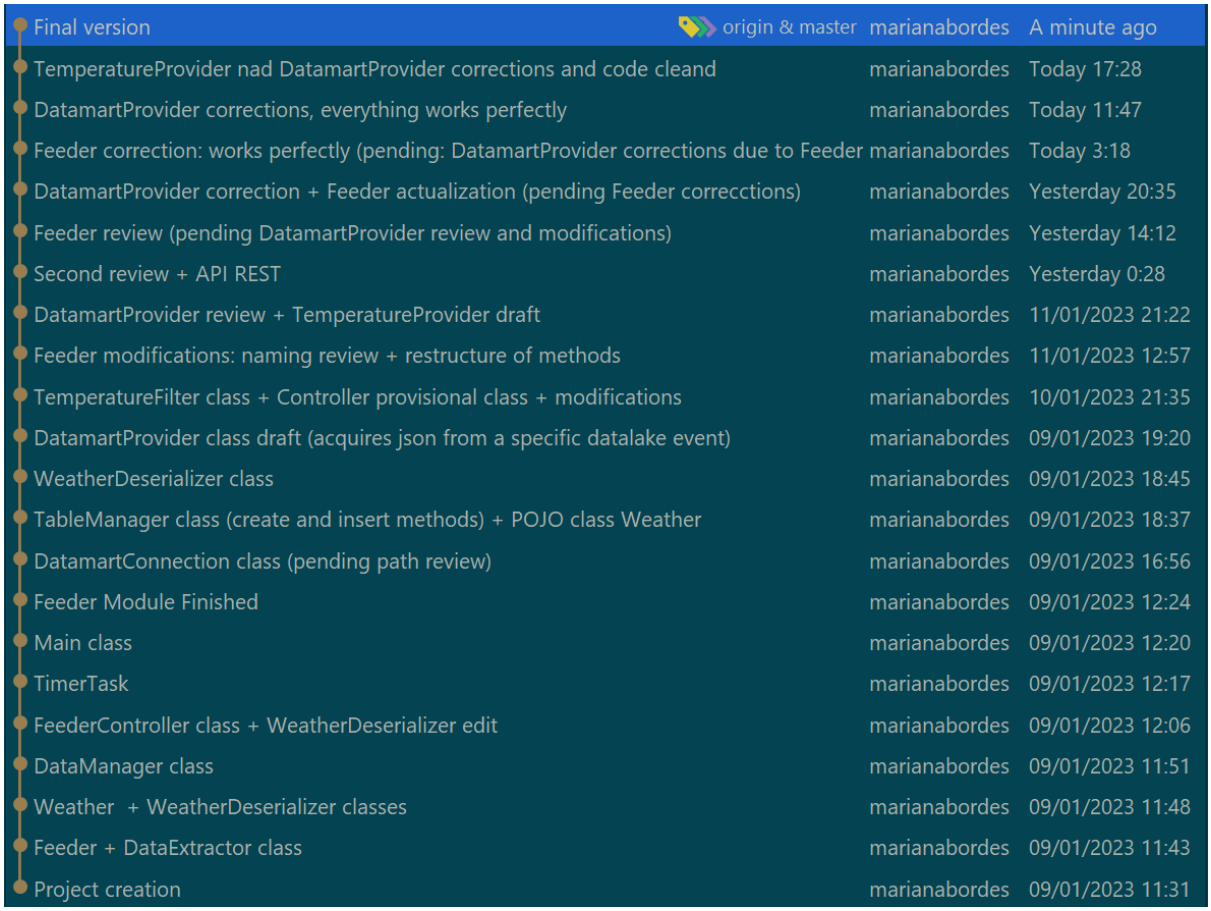
AEMET

PROYECTO FINAL

Mariana Bordes Bueno 2ºGCID DACD EII ULPGC

Mariana Bordes Bueno 2ºGCID DACD EII ULPGC



Índice

Resumen 1

Recursos utilizados 2

Diseño 2

Conclusiones 2

Líneas de regresión futuras 3

Bibliografía: 4

Resumen

El resumen de este proyecto se divide en los tres módulos que corresponden a los módulos del proyecto.

El módulo Feeder se encarga de almacenar y actualizar cada hora los datos adquiridos de AEMET en un datalake dividido en ficheros según el día en que se tomaron los datos.

El Feeder trabaja de manera que se conecta a la API de AEMET a través de la dirección url proporcionada resultando del método getAemetWeathers() una lista de objetos tipo Weather (que es el tipo de objeto con el que trabajaremos a lo largo del proyecto) que contiene datos de todas las estaciones a nivel nacional.

Los datos requeridos para esta aplicación son sólo los que provengan de estaciones de Gran Canaria; con la clase WeathersGcTodaySelection conseguimos filtrar los datos provengan de estaciones de Gran Canaria según su latitud y su longitud convirtiendo todos los eventos proporcionados en un stream para que facilite la tarea obteniendo así los datos de las últimas 24h de todas las estaciones de Gran Canaria, pero como la aplicación está diseñada para ejecutarse constantemente, descartamos los del día anterior porque ya estarían guardados en su respectivo fichero; por lo tanto, el método de la clase getWeathersFromGcToday() va a devolvernos una nueva lista que contiene sólo los datos del mismo día, excluyendo los que sean del día anterior para mejorar el rendimiento de la aplicación.

La nueva lista ya contiene los datos requeridos por la aplicación y sólo quedaría guardarlos con la clase AemetWeathersStorer que funciona de la siguiente manera: Para comenzar accede a un fichero del directorio datalake (en caso de no existir, lo crea) y adquiere todo su contenido en formato Weather almacenándolo en una colección (Set) que utilizamos para saber si los elementos de la lista que contiene los datos ya han sido introducidos en el fichero para evitar almacenar el mismo dato dos veces y que si la aplicación dejara de utilizarse durante unas horas, al volver a ejecutarse añada únicamente los datos que no han sido guardados. De esta manera tendremos una carpeta con un fichero para cada día que se ejecute la aplicación que almacena los datos meteorológicos de la AEMET para las 24 horas del día cuyo nombre será la fecha.

El módulo DatamartProvider funciona de manera que crea una conexión a una base de datos SQLite donde se van a almacenar, en dos tablas distintas, el valor máximo y mínimo de cada día.

La clase DatamartProvider se encarga de adquirir los datos del datalake para administrarlos en el datamart. Para esto cuenta con un método que identifica el fichero del día actual que recorre línea a línea guardando la información, ya deserializada, en una ArrayList de Weathers que almacena todos los datos de todas las horas del día. Posteriormente, esta ArrayList pasa por unos filtros de temperatura (también utilizando streams) que devuelven el mayor y menor valor de las temperaturas del ArrayList que son los que se insertan en las tablas.

Por último, el módulo TemperatureProvider ofrece una API REST para realizar consultas al datamart SQLite del módulo anterior. Cuenta con cuatro métodos, dos para realizar consultas a la tabla completa y otros dos para consultar los valores en un rango de días determinado.

Recursos utilizados

El entorno de desarrollo utilizado ha sido Intelllj. Me he sentido muy cómoda con él principalmente porque facilita mucho el trabajo que el entorno resalte todos los errores automáticamente y que el Alt + Intro te sugiera cómo solucionarlo. Además, gracias a los shortcuts he podido programar de una manera bastante rápida y eficiente.

Con respecto a las herramientas de control de versiones, he utilizado Git. Me ha parecido muy cómodo a la hora de trabajar y de hacer y deshacer cambios saber que puedo ir guardando las versiones que me han ido sirviendo y volver a ellas en cualquier momento.

Como herramienta de documentación he utilizado la documentación oficial de java (docs.oracle.com) y numerosas consultas a webs como stackoverflow.

He utilizado varias dependencias de maven, entre ellas: gson, spark-core, sqlite-jdbc y jsoup.

La aplicación está desarrollada con la versión 19 de Java.

Diseño

El sistema de la aplicación sigue la arquitectura lambda, ya que maneja datos la Batch Layer (por ejemplo cuando accede al datalake para administrarle la información al datamart) y también trabaja con datos en realtime(almacenándolos en el datalake o en la API REST).

Con respecto a los principios de diseño, la aplicación sigue uno de los principios SOLID que hemos estado trabajando a lo largo del curso que es el de Single Responsibility; cada clase tiene una única responsabilidad, por ejemplo, si la clase AemetWeatherExtractor se encarga de extraer los datos de AEMET, sólo cuenta con los métodos necesarios para la extracción de datos.

Conclusiones

A diferencia con el otro proyecto, para este me preparé mejor antes de empezar a programar, me hice esquemas y organicé las ideas principales, identifiqué los requisitos, etc.… Aunque es verdad que he “perdido” mucho tiempo haciendo y deshaciendo cosas y buscando porque todavía no domino el lenguaje y le he dedicado mucho tiempo a un proyecto que podría haberse hecho mucho más rápido.

Otra vez comencé el proyecto sin utilizar el Git como herramienta de control de versiones, así que cuando leí las correcciones del proyecto anterior creé un nuevo proyecto en el que fui copiando lo que tenía y realizando los commits que consideraba que debería haber hecho y a partir de ahí fui utilizándolo a medida que iba programando (aunque a veces se me olvidaba realizar el commit cuando me tocaba), poco a poco es algo que voy teniendo más presente cuando estoy programando.

También sobre el Git, me quedo con el pensamiento de que debería haber utilizado branchs y tags que me podrían haber facilitado el trabajo en ciertos puntos y es algo que tendré en cuenta a la hora de realizar el siguiente proyecto.

Otra cosa que considero que debería haber utilizado son interfaces, porque lo he creado todo en clases y, por ejemplo, algún método que he querido utilizar en dos clases a la vez podría haber sido posible con una interfaz. Es algo que me estoy planteando ahora que ya es tarde para este proyecto, pero lo tendré en cuenta para el próximo.

En el próximo proyecto también leeré las memorias de proyectos anteriores y las correcciones antes de empezar para tener todas estas cosas en cuenta.

Del proyecto me quedo con todo lo que he aprendido tanto de java específicamente (como por ejemplo: crear una API, los conjuntos como contenedores de información, los ndjson que utilizo en el Feeder…) como de programación en general, ya que sobre todo considero que he aprendido a buscar la información que necesito de forma más óptima, a entender mejor las excepciones y los errores, a identificar los fallos mucho más rápidamente, a pensar mejor los nombres de las variables, las clases y los métodos y que poco a poco voy cogiendo más práctica; incluso he aprendido cosas que considero que me van a ayudar mucho en el futuro como que programar con nervios o frustración no es buena idea y que cuando estoy bloqueada con algo debería respirar hondo y hacer un descanso porque cuando examinas el código con tranquilidad y paciencia ves muchos fallos (como por ejemplo, una coma fuera de lugar) y piensas en cosas que con la mente bloqueada no se te ocurren.

Líneas de regresión futuras

Considero que esta aplicación, con algunas modificaciones, podría ser utilizada para promover los deportes acuáticos en las islas, ya que podría adaptarse al teléfono móvil y que puedan ver las condiciones climatológicas que afecten a su deporte para saber qué zona de la isla tiene las mejores condiciones por ejemplo para surfear o para navegar, incluso se podría crear un apartado similar al Tinder donde conocer a gente interesada en el mismo deporte que tú o incluso gente profesional que quiera dedicarse a enseñar a principiantes. También sería interesante la posibilidad de ver cuánta gente hay en cada playa ya que habrá personas que prefieran ir a sitios con mucha gente o personas que prefieran estar solos, así sabrían que en qué playa disfrutarían más. También habría un apartado de predicciones para los próximos días e incluso se podría hacer un apartado que en base a las condiciones climatológicas te recomiende un deporte acuático; por ejemplo si las olas están fuertes y hace mucho viento que recomiende el windsurf o si el mar está calmado que recomiende más bien pádel surf o notifique a los más principiantes para que sepan que es un buen día para aprender y por el contrario si las condiciones son más extremas pues avisar por ejemplo a los surfistas más avanzados.

Un poco en la línea de la aplicación anterior sería una aplicación que recomendara planes en base a las condiciones meteorológicas (no sólo acuáticas); por ejemplo, al iniciar sesión el usuario especifica que le gusta hacer senderismo y que prefiere hacerlo cuando el día está nublado y la

sensación térmica está entre los 18 y los 22 Cº, pues la aplicación podría avisarle (por ejemplo el jueves) de que el sábado va a ser un buen día para escalar.

Bibliografía:

- oracle help center
- SQLite tutorial
- stackoverflow
- open bootcamp