FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Informe de Trabajo Práctico Integración de Bases de Datos Relacionales y NoSQL

Base de Datos
Segundo cuatrimestre de 2024
Cátedra Merlino

Apellido/s	Nombres	Padrón	Correo electrónico
Álvarez	Joaquín	107183	joaalvarez@fi.uba.ar
Olaran	Sebastian	109410	solaran@fi.uba.ar
Villani	Cristian	93358	cvillani@fi.uba.ar
Hsieh	Cindy Teresa	108051	chsieh@fi.uba.ar
Basso	Catalina	108564	cbasso@fi.uba.ar
Ruiz	Lucas	109674	lruiz@fi.uba.ar

Índice

1. Introducción

Este trabajo presenta el desarrollo de una aplicación web interconectada con dos bases de datos, una del tipo SQL y otro NoSQL, pudiendose realizar en la misma operaciones de alta, baja, modificación y consulta en ambas bases.

El grupo decidio basar su proyecto en el desarrollo de una aplicación para el registro de viajes, contando con una seccion para guardar en una tabla los distintos viajes y otra para guardar comentarios acerca de las distintas aerolineas.

LINK AL REPOSITORIO

2. Elección de las Tecnologías

Para el trabajo práctico se realizó una API rest, que consta de las siguientes partes:

- Frontend implementado con el framework de Javascript, React. Elegimos este framework por su capacidad para integrarse eficientemente con la API REST. Además al ser una tecnología ampliamente adoptada, React garantiza acceso a una gran comunidad, recursos educativos, y bibliotecas externas, asegurando un soporte sólido durante el desarrollo del proyecto.
- Backend se utilizó java con su framework Spring Boot, por su soporte nativo para construir servicios RESTful, simplificando la integración entre el frontend y el backend.

Para las bases de datos se utilizó en la parte relacional MySQL, para manejar los datos estructurados, ya que es una buena opción dentro de las bases de datos existentes, en la misma se guarda la información estructurada de los viajes.

Para la base de datos que guarda los comentarios de las aerolíneas, se utilizó MongoDB , ya que se requería, guardar datos desestructurados y que superen los 255 caracteres.

3. Configuración y Conexión a Bases de Datos

Para la gestión de datos relacionales, se utilizó MySQL, configurado como servicio remoto a través de Railway. Esta elección se fundamentó en la capacidad de MySQL para manejar transacciones complejas y consultas estructuradas con alto rendimiento. En Railway, se aprovechó su integración nativa con bases de datos, lo que facilitó la conexión y administración. La base de datos se diseñó con una estructura eficiente, garantizando integridad referencial y optimización en el acceso a los datos. Adicionalmente, se utilizó MongoDB Atlas como base de datos no relacional, ideal para almacenar datos de naturaleza más dinámica. La conexión con MongoDB Atlas se configuró utilizando la URI remota, que asegura un acceso seguro y flexible desde cualquier ubicación. Esta configuración permitió manejar información no estructurada de manera eficiente, aprovechando las capacidades nativas de MongoDB para trabajar con documentos en formato JSON.

4. Diagrama de Arquitectura

La arquitectura de la aplicación es la siguiente:

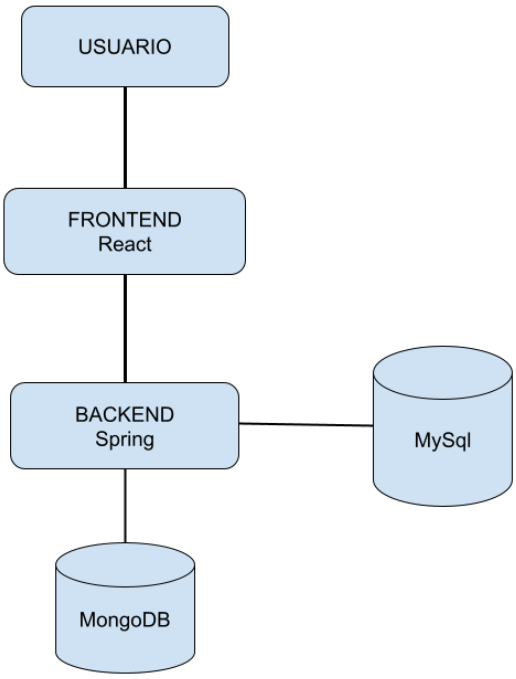

Figura 1: Arquitectura general del sistema

5. Operaciones CRUD en Interfaz

5.1. Leer Registros

Figura 2: Pagina Principal con vista a un listado de aerolineas y opiniones de

5.2. Crear Registros

Figura 3: Ventana para agregar viajes a la base de datos MySQL

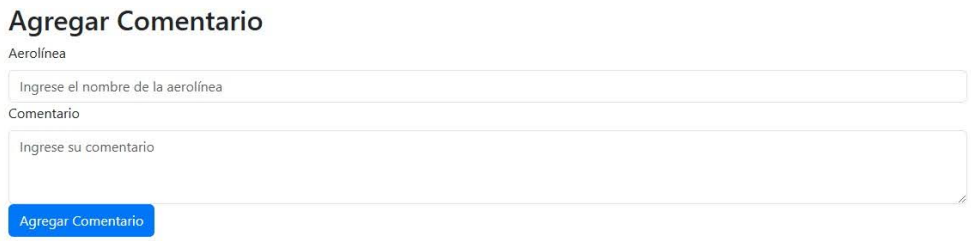

Figura 4: Ventana para agregar comentarios a la base de datos MongoDB

5.3. Actualizar Registros

Figura 5: Ventana para editar viajes en la base de datos MySQL

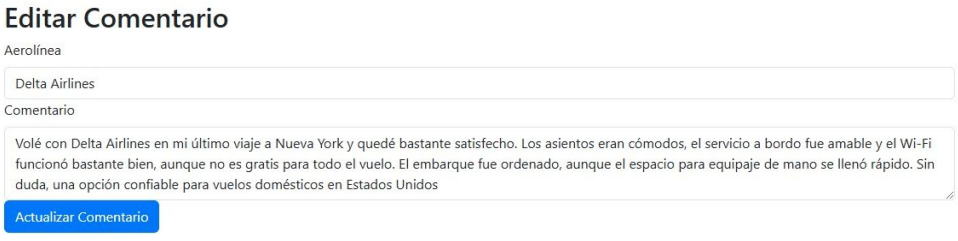

Figura 6: Ventana para editar comentarios en la base de datos MongoDB

5.4. Eliminar Registros

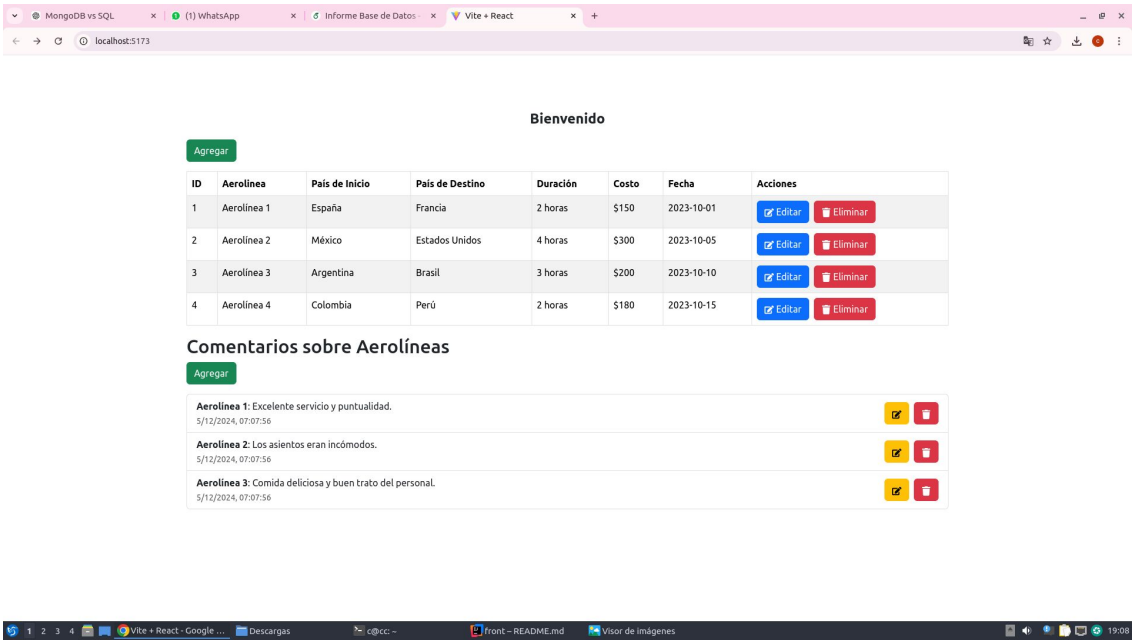

Figura 7: Ventana principal con cuatro viajes y tres comentarios

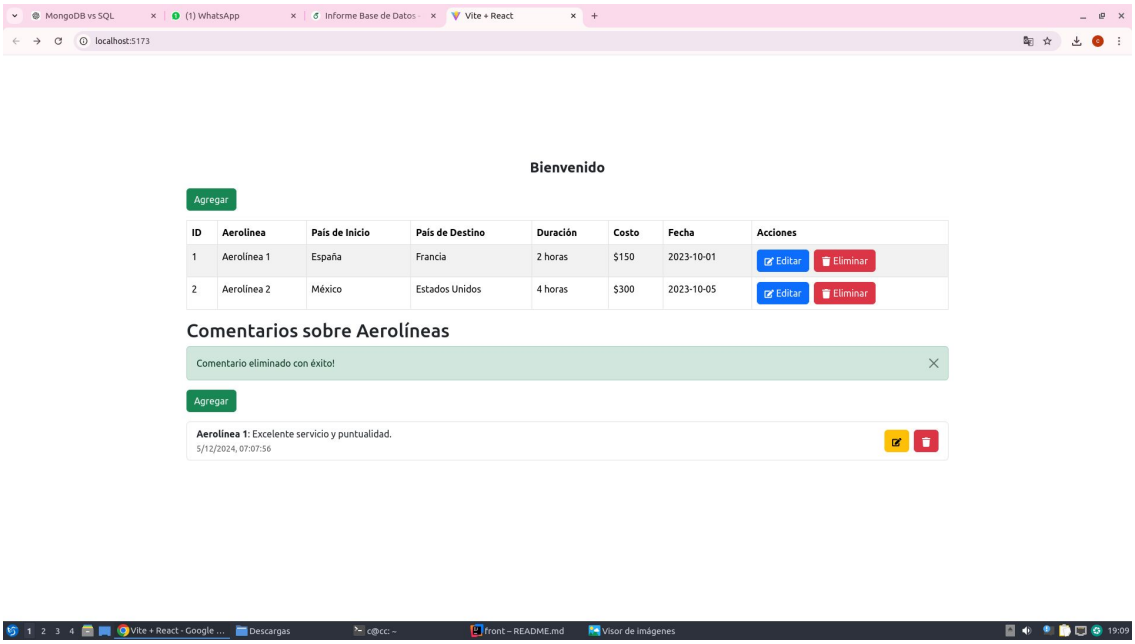

Figura 8: Eliminación de dos registros de viajes y un comentario

6. Comparación entre Bases de Datos Relacionales y NoSQL

6.1. Bases de datos relacionales

Ventajas

- **Portabilidad:** funcionan en varios dispositivos (computadoras, servidores, celulares).
- **Experiencia y madurez:** amplio soporte y herramientas desarrolladas por su larga trayectoria.
- **Atomicidad:** operaciones seguras y consistentes según criterios predefinidos.
- **Escritura sencilla:** es fácil de aprender y usar debido a su similitud con el lenguaje.

Desventajas

- **Dificultades de crecimiento:** costos elevados al escalar en volumen.
- **Cambios estructurales complicados:** las modificaciones en los datos requieren ajustes en la estructura.
- **Interfaz compleja:** mayor dificultad comparada con otras opciones.

Las bases de datos relacionales son preferibles en aplicaciones donde la integridad de los datos y las relaciones complejas son esenciales.

6.2. Bases de datos NoSQL

Ventajas

- **Ideal para big data:** maneja grandes volúmenes con eficiencia.
- **Menor administración:** simplificación en ajustes y mantenimiento automático.
- **Versatilidad:** adaptable a cambios sin necesidad de modificar estructuras.
- **Crecimiento horizontal:** escalable sin afectar el rendimiento.

Desventajas

- **Falta de atomicidad:** puede haber inconsistencias en los datos.
- **Documentación limitada:** información insuficiente sobre herramientas y uso.
- **Baja estandarización:** lenguaje y métodos varían entre motores.
- **Escasez de interfaces gráficas:** requiere conocimientos técnicos avanzados.

Por otro lado, las bases NoSQL son más convenientes en escenarios donde la escalabilidad, el manejo de datos no estructurados o la flexibilidad son primordiales.

7. Dificultades y Aprendizajes

Enfrentamos diversas dificultades en el sentido de la optimización y ejecución en diferentes entornos. Aunque logramos diseñar un modelo funcional en nuestra máquina principal, el despliegue en otras computadoras tuvo complicaciones inesperadas. Problemas como inconsistencias en las configuraciones del servidor de base de datos, diferencias en versiones de software, y limitaciones de hardware comprometieron el rendimiento y la correcta ejecución. Si bien finalmente logramos el funcionamiento, este proceso demandó tiempo y esfuerzo adicional, resaltando la importancia de considerar la portabilidad desde las primeras etapas del desarrollo.

En el tp se aprendió la diferencia entre bases de datos relacionales, las cuales son muy importantes para seleccionar el modelo de base de datos adecuado según los requerimientos del sistema. La flexibilidad que ofrecen las bases de datos no relacionales frente a cambios en los esquemas de datos. La relevancia de diseñar correctamente las consultas y optimizar las operaciones en ambas tecnologías para asegurar un buen rendimiento. Este proyecto también reforzó nuestra comprensión sobre la creación de APIs REST y la interacción entre diferentes capas de software, consolidando habilidades esenciales para el desarrollo de aplicaciones modernas y escalables.