ULPGC
Universidad de
Las Palmas de
Gran Canaria

Escuela de
Ingeniería Informática

Developing a datalake

2º curso

Grado en Ciencia e Ingeniería de Datos

Desarrollo de Aplicaciones para Ciencia de Datos

Jaime Ballesteros Domínguez

13/01/2023

Tercera versión

Última revisión: 13/01/2023

Resumen

Explicaremos su funcionamiento parte por parte en rasgos generales. Dichas clases son:

- Feeder: El feeder es el encargado de obtener los datos de la AEMET y guardarlos en ficheros. Lo primero que hacemos es conectarnos a la página de la AEMET y obtenemos todos los datos de las últimas 24 horas filtrándolos mediante las latitudes y longitudes mínimas y máximas de la isla. Posteriormente lo guardamos como un objeto de la clase Weather, la cual habremos creado previamente y procedemos a crear los ficheros. Para no entrar en excesivo nivel de detalle, la carpeta datalake, donde guardaremos los ficheros txt diarios será creada solo una vez y, posteriormente iremos fabricando los txt. El programa comprobará si ya hay algún fichero txt que corresponda con las fechas de las últimas 24 horas y creará o actualizará dicho elemento con los datos recogidos de la AEMET.
- Datamart-provider: La misión del datamart es guardar en una base mostrar en dos tablas los valores máximos y mínimos de cada día. Lo primero que haremos será destruir las tablas de datos en caso de que hayan sido creadas. Después iremos analizando línea a línea todos los eventos que hay en cada txt y extraeremos tanto el valor máximo como el mínimo. Seguidamente, crearemos las correspondientes tablas si no han sido formadas todavía e iremos introduciendo los datos máximos y mínimos de cada fichero en su respectiva tabla. Una vez que se acabe la ejecución, podremos visualizar los valores extremos de cada día en nuestra datamart.db.
- Temperature-service: Para acabar, el datamart deberá establecer una conexión con la API y, mediante llamadas, podremos obtener los máximos o mínimos diarios en el rango de días que le pasemos. Lo primero que haremos será establecer la conexión con la API, para así poder realizar las peticiones. Una vez introducido un día inicial y uno final, obtendremos dichos elementos de la petición y comenzaremos con el filtrado. Para ello, nos conectaremos con la tabla de datos creada previamente y le solicitaremos los valores que poseen. Una vez tengamos los datos totales, filtraremos en base a los días que hemos introducido y que habremos recogido de la petición e introduciremos todas las coincidencias en una lista. Ya con la lista de coincidencias completada, pasaremos cada elemento a un nuevo formato parecido a la clase Weather y los introduciremos a una lista que posteriormente será el resultado que imprimiremos en formato json.

Cabe destacar que, tanto el feeder como el datamart-provider poseen un timertask para actualizar los datos cada hora. Al ejecutar, se irán creando en el orden en el que hemos explicado, los tres módulos.

IMPORTANTE: **Al descargar el archivo se ha de cambiar tanto el Working Directory como Path to JAR del directorio de los artefactos con la ruta absoluta del nuevo fichero, ya que su ruta predeterminada se corresponde a mi entorno de trabajo y por lo tanto daría error.**

Índice

Recursos utilizados

Entornos de desarrollo: El entorno de desarrollo utilizado para este trabajo fue IntelliJ IDEA desarrollado por JetBrains.

Herramientas de control de versiones: El software de control de versiones utilizados ha sido Github, en el cual hemos introducido tanto el código en sí, como esta memoria y los diagramas de clases correspondientes.

Herramientas de documentación: Para documentar el trabajo realizado se ha utilizado el software de tratamiento de textos Microsoft Word.

Diseño

En este trabajo se intentó seguir el estilo de diseño Model View Controller (MVC). El feeder se corresponde al modelo, ya que es el encargado de proporcionar la información requerida, siendo en este caso en forma de ficheros txt. Por su parte, el temperatura-service sería un ejemplo de la vista ya que es el que, a través de la API, muestra al usuario los datos que él o ella desee. Para acabar, el datamart-provider se identificaría como controlador ya que es el encargado de gestionar los eventos y datos, siendo crucial al presentar los datos del feeder en tablas, las cuales serán utilizadas por el temperature-service para poder mostrar un resultado a las llamadas API.

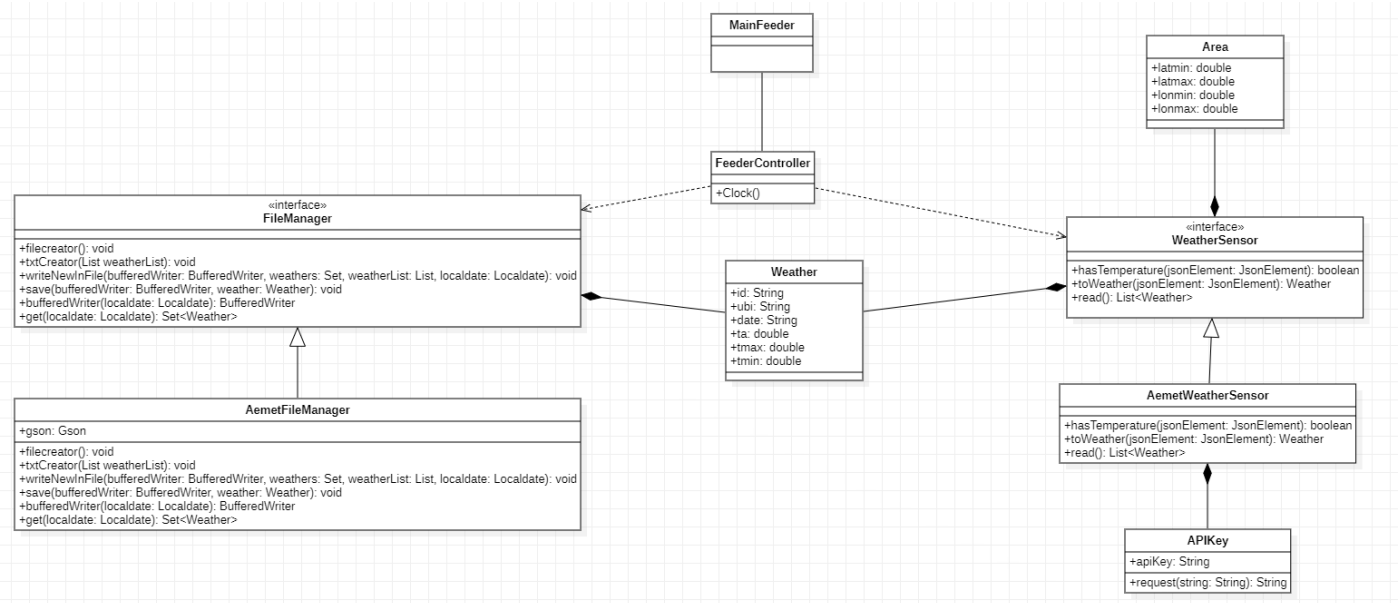

Diagrama de clases Feeder

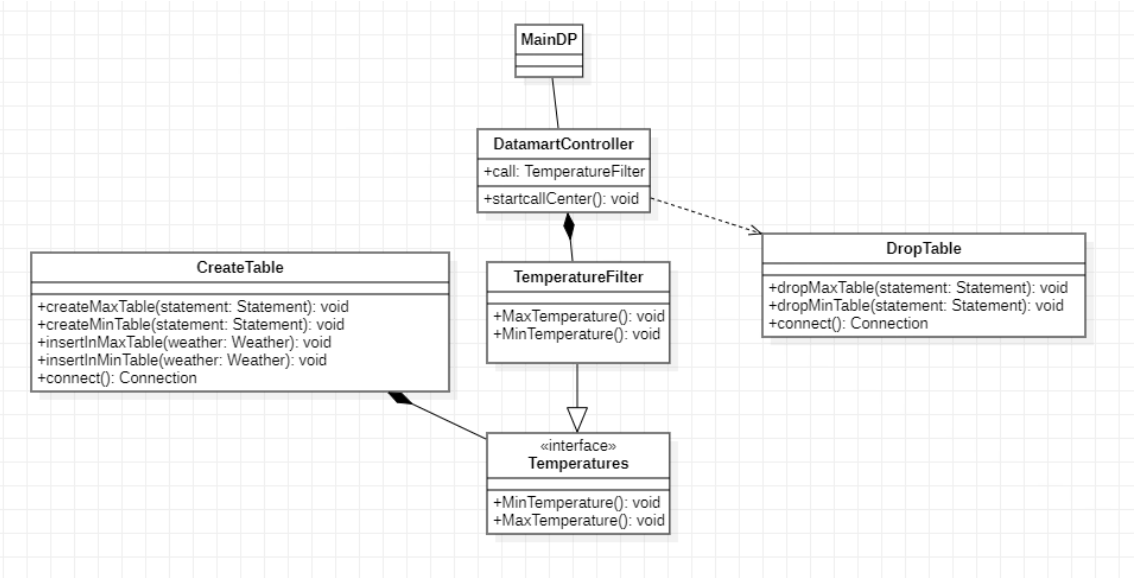

Diagrama de clases Datamart-provider

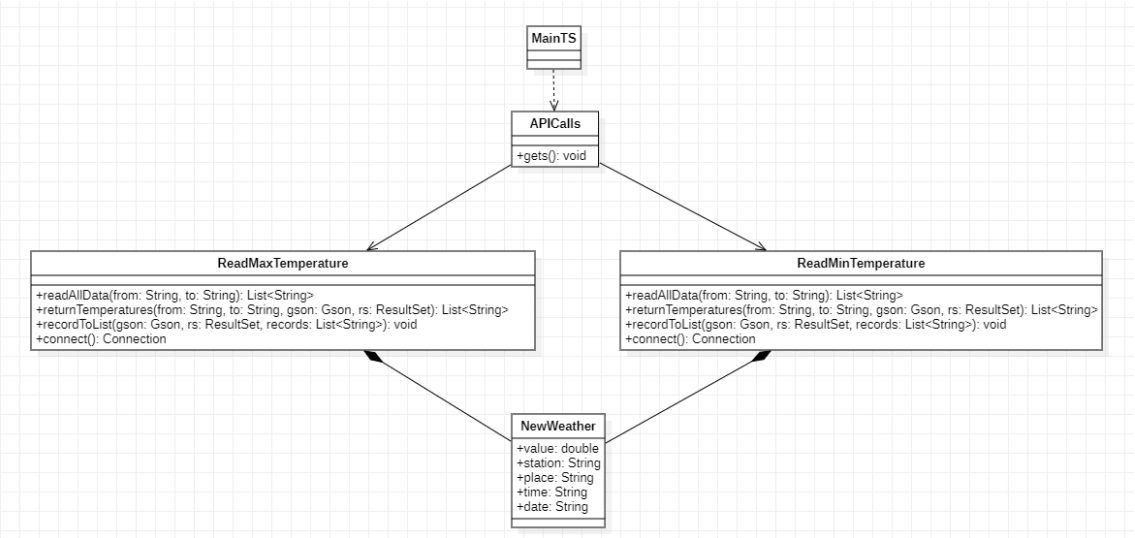

Diagrama de clases Temperature-service

Aquí se puede apreciar los diagramas de clases resultantes, el cual se puede ver con mayor calidad en el repositorio.

Conclusiones

En el proceso de realización de este trabajo he podido aprender tanto métodos como conceptos nuevos y formas de trabajar con ellos. Algunos ejemplos pueden ser:

- La creación de carpetas y ficheros txt a partir de código, además de la comparación y comprobación y actualización de datos dentro de dichos ficheros.
- Tratar con elementos BufferedWriter y Localdate así como con todas las maneras de representar las fechas.
- El uso de un TimerTask para controlar las ejecuciones de datos.
- Usar clases y funciones creadas en otros módulos, como por ejemplo la clase Weather.
- El uso de la instrucción SELECT de Sqlite para la obtención de datos.

Líneas futuras

Para que el producto obtenido se pueda convertir en uno comercializable se podría proporcionar la información al pasarle también el nombre de la zona geográfica que se quiere analizar en vez de tener que estar introduciendo unas coordenadas.

Bibliografía

https://www.appsdeveloperblog.com/convert-a-json-array-to-a-list-with-java-jackson/

https://community.kodular.io/t/how-to-convert-jsonarray-to-list/175107

https://www.w3schools.com/js/js_json_arrays.asp

https://stackoverflow.com/questions/49244585/java-arraylist-how-to-ignore-a-value-from-the-object-when-using-contains

https://www.baeldung.com/gson-string-to-jsonobject

https://www.digitalocean.com/community/tutorials/java-filewriter-example

https://stackoverflow.com/questions/12595019/how-to-get-a-string-between-two-characters

http://www.beginwithjava.com/java/file-input-output/writing-text-file.html

https://www.w3schools.com/java/java_files_read.asp

https://stackoverflow.com/questions/13185727/reading-a-txt-file-using-scanner-class-in-java

https://www.geeksforgeeks.org/file-getpath-method-in-java-with-examples/

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

https://www.baeldung.com/java-datetimeformatter

https://www.digitalocean.com/community/tutorials/java-timer-timertask-example

https://www.delftstack.com/howto/java/java-create-file-if-not-exists/

https://www.java67.com/2015/07/how-to-append-text-to-existing-file-in-java-example.html

https://stackoverflow.com/questions/7869006/import-a-custom-class-in-java

https://www.youtube.com/watch?v=HIuNUN3I-bk

https://www.youtube.com/watch?v=cNVzAmLGBYI

https://www.w3resource.com/java-tutorial/util/date/java_date_before.php

https://refactorizando.com/como-convertir-string-a-localdate/