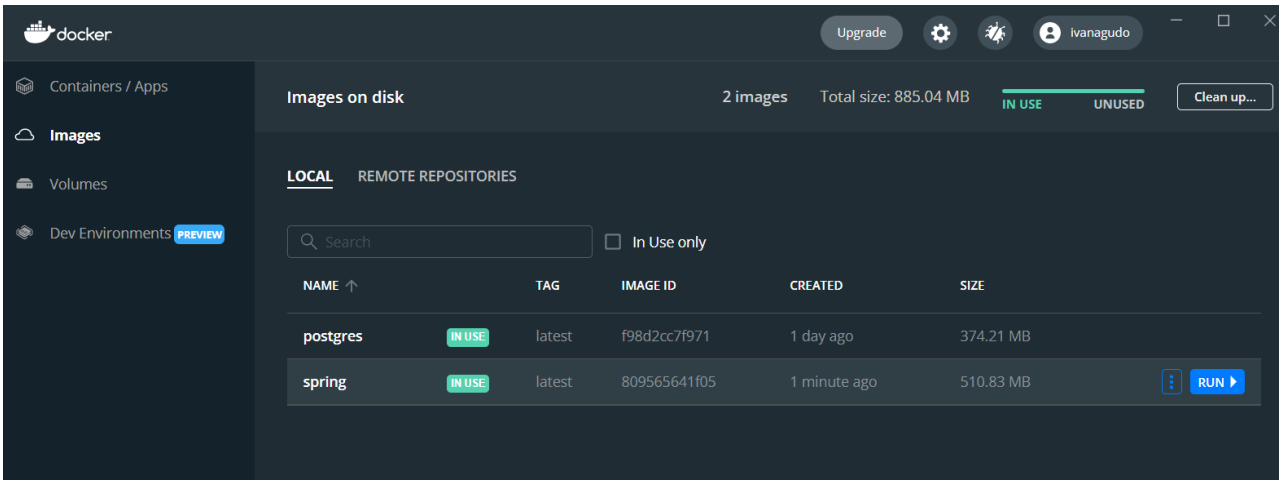
Documentación Docker

Dockerizamos el postgre

Tenemos que tener dos imágenes de Docker, una con postgre y otra con nuestro programa en spring



Este comando te bajará la imagen de postgre. para dockerizar postgre tendremos que ejecutar primero:

```
docker run postgres
```

Ahora creamos una network:

```
docker network create mynetwork
```

```
C:\Users\ivan.agudo>docker network create mynetwork
386e63ce83b592ac1bc10d0a2660872891c414f48c556712b3865f40a6cfe21f
```

Ejecutas el servidor de postgre:

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

```
C:\Users\ivan.agudo>docker run --network mynetwork --name postgres_test -ePOSTGRES_USER=postgres -e POSTGRES_PASSWORD=contrasena -e
POSTGRES_DB=test -p5432:5432 postgres
The files belonging to this database system will be owned by user "postgres".
This user must also own the server process.

The database cluster will be initialized with locale "en_US.utf8".
The default database encoding has accordingly been set to "UTF8".
The default text search configuration will be set to "english".

Data page checksums are disabled.

fixing permissions on existing directory /var/lib/postgresql/data ... ok
creating subdirectories ... ok
selecting dynamic shared memory implementation ... posix
selecting default max_connections ... 100
selecting default shared_buffers ... 128MB
selecting default time zone ... Etc/UTC
creating configuration files ... ok
running bootstrap script ... ok
performing post-bootstrap initialization ... ok
syncing data to disk ... initdb: warning: enabling "trust" authentication for local connections
You can change this by editing pg_hba.conf or using the option -A, or
--auth-local and --auth-host, the next time you run initdb.
ok
```

Dockerizamos el spring

Sobre el DB1-CRUD añades la dependencia de postgresql en el pom:

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

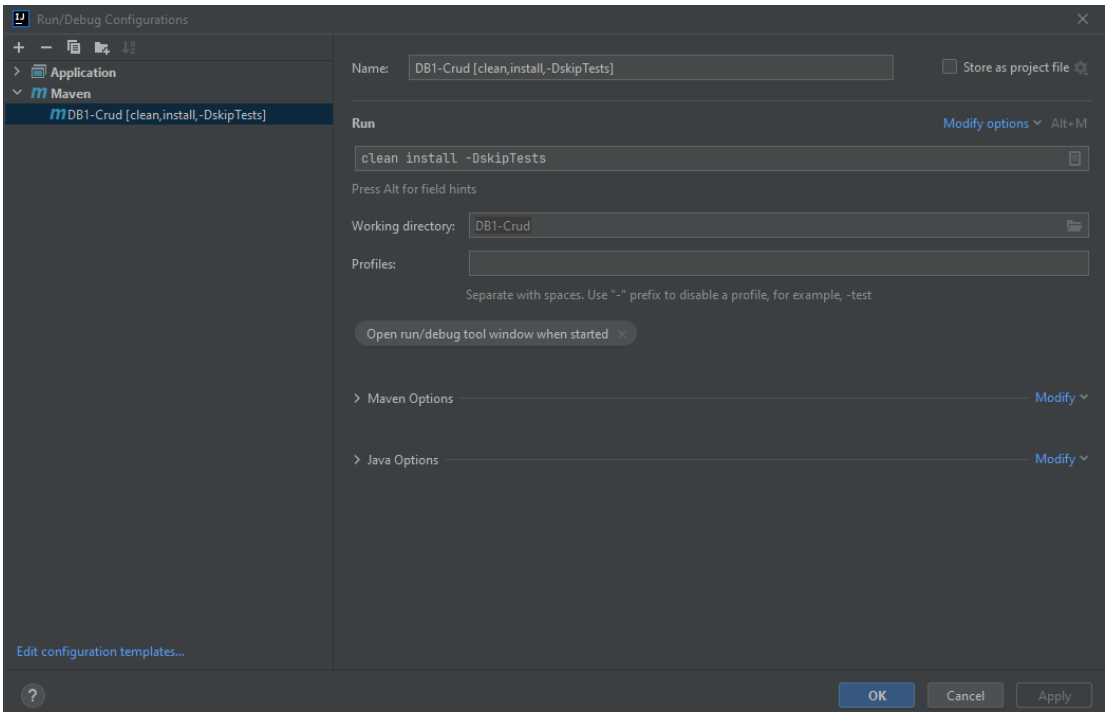
Añadimos las propiedades necesarias:

```
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
```

Tenemos que hacer un Dockerfile en la carpeta raíz del proyecto con el siguiente contenido:

```
FROM openjdk:17
COPY /target/*.jar /usr/local/lib/spring.jar
EXPOSE 8081
ENTRYPOINT ["java", "-jar", "/usr/local/lib/spring.jar"]
```

Te haces una configuración nueva para que te genere un jar:



Nos posicionamos en la ruta del jar y vamos a la carpeta justo anterior y ejecutamos el siguiente comando:

```
docker build -t spring .
```

```
C:\Users\ivan.agudo\Documents\ProyectosEjercicios\DB1-Crud\DB1-Crud>docker build -t spring .
[+] Building 2.6s (8/8) FINISHED
 => [internal] load build definition from Dockerfile                                                0.0s
 => => transferring dockerfile: 171B                                                                0.0s
 => [internal] load .dockerignore                                                                   0.0s
 => => transferring context: 2B                                                                     0.0s
 => [internal] load metadata for docker.io/library/openjdk:17                                       1.7s
 => [auth] library/openjdk:pull token for registry-1.docker.io                                      0.0s
 => [internal] load build context                                                                   0.3s
 => => transferring context: 39.65MB                                                                0.3s
 => CACHED [1/2] FROM docker.io/library/openjdk:17@sha256:2c3ffd17a95947a187cbf00b454656566063cc53f21ea419929c0d2  0.0s
 => [2/2] COPY /target/*.jar /usr/local/lib/spring.jar                                              0.2s
 => exporting to image                                                                              0.2s
 => => exporting layers                                                                             0.2s
 => => writing image sha256:809565641f0572d31306a1cbf5d2277fe88642dd1d96918f09d46e849c0819bb        0.0s
 => => naming to docker.io/library/spring                                                           0.0s

C:\Users\ivan.agudo\Documents\ProyectosEjercicios\DB1-Crud\DB1-Crud>docker run --network mynetwork --name programa_sprin
g -p8080:8080 spring

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.6.3)

2022-03-03 12:36:45.203  INFO 1 --- [           main] com.example.DB1Crud.Db1CrudApplication   : Starting Db1CrudApplica
tion v0.0.1-SNAPSHOT using Java 17.0.2 on 72dccb1caf00 with PID 1 (/usr/local/lib/spring.jar started by root in /)
2022-03-03 12:36:45.205  INFO 1 --- [           main] com.example.DB1Crud.Db1CrudApplication   : No active profile set,
falling back to default profiles: default
2022-03-03 12:36:45.789  INFO 1 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Da
ta JPA repositories in DEFAULT mode.
2022-03-03 12:36:45.844  INFO 1 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data re
pository scanning in 48 ms. Found 3 JPA repository interfaces.
2022-03-03 12:36:46.355  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with
```

Con eso ya podemos realizar peticiones desde postman en nuestra maquina Windows y que sean respondidas por nuestras apps dockerizadas.