Proyecto datalake a datamart

DESARROLLO DE APLICACIONES PARA CIENCIA DE DATOS

CIENCIA E INGENIERÍA DE DATOS

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

LAURA LASSO GARCÍA
13/01/2023
VERSIONES: 1.0-SANPSHOT

Resumen

Este trabajo se divide en 3 módulos;
→**Feeder**: Se va a encargar de recoger los eventos de las últimas 24 horas de Aemet, serializar esos eventos y posteriormente guardarlos en ficheros (que tienen como nombre la fecha de esos eventos) dentro del datalake. Se compone de varias clases; la más importante es la Controller. Ahí se inicializan las interfaces que van a implementar las clases encargadas de recoger los datos y de escribirlos en el datalake. Luego tendrá un timetask que hará que se recojan datos y se escriban en el datalake cada hora. En esa misma clase, se leerán los datos y se compararán con los recogidos para luego pasarle esa diferencia de eventos (los que faltan por escribir en el datalake) a la función store() de la clase SystemDatalake.
→**Datamart-builder**: Este módulo se va a encargar de leer los datos de los archivos que se encuentran en el datalake (ya con todos los datos del día completos) y filtrar de cada archivo (es decir, de cada día) el evento con temperatura máxima y el evento con temperatura mínima. Luego, esos eventos los escribirá en el datamart, que cuenta con las tablas correspondientes a las temperaturas máximas y las temperaturas mínimas.
→**Temperature-api**: Este módulo se encargará de crear una api que consulte los datos del datamart y los pueda filtrar por fecha. Por ello, deberá tener una clase que lea esos eventos del datamart (sqliteReader), luego otra que se encargue de seleccionar el evento que tenga la temperatura máxima y mínima y tenga una función que se encargue de serializar esos eventos (EventsManager). Por último, deberá tener otra clase que se encargue de definir las peticiones get (SparkWebService).

Índice

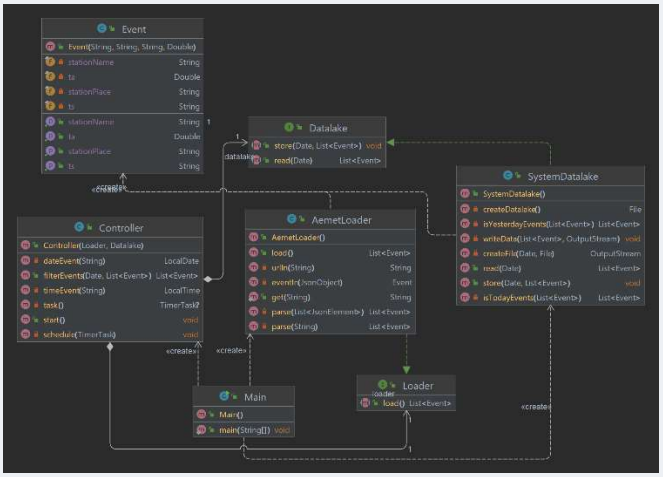
Recursos utilizados

El entorno de desarrollo es IntelliJ e IntelliJ Ultimate y las herramientas de documentación es Word.

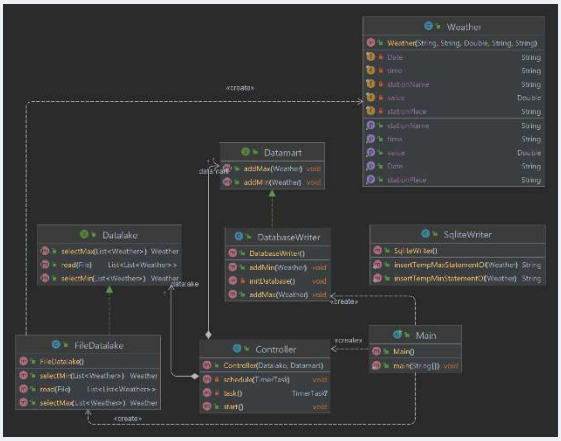
Diseño

Los principios implementados son los SOLID para así mantener un código limpio, ordenado, fácil de mantener de manera que podamos facilitar la nueva entrada de funcionalidades a nuestra aplicación y corrección de errores. Tal y como podemos ver, cada módulo implementado se encarga de una sola responsabilidad, como habíamos explicado antes.

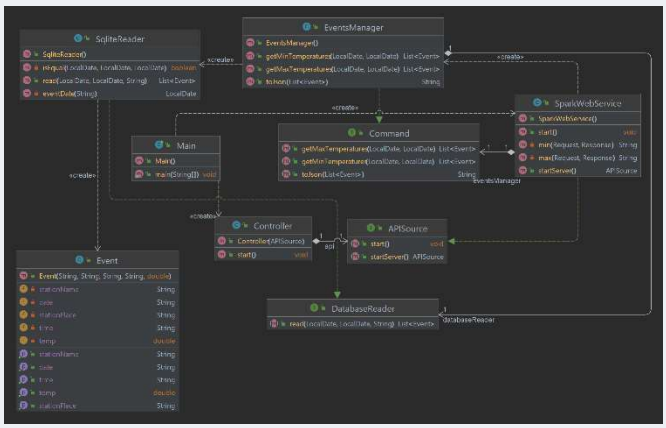
En cuanto al diagrama de clases y relaciones de dependencia, se ha realizado con IntelliJ IDEA Ultimate, gracias a una de sus funcionalidades implementadas. Cada módulo tendrá su propio diagrama de clases:
Feeder→



Datamart-builder→



Temperature-api→



Conclusiones

Este proyecto se puede enfocar de múltiples formas; no usando las clases POJO o implementar menos módulos. Sin embargo, he optado por usar más módulos para que el código sea mucho más fácil de comprender y que de esta forma esté más ordenado y sencillo.

Por otro lado, en el módulo de la API, se podría haber serializado de otra manera, como por ejemplo haber usado una de las funciones que implementa Gson. Sin embargo, he decidido optar por crear una función que serialice un objeto tipo Event, ya que a lo largo del proyecto siempre se trata con objetos de tipo Event.
También, en el módulo datamart-builder, se hubiese podido usar sentencias sql en donde te haga el filtrado de eventos según las fechas, en vez de usar streams que es como está desarrollado en el proyecto adjunto.

Líneas futuras

En cuestión de los usos que se pueden dar a este proyecto desde el punto de vista empresarial, es poder desarrollarlo de manera que, desde una aplicación podamos consultar las temperaturas máximas y mínimas de un sitio. Por ejemplo, desde una aplicación del tiempo, podamos ver las temperaturas de una fecha.

Bibliografía

Principios del diseño de clases - Adictos a la Informática (adictosalainformatica.com)
Doce principios de diseño que todo desarrollador debería conocer (genbeta.com)