Autor: Antonio Miguel Morales Caldero

Proceso de Despliegue de la Aplicación

Descripción General

Este documento describe el proceso de despliegue de mi aplicación de Spring Boot en Render y la configuración de la base de datos PostgreSQL en Railway.

Paso 1: Configuración de la Base de Datos en Railway

1. Fuí a Railway.app y creé una cuenta..
2. Creé un nuevo proyecto y seleccioné "Deploy PostgreSQL".
3. Obtuve las credenciales de la base de datos.
4. Puse las credenciales correctamente configuradas en application.properties.

Paso 2: Preparación del Proyecto

1. **Estructura del Proyecto**: Adaptar el proyecto, para ello ha habido que cambiar el application properties y poner credenciales de la base de datos de railway y quitar todo lo relacionado con el localhost que era donde estaba la base de datos antes.También ha habido que incluir en el pom varias dependencias como puede ser la del postgresql, y crear un archivo Dockerfile y un archivo render.yaml en el raíz del proyecto para que render pueda crear el servicio y de esta manera desplegar el proyecto.

Archivo pom.xml: Incluye la dependencia de PostgreSQL en tu pom.xml:
xml
Copiar código

```xml
<dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <version>42.5.4</version>
</dependency>
```

Paso 3: Configuración del Archivo application.properties

Configura el archivo application.properties para que utilice PostgreSQL:

```
spring.application.name=RA3RA7MoralesCalderoAntonioMiguel
spring.datasource.url=jdbc:postgresql://viaduct.proxy.rlwy.net:29547/railway
spring.datasource.username=postgres
spring.datasource.password=kHXQLjWUdsPWhDxIoINXzmrPViefuJQn
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.generate-ddl=true
spring.servlet.multipart.max-file-size=128KB
spring.servlet.multipart.max-request-size=128KB
spring.thymeleaf.prefix=classpath:/templates/
spring.thymeleaf.suffix=.html
spring.thymeleaf.mode=HTML
spring.mvc.format.date=yyyy-MM-dd
```

Paso 4: Creación del Archivo Dockerfile

```
# Usa una imagen base de Maven para construir la aplicación
FROM maven:3.8.4-openjdk-17 AS build
WORKDIR /app
COPY . .
RUN mvn clean package -DskipTests

# Usa una imagen base de OpenJDK para ejecutar la aplicación
FROM openjdk:17-jdk-slim
WORKDIR /app
COPY --from=build /app/target/*.jar app.jar
EXPOSE 8080
ENTRYPOINT ["java", "-jar", "app.jar"]
```

Paso 5: Configuración del Archivo render.yaml

Creé un archivo render.yaml en la raíz del proyecto:

```
services:
  - type: web
    name: RA3RA7MoralesCalderoAntonioMiguel
    runtime: docker
    envVars:
      - key: SPRING_PROFILES_ACTIVE
        value: prod
```

plan: free
repo: https://github.com/AntonioMoralesCaldero/RA3RA7MoralesCalderoAntonioMiguel
branch: Entrega3
dockerfilePath: Dockerfile

Paso 6: Despliegue en Render

1. Fuí a Render.com y creé una nueva cuenta.
2. Conecté mi cuenta de GitHub.
3. Seleccioné el repositorio y la rama Entrega 3.
4. Render detecta automáticamente el archivo render.yaml y configuró el servicio.
5. Añadí como variables de entorno las credenciales de la base de datos en remoto para el correcto funcionamiento de la aplicación.

Paso 7: Verificación y Pruebas

1. Accedí a la URL proporcionada por Render para verificar que la aplicación esté funcionando.
2. Realicé las pruebas básicas de CRUD para asegurarme de que la conexión a la base de datos funciona correctamente.

Conclusión

Siguiendo estos pasos, he desplegado exitosamente una aplicación de Spring Boot en Render con una base de datos PostgreSQL en Railway.

Por aquí te dejo la url para probar la web:

https://ra3ra7moralescalderoantoniomiguel.onrender.com