Memoria de Práctica ISD 2013/2014

Grupo: isg016

Óscar Blanco Novoa

Manuel Sánchez Naveira

Índice

Pág

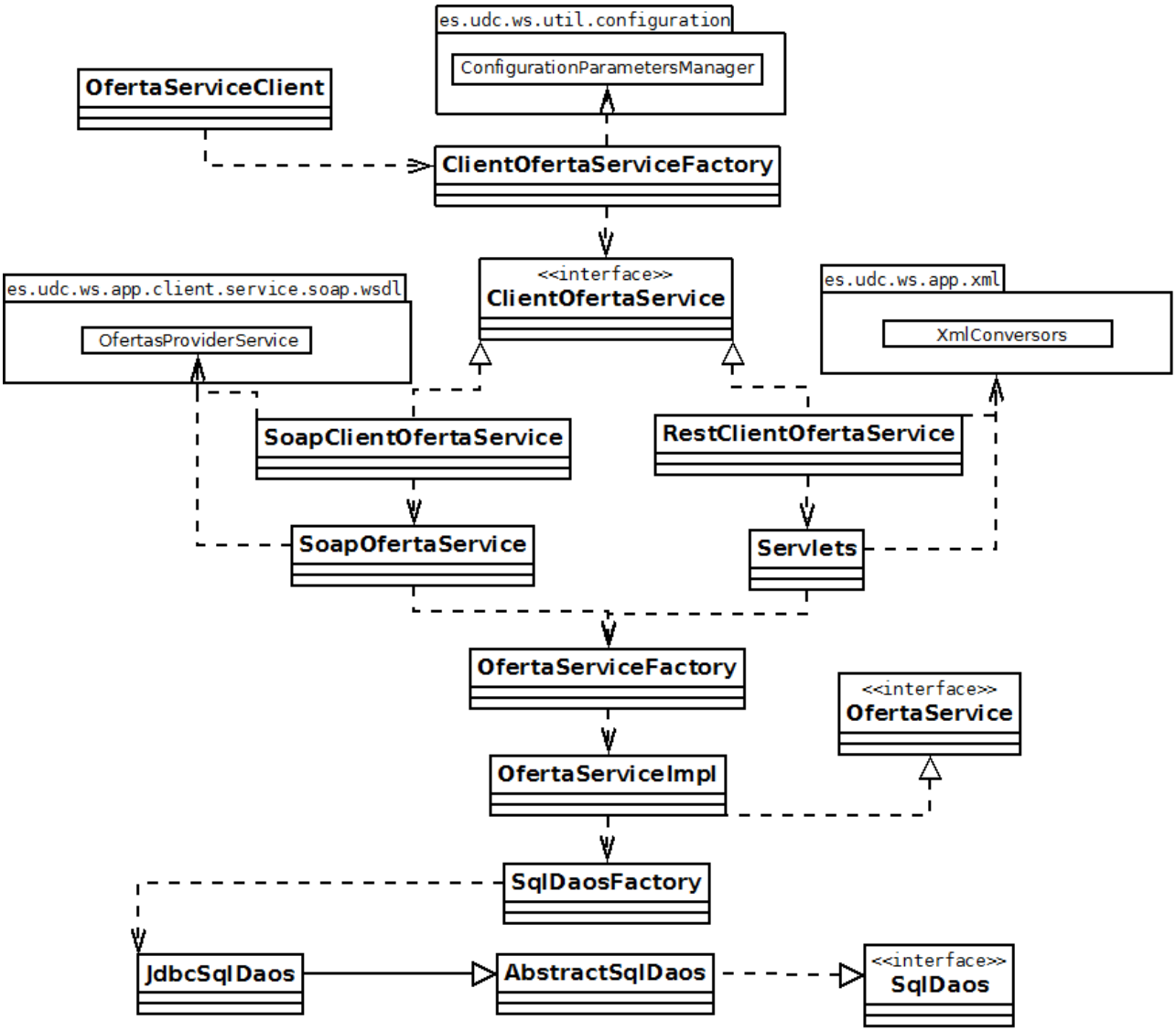
Introducción

En esta memoria se presenta una práctica de ISD del curso 2013/2014, concretamente la del grupo isg016. En ella comentaremos tanto la arquitectura global de la práctica como cada uno de sus módulos, además de sus instrucciones de compilación e instalación.

Aunque hemos hecho la página de prueba en Facebook que representa la hipotética página de Letsmalus, NO hemos implementado el servicio de cuando se crea una oferta se publica también un post con los datos de la misma ni otras peculiaridades de la misma.

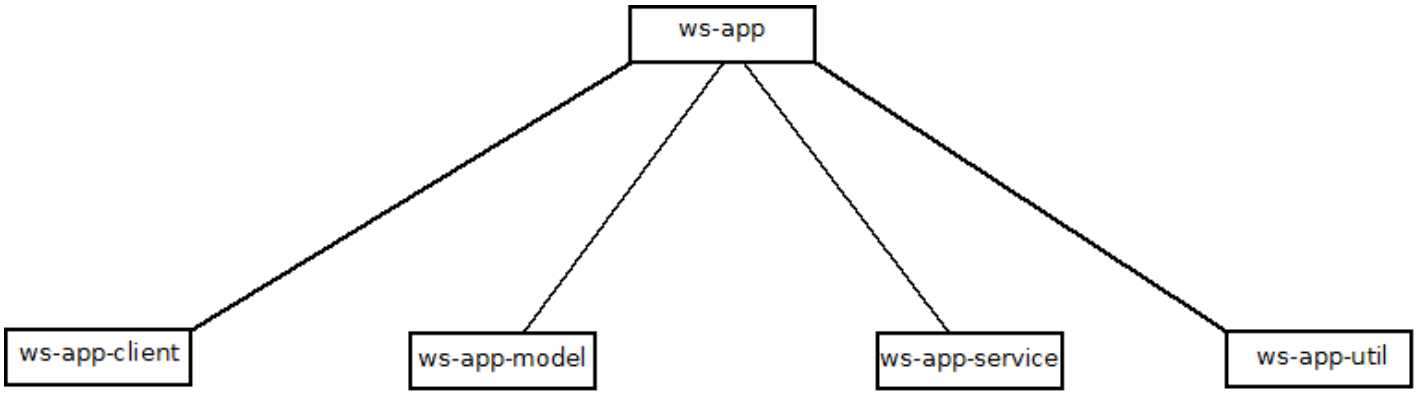
Arquitectura global

La práctica(ws-app) consta de 4 módulos: ws-app-client, ws-app-model, ws-app-service, ws-app-util. Donde como se entiende respectivamente por sus nombres el primero se corresponde con el cliente, el segundo con el modelo, el tercero con la capa de servicios, y el cuarto contiene DTOs, excepciones y conversores de Xml a Dto y viceversa.

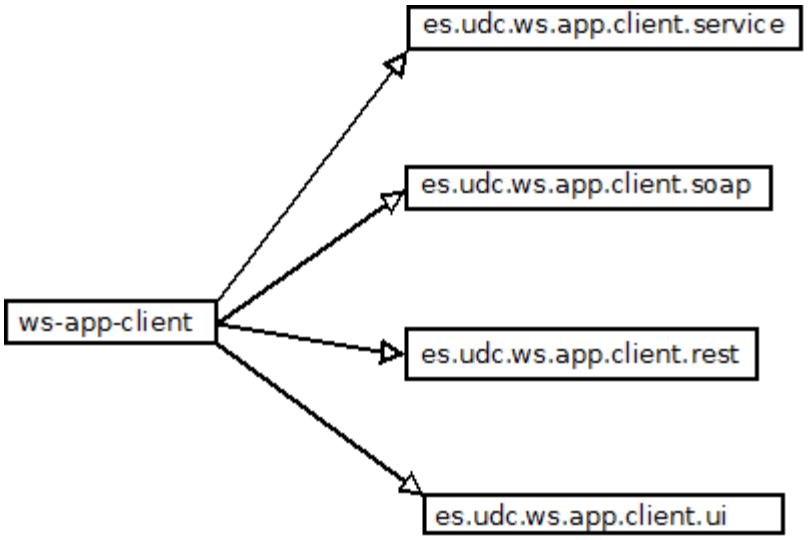


Donde las clases cuyo nombre contiene la palabra Client y el OfertasProviderService del wsdl autogenerado se corresponden con el módulo ws-app-client. Los XmlConversors corresponden al módulo ws-app-util. SoapOfertaService y los Servlets son de ws-app-service. Y por último OfertaService, su Impl, y su factoría además de las clases que contienen la palabra Daos corresponden con los Dao de las ofertas y reservas de ws-app-model.

Módulo i-ésimo



ws-app-client



En el primer submódulo service tenemos la factoría ClientOfertaServiceFactory que es el motor de la aplicación junto con el ConfigurationParametersManager(lector de la configuración del cliente) del módulo ws-util, y ClientOfertaService que es la interfaz de la que heredan RestClientOfertaService del submódulo rest y SoapClientOfertaService del submódulo soap. Además el submódulo soap contiene un conversor de Dto a SoapDto(wsdl.OfertaDto) y viceversa. El submódulo ui es el que recoge del CLI(Command Line Interface) las operaciones que desea el cliente junto con sus parámetros. Además en src/main/resources encontraremos el archivo ConfiguratiónParameters. properties que contiene la principal configuración de la aplicación(Soap/Rest, Tomcat -p 8080 o Jetty -p 9090)

En la Figura 1 que mostramos a continuación se representa el acceso al servicio bien sea empleando Soap o Rest, mostrando la factoría del módulo y su interfaz principal junto a los DTOs implicados.

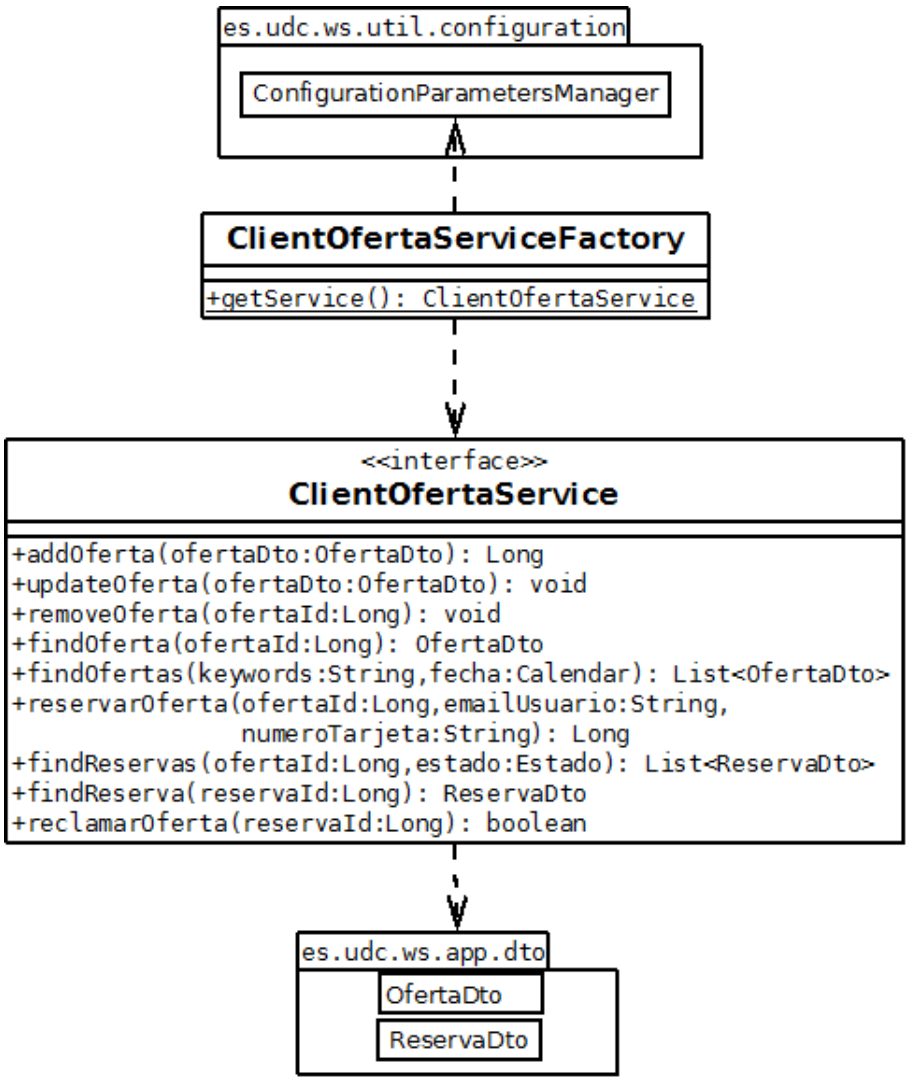
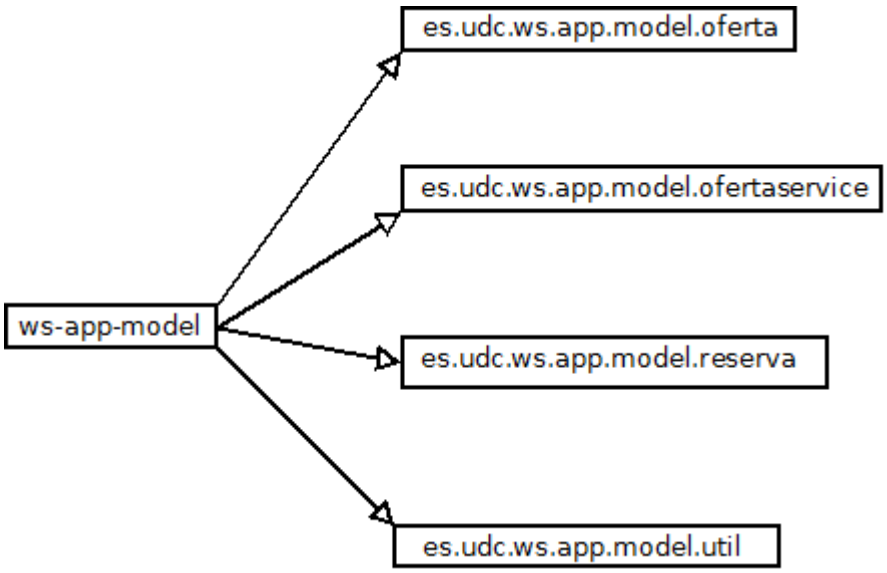


Fig. 1 – Acceso al Servicio con Dto

ws-app-model



En el módulo del modelo están las clases oferta y reserva (Fig. 2) (representadas en tablas mysql en src/sql/MySqlCreateTables.sql) en sus respectivos submódulos oferta y reserva junto con la factoría SqlXXXDaoFactory(XXX bien sea Oferta o Reserva) y la interfaz SqlXXXDao que implementa AbstractSqlXXXDao y esta a su vez hereda Jdbc3CcSqlXXXDao. En la Fig. 3 se representa la estructura para el Dao de Oferta.

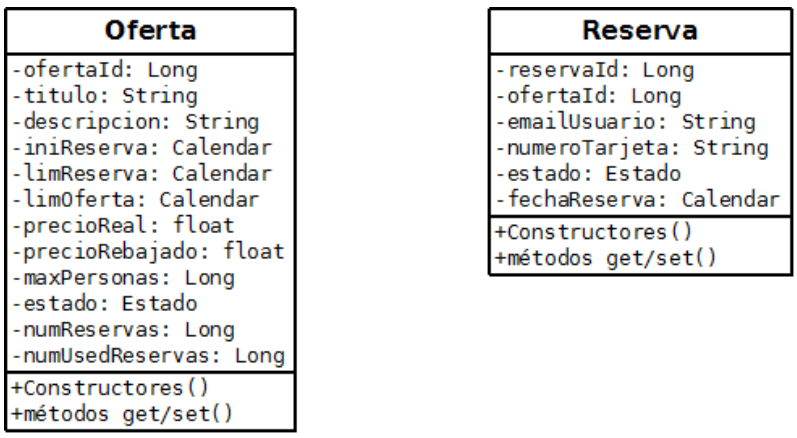


Fig. 2 – Clases Oferta y Reserva

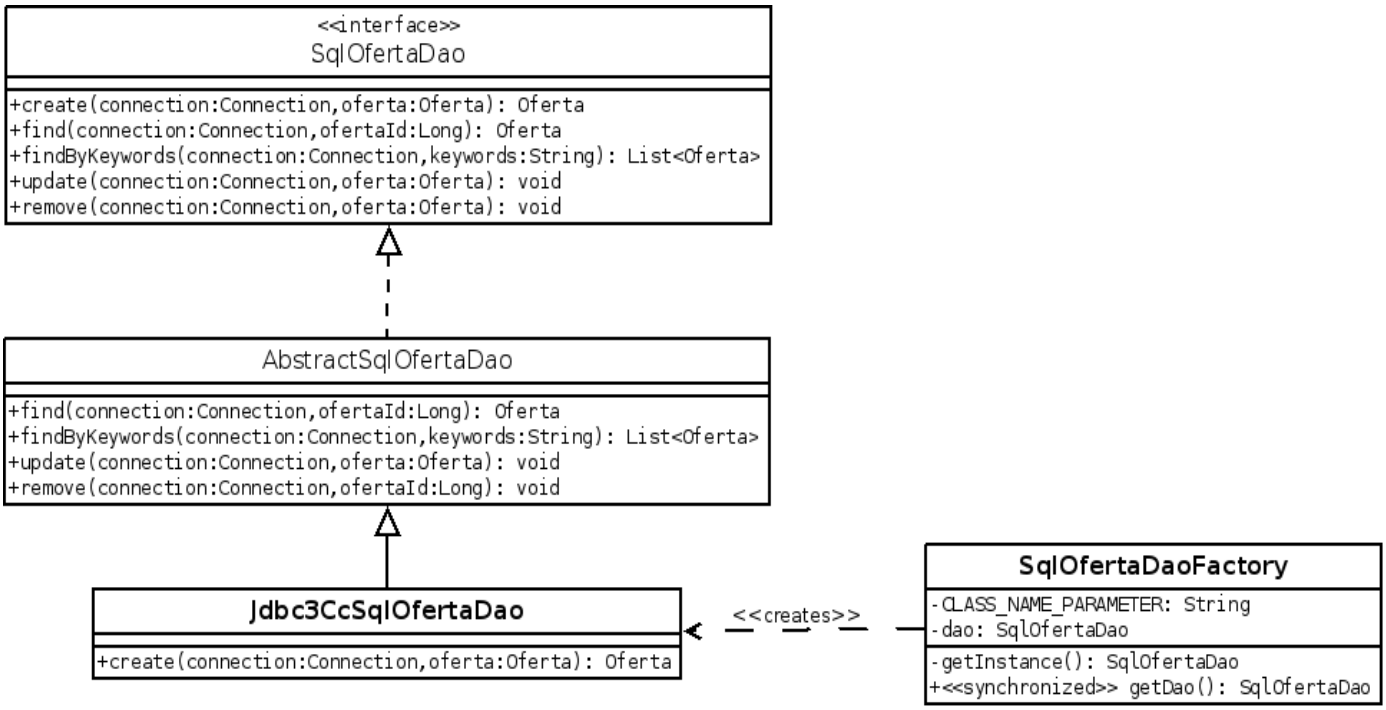


Fig. 3 – Jerarquía de dependencias de OfertaDao

El submódulo del servicio del modelo ofertaservice, contiene la factoría OfertaServiceFactory y la interfaz OfertaService que implementa OfertaServiceImpl. Respectivamente OfertaServiceFactory es la factoría que usamos para ocultar detalles de implementación en la capa de servicios en las clases SoapOfertaService o en los Servlets del Rest. OfertaService es la interfaz que establece los casos de uso y OfertaServiceImpl es su implementación. Aquí mostramos en la Fig. 4 la estructura de este submódulo.

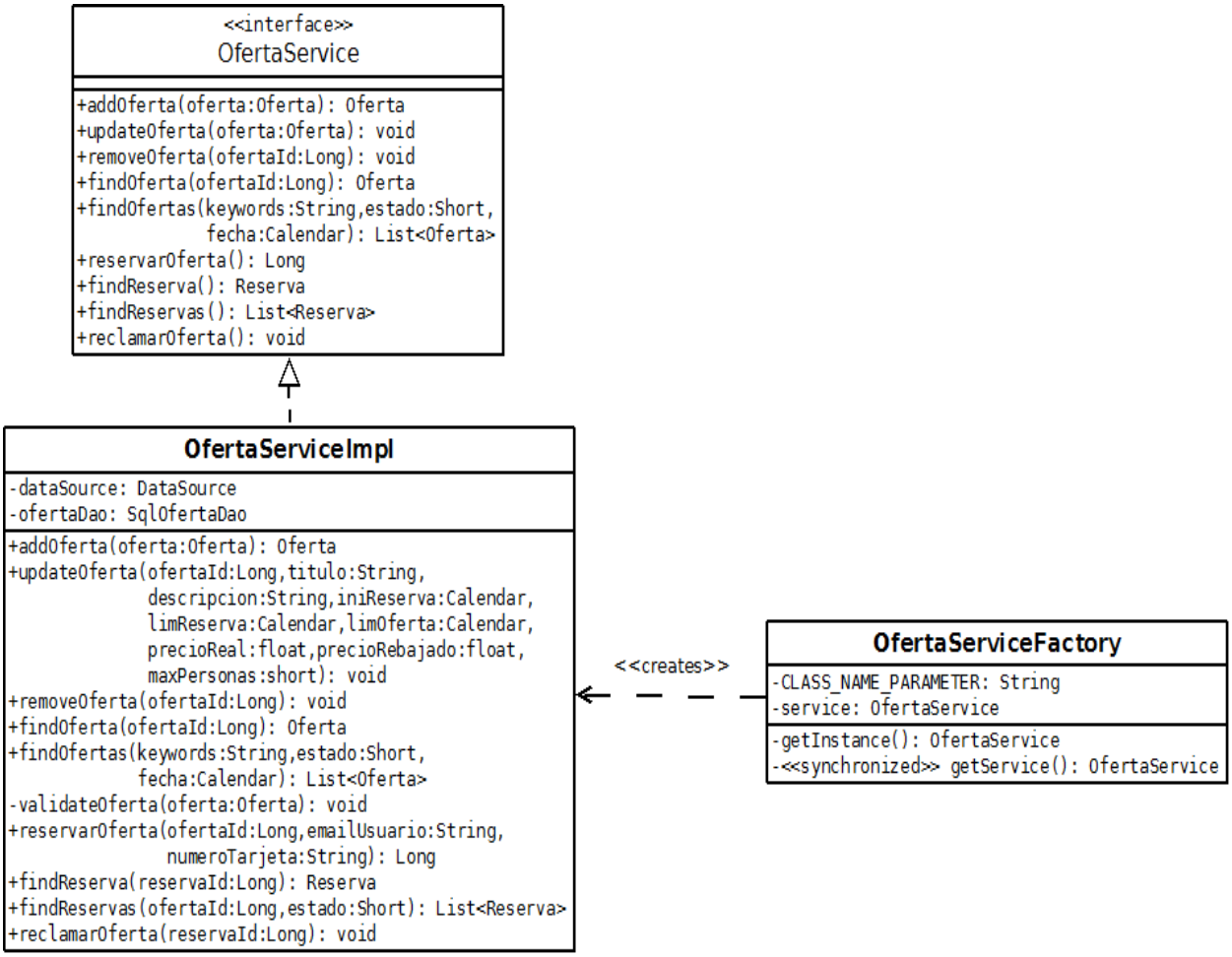
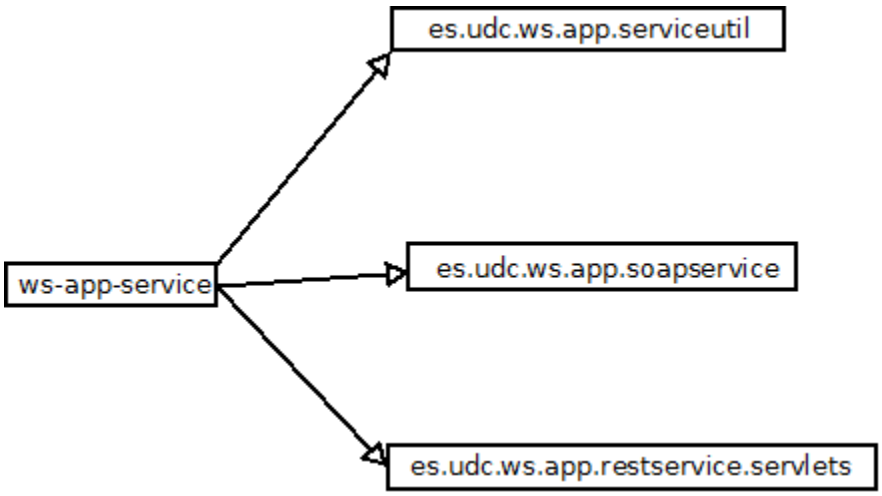


Fig. 4 – Clases principales del submódulo ofertaservice

Por último el submódulo útil del modelo contiene únicamente el nombre de nuestro DataSource en una clase llamada DataSourceConstant(este nombre se mapea en una lista para asignar a nuestra aplicación una conexión con la BD). Esta constante se utiliza como atributo como refleja el diagrama de la Fig. 4 en la clase OfertaServiceImpl.

ws-app-service



El primer submódulo de la capa de servicio, serviceutil, contiene únicamente clases para realizar conversiones de OfertaDto a Oferta y viceversa(lo mismo para las reservas). Cabe señalar como detalle de implementación que, en toOferta de OfertaToOfertaDtoConversor ponemos directamente Oferta.Estado.Creada porque este método solo lo usamos para añadir ofertas, y una oferta se añade con el estado Creada.

El segundo y tercer submódulo los usamos según usemos Soap o Rest. El de Soap contiene la lista de excepciones que tenemos organizadas de un modo concreto como vimos en las transparencias donde cada excepción se divide en dos archivos, que básicamente uno de los cuales termina en Info y contiene los atributos y métodos get/set que es usado por el otro. Luego en este submódulo Soap está su archivo más importante SoapOfertaService donde están definidos cada uno de los métodos, llamados WebMethod según el estándar, además está WebService con sus parámetros, estos dos conforman la clase principal que permite autogenerar el wsdl de soap en el cliente y realizar la comunicación.

En la Fig. 5 ilustramos esta clase principal del servicio Soap que acabamos de comentar junto los DTO (Data Transfer Object).

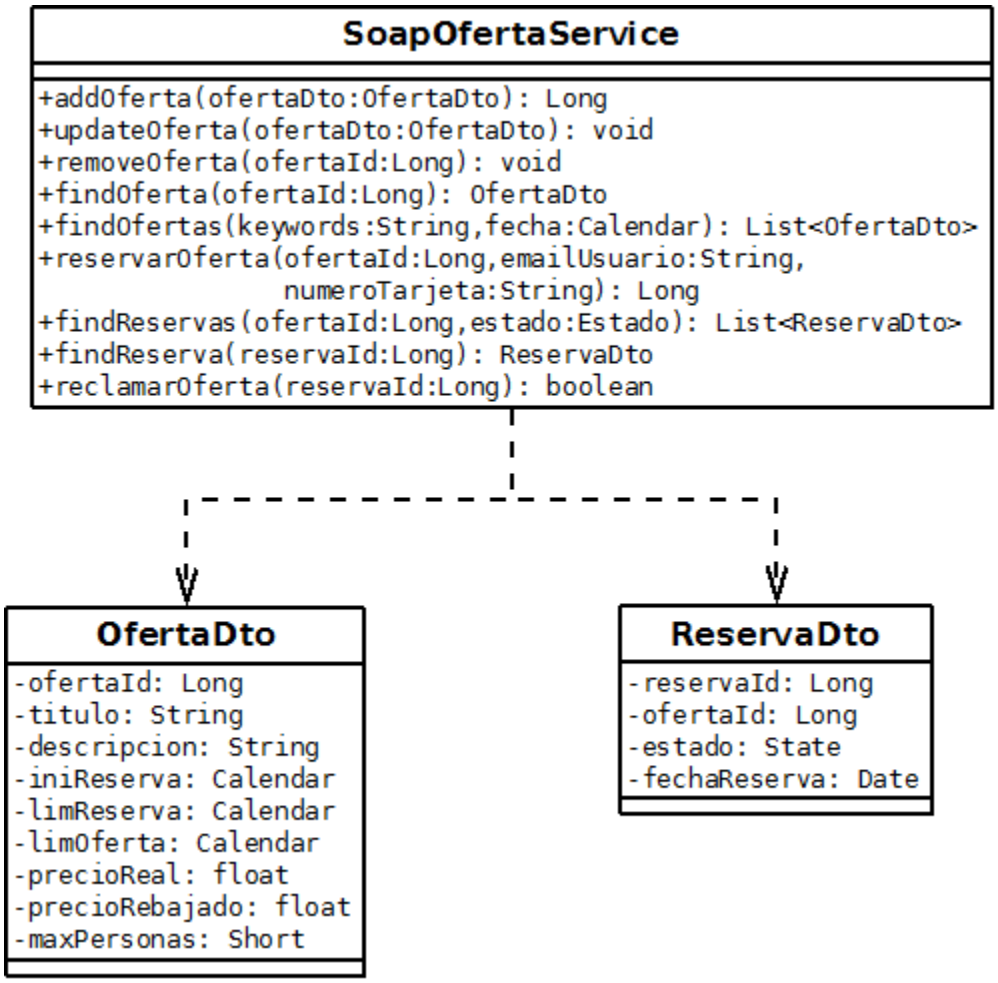
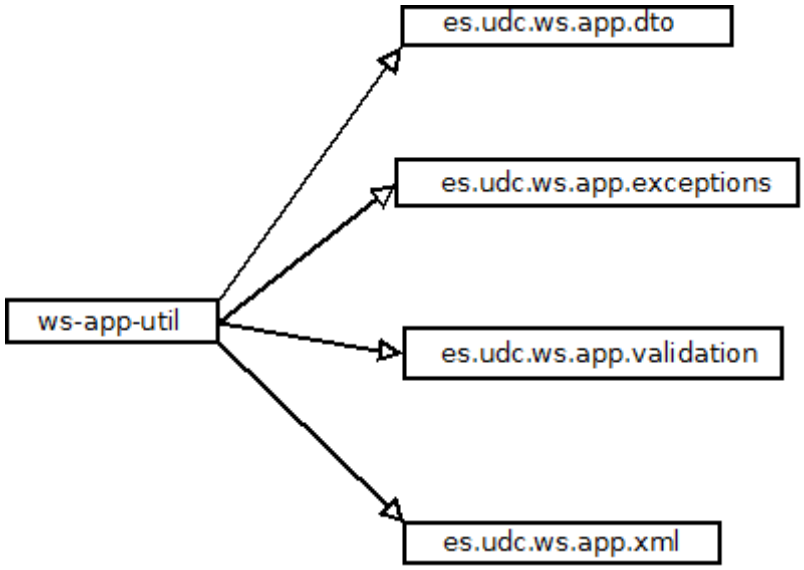


Fig. 5 – Clase principal SoapOfertaService junto con los DTOs

Por último el submódulo Rest contiene únicamente los servlets OfertasServlet y ReservasServlet que heredan de HttpServlets modificando los métodos doGet, doPost que corresponda según la operación a ejecutar(e.g. para addOferta usamos doPost) y devolvemos el código de la excepción que corresponda si ha habido algún fallo o OK o CREATED o NO_CONTENT (para put update esta última) si la operación ha salido perfecta.

No vamos a poner un diagrama, porque heredamos y modificamos doPost, doGet, o lo que corresponda y es de muy bajo nivel, los parámetros que se le pasan a sus métodos es la petición/respuesta (HttpServletRequest/Response) que son siempre los mismos tanto para OfertasServlet y ReservasServlet. Para saber sus dependencias con las otras clases ir al diagrama genérico de la Arquitectura global. Básicamente usan la factoría del servicio del modelo para conocer los detalles de implementación de la operación solicitada.

ws-app-util



El primer submódulo es de los DTO, como tenemos dos clases a representar Oferta y Reserva, tenemos también dos DTO, que ocultan algunos atributos no deseados o innecesarios de las clases originales con el fin de viajar por la red.

El segundo submódulo es el de las excepciones lógicas, e.g. un usuario que reserva dos veces la misma oferta ==> OfertaEmailException, e.g. 2 una oferta que expiro su tiempo de reclamación o de disfrute ==> OfertaReclamaDateException, etc.

El submódulo validation es nuestro propio PropertyValidator, donde definimos a partir del PropertyValidator que nos proporcionabais algunos métodos y los modificamos o bien añadimos uno nuevo como el validateEmail. Este PropertyValidator se emplea en el método ValidateOferta de OfertaServiceImpl que veíamos en una de las Figuras/DIagramas anteriores.

Por último el módulo xml, contiene los conversores xml que usamos junto al estilo Rest: XmlExceptionConversor, XmlOfertaDtoConversor, XmlReservaDtoConversor. Y también contiene un ParsingException que lanzamos cuando hay un problema al parsear en alguno de los ficheros anteriores.

Compilación e Instalación

Así como en la corrección de las anteriores iteraciones, se supondrá un entorno bien configurado, nosotros no usamos ningún aspecto particular de configuración, es similar a los ejemplos (mysqld en puerto por defecto, nombres de la bd ws, Tomcat en 8080, Jetty en 9090, DataSource de los ejemplos ws-javaexamples-ds).

1. Arrancar la BD con mysqld.
2. Configurar src/main/resources/ConfigurationParameters.properties según queramos usar Soap/Rest con Jetty/Tomcat. Para Soap dejamos sin comentar su primera línea, comentando la de Rest(la segunda), y viceversa. Ponemos el puerto que sea según el servidor.
3. En ws-app: mvn sql:execute install.
4. Si usamos Tomcat copiamos el .war generado en el directorio target de ws-app-service a la carpeta webapps de Tomcat, también conviene borrar el directorio ws-app-service dentro de webapps.
5. Ejecutamos nuestro servidor de aplicaciones Jetty/Tomcat.
6. Añadimos las operaciones deseadas por línea de comandos en ws-app-client, e.g. Añadir Oferta:

```
mvn exec:java -Dexec.mainClass="es.udc.ws.app.client.ui.OfertaServiceClient" -
Dexec.args="-a 'Fiesta de Nochevieja ISD1' 'Fiesta de Nochevieja con todo
incluido' '12/12/2013 09:00' '29/03/2014 23:00' '31/03/2014 04:00' 60 40 2"
```