Ampliacion Proyecto JSON Unidad 3 - PRÁCTICA - JDBC-H2

Tabla de contenido

Introducción:

Breve descripción del proyecto y su objetivo principal.

El presente proyecto tiene como objetivo ampliar una aplicación Java que realiza peticiones a una API de predicciones meteorológicas por concello, almacenando la información obtenida en una base de datos local H2 y MySQL. Para lograr este propósito, se ha extendido el código existente añadiendo un menú de opciones, la implementación de clases de conexión con H2 y MySQL, así como métodos de gestión para manipular los datos en ambas bases de datos. Además, se ha utilizado la clase LeerSQL, la cual permite ejecutar sentencias SQL contenidas en archivos, facilitando la creación de esquemas y tablas en la base de datos tanto para H2 como para MySQL.

Implementación de la conexión a la base de datos

Descripcion del proceso

Se han desarrollado dos clases para gestionar la conexión a la base de datos, una para H2 y otra para MySQL. Ambas clases proporcionan métodos para obtener una conexión activa a la base de datos y cerrarla correctamente una vez finalizada su utilización.

H2:

Descripción de la clase ConexionH2, explicando su propósito y funcionalidad.

Clase ConexionH2

La clase ConexionH2 facilita la conexión con una base de datos H2 en memoria. Utiliza el controlador JDBC proporcionado por H2 para establecer la conexión. Además, implementa métodos para obtener y cerrar la conexión de manera segura.

```java
// Clase para conectar con la base de datos H2 en memoria
public class ConexionH2 {

    public static Connection conn;

    // Datos de conexión a la base de datos
    private static final String URL = "jdbc:h2:mem:test";
    private static final String USUARIO = "";
    private static final String CONTRASENA = "";

    /**
     * Obtiene y devuelve una conexión a la base de datos.
     * Si la conexión ya está establecida, la devuelve; de lo contrario, intenta
     * establecerla.
     *
     * @return la conexión a la base de datos.
     */
    public Connection obtenerConexion() {
        // Verificar si la conexión ya está establecida
        if (conn == null) {
            try {
                // Cargar el driver de H2
                Class.forName(className:"org.h2.Driver");

                // Intentar establecer la conexión utilizando la URL, el usuario y la contraseña
                conn = DriverManager.getConnection(URL, USUARIO, CONTRASENA);

                // Imprimir mensaje de conexión exitosa
                System.out.println(x:"Conexión exitosa a la base de datos");
            } catch (Exception e) {
                // Capturar y manejar cualquier excepción que pueda ocurrir durante el proceso
                System.out.println(x:"No se encontró el driver de H2.");
                e.printStackTrace();
            }
        }
        // Devolver la conexión, ya sea la existente o la recién establecida
        return conn;
    }
```

MYSQL:

Descripción de la clase ConexionMYSQL, explicando su propósito y funcionalidad.

Clase ConexionMYSQL

La clase ConexionMYSQL permite la conexión a una base de datos MySQL. Emplea el controlador JDBC de MySQL para establecer la conexión. Proporciona métodos para obtener una conexión activa, cerrarla adecuadamente y configurar los parámetros de conexión, como el nombre de usuario, la contraseña y el nombre de la base de datos

```java
public class ConexionMYSQL {

	// Nombre de usuario para la conexión a la base de datos
	private static final String userName = "root";

	// Contraseña para la conexión a la base de datos
	private static final String password = "abc123.";

	// Sistema de gestión de bases de datos (DBMS) que se utilizará (MySQL en este caso)
	private static final String dbms = "mysql";

	// Dirección IP del servidor de la base de datos
	private static final String serverName = "127.0.0.1";

	// Número de puerto del servidor de la base de datos
	private static final String portNumber = "3306";

	// Nombre de la base de datos
	public static final String dbName = "predicciónconcellos";

	// URL para la conexión utilizando MariaDB
	private static final String URL = "jdbc:mariadb://localhost:" + portNumber + "/" + dbName;

	// URL para la conexión utilizando MySQL
	private static final String URL2 = "jdbc:mysql://localhost:" + portNumber;

	// Declaracion de variable estática que será utilizada para almacenar la conexión a la base de datos.
	private static Connection conn = null;

	/**
	 * Obtiene una conexión a la base de datos.
	 *
	 * @return La conexión a la base de datos.
	 */
	public Connection getConexion() {
		if (conn == null) {
			try {
				// Cargar el driver de MySQL
				Class.forName("com.mysql.cj.jdbc.Driver");

				// Obtener la className:conexión
				conn = DriverManager.getConnection(URL2, userName, password);
				System.out.println("Conexión exitosa a la base de datos");
			} catch (Exception e) {
				// Si ocurre algunax: excepción durante el proceso, se captura y se imprime un
				// mensaje de error junto con la traza de la excepción.
				System.out.println("No se encontró el driver de MySQL");
				e.printStackTrace();
```

Descripción del menú de opciones:

Explicación del propósito del menú y su función dentro de la aplicación.

Se ha implementado un menú de opciones para ofrecer al usuario diferentes funcionalidades dentro de la aplicación. El menú permite al usuario realizar acciones como conectar con la base de datos H2, crear tablas, insertar datos, entre otras opciones relevantes para la gestión de la información meteorológica.

Enumeración de las diferentes opciones disponibles en el menú y su significado.

```java
public static int OpcionesMenu(Scanner sc) {
    int opcion = -1;
    System.out.println(x:"1 - Conectar BBDD H2.");
    System.out.println(x:"2 - Crear Tablas en BBDD H2.");
    System.out.println(x:"3 - Insertar datos de prueba en la BBDD H2.");
    System.out.println(x:"4 - Cerrar conexion BBDD H2.");
    System.out.println(x:"5 - Por cada ciudad realizamos una peticion.");
    System.out.println(x:"6 - Guardar los datos de las 7 peticiones en el fichero CSV.");
    System.out.println(x:"7 - Mostrar datos Predicciones principales ciudades desde memoria");
    System.out.println(x:"8 - Guardar Predicciones en la BBDD H2");
    System.out.println(x:"9 - Mostrar datos por pantalla de BBDD H2");
    System.out.println(x:"10 - Conexion BBDD MYSQL");
    System.out.println(x:"11 - Cerrar conexion MYSQL");
    System.out.println(x:"12 - Insertar concello prueba");
    System.out.println(x:"13 - Crear Tablas en BBDD MYSQL");
    System.out.println(x:"14 - Insertar datos pruebas en BBDD MYSQL");
    System.out.println(x:"15 - Eliminar datos de todas las tablas.");
    System.out.println(x:"16 - Insertar datos Predicciones desde cache a BBDD MYSQL.");
    System.out.println(x:"17 - Mostrar datos tablas BBDD MYSQL.");
    System.out.println(x:"18 - SALIR.");
    try {
        opcion = Integer.parseInt(sc.nextLine());
    } catch (Exception e) {
        e.printStackTrace();
    }
    return opcion;
}
```

Detalles sobre cómo se implementa la lógica de selección de opciones.

La lógica de selección de opciones se implementa mediante un switch-case en el método Menu. Este método recibe como parámetros la opción seleccionada, así como las instancias de las clases ConexionH2 y ConexionMYSQL, que se utilizan para establecer las conexiones respectivas con las bases de datos.

Dentro del switch-case, se evalúa la opción seleccionada y se ejecuta el código correspondiente para cada caso. Por ejemplo:

Para la opción 1, se verifica si la conexión a la base de datos H2 ya está establecida. Si no lo está, se obtiene la conexión utilizando el método obtenerConexion() de la clase ConexionH2.
Para la opción 2, se verifica si la conexión a la base de datos H2 está establecida. Si lo está, se ejecuta el método ejecutarSentenciasFicheroSQL() de la clase LeerSQL para crear tablas en la base de datos.
Para la opción 3, se verifica si la conexión a la base de datos H2 está establecida. Si lo está, se llama al método insertarDatosPrueba() de la clase GestionPredicciones para insertar datos de prueba en la base de datos.

Este proceso se repite para cada opción, con la lógica específica correspondiente a cada caso. Cada opción puede implicar la interacción con la base de datos, la realización de operaciones específicas o la llamada a otros métodos de las clases definidas en el proyecto.

```
public static void Menu(int opcion, ConexionH2 conexionH2, ConexionMYSQL conexionMYSQL) {

    switch (opcion) {
        case 1: // Conectar BBDD H2.
            if (connH2 == null) {
                connH2 = conexionH2.obtenerConexion();
            } else {
                System.out.println(x:"La conexión ya está establecida.");
            }
            System.out.println();
            break;
        case 2: // Crear tablas en la BBDD H2
            if (connH2 != null) {
                if (LeerSQL.ejecutarSentenciasFicheroSQL(connH2, rutaCrearTablasMYSQL)) {
                    System.out.println(x:"Tablas creadas correctamente.");
                }
            } else {
                System.out.println(x:"Error. Primero debe de establecerse una conexión.");
            }
            System.out.println();
            break;
        case 3: // Insertar datos de prueba en la BBDD H2
            if (connH2 != null) {
                GestionPredicciones.insertarDatosPrueba(connH2);
                System.out.println(x:"Datos de prueba insertados correctamente.");
            } else {
                System.out.println(x:"Error. Primero debe realizarse la conexion la BBDD de H2.");
            }
            System.out.println();
            break;
        case 4: // Cerrar conexion H2.
            if (connH2 == null) {
                conexionH2.cerrarConexion();
            } else {
                System.out.println(x:"La conexión ya se encuentra cerrada.");
            }
            System.out.println();
            break;
        case 5: // Peticiones a la API con las principales ciudades
            listaPrediccionesCiudadesImportantes = RealizarPeticionesPredicciones();
            System.out.println();
            break;
        case 6: // Guardar los datos de las 7 peticiones en el fichero CSV
            if (listaPrediccionesCiudadesImportantes != null && descripcionParser != null
                    && nombreFicheroCSV != null) {
                guardarDatosEnCSV(listaPrediccionesCiudadesImportantes, descripcionParser, nombreFicheroCSV);
            } else {
```

Clases y objetos utilizados en la implementación:

Enumeración de todas las clases y objetos relevantes para el funcionamiento del proyecto.

Durante el desarrollo del proyecto, se han utilizado varias clases y objetos para facilitar la implementación de las funcionalidades requeridas. Estos incluyen:

Clase ConexionH2: Encargada de gestionar la conexión a la base de datos H2.

Clase ConexionMYSQL: Encargada de gestionar la conexión a la base de datos MYSQL.

Clase GestionPredicciones: Encargada de proporcionar métodos para insertar y gestionar datos en la

base de datos H2 y MySQL.

La clase LeerSQL proporciona métodos para leer archivos SQL y ejecutar las sentencias contenidas en ellos. Este componente ha sido fundamental para la creación de esquemas y tablas en la base de datos, tanto en H2 como en MySQL. La clase utiliza una instancia de BufferedReader para leer el archivo SQL línea por línea, y ejecuta las sentencias SQL utilizando un objeto Statement.

Métodos de obtención de datos:

Ejemplos de funciones:

- RealizarPeticionesPredicciones();
- obtenerIdConcelloPorNombre(),

Utilizados para obtener datos de predicciones meteorológicas y de concellos respectivamente.

Proceso de desarrollo:

Descripción paso a paso del proceso seguido para implementar la funcionalidad de conexión a la base de datos H2.

Durante el desarrollo del proyecto, se han seguido los siguientes pasos:

Análisis de requisitos: Se identificaron los requisitos adicionales para ampliar la aplicación existente.
Diseño de la solución: Se planificó la estructura del código los paquetes y se definieron las clases y métodos necesarios.
Implementación: Se escribió el código siguiendo las especificaciones del diseño.
Depuración y optimización: Se corrigieron errores y se optimizó el código para mejorar su rendimiento y legibilidad.
Documentación: Se documentarion los métodos y clases, así como los objetos más importantes de cada función.
Pruebas finales: Se realizaron pruebas para asegurar que todas las funcionalidades operaran correctamente.

Menciona cualquier decisión de diseño o arquitectura tomada durante el proceso de desarrollo.

Durante el proceso de desarrollo, se tomaron varias decisiones de diseño y arquitectura para garantizar la eficiencia, la escalabilidad y la mantenibilidad del proyecto. Algunas de estas decisiones incluyen:

Separación de responsabilidades: Se implementó un diseño basado en capas, donde cada componente del sistema tiene una responsabilidad específica. Por ejemplo, se separaron las clases encargadas de la lógica de negocio de las clases encargadas de la interacción con la base de datos.

Uso de patrones de diseño: Se aplicaron patrones de diseño como el patrón DAO (Data Access Object) para encapsular el acceso a los datos y facilitar la modularidad y reutilización del código.

Manejo de excepciones: Se implementó un manejo adecuado de excepciones para gestionar errores de forma robusta y proporcionar mensajes de error claros al usuario. Esto ayuda a mejorar la robustez y la usabilidad del sistema.

Optimización de consultas: Se optimizaron las consultas a la base de datos para mejorar el rendimiento del sistema. Esto incluyó la indexación adecuada de las tablas, el uso de consultas parametrizadas para prevenir ataques de inyección SQL y la minimización del número de consultas ejecutadas.

Conclusiones:

Recapitulación de los principales logros del proyecto.

La ampliación de la aplicación para gestionar predicciones meteorológicas ha sido un desafío gratificante que ha permitido poner en práctica los conocimientos adquiridos en el ciclo de programación. La implementación del menú de opciones y la gestión de la conexión con la base de datos H2 y también con la base de datos MYSQL han enriquecido la funcionalidad de la aplicación, proporcionando una experiencia más completa para el usuario agregando nuevas funcionalidades como mantener la persistencia de datos, consulta de datos desde la base de datos, etc.

Posibles áreas de mejora o expansión para futuras iteraciones del proyecto.

Mejora de la interfaz de usuario (UI): Implementar una interfaz gráfica de usuario (GUI) más atractiva y fácil de usar podría mejorar significativamente la experiencia del usuario. Esto incluiría el diseño de pantallas más intuitivas, el uso de gráficos y visualizaciones para presentar datos de manera más efectiva.

Referencias:

Enlaces o recursos utilizados durante el desarrollo del proyecto, como documentación de JDBC, H2, etc.

Documentación oficial de JDBC: https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/

Documentación de H2 Database: https://www.h2database.com/html/main.html