Hemos seguido una estructura de paquetes dividiendo el proyecto por las funciones de las clases: model, para las POJO; service, para utilidades puntuales, como la generación de gráficas, mapeo entre ciudad y su respectivo código, mapeo entre CSV y XML... ; io, en relación a la lectura y escritura de ficheros ; y controller, con un método de inicio que llama a las demás funciones para obtener el resultado deseado

En esta ocasión hemos utilizado la Api Stream a lo largo de toda la aplicación, siendo especialmente relevante su presencia a la hora de procesar los datos de medias. Hemos utilizado DoubleSummaryStatics, ya que nos interesaban la media, el mínimo y el máximo de distintos conjuntos de datos, haciéndola ideal para esta función.

También hemos utilizado diferentes parsers a lo largo del problema. El que más peso tiene en el proyecto es JDOM, ya que nos ha servido tanto para la lectura como para la escritura en los primeros pasos, sirviendo también como práctica de DOM y SAX. Posteriormente, para generar la base de datos XML con los datos procesados, hemos decidido utilizar JAXB, a parte de para practicar en el manejo de esta librería, también para facilitar la generación del XSD posteriormente. Para su lectura, que sería utilizada a la hora de obtener los datos del fichero markdown, usamos Xpath, evitando así leer contenido innecesario o que no nos era de interés para ese informe.

Adicionalmente se ha utilizado la librería JFreeChart para generar los informes mostrados en el html. También se usa lombok en las diferentes clases para dejar un código más reducido sin getters y setters.

Clases interesantes:

HtmlWriter, que genera un archivo html ampliable, gracias a la función APPEND de Files.write(). Este tiene una estructura sencilla, pero refleja los datos más relevantes, y expone algunas gráficas sobre ellos.

MardownWriter, que navega a través de la base de datos XML generada con Xpath para obtener los datos de interés y almacenarlos en una lista junto a marcas de este lenguaje para finalmente escribirlos en un fichero .md

Utiles contiene un método de normalización, para evitar errores, sobre todo con ciudades como Rivas-Vaciamadrid, donde se puede omitir el guion, poner junto, en mayus, minus… Para que el programa aceptase todas esas variaciones, se optó por esta normalización, combinada con los mapas de la clase MapeoCiudadCodigo, permiten pasar de ciudad a código y a “modo estándar”, con su primera letra en mayúscula, acentos, guiones...

CLASE CONTROLLER

Es la primera que se encuentra en el main. En ella hay llamadas a las clases Service tanto de CSV como de XML. Tambien delega al Generador de gráficas, al Generador de HTML y al Generador de la base de datos xml.

CLASE CSVSERVICE

Es una clase singleton.

Se usa JDOM.

Es la encargada de convertir el archivo CSV en XML.

El método createLector genera un XML vacío con un elemento root especificado.

El método convertCSVtoXML filtra por ciudad y convierte a XML sólo los registros de esa ciudad, siendo mucho más útil y práctico que otras formas.

El método preProcess es el encargado de darle forma y formato al xml para escribirlo.

El método writeXMLFile es el encargado de pintar el XML en la ruta especificada.

CLASE XMLSERVICE

La encargada de mapear los XML generados en objetos.

CLASE CONTAMINACIONSERVICE y METEOSERVICE

En ella hay dos métodos :

- filtrarMagnitudesContaminacion() o filtrarMagnitudesTemperatura ---> contiene un switch que por una lista de tipo Medicion, filtra por magnitud e inserta esa Medicion en una lista de Mediciones. Obteniendo así todas las listas de Magnitudes, en las clases MedicionesMeteo y MedicionesContaminacion.

Son clases formadas unicamente por listas de magnitudes.

-getEstatisticsMeteo() o getEstatisticsContaminacion ---> Devuelve una lista con todas las estadísticas generadas sobre las listas anteriormente mencionadas, en las clases MedicionesMeteo y MedicionesContaminacion.

Estas estadísticas se generan llamando a la clase estática "generarEstadisticas"

CLASE INFORMESERVICE

El método generarXMLbbdd con los datos ya filtrados, genera una base de datos llamada mediciones.xml como se pide en la práctica.

Tiene un método llamado generarEstadisticas, es un método genérico pensado para su uso tanto en mediciones de meteorologia como de contaminación. Recibe un parámetro: una lista de Mediciones. Y sobre ella con la API STREAM y la clase DoubleSummaryStatistics se generan

las medias, maximas y minimas de cada lista. Si no hubiese registros de una magnitud el programa no devolverá NullPointer porque se ha filtrado para que devuelva un mensaje.

Proyecto de Mario González Gómez y Andrea Gómez de Pablo