Informe Final: Integración de Bases de Datos Relacionales y NoSQL

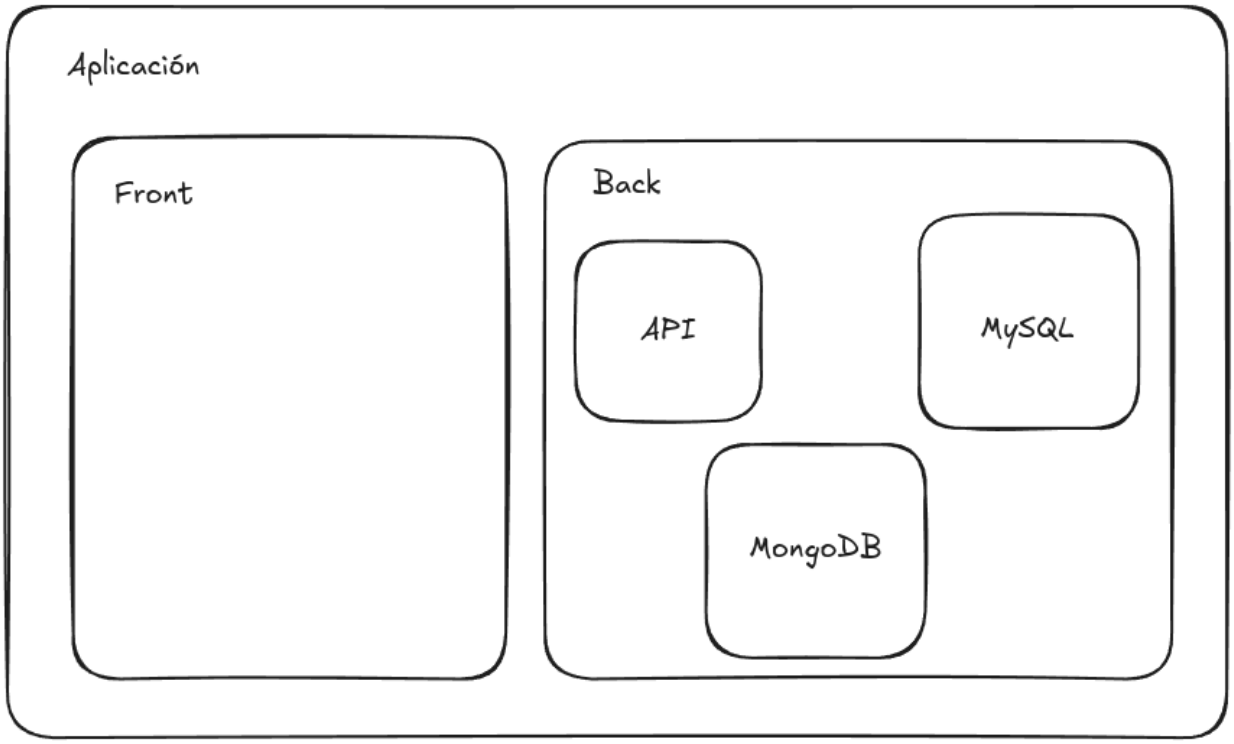
1. Elección de las Tecnologías

Para el desarrollo de este trabajo práctico, se eligieron las siguientes tecnologías:

- **Frontend**: **React** fue elegido como framework principal para el desarrollo de la interfaz de usuario debido a su facilidad de integración con APIs y su capacidad para manejar estados de manera eficiente con hooks. Además, React es altamente popular en el desarrollo web, lo que facilita la reutilización de componentes y la escalabilidad de la aplicación.
- **Backend**: La aplicación utiliza **Spring Boot** en Java, que ofrece una integración rápida y eficiente con bases de datos tanto relacionales como NoSQL. Se eligió Spring Boot por su robustez, seguridad, y facilidad para manejar microservicios y REST APIs, lo que facilita la comunicación entre el frontend y las bases de datos.
- **Bases de Datos**:
  - **Relacional**: Se utilizó **MySQL** como base de datos relacional. MySQL es ampliamente utilizado y adecuado para gestionar datos estructurados y normalizados, lo que facilita la manipulación de grandes volúmenes de datos de manera eficiente.
  - **NoSQL**: Para la base de datos NoSQL, se eligió **MongoDB**, que es flexible y escalable, ideal para almacenar datos no estructurados o semi-estructurados. MongoDB es particularmente útil cuando los datos no tienen una estructura fija y pueden variar entre registros.

2. Diagrama de Arquitectura

El diagrama de arquitectura de la aplicación muestra la interacción entre los siguientes componentes:



1. El **Frontend** (React) se comunica con el **Backend** (Spring Boot) a través de una API REST.
2. El **Backend** gestiona las peticiones y conecta con las bases de datos **MySQL** (relacional) y **MongoDB** (NoSQL) para realizar las operaciones CRUD.
3. Ambas bases de datos son utilizadas para almacenar datos, pero con estructuras diferentes: **MySQL** maneja datos estructurados, mientras que **MongoDB** permite un almacenamiento más flexible y dinámico.

3. Descripción de Funcionalidades CRUD

Las funcionalidades CRUD se implementaron en el backend y frontend para interactuar con ambas bases de datos.

- **Crear**:
  - El frontend tiene formularios para ingresar datos nuevos. Cuando se envía un formulario, los datos son enviados al backend, que a su vez los agrega a la base de datos correspondiente (MySQL o MongoDB).
  - En el frontend, se utiliza el hook `useState` para gestionar el estado de los formularios de entrada. En el backend, se emplean métodos `POST` en las

rutas correspondientes para agregar los nuevos registros.

- **Leer**:
  - La funcionalidad de lectura se realiza mediante las funciones `getPeopleSQL` y `getPeopleNoSQL`, que realizan solicitudes **GET** al backend. Estas funciones permiten obtener los datos almacenados en ambas bases de datos y presentarlos en tablas en el frontend, utilizando **React Table** para mostrar la información.
- **Actualizar**:
  - Los registros de ambas bases de datos pueden ser editados seleccionando un registro específico en la tabla. El frontend envía los datos actualizados al backend, que se encarga de realizar la operación de actualización en la base de datos correspondiente utilizando métodos `PUT` en las rutas definidas.
- **Eliminar**:
  - Se implementó la funcionalidad de eliminación de registros mediante el envío de una solicitud **DELETE** desde el frontend. Al seleccionar un registro en la tabla y presionar el botón de eliminación, el backend elimina el registro de la base de datos correspondiente.

4. Comparación entre Bases de Datos Relacionales y NoSQL

Ventajas de la Base de Datos Relacional (MySQL):

- **Integridad y normalización**: Ideal para datos estructurados con relaciones claras entre entidades (por ejemplo, una persona tiene una dirección, una dirección pertenece a una persona).
- **Transacciones**: MySQL ofrece un manejo robusto de transacciones, lo que asegura que las operaciones complejas sean atómicas, consistentes, aisladas y duraderas (ACID).

Ventajas de la Base de Datos NoSQL (MongoDB):

- **Escalabilidad**: MongoDB es altamente escalable y puede manejar grandes volúmenes de datos no estructurados, como registros de usuarios con información variable.

- **Flexibilidad**: MongoDB permite almacenar datos sin necesidad de seguir un esquema rígido, lo que lo hace ideal para aplicaciones que manejan datos con diferentes estructuras.

Desventajas:

- **MySQL**: La mayor limitación es la rigidez en la estructura de los datos, que puede ser un obstáculo cuando se requiere flexibilidad.
- **MongoDB**: Si bien es muy flexible, la ausencia de relaciones entre los datos puede dificultar ciertas consultas complejas, como las que involucran varias tablas o entidades relacionadas.

5. Dificultades y Aprendizajes

Dificultades:

- **Manejo de estados en React**: Uno de los desafíos fue gestionar correctamente los estados de los formularios para las operaciones CRUD. Usamos `useState` para manejar el estado de cada campo en el formulario y `useEffect` para realizar las solicitudes de lectura.
- **Configuración de conexiones**: Configurar correctamente las conexiones entre el backend y las bases de datos fue un desafío, especialmente en la integración de MongoDB, que requería especificar correctamente la URI de conexión.

Aprendizajes:

- La integración de bases de datos SQL y NoSQL en una misma aplicación web permite aprovechar lo mejor de ambos mundos: la integridad y las relaciones de MySQL, junto con la flexibilidad de MongoDB.
- Aprendimos a manejar diferentes tipos de conexiones de base de datos, adaptando las solicitudes de manera eficiente según el tipo de base de datos.

6. Conclusiones

Este trabajo práctico ha sido una excelente oportunidad para explorar las diferencias entre bases de datos relacionales y NoSQL, implementando operaciones CRUD en ambas. La elección de tecnologías como **React** para el frontend y **Spring Boot** para el backend permitió desarrollar una solución robusta

y escalable. Al utilizar **MySQL** y **MongoDB** de manera complementaria, pudimos entender mejor cuándo es conveniente usar cada tipo de base de datos, dependiendo de los requisitos de flexibilidad o estructuración de los datos.

Entrega del Proyecto

El código fuente completo y el informe detallado del proyecto se encuentran disponibles en un repositorio de **GitHub**. Además, se incluyen instrucciones claras para la instalación y ejecución del proyecto.