Desarrollo de una Aplicación Web con Spring Boot, MySQL y Docker.

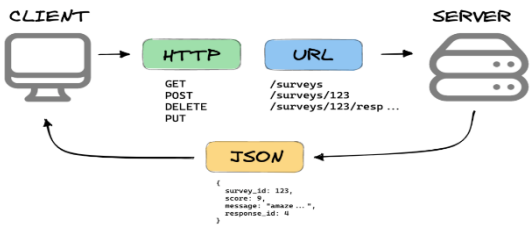
Introducción

Desarrollo de una aplicación web basada en arquitectura cliente-servidor utilizando Spring Boot para el backend, MySQL como base de datos relacional y Docker para contenerización.

Tecnologías

- **Frontend:** HTML, CSS, JavaScript
- **Backend:** Spring Boot (Java)
- **Base de Datos:** MySQL
- **Contenerización:** Docker
- **Otros:** Git, Maven

Arquitectura del Sistema



Desarrollo del Proyecto

Primero se creó una base de datos en MySQL, y dentro de ella una tabla donde se insertaron algunos datos de prueba. Esta base de datos se montó dentro de un contenedor usando Docker, lo que facilitó tenerla lista sin necesidad de instalar MySQL directamente en el sistema operativo. Esto también ayudó a mantener el proyecto más organizado y portable.

Luego comenzamos a construir la API REST usando Spring Boot. El proyecto se creó desde **Spring Initializr**, seleccionando las dependencias necesarias como Spring Web y Spring Data JPA. Dentro de la estructura del proyecto, se usaron varias capas que ayudan a mantener el código más limpio y modular:

- **Entity:** Representa la tabla de la base de datos como una clase Java. Aquí definimos los atributos que corresponden a las columnas de la tabla, y también usamos anotaciones como `@Entity` y `@Id` para indicarle a JPA cómo mapear los datos.
- **Repository:** Es una interfaz que extiende de `JpaRepository`. Esta capa se encarga de manejar todas las operaciones básicas con la base de datos (como guardar, buscar, eliminar, etc.) sin necesidad de escribir código SQL.
- **Service:** Esta capa se encarga de la lógica del negocio. Aquí es donde se realizan las operaciones que combinan varias tareas del repositorio o que contienen reglas específicas del sistema. Además, ayuda a separar la lógica del controlador.
- **Controller:** Es la parte que se comunica con el frontend o con cualquier cliente externo. Se encarga de recibir las solicitudes HTTP (GET, POST, PUT, DELETE) y llamar a los métodos del servicio para responder con los datos o realizar acciones.

Esta estructura en capas hace que la aplicación sea más fácil de entender, mantener y escalar con el tiempo.

Finalmente, trabajamos el frontend utilizando HTML, CSS y JavaScript. Desde ahí, se hizo el consumo de los datos de la API a través de peticiones `fetch` para mostrar la información en pantalla. Se logró una conexión exitosa entre el cliente, el backend y la base de datos.

Pruebas y Solución de Problemas

Al finalizar el desarrollo, se realizaron pruebas para asegurarse de que todo funcionara correctamente. Se revisó la conexión entre el frontend, la API y la base de datos. Durante estas pruebas se detectaron algunos errores menores que fueron solucionados, logrando que la aplicación quedara funcionando de manera estable.