Aemet Datalake

Desarrollo de Aplicaciones para Ciencia de Datos.

Segundo año del Grado en Ciencia e Ingeniería de Datos.

ULPGC
Escuela de Ingeniería Informática

Proyecto realizado por Gerardo León Quintana.

Fecha de realización de la memoria 12/01/2023.

Resumen

El proyecto consiste en la realización de un datalake conformado información meteorológica, dicha información la obtenemos a través de llamadas a la API de Aemet, para a partir de dicho datalake generar un datamart compuesto por las temperaturas máximas y mínimas de cada petición a la API de Aemet, con el fin ofrecer una API Rest que otorgue los registros de temperaturas máximas o mínimas en el rango de fechas determinadas.

Dicho proyecto se ha dividido en tres módulos; Feeder, DatamartProvider y TemperatureService. Los cuales procederemos a explicar su funcionamiento en dicho orden.

El objetivo principal del módulo Feeder es la creación de un datalake, y de los ficheros que lo componen, a partir de los registros meteorológicos obtenidos a través de llamadas a la API de Aemet. En este proyecto sólo se nos pide obtener los registros meteorológicos de la isla de Gran Canaria, por tanto, la obtención de los registros será alrededor de dicha condición.

El Feeder posee dos records de objetos esenciales para su funcionamiento; Weather y Area.

Weather posee tres atributos String llamados id, ubi y date, los cuales representan la id del sensor, la ubicación geográfica del sensor y la fecha y tiempo exacto del registro, además cuenta con otros tres atributos double llamados ta, tempmax y tempmin, los cuales representan la temperatura ambiente, la temperatura máxima y la temperatura mínima del registro.

Area está compuesta por cuatro atributos del tipo double llamados latMax, latMin, lonMax y lonMin, los cuales representan la latitud máxima y mínima, así como la longitud máxima y mínima.

Aparte, el Feeder está compuesto por dos interfaces; WeatherSensor y FileManager.

WeatherSensor se dedica a la obtención de los registros meteorológicos obtenidos por sensores meteorológicos, ello posee una función llamada readWeatherArea, la cual al otorgarle un objeto Area, como se ha mencionado previamente será un objeto Area con las latitudes y longitudes máximas y mínimas de Gran Canaria, nos devolverá una lista de objetos del tipo Weather. Dicho método ha sido heredado por una clase llamada AemetWeatherSensor, la cual ha sido creada específicamente para obtener los registros meteorológicos de sensores de Aemet a través de llamadas a su API.

FileManager es la encargada de la creación del datalake así como de los ficheros compuestos de los registros obtenidos de las peticiones de la API que componen al datalake, esto se lleva a cabo a través de sus métodos createDatalakeDirectory la cual se encarga de lo primero, y posteriormente createFiles, la cual requiere de que le otorguemos una lista de objetos Weather. Ambos métodos han sido heredados por la clase AemetFileManager, cuyo objetivo es crear un datalake compuesto por ficheros en los que se encuentren los registros meteorológicos obtenidos a través de la API de Aemet, clasificados por la fecha a la que pertenece dicho registro, para ello se usan los métodos mencionados anteriormente.

Finalmente, el módulo Feeder posee una clase Controller, la cual posee un método privado llamado executeTask el cual se encarga de ejecutar cada hora los procesos necesarios para el funcionamiento del Feeder, el cual se llama en la clase Main a través del método run.

Continuamos con el módulo DatamartProvider, dicho módulo se encarga de la creación un datamart que se nutre a partir de los datos recopilados en el datalake.

Para ello se ha creado una clase llamada ContentManager, la cual se encarga de recoger las temperaturas máximas y mínimas de cada fichero existente en el datalake y recopilarlas en listas de elementos Weather, a través de las funciones findMaxTemperatures y findMinTemperatures.

Además, DatamartProvider cuenta con la clase DatabaseManager, la cual se encarga de la creación de una base de datos sqlite llamada database, la cual poseerá dos tablas, maxTemperature y minTemperature, a parte posee dos métodos para rellenar dichas tablas llamados insertMaxTemperature y insertMinTemperature, los cuales al pasarles un objeto del tipo Weather lo insertaran en la tabla correspondiente.

Finalmente, DatamartProvider posee una clase controller con método privado llamado executeTask el cual se encarga de ejecutar cada hora los procesos necesarios para el funcionamiento del DatamartProvider, el cual se llama en la clase MainDamart a través del método run.

Por último, tenemos el módulo TemperatureService, el cual se encarga de inicializar la API.

Para ayudarnos a realizar dicho objetivo disponemos de un record de objetos llamado Response, compuesto por un atributo LocalDate para la fecha, tres atributos String para el tiempo, el lugar y la id de la estación, y un atributo double para el valor de la temperatura.

De seguido, se ha creado la clase AemetDatamartReader, la cual posee dos métodos getMaxTemperatures y getMinTemperatures, ambos necesitan como parámetros de entrada dos objetos LocalDate, con ello cada uno accederá a una de las tablas del database y obtendrá todos sus datos, los convertirá en objetos tipo Response y los guardará en una lista de objetos Response, para así filtrarlos en función al rango de los objetos LocalDate, pasados como parámetros, a través de Streams, para finalmente devolver un objeto String en formato de evento de JSON.

De seguido, crearemos una clase llamada WebService con la que crearemos la API, con la ayuda de la librería Spark, la cual posee un método llamado startAPI, el cuál posee dos métodos GET, uno para las temperaturas máximas y otro para las temperaturas mínimas, ambos devolverán las temperaturas máximas o mínimas en función a las fechas pasadas como parámetro de la query, es por ello por lo que inyectaremos a la clase AemetDatamartReader para que se encargue de dicho aspecto con sus métodos.

Finalmente se ha creado una clase Controller la cual se encarga de llevar a cabo todos los procesos del módulo, a través del método run, que posteriormente se ejecutará en la clase MainTemperature.

AVISO: para que el programa se ejecute de forma secuencial se deberá cambiar el working directory y el path de los archivos .jar utilizados para realizar dicha configuración de ejecución.

Índice

- Recursos utilizados: Página 6.
- Diagramas de clase:
  - Feeder: Página 6.
  - DatamartProvider: Página 7.
  - TemperatureService: Página 7.
- Líneas futuras: Página 7.
- Conclusiones: Página 8.
- Bibliografía: Página 8.

Recursos utilizados

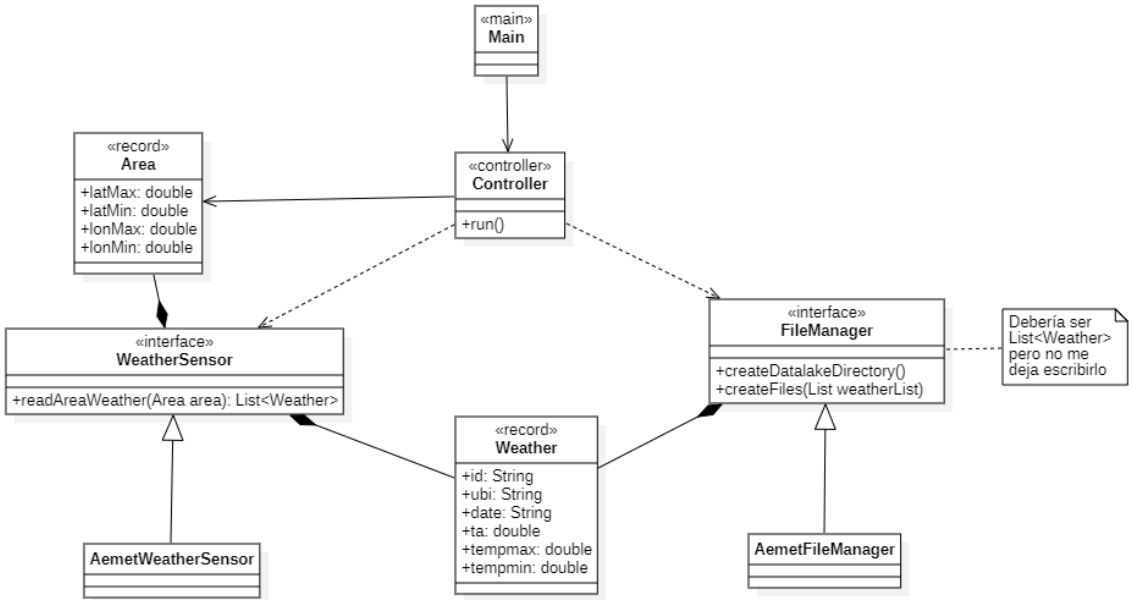
La realización integra del proyecto se ha realizado en Intellij.

Diseño

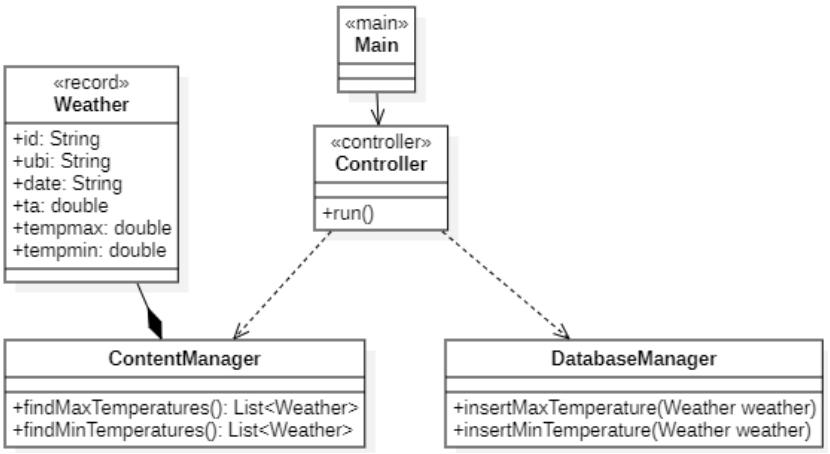
El diseño de este proyecto se considera como un MVC, dónde el modelo correspondería con el módulo Feeder, ya que es él quién quien posee toda la información y recursos del datalake, la vista correspondería con el módulo TemperatureService, debido a que nos muestra una API con la que visualizar el contenido de nuestro datamart, y el controller con el módulo DatamartProvider, ya que es el encargado de construir el datamart que posteriormente será utilizado para responder a las llamadas de la API.

Diagramas de clase

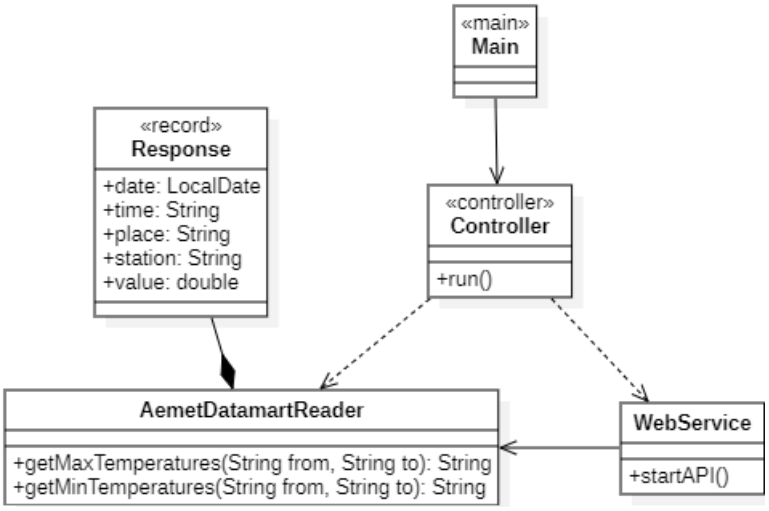
Feeder



DatamartProvider



TemperatureService



Líneas futuras

Cómo línea futura estaría bien remodelar partes del proyecto para utilizar los args del main en vez de copiar el path de del datatalke en varias clases.

Conclusiones

Este proyecto ha sido sin lugar a duda el que mayor complejidad me ha llevado, sin embargo me ha resultado satisfactorio el hecho de haber trabajado tanto en él, y me siento orgulloso de haber conseguido todo hasta lo que he llegado, pese a que claramente, el proyecto como los diagramas de clase, en sí podrían ser mucho mejor si fuese más experimentado.

Bibliografía

- https://sparkjava.com/
- https://www.geeksforgeeks.org/
- https://stackoverflow.com/
- https://www.sqlitetutorial.net/sqlite-java/