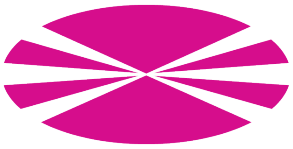


UNIVERSIDADE DA CORUÑA

Memoria Programación Avanzada
CinemaApp

Marcos Puente Blanco
Jesús López Becerra
Xoan Iago Suárez Canosa

1. Arquictectura Global

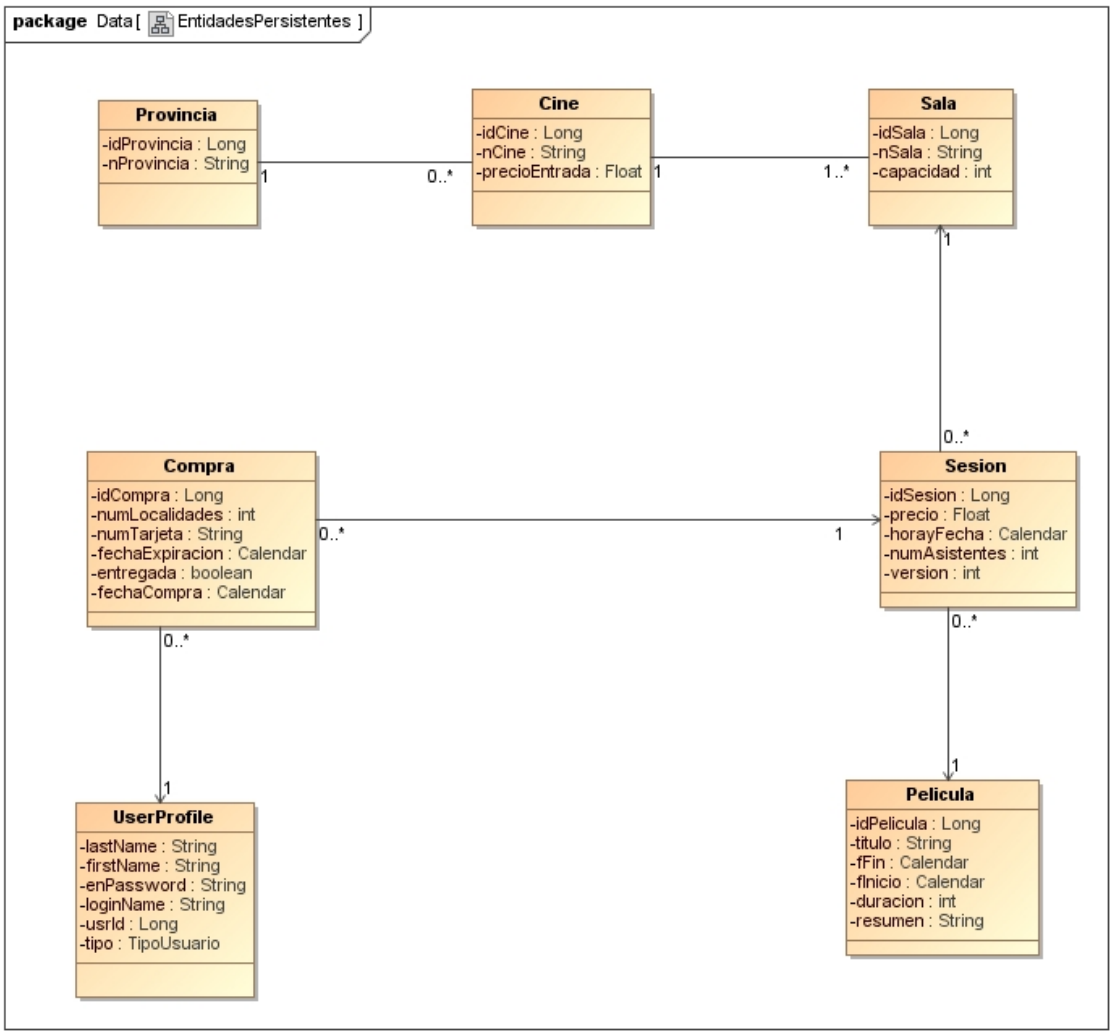
La aplicación **CinemaApp™** se divide en 2 paquetes principales: `src` y `target`.

- `src`: Contiene los ficheros fuente de nuestra aplicación. Se divide en varios paquetes:
  - `main`: Contiene los ficheros de implementación de nuestra aplicación, tanto de la parte modelo como de la parte web. Se divide fundamentalmente en 2 paquetes principales: `java` y `resources`.
    - `java`: Conformada por los paquetes que contienen las clases de implementación de las clases persistentes, así como los diferentes servicios del modelo y clases utilidad.
    - `resources`: Contiene fundamentalmente todos los ficheros de tapestry (páginas, componentes, ficheros `*.properties`) que conforman la parte web de nuestra aplicación.
    - `webapp`: Alberga los ficheros necesarios para la aplicación web, como son los css, javascript, las imágenes necesarias en la web,etc...
  - `sql`: Contiene el fichero de creación de las tablas que vamos a utilizar en nuestra aplicación, así como un fichero para crear datos de prueba que se utilizarán para verificar el correcto funcionamiento de la aplicación.
  - `test`: Fundamentalmente contiene los ficheros fuente que serán las pruebas de integración de la capa modelo. En ellos se verifica el correcto funcionamiento de los servicios implementados: user, catalogo y compra.
- `target`: Contiene los ficheros compilados a partir de los ficheros fuente anteriores, además del fichero `*.war` que utilizaremos para ejecutar nuestra aplicación en Tomcat(Producción).

2. Modelo

2.1. Clases Persistentes

Diagrama de Entidades Persistentes



Como se puede observar en el diagrama, un Cine estará localizado en una única Provincia y en dicha Provincia pueden localizarse varios cines. En esta aplicación se ha decidido que la navegación en esta relación sea bidireccional, por lo que el Cine contendrá su Provincia, y las Provincias mantendrán un conjunto de Cines que están localizadas en ella.

De forma similar se ha procedido con los Cines y las Salas, éstas permitirán navegar al Cine al que pertenecen y de forma análoga a la situación anterior, los cines mantendrán el conjunto de salas que lo forman.

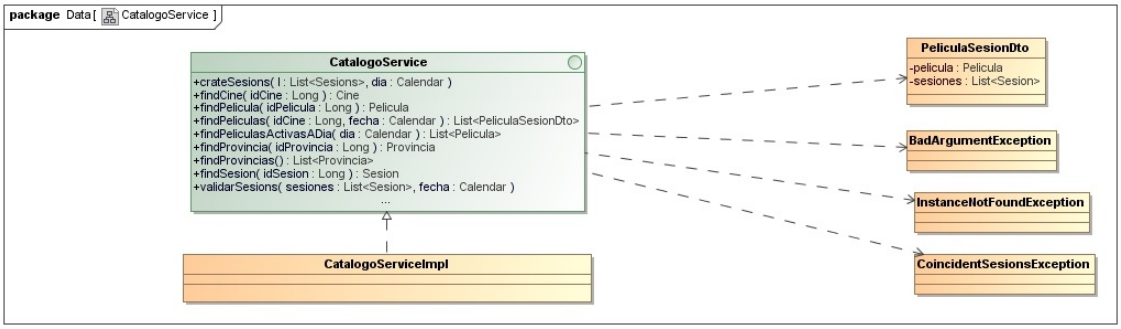
La entidad Sesión, además de las propiedades que se observan en el diagrama estará relacionada con la Película que se proyecta y la Sala en donde tendrá lugar. Por lo tanto, una Sesión permitirá navegar hacia la Sala y la Película, pero no al revés, ya que en una Sala se producirán muchas Sesiones a lo largo del tiempo y una Película se puede proyectar un número elevado de veces.

La entidad Compra representa, como bien indica su nombre, una compra que un Usuario realiza de una Sesión. Por lo tanto, estará relacionada con la Sesión correspondiente y con el Usuario que la ha efectuad y permitirá la navegación correspondiente.

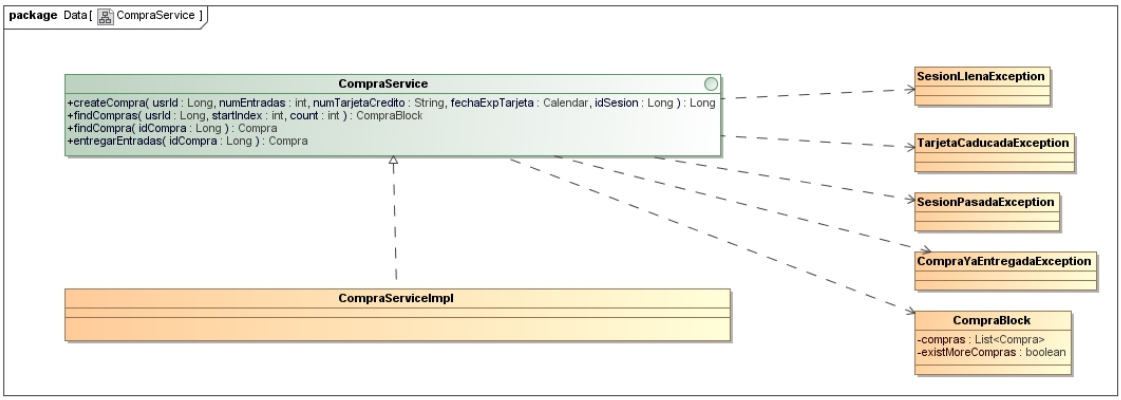
2.2. Interfaces de los Servicios Ofrecidos por el Modelo

En cuanto a los servicios del modelo, se ha optado por una división funcional en 3 diferentes servicios:

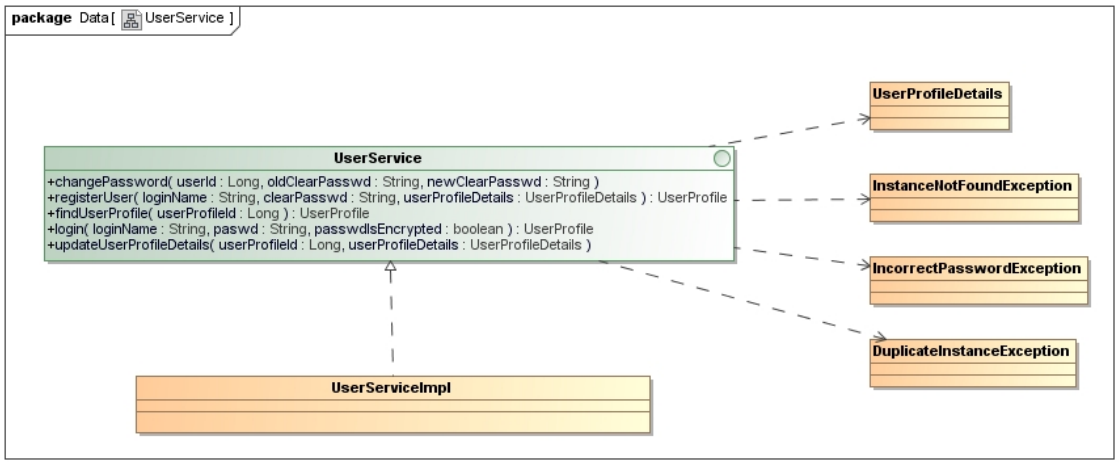
- `CatalogoService:` Contiene las operaciones del modelo relacionadas con la información al usuario: Películas en Cartelera, detalles de las películas, detalles de las sesiones...Así como la operación del tipo de usuario *Administrador* para confeccionar la cartelera de un cine en un día concreto.



- `CompraService:` Contiene las operaciones del modelo que permiten a los usuarios realizar una compra de entradas para una determinada sesión, así como de ver su historial de compras. A su vez, también contiene las operaciones que permiten al tipo de usuario *Taquillero* comprobar las entradas de un espectador y entregárselas para que pueda disfrutar de la película.



- `UserService:` Contiene las operaciones que permiten la gestión de los usuarios: registrar, loguear, actualizar información de usuario y cambiar contraseña.



2.3. Otros Aspectos

Cabe destacar que tanto las operaciones de los servicios como de los DAOs están sujetas a situaciones que se pueden producir en el transcurso normal de la aplicación y que consideramos como excepciones, algunos ejemplos: Intentar comprar más entradas de lo permitido, crear sesiones en un día en que ya las hay, intentar crear un usuario con un login que ya existe... Para estas situaciones utilizamos las excepciones tipo *çhecked"*. Se ha decidido ubicar estas clases además de otras clases de utilidad para el funcionamiento de la aplicación en un paquete denominado `util` dentro del modelo.

3. Interfaz Gráfica

4. Partes Opcionales

4.1. AJAX

Se ha implementado la parte opcional de la práctica correspondiente a la utilización de AJAX en la parte web de la aplicación. Se ha decidido utilizarla en casos en los que resulte realmente útil para el usuario final, detectando 2 situaciones concretas:

- **Selección de cine:** A la hora de escoger el cine favorito aparecerá un componente de selección que permite seleccionar la provincia del cine. La selección de la provincia iniciará un evento que mostrará otro componente selección para el cine. Esto es una gran ayuda de cara al usuario en vez de un enfoque sin AJAX como seria un componente selección que contuviera todos los cines con un guión y la provincia a la que pertenecen.

- **Confección de cartelera:** En la página para confeccionar la cartelera se dispone de 2 zonas Ajax.

  Una muestra una tabla con las sesiones creadas hasta el momento y la otra el formulario en el cual introducimos los datos de las nuevas sesiones que se van creando. Una vez que se hace el submit del formulario, se recarga la tabla en caso de que la sesión cumpla los criterios de validación y en caso contrario se muestra el error pertinente. También se recargará la tabla en caso de que se elimine una de las sesiones que la forman mediante un click sobre el botón de eliminar.

Al finalizar se redirigirá a una página de éxito de confección de la cartelera.

4.2. Selenium

Se ha implementado la parte opcional de la practica correspondiente a las pruebas con Selenium. En primer lugar, se han añadido las lineas necesarias al `pom.xml` para resolver las dependencias. Se ha creado un paquete dentro de `src/test` con dos clases:

- `login:` Se implementa el caso de uso en el que un usuario entra en la aplicación y se registra. Para asegurarse de su correcto funcionamiento, se comprueba que aparece el nombre del usuario en la barra superior.
- `comprarEntrada:` Se implementa el caso de uso en que un usuario sin autenticar selecciona el cine preferido, posteriormente escoge la primera sesión de la primera película y procede a comprarla. Como no está autenticado, se le redirige al formulario de autenticación y se autentica. Al hacer esto, se le redirige al anterior formulario de compra y compra una entrada de esa sesión. Para comprobar que se ha realizado correctamente, se comprueba que el identificador de compra del primer elemento del historial de compras (el más reciente) es el mismo que el identificador devuelto por la página de compra exitosa anterior.

5. Compilación e Instalación de la Aplicación

Para la instalación y compilación de la aplicación se realizaran los siguientes pasos, dado un entorno igual al establecido en la asignatura de "Programación Avanzada" previamente a la realización de la práctica:

NOTA: Los pasos para realizar la compilación e instalación de la aplicación son igualmente aplicables a Windows como a Linux, sin embargo los comandos utilizados son válidos únicamente para Linux.

1. Crearemos una carpeta para la descarga de los archivos fuente de la aplicación a través de Subversion y accederiamos a ella:
   - `mkdir defensa`
   - `cd defensa`
2. A través del uso de un cliente de subversion nos bajaremos de nuestro repositorio de Subversion el tag "it-3" correspondiente a la última entrega de la práctica a la carpeta previamente creada.
   - `svn co https://svn.fic.udc.es/grao3/pa/13-14/pa005/pojo-app/tags/it-3/`
3. Una vez descargados estos archivos entraremos a la carpeta "it-3" que debería estar localizada dentro de la carpeta anteriormente creada y ejecutaremos la compilación de los archivos fuente usando Maven con el siguiente comando:
   - `cd it-3/`
   - `mvn sql:execute install`

   Con esto tendremos la tablas creadas en la base de datos y nuestra aplicación web lista para ser copiada a la carpeta de aplicaciones de Tomcat.

4. Copiaremos el archivo .war localizado dentro de la carpeta “pojo-cinema-app” a la carpeta “webapps” del directorio de instalación de Tomcat. Así mismo copiaremos el driver jdbc de mysql que se encuentra en el repositorio local de maven a la carpeta lib dentro de Tomcat para poder ejecutar la aplicación (Si la carpeta lib no existiera la crearíamos).

5. Para finalizar la configuración de Tomcat nos iríamos a la carpeta “conf” localizada en su directorio de instalación y modificaríamos el archivo “server.xml” añadiendo las líneas para la configuración del Data Source.

   Añadir las siguientes líneas a conf/context.xml, dentro del <Context>tag.

   <ResourceLink name=”jdbc/pojo-examples-ds.”
   global=”jdbc/ws-javaexamples-ds”
   type=”javax.sql.DataSource/>

6. En la carpeta de tomcat por último debemos modificar el fichero bin/context.xml incluyendo el nombre del datasource como `jdbc/ws-javaexamples-ds`

7. Una vez realizado esto podremos ejecutar Tomcat para correr nuestra aplicación web. En nuestro caso Tomcat está cofigurada para correr sobre el puerto 8080, así que podríamos comprobar que se está ejecutando accediendo a la siguiente URL desde nuestro navegador:

   - `http://localhost:8080`

   Arrancar Tomcat:

   - `cd «Tomcat home»/bin`
   - `startup.sh`

   Parar Tomcat:

   - `shutdown.sh`

   Llegados a este punto ya tenemos nuestra aplicación web lista y preparada para recibir peticiones.

6. Problemas Conocidos

En la realización de la práctica se ha intentado ser lo más fiel posible al enfoque orientado en la asignatura y a los estándares de codificación, por lo que se espera y desea que los errores sean los menores posibles. En todo caso, en un principio no se han detectado errores antes de la entrega de la práctica.