



## **CORPORACION UNIVERSITARIA REMINGTON**

Asignatura: Lenguaje de programación avanzado 2  
Parcial 2

Presentado por:  
Gloria Patricia Cardona Cuervo

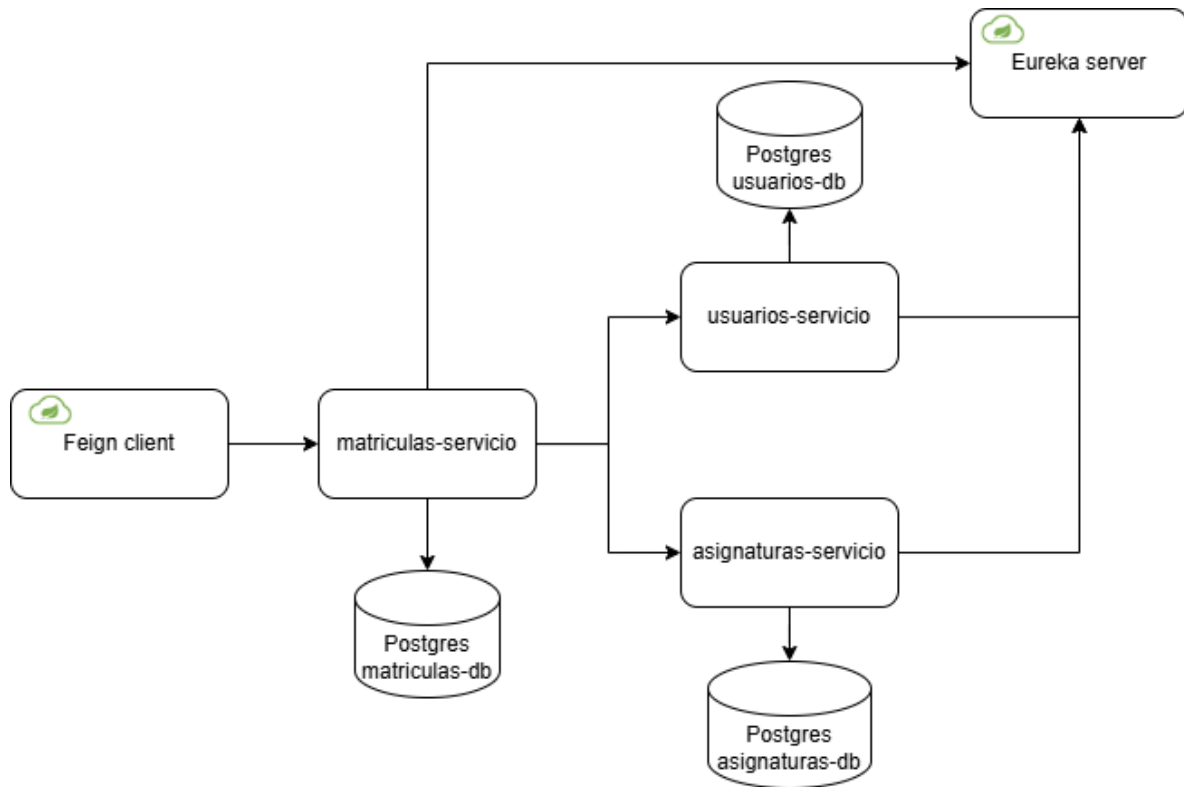
Presentado a:  
Profesora Martha Nicolasa Amaya Becerra

Ingeniería de Sistemas - Virtual  
abril de 2025

## Contenido

Arquitectura del proyecto .....	3
Despliegue de contenedor PostgreSQL en Windows.....	4
Microservicio para Eureka Server .....	8
Microservicio de Usuarios .....	9
Microservicio de Asignaturas.....	13
Microservicio de Matrículas .....	19

## Arquitectura del proyecto

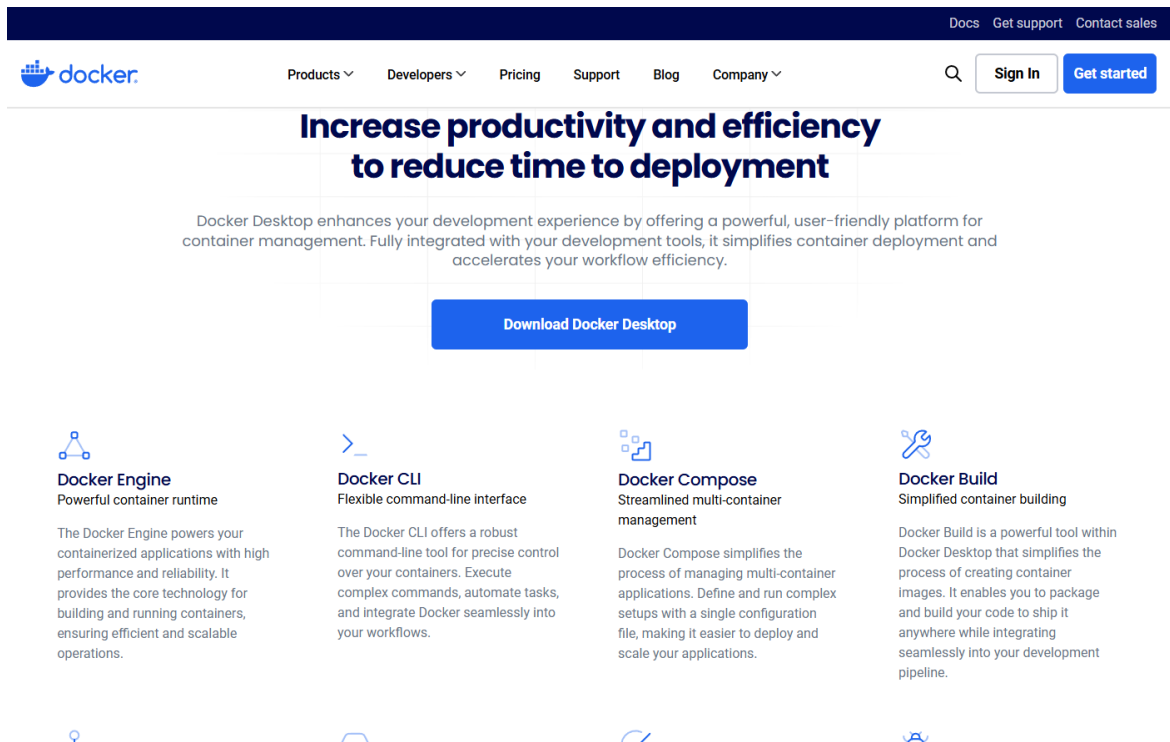


# Despliegue de contenedor PostgreSQL en Windows

Para desplegar un contenedor de PostgreSQL en un entorno Windows utilizando Docker, se llevaron a cabo los siguientes pasos:

## Instalación de Docker Desktop

Se accedió al sitio oficial de Docker (<https://www.docker.com/products/docker-desktop>) y se descargó Docker Desktop para Windows. Posteriormente, se realizó la instalación y se verificó que el servicio estuviera activo mediante el icono de la ballena visible en la barra de tareas.



The screenshot shows the Docker website homepage. At the top, there is a dark blue navigation bar with links for 'Docs', 'Get support', and 'Contact sales'. Below this is a white header with the Docker logo, navigation links for 'Products', 'Developers', 'Pricing', 'Support', 'Blog', and 'Company', a search icon, and 'Sign In' and 'Get started' buttons. The main content area features a large heading 'Increase productivity and efficiency to reduce time to deployment' with a subtext describing Docker Desktop. A prominent blue button labeled 'Download Docker Desktop' is centered below the text. At the bottom, there are four columns, each with an icon and a title: 'Docker Engine' (Powerful container runtime), 'Docker CLI' (Flexible command-line interface), 'Docker Compose' (Streamlined multi-container management), and 'Docker Build' (Simplified container building). Each column contains a brief description of the tool's capabilities.


Docs Get support Contact sales

docker Products Developers Pricing Support Blog Company Sign In Get started

## Increase productivity and efficiency to reduce time to deployment

Docker Desktop enhances your development experience by offering a powerful, user-friendly platform for container management. Fully integrated with your development tools, it simplifies container deployment and accelerates your workflow efficiency.


**Download Docker Desktop**



### Docker Engine

Powerful container runtime


The Docker Engine powers your containerized applications with high performance and reliability. It provides the core technology for building and running containers, ensuring efficient and scalable operations.



### Docker CLI

Flexible command-line interface


The Docker CLI offers a robust command-line tool for precise control over your containers. Execute complex commands, automate tasks, and integrate Docker seamlessly into your workflows.



### Docker Compose

Streamlined multi-container management

Docker Compose simplifies the process of managing multi-container applications. Define and run complex setups with a single configuration file, making it easier to deploy and scale your applications.



### Docker Build

Simplified container building

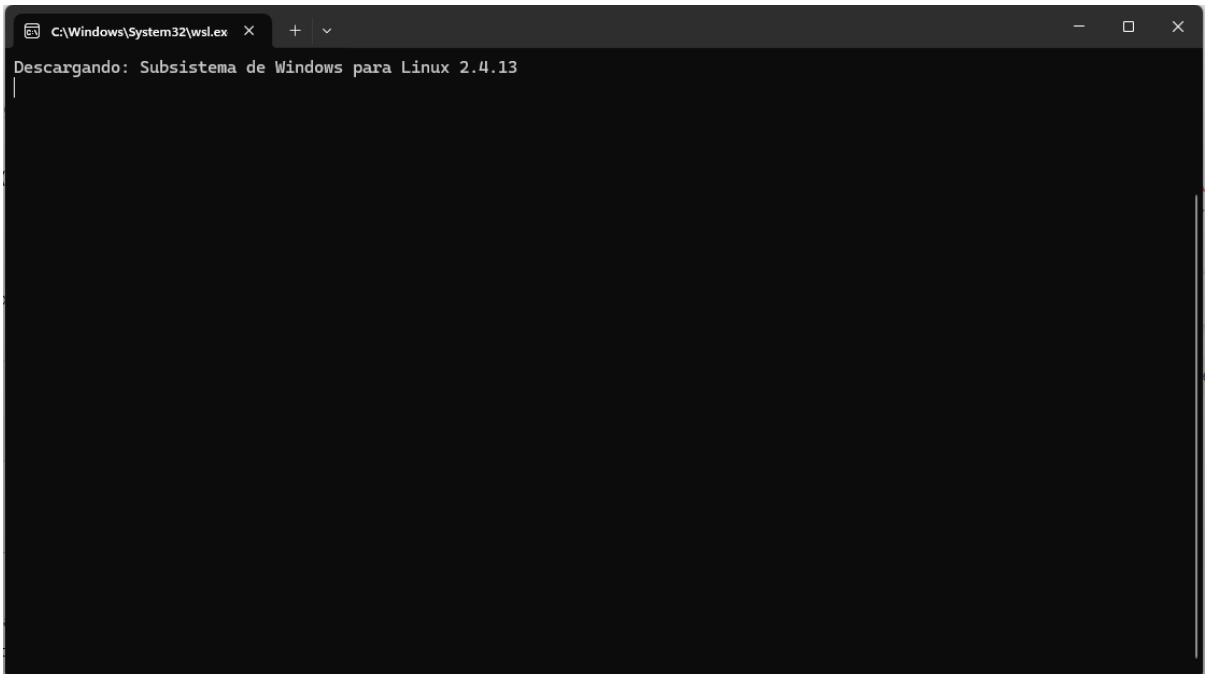
Docker Build is a powerful tool within Docker Desktop that simplifies the process of creating container images. It enables you to package and build your code to ship it anywhere while integrating seamlessly into your development pipeline.

## Docker Desktop 4.40.0

### Unpacking files...

```
Unpacking file: resources/cli-plugins/docker-compose.exe
Unpacking file: resources/cli-plugins/docker-cloud.exe
Unpacking file: resources/cli-plugins/docker-buildx.exe
Unpacking file: resources/cli-plugins/docker-ai.exe
Unpacking file: resources/bin/kubectl.exe
Unpacking file: resources/bin/hub-tool.exe
Unpacking file: resources/bin/extension-admin.exe
Unpacking file: resources/bin/docker.exe
Unpacking file: resources/bin/docker-credential-wincred.exe
Unpacking file: resources/bin/docker-credential-ecr-login.exe
Unpacking file: resources/bin/docker-credential-desktop.exe
Unpacking file: resources/bin/compose-bridge.exe
Unpacking file: Owin.dll
Unpacking file: NLog.dll
Unpacking file: Newtonsoft.Json.dll
Unpacking file: Newtonsoft.Json.Bson.dll
Unpacking file: netstandard.dll
Unpacking file: Microsoft.Windows.ComputeVirtualization.dll
Unpacking file: Microsoft.Win32.Primitives.dll
Unpacking file: Microsoft.Owin.Hosting.dll
Unpacking file: Microsoft.Owin.dll
Unpacking file: microsoft.management.infrastructure.native.unmanaged.dll
Unpacking file: microsoft.management.infrastructure.native.dll
Unpacking file: microsoft.management.infrastructure.dll
Unpacking file: InstallerCli.exe
Unpacking file: HttpOverStream.Server.Owin.dll
Unpacking file: HttpOverStream.NamedPipe.dll
Unpacking file: HttpOverStream.dll
Unpacking file: HttpOverStream.Client.dll
Unpacking file: frontend/vulkan-1.dll
Unpacking file: frontend/vk_swiftshader.dll
Unpacking file: frontend/resources/app.asar.unpacked/build/Release/winpty.dll
Unpacking file: frontend/resources/app.asar.unpacked/build/Release/winpty-agent.exe
Unpacking file: frontend/libGLSv2.dll
Unpacking file: frontend/libEGL.dll
Unpacking file: frontend/ffmpeg.dll
Unpacking file: frontend/Docker Desktop.exe
Unpacking file: frontend/d3dcompiler_47.dll
Unpacking file: DockerCli.exe
Unpacking file: Docker.Core.dll
Unpacking file: Docker Desktop.exe
Unpacking file: Docker Desktop Installer.exe
Unpacking file: courgette64.exe
Unpacking file: resources/wsl/wsl-data.tar
Unpacking file: resources/wsl/wsl-bootstrap.tar
```

Se debió actualizar el subsistema de Windows par Linux:



### Ejecución del contenedor de PostgreSQL

Se abrió una terminal y se ejecutó el siguiente comando:

```
docker run --name postgres-container -e POSTGRES_USER=postgres -e POSTGRES_PASSWORD=postgres -p 5434:5432 -d postgres
```

Este comando realizó lo siguiente:

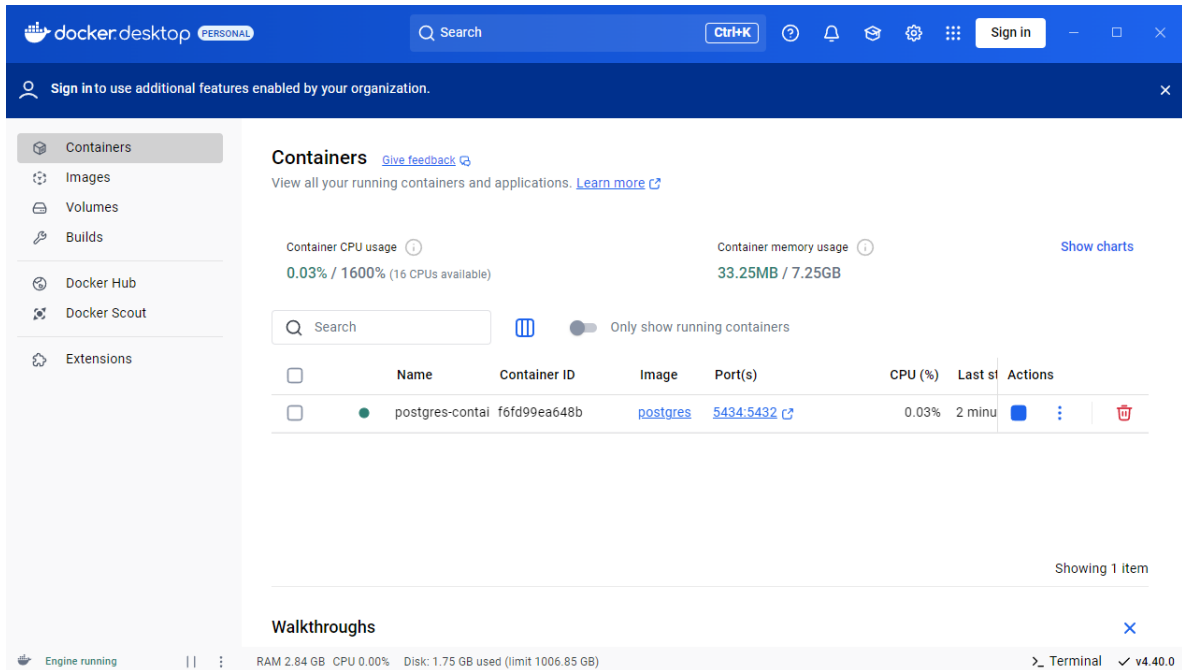
- Asignó el nombre postgres-container al contenedor.
- Estableció tanto el usuario como la contraseña como postgres.
- Expuso el puerto 5432 del contenedor al puerto 5434 del host.
- Ejecutó el contenedor en segundo plano (-d) usando la imagen oficial de PostgreSQL.

### Verificación del estado del contenedor




Se utilizó el siguiente comando para verificar que el contenedor estuviera corriendo:

```
docker ps
```

La salida mostró una fila correspondiente al contenedor postgres, confirmando que se encontraba activo y escuchando en el puerto 5434.



The screenshot shows the Docker Desktop application window. The top bar is blue with the Docker logo, 'docker desktop PERSONAL', a search bar, a 'Ctrl+K' button, and a 'Sign in' button. Below the top bar is a dark blue banner with a user icon and the text 'Sign in to use additional features enabled by your organization.' The left sidebar contains a list of navigation items: Containers (selected), Images, Volumes, Builds, Docker Hub, Docker Scout, and Extensions. The main area is titled 'Containers' and shows 'View all your running containers and applications. Learn more'. It displays two usage metrics: 'Container CPU usage 0.03% / 1600% (16 CPUs available)' and 'Container memory usage 33.25MB / 7.25GB'. Below these is a search bar and a toggle for 'Only show running containers'. A table lists the running containers:

	Name	Container ID	Image	Port(s)	CPU (%)	Last s	Actions
<input type="checkbox"/>	postgres-contai	f6fd99ea648b	postgres	5434-5432	0.03%	2 minu	  

Showing 1 item

Below the table is a 'Walkthroughs' section with a close button. At the bottom, a status bar shows 'Engine running', system resources (RAM 2.84 GB, CPU 0.00%, Disk: 1.75 GB used), and a terminal icon with version 'v4.40.0'.

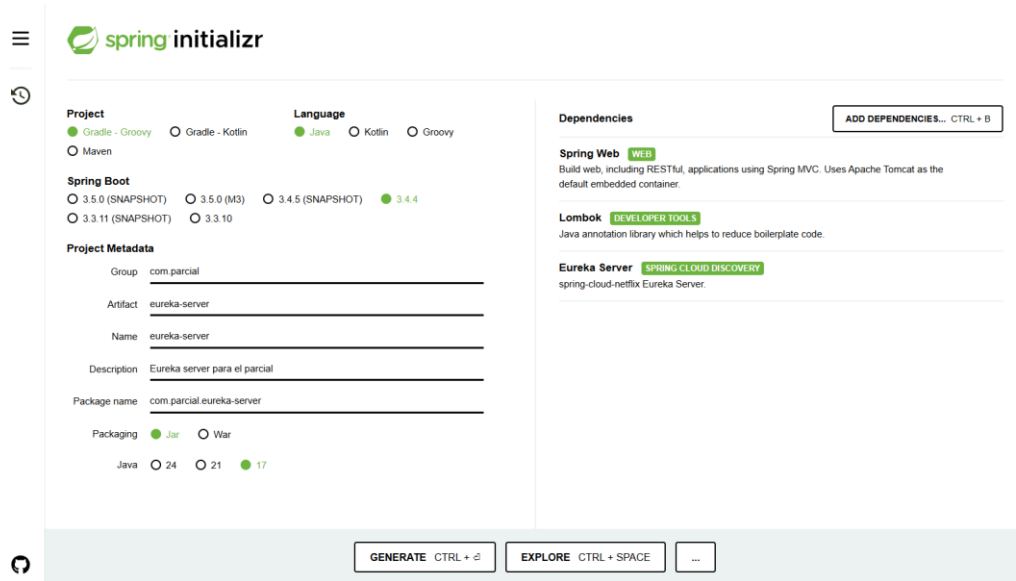
- El comando docker run se empleó **solo la primera vez**, ya que crea y lanza un nuevo contenedor. Una vez creado, **no fue necesario volver a usarlo** para iniciar el contenedor.
- Para usos posteriores, se utilizó el comando:
- docker start postgres-container

Y para detener el contenedor:

docker stop postgres-container

- Se tuvo en cuenta que intentar ejecutar nuevamente docker run con el mismo nombre (postgres-container) provocaría un error como el siguiente:
- docker: Error response from daemon: Conflict. The container name "/postgres-container" is already in use.

## Microservicio para Eureka Server



The screenshot shows the Spring Initializr web interface. On the left, there's a sidebar with a hamburger menu and a refresh icon. The main area is divided into sections: 'Project' (Language: Java, Spring Boot: 3.5.0 (M3)), 'Project Metadata' (Group: com.parcial, Artifact: eureka-server, Name: eureka-server, Description: Eureka server para el parcial, Package name: com.parcial.eureka-server, Packaging: Jar, Java: 21), and 'Dependencies' (Spring Web, Lombok, Eureka Server). At the bottom, there are buttons for 'GENERATE', 'EXPLORE', and a menu icon.

**Project**

Language: ☒ Java ☐ Kotlin ☐ Groovy

Spring Boot: ☐ 3.5.0 (SNAPSHOT) ☐ 3.5.0 (M3) ☐ 3.4.5 (SNAPSHOT) ☒ 3.4.4 ☐ 3.3.11 (SNAPSHOT) ☐ 3.3.10

**Project Metadata**

Group:

Artifact:

Name:

Description:

Package name:

Packaging: ☒ Jar ☐ War

Java: ☐ 24 ☐ 21 ☒ 17

**Dependencies**

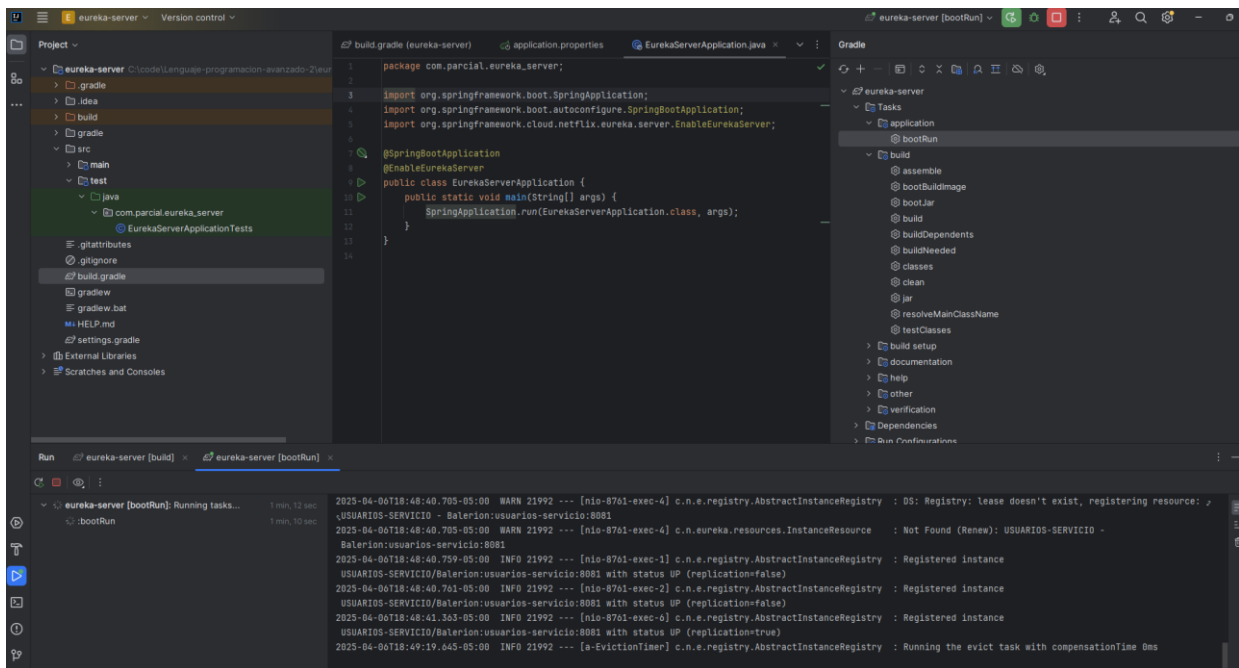
**Spring Web** **WEB**  
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

**Lombok** **DEVELOPER TOOLS**  
Java annotation library which helps to reduce boilerplate code.

**Eureka Server** **SPRING CLOUD DISCOVERY**  
spring-cloud-netflix Eureka Server.

**ADD DEPENDENCIES...** CTRL + B

**GENERATE** CTRL + G **EXPLORE** CTRL + SPACE ...



The screenshot shows an IDE with the following components:

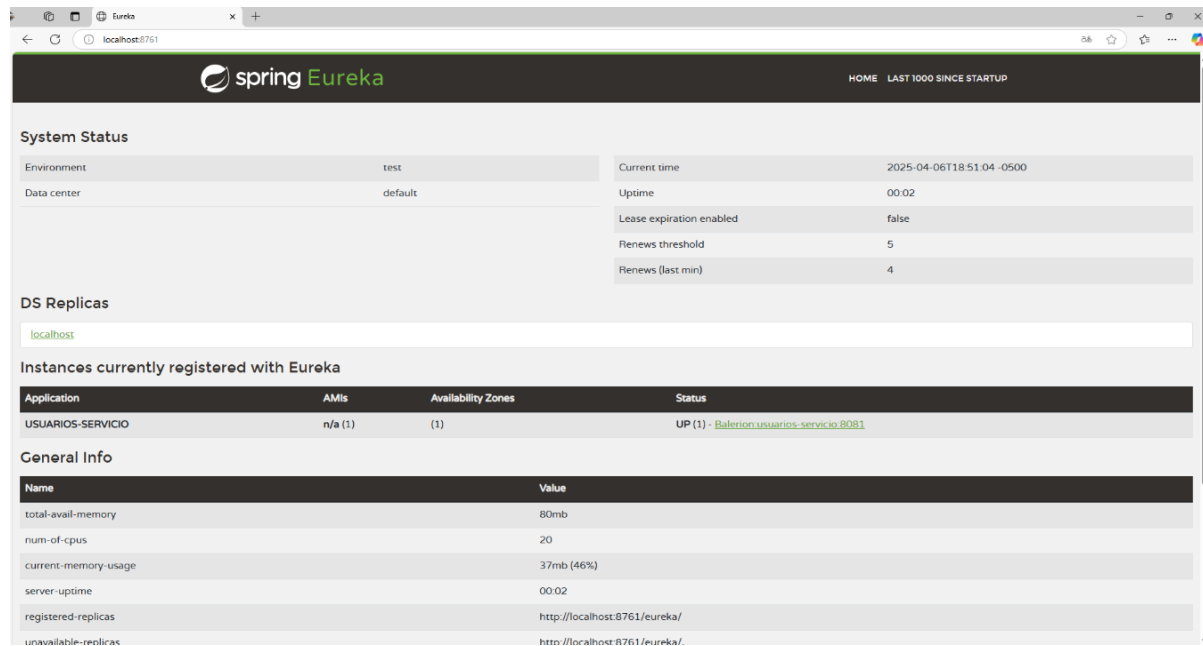
- Project Explorer:** Shows the project structure with folders like 'eureka-server', 'gradle', 'src', 'main', 'test', and 'java'. The 'java' folder contains 'com.parcial.eureka\_server' and 'EurekaServerApplicationTests'.
- Editor:** Displays the 'EurekaServerApplication.java' file. The code is as follows:

```
1 package com.parcial.eureka_server;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;
6
7 @SpringBootApplication
8 @EnableEurekaServer
9 public class EurekaServerApplication {
10     public static void main(String[] args) {
11         SpringApplication.run(EurekaServerApplication.class, args);
12     }
13 }
14
```
- Run Console:** Shows the output of the 'bootRun' task. The logs indicate that the Eureka Server is running successfully on port 8081, with various warnings and information messages from the Spring Cloud Discovery module.



## Microservicio de Usuarios

Al iniciar solamente usuarios-servicio:



The screenshot shows the Spring Eureka dashboard in a web browser. The dashboard displays the following information:

- System Status:** A table showing environment (test), data center (default), current time (2025-04-06T18:51:04 -0500), uptime (00:02), lease expiration enabled (false), renews threshold (5), and renews (last min) (4).
- DS Replicas:** A table showing the replica count for the application (1).
- Instances currently registered with Eureka:** A table showing the application (USUARIOS-SERVICIO), AMIs (n/a (1)), availability zones (1), and status (UP (1) - [Balancer: usuarios.servicio.8081](#)).
- General Info:** A table showing various system metrics such as total-avail-memory (80mb), num-of-cpus (20), current-memory-usage (37mb (46%)), server-uptime (00:02), registered-replicas (http://localhost:8761/eureka/), and unavailable-replicas (http://localhost:8761/eureka/).

Creación de contenedor Docker para base de datos Postgres para usuarios-servicio:

```
C:\Windows\System32\cmd.e x + v

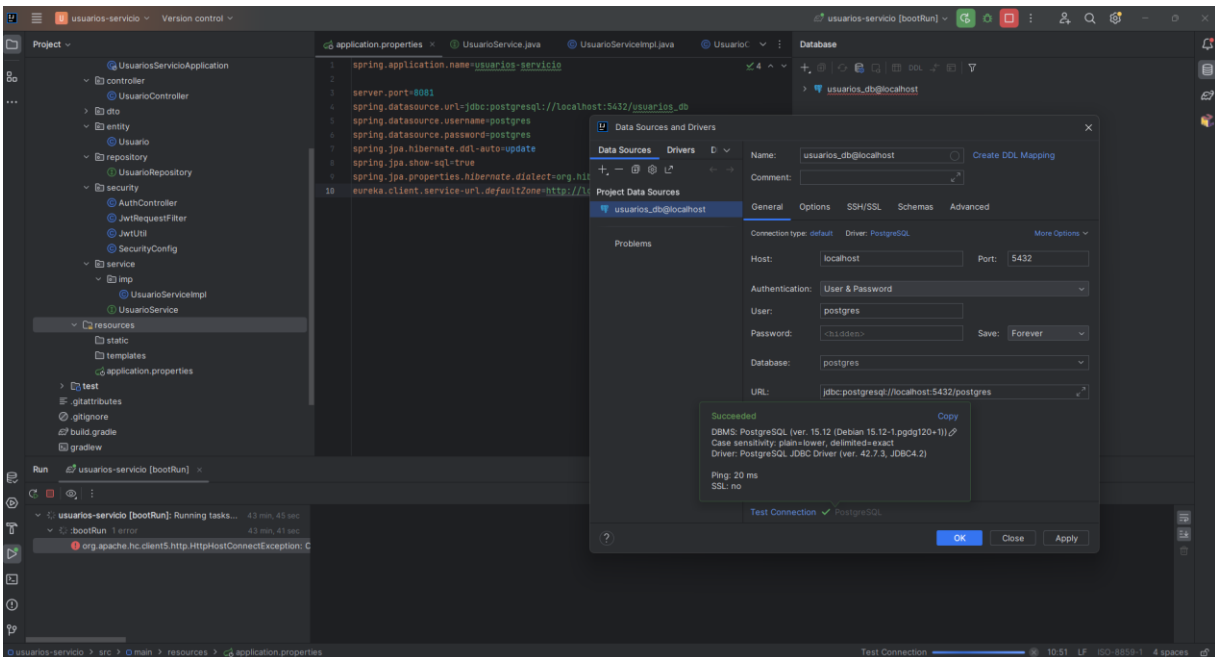
See 'docker run --help'.

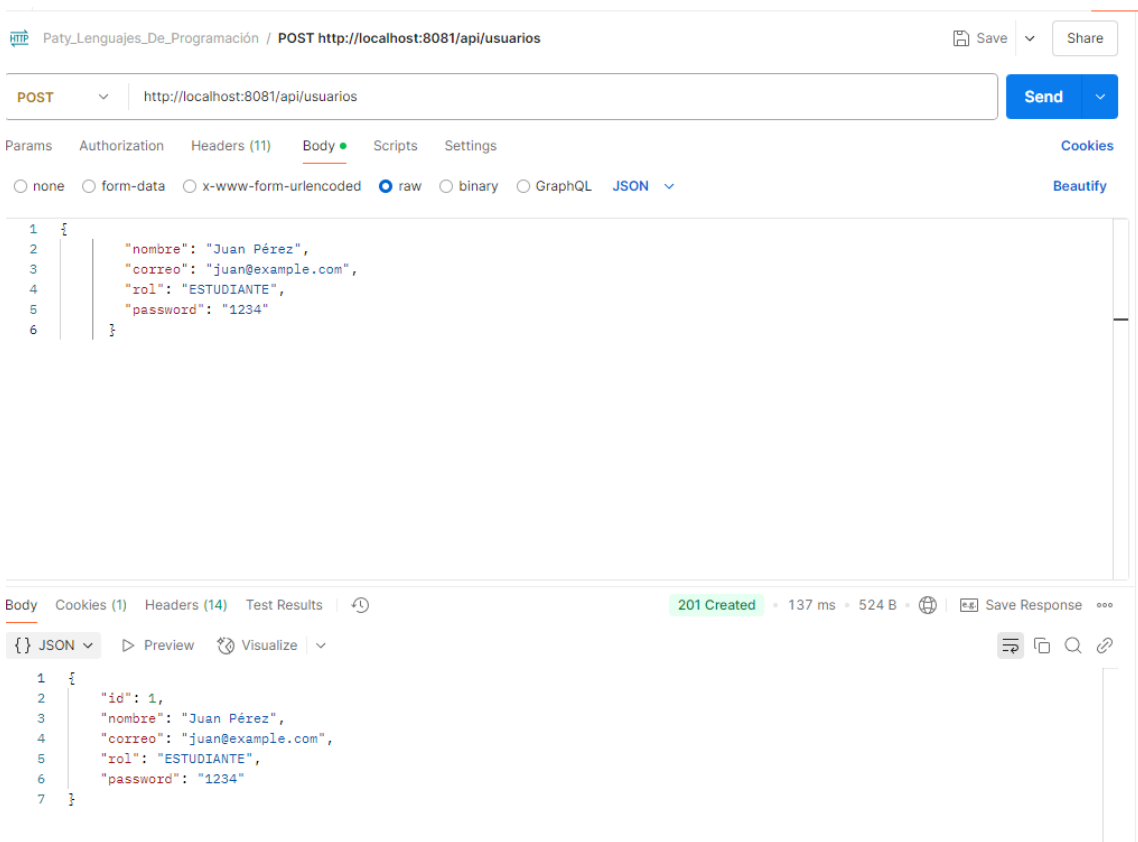
C:\code\Lenguaje-programacion-avanzado-2\usuarios-servicio>docker run --name postgres-usuarios -e POSTGRES_USER=postgres
-e POSTGRES_PASSWORD=postgres -e POSTGRES_DB=usuarios_db -p 5432:5432 -d postgres:15
Unable to find image 'postgres:15' locally
15: Pulling from library/postgres
6e909acdb790: Pull complete
8414d53e2438: Pull complete
27575b4803be: Pull complete
45b3aebd39ee: Pull complete
b860fad5d5df: Pull complete
4837fd8608b0: Pull complete
fbd4eef6281f: Pull complete
467f5a6623a6: Pull complete
c2ef34ce685b: Pull complete
f687ffcdcccf: Pull complete
bf32901fbacb: Pull complete
9a73c0e6b5ee: Pull complete
8b0faa55b3e6: Pull complete
169f3b0e9597: Pull complete
Digest: sha256:9e9298817d19f4bd60c5028a25762b394db37dda173dd3d035a1bc155542051a
Status: Downloaded newer image for postgres:15
614b46d68a1404a0f4fc4620fe7851478a39d821ef60cc9175f5a320c40673ab

C:\code\Lenguaje-programacion-avanzado-2\usuarios-servicio>docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                               NAMES
614b46d68a14   postgres:15    "docker-entrypoint.s..." 12 seconds ago Up 11 seconds 0.0.0.0:5432->5432/tcp          postgres-usuarios

C:\code\Lenguaje-programacion-avanzado-2\usuarios-servicio>
```

Conexión de usuarios-servicio a base de datos:





## Login y generación del JWT:

POST http://localhost:8081/auth/login

Params Authorization Headers (11) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   |   "correo": "juan@example.com",
3   |   "password": "1234"
4   | }

```

Body Cookies (1) Headers (14) Test Results

200 OK · 188 ms · 649 B

Save Response

JSON Preview Visualize

```
1 {
2   |   "token": "eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJqdWV4Ym1wbGUuY29tIiwicm9sIjoiaWVudVURJQU5URSI6Im1hdCI6MTc0Mzk4OTcxNiwiZXhwIjoxNzQ4MDc2MTE2fQ.9NTYDtsMk6s5VzMIn17gZ71ttFDxaLfBDNKAICs2WnxkGcYf1scwbRmN4gsj9y2Lu84eI07K6wAzDNznL2_Q"
3   | }

```

## Usar el token para ver los detalles de un usuario

Paty\_Lenguajes\_De\_Programación / GET usuarios http://localhost:8081/api/usuarios/1

GET http://localhost:8081/api/usuarios/1

Params Authorization Headers (9) Body Scripts Settings

Headers 8 hidden

Key	Value	Description
<input checked="" type="checkbox"/> Authorization	Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJqdWV4Ym1wbGUuY29tIiwicm9sIjoiaWVudVURJQU5URSI6Im1hdCI6MTc0Mzk4OTcxNiwiZXhwIjoxNzQ4MDc2MTE2fQ.9NTYDtsMk6s5VzMIn17gZ71ttFDxaLfBDNKAICs2WnxkGcYf1scwbRmN4gsj9y2Lu84eI07K6wAzDNznL2_Q	
Key	Value	Description

Body Cookies (1) Headers (14) Test Results

200 OK · 37 ms · 519 B

Save Response

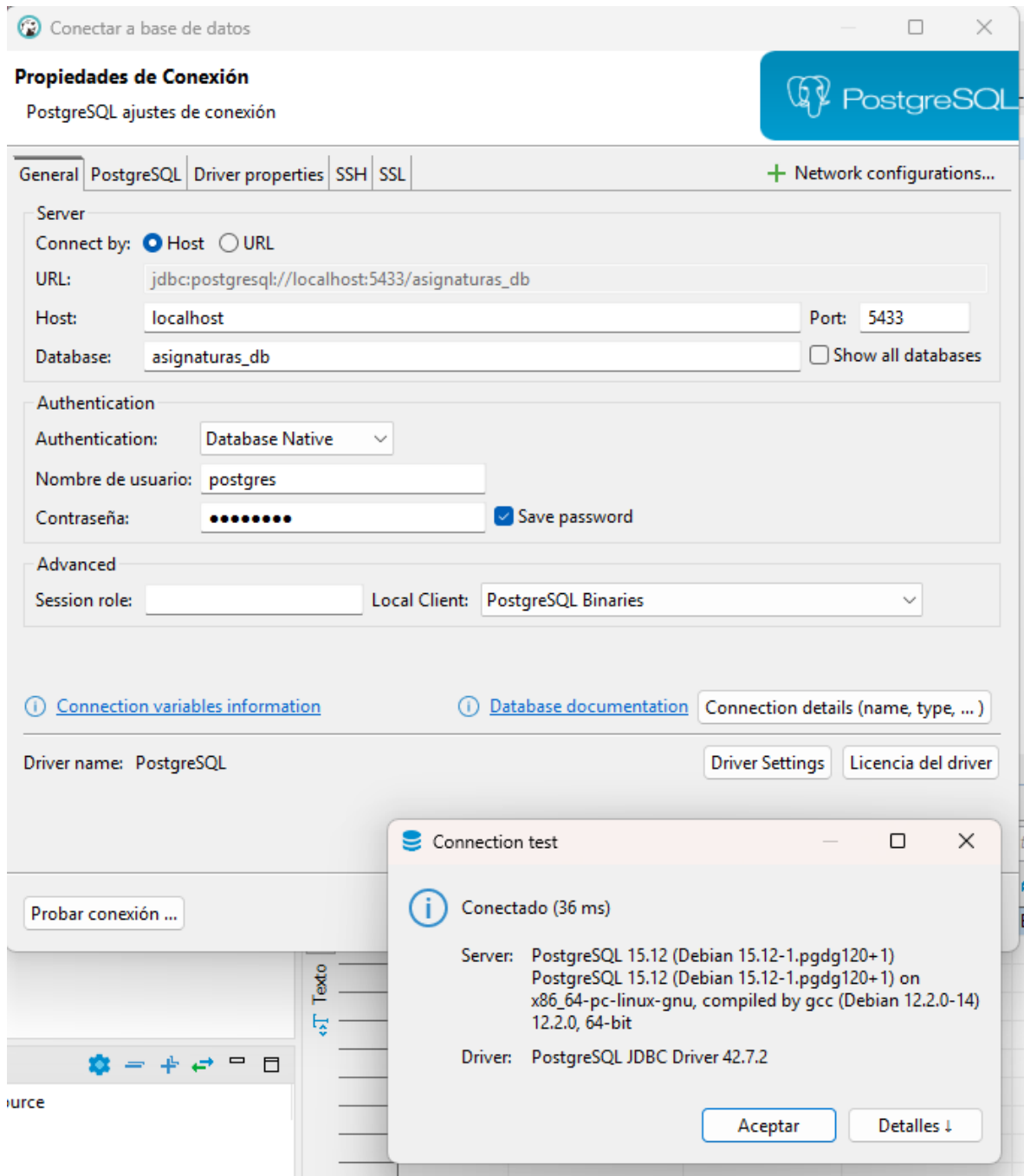
JSON Preview Visualize

```
1 {
2   |   "id": 1,
3   |   "nombre": "Juan Pérez",
4   |   "correo": "juan@example.com",
5   |   "rol": "ESTUDIANTE",
6   |   "password": "1234"
7   | }

```



Conexión a la base de datos de asignaturas:



The image shows a 'Conectar a base de datos' (Connect to database) window for PostgreSQL. The 'Propiedades de Conexión' (Connection Properties) tab is active, showing the 'General' sub-tab. The 'Server' section has 'Connect by' set to 'Host', with 'URL' as 'jdbc:postgresql://localhost:5433/asignaturas\_db', 'Host' as 'localhost', 'Port' as '5433', and 'Database' as 'asignaturas\_db'. The 'Authentication' section has 'Authentication' set to 'Database Native', 'Nombre de usuario' (Username) as 'postgres', and 'Contraseña' (Password) masked with dots, with 'Save password' checked. The 'Advanced' section has 'Session role' empty and 'Local Client' set to 'PostgreSQL Binaries'. Below the tabs are links for 'Connection variables information', 'Database documentation', and 'Connection details (name, type, ...)'. The 'Driver name' is 'PostgreSQL', with buttons for 'Driver Settings' and 'Licencia del driver'. A 'Probar conexión ...' (Test connection ...) button is at the bottom left. A 'Connection test' dialog box is open in the foreground, showing a successful connection: 'Conectado (36 ms)'. It lists the server as 'PostgreSQL 15.12 (Debian 15.12-1.pgdg120+1) on x86\_64-pc-linux-gnu, compiled by gcc (Debian 12.2.0-14) 12.2.0, 64-bit' and the driver as 'PostgreSQL JDBC Driver 42.7.2'. The dialog has 'Aceptar' (Accept) and 'Detalles ↓' (Details ↓) buttons.

Conectar a base de datos

**Propiedades de Conexión**  
PostgreSQL ajustes de conexión

General | PostgreSQL | Driver properties | SSH | SSL | + Network configurations...

Server

Connect by: ☒ Host ☐ URL

URL: jdbc:postgresql://localhost:5433/asignaturas\_db

Host: localhost Port: 5433

Database: asignaturas\_db ☐ Show all databases

Authentication

Authentication: Database Native

Nombre de usuario: postgres

Contraseña: ..... ☒ Save password

Advanced

Session role: Local Client: PostgreSQL Binaries

[Connection variables information](#) [Database documentation](#) Connection details (name, type, ...)

Driver name: PostgreSQL Driver Settings Licencia del driver

Probar conexión ...

Connection test

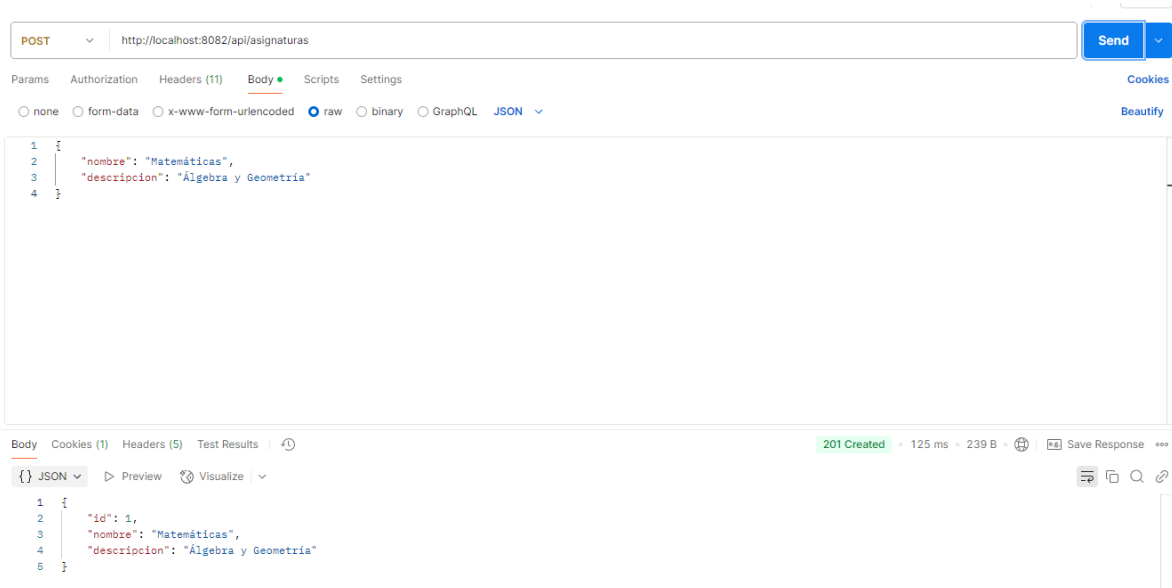
Conectado (36 ms)

Server: PostgreSQL 15.12 (Debian 15.12-1.pgdg120+1)  
PostgreSQL 15.12 (Debian 15.12-1.pgdg120+1) on  
x86\_64-pc-linux-gnu, compiled by gcc (Debian 12.2.0-14)  
12.2.0, 64-bit

Driver: PostgreSQL JDBC Driver 42.7.2

Aceptar Detalles ↓

## Creación de una asignatura:



POST http://localhost:8082/api/asignaturas

Params Authorization Headers (11) Body **Scripts** Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON**

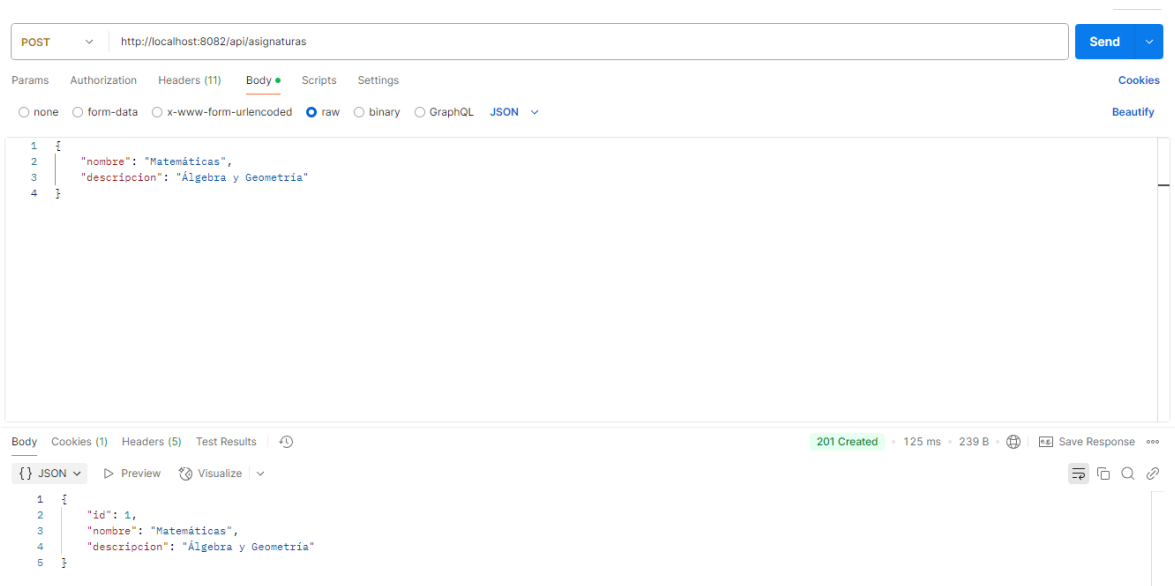
```
1 {
2   "nombre": "Matemáticas",
3   "descripcion": "Álgebra y Geometría"
4 }
```

Body Cookies (1) Headers (5) Test Results **201 Created** · 125 ms · 239 B · Save Response

**JSON** Preview Visualize

```
1 {
2   "id": 1,
3   "nombre": "Matemáticas",
4   "descripcion": "Álgebra y Geometría"
5 }
```

## Creación de una asignatura:



POST http://localhost:8082/api/asignaturas

Params Authorization Headers (11) Body **Scripts** Settings

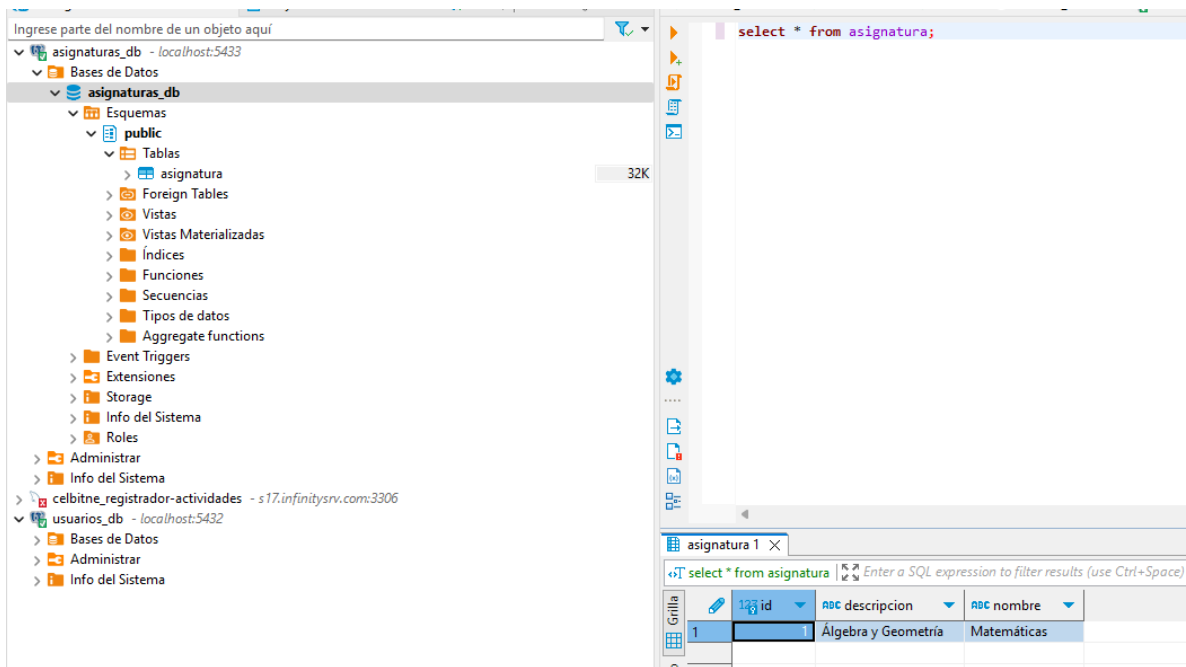
☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON**

```
1 {
2   "nombre": "Matemáticas",
3   "descripcion": "Álgebra y Geometría"
4 }
```

Body Cookies (1) Headers (5) Test Results **201 Created** · 125 ms · 239 B · Save Response

**JSON** Preview Visualize

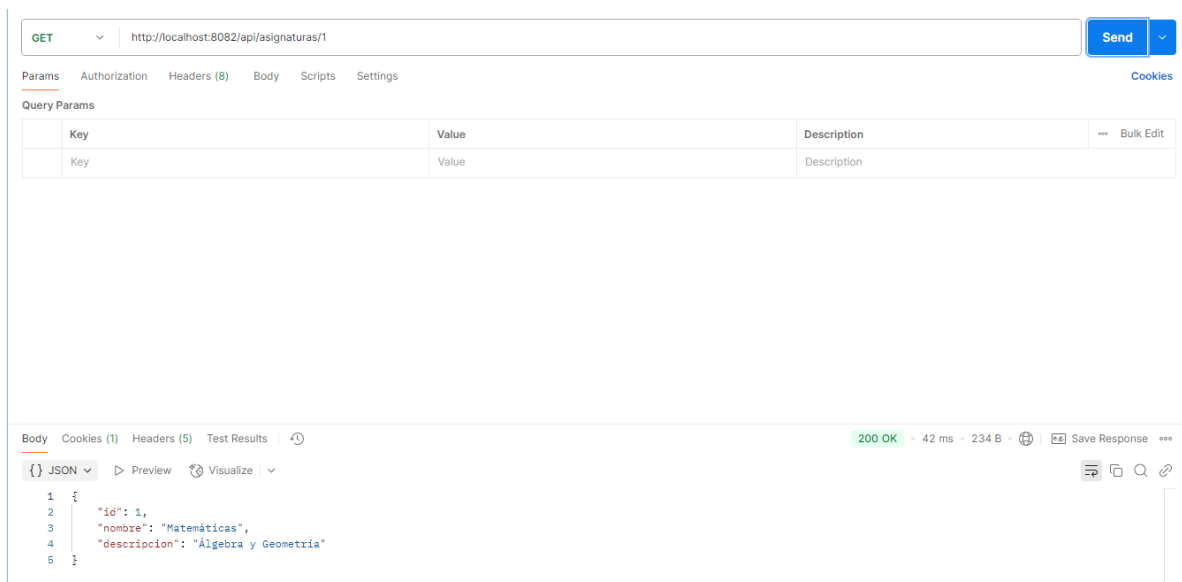
```
1 {
2   "id": 1,
3   "nombre": "Matemáticas",
4   "descripcion": "Álgebra y Geometría"
5 }
```



The screenshot shows a database management interface. On the left, a tree view displays the database structure for 'asignaturas\_db'. The 'public' schema is expanded, showing tables, foreign tables, views, materialized views, indices, functions, sequences, data types, aggregate functions, event triggers, extensions, storage, system information, and roles. The 'asignatura' table is highlighted. On the right, a SQL query window shows the query 'select \* from asignatura;'. Below the query window, a table grid displays the results of the query, showing one row with columns 'id', 'descripcion', and 'nombre'.

id	descripcion	nombre
1	Álgebra y Geometría	Matemáticas

Obtener asignatura por id:



The screenshot shows a REST client interface. The top bar displays the method 'GET' and the URL 'http://localhost:8082/api/asignaturas/1'. The 'Send' button is visible. Below the URL bar, the 'Params' tab is selected, showing a table with columns 'Key' and 'Value'. The 'Body' tab is also visible. At the bottom, the 'Body' tab is selected, showing the JSON response of the request. The response is a JSON object with fields 'id', 'nombre', and 'descripcion'.

```

{
  "id": 1,
  "nombre": "Matemáticas",
  "descripcion": "Álgebra y Geometría"
}

```



Verificar si existe asignatura:

GET http://localhost:8082/api/asignaturas/existe?nombre=Matemáticas Send

Params Authorization Headers (8) Body Scripts Settings Cookies

Query Params

<input checked="" type="checkbox"/> Key	Value	Description	⋮ Bulk Edit
<input checked="" type="checkbox"/> nombre	Matemáticas		
Key	Value	Description	

Body Cookies (1) Headers (5) Test Results 200 OK 88 ms 168 B Save Response

{ } JSON Preview Visualize

```
1 true
```

Obtener todas las asignaturas:

GET http://localhost:8082/api/asignaturas Send

Params Authorization Headers (8) Body Scripts Settings Cookies

Query Params

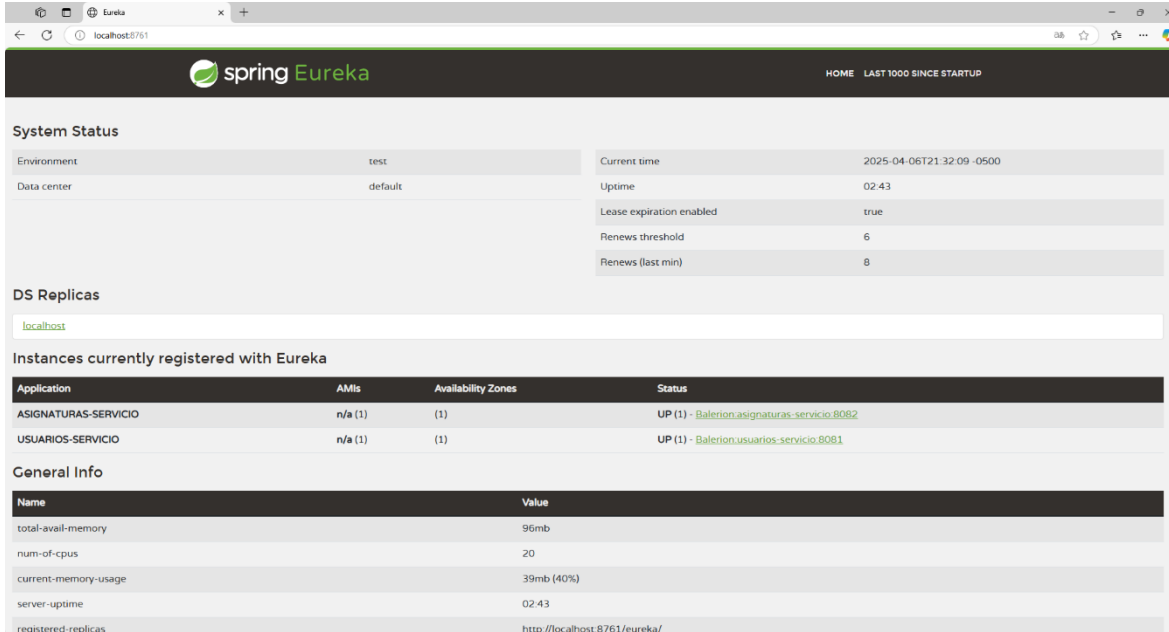
<input type="checkbox"/> Key	Value	Description	⋮ Bulk Edit
Key	Value	Description	

Body Cookies (1) Headers (5) Test Results 200 OK 184 ms 236 B Save Response

{ } JSON Preview Visualize

```
1 [
2   {
3     "id": 1,
4     "nombre": "Matemáticas",
5     "descripcion": "Álgebra y Geometría"
6   }
7 ]
```

Validación de que los dos servicios desarrollados hasta el momento están registrados en Eureka y funcionando:



The screenshot shows the Spring Eureka web interface. The top navigation bar includes 'HOME' and 'LAST 1000 SINCE STARTUP'. The 'System Status' section displays the following information:

Environment	test	Current time	2025-04-06T21:32:09 -0500
Data center	default	Uptime	02:43
		Lease expiration enabled	true
		Renews threshold	6
		Renews (last min)	8

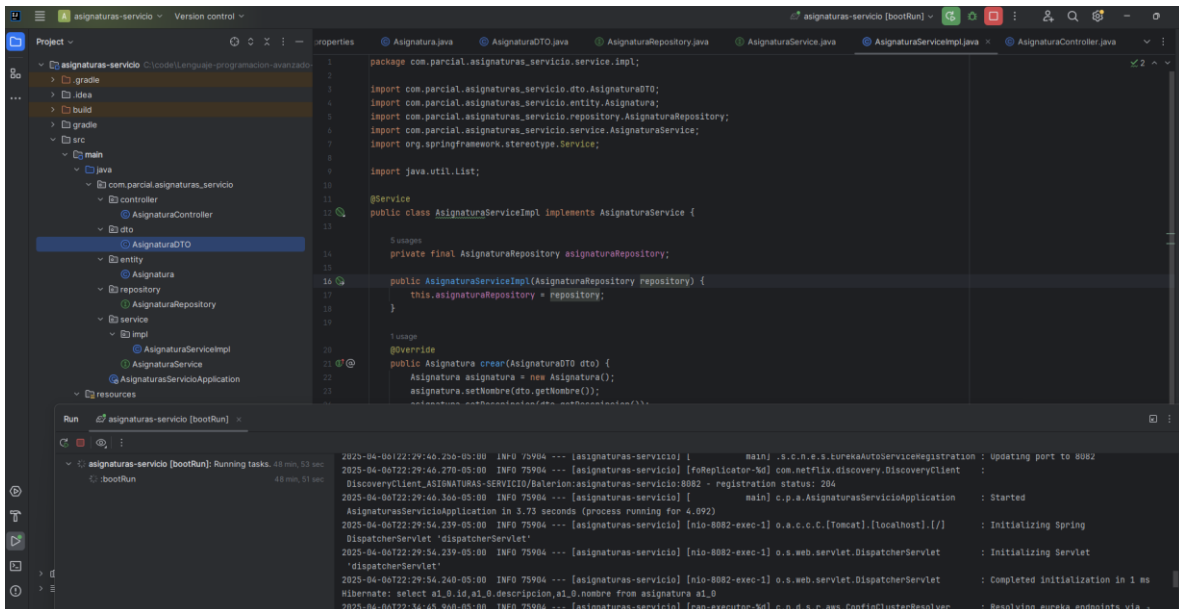
The 'DS Replicas' section shows 'localhost'.

The 'Instances currently registered with Eureka' section displays a table with the following data:

Application	AMIs	Availability Zones	Status
ASIGNATURAS-SERVICIO	n/a (1)	(1)	UP (1) - <a href="#">Balerion.asignaturas-servicio.8082</a>
USUARIOS-SERVICIO	n/a (1)	(1)	UP (1) - <a href="#">Balerion.usuarios-servicio.8081</a>

The 'General Info' section displays the following information:

Name	Value
total-avail-memory	96mb
num-of-cpus	20
current-memory-usage	39mb (40%)
server-up-time	02:43
registered-replicas	<a href="http://localhost:8761/eureka/">http://localhost:8761/eureka/</a>



The screenshot shows an IDE with the code for the 'Asignaturas-Servicio' application. The code is in the 'AsignaturaServiceImpl' class, which implements the 'AsignaturaService' interface. The code includes imports for 'AsignaturaDTO', 'AsignaturaRepository', 'AsignaturaService', and 'AsignaturaController'. The 'AsignaturaServiceImpl' class has a 'create' method that creates a new 'Asignatura' object and saves it to the repository.

```
package com.parcial.asignaturas_servicio.service.impl;

import com.parcial.asignaturas_servicio.dto.AsignaturaDTO;
import com.parcial.asignaturas_servicio.entity.Asignatura;
import com.parcial.asignaturas_servicio.repository.AsignaturaRepository;
import com.parcial.asignaturas_servicio.service.AsignaturaService;
import org.springframework.stereotype.Service;
import java.util.List;

@Service
public class AsignaturaServiceImpl implements AsignaturaService {

    @Autowired
    private final AsignaturaRepository asignaturaRepository;

    public AsignaturaServiceImpl(AsignaturaRepository repository) {
        this.asignaturaRepository = repository;
    }

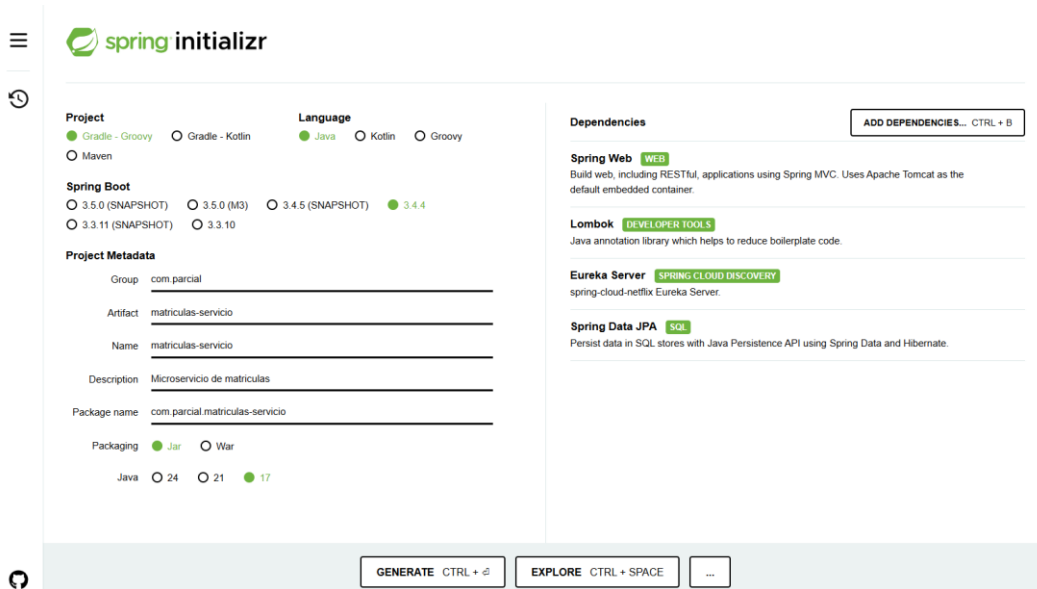
    @Override
    public Asignatura crear(AsignaturaDTO dto) {
        Asignatura asignatura = new Asignatura();
        asignatura.setNombre(dto.getNombre());
        asignatura.setDescripcion(dto.getDescripcion());
    }
}
```

The 'Run' console shows the following logs:

```
2025-04-06T22:29:46.250-05:00 INFO 75904 --- [signaturas-servicio] l main) .s.c.h.e.s.turekaAutoServiceRegistration : Updating port to 8082
2025-04-06T22:29:46.270-05:00 INFO 75904 --- [signaturas-servicio] [fmrReplicator-Md] com.netflix.discovery.DiscoveryClient :
DiscoveryClient_ASIGNATURAS-SERVICIO/Balerion:signaturas-servicio:8082 - registration status: 204
2025-04-06T22:29:46.356-05:00 INFO 75904 --- [signaturas-servicio] l main) o.p.a.AsignaturasServicioApplication : Started
AsignaturasServicioApplication in 3.73 seconds (process running for 4.092)
2025-04-06T22:29:54.219-05:00 INFO 75904 --- [signaturas-servicio] [nio-8082-exec-1] o.a.c.c.f.[Tomcat].[localhost].[/] : Initializing Spring
DispatcherServlet 'dispatcherServlet'
2025-04-06T22:29:54.239-05:00 INFO 75904 --- [signaturas-servicio] [nio-8082-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet
'dispatcherServlet'
2025-04-06T22:29:54.240-05:00 INFO 75904 --- [signaturas-servicio] [nio-8082-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 1 ms
Hibernate: select a1_0.id,a1_0.descripcion,a1_0.nombre from asignatura a1_0
2025-04-06T22:34:45.940-05:00 INFO 75904 --- [signaturas-servicio] [rap-executor-Md] c.n.d.s.f.ans.ConfigClusterResolver : Resolving eureka endpoints via
```

# Microservicio de Matrículas

Generación de Matriculas servicio:



The screenshot shows the Spring Initializr web application. The 'Project' section has 'Gradle - Groovy' selected. The 'Language' section has 'Java' selected. The 'Spring Boot' section has '3.4.4' selected. The 'Project Metadata' section shows 'Group: com.parcial', 'Artifact: matriculas-servicio', 'Name: matriculas-servicio', 'Description: Microservicio de matriculas', and 'Package name: com.parcial.matriculas-servicio'. The 'Packaging' section has 'Jar' selected. The 'Dependencies' section shows 'Spring Web' (WEB), 'Lombok' (DEVELOPER TOOLS), 'Eureka Server' (SPRING CLOUD DISCOVERY), and 'Spring Data JPA' (SQL). The bottom buttons are 'GENERATE CTRL + G', 'EXPLORE CTRL + SPACE', and '...'.

Generación de contenedor de base de datos para matriculas-servicio:

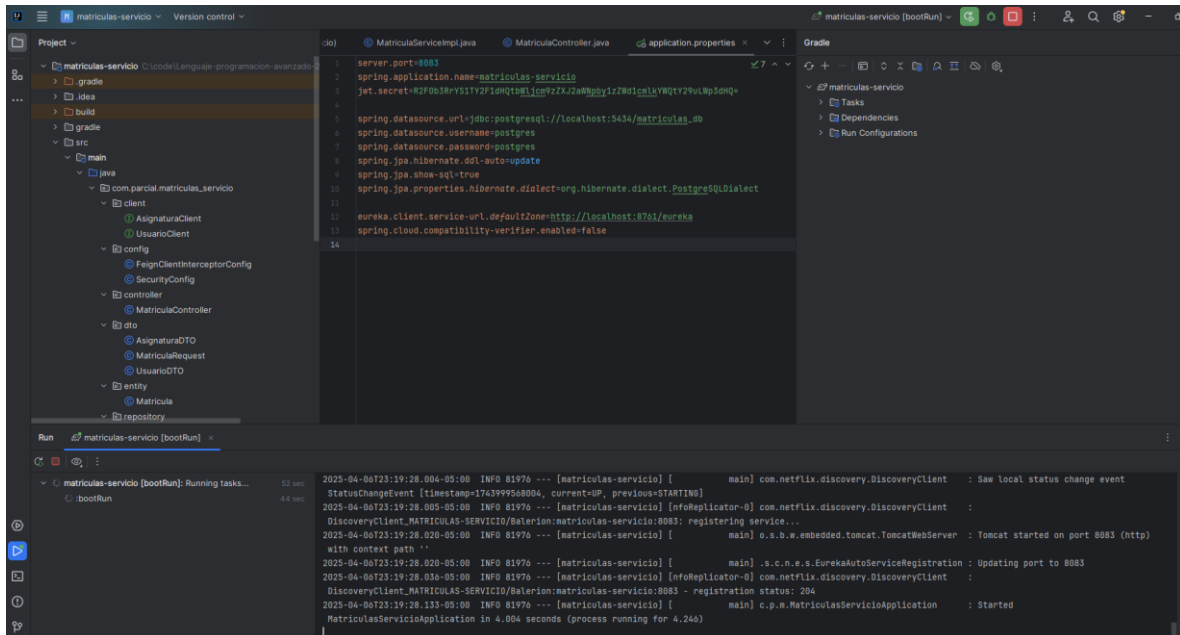
Comando:

```
docker run --name postgres-matriculas -e POSTGRES_USER=postgres -e POSTGRES_PASSWORD=postgres -e POSTGRES_DB=matriculas_db -p 5434:5432 -d postgres:15
```

```
C:\code\lenguaje-programacion-avanzado-2\usuarios-servicio>docker run --name postgres-matriculas -e POSTGRES_USER=postgres -e POSTGRES_PASSWORD=postgres -e POSTGRES_DB=matriculas_db -p 5434:5432 -d postgres:15
43f6851db656567242bc6631897fc19693d19e5f0c185a351178f8d972ba3d18

C:\code\lenguaje-programacion-avanzado-2\usuarios-servicio>docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED       STATUS      PORTS                               NAMES
43f6851db656   postgres "docker-entrypoint.s..." 5 seconds ago Up 4 seconds 0.0.0.0:5434->5432/tcp   postgres-matriculas
32f4dd91dee    postgres "docker-entrypoint.s..." 57 minutes ago Up 57 minutes 0.0.0.0:5433->5432/tcp   postgres-asignaturas
614b46d08a14   postgres "docker-entrypoint.s..." 4 hours ago   Up 4 hours   0.0.0.0:5432->5432/tcp   postgres-usuarios
```

## Ejecución del microservicio de matrículas:



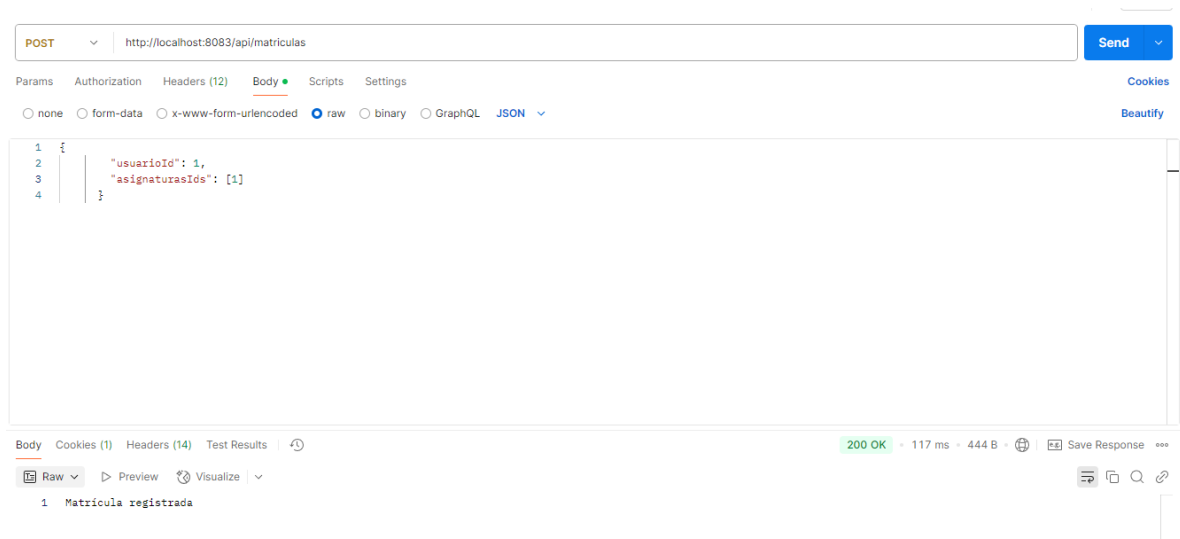
```

server.port=8083
spring.application.name=matriculas-servicio
spring.datasource.url=jdbc:postgresql://localhost:5434/matriculas_db
spring.datasource.username=postgres
spring.datasource.password=postgres
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
eureka.client.service-url.defaultZone=http://localhost:8761/eureka
spring.cloud.compatibility-verifier.enabled=false
  
```

```

2025-04-06T23:19:28.004-05:00 INFO 81976 --- [matriculas-servicio] [main] com.netflix.discovery.DiscoveryClient : Saw local status change event
StatusChangeEvent [timestamp=194399558004, current-up, previous=STARTING]
2025-04-06T23:19:28.005-05:00 INFO 81976 --- [matriculas-servicio] [nfoReplicator-0] com.netflix.discovery.DiscoveryClient :
DiscoveryClient_MATRICULAS-SERVICIO/Salerion:matriculas-servicio:8083: registering service...
2025-04-06T23:19:28.020-05:00 INFO 81976 --- [matriculas-servicio] [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8083 (http
with context path '')
2025-04-06T23:19:28.020-05:00 INFO 81976 --- [matriculas-servicio] [main] .s.c.n.e.s.EurekaAutoServiceRegistration : Updating port to 8083
2025-04-06T23:19:28.036-05:00 INFO 81976 --- [matriculas-servicio] [nfoReplicator-0] com.netflix.discovery.DiscoveryClient :
DiscoveryClient_MATRICULAS-SERVICIO/Salerion:matriculas-servicio:8083 - registration status: 204
2025-04-06T23:19:28.113-05:00 INFO 81976 --- [matriculas-servicio] [main] c.p.w.MatriculasServicioApplication : Started
MatriculasServicioApplication in 4.004 seconds (process running for 4.246)
  
```

Para la creación de una matrícula, primero se debe hacer un login en usuarios y obtener el token jwt, con este token se envía un request a matrículas-servicio que hace una validación del token y adicionalmente consulta a usuarios-servicio si el usuario existe y consulta a asignaturas-servicio si la asignatura existe. Si todos existen, realiza la matrícula, como se muestra a continuación. Internamente los consumos se realizan utilizando Feign Client:



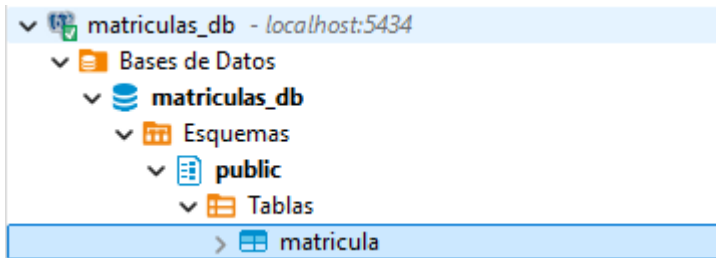
```

POST http://localhost:8083/api/matriculas

{
  "usuarioId": 1,
  "asignaturasIds": [1]
}
  
```

```

200 OK - 117 ms - 444 B
1 Matricula registrada
  
```



El registro fue creado en la base de datos exitosamente:

Propiedades Datos Diagrama ER					
matricula <i>Enter a SQL expression to filter results (use Ctrl+Space)</i>					
Grilla	id	asignatura_id	fecha_matricula	usuario_id	
1	1	1	2025-04-06	1	

Respecto al estado de los servicios en Eureka Server, se puede observar que los tres microservicios están en ejecución:

**spring Eureka** HOME LAST 1000 SINCE STARTUP

**System Status**

Environment	test	Current time	2025-04-06T23:20:39 -0500
Data center	default	Uptime	04:32
		Lease expiration enabled	true
		Renews threshold	8
		Renews (last min)	10

**DS Replicas**

localhost

**Instances currently registered with Eureka**

Application	AMIs	Availability Zones	Status
ASIGNATURAS-SERVICIO	n/a (1)	(1)	UP (1) - <a href="#">Balerion.asignaturas-servicio.8082</a>
MATRICULAS-SERVICIO	n/a (1)	(1)	UP (1) - <a href="#">Balerion.matriculas-servicio.8083</a>
USUARIOS-SERVICIO	n/a (1)	(1)	UP (1) - <a href="#">Balerion.usuarios-servicio.8081</a>

**General Info**

Name	Value
total-avail-memory	96mb
num-of-cpus	20
current-memory-usage	36mb (37%)
server-uptime	04:32