Practica Tolerancia a fallos

Segarra Ortiz Johnny David
Azuay, Ecuador
Email: jsegarrao1@est.ups.edu.ec

El código fuente completo del sistema, incluyendo los archivos de configuración, Dockerfiles y documentación, se encuentra disponible en el siguiente repositorio:

https://github.com/johnny567w/SegarraJ_Evaluacion/tree/main

I. Creación del Backend

Se desarrolló el backend del sistema de universidades con Spring Boot. Se implementó un controlador REST que expone el endpoint `/api/universidades`, el cual permite acceder a los registros almacenados en la base de datos. La conexión fue configurada exitosamente y se probó con resultados correctos.

```java
import java.util.List;
@RestController
@RequestMapping("/api/universidades")
@CrossOrigin(origins = "*")
public class UniversidadController {

    @Autowired
    private UniversidadRepository repo;

    @GetMapping
    public List<Universidad> listar() { return repo.findAll(); }
```

Fig. 1. Respuesta exitosa del endpoint /api/universidades

II. Dockerización del Backend

A. Creación del Dockerfile

Para ejecutar el backend en un contenedor, se generó un archivo `Dockerfile` que compila y ejecuta el artefacto `.jar` resultante del proyecto. El contenido puede verse a continuación.

```
application.properties   pom.xml (Practica11)   Dockerfile
1  FROM maven:3.9.2-eclipse-temurin-17 AS build
2  WORKDIR /app
3  COPY . .
4  RUN mvn clean package -DskipTests
5
6  FROM eclipse-temurin:17-jdk-alpine
7  WORKDIR /app
8  COPY --from=build /app/target/*.jar app.jar
9  EXPOSE 8080
10 ENTRYPOINT ["java", "-jar", "app.jar"]
11
12
```

Fig. 2. Contenido del Dockerfile en el backend

Previo a publicar la imagen, se verificó su ejecución de forma local utilizando un archivo `application.properties` adecuado.

```
spring.datasource.username=usuario1
spring.datasource.password=contraseña
spring.datasource.driver-class-name=org.postgresql.Driver
spring.jpa.hibernate.ddl-auto=none
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.properties.hibernate.jdbc.lob.non_contextual_creation=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
server.address=0.0.0.0
server.port=8080
```

Fig. 3. Prueba local del backend con propiedades personalizadas

B. Errores en Docker Compose

Durante la ejecución con Docker Compose surgió un error relacionado con el nombre de usuario para la conexión a PostgreSQL. Este problema fue resuelto revisando y corrigiendo las propiedades del backend.

```
2025-07-22T22:22:25.835Z ERROR 1 --- [0.0-8080-exec-1] o.h.engine.jdbc.spi.SqlExceptionHelper   : FATAL: password authentication failed for user "usuario1"
2025-07-22T22:22:25.852Z ERROR 1 --- [0.0-8080-exec-1] o.a.c.c.C.[.[.[/].[dispatcherServlet]    : Servlet.service() for servlet [dispatcherServlet] in context with path [] threw exception [Request processing failed: org.springframework.transaction.CannotCreateTransactionException: Could not open JPA EntityManager for transaction] with root cause

org.postgresql.util.PSQLException: FATAL: password authentication failed for user "usuario1"
        at org.postgresql.core.v3.ConnectionFactoryImpl.doAuthentication(ConnectionFactoryImpl.java:704) ~[postgresql-42.7.5.jar!/:42.7.5]
```

Fig. 4. Error por nombre de usuario incorrecto en Docker Compose

Luego de solucionarlo, los servicios fueron levantados correctamente.

```
C:\Users\Johnny\Desktop\SegarraJ_Unidad4\Back>docker-compose up -d
time="2025-07-22T17:21:01-05:00" level=warning msg="C:\\Users\\Johnny\\Desktop\\SegarraJ_Unidad4\\Back\\docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 15/15
 ✔universidades-db Pulled                                    18.2s
   ✔59e22667830b Pull complete                               5.8s
   ✔5b842c2e9435 Pull complete                               5.9s
   ✔3126a7e35885 Pull complete                               6.2s
   ✔a7b0939f022a Pull complete                               6.3s
   ✔946b7e7ef46f Pull complete                               7.1s
   ✔ac5fafa49b34 Pull complete                               7.3s
   ✔a33603fff7c3 Pull complete                               7.3s
   ✔354c5a87ab49 Pull complete                               7.4s
   ✔ce8b165d222a Pull complete                               15.4s
   ✔4ffc2edad12e Pull complete                               15.4s
   ✔3494bee228ec Pull complete                               15.5s
   ✔a0d99ae7ffef Pull complete                               15.5s
   ✔a056f7e6f752 Pull complete                               15.5s
   ✔f59058ae83b2 Pull complete                               15.5s
[+] Running 4/4
 ✔Network back_default          Created                      0.1s
 ✔Volume "back_pgdata"          Created                      0.0s
 ✔Container Universidad         Started                      1.3s
 ✔Container universidades-api   Started                      1.0s
```

Fig. 5. Ejecución exitosa de docker-compose

C. Publicación en Docker Hub

La imagen fue publicada en el repositorio personal del autor en Docker Hub para su uso posterior en Kubernetes.

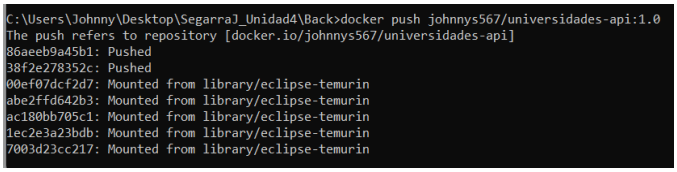

Fig. 6. Subida de la imagen del backend a Docker Hub

III. Despliegue en Kubernetes con Minikube

A. Transformación de manifiestos con Kompose

Se usó la herramienta Kompose para convertir el archivo `docker-compose.yml` a manifiestos de Kubernetes, facilitando así su despliegue.

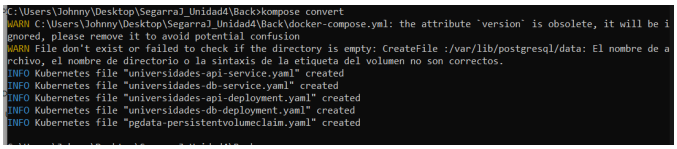

Fig. 7. Archivos `.yml` generados automáticamente por Kompose

B. Despliegue en Minikube y verificación

Los manifiestos generados fueron aplicados en Minikube. Se accedió al dashboard gráfico para verificar el estado de los pods, que se encontraban ejecutándose sin errores.

Fig. 8. Verificación del despliegue exitoso en Minikube Dashboard

C. Problemas de conexión iniciales

Al probar el backend desde Kubernetes, se encontró un error HTTP 500 en el endpoint `/api/universidades`. Esto indicaba una falla en la conexión con la base de datos.

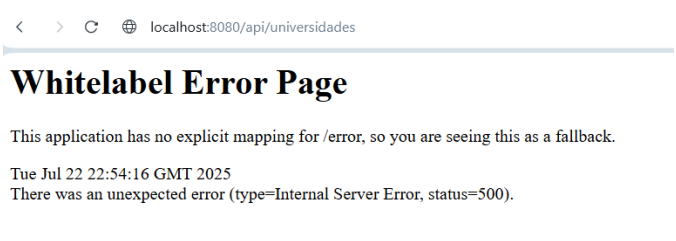

Fig. 9. Error 500 al acceder al endpoint desde Kubernetes

D. Solución del problema de conexión

Se ajustó la propiedad `spring.jpa.hibernate.ddl-auto` a `update` para forzar la actualización del esquema sin intentar crearlo nuevamente.

Fig. 10. Cambio de la propiedad ddl-auto a update

Finalmente, el error fue resuelto y la aplicación se conectó correctamente con la base de datos.

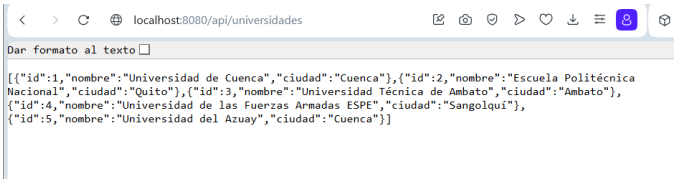

Fig. 11. Acceso exitoso a la base de datos desde Kubernetes