LETSMALUS

Daniel Ruiz Pérez

Rodrigo Martín Prieto

Ismael Barbeito Vázquez

Grupo ISG002

Índice

Introducción

El presente documento conforma la memoria de prácticas del grupo isg002 de Internet y Sistemas Distribuidos, curso 2013/14. En concreto, se ha hecho el diseño e implementación de un programa para la empresa de ofertas LetsMalus. Por problemas de tiempo, se ha pactado con la empresa la **no** realización de la integración con Facebook de la aplicación, quedando pendiente para ser hecho a posteriori por otro equipo de desarrollo.

Antes de proceder con la documentación requerida, se han de tener en cuenta los siguientes aspectos:

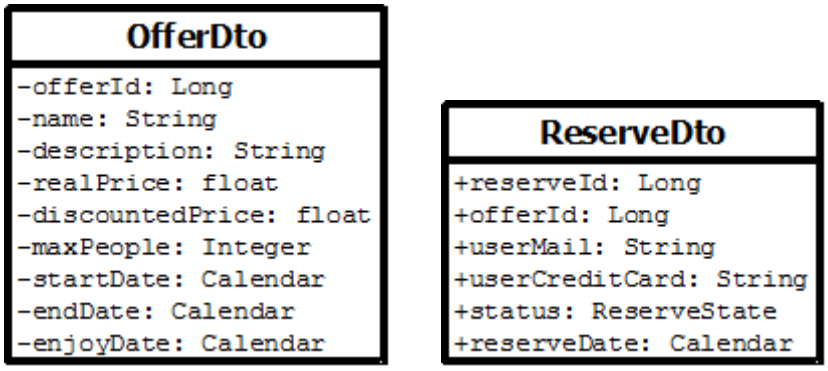
1. Todos los diagramas presentes en la memoria han sido realizados usando el software libre Dia. Esta herramienta carece de manera formal de representar las excepciones lanzadas por los métodos, motivo por el cual no aparecen reflejadas.
2. A la hora de realizar el compilado y la ejecución, se supone un equipo con las configuraciones indicadas por el profesorado de la asignatura a todos los niveles (servidor de aplicaciones, bases de datos…).
3. Para las pruebas, se adjunta un fichero “Comandos.txt” en el directorio raíz del proyecto con los comandos necesarios para las ejecuciones indicadas en el enunciado de la práctica.

Diseño

El programa está diseñado siguiendo una arquitectura por capas. A modo genérico, contamos con cinco subproyectos o módulos a considerar:

1. **ws-util**: proporciona clases básicas para la gestión de archivos de configuración, excepciones, validaciones… Fue proporcionado por el profesorado a principio de curso.

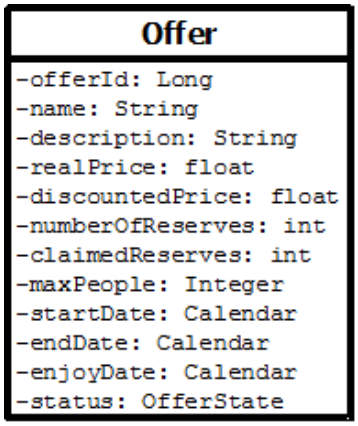
2. **ws-app-util**: contiene los siguientes paquetes
   - dto: engloba los Dto de las dos clases persistentes usadas:

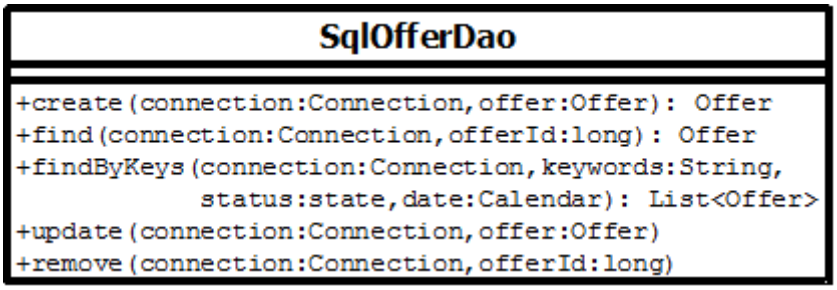


   - exceptions: engloba las excepciones lógicas diseñadas: DuplicatedReserveException, FullOfferException, NotRemovableOfferException, NotUpdatableOfferException, OfferExpirationException y ReserveExpirationException.
   - format: contiene la clase FormatUtils, con métodos pensados para los distintos parseos necesarios durante el programa.
   - validation: similar al validation del ws-util. Simplemente añade nuevos métodos para hacer validación en el modelo.
   - xml: contiene los conversores para realizar el paso de objetos java a XML.

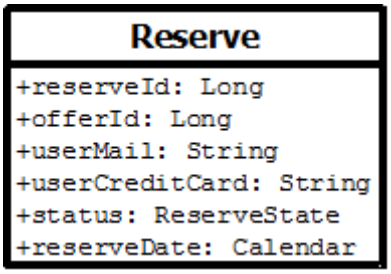
3. **ws-app-model**: implementa la lógica de negocio y la capa de acceso a datos, organizados en los siguientes paquetes:

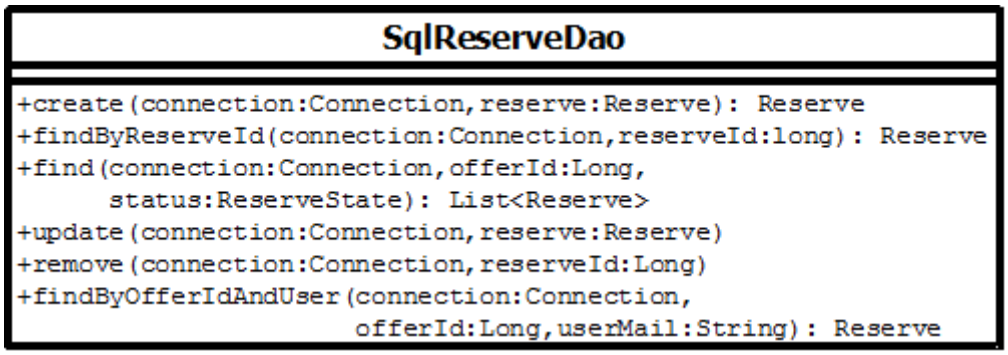
- offer: contiene el Dao y clase persistente de la entidad Oferta.



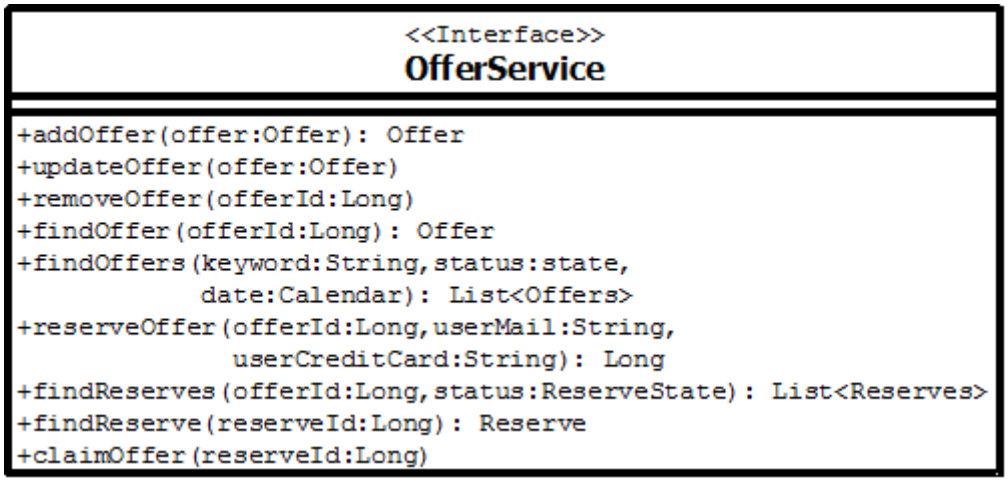


- reserve: similar al paquete offer, pero para la entidad Reserva





- offerservice: contiene la implementación de los casos de uso del cliente



- util: contiene la clase ModelConstants.

4. **ws-app-service**: contiene las implementaciones del servicio, en base a las tecnologías solicitadas por el cliente. La organización de paquetes es la que sigue:
   - servlets: contiene los servlets usados para Rest (se han agrupado los casos de uso de forma lógica en dos sevlets: OffersServlet y ReservesServlet)
   - serviceutil y util: ambos contienen los conversores de offer y reserve a offerdto y reservedto. Duplicado siguiendo las directrices del ejemplo de Movies.
   - soapservice: contiene la implementación Soap del servicio, así como las SoapException

5. **ws-app-client**: engloba la capa de acceso al servicio y la interfaz gráfica en los siguientes paquetes:
    - service: contiene la interfaz de acceso que deben implementar Soap y Rest, así como un mock de pruebas y la factoría de esta interfaz.

    - rest: implementación rest de la interfaz del paquete anterior
    - soap: implementación soap.
    - ui: la interfaz por comandos diseñada como ejemplo

Compilación e instalación

Asumiendo, como ya se mencionó en la introducción, un PC correctamente configurado, simplemente sería necesario descargar el programa desde el repositorio (tanto por terminal como con el plugin de svn para Eclipse) y realizar los siguientes pasos:

1. Importar el proyecto con Maven en Eclipse.
2. Arrancar el servicio de MySQL .
3. En la carpeta raíz del proyecto (LetsMalus), ejecutar el comando “mvn install”.
4. Una vez finalizado el paso anterior, arrancar el servicio. Durante las prácticas hemos trabajado con dos servidores de aplicaciones: jetty y tomcat.
5. *(EN CASO DE ESCOGER JETTY)* Una forma sencilla es crear un perfil de configuración maven en Eclipse, añadiendo como meta “jetty:run”
5. *(EN CASO DE ESCOGER TOMCAT)* Dentro de la carpeta de tomcat, en el directorio webapps, se debe situar el archivo .war que se genera con el comando mvn install en el directorio target del módulo del servicio. Luego simplemente habría que arrancar el startup.bat.
6. Una vez arrancado el servicio, solo faltaría ejecutar el cliente. Para ello, se puede usar el comando que sigue:

   exec:java Dexec.mainClass="es.udc.ws.app.client.ui.OfferServiceClient" -Dexec.args="ARGUMENTOS”

   A continuación se deja una relación de los argumentos soportados por el programa

```
Usage:
[add] OfferServiceClient -updateOffer <name> <description> <price>
          <discountedPrice> <MaxReserves> <startDate> <endDate>
          <enjoyDate>
[update] OfferServiceClient -updateOffer <offerId> <name> <description>
                <price>   <discountedPrice>   <MaxReserves>   <startDate>
<endDate>
                <enjoyDate>
[remove] OfferServiceClient -removeOffer <offerId>
[findOffer] OfferServiceClient -findOffer <offerId>
[findOffers] OfferServiceClient -findOffers <keywords>
[addReserve] OfferServiceClient -reserveOffer <offerId> <userMail>
          <userCreditCard>
[findReserve] OfferServiceClient -findReserve <reserveId>
[findReserves] OfferServiceClient -findReserves <offerId> <state>
[claimOffer] OfferServiceClient -claimReserve <reserveId>
```

Errores conocidos

Para finalizar con la documentación, creemos pertinente informar de una serie de comportamientos que podrían resultar confusos a nivel de funcionalidad:

- Si se trata de reclamar una reserva ya reclamada, el sistema permitirá hacerlo siempre y cuando dicha reserva no haya expirado. Aunque puede ser considerado una práctica errónea, lo hemos hecho de esta manera ya que el cliente no había especificado que hacer en esta situación concreta. Además, nos pareció que podría ser interesante permitir que el usuario final pudiera consultar su código de reserva en más de una ocasión, sin tener que modelar una operación propia para ello.
- A la hora de ejecutar el script de creación de las bases de datos, Maven detectará errores en caso de existir previamente los índices, pero proseguirá con la compilación de forma normal.