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

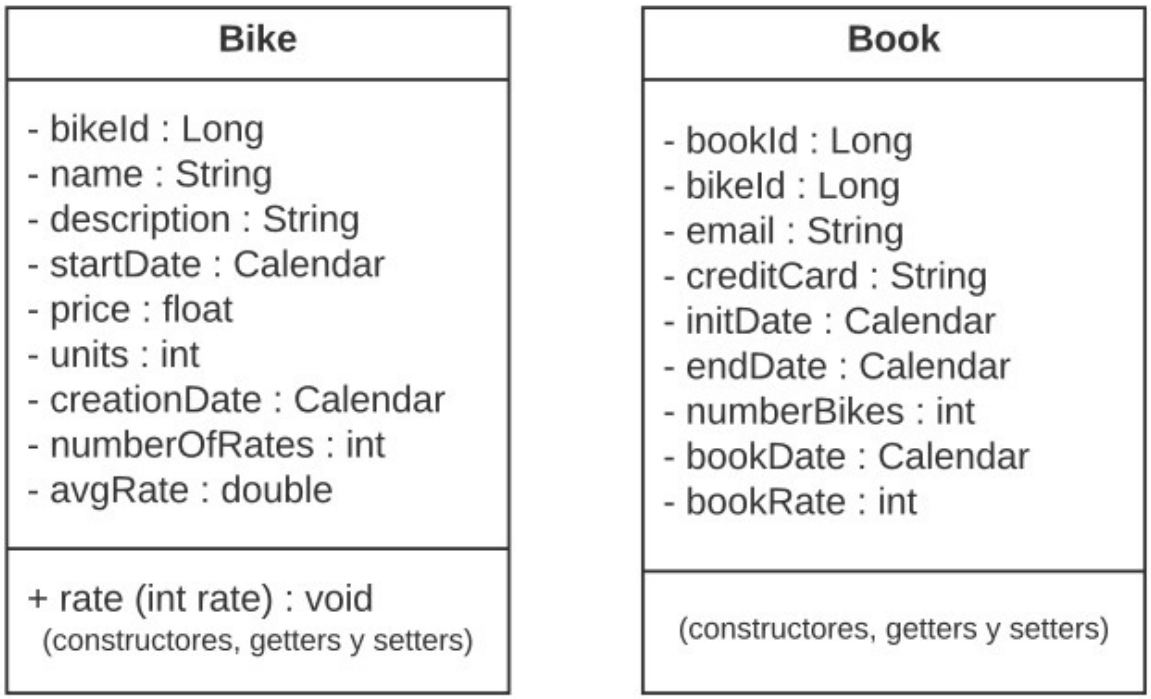
1. Introducción

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

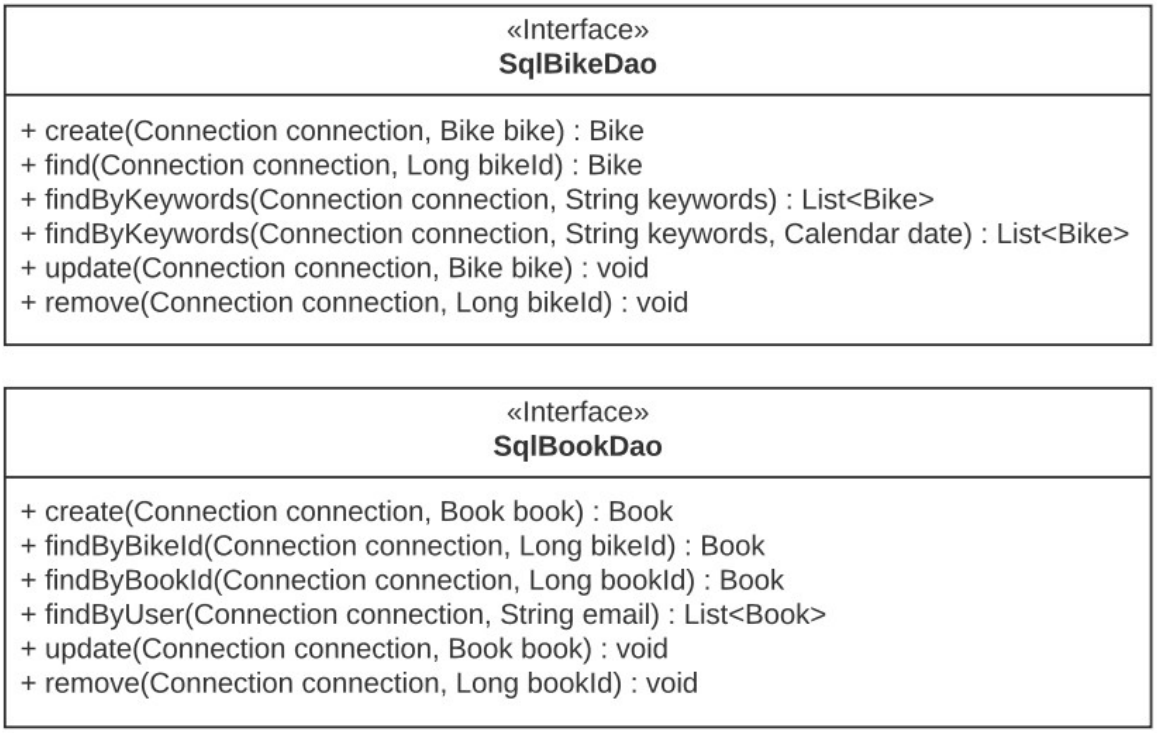
2.1. Entidades



En el diseño de las entidades se incluye el método *rate()* que, al puntuar una reserva este método es llamado para recalcular la media de las puntuaciones de una bicicleta, ya que *avgRate* guarda la media de cada bici. 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

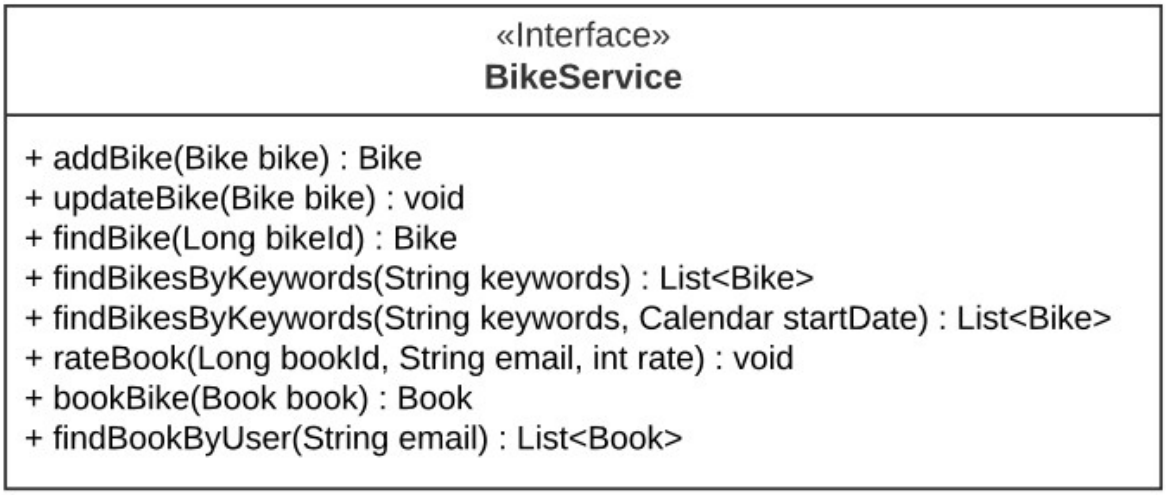


Cabe destacar que para el diseño del DAO de la entidad *Bike* (*SqlBikeDao*) se ha decidido usar la sobrecarga en el método *findByKeywords* para los casos en los que se decide incluir la fecha además de las palabras clave.

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 para la correcta implementación de la aplicación.

Además, en ambos DAOS se incluye un método remove necesario para la realización de las pruebas de la capa modelo.

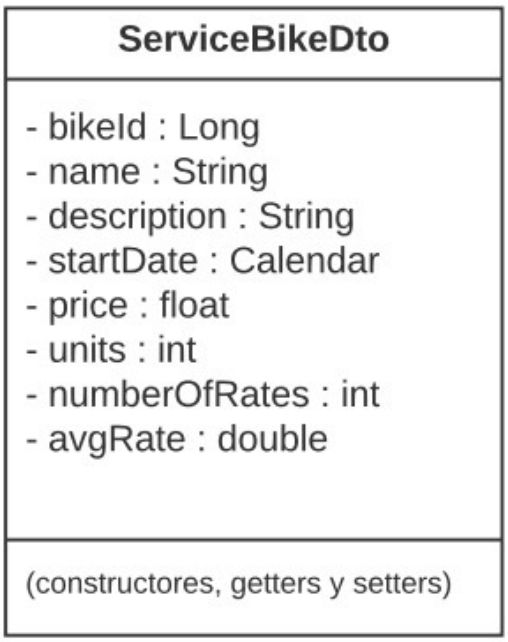
2.3. Fachadas



En el caso de la interfaz de bikeService cabe resaltar que, al igual que con el DAO de *Bike* (*SqlBikeDao*), se utiliza la sobrecarga en el método *finfBikesByKeywords* dependiendo del número de parámetros utilizados.

3. Capa servicios

3.1. DTOs



Dentro del ServiceBookDto incluimos el método *getDays()* para obtener el número de días que dura cada reserva.

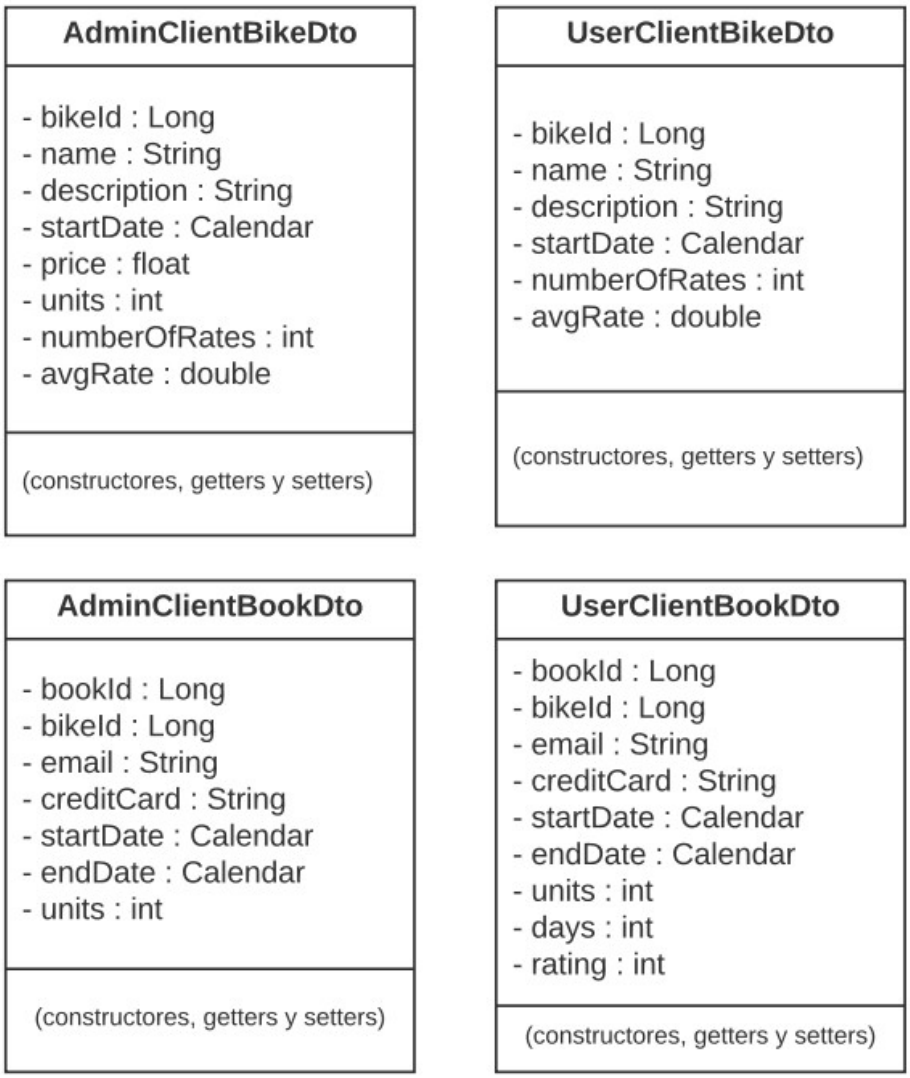
3.2. REST

	URL simplificada	Metodo invocación	Parámetros o DTO	Posibles códigos HTTP respuesta	DTO devuelto
addbike	/bikes	POST	serviceBikeDto	201 CREATED 400 BAD REQUEST	serviceBikeDto
updateBike	/bikes	PUT	serviceBikeDto	400 BAD REQUEST 204 NO CONTENT 404 NOT FOUD	-
findBike	/bikes/{bikeId}	GET	bikeId	200 OK 404 NOT FOUND	serviceBikeDto
findBikesByKeywords	bikes/ {keywords}	GET	keywords	200 OK 404 NOT FOUND	List<serviceBikeDto>
findBikesByKeywords	bikes/ {keywords, startDate}	GET	Keywords, startDate	200 OK 404 NOT FOUND	List<serviceBikeDto>
RateBook	/books/{id}	PUT	BookId, Email, rating	400 BAD REQUEST 204 NO CONTENT 404 NOT FOUD	-
bookBike	/books/ {bookId} {bikeId}/ {Email}/ {creditCard}/ {initDate}/ {endDate}/ {numberBikes} /{rating}/	POST	BookId, bikeId, Email, creditCard, initDate, endDate, numberBikes, rating	201 CREATED 400 BAD REQUEST 404 NOT FOUD	serviceBookDto
findBookByUser	/books/{user}	GET	Email	200 OK 404 NOT FOUND	List<serviceBookDto>

Para las excepciones donde no se encuentra la instancia buscada dentro de la BBDD devolvemos el código 404 Not found, 204 No content se utiliza cuando el contenido se actualiza correctamente. Por otra parte, tenemos el 200 Ok indicando que todo ha sido correcto, el 201 Created para indicar que se ha añadido una nueva tupla dentro de la BBDD y el 401 Bad request para indicar que los datos introducidos dentro de una consulta son incorrectos.

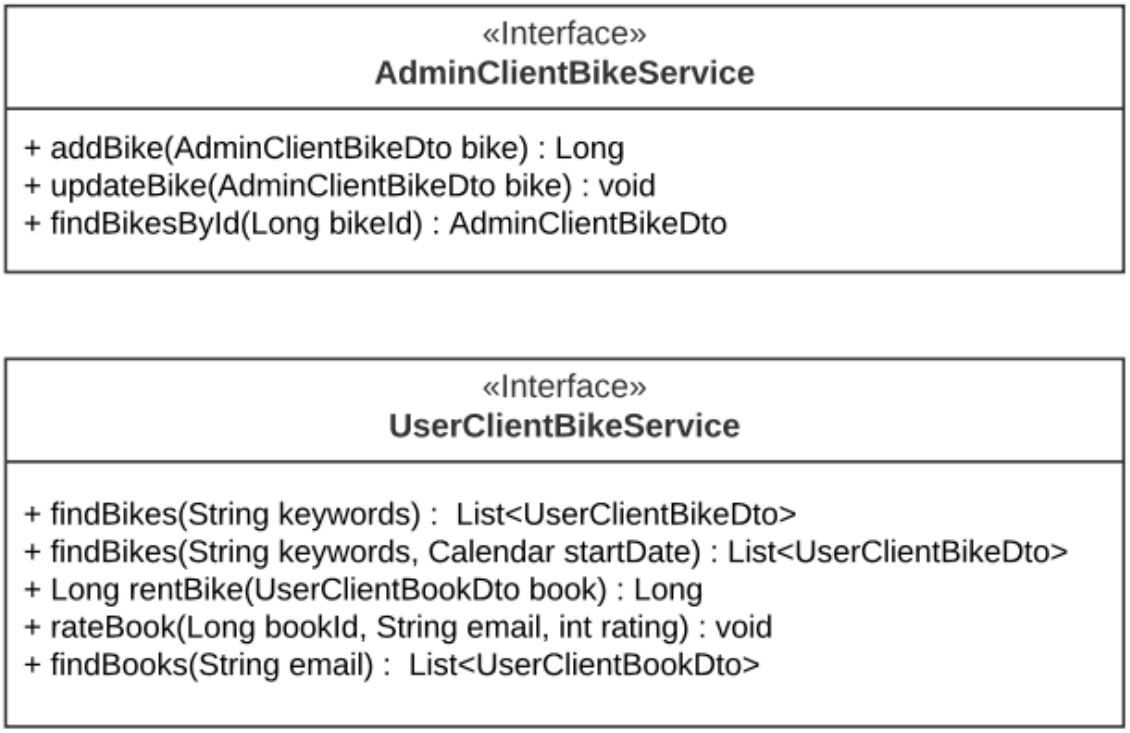
4. Aplicaciones cliente

4.1. DTOs



Utilizamos un parámetro days donde guardamos el número de días que dura cada reserva, asi como un campo rating con la valoración de esa reserva en concreto.

4.2. Capa acceso al servicio



En este caso se dividió la capa de acceso a servicio entre dos posibles clientes, por una parte, tenemos el administrador, encargado de añadir y modificar bicis dentro de la BBDD, así como buscarlas a través de su id.
Por otra parte, tenemos el usuario, encargado de crear y puntuar las diferentes reservas que realize dentro de la BBDD, además, este usuario podrá buscar en la BBDD 'Bike' a través de palabras clave, así como las reservas de un usuario dentro de 'Book'.

5. Errores conocidos

La captura de algunas excepciones puede no realizarse de manera correcta, por lo que en algunos casos puede fallar y no se muestra correctamente la traza.