Memoria práctica ISD

Grupo isd069

Misa Gómez, Miguel (miguel.misa@udc.es)

Mouro González, Samuel (s.mourog@udc.es)

París Rojo, Javier (javier.parisr@udc.es)

Apéndice

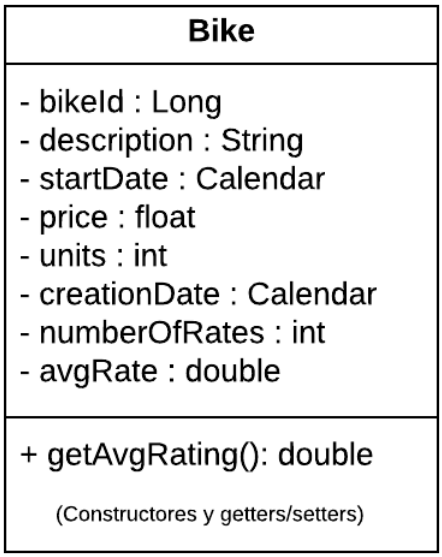
1. Introducción

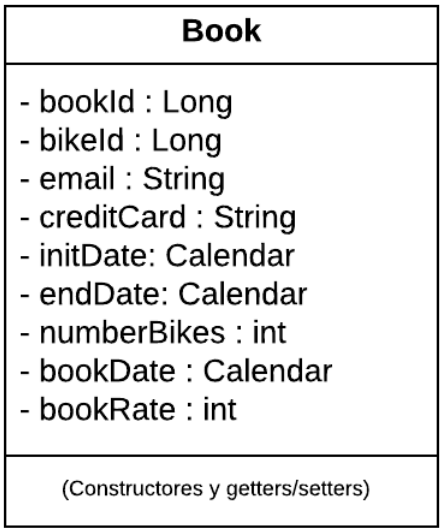
Esta memoria trata de explicar el diseño de la aplicación partiendo de las diferentes capas que esta tiene: modelo, servicio y cliente. Para ello se incorporan los diagramas UML que detallan el diseño de cada capa.

En este caso, se ha decidido no hacer los trabajos tutelados propuestos para el desarrollo de la aplicación.

2. Capa modelo

2.1. Entidades

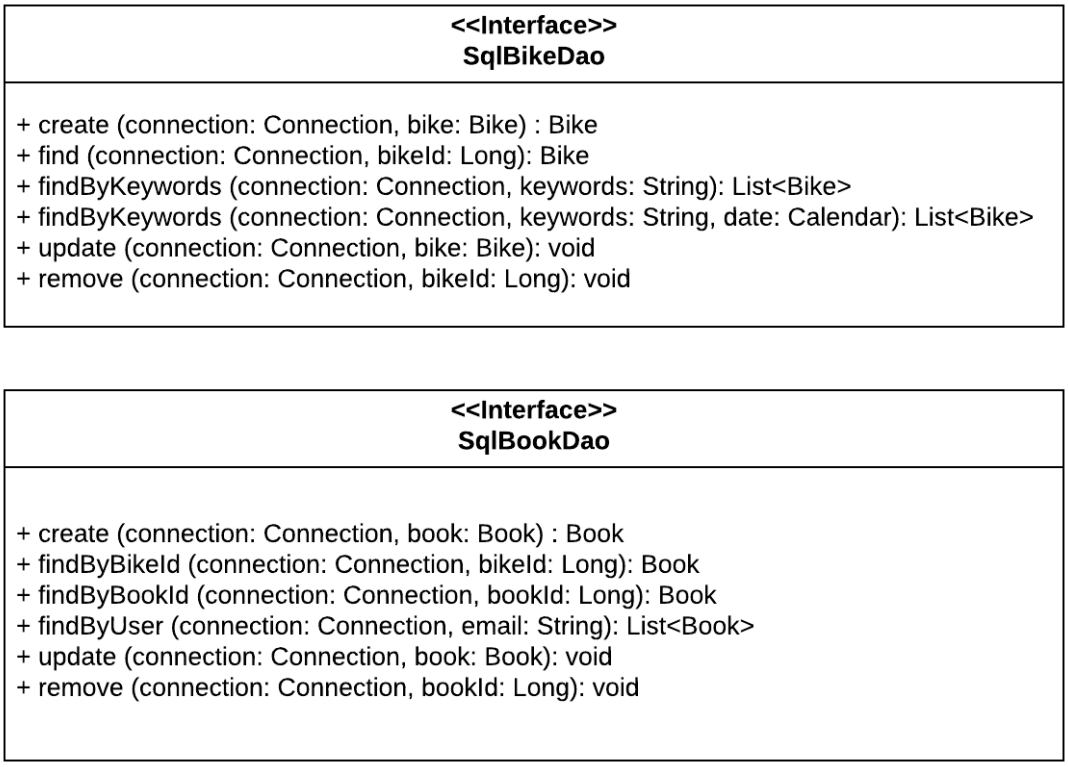




Para el diseño de las entidades se puede destacar que se incluye un método *getAvgRating()* para calcular la media de las puntuaciones de una bicicleta, ya que *avgRate* funciona como contador que suma todas las puntuaciones que se hacen sobre una bicicleta. De este modo no hay que recalcular la media cada vez que hay una nueva puntuación, solo cuando es necesario.

Además, se añade un atributo *bookRate* a la entidad *Book* para poder almacenar la puntuación de la reserva. Se desnormaliza en la base de datos la media, y se tienen atributos para este fin en las entidades *Book* y *Bike,* porque es menos costoso tener calculado en *Bike* las medias, que tener que acceder a todas las reservas en la base de datos cada vez que se quiera calcular una media.

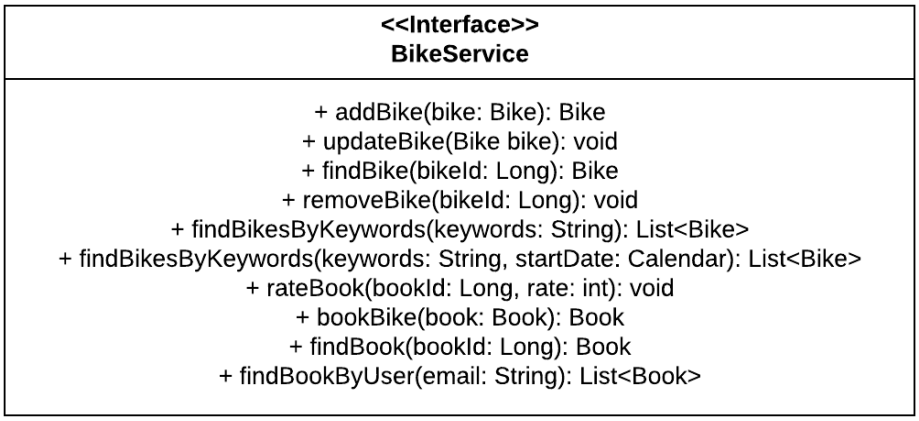
2.2. DAOs



Cabe destacar que para el diseño del DAO de la entidad *Bike* (*SqlBikeDao*) se ha decidido usar la sobrecarga en el método *findByKeywords* para los casos en los que se decide incluir la fecha además de las palabras clave.

En el caso del DAO de la entidad *Book* (*SqlBookDao*) se ha decidido separar el método *find* en tres métodos diferentes: *findByBikeId, findByBookId* y *findByUser*. En el caso de los IDs, se separa dado que usar 2 métodos diferentes clarifica qué hace cada uno. Ya que, aunque el ID es de tipo *Long* en ambos casos, hacer una sobrecarga podría resultar confuso. Y en el caso de *findByUser*, se requiere dados los casos de uso necesarios para la correcta implementación de la aplicación.

2.3. Fachadas



En el caso de la interfaz del bikeService cabe resaltar que, al igual que con el DAO de *Bike* (*SqlBikeDao*), se utiliza la sobrecarga en el método *finfBikesByKeywords* según si se pasa una fecha a partir de la cual buscar bicicletas o no.

3. Capa servicios

3.1. DTOs

Incluye un diagrama UML que muestra el diseño de los DTOs que utiliza el servicio para recibir y devolver datos. Para cada DTO sólo se deben mostrar sus atributos privados (no se mostrarán los métodos get/set). En el texto del apartado, se deben comentar/justificar los detalles que se consideren relevantes.

3.2. REST

Especifica cómo invocar cada caso de uso que ofrece el servicio y cuáles son sus posibles respuestas, indicando para cada uno:

• URL simplificada del recurso (e.g. /product/{id}, /products)

• Método de invocación (GET, POST, PUT o DELETE)

• Parámetros o DTO de entrada (si aplicable)

• Posibles códigos HTTP de respuesta

• DTO devuelto (si aplicable)

En el texto del apartado, se deben comentar/justificar los detalles que se consideren relevantes. En particular, debe explicarse el criterio seguido para escoger los códigos HTTP de respuesta asociados a cada excepción.

4. Aplicaciones cliente

4.1. DTOs

Incluye un diagrama UML que muestra el diseño de los DTOs que se utilizan en la capa acceso al servicio de las aplicaciones cliente. Para cada DTO sólo se deben mostrar sus atributos privados (no se mostrarán los métodos get/set). En el texto del apartado, se comentan/justifican los detalles que se consideren relevantes.

4.2. Capa acceso al servicio

Para cada aplicación cliente incluye un diagrama UML que muestra únicamente las interfaces de la capa acceso al servicio que utiliza esa aplicación. Cada interfaz debe especificar la firma completa de sus operaciones. No es necesario especificar las excepciones, ni incluir los diagramas UML de las mismas. En el texto del apartado, se deben comentar/justificar los detalles que se consideren relevantes.

5. Errores conocidos

Se indican los errores que se conoce que tiene el código.