SITIO WEB DE VENTAS DE ENTRADAS DE CINE

PRÁCTICA DE PROGRAMACIÓN AVANZADA

Ingeniería del Software
Curso académico 2019-2020

Julia Álvarez Gurdiel – julia.gurdiel
Jaime Cabero Creus – Jaime.cabero
Ángel Paderne Cózar – a.pcozar

UNIVERSIDADE DA CORUÑA

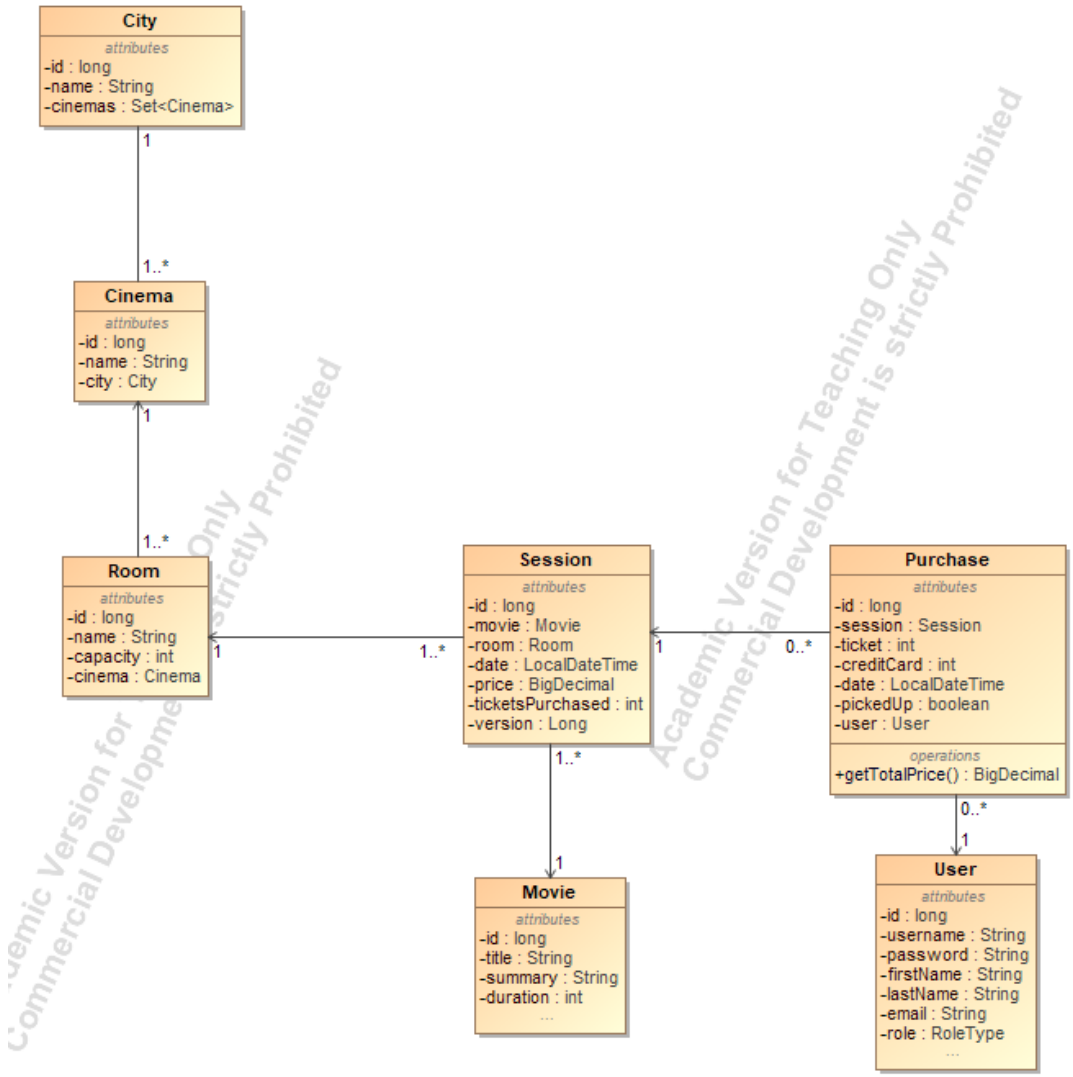
fic facultade de informática da coruña

ÍNDICE

1. BACKEND

1.1 Capa acceso a datos

Para cumplir con el acometido de la práctica, se han creado distintas entidades con sus correspondientes atributos y las siguientes relaciones entre ellas:



Podemos diferenciar siete entidades diferentes:

- **Room:** Contiene todos los atributos que describen las salas de un determinado cine. Entre ellos podemos diferenciar un identificador para cada sala, un nombre, la capacidad máxima que tiene esa sala, y el cine al que pertenece. Como se puede observar en el diagrama, una sala puede albergar varias sesiones.

- **Session:** Esta entidad representa un pase de una determinada película. Los atributos son: un identificador de la propia sesión, la película que se va a emitir, la sala en la que se va a emitir, la fecha completa correspondiente y el precio, que puede ser distinto para cada sesión. Además posee el atributo *ticketsPurchased* que guarda el número de tickets vendidos, para así a través del *SessionDao* poder calcular cuántos tickets quedan disponibles para una determinada sesión. En una sala se pueden realizar múltiples sesiones, como se ha explicado anteriormente. Además, en una sesión se emite únicamente una película, y una compra es de una única sesión.

- **Movie:** Define todos los atributos de la película, en este caso, un identificador único para cada película, el título de la misma, una pequeña sinopsis y la duración, en minutos. Una película se puede emitir en varias sesiones.

- **Purchase:** Almacena los datos relacionados con la compra de una determinada sesión de una película. Cada compra tiene un identificador, la sesión que se va a comprar, el número de entradas que se desea comprar (con un límite máximo de 10 entradas por compra), el usuario que realiza la compra junto con la tarjeta de crédito con la que va a realizar la compra, la fecha completa en la que se hizo la compra y por último, el atributo booleano *pickedUp* que da información acerca de si las entradas han sido recogidas (true) o no (false) por el usuario. Además esta entidad tiene un método *@Transient* para poder calcular el precio total de la compra, que se calcula multiplicando el número total de entradas a comprar por el precio de cada entrada.

  Esta entidad tiene relación 1-N tanto con la entidad *Session*, como con la entidad *User*, ya que una compra siempre la realiza un usuario y es de una determinada sesión.

- **User:** Comprende los datos relacionados con el perfil de usuario que interactúa con la aplicación web. Contiene un identificador propio para cada usuario, un *username* con su correspondiente contraseña, así como el nombre completo y un correo electrónico. A su vez un usuario puede tener diferentes roles, se tiene el rol "espectador" que hace referencia a todos aquellos usuario que utilizan la aplicación para comprar entradas de una película, y el rol "taquillero" que es la persona encargada de la parte administrativa, como por ejemplo entregar las entradas al cliente. Respecto

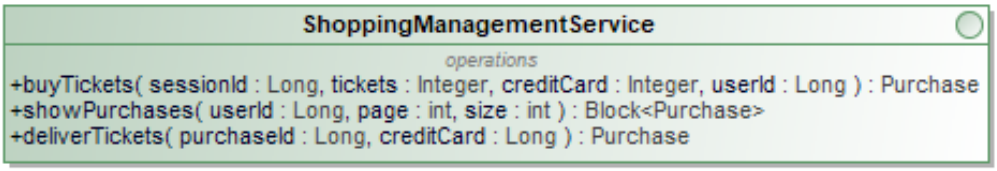
a las relaciones de entidades, un mismo usuario puede realizar varias compras.

Se ha implementado la parte opcional: Selección de cine. Para este desarrollo se implementaron las siguientes entidades:

- **City:** Hace referencia a las diferentes ciudades donde se pueden encontrar los cines. Tiene como atributos un identificador propio de cada ciudad, el nombre de la misma y los cines que contiene. Cada ciudad puede tener N cines. Además esta entidad tiene un método *@Transient* para poder obtener los cines existentes en cada ciudad.

- **Cinema:** Entidad que guarda los datos de cada cine, el identificador de cada uno de ellos, la ciudad a la que pertenecen y el nombre del cine. La entidad cine está relacionada con las ciudades, es decir, un cine pertenece únicamente a una ciudad, y con las salas, ya que cada cine puede tener de 1 a N salas.

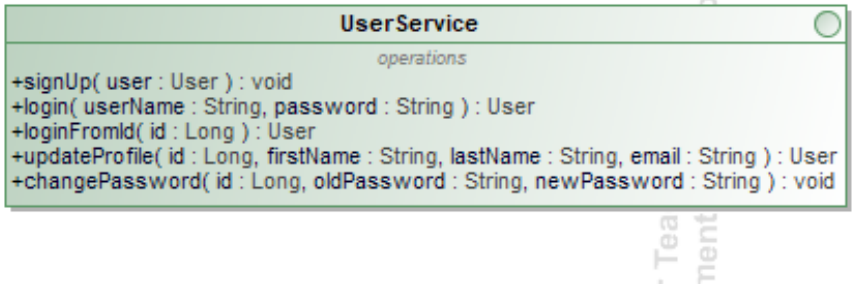
1.2 Capa lógica de negocio



- **ShoppingManagementService:** Interfaz que contiene los casos de uso relacionado con la venta de entradas. Por un lado, un usuario de tipo "espectador" puede adquirir las entradas para una determinada sesión, indicando el número de tarjeta bancaria con la que va a pagar. Por otro lado, se puede obtener una lista con todas las compras realizadas por un usuario. Y por último, un usuario de tipo "taquillero" puede entregar las entradas de una determinada compra de un usuario, siempre y cuando éste le proporcione la tarjeta de crédito con la que pagó.

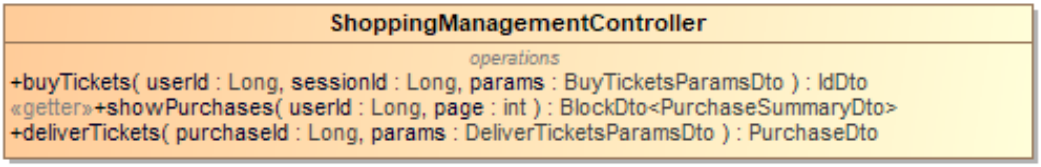
- **BillboardService:** Es la interfaz encargada de las operaciones relacionadas con la cartelera del cine. Cualquier usuario puede ver las películas que existen en ese momento en la cartelera, buscando las sesiones del día de hoy, ordenadas por el título de la película, de tal manera que así sea más sencillo poder mostrarle al usuario las distintas películas con sus respectivas sesiones. Además tiene la posibilidad de seleccionar la fecha de hoy o de cualquiera de los 6 días siguientes. Esta interfaz también nos permite mostrar el detalle de una determinada película o sesión.

  Se incluyen dos métodos de la parte opcional de la práctica para poder mostrar un listado de todas las ciudades existentes y de los cines existentes en cada ciudad.



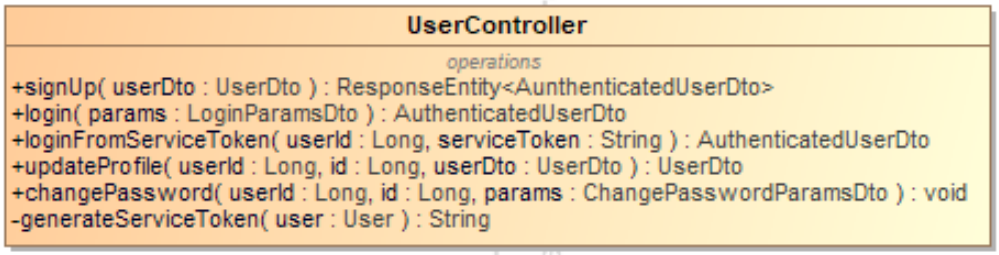
- **UserService:** Interfaz que contiene las distintas operaciones que un usuario puede realizar en la aplicación relacionadas con la identificación y realización de distintas operaciones según los privilegios de cada usuario. Mediante este servicio un usuario no autenticado se puede registrar como espectador. Si ya está registrado, tanto si es espectador como si es taquillero, puede cambiar su contraseña, para una mayor seguridad en la aplicación, y su información, aportando el nombre completo del usuario y el email con el que se creó la cuenta.

1.3 Capa servicios REST



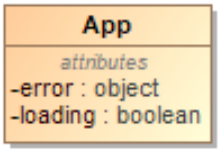
- **ShoppingManagementController:** A través de este controlado se tratan todas las operaciones relacionadas con la compra y entrega de entradas. Por ello, un usuario puede comprar las entradas de una determinada sesión, restringiendo la compra a como máximo 10 entradas por usuario. Una vez realizada la compra, la aplicación le devolverá al cliente un identificador de compra.
  Por otro lado, también es posible recuperar todas las compras de un usuario. Para poder mostrarle los atributos correctos al usuario, se ha creado un *PurchaseSummaryDto* que devuelve la fecha de compra, la fecha completa de la sesión, el título de la película, el número de entradas compradas y el precio total, así el cliente puede tener un resumen de su compra.
  Y por último, a través de este controlador un usuario de tipo "taquillero" entrega las entradas de una compra. En esta operación se devuelve un *PurchaseDto* con los atributos más significativos para el usuario.

- **BillboardController:** Es el controlador encargado de todas las operaciones relacionadas con la cartelera que puede ver cualquier usuario independientemente de su rol. Tenemos dos métodos para poder mostrar las ciudades con sus correspondientes cines, facilitando al usuario la búsqueda de su cine. Cualquier usuario de la aplicación puede ver la cartelera que existe en ese momento y las sesiones a las que aún puede asistir. Por último, los dos últimos métodos nos permiten visualizar los detalles de una determinada película y una determinada sesión, devolviendo en ambos casos un DTO específico para dar la mejor información al usuario.

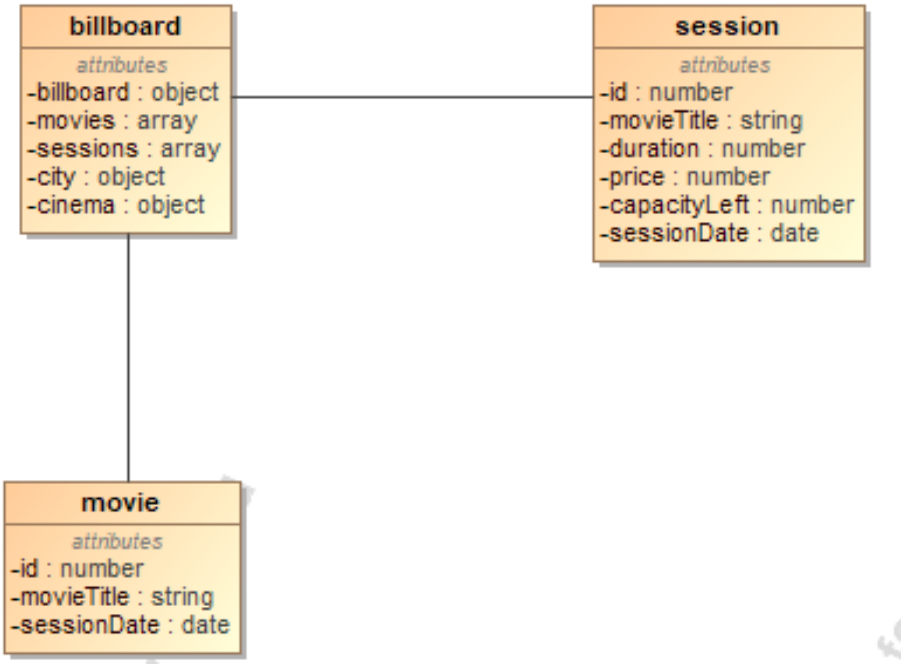


- **UserController:** Controlador relacionado con las operaciones del usuario será el encargado de realizar las conversiones necesarias de código *JSON* a *Dto* y viceversa que permitan completar las operaciones de creación y modificación de los datos asociados a una cuenta de usuario. Este módulo también se encarga de la generación del *token JWT*, que permite identificar de manera unívoca a un usuario aportando la seguridad que se necesita a la aplicación, pues el *token* es un componente que no puede ser simulado por una persona que intente acceder a la aplicación de manera malintencionada para suplantar una identidad.

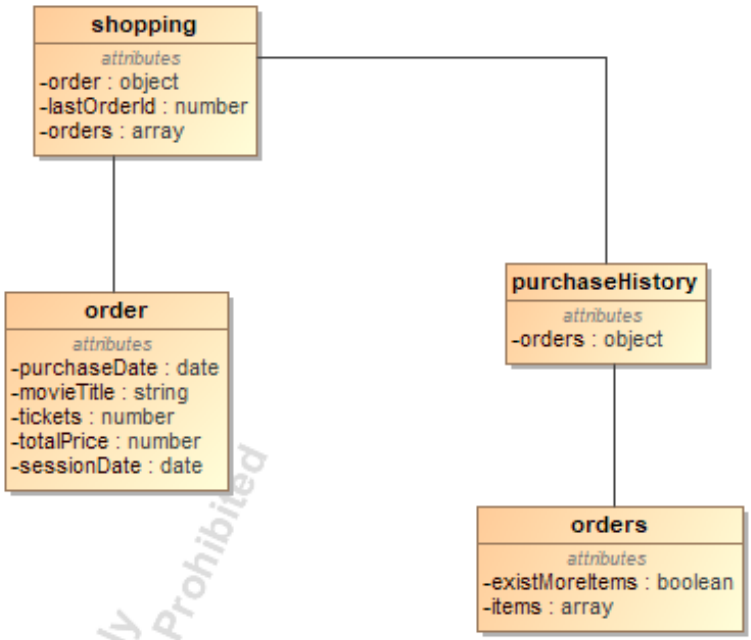
2. FRONTEND



- **App**: Este módulo permite visualizar los componentes principales de nuestra página web.



- **Billboard:** Este módulo contiene diferentes componentes de la interfaz de la aplicación que realiza varios casos de uso implementados. Por un lado seleccionando el día que queramos se ejecuta el caso de uso de ver cartelera, permitiendo ver las películas con sus respectivas sesiones. Una vez visualizado la cartelera el usuario podrá visualizar el detalle de una película o una sesión en concreto, con lo que la página se renderizará mostrándose el componente de cada detalle.



- **<u>Shopping:</u>** Este módulo de la capa frontal de la aplicación tiene como acometido encapsular los diferentes componentes que hacen posible la compra de entradas. Esta operación solo es posible realizarla si el espectador se encuentra *loggeado*, de tal manear que aportando el número de entradas a comprar, e introduciendo una tarjeta de crédito válida podrá realizar una compra adecuada.

  Por otro lado, en cabecera de la aplicación, si se está, se podrá visualizar el botón "Historial de compras", el cual permitirá ejecutar el caso de uso de visualizar el histórico de compras de un usuario, visualizando primero las compras más recientes. Por defecto en la aplicación se muestran 10 productos, por lo que en el caso de que haya más añadidos en la base de datos, deberán ser mostrados en distintas páginas, por lo que en la parte inferior de la página se encuentran los botones "Anterior" y "Siguiente", que forman el componente de los resultados de la búsqueda de una compra y estarán activados o desactivados para su uso en función de si existen o no más compras.

  Por último, este módulo gestiona el caso de uso de entregar las entradas de una compra. En la interfaz visualizamos el botón "Entregar entradas", de tal manera que solo el perfil que posea el rol "taquillero" podrá acceder a él. Introduciendo en el formulario el identificador de la compra y la tarjeta de crédito, podrán ser entregadas al usuario final correctamente.

- **User:** Módulo que contiene los diferentes estados de los componentes asociados a los perfiles de los usuarios creados en el sistemas. En este módulo encontramos los botones para poder mostrar el menú desplegable de la parte derecha de la cabecera mediante el cual se puede actualizar el perfil o cambiar la contraseña de una cuenta. También nos da la posibilidad de registrar nuevos usuarios y autenticarse una vez creado.

3. TRABAJOS TUTELADOS

3.1 Backend: BatchSize

Tras analizar y entender el problema de las n+1 consultas, se observó que esta aplicación se ve afectada por él cuando se desea visualizar el listado de las películas en cartelera. De esta manera, tal y como tenemos desarrollado en esta práctica el *backend* necesitaríamos realizar una consulta para recuperar todas las sesiones, y una consulta adicional por cada sesión de película para poder obtener el título de la película que se quiere visualizar en la cartelera.

En el caso de consultar el histórico de compras llevadas por un espectador, también incurre en este problema, ya que se desea mostrar el título de la película, su respectiva sesión y, como se ha implementado la parte opcional, el nombre del cine para el que se compraron las entradas. De esta forma, se necesitaría realizar una consulta para recuperar todas las compras llevadas por el espectador, una consulta adicional por cada compra para poder obtener el título de la película, otra consulta adicional por cada compra para obtener la sesión, otra consulta adicional por cada compra para obtener el identificador de cada sala de un determinado cine, y otra consulta adicional por cada compra para recuperar el nombre del cine.

Para resolver dicho problema existen diferentes posibilidades, entre las cuales se plantea la modificación de tipo de ejecución, cambiando el parámetro *FetchType* de *Lazy* a *Eager*. Esta solución no es la más correcta ni la más óptima ya que puede ocasionar "El problema del producto cartesiano", por lo que se ha decidido descartarla.

Finalmente para solucionar el problema se ha optado por utilizar el algoritmo *batch fetching*, que consiste en optimizar el número de consultas realizadas. Realizando un único SELECT se obtiene, a partir de los proxies, todos los datos de las sesiones y de las compras realizadas sin la necesidad de tener que hacer un SELECT independiente por cada entidad. De esta forma, basándonos en el uso de la entidad persistente y conociendo la existencia de varias instancias en la misma, a través de este algoritmo es posible realizar esta optimización del número de consultas.

Para llevar a cabo esta solución se utiliza la anotación @org.hibernate.annotations.BatchSize en la entidad *Movies*, para solucinar el problema en la visualización de la cartelera, indicando el tamaño de entidades que se quieren cargar en el primer SELECT que se produzca al visualizar la cartelera.

Esta solución a menudo se llama *blind-guess optimization* porque se desconoce el número de *proxies* no inicializados de las entidades. Por tanto vamos a suponer un tamaño de 10 porque tampoco queremos cargar demasiados datos en memoria sobre todo si no estamos seguros de si los vamos a necesitar.

Para solucionar este problema en el histórico de compras, también utilizaremos esta anotación en las siguientes entidades: *Cinema, Session y Room*. Analizando este caso de uso individualmente también se anotaría en la entidad *Movie* pero ya lo añadimos en el caso de uso anterior. Indicamos el tamaño de entidades que se desean cargar en el primer SELECT al realizar el historial de compras, en este caso también determinamos una cantidad de 10 en todas las entidades, ya que es el número de compras que podemos ver por página.

Esta solución es óptima ya que en lugar de tener n+1 consultas tendremos n+1/10 consultas, reduciendo en una cantidad significativa las consultas y mejorando por ello el rendimiento de la aplicación.

4. PROBLEMAS CONOCIDOS

1. Backlink del detalle de las películas no funciona correctamente.
2. Parte opcional de selección de cine y ciudad está incompleta en el frontend (si está implementada en el backend).
3. Si se introduce manualmente en la URL del navegador una sesión ya comenzada o no existente se produce un fallo de comunicación entre el frontend y el backend.