TALLER DE PROGRAMACIÓN

GUÍA 3

MIGUEL ANGEL ZAMBRANO

HUGO ANDRES FORERO

SAMUEL ANTONIO TARAZONA

TALLER DE PROGRAMACIÓN

UNIVERSIDAD MANUELA BELTRAN

2024

Sesión 2

https://drive.google.com/file/d/1GY_xurS0SPa_VB8TRBVtTCnp5kjTOHZa/view?usp=drive_link

Actividad de Trabajo Autónomo

1. Diferencia entre ODBC y JDBC

ODBC (Open Database Connectivity) y JDBC (Java Database Connectivity) son dos tecnologías utilizadas para conectar aplicaciones a bases de datos, pero tienen algunas diferencias clave:

- ODBC:
  - Lenguaje: Generalmente se usa en aplicaciones desarrolladas en lenguajes como C o C++.
  - Plataforma: Es independiente del lenguaje de programación y está diseñado para ser utilizado en entornos no específicos de un lenguaje.
  - Interfaz: Proporciona una interfaz estándar para conectar aplicaciones a bases de datos, soportando múltiples bases de datos a través de un único controlador.
- JDBC:
  - Lenguaje: Específicamente diseñado para aplicaciones Java.
  - Plataforma: Integrado en el entorno Java, permite a las aplicaciones Java conectarse y ejecutar consultas en bases de datos.
  - Interfaz: Ofrece una API para interactuar directamente con bases de datos desde aplicaciones Java, utilizando drivers JDBC específicos.

2. Diferencias y Similitudes entre Bases de Datos Relacionales y NoSQL

Similitudes:

- Almacenan datos: Ambos tipos de bases de datos se utilizan para almacenar y gestionar datos.
- Manejo de grandes volúmenes de datos: Ambos tipos pueden manejar grandes volúmenes de datos, aunque NoSQL está más optimizada para grandes escalas.
- Acceso concurrente: Tanto las bases de datos relacionales como NoSQL ofrecen mecanismos para permitir el acceso concurrente a los datos.

Diferencias:

Estructura de los datos:

- Relacionales: Utilizan una estructura tabular con filas y columnas, y la información se almacena en forma de tablas con relaciones predefinidas entre ellas (llaves foráneas y primarias).

- NoSQL: No siguen un esquema rígido de tablas; en su lugar, los datos se organizan en estructuras más flexibles como documentos (JSON), pares clave-valor, columnas o grafos.

Lenguaje de consulta:

- Relacionales: Usan SQL (Structured Query Language) para gestionar y consultar datos.
- NoSQL: No tienen un lenguaje estándar de consulta. Utilizan lenguajes específicos según el tipo de base de datos (como MongoDB con su propio formato de consulta para bases de datos de documentos).

Escalabilidad:

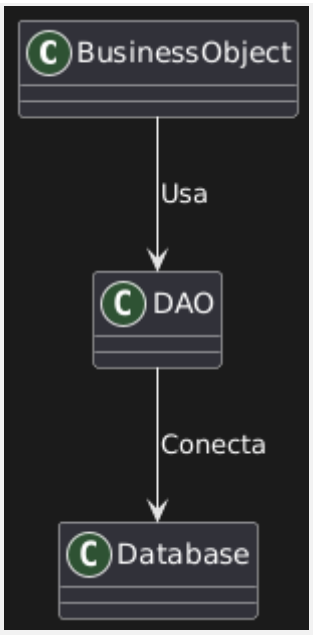
- Relacionales: Escalabilidad principalmente vertical (mejorar el hardware del servidor).
- NoSQL: Escalabilidad horizontal (añadir más servidores para repartir la carga).

Transacciones:

- Relacionales: Altamente consistentes con soporte completo para transacciones ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad).
- NoSQL: Generalmente están diseñadas para ser altamente disponibles y escalables, pero con menos énfasis en la consistencia fuerte; algunas implementan transacciones pero con menos garantías ACID.

3. Patrón Data Access Object (DAO)

El patrón DAO separa la lógica de negocio del acceso a los datos, proporcionando una interfaz para realizar operaciones CRUD sin exponer la implementación de la base de datos.



4. Patrones MVC y DTO

- MVC: Divide la aplicación en Modelo (datos), Vista (interfaz) y Controlador (gestiona la lógica entre ambos).
- DTO: Objeto simple que transporta datos entre capas o sistemas, sin lógica de negocio.

5. Qué es JSON?

JSON (JavaScript Object Notation) es un formato ligero de intercambio de datos, basado en pares clave-valor, utilizado principalmente en APIs web para comunicación clien¿Qte-servidor. Es fácil de leer y usar en cualquier lenguaje de programación.

Actividad de Trabajo Autónomo (Video)

https://www.youtube.com/watch?v=Vs1Y9tELtY8

Preguntas Orientadoras

Aprendizajes obtenidos:

1. **Conexión con JDBC**: Aprendí a conectar Java con bases de datos, lo cual será útil para desarrollar aplicaciones que manejen grandes volúmenes de datos.
2. **Consultas SQL en Java**: Entendí cómo realizar operaciones CRUD desde Java, algo esencial para manejar datos eficientemente en futuras aplicaciones profesionales.
3. **Seguridad con PreparedStatement**: Aprendí a evitar inyecciones SQL usando `PreparedStatement`, una habilidad clave para el desarrollo seguro de software.

Dificultades encontradas y soluciones:

1. **Carga del controlador JDBC**: Problema con la inclusión del JAR del controlador, solucionado verificando el classpath.
2. **Conexión a la base de datos**: Error en la URL de conexión, resuelto consultando la documentación de MariaDB.
3. **Manejo de ResultSet**: Dificultades al recorrer los resultados, solucionado con la implementación correcta de bucles y métodos adecuados.