Weather datalake – Memoria

Desarollo de Applicaciones para Ciencia de Datos

Grado en Ciencias e Ingeniería de Datos

Escuela de Ingeniería Informática

Universidad de Las Palmas de Gran Canaria

Resumen

El proyecto de Weather Datalake consiste de tres programas escritos en Java:

- **Weather feeder**: Se descarga datos meteorológicos de una API y los guarda en un datalake.
- **Weather datamart**: Lee los datos del datalake y crea o actualiza un datamart para obtener el tiempo y lugar de extremos en la temperatura.
- **Weather service**: Un servidor HTTP que provee un API REST para obtener el tiempo y lugar de extremos de temperatura.

Índice

1. Recursos utilizados

Los siguientes recursos se utilizaron para la realización del proyecto:

- El IDE usado es **IntelliJ IDEA**
- Para la versión de control se utilizó **git** con **GitHub**
- Para la escritura de la memoria, se utilizó LaTeX
- Para los diagramas de clase se utilizó **StarUML**

2. Diseño

2.1. Weather feeder

2.1.1. Como usar

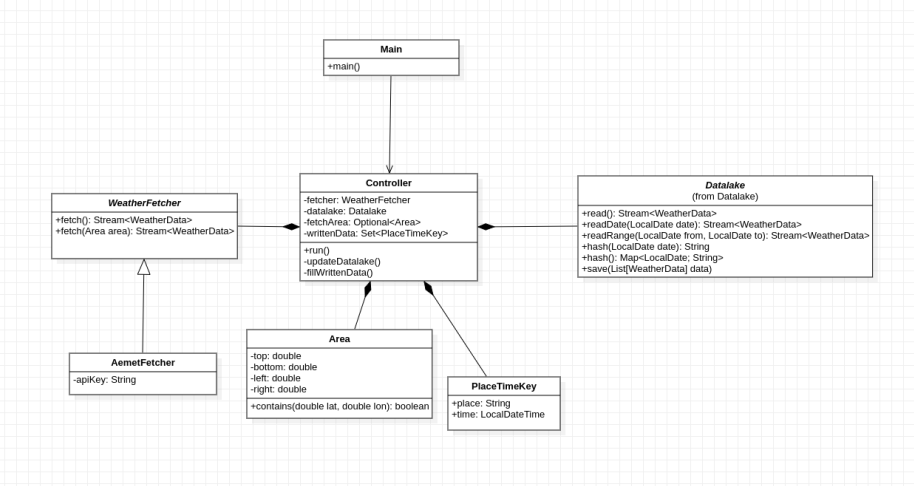
El programa necesita que se le pasen dos variables del entorno para funcionar:

- `API_KEY`**:** ya que actualmente el programa usa Aemet como fuente de datos meteorológicos, necesita esta clave para acceder a Aemet.
- `DATALAKE_PATH`**:** Un camino a una carpeta que será el datalake.

2.1.2. Funcionamiento

El API de Aemet provee de datos meteorológicos colleccionados en las últimas 24 horas de estaciones de toda España. Por esto, la parte de actualización del datalake que coge los datos de la API funciona se ejecuta cada hora. Algo notable que nó esta documentado por Aemet es que no todas las estaciones son actualizadas cada hora. Hay algunas estaciones que solo actualizan sus datos cada dos horas, y otras estaciones que tardan hasta el próximo día para actualizarse. Por lo que no basta con tener un contador de la última hora actualizada, filtrando todos los eventos antes de esta para guardarlos al datalake, dado que algunos eventos que han tardado varias horas despues de su grabación serían erroneamente filtrados y no guardados. Esto se puede observar, por ejemplo, con que el feeder guarda 23 nuevas observaciones, y 27 en la siguiente hora, alternando entre estos. Para solucionar esto, el feeder usa un Set del producto cartesiano del tiempo y estación de cada evento, guardando todos los eventos que no aparezcan en este set. Al inicializar el programa, se leen los eventos del datalake para construir el set con los eventos de las últimas 24 horas. Y al obtener nuevos eventos del datalake, se van borrando los elementos del set que ocurrieron hace más de 24 horas, para no llenar la memoria.

2.1.3. Diagrama de clases



2.2. Weather datamart

2.2.1. Como usar

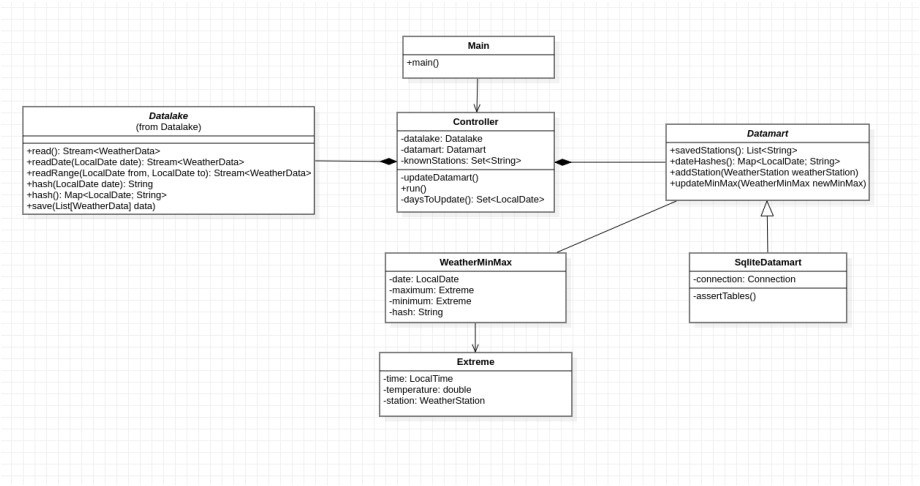
El Weather datamart necesita un camino al datalake y otro al datamart para su funcionamiento. El camino al datalake se especifica usando la variable de entorno `DATALAKE_PATH`. Y el camino al datamart se especifica como argumento del programa.

2.2.2. Funcionamiento

El programa encargado con el datamart transofma el contenido del datalake a una base de datos que indica la temperatura máxima y mínima de un dia. Para la implementación de esta, una decisión es sobre como se debe actualizar el datamart. Si se reconstruye desde cero, o si se intenta actualizar. Dado que el datamart tendrá que ser actualizado cada hora, la estrategia de construirlo desde cero no es muy bueno cuando el datalake crezca a tener cientos de días, dado que cada hora tendría que estar procesando los datos meteorológicos de cada estación en Gran Canaria de cada hora de los últimos varios meses. En cuanto a la estrategia de actualizar el datamart, sube la cuestión sobre como se sabe cuales días hay que actualizar. Una estrategia ingenua sería actualizar solo los cambios de hoy. Pero hay que acordarse que se posible que Aemet no presente algunos eventos de varias estaciones hasta el siguiente día, por lo que esta estrategia se estaría olvidando de nuevos eventos que ocurrieron el día pasado. La técnica utilizada para la implementación hace uso del hecho de que cada día en el datalake corresponde a un archivo, y cada día en el datamart corresponde a una tupla. De forma de que se sabe si han ocurrido cambios con el uso de hashes, específicamente el de MD5. En el datamart, las tablas que

contienen las máximas y mínimas temperaturas además contienen un atributo que corresponde al hash del archivo que tuvo cuando se computó los extremos del día. De forma de que si hay nuevos eventos en un día, este archivo tendría un hash diferente de forma de que el programa sabrá de que hay que actualizar este dia en el datamart.

2.2.3. Diagrama de clases



2.3. Weather service

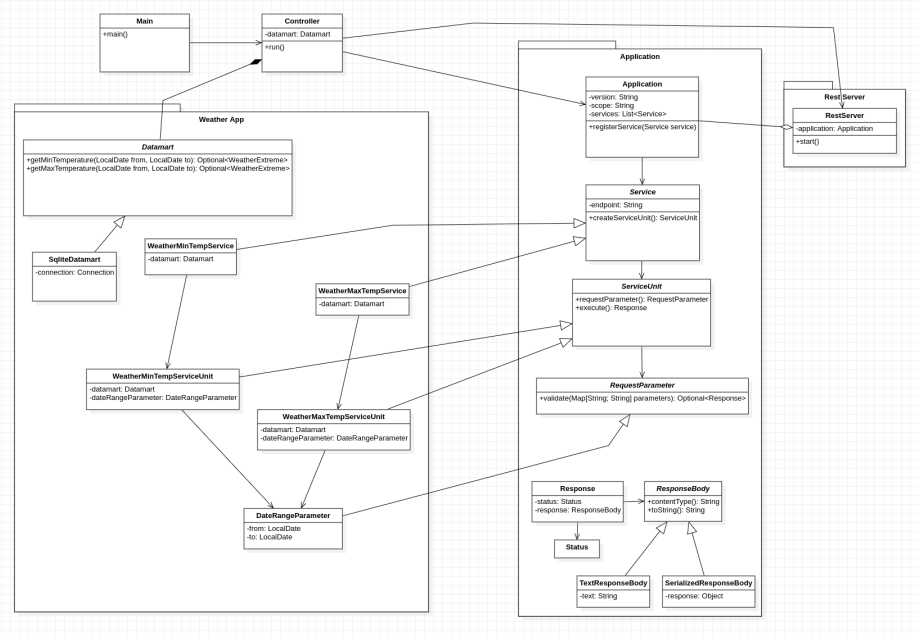
2.3.1. Como usar

El Weather service únicamente necesita el path al datamart. Este se proporciona como un argumento al programa cuando se ejecuta. Debe ser notable que si no hay un archivo sqlite en el momento de arranque, el programa se para.

2.3.2. Funcionamiento

En el servicio REST, se intenta segregar los detalles de la implementación del servidor REST con lo que es la lógica de negocio del proyecto. Existe el módulo application, que pretende set una abstracción de una application, independiente de que típo de lógica de negocio o de como se presentaría al usuario. Esta consiste de una applicación con un conjunto de servicios. De forma que además cada servicio tiene una clase asociada para obtener varios parametros de la petición. Esta clase se encarga de no solo obtener los parametros de la petición, sino que además de validarlos. De forma que el servicio puede preocuparse solo de la lógica del negocio, sino que también tendrá los datos que necesita ya comprobados y con el tipaje adecuado. Cada servicio tiene que tener en ella su propia

2.3.3. Diagrama de clases

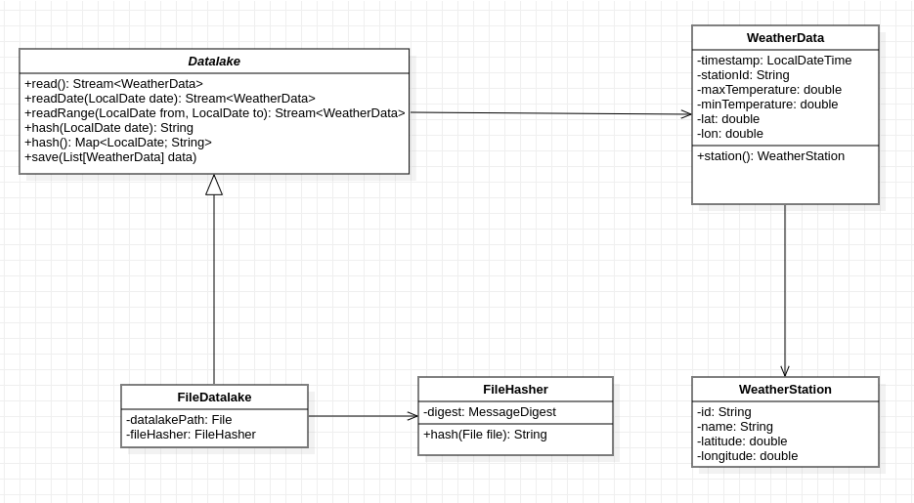


2.4. Datalake

Además de los tres programas del proyecto, también hay un paquete para el manejo del datalake.

Este

2.4.1. Diagrama de clases



3. Conclusiones

Un aspecto en el carece el programa es en la falta de herramientas adecuadas en cuanto al logging. Actualmente solo se utiliza `system.out.println` y `system.err.println`. Pero sería mejor utilizar librerías como `SLF4J` y `LOG4J`. La razón por la que no se pudo añadir estos a los programas es por una falta de tiempo.

Otro aspecto en el que se podría mejorar el proyecto es expandiendo el área del cual se capturan los datos, pero además proporcionando más control al usuario mediante una mayor variedad de criterios con los que puede selecionar o filtrar los resultados que quiere al realizar la petición al API Rest.

4. Líneas futuras

Actualmente, el proyecto puede tener varios beneficios en cuanto a temas científicos o agriculturales, sabiendo cuando y donde Canarias tuvo ciertas temperaturas bajas o altas puede ser útil para encontrar explicaciones a varios fenómenos como puede ser el de las cosechas de varias fincas. Pero si el proyecto se expande a que además tenga capabilidades de predicción. La habilidad de

predecir que partes de la isla tendrán ciertas temperaturas dado unos días puede ser interesante a la industria turística, dado que podría indicarle a turistas a qué partes de la isla pueden ir para sentir los climas más caloríficos o fríos que provee Gran Canaria.

Aurora Zuoris

La memoria se ha creado el 13 de enero de 2023 con LaTeX

13 de enero de 2023