

# Sitio Web de Venta de Entradas de Cines

Programación Avanzada

7 de Junio de 2018

Pablo Rivas Camino ([pablo.rivas.camino@udc.es](mailto:pablo.rivas.camino@udc.es))

Yanko Nión Ferreiro ([yanko.ferreiro@udc.es](mailto:yanko.ferreiro@udc.es))

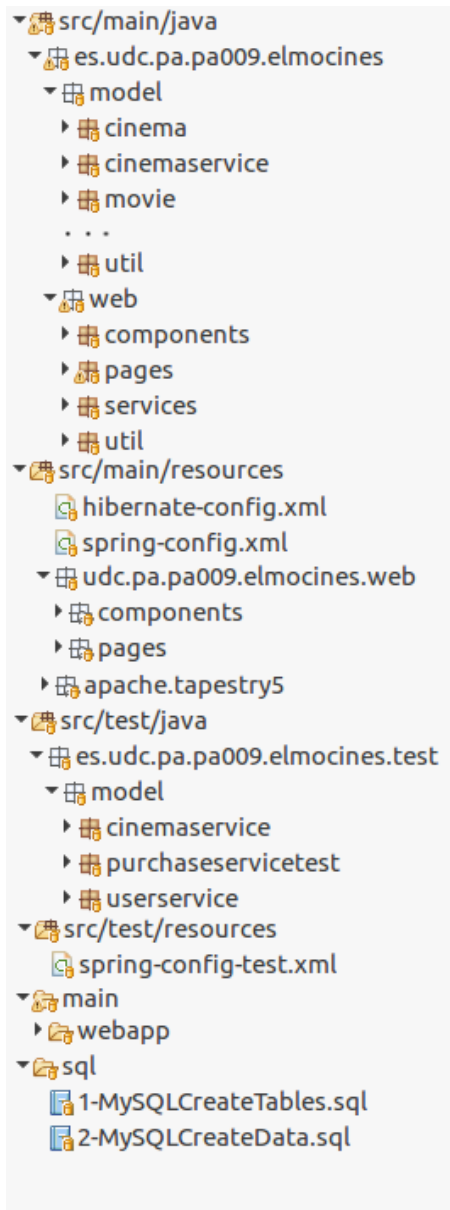
Adrián Gómez García ([a.gomezg@udc.es](mailto:a.gomezg@udc.es))

# Índice

<b>1. Arquitectura Global</b>	<b>2</b>
<b>2. Modelo</b>	<b>3</b>
2.1. Clases persistentes . . . . .	3
2.2. Interfaces de los Servicios Ofrecidos por el Modelo . . . . .	3
2.3. Otros aspectos . . . . .	4
<b>3. Interfaz gráfica</b>	<b>5</b>
<b>4. Trabajos tutelados</b>	<b>9</b>
4.1. AJAX . . . . .	9
4.2. Pruebas de Integración con Selenium . . . . .	9

# 1. Arquitectura Global

La estructura de directorios empleada a la hora de realizar la práctica es la siguiente, dividiéndose en una capa correspondiente al modelo y otra al servicio web:



- **/src/main/java/es.udc.pa.pa009.elmocines.model:** carpeta donde está contenido el código referente al modelo incluyendo tanto entidades, como DAOs, como servicios. Por una parte, cada entidad está almacenada junto con su DAO e implementación en un paquete único. Por otra, los tres servicios están almacenadas en carpetas independientes con sus respectivas implementaciones y excepciones.
- **es.udc.pa.pa009.elmocines.web:** carpeta donde se encuentran almacenados los archivos java que gestionan los eventos lanzados por la capa web y los servicios autenticación, filtros, etc.
- **src/main/resources:** carpeta donde se encuentran los ficheros de configuración de *Spring* e *Hibernate*; las plantillas de las páginas, componentes y mensajes de la capa web; y los mensajes de error proporcionados por el core de *Tapestry* internacionalizados (dentro del subpaquete *org.apache.tapestry5*).
- **/src/test/java:** se ubica el código .java de las pruebas contra el modelo y la interface web de los servicios usados en la práctica.
- **/src/test/resources:** se ubica el fichero de configuración de *Spring* con las variaciones necesarias a la hora de ejecutar las pruebas de unitarias y de integración de la aplicación.
- **/src/main/webapp/WEB-INF:** se encuentra el fichero *web.xml* con configuración de la aplicación y la internacionalización global de la aplicación.
- **/src/sql:** carpeta donde están almacenados los scripts sql para la creación de tablas e inserción de datos.

Figura 1: Estructura de paquetes empleada en la práctica.

## 2. Modelo

En esta sección comentamos los aspectos relevantes a cerca de la capa modelo.

### 2.1. Clases persistentes

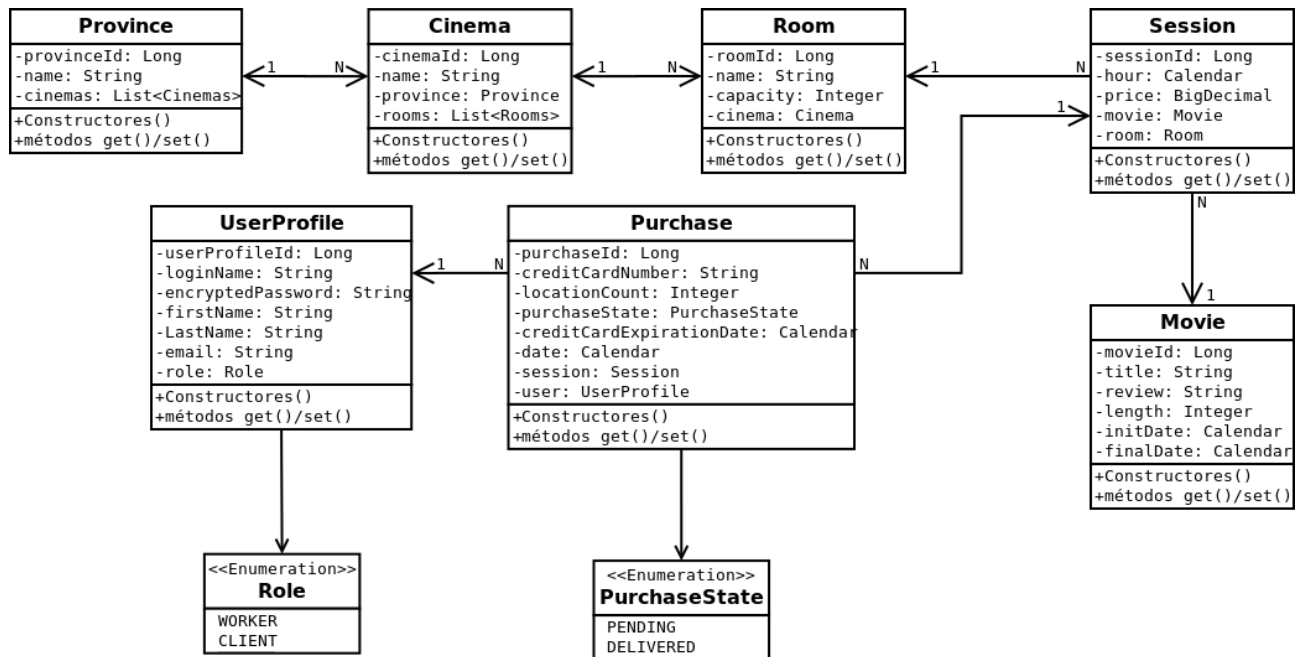


Figura 2: Diagrama de clases persistentes con sus relaciones

### 2.2. Interfaces de los Servicios Ofrecidos por el Modelo

Para realizar la práctica hemos considerado agrupar los casos de uso en tres servicios diferentes:

- **UserService:** contiene todas las operaciones de registro, autenticación y gestión de perfiles de usuario.

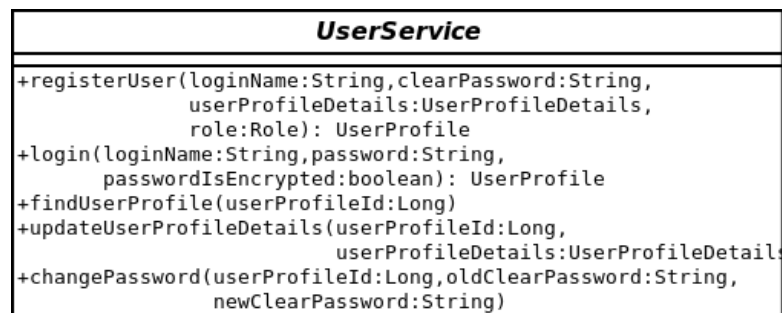


Figura 3: Interfaz del servicio UserService

- **CinemaService:** agrupa los casos de uso dedicados a obtener las características de los cines y de las sesiones de los cines de la cadena.

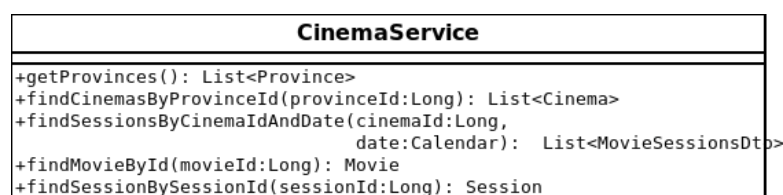


Figura 4: Interfaz del servicio CinemaService

- **PurchaseService:** agrupa los casos de uso de tratamiento de entradas.

PurchaseService
<pre> +purchaseTickets(sessionId:Long,locationsAmount:int,                   sessionId:Long,date:Calendar,                   creditCardNumber:BigDecimal,                   creditCardExpirationDate:Calendar): Purchase +getPurchases(userId:Long,startIndex:int,                count:int): Block&lt;Purchase&gt; +getPurchase(purchaseId:Long): Purchase +collectTickets(purchaseId:Long): Purchase </pre>

Figura 5: Interfaz del servicio *PurchaseService*

## 2.3. Otros aspectos

- Hemos decidido incorporar un DTO de películas con sus sesiones asociadas para que la interfaz web pueda recuperar los datos de las películas con sus sesiones asociadas de manera más sencilla.

MovieDto
<pre> -movie: Movie -sessions: List&lt;Session&gt; +Constructores () +métodos get () / set () </pre>

Figura 6: DTO de las películas con sus sesiones asociadas

- Respecto a las excepciones, hemos decidido añadir al modelo las siguientes dentro de los dos nuevos servicios implementados (*CinemaService* y *PurchaseService*):
  - **InputValidationException:** lanzada cuando falla una validación de los datos.
  - **ExpiredDateException:** si se intenta comprar un ticket de una sesión que ya terminó.
  - **TicketsAlreadyCollectedException:** al tratar de entregar tickets que ya han sido entregados.
  - **TooManyLocationsException:** si se intenta comprar más de 10 tickets para una misma sesión.
- Todas las relaciones han sido marcadas como *LAZY* para evitar recuperar los objetos asociados a las entidades en los casos que no sea necesario.

### 3. Interfaz gráfica

A continuación se muestra en imágenes el comportamiento de la interfaz al ejecutar algunos casos de uso:



Figura 7: *Página principal de la aplicación*


A Coruña
Espacio Coruña
Buscar cartelera
Autenticarse

Sesiones de Espacio Coruña

★ Cine Favorito

Fecha: 08-jun-2018

Película



El Recolector de Basura

01:46
21:47
23:47



Tecnología Electrónica, El Regreso

02:46
19:47
23:47

Figura 8: *Página de visualización de cartelera*


A Coruña
Espacio Coruña
Buscar cartelera
Cliente

Comprar entradas para El Recolector de Basura



Cine:  
Espacio Coruña  
Sala:  
Sala Elmo  
Sesión:  
08-jun-2018 - 01:46  
Duración:  
150 min.

Precio:  
7€  
Asientos Disponibles:  
100

Nº  
Tarjeta  
Crédito  
Fecha  
Caducidad  
Tarjeta  
Asientos - 0 +  

Comprar Entradas

Elmocines - pa009 - Universidad de A Coruña

Figura 9: *Página de visualización de los detalles de una sesión*





<div>  <div> <div>A Coruña</div> <div>Espacio Coruña</div> <div>  <div>Buscar cartelera</div> </div> </div> <div>Ciente</div> </div>						
Fecha Compra	Película	Cine	Asientos	Precio	Fecha Sesión	Sesión
07-jun-2018	El club de los objetos muertos	Espacio Coruña	3	7.3	13-jun-2018	23:47
07-jun-2018	Instanciar a un Ruiseñor	Espacio Coruña	3	6	12-jun-2018	22:47
07-jun-2018	Tecnología Electrónica, El Regreso	Espacio Coruña	3	7	11-jun-2018	21:47
07-jun-2018	Segmentation Fault Core Dumped	Espacio Coruña	3	7	10-jun-2018	20:47
07-jun-2018	Lo que la concurrencia se llevó	Espacio Coruña	3	7.5	09-jun-2018	19:47
07-jun-2018	Tecnología Electrónica, El Regreso	Espacio Coruña	3	7	08-jun-2018	23:47
07-jun-2018	El Recolector de Basura	Espacio Coruña	3	7	08-jun-2018	23:47
07-jun-2018	La luz es mi fuerza	Espacio Coruña	3	7.5	08-jun-2018	23:47
07-jun-2018	El Recolector de Basura	Espacio Coruña	3	8.5	08-jun-2018	21:47
07-jun-2018	El club de los objetos muertos	Espacio Coruña	3	7.3	08-jun-2018	21:47
<div> <div>← Anterior</div> <div>Siguiente →</div> </div>						
<div>Elmocines - pa009 - Universidad de A Coruña</div> 						


Figura 10: *Página de visualización las compras de un usuario*





A Coruña ▾

Espacio Coruña ▾


Buscar cartelera


Taquillero ▾

## Buscar una compra

Identificador de la compra:

Buscar

## Detalles de la compra



**Cine:**  
Espacio Coruña

**Sala:**  
Sala Elmo

**Título de la película:**  
El club de los objetos muertos

**Sesión:**  
08-jun-2018 - 21:47

**Localidades compradas:**  
3

**Fecha de la compra:**  
07-jun-2018 - 19:47

**Tarjeta de crédito:**  
1234567AB

**Status:**  
PENDING

Entregar entradas >

Elmocines - pa009 - Universidad de A Coruña

Figura 11: *Página de visualización una compra para entregarla*

Respecto a la capa web hay que tener en cuenta las siguientes consideraciones:

- **Cookies:** para poder guardar el cine favorito tanto si se está autenticado en el sistema como que no, ha sido necesario añadir una cookie “*favouriteCinema*” guardada cuando este se establece que almacene el id del cine favorito seleccionado.
- **Control de usuarios:** como no todos los usuarios pueden acceder a las diferentes páginas de la aplicación, ha sido necesario crear dos nuevos roles “*CLIENT*” y “*WORKER*” en *UserProfile*, teniendo que añadirlos al enumerado *AuthenticationPolicyType* para su posterior implementación en el servicio *Authentication-Validator* pudiendo así usarlos con la etiqueta *@AuthenticationPolicy* en los ficheros java de las páginas en las que queramos controlar el acceso. A mayores, ha sido necesario incluir un tercer campo “role” en la clase *UserSession* para no tener que andar recuperándolo de la base de datos constantemente cuando queramos procesar una política de control.
- **Open Session in View:** al estar todas las relaciones del modelo marcadas como *LAZY*, no vamos a poder inicializar los proxies al haber finalizado la transacción en *Hibernate* por lo que para extender el tiempo de la conexión, hemos tenido que implementar este mecanismo para poder recuperar los objetos desde la parte web.
- **Internacionalización:** se ha implementado la internacionalización de mensajes y fechas en los idiomas inglés, gallego y castellano.

8

## 4. Trabajos tutelados

### 4.1. AJAX

Con el fin de optimizar el sistema recargando parcialmente las páginas y por tanto hacer menos llamadas a los servicios y a la base de datos, se ha decidido implementar AJAX mediante el uso de zonas en los siguientes casos:

- **Selección de cines tras seleccionar una provincia:** se ha añadido un componente zone a la plantilla tml que limite la región del selector de cines para que cuando el valor del primer selector de provincias cambie, el de cines actualice sus valores acordes a la provincia seleccionada en el selector de provincias.
- **Selección de fechas en la cartelera:** tanto en la página principal en caso de que se haya seleccionado un cine favorito como en la página de visualizar la cartelera de un cine, se ha añadido a una zona todas las películas del día y sus sesiones para que cuando cambie el valor del selector de fecha ofrecido en la página, estas se actualicen acordes a la fecha seleccionada.
- **Selección de cine favorito:** para no tener que recargar toda la página una vez se pincha en el botón de “Cine Favorito” dentro de la página de visualizar cartelera de un cine, hemos decidido añadir una zona en este botón para que actualice su estado sin alterar el resto de la página.
- **Aumentar y disminuir número de localidades en la compra:** como muchas veces no va a interesar meter el número de localidades que se desean comprar directamente en formato numérico, dentro de la página de comprar una entrada se ha decidido implementar unos botones de aumento y decremento del número de localidades englobados dentro de una zona para que nuevamente, el cambio de este valor no requiera que se recargue toda la página con todos los mismos datos una y otra vez.

### 4.2. Pruebas de Integración con Selenium

Para la implementación de los casos de prueba funcionales contra la interface web se ha utilizado el componente *webDriver* de la herramienta *Selenium* para su automatización. Para su correcta utilización hemos definido una base de datos “*pojowebtest*” contra las que se ejecutarán dichas pruebas en adicción a un perfil en maven “*webtest*” para que se puedan ejecutar las pruebas del modelo con la ejecución por defecto y las pruebas contra la interface web con la ejecución con este nuevo perfil con el comando “*mvn -Pwebtest test*”. Para ello hemos tenido que redefinir el plugin de maven-surefire en ambos perfiles y configurar sus datasources en el fichero de configuración de maven. Para que estas pruebas abran un navegador y se ejecuten de manera automática en el mismo, ha sido necesario añadir también una propiedad “*geckodriver*” con su fichero correspondiente al sistema operativo en el que se estén ejecutando las pruebas.

Para este apartado se ha decidido implementar tres casos de prueba dentro del paquete *es.udc.pa.pa009.elmocines.test.web*:

- **authenticationTest:** implementa el caso de uso de autenticación de usuarios. Desde la página principal de la aplicación, se inicia sesión con el usuario “client” previamente registrado en el sistema, pulsando en el botón de autenticarse. Se rellena el formulario de autenticación con el nombre y la contraseña de dicho usuario y tras la redirección a la página principal se comprueba que el saludo sea su nombre correspondiente.
- **authenticationWithRegisterTest:** implementa de nuevo el caso de uso de autenticación de usuarios pero con un registro previo. Desde la página principal se accede al formulario de autenticación mediante el botón de autenticarse y desde ahí se accede al formulario de registro. Una vez en este punto, se introducen los datos de un usuario de prueba y tras pulsar en el botón de envío se comprueba que el saludo sea el nombre correspondiente al usuario creado.
- **purchaseTicketTest:** implementa el caso de uso de realizar una compra de una entrada sin estar previamente autenticado en el sistema. Desde la página principal se busca el primer cine correspondiente a “Espacio Coruña” y desde la página de la cartelera se establece como cine favorito. Se vuelve a la página principal mediante el link de la barra superior correspondiente al layout de la página y desde ahí se comprueba que el nombre del cine que aparece como favorito sea el de “Espacio Coruña”. Una vez en este punto se accede a la primera sesión que ofrece el cine y se pulsa en el botón de autenticarse que nos redirige al formulario de autenticación. Ponemos los datos del usuario “client” previamente registrado en la base de datos de prueba y tras darle al botón de envío se rellenan los campos necesarios para realizar la compra y se pulsa en el botón de comprar entrada. Se accede a la página de ver mis compras y se busca la compra con id 22 y se mira que el nombre de la película coincida al nombre de la película de la sesión que se ha comprado.