Datalake To Datamart

Developing a Datalake

Escuela de Ingeniería Informática - ULPGC
Desarrollo de Aplicaciones para Ciencia de Datos
2º Curso Grado en Ciencia e Ingeniería de Datos

Resumen

El programa consiste en descargar datos de la AEMET, creando archivos para cada día en los que estarán los eventos. Estos archivos se guardan en un directorio llamado “datalake” y se crea un datamart con la información de este mismo filtrando las temperaturas máximas y mínimas. Posteriormente se crea una API con la información del datamart.

El trabajo consta de tres módulos: feeder, datamart provider y API.

El feeder descarga los datos de la AEMET y los filtra únicamente para Gran Canaria, cogiendo los campos de “timestamp”, “stationName”, “stationPlace” y “temperature”.

Se transforma la respuesta String a Gson, y se itera por cada estación para obtener los atributos solicitados. Para obtener los atributos solo de Gran Canaria, se hace un if estipulando las condiciones de latitud y longitud de esta zona una vez obtenidos los datos de la AEMET en cada iteración. Se crea el directorio si no existe, al igual que el archivo, y se escribe este último con los respectivos eventos.

Para evitar eventos duplicados, se crean dos sets: “events” y “newEvents”. “newEvents” son los eventos que se van descargando, y se van añadiendo sucesivamente a events cada vez que se produce una nueva descarga. Si un evento nuevo no está en “events”, se añade a “newEvents”. Para evitar información de varios días en un mismo archivo, se descartan los eventos que no son del día actual.

El datamart provider crea una base de datos con las temperaturas máximas y mínimas de cada día.

Para ello, crea dos tablas SQLite con la información del datalake: una para las temperaturas máximas y otra para las mínimas. Para obtener las temperaturas máximas, se leen los archivos iterando por cada evento. Si el valor de la temperatura del siguiente evento es mayor al anterior, se sobrescribe la variable MaxEvent, y así hasta terminar la lectura de todos los eventos. Se aplica un proceso análogo para las temperaturas mínimas.

Para evitar temperaturas duplicadas en la base de datos, se añade a la sentencia SQL INSERT “ON CONFLICT(date) DO NOTHING” estableciendo “date” como una PRIMARY KEY a la hora de hacer el CREATE TABLE.

En “API” se lee el datamart y se crea la API REST con la información de este último. Usa el enlace “localhost:4567/v1/places/with-max-temperature?from={date}&to={date}” para las temperaturas máximas y el “localhost:4567/v1/places/with-min temperature?from={date}&to={date}” para las mínimas, donde se muestra un jsonobject de los lugares con las temperaturas máximas o mínimas respectivamente en el rango de días especificado. Para ello, se seleccionan las temperaturas con una sentencia SQL, y se filtran según los parámetros ingresados por el usuario. Se creo un récord específico para la API, que solo contiene los valores de nombre, fecha y temperatura.

Índice

Recursos utilizados

Entornos de Desarrollo

IntelliJ IDEA 2022.2.3 (Ultimate).

Herramientas de Documentación

Microsoft Word 2021.

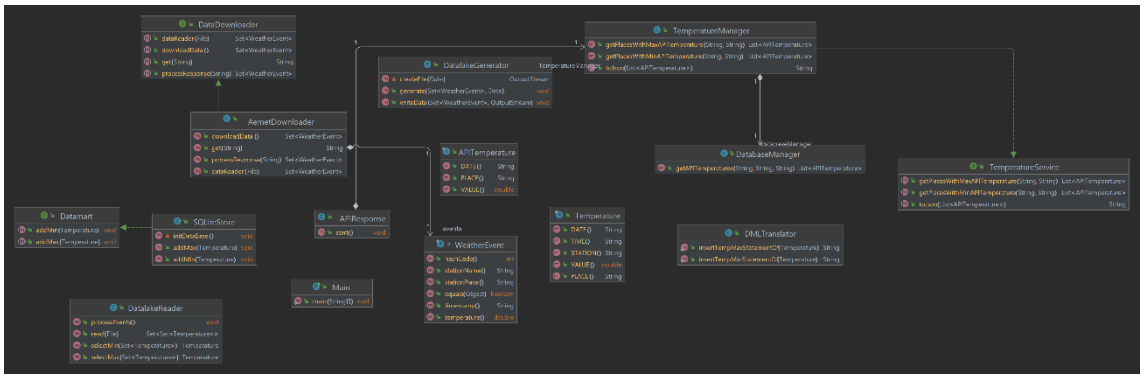
Diseño

Patrones y principios de diseño utilizados

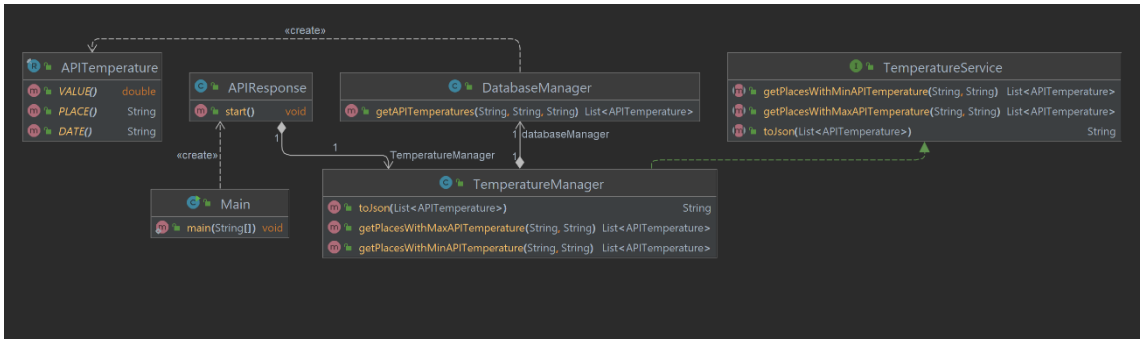
Se han aplicado los principios SOLID, y se ha usado el patrón de diseño MVC.

Diagrama de clases

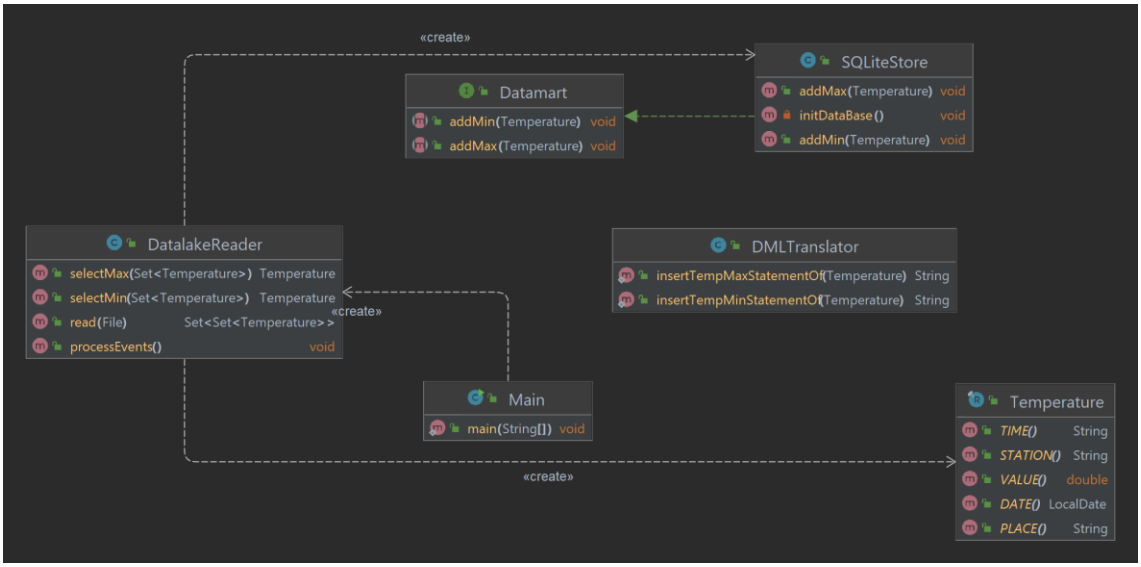
Aemet-dacd (proyecto entero):



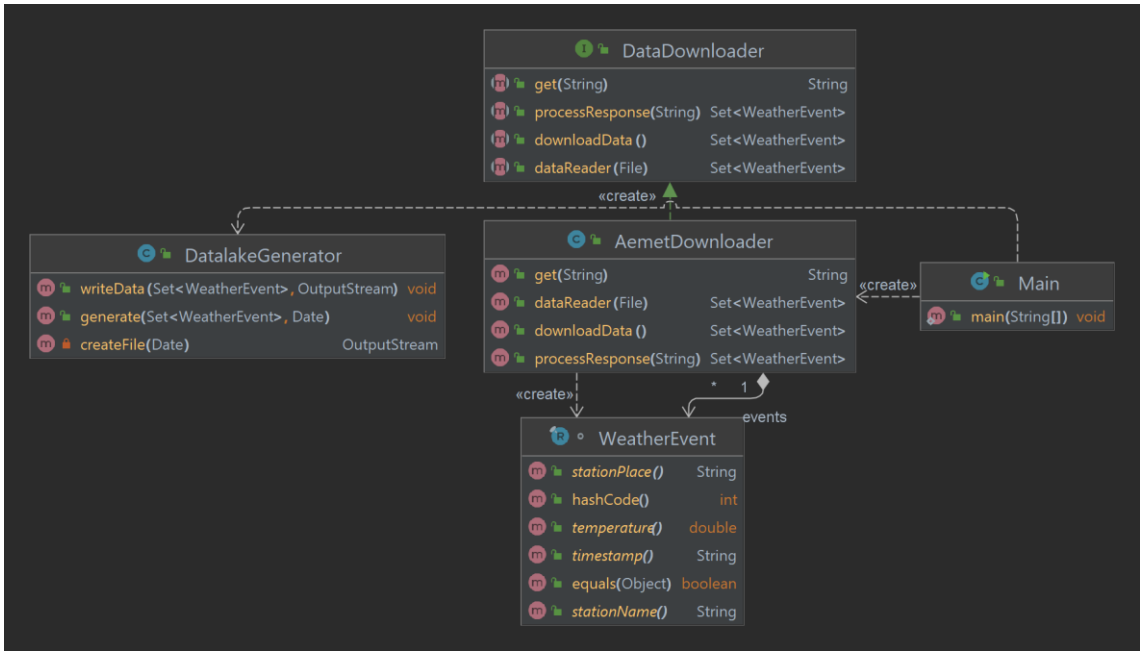
API:



Datamart-provider:



Feeder:



Conclusiones

En primer lugar, intenté evitar los eventos duplicados del datamart con sets, pero el método contains() siempre devolvía falso porque modificaba los elementos del set, y esto provocaba que cambiasen su HashCode. Entonces, tendría que haber usado el ON CONFLICT desde el principio teniendo en cuenta que es un recurso mucho más práctico y rápido que filtrar los datos en el mismo código, teniendo en cuenta que invertí varios días en este problema de manera infructuosa.

Por otra parte, son muy útiles las herramientas de desarrolladores de Google para obtener más información sobre las respuestas de la API, tendría que haberlas usado más.

Líneas Futuras

Por una parte, estaría bien que el usuario pudiese filtrar temperaturas no solo por días, sino también por lugar, o establecer un filtro para mostrar temperaturas máximas o mínimas a partir de una temperatura en concreto establecida por el usuario.

También le añadiría una interfaz para que el usuario se sintiese más cómodo y familiarizado a la hora de usar la API.

Bibliografía

https://stackoverflow.com/questions/30614498/element-is-present-but-set-containselement-returns-false

https://stackoverflow.com/questions/15609306/convert-string-to-json-array

https://stackoverflow.com/questions/54746814/jsonwriter-add-a-new-line

https://docs.oracle.com/javase/8/docs/api/java/util/Date.html

https://docs.oracle.com/javase/8/docs/api/java/time/LocalDate.html

Miriam Méndez Romero

13/01/2023 1.0