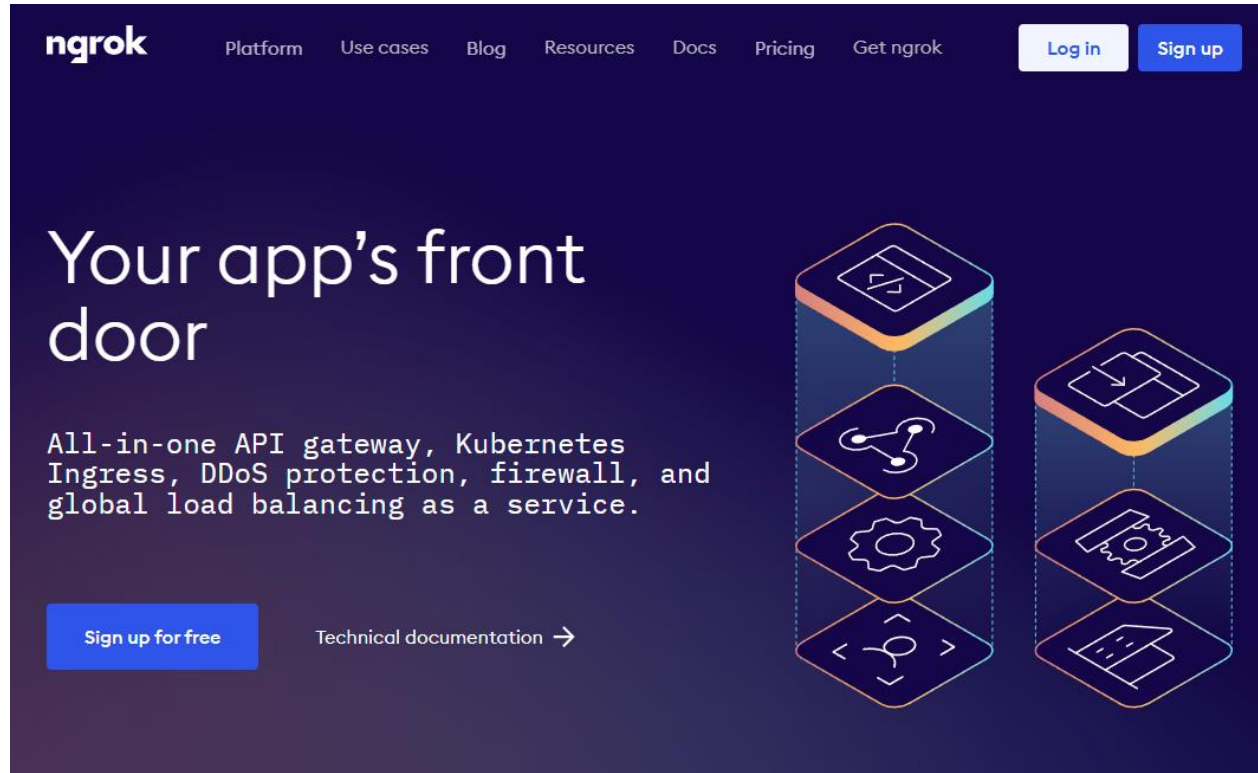
# Probando nuestro proyecto



# Probando nuestro proyecto



# Accediendo desde un ambiente externo



The image shows the Ngrok website homepage. At the top, there is a navigation bar with links for Platform, Use cases, Blog, Resources, Docs, Pricing, and Get ngrok. On the right side of the navigation bar are buttons for Log in and Sign up. The main heading is "Your app's front door". Below it, the text reads: "All-in-one API gateway, Kubernetes Ingress, DDoS protection, firewall, and global load balancing as a service." There is a "Sign up for free" button and a link to "Technical documentation" with a right arrow. On the right side, there is a vertical stack of four hexagonal icons representing different features: a server, a network, a gear, and a shield.

ngrok

Platform Use cases Blog Resources Docs Pricing Get ngrok

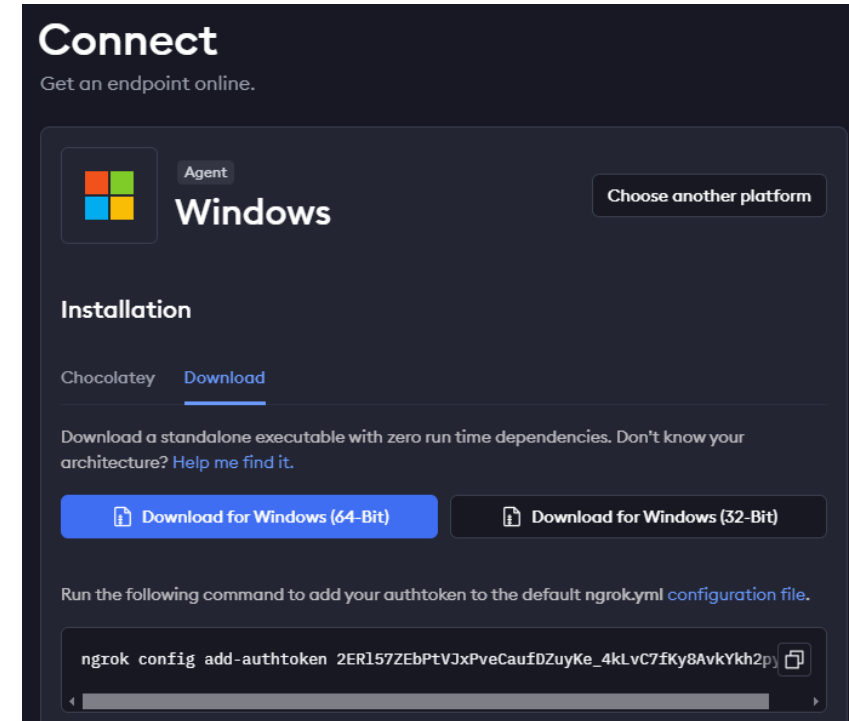
Log in Sign up

## Your app's front door

All-in-one API gateway, Kubernetes Ingress, DDoS protection, firewall, and global load balancing as a service.

Sign up for free

Technical documentation →



The image shows the Ngrok Connect page for Windows. The heading is "Connect" with the subtext "Get an endpoint online." Below this, there is a section for "Agent" with the Windows logo and the text "Windows". There is a button "Choose another platform". Under the "Installation" section, there are two tabs: "Chocolatey" and "Download". The "Download" tab is active. Below the tabs, there is a text block: "Download a standalone executable with zero run time dependencies. Don't know your architecture? Help me find it." There are two buttons: "Download for Windows (64-Bit)" and "Download for Windows (32-Bit)". Below these buttons, there is a text block: "Run the following command to add your authtoken to the default ngrok.yml configuration file." There is a code block with the command: `ngrok config add-authtoken 2ER157ZEBtVJxPveCauFDZuyKe_4kLVC7fKy8AvkYkh2p`. There is a copy icon next to the command.

## Connect

Get an endpoint online.

Agent

Windows

Choose another platform

### Installation

Chocolatey Download

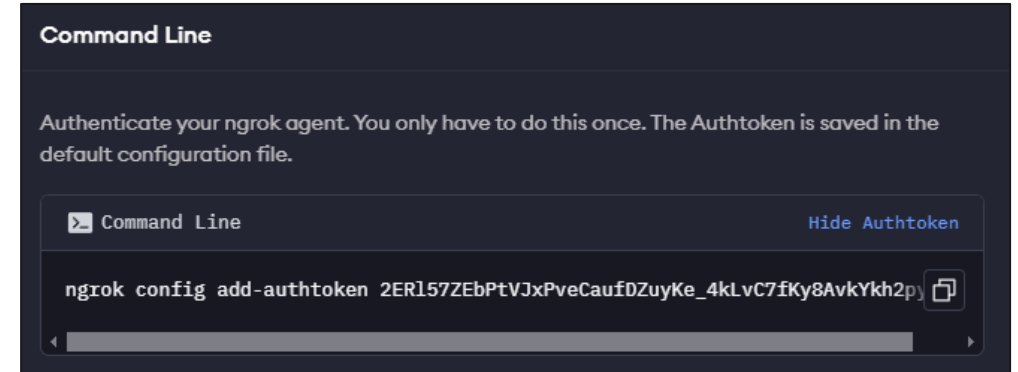
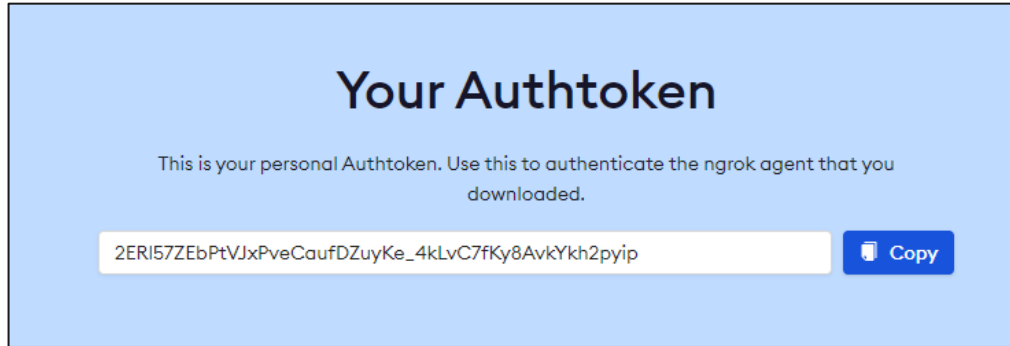
Download a standalone executable with zero run time dependencies. Don't know your architecture? Help me find it.

Download for Windows (64-Bit) Download for Windows (32-Bit)

Run the following command to add your authtoken to the default ngrok.yml configuration file.

```
ngrok config add-authtoken 2ER157ZEBtVJxPveCauFDZuyKe_4kLVC7fKy8AvkYkh2p
```

# Accediendo desde un ambiente externo



# Accediendo desde un ambiente externo

Port Forwarding

```
$ ngrok http <port>
```



```
C:\Users\Benjamin\Desktop\ngrok.exe - ngrok http 8090
ngrok
Join us in the ngrok community @ https://ngrok.com/slack
Session Status      online
Account             b.castillo@pucp.pe (Plan: Free)
Version             3.0.7
Region              South America (sa)
Latency              -
Web Interface       http://127.0.0.1:4040
Forwarding           https://c8a9-38-25-18-224.sa.ngrok.io -> http://localhost:8090
Connections
  ttl    opn    rt1    rt5    p50    p90
   0      0     0.00  0.00  0.00  0.00
```

# Accediendo desde un ambiente externo

