Desarrollo de Aplicaciones para Ciencia de Datos

2º Grado en Ciencia e Ingeniería de Datos

Universidad de las Palmas de Gran Canaria, EII

BOOKING SCRAPPER CON API RESTFUL

Ricardo Juan Cárdenes Pérez

RICARDO JUAN CÁRDENES PÉREZ

12/01/2023

ÍNDICE

RESUMEN

En esta memoria se explica brevemente el funcionamiento, diseño y arquitectura de la aplicación JAVA BookingScrapper presentada, la cual consiste en una aplicación que nos ofrece los datos disponibles en Booking de un hotel en concreto mediante una API RESTFul.

RECURSOS UTILIZADOS

La plataforma escogida para llevar a cabo el desarrollo de este proyecto a sido IntelliJ por todas las facilidades que nos brinda a la hora de programar, además de permitirnos trabajar con el control de artefactos Maven. Este proyecto logra su correcto funcionamiento gracias a tres dependencias importadas, entre las cuales se encuentran las siguientes:

- JDBC-SQLite: esta dependencia hace posible la conexión con la base de datos principal del programa.
- Spark: nos permite crear una API web en nuestro servidor local LocalHost.
- Gson: nos permite obtener la serialización en formato JSON de cualquier objeto que queramos.

Este proyecto utiliza Git como control de versiones, y cuenta con un repositorio público en GitHub al cual puede accederse mediante el siguiente enlace

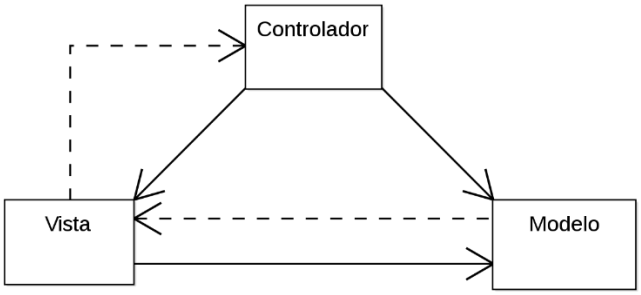
https://github.com/ricardocardn/BookingScrapper

DISEÑO

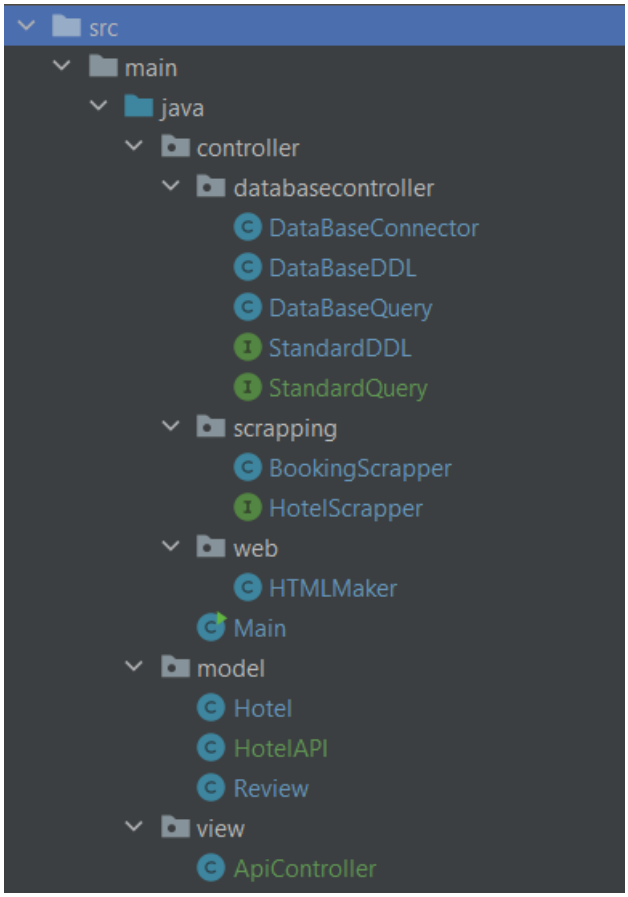
Para este proyecto habíamos pensado en optar por el uso de un estilo arquitectónico que divide la lógica de la aplicación en dos paquetes: Model y Controller.

- Model: este es el componente central de nuestro arquitectura. Contiene la representación abstracta de la información con el que el sistema opera.
- Controller: dedicado a responder a eventos, ya sean internos o provocados por el usuario a partir de la interacción con la API RESTFul, e invoca peticiones al modelo cuando se solicita alguna información.

En esta arquitectura, el paquete de control depende del modelo pero jamás el modelo del control. No obstante, esta arquitectura podría adaptarse para que sea compatible con el estilo MVC, el cual es utilizado en la gran mayoría de proyectos software de esta índole a día de hoy. Para ello, podemos pensar la API en sí misma como la vista de nuestra aplicación, ya que es lo que utiliza el usuario tanto para obtener información de la misma, como para modificarla, aunque solo sea añadiendo información a nuestra base de datos. De esta forma, podríamos visualizar de la siguiente forma nuestra arquitectura:

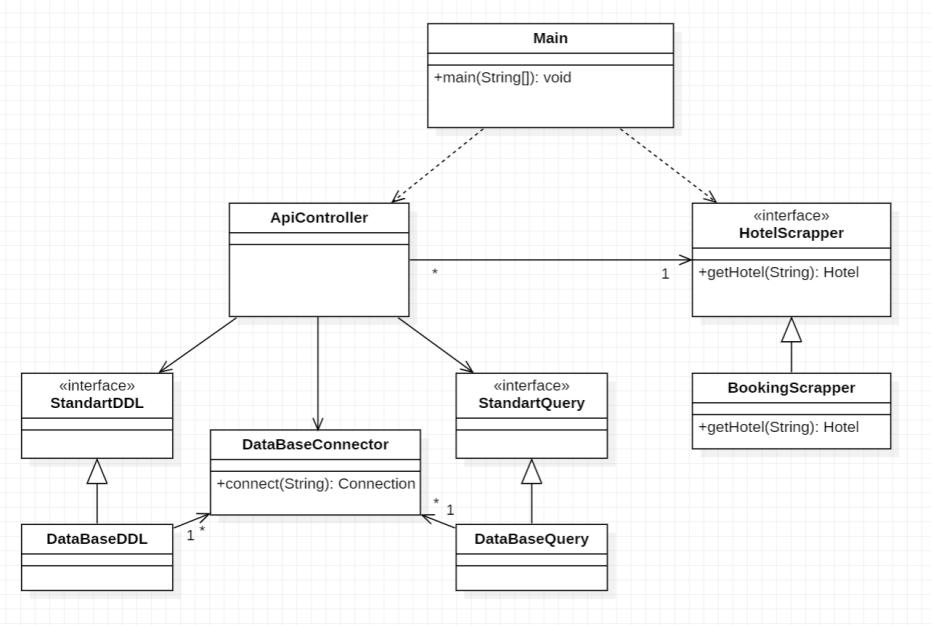


Esto que acabamos de comentar podemos visualizarlo simplemente con abrir nuestro proyecto en IntelliJ donde, en el apartado de archivos del proyecto, podemos apreciar la siguiente estructura



Principios de Diseño

Si nos paramos a analizar el código implementado, podemos ver que este podría representarse mediante el siguiente diagrama de clases si seguimos el estándar UML:

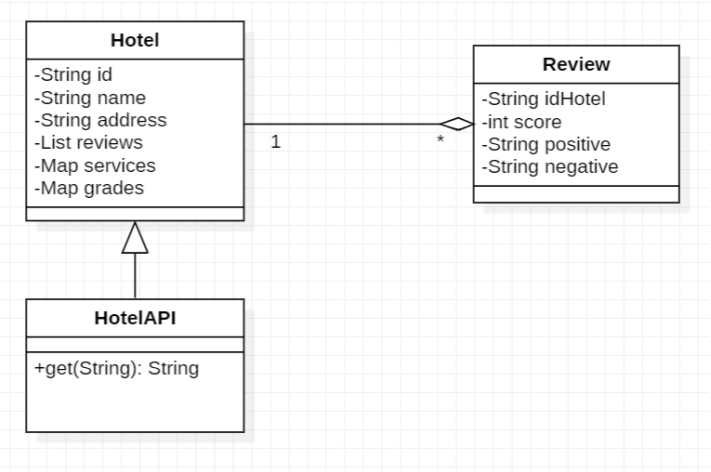


Bastaría con observar el nombre de las clases para ver que cumplimos correctamente el principio SOLID de responsabilidad única, pues existe una clase para cada función que realiza nuestra aplicación. Por ejemplo, existen tres clases para gestionar la base de datos, de las cuales una se encarga de crear la conexión, otra de crear e insertar datos en las tablas y otra de realizar consultas a las mismas.

Se cumple además el principio de sustitución de Liskov, pues utilizamos interfaces para todas aquellas partes del programa que puedan ser modificadas o incluso cambiadas por otras en la vida útil del mismo. De esta forma, si cambiamos de sistema de base de datos nos bastará con crear dos nuevas clases, una para la inserción y creación de datos y otra para la consulta de ellos, las cuales implementen las interfaces *StandardDDL* y *StandardQuery*, respectivamente.

También podemos apreciar cómo se satisface el principio de inversión de dependencia, pues clases abstractas no dependen de clases de bajo nivel. En el caso de la clase ApiController, esta depende únicamente de interfaces, a excepción de la clase DataBaseConnector que suponemos que no cambiara a lo largo del proceso de vida del producto.

En cuanto al modelo de la aplicación, podemos visualizarlo mediante el siguiente diagrama de clases:



La clase HotelAPI es una clase que extiende de la clase Hotel y que únicamente nos aporta un método adicional a la clase hotel, el cual nos facilita bastante el código en la clase ApiController a parte de estilizarlo. Pese a crear objetos de este tipo de clase en el programa, al instanciarlos nos referimos a ellos como objetos de tipo Hotel, de la siguiente forma:

```
Hotel hotel = new HotelAPI();
```

Y es el polimorfismo de clases el que no permite llamar al método get implementado en la clase HotelApi de ese objeto. La razón principal por la que se ha hecho esto es por no modificar clases libre de bugs que ya estaban en nuestro programa desde las primeras versiones (Open-Closed Principle).

Conclusiones

Creo que este proyecto es bastante útil para nosotros a nivel académico. La información y los datos es algo muy importante en una carrera como la nuestra. Uno de los principales problemas en el campo en el que nos estamos moviendo, la Ciencia de Datos, es la escasez de los mismo. Ya sea para entrenar una inteligencia artificial o para elaborar un buen estudio estadístico necesitamos, no solo de una cantidad masiva de datos, sino de una gran cantidad de datos correctamente estructurados en el formato que nos interesa. Esto es lo que estamos consiguiendo con este trabajo. Considero que es un trabajo que nos podrían pedir como Ingenieros de Datos en cualquier empresa en la que estemos trabajando y creo que haberlo hecho apenas empezando la carrera es algo bastante valioso.

Líneas Futuras

Como líneas futuras, me gustaría publicar este servicio web en internet, de forma que cualquier usuario pueda acceder a él desde cualquier parte, a parte de otros los cuales explico en las líneas futuras del trabajo final de Aemet.

Bibliografía

- https://docs.oracle.com/javase/7/docs/api/
- https://opendata.aemet.es/dist/index.html
- https://sparkjava.com/documentation
- https://www.sqlitetutorial.net/sqlite-java/
- https://mvnrepository.com/repos/central

Ricardo Juan Cárdenes Pérez

Universidad de las Palmas de Gran Canaria