

ADAPTAR DOCKER A APLICACION SPRING BOOT Y MIGRAR DE H2 A POSTGRE POR PABLO ROYO.

FASE 1

En esta fase 1 realizaremos las adaptaciones necesarias para que nuestra aplicación funcione con PostgreSQL.

1. En la adaptación a nuestro proyecto, hay un fallo cuando pasamos de H2 a Postgre, el cual es crucial para que la BD se cree. El fallo es que en la tabla personas intentamos crearla y no es posible dado que la columna "user" es una palabra reservada y la hemos cambiado a "usuario".

```
//user es una keyword en postgres!!  
@Column(name = "usuario")  
private String user;
```

2. Posteriormente necesitamos crear un archivo llamado "Dockerfile" que es un archivo de texto plano que contiene una serie de instrucciones necesarias para crear una imagen docker.

```
FROM openjdk:17  
COPY /target/*.jar /usr/local/lib/spring.jar  
EXPOSE 8080  
ENTRYPOINT ["java", "-jar", "/usr/local/lib/spring.jar"]
```

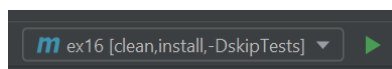
3. Posteriormente deberemos reconfigurar nuestro archivo "application.properties".

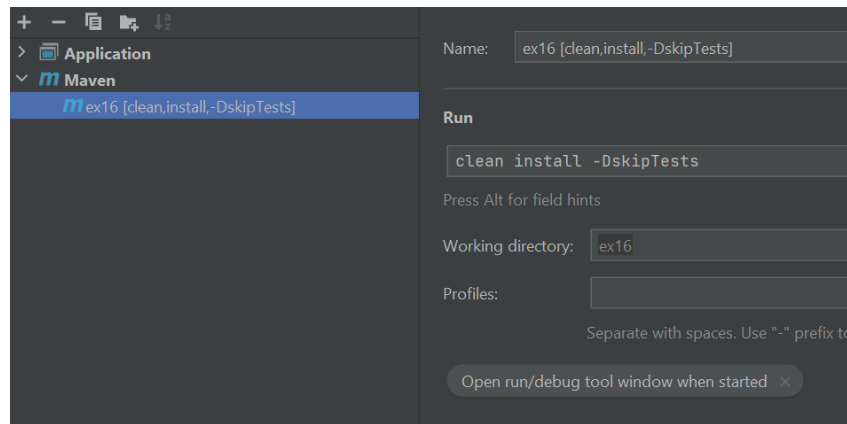
```
spring.jpa.properties.hibernate.dialect =  
org.hibernate.dialect.PostgreSQLDialect  
spring.jpa.hibernate.ddl-auto=update  
#spring.jpa.hibernate.show-sql=true  
spring.datasource.url=jdbc:postgresql://postgres_test:5432/postgres  
spring.datasource.username=postgres  
spring.datasource.password=contrasena  
spring.datasource.initialization-mode=always  
spring.datasource.initialize=true  
spring.datasource.continue-on-error=true
```

4. Añadiremos al **pom.xml** la correspondiente dependencia de postgres.

```
<dependency>  
  
    <groupId>org.postgresql</groupId>  
    <artifactId>postgresql</artifactId>  
    <scope>runtime</scope>  
  
</dependency>
```

5. Por último, configuraremos nuestro IntelliJ para que recompile el programa mediante Maven de la siguiente manera:





FASE 2

En esta fase 2 mostramos los comandos necesarios para que nuestra aplicación funcione correctamente, recordamos que tendremos 2 imágenes docker corriendo a la vez, una con la app de Spring y otra con el servidor de BD de Postgre.

FASE 2.1 NETWORK

Primero de todo, crearemos la red correspondiente, en la cual se ejecutarán ambas imágenes de docker (spring y postgre), para ello usaremos el siguiente comando;

```
docker network create mynetwork
```

FASE 2.2 POSTGRE

Posteriormente creamos la imagen de postgre, la cual se configurará y descargará automáticamente mediante el siguiente comando.


```
docker run --network mynetwork --name postgres_test -ePOSTGRES_USER=postgres -e  
POSTGRES_PASSWORD=contrasena -e POSTGRES_DB=test -p5432:5432 postgres
```

FASE 2.3 SPRING BOOT


1. Ejecutamos el botón de Maven en IntelliJ para generar el jar en la carpeta target.
2. Ejecutamos el comando “`docker build -t spring .`” desde la carpeta raíz del proyecto en el cmd para crear la imagen de docker que posteriormente ejecutaremos.
3. Por último, ejecutaremos el comando “`docker run --network mynetwork --name programa_spring -p8080:8080 spring`”


FIN


Para finalizar comprobaremos que funcionan las peticiones HTTP mediante Postman y que se listan las imágenes y contenedores en la aplicación de Docker.


 docker


Upgrade







 Sign in









Images on disk

2 images

Total size: 894.52 MB


IN USE

UNUSED

Clean up...

LOCAL

REMOTE REPOSITORIES

 Search

☐ In Use only

NAME ↑		TAG	IMAGE ID	CREATED	SIZE	
postgres	<div>IN USE</div>	latest	f98d2cc7f971	1 day ago	374.21 MB	<div><div></div><div>RUN ▶</div></div>
spring	<div>IN USE</div>	latest	9e19f19c5d95	less than a minute ago	520.31 MB	