Guía a usar para crear SERVIDOR: SPRING BOOT con KOTLIN y PostgreSQL

CREACIÓN DE BASE DE DATOS DE PRUEBA:

Base de datos:
Host: node180456-laboratorio-opoinf.mircloud.us
Port: 11015
Login: webadmin
Password: LMKcdv60734

PROCEDIMIENTO:

- **Base de datos PostgreSQL en MirHosting**
- **Usando una estructura base de Spring Boot con KOTLIN:**
  - **Usando** Spring Initializr
  - Una estructura similar en Git: EN ESTE CASO Git : opoinffrancisco/laboratorio-opoinf at devBase (github.com)
- **Desarrollar el código e implementar librerías (actualmente):** opoinffrancisco/laboratorio-opoinf: Servidor para el proyecto: SPRING BOOT con KOTLIN (github.com)

1. Debuggear con **postman**
2. Compilar **archivo. Jar (termina creado en "/directorio-proyecto/Build/libs/")**
3. Crear archivo **Dockerfile**
4. Crear archivo docker-compose.yml
5. Realizar **debuggeo** con Docker-setup para compilar y levantar una imagen para probar localmente
6. Se debe crear ambiente en **MirHosting** con imagen de la tienda, su nombre "**Docker Engine CE**"
7. Instalar **JAVA (en este caso la versión 17)** para cuando haya problemas al instalar librerías con los gestores tradicionales (dnf, `apt, yum, curl, etc...`):

1 - Download the Binary: Go to the Adoptium release page and download the tar.gz for Java 17:

```
wget https://github.com/adoptium/temurin17-binaries/releases/download/jdk-17.0.11%2B9/OpenJDK17U-jdk_x64_linux_hotspot_17.0.11_9.tar.gz
```

2 - Extract the Binary: Extract the downloaded tar.gz file:

```
tar -xzf OpenJDK17U-
jdk_x64_linux_hotspot_17.0.11_9.tar.gz
```

3 - Move to a Suitable Location: Move the extracted files to `/usr/local`:

```
sudo mv jdk-17.0.11+9 /usr/local/java-17
```

4 - Set Up Environment Variables: Add Java to your PATH and set JAVA_HOME:

```
echo "export JAVA_HOME=/usr/local/java-17" | sudo
tee -a /etc/profile
echo "export PATH=\$PATH:\$JAVA_HOME/bin" | sudo tee
-a /etc/profile
source /etc/profile
```

5 - Verify Installation: Check the installed version of Java:

```
java -version
```

- **Instalar PostgreSQL en el ambiente (OPCIONAL):**

  yum install postgresql
  También se puede hacer manual, pero esto solo es para comprobar si se puede conectar a la base de datos desde el ambiente

- **Clonar el repositorio del proyecto con:**

git clone URL https://github.com/opoinffrancisco/laboratorio-opoinf.git

- Colocar las **variables de entorno** en el ambiente de **MirHosting** creado para el proyecto con **Docker Engine CE**. Se pude agregar a las variables de entornos por el gestor o por el archivo .jelenv ubicado en la raiz del ambiente donde se encuentran las carpetas del sistema operativo LINUX

```
#VARIABLES DE ENTORNO

DB_DDLAUTO=update
DB_DRIVER=org.postgresql.Driver
DB_PASSWORD=LMKcdv60734
DB_PLATAFORM=org.hibernate.dialect.PostgreSQLDialect
DB_SHOWSQL=true
DB_URL=jdbc:postgresql://node180456-laboratorio-opoinf.mircloud.us/laboratorio-opoinf
DB_USERNAME=webadmin

JWT_ACCESS_TOKEN_EXPIRATION=3600000
JWT_KEY=12345678910123456789101234567891023
JWT_REFRESH_TOKEN_EXPIRATION=7200000

MAIL_HOST=smtp.gmail.com
MAIL_PASSWORD=wxomaipjcyanhiqq
MAIL_PORT=587
MAIL_SMTPAUTH=true
MAIL_SMTPSTARTTLSENABLE=true
MAIL_USERNAME=soporte.opoinf@gmail.com
```

- **Si llega a ser necesario para eliminar todo lo hecho con la imagen,usar:**

  `docker-compose down --rmi all`

- **Así luego podrás intentar nuevamente:**

  docker-compose build
  docker-compose up (SOLO PARA HACER PRUEBAS)
  docker-compose ps
  docker-compose logs

- **CAMBIAR DE 32 CLOUDLETS A 2 CLOUDLETS.**
  (32 SON INSTALADOS POR DEFECTO) (PARA QUE FUNCIONE GRATIS)

  docker-compose run -d laboratorio-opoinf -p 8080:8080

- **LUEGO ESPERAR 3 A 7 HORAS PARA QUE LEVANTE COMPLETAMENTE EL SERVICO EN EL AMBIENTE.** (PARA QUE FUNCIONE GRATIS)

- Comprobar con Postman: JSON PARA CONFIGURAR PRUEBAS

  https://api.postman.com/collections/10278539-e28cbd20-790c-4246-9b16-81bc73ff7ca4?access_key=PMAT-01J4MCXXGHFPCACX78P1MYQAQY