

CASO DE ESTUDIO 1: DOCKER

Angie Ayala Sosa

Julián Rodrigo Caro

Yilber Alejandro Pérez

Luis Alejandro Molina

Ingeniero: Carlos Manrique

Universidad Pedagogica y Tecnologica de Colombia

Facultad de Ingeniería

Tunja, Colombia

DOCKER

- **¿QUE ES?**

Docker es una tecnología de contenedores que nos permite crear y utilizar contenedores, es una plataforma de software que permite a los desarrolladores crear, probar e implementar rápidamente aplicaciones en contenedores. Un contenedor es una unidad de software que contiene todo lo necesario para ejecutar una aplicación, incluido código, bibliotecas y archivos de configuración.

El término "Docker" se aplica a varios conceptos, incluido el proyecto comunitario de código abierto y sus herramientas, Docker Inc. (la empresa principal detrás del proyecto) y herramientas construidas a partir de contenedores de Linux. La tecnología es simple y única y brinda a los usuarios un acceso sin precedentes a las aplicaciones, una implementación rápida y control sobre las versiones y la distribución.

- **¿PARA QUE SIRVE?**

.Algunos de los principales usos, beneficios y fines de Docker son:

Portabilidad: los contenedores Docker son livianos y portátiles, lo que significa que las aplicaciones pueden ejecutarse en cualquier máquina que tenga Docker instalado, independientemente de las diferencias en los entornos de ejecución.

Eficiencia: Docker utiliza una arquitectura de microservicios, lo que permite un enfoque más granular y controlable, priorizando la eficiencia en el desarrollo y la implementación de aplicaciones.

Aislamiento: los contenedores Docker proporcionan un entorno aislado para ejecutar aplicaciones, lo que significa que no afectarán a otras aplicaciones ni al sistema operativo subyacente.

Escalabilidad: Docker facilita la implementación y escalamiento de aplicaciones en cualquier entorno, ya sea una implementación a pequeña escala o una aplicación empresarial a gran escala. Control de versiones: Docker proporciona control granular sobre las versiones y distribución de las aplicaciones, lo que facilita la gestión del ciclo de vida de las aplicaciones.

Implementar aplicaciones: Docker facilita la implementación de aplicaciones en cualquier entorno, ya sea local, en la nube o en un entorno híbrido.

Automatización: Docker se puede utilizar para automatizar tareas de desarrollo y operaciones, como crear imágenes, ejecutar contenedores y administrar contenedores.

Seguridad: Docker se puede utilizar para aislar aplicaciones entre sí, lo que puede ayudar a mejorar la seguridad.

- **PRINCIPALES CONCEPTOS**

Imagen de Docker: un archivo que contiene todo lo necesario para ejecutar una aplicación en un contenedor Docker, incluyendo el código, las bibliotecas, las herramientas y los archivos del sistema

Docker Container: Es una instancia en ejecución de una imagen de Docker. Los contenedores son entornos aislados que ejecutan aplicaciones y no afectan a otras aplicaciones o al sistema operativo subyacente

Dockerfile: Es un archivo de texto que contiene instrucciones para crear una imagen de Docker. Las imágenes de Docker son plantillas para la creación de contenedores

Docker Hub: Repositorio de imágenes Docker público y privado.

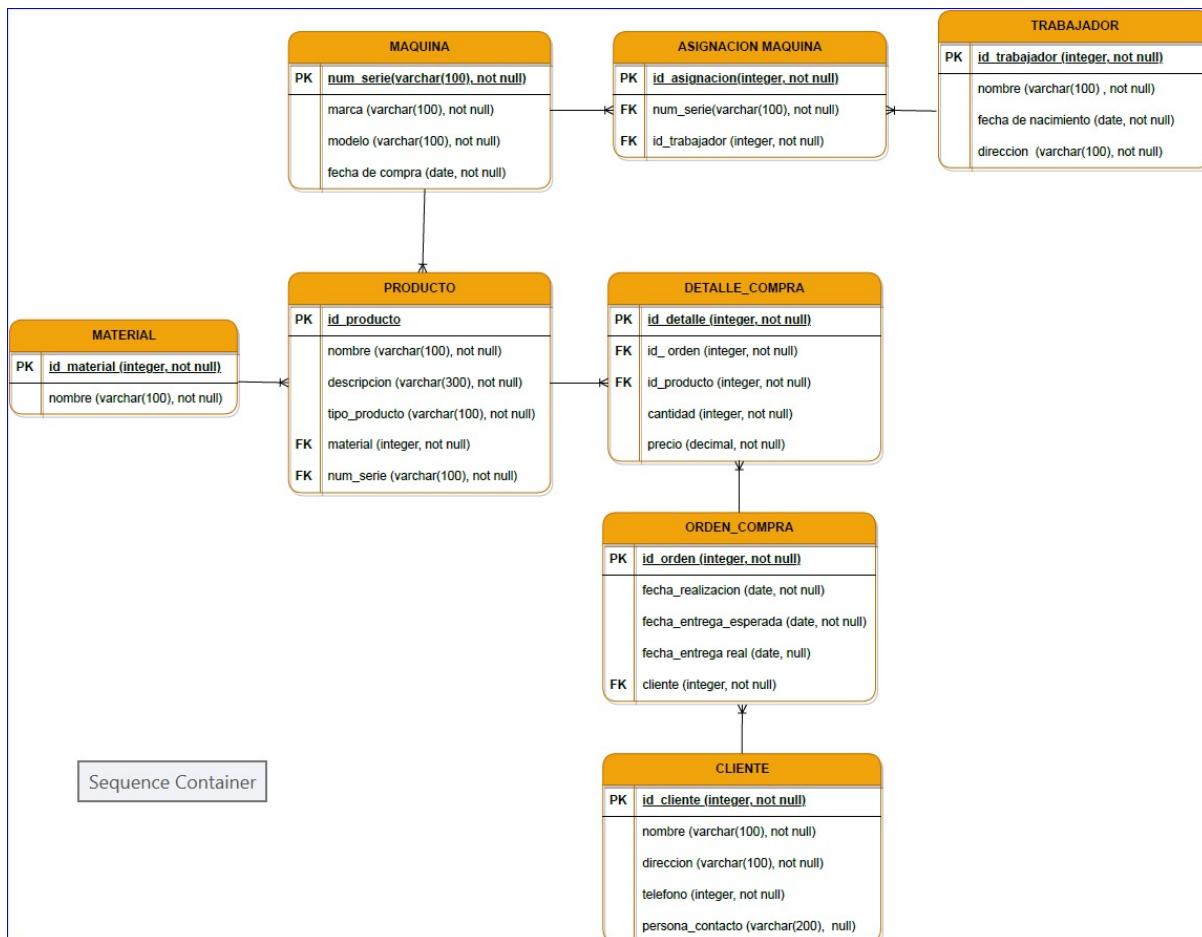
Docker Terminal: una herramienta de línea de comandos que se utiliza para interactuar con Docker.

- **DEMOSTRACIÓN DEL CASO DEL ESTUDIO**

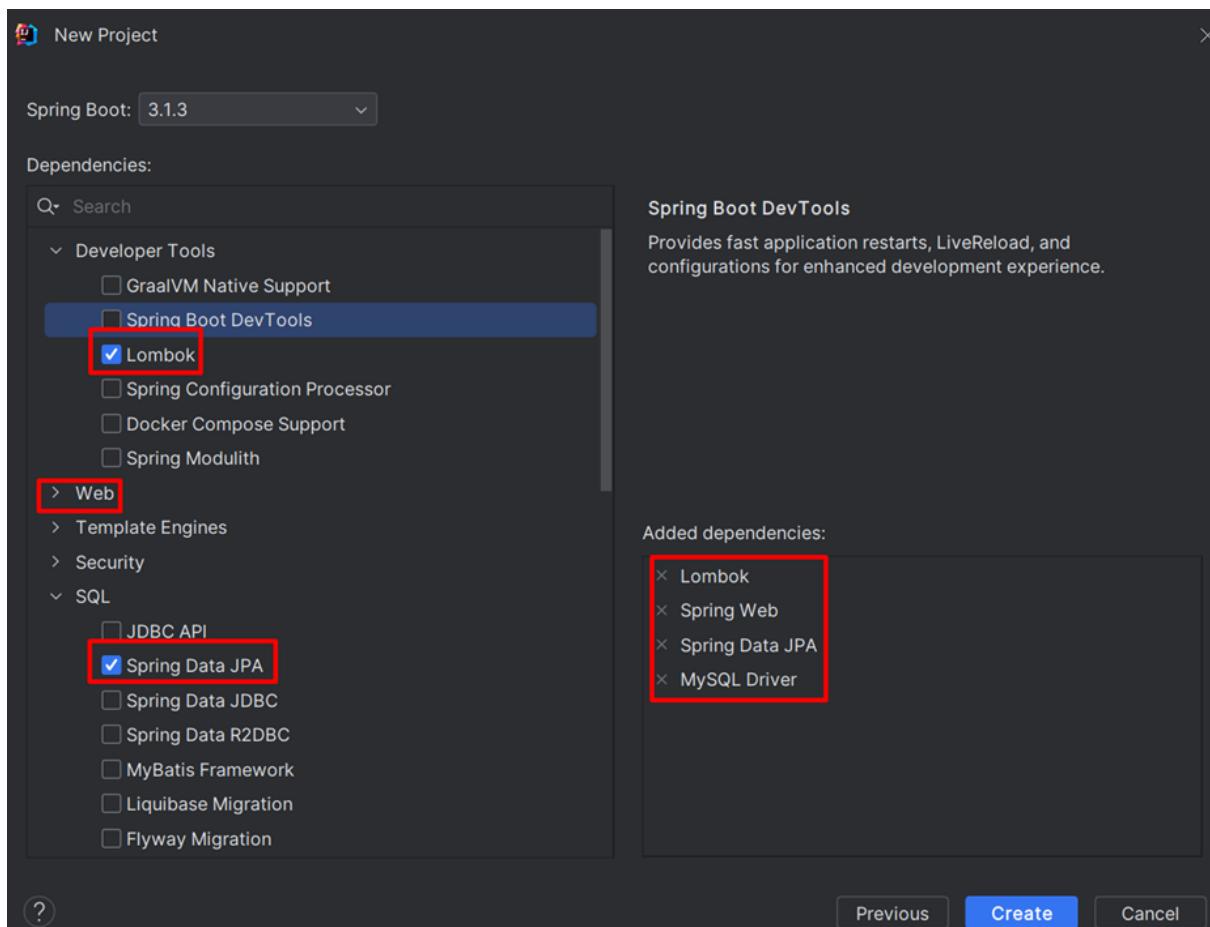
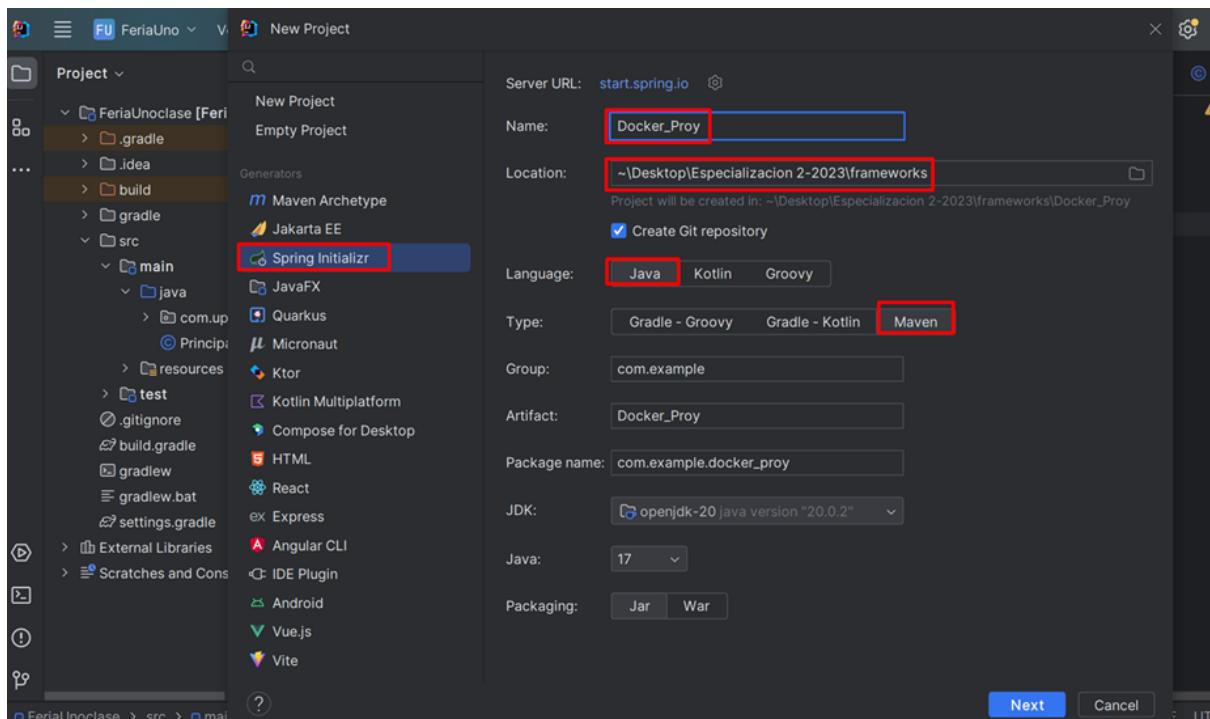
21 de Septiembre de 2023

relaciones:

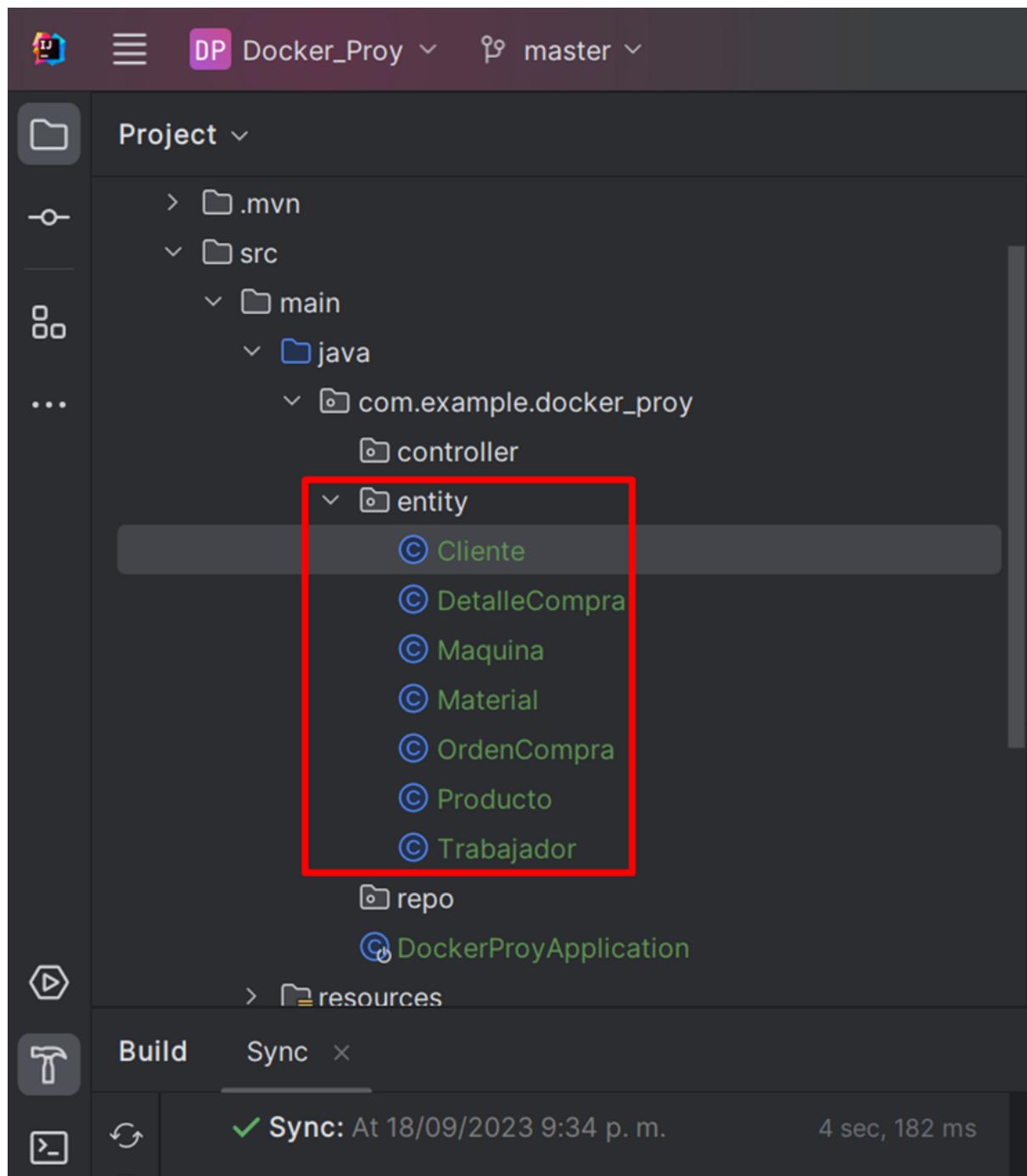
Para iniciar con la implementación del caso de estudio es necesario realizar la construcción del modelo entidad-relación del problema propuesto para esto se identifican 8 entidades y sus respectivos atributos, también las relaciones que existen entre ellos a continuación se muestra el modelo diseñado.

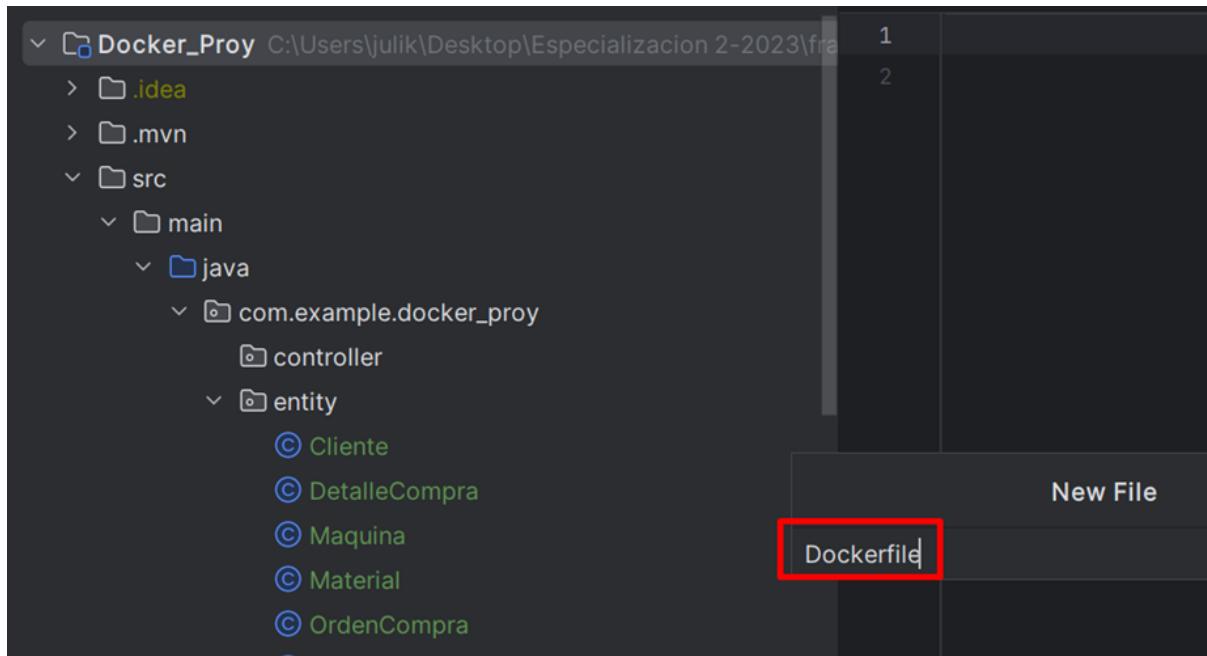


Para implementar el modelo se debe crear un proyecto en IntelliJ IDEA, para esto es necesario abrir IntelliJ IDEA, en la pantalla de bienvenidas se crea nuevo proyecto, en la ventana Crear nuevo proyecto, se selecciona el tipo de proyecto que desea crear. Para este caso de estudio, selecciona Java, en la ventana Crear nuevo proyecto, ingrese el nombre del proyecto y la ubicación donde desea guardarlo. Se selecciona tipo Maven, en la ventana de dependencias se agregan las dependencias de Lombok, spring web, spring data jpa, mysql Driver para finalizar se crea el proyecto, a continuación se evidencian los anexos de la creación del proyecto



Para poder representar los datos en la base de datos se crea una entidad que proporciona una capa de abstracción entre la aplicación Java y la base de datos. Se crean las clases correspondientes al modelo Entidad-Relación, se agregan atributos a la clase para representar los datos de la entidad, constructores a la clase para crear nuevas instancias de la entidad, se crean las relaciones de las entidades.





se crea el archivo Dockerfile este se utiliza para crear una imagen Docker que contenga la aplicación Java. La imagen Docker se puede utilizar para ejecutar la aplicación Java en cualquier entorno que tenga instalado Docker. Las instrucciones del archivo Dockerfile se pueden dividir en tres categorías:

- Instrucciones de construcción: Estas instrucciones se utilizan para instalar las dependencias y configurar la aplicación Java.
- Instrucciones de compilación: Estas instrucciones se utilizan para compilar la aplicación Java.
- Instrucciones de etiquetado: Estas instrucciones se utilizan para etiquetar la imagen Docker con un nombre y una versión.

Al crear el archivo Dockerfile se crea una imagen Docker que utiliza la imagen base openjdk:17. El archivo Dockerfile también copia el archivo app.jar de la aplicación Java en el directorio /app del contenedor Docker. Finalmente, el archivo Dockerfile establece el punto de entrada del contenedor Docker en el comando java -jar app.jar.

```

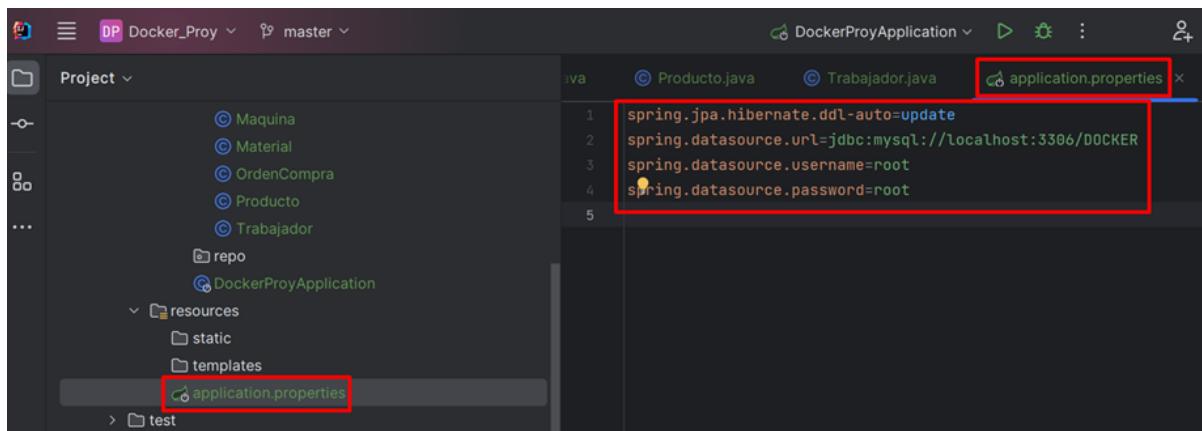
FROM openjdk:17
ADD target/springboot-mysql-docker.jar springboot-mysql-docker.jar
ENTRYPOINT ["java","-jar","/springboot-mysql-docker.jar"]

```

El archivo application.properties es un archivo de configuración utilizado por las aplicaciones Java para almacenar información de configuración. Esta información puede incluir la configuración de la base de datos, la configuración del servidor web y la configuración de la aplicación. Para poder adaptar el archivo application.properties para el caso de estudio, se deben realizar los siguientes cambios:

- Instrucciones de construcción: Estas instrucciones se utilizan para instalar las dependencias y configurar la aplicación Java.
- Instrucciones de compilación: Estas instrucciones se utilizan para compilar la aplicación Java.
- Instrucciones de etiquetado: Estas instrucciones se utilizan para etiquetar la imagen Docker con un nombre y una versión.

Al añadir la configuración del servidor web: Se debe añadir la configuración del servidor web al archivo application.properties. La configuración del servidor web debe incluir el puerto en el que se ejecutará el servidor web. `server.port=8081`



a continuación se verifica la versión Docker que se está utilizando, se verifica las imágenes Docker que se encuentran en el sistema, se descarga la imagen Docker de MySQL 8.0.32 del repositorio Docker Hub para finalizar se vuelve a comprobar la imagen descargada

```
C:\ Símbolo del sistema
Microsoft Windows [Versión 10.0.19045.3448]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\julik>docker --version
Docker version 24.0.6, build ed223bc

C:\Users\julik>docker images
REPOSITORY      TAG          IMAGE ID      CREATED      SIZE
C:\Users\julik>docker pull mysql:8.0.32
8.0.32: Pulling from library/mysql
328ba678bf27: Pull complete
f3f5ff008d73: Pull complete
dd7054d6d0c7: Pull complete
70b5d4e8750e: Pull complete
cdc4a7b43bdd: Pull complete
3e9c0b61a8f3: Pull complete
806a08b6c085: Pull complete
021b2cebd832: Pull complete
ad31ba45b26b: Pull complete
0d4c2bd59d1c: Pull complete
148cef42e3b: Pull complete
Digest: sha256:f496c25da703053a6e0717f1d52092205775304ea57535cc9fcaa6f35867800b
Status: Downloaded newer image for mysql:8.0.32
docker.io/library/mysql:8.0.32

What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview mysql:8.0.32

C:\Users\julik>docker images
REPOSITORY      TAG          IMAGE ID      CREATED      SIZE
mysql           8.0.32       412b8cc72e4a   5 months ago  531MB
```

se ejecuta `docker run -p 3307:3306 --name mysqlDbSpring -e MYSQL_ROOT_PASSWORD=root -e MYSQL_DATABASE=DOCKER mysql:8.0.32` esto se realiza para poder ejecutar un contenedor Docker a partir de la imagen Docker de MySQL 8.0.32.

El contenedor Docker se llama mysqldbsspring y el puerto 3307 del host se publica en el puerto 3306 del contenedor Docker. Las variables de entorno MYSQL_ROOT_PASSWORD y MYSQL_DATABASE se establecen en root y DOCKER, respectivamente.

```
C:\Users\juliok>docker run -p 3307:3306 --name mysqldbsspring -e MYSQL_ROOT_PASSWORD=root -e MYSQL_DATABASE=DOCKER mysql:8.0.32
2023-09-19 04:28:14+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.32-1.el8 started.
2023-09-19 04:28:14+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'.
2023-09-19 04:28:14+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.32-1.el8 started.
2023-09-19 04:28:14+00:00 [Note] [Entrypoint]: Initializing database files
2023-09-19T04:28:14.846207Z 0 [Warning] [MY-011068] [Server] The syntax '--skip-host-cache' is deprecated and will be removed in a future release. Please use SET GLOBAL host_cache_size=0 instead.
2023-09-19T04:28:14.871099Z 0 [System] [MY-013169] [Server] /usr/sbin/mysqld (mysqld 8.0.32) initializing of server in progress as process 81
2023-09-19T04:28:14.871432Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
2023-09-19T04:28:15.717248Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
2023-09-19T04:28:18.199231Z 6 [Warning] [MY-010453] [Server] root@localhost is created with an empty password ! Please consider switching off the --initialize-insecure option.
2023-09-19 04:28:25+00:00 [Note] [Entrypoint]: Database files initialized
2023-09-19 04:28:25+00:00 [Note] [Entrypoint]: Starting temporary server
2023-09-19T04:28:25.555850Z 0 [Warning] [MY-011068] [Server] The syntax '--skip-host-cache' is deprecated and will be removed in a future release. Please use SET GLOBAL host_cache_size=0 instead.
2023-09-19T04:28:25.559907Z 0 [System] [MY-010116] [Server] /usr/sbin/mysqld (mysqld 8.0.32) starting as process 126
2023-09-19T04:28:25.590706Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
2023-09-19T04:28:26.070906Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
2023-09-19T04:28:26.636817Z 0 [Warning] [MY-010968] [Server] CA certificate ca.pem is self signed.
2023-09-19T04:28:26.636902Z 0 [System] [MY-013602] [Server] Channel mysql_main configured to support TLS. Encrypted connections are now supported for this channel.
2023-09-19T04:28:26.643179Z 0 [Warning] [MY-011810] [Server] Insecure configuration for --pid-file: Location '/var/run/mysqld' in the path is accessible to all OS users. Consider choosing a different directory.
2023-09-19T04:28:26.686426Z 0 [System] [MY-011323] [Server] X Plugin ready for connections. Socket: /var/run/mysqld/mysqlx.sock
2023-09-19T04:28:26.686923Z 0 [System] [MY-010931] [Server] /usr/sbin/mysqld: ready for connections. Version: '8.0.32' socket: '/var/run/mysqld/mysqld.sock' port: 0 MySQL Community Server - GPL.
2023-09-19 04:28:26+00:00 [Note] [Entrypoint]: Temporary server started.
'/var/lib/mysql/mysql.sock' -> '/var/run/mysqld/mysqld.sock'
Warning: Unable to load '/usr/share/zoneinfo/iso3166.tab' as time zone. Skipping it.
Warning: Unable to load '/usr/share/zoneinfo/leap-seconds.list' as time zone. Skipping it.
Warning: Unable to load '/usr/share/zoneinfo/leapseconds' as time zone. Skipping it.
Warning: Unable to load '/usr/share/zoneinfo/tzdata.zi' as time zone. Skipping it.
Warning: Unable to load '/usr/share/zoneinfo/tzdata.tab' as time zone. Skipping it.
Warning: Unable to load '/usr/share/zoneinfo/zone1970.tab' as time zone. Skipping it.
2023-09-19 04:28:32+00:00 [Note] [Entrypoint]: Creating database DOCKER
2023-09-19 04:28:32+00:00 [Note] [Entrypoint]: Stopping temporary server
2023-09-19T04:28:32.321458Z 11 [System] [MY-013172] [Server] Received SHUTDOWN from user root. Shutting down mysqld (Version: 8.0.32).
```

se realizará la verificación del contenedor creado en docker

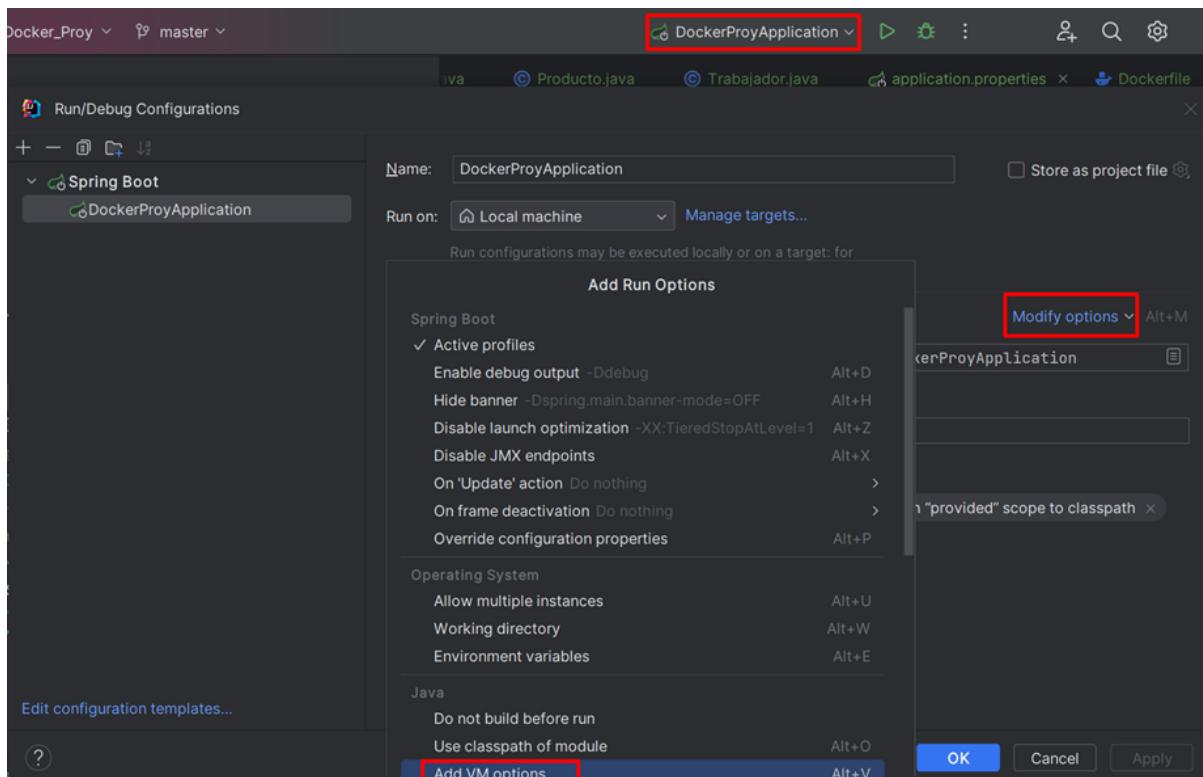
Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
mysqldbsspring	mysql:8.0.32	Running	0.83%	3307:3306	4 minutes ago	●

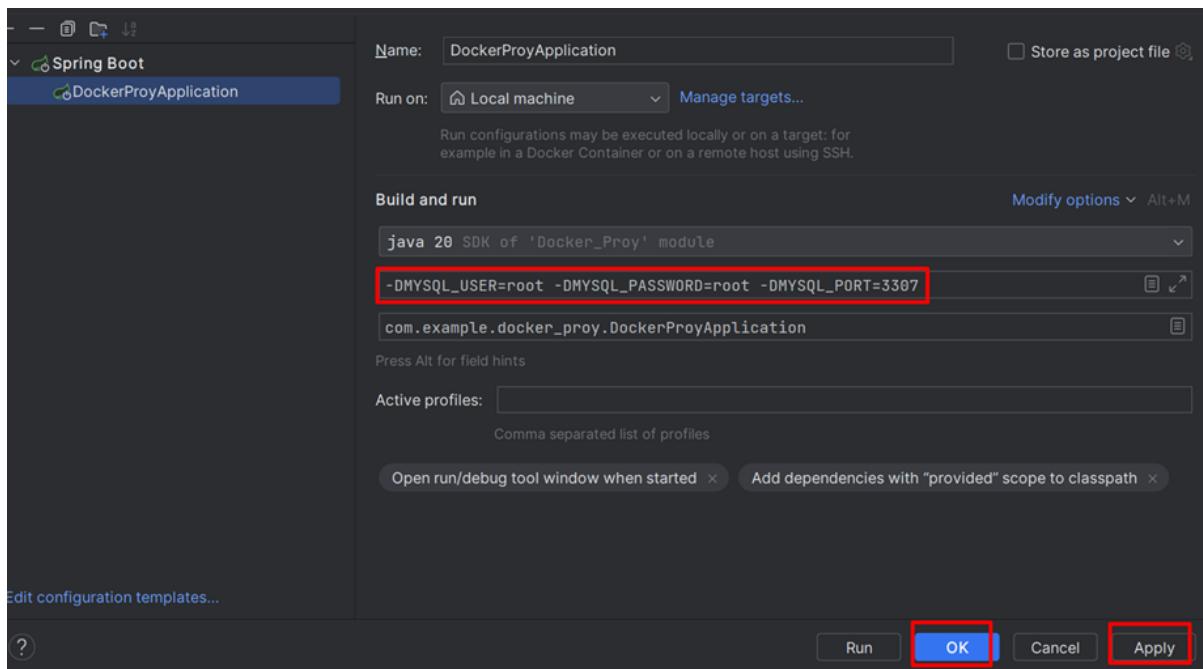
```

1 spring.jpa.hibernate.ddl-auto=update
2 spring.datasource.url=jdbc:mysql://localhost:${MYSQL_PORT:3306}/DOCKER
3 spring.datasource.username=${MYSQL_USER:root}
4 spring.datasource.password=${MYSQL_PASSWORD:root}
5

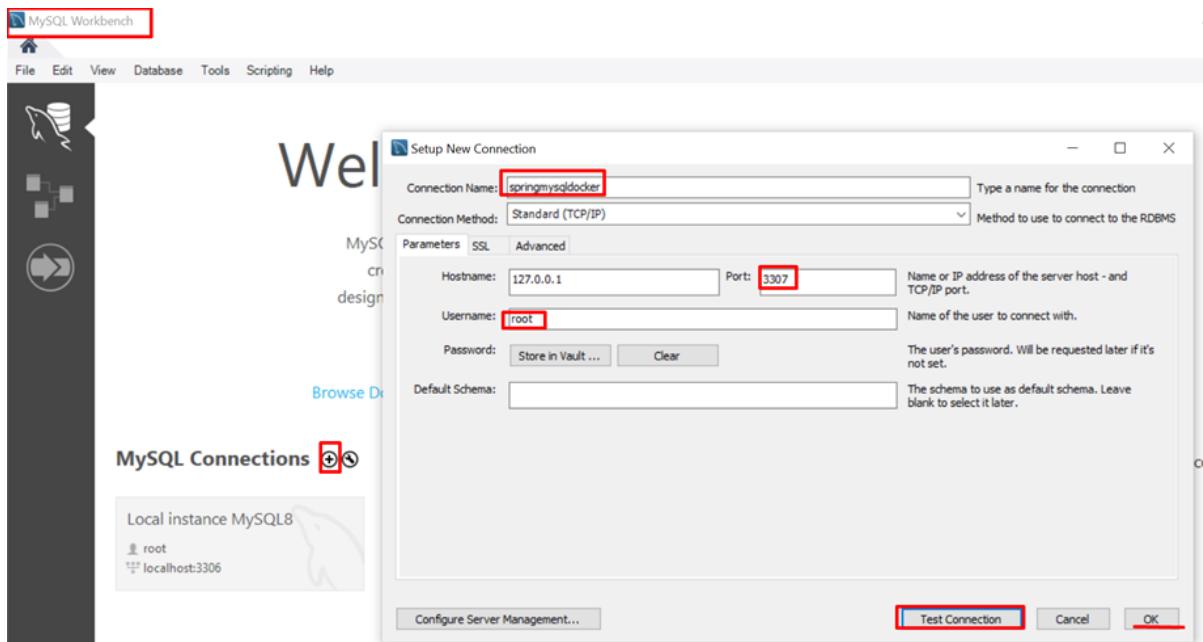
```

realizó una modificación al proyecto agregando build and run es para facilitar la construcción y ejecución de la aplicación. Con esta modificación, se puede construir la imagen Docker





Verificamos la conexión con la base de datos así como seleccionamos los puertos de uso, el usuario y la contraseña agregada anteriormente.



Al crear la imagen de docker podemos confirmar la creación de nuestra base de datos

The screenshot shows the MySQL Workbench interface. In the Navigator pane, under SCHEMAS, a new schema named 'DOCKER' is listed, highlighted with a red box. In the main Query 1 window, there is a single row labeled '1'. Below the Navigator, the tabs 'Administration' and 'Schemas' are visible, with 'Schemas' selected. In the bottom section, titled 'Schema: DOCKER', there is an 'Output' tab. At the bottom, a table lists Docker containers. One container, named 'mysqlDbSpring' with the ID 'e19ed301dd7f', is shown with its status as 'Exited'. A red box highlights the 'Actions' column for this container.

Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
mysqlDbSpring e19ed301dd7f	mysql:8.0.32	Exited	N/A	3307:3306	56 minutes ago	▶ ⋮ ✖

Usamos Docker Network para conectar contenedores Docker entre sí. Esto permite que los contenedores se comuniquen entre sí, incluso si se ejecutan en diferentes máquinas, se crea una red Docker con *docker network create spring-net* la red de puente es llamada *spring-net*.

```
C:\Users\julik>docker network
Usage: docker network COMMAND

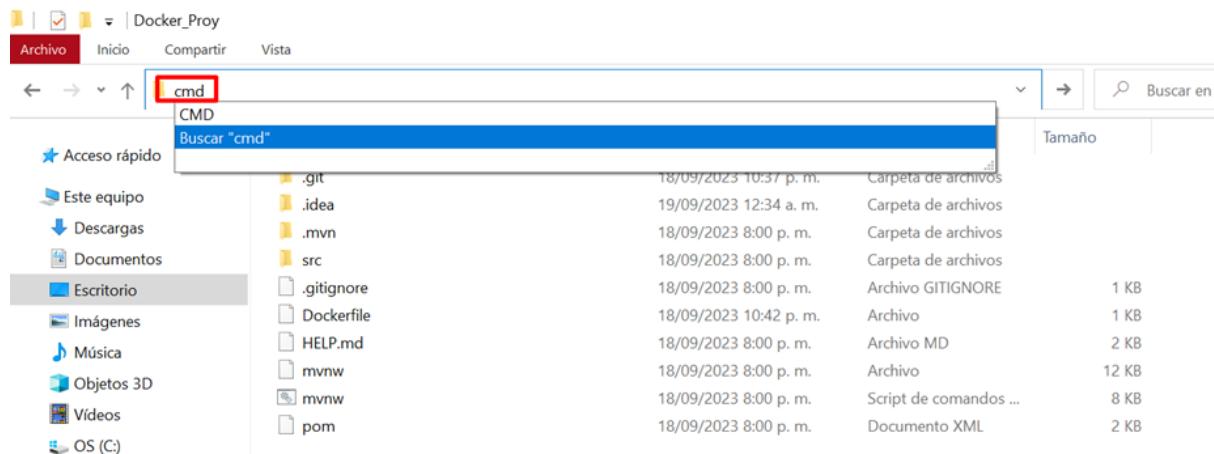
Manage networks

Commands:
connect      Connect a container to a network
create       Create a network
disconnect   Disconnect a container from a network
inspect     Display detailed information on one or more networks
ls          List networks
prune       Remove all unused networks
rm          Remove one or more networks

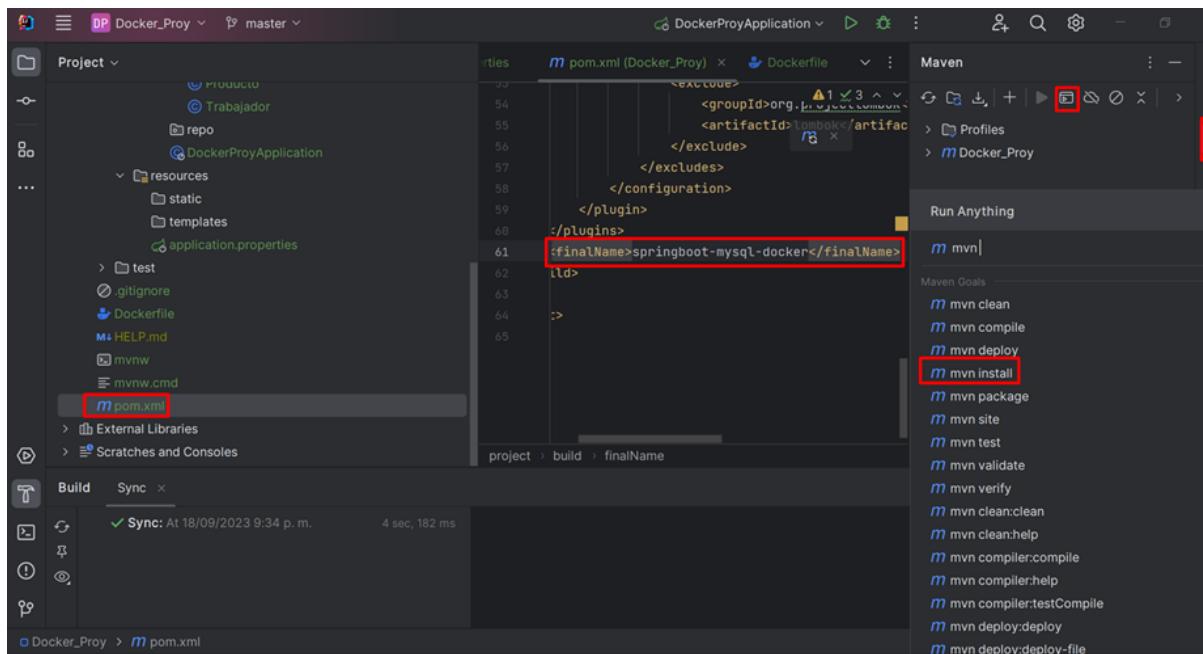
Run 'docker network COMMAND --help' for more information on a command.

C:\Users\julik>docker network create spring-net
51b6bc24e2ef74ebec9a90b654e13956cd67e85d6f48fb82c218c1b2e8d4df46
```

nos ubicamos en la localización del proyecto creado y abrimos una ventana cmd



Se realizan modificaciones en el archivo de pom.xml en el proyecto Java para configurar el proyecto, agregar dependencias y definir el proceso de construcción. El archivo pom.xml es un archivo XML que define el proyecto Java. El archivo pom.xml contiene información sobre el proyecto, como su nombre, versión, dependencias y proceso de construcción.



se realiza la ejecución del comando `docker build -t springboot-mysq=docker`. para poder construir la imagen Docker de la aplicación, se crea una imagen Docker con el nombre `springboot-mysql:docker` a partir del contenido del directorio actual.

```
C:\Users\julik\Desktop\Especializacion 2-2023\frameworks\Docker_Proj>docker build -t springboot-mysql-docker .
[+] Building 4.9s (/)
[+] FINISHED
=> [internal] load build definition from Dockerfile
=> [internal] transfering dockerfile: 181B
=> [internal] load .dockerrcignore
=> [internal] transfering context: 2B
=> [internal] load metadata for docker.io/library/openjdk:17
=> CACHED [1/2] FROM docker.io/library/openjdk:17@sha256:528707081fdb9562eb819128a9f85ae7fe000e2fbaf9f87662e7b3f38cb7d8
=> [internal] load build context
=> [internal] transfering context: 44.91MB
=> [2/2] ADD target/springboot-mysql-docker.jar springboot-mysql-docker.jar
=> exporting to image
=> => exporting layers
=> => writing image sha256:8f5f445d2300f215699cc924d2b0a10fe7cf63f53126c7937096727dc20527
=> => naming to docker.io/library/springboot-mysql-docker
WARNING: buildx: git was not found in the system. Current commit information was not captured by the build

What's Next?
 1. Sign in to your Docker account → docker login
 2. View a summary of image vulnerabilities and recommendations → docker scout quickview
```

verificamos la creación de la imagen `springboot-mysql-docker`

```
C:\Users\julik\Desktop\Especializacion 2-2023\frameworks\Docker_Proj>docker images
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
springboot-mysql-docker   latest   8f5f445d2300   3 minutes ago  516MB
mysql                8.0.32  412b8cc72e4a   5 months ago  531MB
```

listamos todas las redes Docker que se encuentran en el sistema y verificamos nuestra red creada, conectamos el contenedor Docker `spring-mysqldb` a la red Docker `spring.net`.

```
C:\Users\julik\Desktop\Especializacion 2-2023\frameworks\Docker_Proj>docker ps
CONTAINER ID   IMAGE      COMMAND       CREATED          STATUS          PORTS
e19ed301dd7f   mysql:8.0.32 "docker-entrypoint.s..." 2 hours ago   Up 56 minutes   33060/tcp, 0.0.0.0:3307->3306/tcp   NAMES
                                                               mysql dbspring

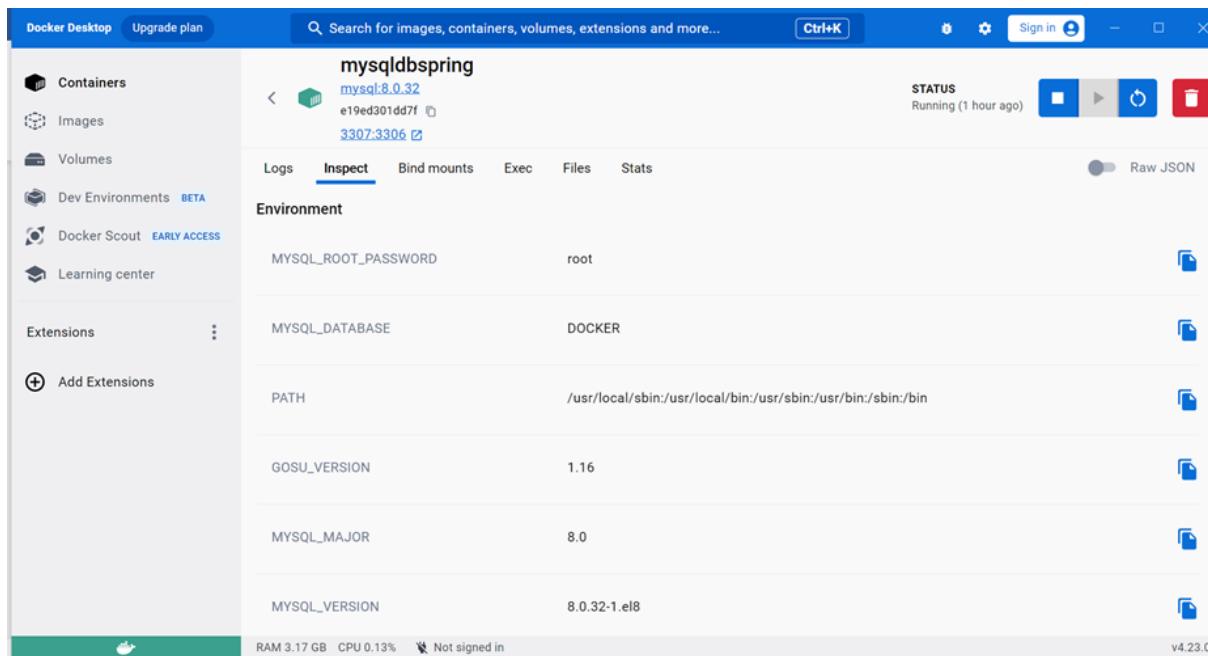
C:\Users\julik\Desktop\Especializacion 2-2023\frameworks\Docker_Proj>docker network ls
NETWORK ID     NAME      DRIVER      SCOPE
eda9b6ee456d   bridge    bridge      local
4b4de0459c7b   host      host       local
9e7193a40bbf   none     null       local
51b6bc24e2ef   spring-net bridge      local

C:\Users\julik\Desktop\Especializacion 2-2023\frameworks\Docker_Proj>docker network connect spring-net mysql dbspring
C:\Users\julik\Desktop\Especializacion 2-2023\frameworks\Docker_Proj>
```

verificamos la información sobre la red docker.

```
c:\Users\julik\Desktop\Especializacion 2-2023\frameworks\Docker_Proj>docker network inspect spring-net
[
  {
    "Name": "spring-net",
    "Id": "51b6bc24e2ef74ebec9a90b654e13956cd67e85d6f48fb82c218c1b2e8d4df46",
    "Created": "2023-09-19T05:28:22.150005773Z",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": {},
      "Config": [
        {
          "Subnet": "172.18.0.0/16",
          "Gateway": "172.18.0.1"
        }
      ]
    },
    "Internal": false,
    "Attachable": false,
    "Ingress": false,
    "ConfigFrom": {
      "Network": ""
    },
    "ConfigOnly": false,
    "Containers": {
      "e19ed301dd7f6a0e076d9d8cd88fed514b9c617745ad56bad43c1a07c2c0904e": {
        "Name": "mysql dbspring",
        "EndpointID": "cb5d3d0130fadcc611f00f4d0a852b2038a125e384bfe5276005793476f2dbfef",
        "MacAddress": "02:42:ac:12:00:02",
        "IPv4Address": "172.18.0.2/16",
        "IPv6Address": ""
      }
    },
    "Options": {},
    "Labels": {}
  }
]
```

Verificamos la red creada anteriormente en docker.



El comando `docker run -p 9090:8080 --name springboot-mysql-docker --net spring-net -e MYSQL_HOST=mysql8spring -e MYSQL_USER=root -e MYSQL_PASSWORD=root -e MYSQL_PORT=3306` ejecuta un contenedor Docker a partir de la imagen Docker `springboot-mysql-docker`.

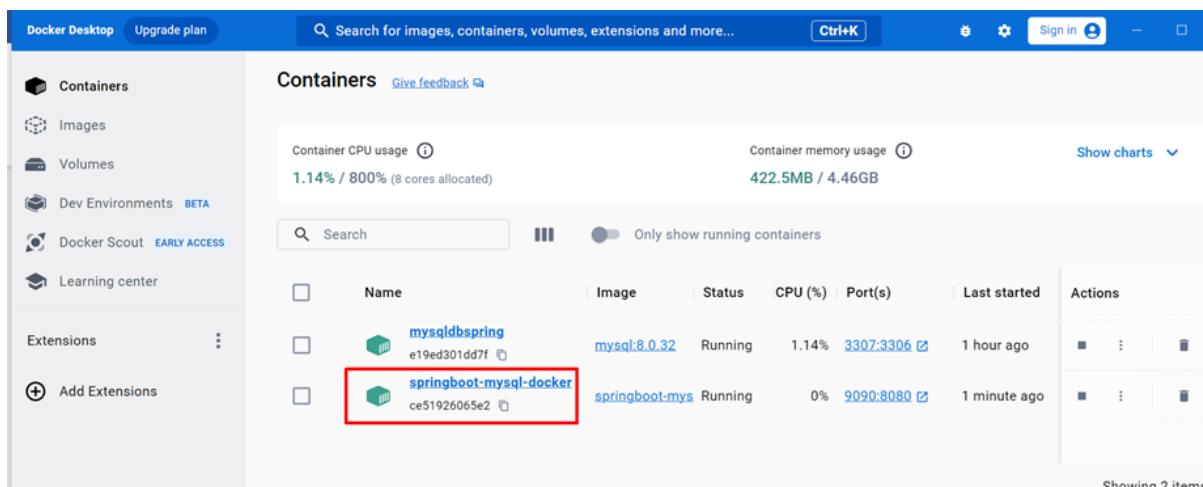
```
C:\Windows\System32\cmd.exe - docker run -p 9090:8080 --name springboot-mysql-docker --net spring-net -e MYSQL_HOST=mysql8spring -e MYSQL_USER=root -e MYSQL_PASSWORD=root -e MYSQL_PORT=3306 springboot-mysql-docker

[...]
:: Spring Boot :: (v3.1.3)

2023-09-19T07:21:17.088Z INFO 1 --- [           main] c.e.docker_proy.DockerProyApplication : Starting DockerProyApplication v0.0.1-SNAPSHOT using Java 17.0.2 with PID 1 (/springboot-mysql-docker.jar started by root in /)
2023-09-19T07:21:17.095Z INFO 1 --- [           main] c.e.docker_proy.DockerProyApplication : No active profile set, falling back to 1 default profile: "default"
2023-09-19T07:21:18.634Z INFO 1 --- [           main] s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA Repositories in DEFAULT mode.
2023-09-19T07:21:18.683Z INFO 1 --- [           main] s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 24 ms. Found 0 JPA repository interfaces.
2023-09-19T07:21:19.900Z INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2023-09-19T07:21:19.927Z INFO 1 --- [           main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2023-09-19T07:21:19.928Z INFO 1 --- [           main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.12]
2023-09-19T07:21:20.119Z INFO 1 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2023-09-19T07:21:20.123Z INFO 1 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 2888 ms
2023-09-19T07:21:20.467Z INFO 1 --- [           main] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: default]
2023-09-19T07:21:20.623Z INFO 1 --- [           main] org.hibernate.Version : HHH000412: Hibernate ORM core version 6.2.7.Final
2023-09-19T07:21:20.629Z INFO 1 --- [           main] org.hibernate.cfg.Environment : HHH000406: Using bytecode reflection optimizer
2023-09-19T07:21:20.908Z INFO 1 --- [           main] o.h.b.i.BytecodeProviderInitiator : HHH000021: Bytecode provider name : bytebuddy
2023-09-19T07:21:20.207Z INFO 1 --- [           main] o.s.o.j.p.SpringPersistenceUnitInfo : No LoadTimeWeaver setup: ignoring JPA class transformer
2023-09-19T07:21:21.234Z INFO 1 --- [           main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2023-09-19T07:21:22.181Z INFO 1 --- [           main] com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Added connection com.mysql.cj.jdbc.ConnectionImpl@41d7b27f
2023-09-19T07:21:22.189Z INFO 1 --- [           main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2023-09-19T07:21:22.962Z INFO 1 --- [           main] o.h.b.i.BytecodeProviderInitiator : HHH000021: Bytecode provider name : bytebuddy
2023-09-19T07:21:24.083Z INFO 1 --- [           main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000496: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
2023-09-19T07:21:24.344Z WARN 1 --- [           main] o.h.t.s.i.ExceptionHandlerLoggedImpl : GenerationTarget encountered exception accepting command : Error executing DDL "create table maquina (num_serie_maq varchar(255) not null auto_increment, fecha_compra datetime(6), marca varchar(255), modelo varchar(255), primary key (num_serie_maq)) engine=InnoDB" via JDBC [Incorrect column specifier for column 'num_serie_maq']
org.hibernate.tool.schema.spi.CommandAcceptanceException: Error executing DDL "create table maquina (num_serie_maq varchar(255) not null auto_increment, fecha_compra datetime(6), marca varchar(255), modelo varchar(255), primary key (num_serie_maq)) engine=InnoDB" via JDBC [Incorrect column specifier for column 'num_serie_maq']

at org.hibernate.tool.schema.internal.exec.GenerationTargetToDatabase.accept(GenerationTargetToDatabase.java:92) ~[hibernate-core-6.2.7.Final.jar!/:6.2.7.Final]
```

Verificamos la construcción del contenedor creado,



Se realiza la creación de entidades de repository para poder proporcionar una interfaz para interactuar con la base de datos. Estas entidades implementan los métodos necesarios para realizar operaciones CRUD (create, read, update, delete)

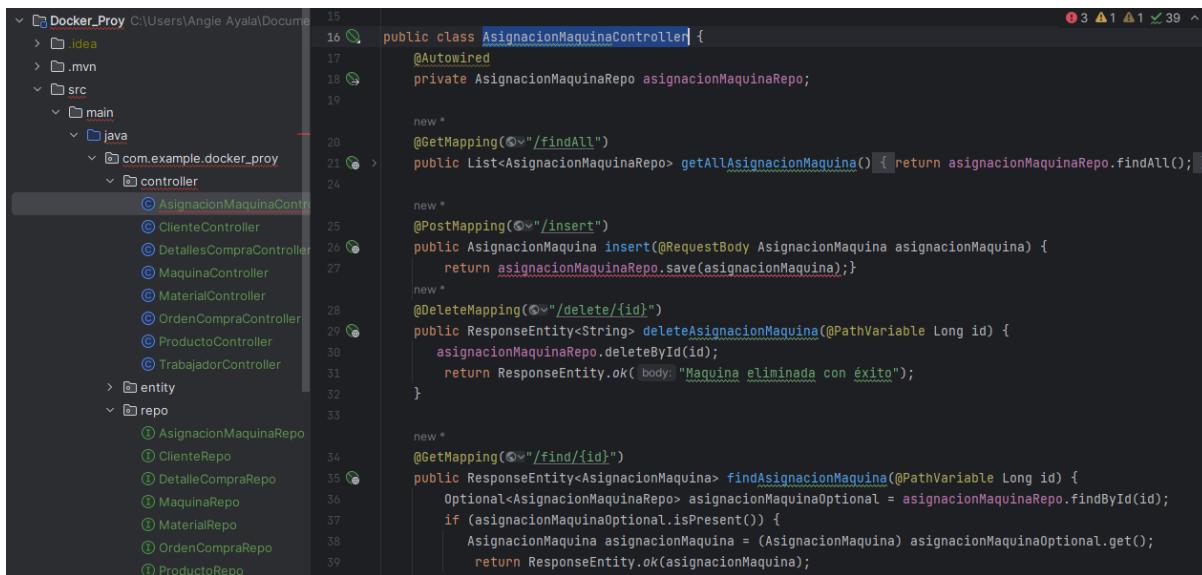
```

1 package com.example.docker_proy.repo;
2
3
4 import org.springframework.data.jpa.repository.JpaRepository;
5
6 usages new *
7 public interface AsignacionMaquinaRepo extends JpaRepository<AsignacionMaquinaRepo , Long >{
}

```

The code snippet shows a Java interface named `AsignacionMaquinaRepo` extending `JpaRepository`. The interface has a single method declaration with a return type of `Long`.

Se realiza la creación de las entidades controller que se relacionan entre sí con repo a través de la inyección de dependencias. La entidad de controller inyecta la entidad de repository para poder realizar operaciones en la base de datos. Por ejemplo, la entidad de controller para la creación de `AsignacionMaquina` podría inyectar la entidad de repository de `asignacionMaquina`



```

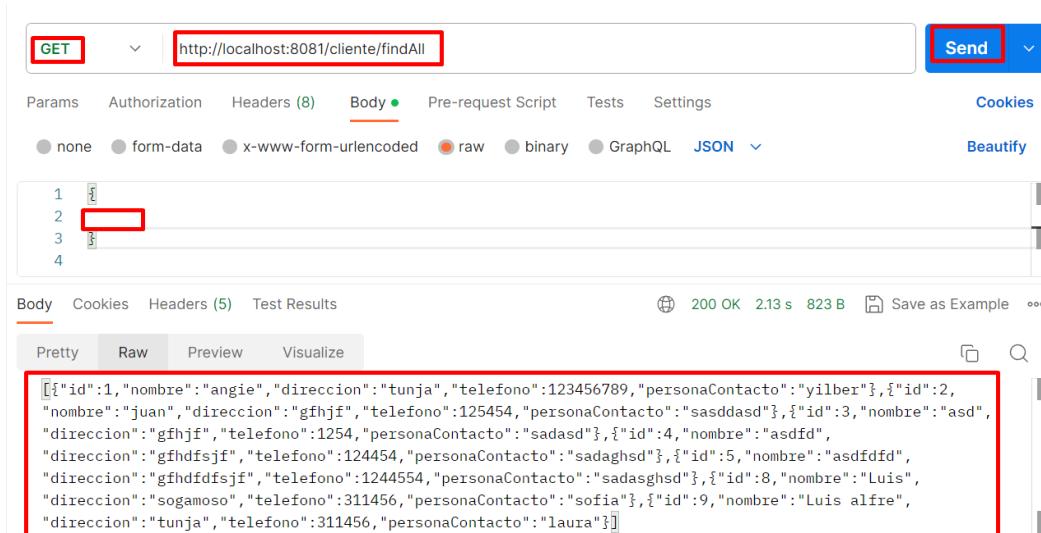
15  public class AsignacionMaquinaController {
16      @Autowired
17      private AsignacionMaquinaRepo asignacionMaquinaRepo;
18
19      new *
20      @GetMapping("findAll")
21      public List<AsignacionMaquinaRepo> getAllAsignacionMaquina() { return asignacionMaquinaRepo.findAll(); }
22
23      new *
24      @PostMapping("insert")
25      public AsignacionMaquina insert(@RequestBody AsignacionMaquina asignacionMaquina) {
26          return asignacionMaquinaRepo.save(asignacionMaquina);
27      }
28
29      new *
30      @DeleteMapping("delete/{id}")
31      public ResponseEntity<String> deleteAsignacionMaquina(@PathVariable Long id) {
32          asignacionMaquinaRepo.deleteById(id);
33          return ResponseEntity.ok( body: "Maquina eliminada con éxito");
34      }
35
36      new *
37      @GetMapping("find/{id}")
38      public ResponseEntity<AsignacionMaquina> findAsignacionMaquina(@PathVariable Long id) {
39          Optional<AsignacionMaquinaRepo> asignacionMaquinaOptional = asignacionMaquinaRepo.findById(id);
40          if (asignacionMaquinaOptional.isPresent()) {
41              AsignacionMaquina asignacionMaquina = (AsignacionMaquina) asignacionMaquinaOptional.get();
42              return ResponseEntity.ok(asignacionMaquina);
43          }
44      }

```

En la anterior imagen La anotación `@Autowired` inyecta una instancia de la interfaz `Asignación Máquina Repo` en el campo `asignación Maquina Repo`. se realiza la creación de la CRUD con:

- el método `getAll Asignacion Máquina()` devuelve una lista de todas las asignaciones de máquinas.
- El método `insert()` guarda una nueva asignación de máquina en la base de datos.
- El método `deleteAsignacionMaquina()` elimina una asignación de máquina de la base de datos por su ID.
- El método `findAsignacionMaquina()` devuelve una asignación de máquina por su ID.
- El método `updateAsignacionMaquina()` actualiza una asignación de máquina en la base de datos

acción findAll



GET <http://localhost:8081/cliente/findAll> Send

Params Authorization Headers (8) Body **JSON** Tests Settings Cookies Beautify

```

1 [
2   {
3     "id": 1,
4     "nombre": "angie",
5     "direccion": "tunja",
6     "telefono": 123456789,
7     "personaContacto": "yilber"
8   },
9   {
10     "id": 2,
11     "nombre": "juan",
12     "direccion": "gfhjf",
13     "telefono": 125454,
14     "personaContacto": "sasddasd"
15   },
16   {
17     "id": 3,
18     "nombre": "asd",
19     "direccion": "gfhjf",
20     "telefono": 1254,
21     "personaContacto": "sadasd"
22   },
23   {
24     "id": 4,
25     "nombre": "asdfd",
26     "direccion": "gfhjf",
27     "telefono": 124454,
28     "personaContacto": "sadaghsd"
29   },
30   {
31     "id": 5,
32     "nombre": "asfdfdf",
33     "direccion": "gfhdfdsjf",
34     "telefono": 1244554,
35     "personaContacto": "sadasghsd"
36   },
37   {
38     "id": 8,
39     "nombre": "Luis",
40     "direccion": "sogamoso",
41     "telefono": 311456,
42     "personaContacto": "sofia"
43   },
44   {
45     "id": 9,
46     "nombre": "Luis alfre",
47     "direccion": "tunja",
48     "telefono": 311456,
49     "personaContacto": "laura"
50   }
51 ]

```

Body Cookies Headers (5) Test Results 200 OK 2.13 s 823 B Save as Example

Pretty Raw Preview Visualize

observamos que esta acción `get` nos devuelve la lista de los clientes presentes en nuestra tabla `accion find/id`

GET http://localhost:8081/cliente/find/1 Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Body (Raw JSON)

```
{
  "id": 1,
  "nombre": "angie",
  "direccion": "tunja",
  "telefono": 123456789,
  "personaContacto": "yilber"
}
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize

200 OK 165 ms 257 B Save as Example

esta accion get nos trae específicamente los datos pertenecientes al cliente identificado con id que le ingresemos en este caso 1

para estas dos acciones las podemos probar desde un buscador cualquier con la misma dirección

localhost:8081/cliente/findAll

localhost:8081/cliente/findAll

```
[
  {"id": 1, "nombre": "angie", "direccion": "tunja", "telefono": 123456789, "personaContacto": "yilber"},
  {"id": 2, "nombre": "juan", "direccion": "gfhjf", "telefono": 125454, "personaContacto": "sasddasd"},
  {"id": 3, "nombre": "asd", "direccion": "gfhjf", "telefono": 1254, "personaContacto": "sadasd"},
  {"id": 4, "nombre": "asfdfd", "direccion": "gfhdfsfjf", "telefono": 124454, "personaContacto": "sadaghsd"},
  {"id": 5, "nombre": "asfdfd", "direccion": "gfhdfsfjf", "telefono": 1244554, "personaContacto": "sadasghsd"},
  {"id": 8, "nombre": "Luis", "direccion": "sogamoso", "telefono": 311456, "personaContacto": "sofia"},
  {"id": 9, "nombre": "Luis alfre", "direccion": "tunja", "telefono": 311456, "personaContacto": "laura"}
]
```

localhost:8081/cliente/find/2

localhost:8081/cliente/find/2

```
{"id": 2, "nombre": "juan", "direccion": "gfhjf", "telefono": 125454, "personaContacto": "sasddasd"}
```

esto solo lo podemos hacer con las acciones get ya que estos buscadores solo envian acciones de este tipo si es otra genera error por eso se hizo el uso de postman

accion insert

The screenshot shows a Postman interface with the following details:

- Method:** POST (highlighted with a red box)
- URL:** http://localhost:8081/cliente/insert (highlighted with a red box)
- Body:** JSON (highlighted with a red box)
- Request Body Content:**

```

1 {
2   "nombre": "jose",
3   "direccion": "tibana",
4   "telefono": 1456789,
5   "personaContacto": "pedro"
6 }
7
    
```
- Response Status:** 200 OK (highlighted with a red box)
- Response Time:** 685 ms
- Response Size:** 255 B
- ResponseBody:** {"id":10,"nombre":"jose","direccion":"tibana","telefono":1456789,"personaContacto":"pedro"} (highlighted with a red box)

vemos que esta acción inserta un nuevo registro en la tabla

acción delete

Result Grid					
	id_cliente	direccion	nombre	persona_contacto	telefono
3	gfhnjf	asd	sadasd	1254	
4	gfndfsjf	asdfd	sadaghsd	124454	
5	gfhdffsfjf	asfdfdf	sadasghsd	1244554	
8	sogamoso	Luis	sofia	311456	
9	tunja	Luis alfre	laura	311456	
10	tibana	jose	pedro	1456789	

se observa que en la base tenemos estos datos vamos a borrar el de id 9

The screenshot shows a Postman interface with the following details:

- Method:** DELETE (highlighted with a red box)
- URL:** http://localhost:8081/cliente/delete/9 (highlighted with a red box)
- Body:** JSON (highlighted with a red box)
- Request Body Content:**

```

1 {
2   "id": 9
3 }
4
    
```
- Response Status:** 200 OK (highlighted with a red box)
- Response Time:** 737 ms
- Response Size:** 192 B
- ResponseBody:** Cliente eliminado con éxito (highlighted with a red box)

nos dice que ha sido eliminado con éxito, si revisamos efectivamente fue eliminado

	id_cliente	direccion	nombre	persona_contacto	telefono
	3	gfhjf	asd	sadasd	1254
	4	gfhdhsjf	asdfd	sadaghsd	124454
	5	gfhdhsjf	asdfdfd	sadasghsd	1244554
	8	soqamoso	Luis	sofia	311456
	10	tibana	jose	pedro	1456789
	HULL	HULL	HULL	HULL	HULL
(2)					

acción update

PUT <http://localhost:8081/cliente/update/3>

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL **JSON**

```

1  {
2    "nombre": "julian",
3    "direccion": "nuevocolon",
4    "telefono": 31748086,
5    "personaContacto": "caro"
6  }
7

```

Body Cookies Headers (5) Test Results 200 OK 4.49 s 260 B

Pretty Raw Preview Visualize

```
{"id":3,"nombre":"julian","direccion":"nuevocolon","telefono":31748086,"personaContacto":"caro"}
```

se observa que el dato 3 ha sido actualizado y verificamos en la base de datos

The screenshot shows a MySQL Workbench Result Grid displaying client information. The columns are labeled: id_cliente, direccion, nombre, persona_contacto, and telefono. The data rows are:

	id_cliente	direccion	nombre	persona_contacto	telefono
▶	1	tunja	angie	yilber	123456789
	2	qfhjf	juan	sasddasd	125454
	3	nuevocolon	julian	caro	31748086
	4	gfhdfsjf	asdfd	sadaghsd	124454
	5	gfhdfdfsfj	asdfsdf	sadasghsd	1244554
	8	sogamoso	gfhdfdfsfj	sofia	311456
	10	tierra	isso	nodeo	1455700