Documentación del Proyecto de Datos Meteorológicos

Este documento describe el proyecto Java para la obtención y visualización de datos meteorológicos de Galicia, España. El proyecto utiliza la API de Meteogalicia, almacena los datos en bases de datos MySQL y SQLite, y presenta la información a través de una interfaz gráfica Swing.

Arquitectura

El proyecto se divide en las siguientes clases principales:

- `ConnectMysql`: Gestiona la conexión y las operaciones con la base de datos MySQL.
- `ConnectSQLite`: Gestiona la conexión y las operaciones con la base de datos SQLite.
- `TiempoRepository`: Se encarga de la comunicación con la API de Meteogalicia y la obtención de los datos meteorológicos, además de la interacción con ambas bases de datos.
- `Tiempo`: Representa un registro de datos meteorológicos, conteniendo información como la localidad, estado del cielo, temperatura, etc.
- `PantallaPrincipal`: Interfaz gráfica Swing para la visualización de los datos.
- `Main`: Clase principal que inicia la aplicación.
- `CsvGenerator`: Clase encargada de generar el archivo CSV con los datos meteorológicos.

Clases

1. `ConnectMysql`

Esta clase gestiona la conexión a la base de datos MySQL y las operaciones CRUD (Crear, Leer, Actualizar, Borrar) sobre la tabla `tiempo`.

- **Atributos:**
  - `URL`, `USER`, `PASSWORD`, `DRIVER`: Constantes para la conexión a la base de datos.
- **Métodos:**
  - `conectar()`: Establece y devuelve una conexión a la base de datos.
  - `createDatabase()`: Crea la base de datos `tiempojson` y la tabla `tiempo` si no existen.
  - `insertTiempo(Tiempo tiempo)`: Inserta un registro de `Tiempo` en la base de datos.
  - `deleteDatosTiempo()`: Elimina todos los registros de la tabla `tiempo`.
  - `getTiempo(String ciudad)`: Obtiene un registro de `Tiempo` de la base de datos por nombre de ciudad.

2. `ConnectSQLite`

Esta clase gestiona la conexión a la base de datos SQLite y las operaciones CRUD sobre la tabla `tiempo`.

- **Atributos:**
  - `URL`: Constante para la conexión a la base de datos.
- **Métodos:**
  - `conectar()`: Establece y devuelve una conexión a la base de datos.
  - `createDatabase()`: Crea la tabla `tiempo` si no existe.
  - `insertTiempo(Tiempo tiempo)`: Inserta un registro de `Tiempo` en la base de datos.
  - `deleteDatosTiempo()`: Elimina todos los registros de la tabla `tiempo`.
  - `getTiempo(String ciudad)`: Obtiene un registro de `Tiempo` de la base de datos por nombre de ciudad.

3. `TiempoRepository`

Esta clase se encarga de la comunicación con la API de Meteogalicia para obtener los datos meteorológicos y de la interacción con ambas bases de datos.

- **Atributos:**
  - `FIND_PLACE_URL`, `WEATHER_URL`, `API_KEY`: Constantes para las URLs y la clave de la API.
  - `connect`: Instancia de `ConnectMysql` para la interacción con la base de datos MySQL.
  - `connectSQLite`: Instancia de `ConnectSQLite` para la interacción con la base de datos SQLite.
- **Métodos:**
  - `getTiempoSql(String tiempoName)`: Recupera los datos del tiempo desde la base de datos MySQL.
  - `getTiempo(String location)`: Obtiene los datos meteorológicos de la API de Meteogalicia para una localidad dada. Realiza dos llamadas a la API, una para obtener las coordenadas y el ID de la localidad y otra para obtener el tiempo.
  - `getFirstValue(JsonNode jsonNode, String nombreVariable)`: Extrae el primer valor de una variable específica del JSON de respuesta de la API.
  - `updateAllTiempoSql()`: Actualiza la base de datos MySQL con los datos meteorológicos de varias ciudades.

4. `Tiempo`

Esta clase representa un registro de datos meteorológicos.

- **Atributos:**
  - `localidad`, `estadoCielo`, `temperatura`, `viento`, `humedad`, `coberturaNubosa`: Almacenan los datos meteorológicos.
- **Constructor:**
  - `Tiempo(String localidad, String estadoCielo, String temperatura, String viento, String humedad, String coberturaNubosa)`: Constructor que inicializa los atributos.
- **Getters:**
  - Métodos getter para cada uno de los atributos.

5. `PantallaPrincipal`

Esta clase define la interfaz gráfica de usuario (GUI) utilizando Swing.

- **Componentes:**
  - `jComboBox1`: ComboBox para seleccionar la ciudad.
  - `jLabel2` - `jLabel8`: Etiquetas para mostrar información y títulos.
  - `estadoCielo`, `temperatura`, `viento`, `humedad`, `cobertura`: Campos de texto para mostrar los datos meteorológicos.
  - `btnDescargar`: Botón para descargar los datos en formato CSV.
- **Eventos:**
  - `jComboBox1ActionPerformed`: Se ejecuta cuando se selecciona una ciudad en el ComboBox. Obtiene los datos meteorológicos y los muestra en los campos de texto.
  - `btnDescargarActionPerformed`: Se ejecuta cuando se pulsa el botón "Descargar". Genera un archivo CSV con los datos meteorológicos mostrados y los inserta en la base de datos SQLite.

6. `Main`

Esta clase contiene el método `main` que inicia la aplicación.

- **Métodos:**
  - `main(String[] args)`: Crea una instancia de `TiempoRepository`, actualiza la base de datos MySQL y muestra la `PantallaPrincipal`.

7. `CsvGenerator`

Esta clase se encarga de generar el archivo CSV con los datos meteorológicos.

- **Métodos:**
  - `generateCSV(ArrayList<Tiempo> tiempos, String nombreArchivo)`: Genera un archivo CSV con los datos de la lista de objetos `Tiempo`.

Funcionamiento

1. La clase `Main` inicia la aplicación. Primero se actualizan los datos de la base de datos MySQL llamando al método `updateAllTiempoSql` de `TiempoRepository`. Luego, se crea y muestra la interfaz gráfica `PantallaPrincipal`.
2. El usuario selecciona una ciudad del ComboBox.
3. El evento `jComboBox1ActionPerformed` se dispara. Se llama al método `getTiempoSql` de `TiempoRepository` para obtener los datos meteorológicos de la ciudad seleccionada desde la base de datos MySQL.
4. Los datos se muestran en los campos de texto de la interfaz gráfica.
5. Si el usuario pulsa el botón "Descargar", se llama al método `generateCSV` de `CsvGenerator` y se crea un archivo CSV con los datos meteorológicos que se muestran en ese momento. Además, los datos se insertan en la base de datos SQLite.

Dependencias

- Librería `mysql-connector-java`: Para la conexión a la base de datos MySQL.
- Librería `sqlite-jdbc`: Para la conexión a la base de datos SQLite.
- Librería `jackson-databind`: Para el procesamiento de JSON.

Compilación y Ejecución

Para compilar y ejecutar el proyecto, se requiere JDK (Java Development Kit) y las dependencias mencionadas. Se puede utilizar un IDE como IntelliJ IDEA o Eclipse, o compilar y ejecutar desde la línea de comandos.