Memoria

Introducción

En este proyecto, hemos desarrollado una aplicación en Java que consulta una API de predicción meteorológica, procesa los datos en formato JSON y los almacena en un archivo CSV.

Para facilitar estas operaciones, hemos integrado las siguientes librerías:

- **OkHttp** para realizar peticiones HTTP.
- **Jackson** para procesar y manipular los datos JSON.
- **OpenCSV** para generar el archivo CSV con la información recopilada.

Uso de la API

La aplicación realiza una solicitud HTTP a la API de MeteoGalicia utilizando OkHttp. La respuesta, que incluye datos meteorológicos en formato JSON, se procesa mediante la librería Jackson. Posteriormente, los datos de interés se formatean y almacenan en un archivo CSV utilizando OpenCSV.

Documentación del código

El código está estructurado en métodos específicos:
1. El Main:

En el método principal de la aplicación tenemos la llamada html, donde le pasamos el api key y las variable que queremos buscar:

```
Request request = new Request.Builder()
        .url(url +city.getLongitude()+ "," + city.getLatitude() +
"&variables=sky_state,temperature,precipitation_amount,wind,relative_humidity,cloud
_area_fraction&API_KEY=" + API_KEY)
        .build();
```

2. processForecastData:
Le paso la respuesta JSON de la API y la procesa para extraer las variables de interés (estado del cielo, temperatura, etc.).

```
        // Obtener los valores de temperatura
        List<JsonNode> temperatureNode =
firstDay.path("variables").get(1).findValues("value");
        if (temperatureNode != null) {
            System.out.println("\nTemperatura:");
            for (JsonNode valueNode : temperatureNode) {
                System.out.println(valueNode.asDouble() + " C");
                temperature.add(valueNode.asDouble());
            }
        } else {
            System.out.println("No se encontró información sobre la temperatura.");
        }
```

Utiliza Jackson para acceder a las estructuras del JSON.

```
 // Parsear el JSON usando ObjectMapper
        ObjectMapper mapper = new ObjectMapper();
        JsonNode root = mapper.readTree(jsonResponse);

 List<JsonNode> skyStateNode =
firstDay.path("variables").get(0).findValues("value");
```

3. writeWeatherDataToCsv:

Recibe los datos procesados y genera un archivo CSV.
Utiliza OpenCSV para escribir los datos con cabeceras adecuadas.

```
// Escribir la cabecera del CSV
        writer.writeNext(new String[]{"Ciudad", "Estado del cielo", "Temperatura
(C)", "Probabilidad de lluvias", "Viento", "Humedad", "Porcentaje de nubosidad"});

        // Escribir los datos de cada ciudad en el archivo CSV
        for (int i = 0; i < cities.length; i++) {
            String cielo = String.join(",", weatherDataArray.get(i).getSky_state()); //
Esto de String.join coge un array y te concatena cada valor con una ,
            String temperatura = String.join(",",
weatherDataArray.get(i).getTemperature().toString());
            String lluvias = String.join(",",
weatherDataArray.get(i).getPrecipitation_amount().toString());
            String viento = String.join(",",
weatherDataArray.get(i).getWind().toString());
            String humedad = String.join(",",
weatherDataArray.get(i).getRelative_humidity().toString());
            String nubosidad = String.join(",",
weatherDataArray.get(i).getCloud_area_fraction().toString());
            writer.writeNext(new String[]{cities[i].getName(), cielo, temperatura,
lluvias, viento, humedad, nubosidad}); // Escribimos

        }
```

Librerías utilizadas

1. OkHttp (okhttp3.OkHttpClient)

Propósito: Esta librería se utiliza para realizar peticiones HTTP de forma eficiente y sencilla. En el proyecto, permite interactuar con la API de predicción meteorológica para obtener los datos en formato JSON.

Por qué es útil:

- Facilita el manejo de las solicitudes HTTP y las respuestas.
- Es ligera y rápida, ideal para aplicaciones con comunicación en red.

Ejemplo de uso:

- Creación de un cliente HTTP: OkHttpClient client = new OkHttpClient();
- Construcción de una solicitud: Request request = new

Request.Builder().url(URL).build();
- Ejecución de la solicitud: Response response = client.newCall(request).execute();

2. Jackson (com.fasterxml.jackson.databind)

Propósito: Es una librería para procesar datos en formato JSON. Aquí se utiliza para parsear las respuestas JSON de la API y extraer los datos relevantes.

Por qué es útil:
- Ofrece un mapeo sencillo entre estructuras JSON y clases de Java.
- Tiene una amplia gama de funcionalidades, desde lectura/escritura hasta validación de JSON.

Ejemplo de uso:
- Parsear un string JSON:

```
ObjectMapper mapper = new ObjectMapper();
JsonNode root = mapper.readTree(jsonResponse);
```

- Acceso a nodos específicos del JSON:

```
JsonNode firstDay = root.at("/features/0/properties/days/0");
```

3. OpenCSV (com.opencsv.CSVWriter)

Propósito: Facilita la creación y manipulación de archivos CSV. En el proyecto, se utiliza para generar un archivo que contiene los datos meteorológicos de las ciudades seleccionadas.

Por qué es útil:
- Simplifica la escritura y lectura de archivos CSV.
- Ofrece funcionalidades como manejo de cabeceras y delimitadores personalizados.

Ejemplo de uso:
-Creación del escritor de CSV:

```
CSVWriter writer = new CSVWriter(new FileWriter(csvFile));
```

-Escritura de datos en el archivo:

```
writer.writeNext(new String[]{"Ciudad", "Temperatura", ...});
```