Explicación de la Tarea: Desarrollo de un Componente de Acceso a Datos en Java

Objetivo:

Crear un componente en Java que permita realizar operaciones básicas sobre una base de datos, incluyendo conexión, consulta y cierre. Además, el componente debe ser configurable y capaz de emitir eventos cuando ocurran determinadas acciones.

1. Introducción al Problema

Contexto:

Para evitar escribir código repetitivo cada vez que queramos conectarnos a una base de datos, lo ideal es encapsular esta lógica en un componente reutilizable.

Requisitos:

El componente debe permitir:

- Configurar la conexión con la base de datos.
- Ejecutar consultas SQL y devolver resultados.
- Cerrar la conexión de manera segura.
- Guardar y cargar la configuración desde un archivo JSON.
- Disparar eventos cuando se realicen acciones clave.

2. Desglose en Módulos

A) Crear la clase DatabaseManager con los métodos principales

- connect() → Establece la conexión a la base de datos.
- executeQuery(String query) → Ejecuta una consulta SQL y devuelve los resultados.
- disconnect() → Cierra la conexión.

Ejemplo:

```
DatabaseManager dbManager = new DatabaseManager("jdbc:mysql://localhost:3306/test", 30);
dbManager.connect();
dbManager.executeQuery("SELECT * FROM usuarios");
dbManager.disconnect();
```

B) Implementar eventos

El componente debe permitir que otras partes del código reciban notificaciones cuando:

- Se establezca la conexión.
- Se ejecute una consulta.
- Ocurra un error.

Ejemplo:

```
dbManager.setOnConnected(msg -> System.out.println("EVENTO: " + msg));
dbManager.setOnError(msg -> System.err.println("ERROR: " + msg));
```

C) Persistencia de la configuración en JSON

El componente tiene que guardar y cargar su configuración desde un archivo JSON (db_config.json).

Ejemplo de JSON:

```
{
  "connectionString": "jdbc:mysql://localhost:3306/test",
  "timeout": 30
}
```

Podéis usar org.json para leer y escribir este archivo.

D) Creación de una API REST con Spring Boot

Para integrar el componente en un entorno real, debéis desarrollar una API con Spring Boot (por ejemplo) que permita:

- Configurar la base de datos (POST /api/database/config).
- Conectarse (GET /api/database/connect).
- Ejecutar consultas (POST /api/database/query).
- Desconectarse (GET /api/database/disconnect).

Ejemplo de configuración usando curl:

```
curl -X POST http://localhost:8080/api/database/config \
     -H "Content-Type: application/json" \
     -d '{"connectionString":"jdbc:mysql://localhost:3306/test","timeout":"30"}'
```

Agregar Dependencias en pom.xml

```xml
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.30</version>
</dependency>
```

3. Empaquetado del Componente como un Archivo JAR

Ejemplo:

Compilar el código

```
javac -cp .;gson-2.8.9.jar DatabaseComponent.java ConfigManager.java
```

Empaquetar en un archivo JAR

```
jar cf DatabaseComponent.jar DatabaseComponent.class ConfigManager.class
```

4. Recursos Adicionales

Documentación de JDBC: https://docs.oracle.com/javase/tutorial/jdbc/

Guía de Spring Boot REST: https://spring.io/guides/gs/rest-service/

Cómo leer y escribir JSON en Java: https://www.baeldung.com/java-org-json