Creación de Proyecto REST con Spring Boot y MySQL - Docker.

1. Objetivo.

El documento tiene como principal objetivo redactar paso a paso el desarrollo de un proyecto con arquitectura *REST* y describir algunos conceptos que permitan dar comprensión de los elementos que son necesarios para construir de forma exitosa dicho proyecto. El proyecto fue creado usando *MySQL* como gestor de base de datos ejecutada dentro de un contenedor *Docker* para la capa de persistencia; para la capa *REST backend* se usó *Spring Boot Java* para crear la conexión, recepción y manejo de peticiones entre la capa de datos y el Cliente que es creado usando *JavaScript, CSS* y *HTML,* sin embargo, por el tipo de arquitectura, permite que los *Endpoints* sean consumidos por cualquier otro tipo de *Cliente web*.

2. Instalación Docker y Configuración MySQL.

Para poder utilizar *Docker* fue necesario poder instalarlo y configurar un contenedor siguiendo los siguientes pasos:

1. Se descargo *Docker Desktop* desde https://www.docker.com/products/docker-desktop;
2. Se ejecutó el archivo *.exe* para realizar la instalación;
3. Se puede verificar la instalación de *Docker* ejecutando el comando *"docker --version"* en terminal o *CMD;*
4. Después de la instalación se creo un contenedor donde se simulo el entorno necesario para instalar *MySQL;*
5. Se hizo una prueba de conexión para ejecutar *MySQL* dentro del contenedor;
6. Se accedió a la base de datos para crear la tabla *"Clientes"* para usarla como entidad de prueba dentro del aplicativo.

3. Proyecto Spring Boot REST.

Para comenzar un Proyecto *Spring Boot* es necesario crear un proyecto dentro del sitio *Spring Initializr (una opción entre muchas otras para crear y configurar un proyecto de desarrollo Spring Boot):*

1. Se ingresó al sitio web propio de *Spring Boot (Initializr)* para crear un Proyecto: https://start.spring.io/, dentro de las características que importan al crear un proyecto de desarrollo con este sitio son la inserción de un servidor enbebido *Tomcat,* administración de dependencias y plugins a través de *Maven (u otro administrador de dependencias como Grandle)* en el archivo *pom.xml,* facilidad de conexión a bases de datos a través del archivo *aplication.properties (dependencia requerida)* y testeo inmediato;
2. Dentro de la creación del proyecto se seleccionó el administrador de dependencias *Maven* y se añadió la dependencia *MySQL* para poder generar la conexión a la base de datos y JPA para el rastreo de entidades dentro de la base de datos *MySQL*;
3. Terminada la selección de dependencias se creó y descargo el proyecto para después comenzar el desarrollo dentro de la carpeta *main/java;*
4. Una vez dentro del proyecto *Spring Boot,* se configuró el archivo *aplication.properties* con las credenciales, dirección IP y puerto de la base de datos *MySQL* y se realizó una prueba de conexión para establecer la comunicación entre ambas capas;
5. La estructura de desarrollo de acuerdo al *Framework Spring* facilita una arquitectura *REST* donde es necesario dividir los archivos por *repositories, services, models* y *controllers,* los cuales tendrán diferentes características*:*
   - ***Models:*** Contendrá las entidades encargadas de mapear tablas dentro de la base de datos, tanto en estructura (campos) como comportamientos (tipo de relación con otras tablas);
   - ***Repositories:*** Contedrá la lógica de negocio que establece los tipos de consultas que se harán a la base de datos y a través de qué *model*;
   - ***Services:*** Se encargará de realizar la conexión directa con la base de datos recibiendo por inyección la estructura de comportamientos dictados en el *Repository* y recibiendo el o los *models* que mapean las tablas a las que apuntará la consulta;
   - ***Controllers:*** Se encargará de recibir las peticiones a través de la capa *Cliente,* contendrán tanto los *endpoints* donde el cliente podrá establecer comunicación con el *Backend* así como las primeras validaciones sobre la petición y los datos recibidos. Si es necesario también se encargará de devolver la respuesta obtenida de la base de datos, por lo tanto también recibirá por inyección de dependencia a los *services*;
6. La arquitectura del proyecto de desarrollo, integrando los elementos anteriores contendrá por lo general y en el caso del ejemplo realizado, puede resumirse y comprenderse a través de los verbos *HTTP Request: CREATE, READ, UPDATE* y *DELETE,* que indican las interacciones que el *cliente* podrá tener con la capa de persistencia a través del consumo de estas interacciones diseñadas dentro del proyecto;
7. El proyecto se diseño para crear comunicación entre una sola tabla llamada *Clientes* que permitió el almacenamiento, lectura, actualización y borrado de registro de clientes;

4. Creación de Capa Cliente con HTML, CSS y JavaScript.

1. Se creo una interfaz gráfica a través de *HTML* y *CSS* para poder representar un sitio web y simular interacciones que permitieran almacenar nuevos clientes a través de un formulario y usando *JavaScript* para enviar dichos datos en formato *JSON*;
2. Para poder consumir los servicios creados dentro del proyecto *Spring Boot* primero en necesario desplegar ambas capas, tanto la base de datos *MySQL* dentro del contenedor *Docker,* como el servidor embebido del proyecto *Spring Boot*.