

# C4 - Desenvolupament backend

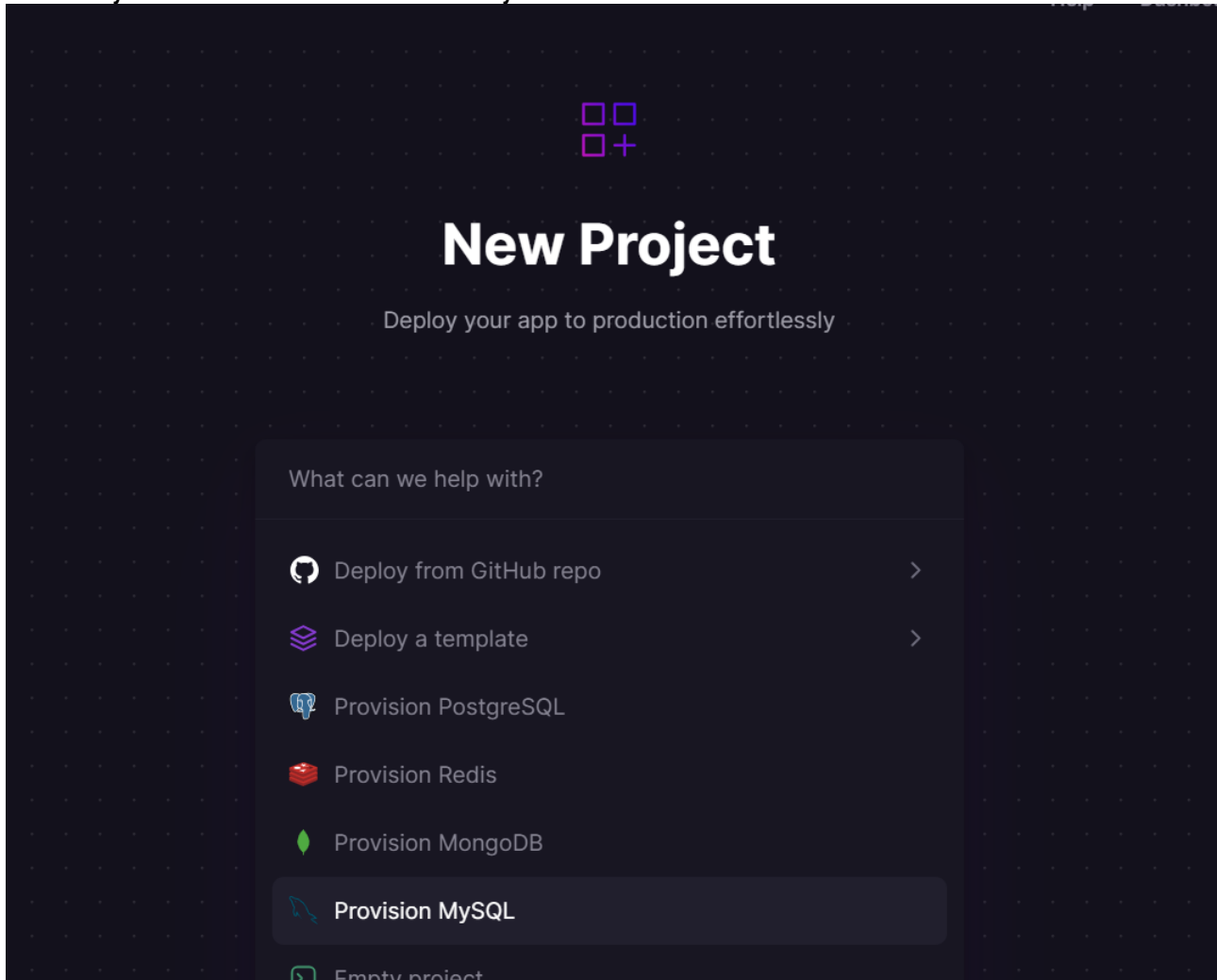
## TA26 - Spring Mysql con Railway

### Index:

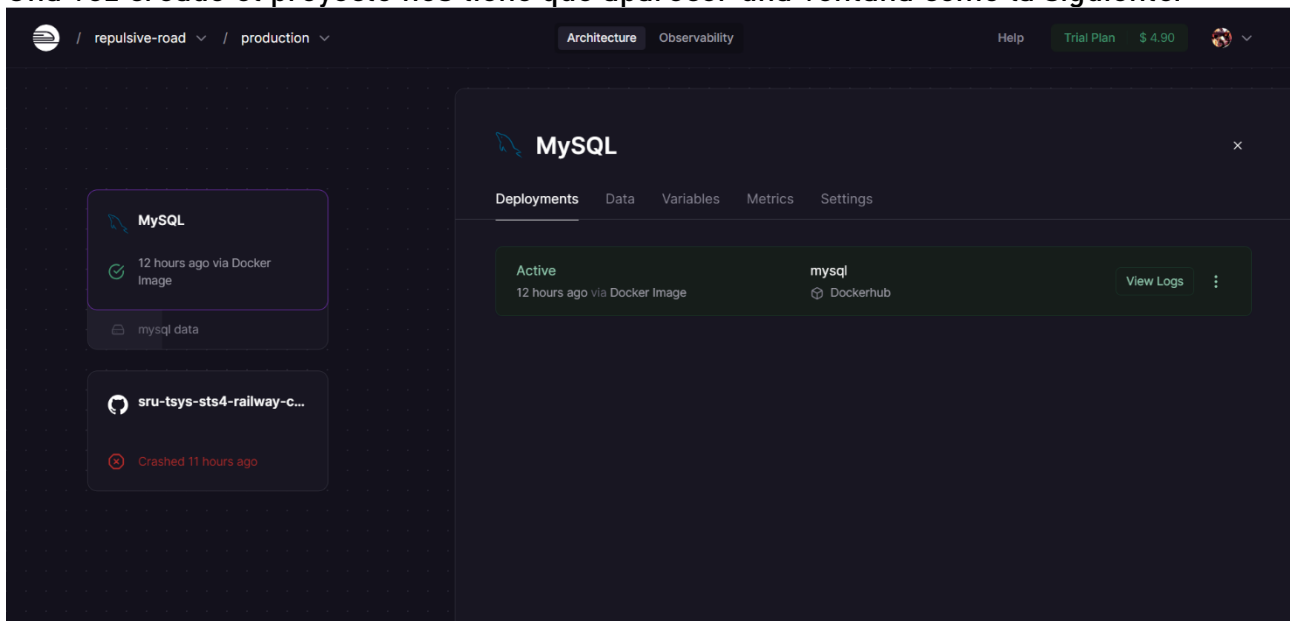
Creación de base de datos:.....	1
Creación de tablas e inserción de datos: .....	3
Añadir la API del Ejercicio 1 a Railway: .....	4
Configurar el entorno de la API en Railway: .....	5
Comprobación del correcto funcionamiento:.....	9

## Creación de base de datos:


Nos dirigimos a <https://railway.app/> y nos registramos, una vez registrados hacemos clic en new y seleccionamos Provision MySQL.



Una vez creado el proyecto nos tiene que aparecer una ventana como la siguiente.




Dentro del apartado de Variables podemos ver los datos de conexión de la base de datos MySQL que posteriormente se utilizaran para poder llenarla de datos de muestra y enlazar la API.

 **MySQL** ×

Deployments   Data   **Variables**   Metrics   Settings

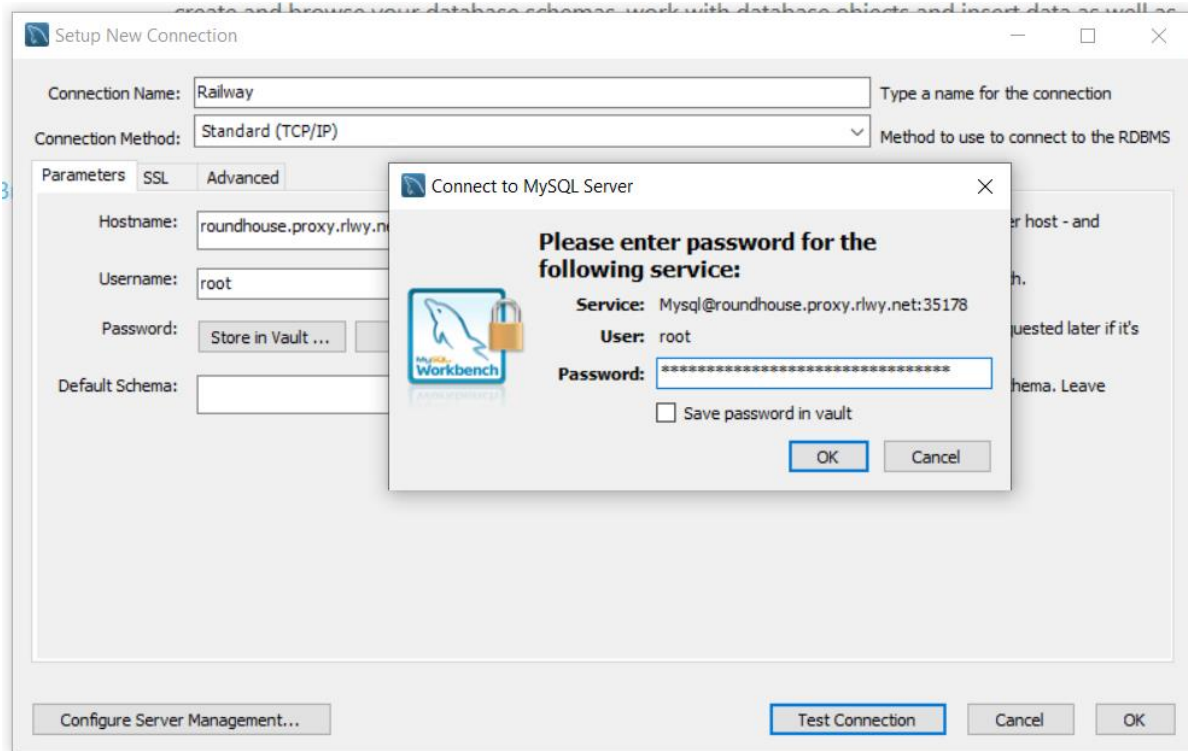
9 Service Variables ↕ Shared Variable { } RAW Editor + New Variable

 Looking to connect this database to another service? Add a [Variable Reference](#) ×

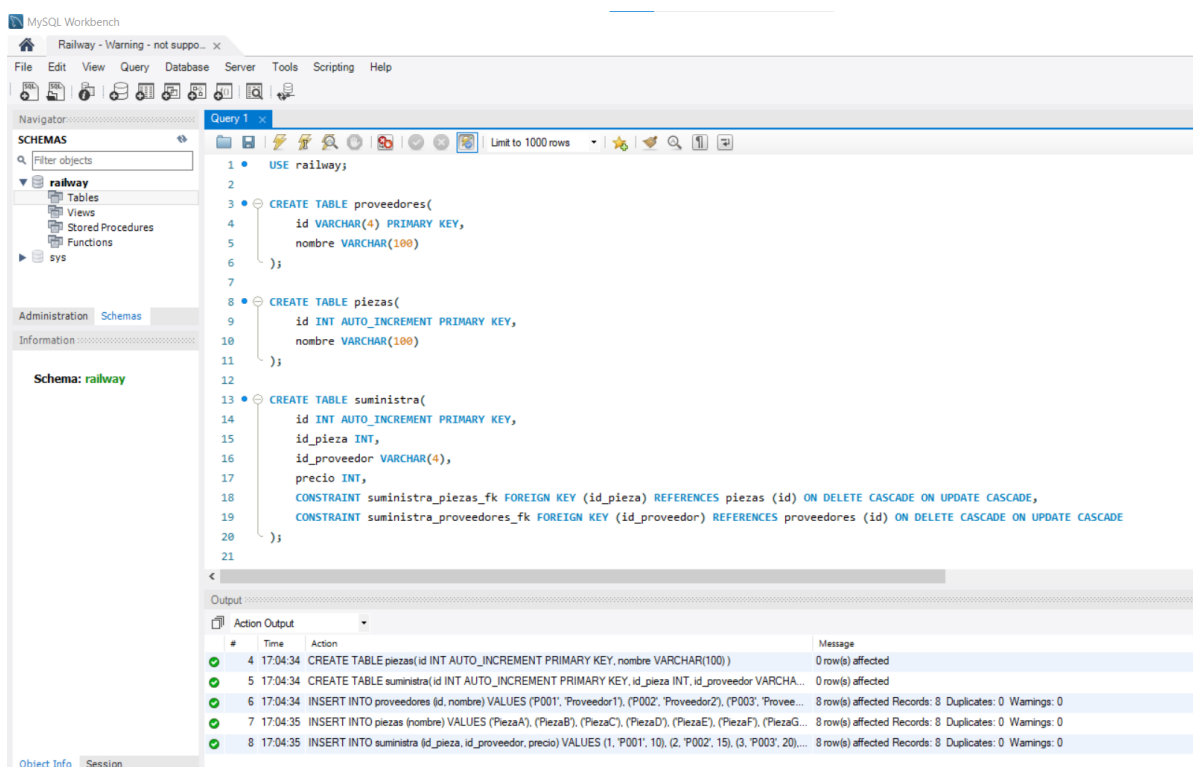
MYSQLDATABASE	*****	⋮
MYSQLHOST	<a href="#">roundhouse.proxy.rlwy.net</a>	⋮
MYSQLPASSWORD	*****	⋮
MYSQLPORT	*****	⋮
MYSQLUSER	*****	⋮
MYSQL_DATABASE	*****	⋮
MYSQL_PRIVATE_URL	*****	⋮
MYSQL_ROOT_PASSWORD	*****	⋮
MYSQL_URL	*****	⋮

## Creación de tablas e inserción de datos:

Una vez creada la base de datos abrimos MySQL Workbench y creamos una nueva conexión, para ello debemos introducir los datos de conexión recogidos de las Variables de entorno de la base de datos en Railway.



Una vez conectados, introducimos y ejecutamos el SQL con la creación de las tablas e inserciones de los datos de muestra.

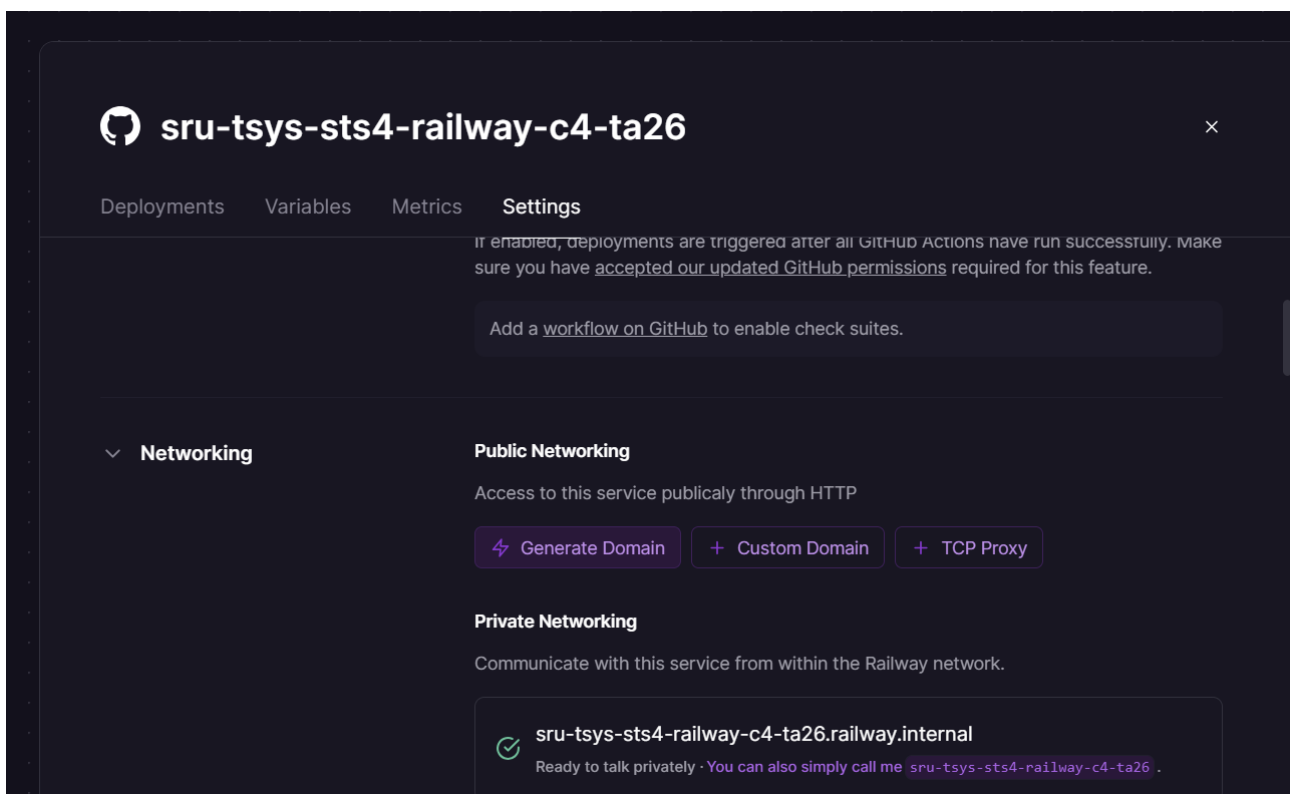


## Añadir la API del Ejercicio 1 a Railway:

Para ello primero parametrizaremos el puerto y la conexión con la base de datos en el `application.properties` de nuestro proyecto, se puede conectar mediante los datos de usuario y contraseña incluidos en la url de la base de datos o como parámetros aparte y subimos el proyecto a nuestro repositorio de GitHub.

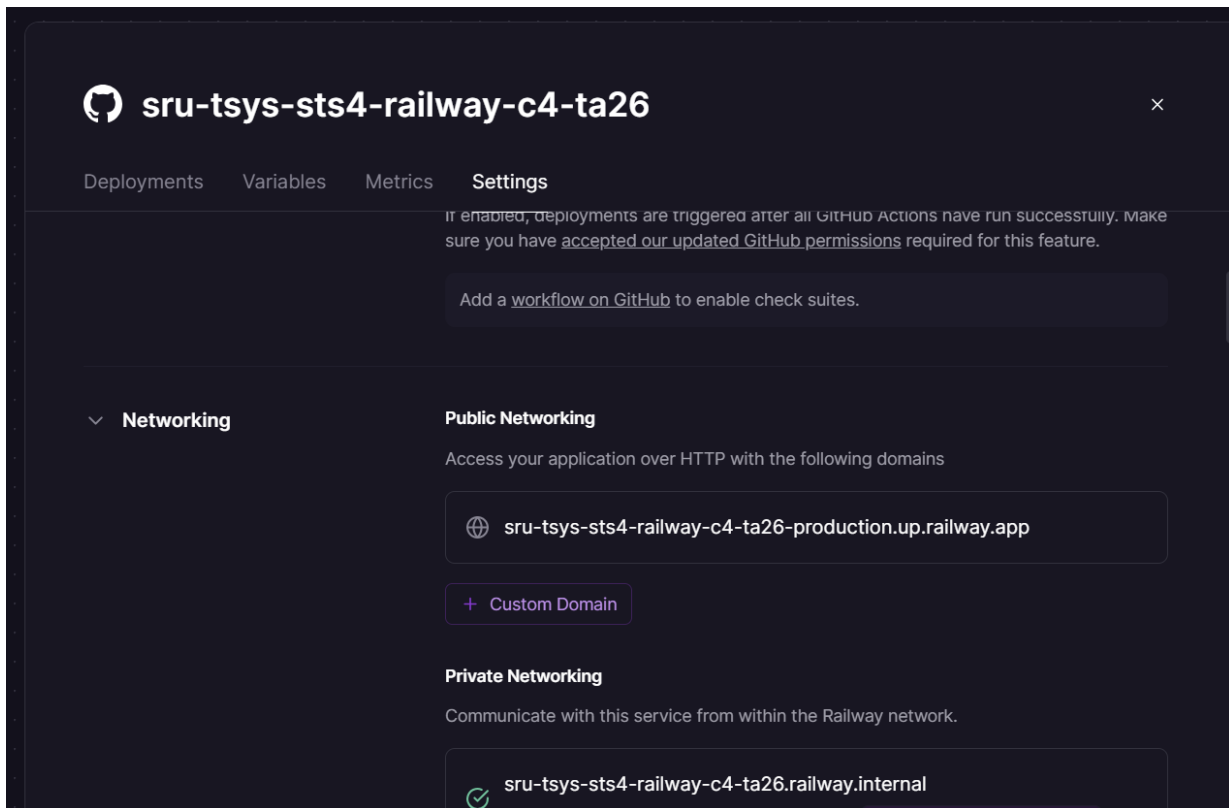
```
application.properties x
1 #Data source
2 #Indica el driver/lib para conectar java a mysql
3 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
4
5 #Url donde esta el servicio de tu mysql y el nombre de la base de datos ?serverTimezone=UTC
6 spring.datasource.url=${URL_DB}
7 #Usuario y contraseña para tu base de datos descrita en la linea anterior
8 #spring.datasource.username=${USER}
9 #spring.datasource.password=${PASSWORD}
10 |
11 #Imprime en tu consola las instrucciones hechas en tu base de datos.
12 #Configuraciones JPA
13 spring.jpa.show-sql=true
14 spring.jpa.open-in-view=true
15 spring.jpa.properties.hibernate.format_sql=true
16 #spring.jpa.hibernate.ddl-auto=create-drop
17
18 #Puerto del servidor Tomcat
19 server.port=${PORT}
```

Ahora nos dirigimos a nuestro proyecto de Railway y hacemos clic en el nuevo para crear el entorno de nuestra API. Seleccionamos GitHub Repo, buscamos el repositorio de nuestro proyecto de Spring y hacemos clic en él. Nos tendría que salir una ventana parecida a la siguiente.

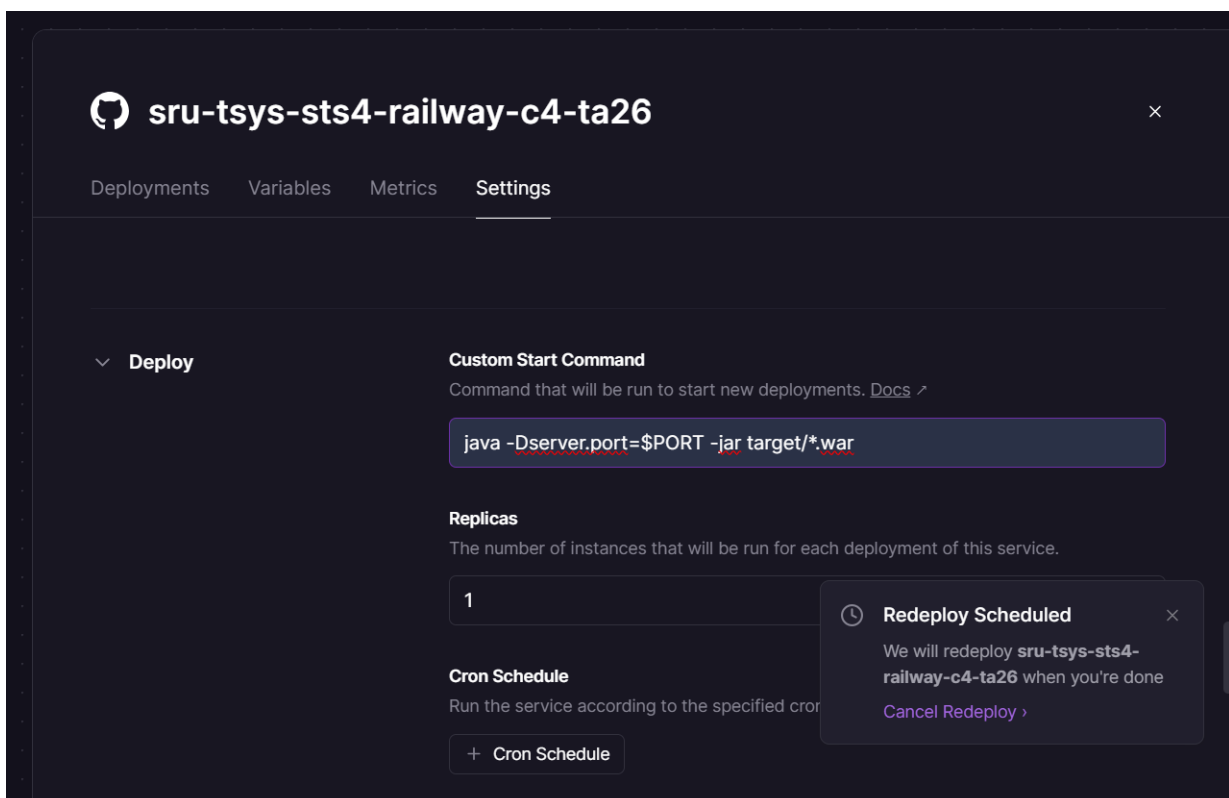


## Configurar el entorno de la API en Railway:

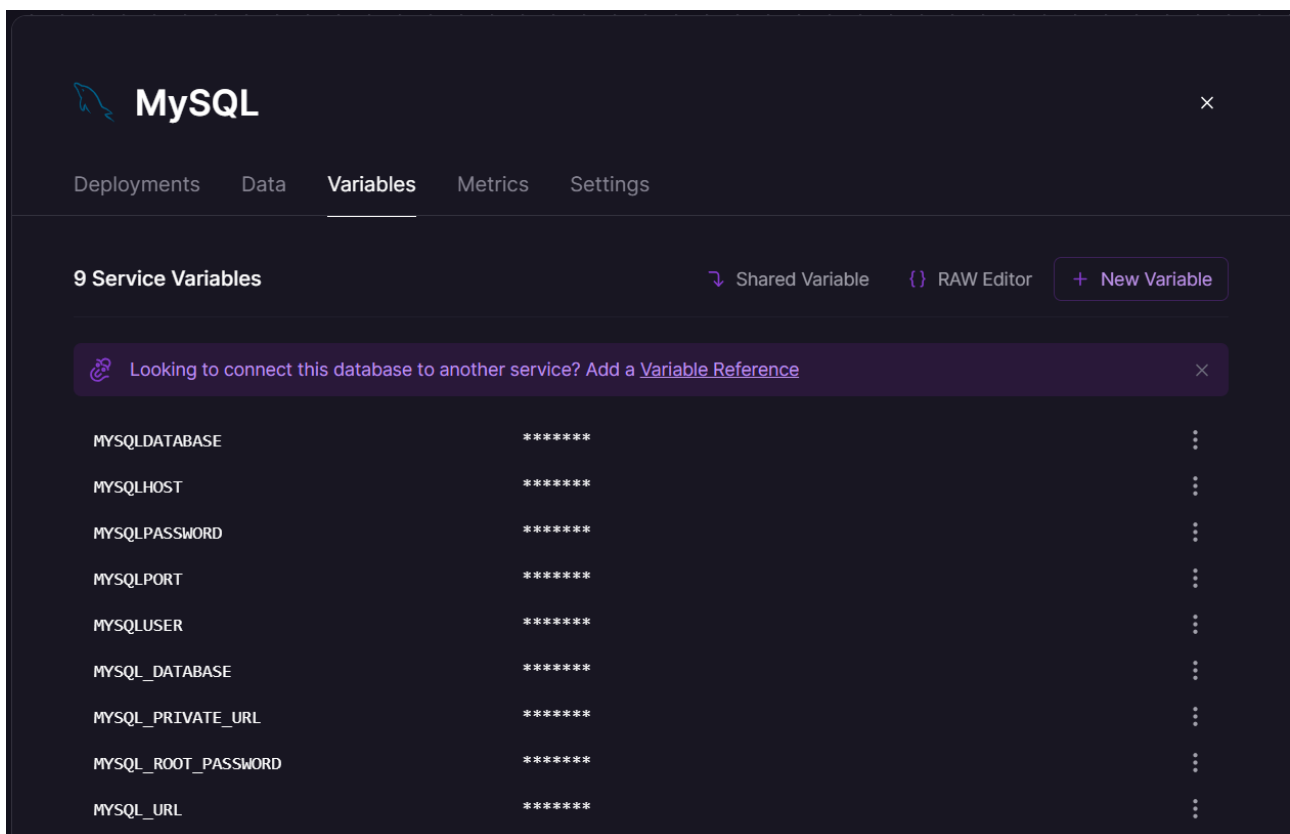
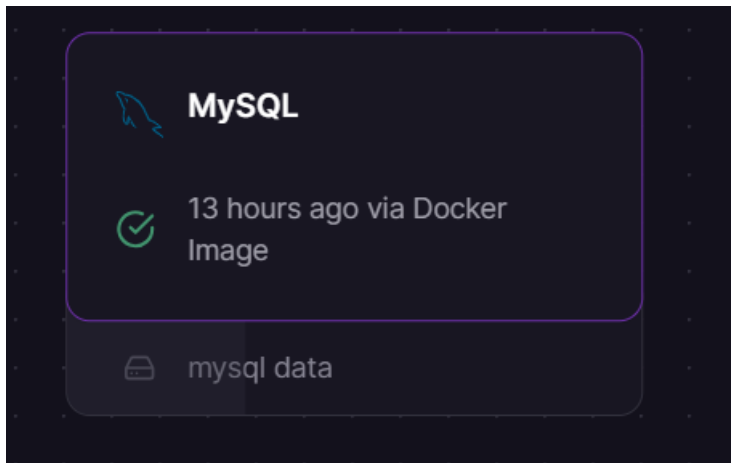
Primero nos dirigimos a la ventana de Settings y en la sección de Networking Añadimos un dominio personalizado haciendo clic en el botón de + Custom Domain.



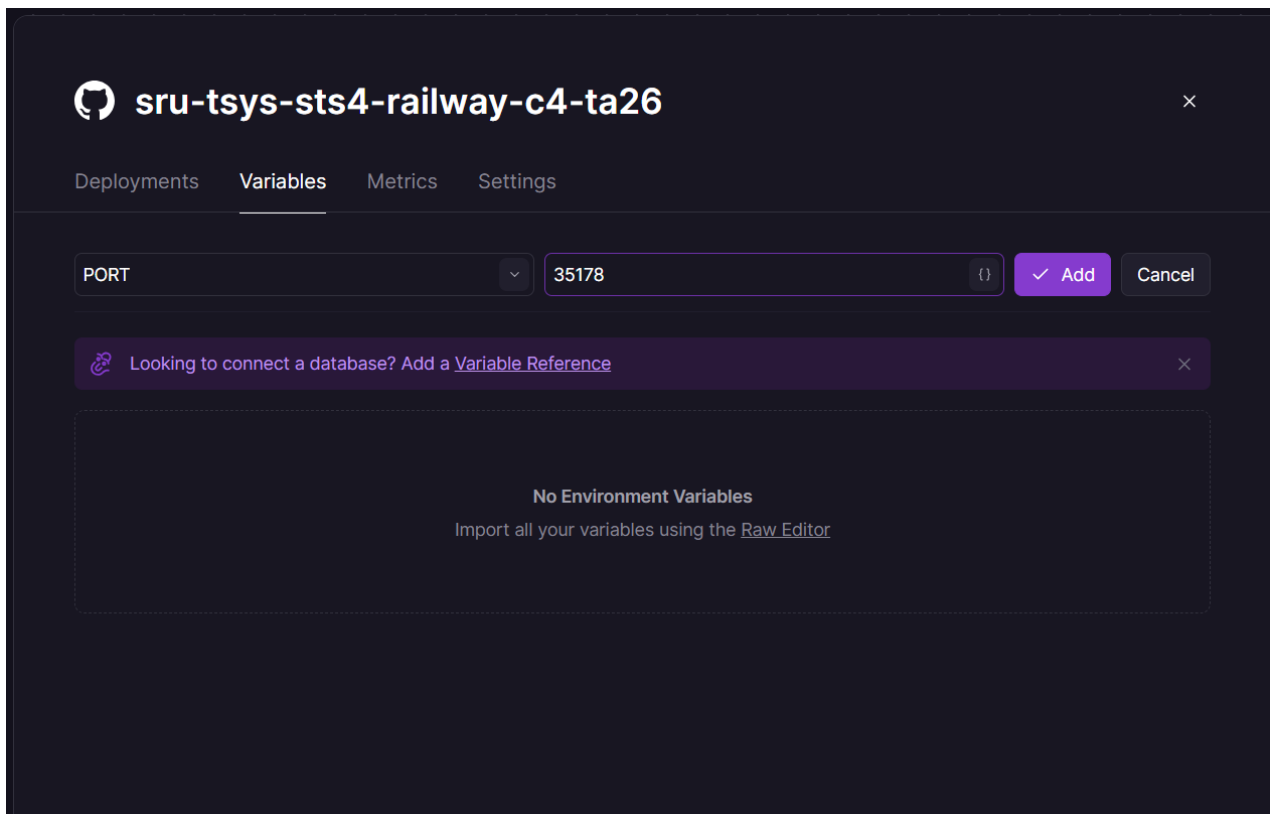
En la sección General podemos introducir la ruta del proyecto en Root Directory. Finalmente en la sección de Deploy añadimos el comando de inicio personalizado.



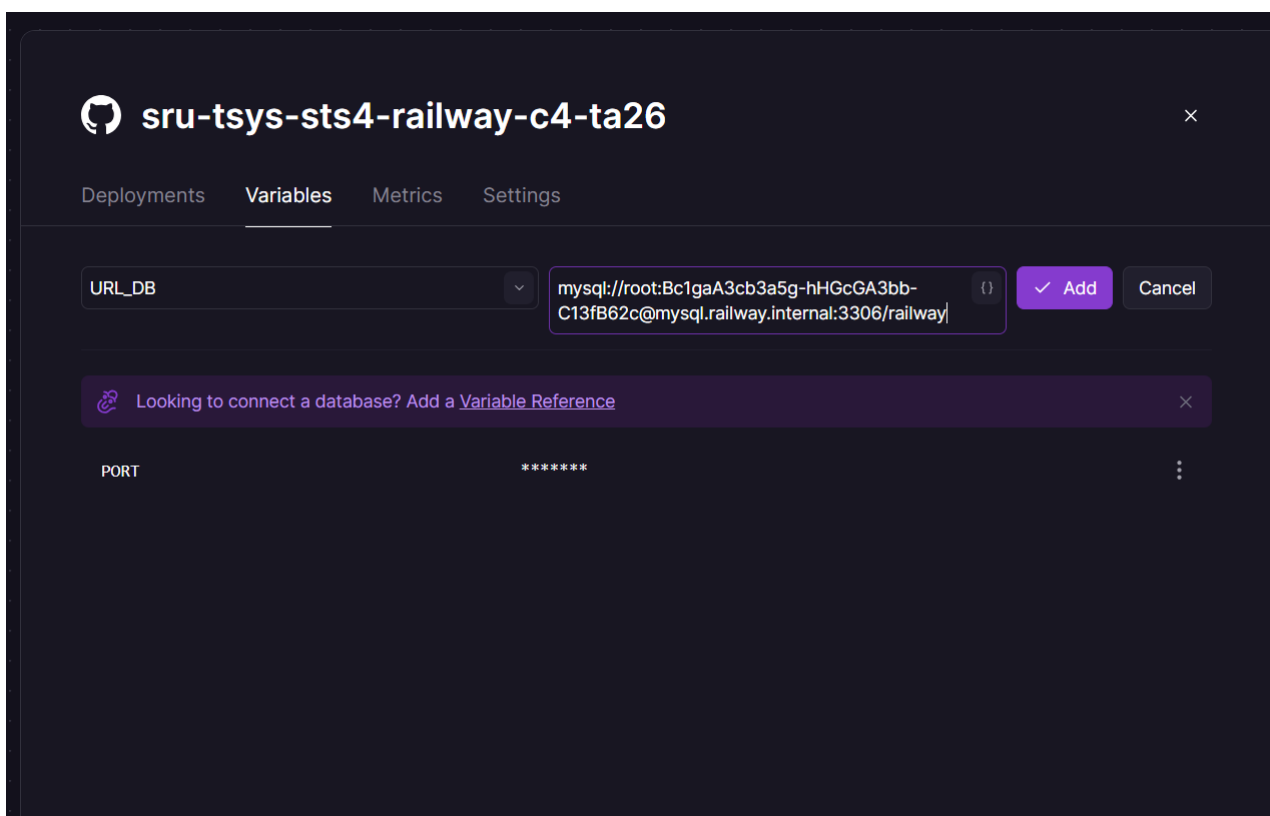
Una vez realizada la ventana de Settings tendremos que crear las variables de entorno que hemos especificando en el application.properties de nuestro proyecto, para ello primero debemos dirigirnos al apartado de Variables del proyecto MySQL en Railway para recoger el puerto y la url privada.



Una vez hemos recogido los datos nos dirigimos al entorno de la API de Spring en Railway, accedemos al apartado de Variables y creamos las variables de entorno correspondientes a nuestro proyecto.



Cabe destacar que en mi caso he añadido mysql:// al principio de la URL de MySQL.





Una vez realizados los pasos anteriores el proyecto ya estará configurado. El entorno hará automáticamente el Deploy ya que al realizar cualquier cambio se reinicia la API. En caso de que no lo realizara se puede reiniciar la aplicación con click derecho. El Log del deploy queda de la siguiente forma.

The screenshot shows the Railway dashboard interface. At the top, the app name is 'sru-tsys-sts4-railway-c4-ta26-production.up.railway.app' with a status of 'ACTIVE'. Below this, there are tabs for 'Details', 'Build Logs', and 'Deploy Logs', with 'Deploy Logs' being the active tab. A search bar is present with the text 'Filter logs using ""', (), AND, OR, -'. The log table has two columns: 'Timestamp' and 'Message'. The logs show the application starting at 06:07:37, displaying ASCII art, and then logging Spring Boot version 3.1.5. Subsequent logs show the application starting at 06:07:37, logging 'No active profile set, falling back to 1 default profile: "default"', and then logging 'Bootstrapping' and 'Finished Spring Data repository scanning in 88 ms. Found 3 JPA repository interfaces.' at 06:07:39.

Timestamp	Message
Nov 16 06:07:37	. _ _ _ _ _
Nov 16 06:07:37	/ \ / _ ' _ _ _ _ ( ) _ _ _ _ \ \ \ \
Nov 16 06:07:37	( ( ) \   ' _   ' _   ' _   ' _   \ \ \ \
Nov 16 06:07:37	W _ )                     (     ) ) ) )
Nov 16 06:07:37	' _     _                     / / / /
Nov 16 06:07:37	===== _ ===== _ -/ / / /
Nov 16 06:07:37	:: Spring Boot :: (v3.1.5)
Nov 16 06:07:37	2023-11-16T05:07:37.790Z INFO 1 --- [ main] c.e.demo.C4Ta26Pt1RailwayApplication : Starting C4Ta26Pt1RailwayApplication v0.0.1-SNAPSHOT using Java 17.0.7 with PID 1 (/app/target/C4-TA26-PT1-RAILWAY-0.0.1-SNAPSHOT.war started by root in /app)
Nov 16 06:07:37	2023-11-16T05:07:37.792Z INFO 1 --- [ main] c.e.demo.C4Ta26Pt1RailwayApplication : No active profile set, falling back to 1 default profile: "default"
Nov 16 06:07:38	2023-11-16T05:07:38.995Z INFO 1 --- [ main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
Nov 16 06:07:39	2023-11-16T05:07:39.094Z INFO 1 --- [ main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 88 ms. Found 3 JPA repository interfaces.

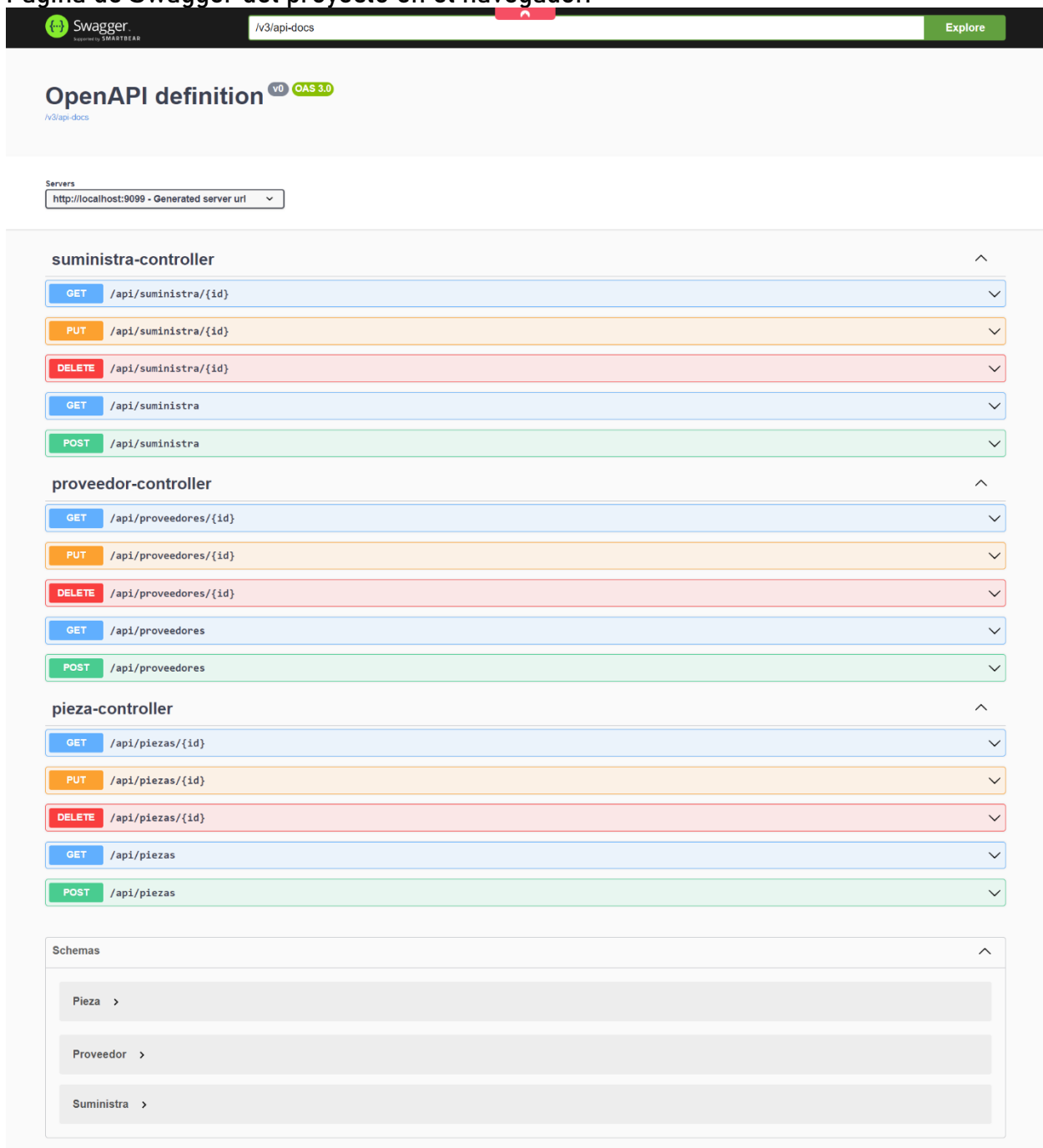
# Comprobación del correcto funcionamiento:

En mi caso en la aplicación de Spring he instalado Swagger por lo cual podremos acceder via web y así tener una documentación y testear la REST API.

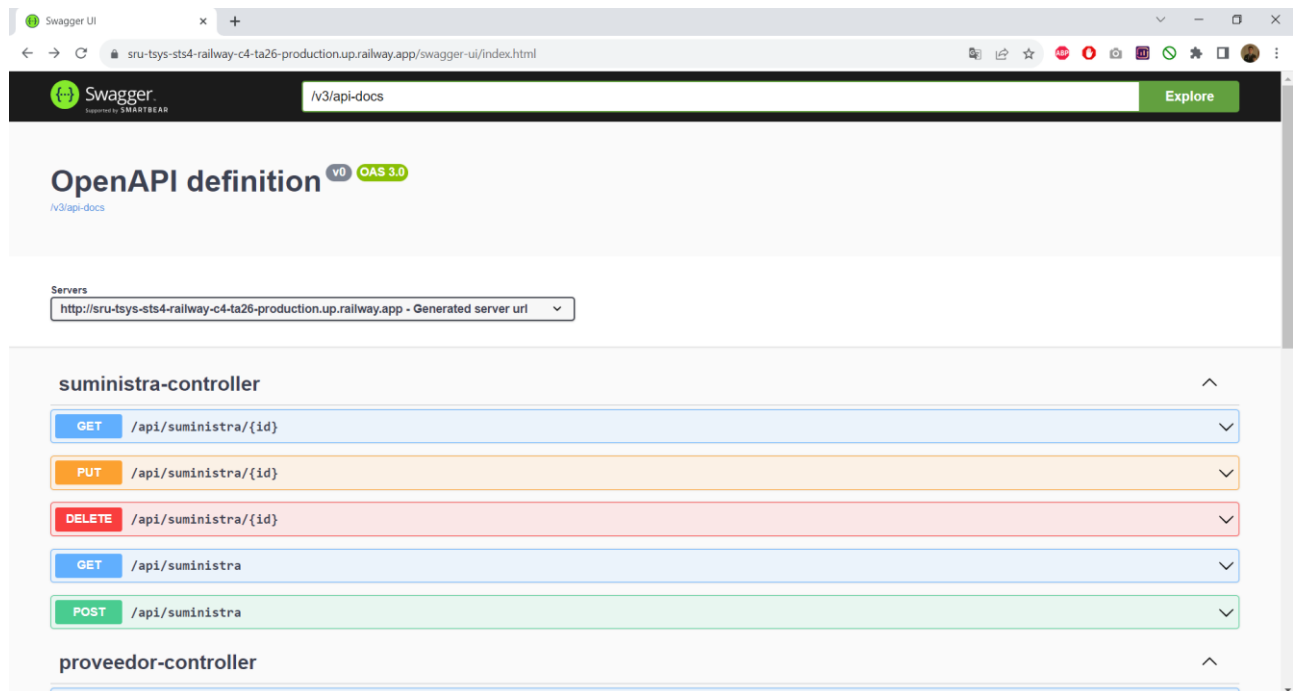
Añadida la siguiente dependencia en el pom.xml del proyecto:

```
<!-- localhost:xxxx/swagger-ui/index.html -->
<dependency>
  <groupId>org.springdoc</groupId>
  <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
  <version>2.2.0</version>
</dependency>
</dependency>
```

Página de Swagger del proyecto en el navegador:



Una vez subidos los cambios al repositorio de Github y reiniciado el entorno en Railway, accedemos a la url del dominio personalizado que hemos creado anteriormente en la configuración del entorno y añadimos la ruta de la página de swagger.



Probamos de acceder a un endpoint para comprobar su correcto funcionamiento. En este caso nos dará error del CORS aunque nuestra aplicación funcione correctamente en local ya que le falta una capa de seguridad la cual añadiremos en la siguiente actividad.

