Proyecto Scraper Booking

Desarrollo de Aplicaciones para Ciencia de Datos (DACD)

Segundo curso

Grado en Ciencia e Ingeniería de Datos

Escuela de Ingeniería Informática (EII)

Universidad de Las Palmas de Gran Canaria (ULPGC)

Índice

1. Resumen:

Para el proyecto he decidido crear un total de seis clases, de las cuales una es de apoyo para otras funciones, y dos interfaces. Este programa debe encargarse de buscar datos en el sitio web de Booking (scraing) y ofrecerlos a los usuarios mediante una api en modo local. Para la realización del proyecto se han usado las librerías externas gson(objetos a json), jsoup(scraping) y spark (web service).

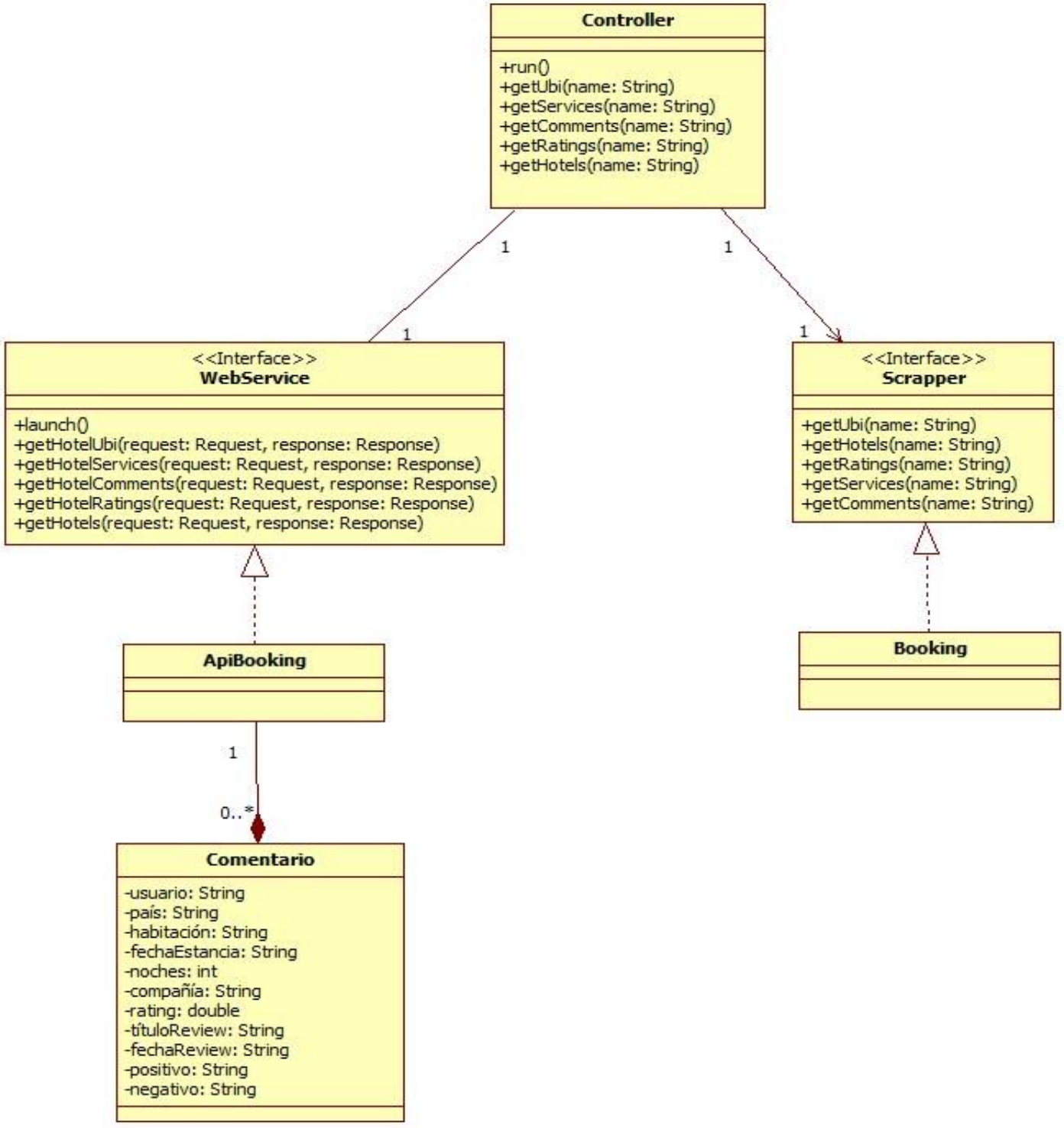
Al ejecutar el programa este lanzará una serie de direcciones que podemos usar para la consulta de datos de Booking. En este caso son:

- /hotels/:name Devuelve la dirección y coordenadas de un hotel.
- /hotels/:name/services Devuelve los servicios disponibles en un hotel.
- /hotels/:name/comments Devuelve las reseñas de los usuarios sobre un hotel.
- /hotels/:name/ratings Devuelve las puntuaciones por categoría de un hotel.
- /:name/hotels Devuelve un listado de los principales hoteles en una ciudad.

Para consultar cualquiera de las direcciones anteriores debemos usar **http://localhost:**4567, para indicar que los recursos se encuentran en local. En las primeras direcciones el parámetro name se corresponde con el nombre de un hotel, mientras, en la última dirección se corresponde con el nombre de una ciudad. En ambos casos el parámetro de contener el nombre en minúsculas y separado por guiones en vez de espacios. Todas las consultas nos devolverán una respuesta en formato json.

2. Diagrama de clases:

A continuación, encontramos el diagrama de clase del proyecto:



3. Recursos usados:

- **IDE:** IntelliJ IDEA Community Edition 2022.2.2.
- **Control de versiones:** GIT (integrado en IntelliJ)**.**
- **Herramienta diagrama de clases:** StarUml.
- **Herramienta de documentación:** Word.
- **Versión java:** 11
- **Librerías externas:** jsoup(1.15.3), gson(2.10) y spark(2.9.4).

4. Diseño:

Para el desarrollo del proyecto he optado por la opción de realizar el scraping en el momento en el que se realiza una petición desde la api. Desde mi punto de vista mantener datos almacenados para mostrarlos en la api podría ahorrarnos tiempo, pero gastaríamos almacenamiento y sería imposible contener todas las respuestas a las peticiones de los usuarios. Por eso creo que es más eficaz realizar el scraping tras obtener la petición:

- En primer lugar, necesitamos una interfaz para el Scraper, así cualquiera de los scrapers que implementen la interfaz tendrán los mismos métodos. Los métodos de esta interfaz nos permitirán obtener una lista de hoteles de una ciudad, y los detalles de un hotel concreto (ubicación, coordenadas, ratings, servicios y comentarios). Todos los métodos

- Para mostrar lo que obtengamos de la implementación del Scraper necesitaremos un web service. Para este WebService crearemos una interfaz con métodos comunes para sus implementaciones. En este caso tendremos un método launch para lanzar la api en local y habilitar las urls de búsqueda, así como métodos para realizar la petición de datos (getHotelUbi(), getHotelServices(), getHotelComments(), getHotelRatings(), getHotels()). Todos los métodos para obtener y mostrar los datos requerirán de un Request y un Response como parámetros.

- La clase Booking implementa la interfaz de Scraper y adapta los métodos a los recursos de Booking Un objeto de Booking requiere de una String que ejercerá como nombre del hotel o de ciudad como parámetro de cualquiera de sus métodos. Todos los métodos usan la librería jsoup para el scraping. El método getUbi devuelve un mapa con la dirección y coordenadas de un hotel, mientras que getServices() devuelve un mapa que tiene como clave el nombre del servicio y como valor de la lista de servicios. Esta lista de servicios se obtiene tras realizar un Split por mayúsculas y espacio sobre una string de tamaño mayor con todos los

servicios juntos. El método getRatings() devuelve un mapa con las valoraciones por categorías y la general. Para devolver las valoraciones las convertimos a double para lo que es necesario sustituir la coma decimal por un punto. Para el método getComments() usamos un objeto comentarios para facilitar la conversión, este método devuelve una lista de comentarios. Por último el método getHotels() devuelve una lista de nombres de hoteles en una ciudad. Todos los métodos devuelven un json mediante la librería gson.

- La clase Comentario mencionada anteriormente es una clase POJO (facilita serialización), que contiene los detalles de cada comentario. En el diagrama de clases podemos observar sus elementos.

- La clase ApiBooking es la implementación de la interfaz WebService para Booking. A Booking le pasamos un controlador por parámetro para que no interactúe directamente con la clase Booking. El método launch lanza las urls. En este caso cada método get designa responsabilidad a la función correspondiente para cada petición que se pueda realizar en la api y que recurrirá al controlador para obtener la información del scraper. Todas las urls necesitan de una parámetro de request llamado name(indica hotel o ciudad).

- La clase Controller se encarga de ser el intermediario entre el Scraper y el WebService. Recibe como parámetro un WebService que en nuestro caso corresponderá con un objeto de ApiBooking. El método run recurre al método launch de la api para iniciarla. Tras eso contiene métodos con el mismo nombre que los del Scraper, que recurren al método requerido por la petición realizada por el usuario en la api.

El principal principio de diseño arquitectónico es el MVC(Model-View-Controller). En el diagrama de clase podemos observar como el controlador se corresponde con un objeto de la clase ApiBooking que implementa la interfaz WebService. Lo mismo sucede con la clase Booking y la interfaz Scraper. La única forma que se me ocurrió para que el WebService pudiese acceder a los métodos del Scraper indirectamente era que el WebService invocará a un controlador y este usará los métodos del Scraper. Este modelo permite separar la vista de la lógica de los datos.

En cuanto a los principios SOLID en el párrafo anterior podemos detectar el principio de inversión de dependencia donde un objeto de la clase Controller contiene un Scraper y un WebService en lugar de usar Booking o ApiBooking dependiendo así de abstracciones. El principio de única responsabilidad se cumple al encargarse cada función de un método de algo particular (funciones para cada petición de datos y scraper). La ley de Demeter se cumple al crear la clase Controller gracias a la cual el WebService no tiene que conocer la estructura del Scraper y viceversa.

5. Conclusiones:

El proyecto me ha permitido trabajar con web services y conocer más a fondo como se organiza la información en páginas web. Tal vez no estoy del todo satisfecho con realizar un scraping cada vez que se hace una petición, ya que se podrían guardar resultados anteriores para aprovecharlos. Sin embargo, me parece la opción más asequible de implementar sin gastar recursos de memoria.

Por otro lado, no estoy del todo convencido con que los objetos de la clase ApiBooking cree controladores para invocar funciones de la clase Booking. Por la misma razón, no me convence tener métodos similares en el Controller a los de Booking.

Por último, cabe destacar la importancia de fijarse en los archivos html de las páginas web. Durante el desarrollo del proyecto me quedaba estancado en ciertas fases por no elegir bien las secciones donde buscaba. Gracias a esto me di cuenta de que debía buscar en las secciones más pequeñas posibles y no tan superficiales.

6. Líneas futuras:

El proyecto cumple con unas funcionalidades básicas, aunque, se pueden realizar mejoras que permitirían ampliarlas. Una de las restricciones principales del programa es que la api solo se ejecuta en local, sería conveniente tener un link fijo en la web como un servicio en la nube.

Respecto a la capacidad de consulta para cada apartado, se podría aumentar el número de comentarios a obtener, al igual que el número de hoteles de cada ciudad. Sería útil algún query param en la búsqueda de hoteles que permita ordenarlos según puntuación, estrellas u otras calificaciones.

7. Bibliografía:

- Web booking: https://www.booking.com/hotel/es/santa-catalina.es.html
- Web reviews booking: https://www.booking.com/reviewlist.es.html?cc1=es;dist=1;pagename=santa-catalina;type=total&&offset=0;rows=10

- Web booking hoteles ciudad:
  https://www.booking.com/city/es/las-palmas-de-gran-canaria.es.html