



FilmsApp. Aplicación web con Spring Boot y Angular

Daniel Salamanca Calderón
Grado Ingeniería Informática
Java EE

Albert Grau Perisé
Santi Caballe Llobet

01/2017



Aquesta obra està subjecta a una llicència de [Reconeixement-NoComercial-SenseObraDerivada 3.0 Espanya de Creative Commons](https://creativecommons.org/licenses/by-nc-nd/3.0/es/)

FITXA DEL TREBALL FINAL

Títol del treball:	<i>FilmsApp. Aplicación web con Spring Boot y Angular</i>
Nom de l'autor:	<i>Daniel Salamanca Calderón</i>
Nom del consultor/a:	<i>Albert Grau Perisé</i>
Nom del PRA:	<i>Santi Caballe Llobet</i>
Data de lliurament (mm/aaaa):	<i>01/2017</i>
Titulació o programa:	<i>Grado de Ingeniería Informática</i>
Àrea del Treball Final:	<i>Java EE</i>
Idioma del treball:	<i>Castellano</i>
Paraules clau	<i>SpringBoot, Angular y REST</i>
<p>Resum del Treball (màxim 250 paraules): <i>Amb la finalitat, context d'aplicació, metodologia, resultats i conclusions del treball</i></p>	
<p>La finalidad de este proyecto sobre tecnologías Java EE es aumentar los conocimientos sobre este tema. Para ellos se ha creado una aplicación web basada principalmente en Spring Boot y Angular, dos tecnologías cada vez más utilizadas.</p> <p>Debido a que cada vez se utiliza más internet para ver películas o series y que el número de personas que van al cine va disminuyendo se ha creado una aplicación web con la que los usuarios puedan ver películas online y beneficiarse de descuentos o promociones en los cines.</p> <p>Para el desarrollo del proyecto se han seguido diferentes fases. Primero pensó una planificación y se empezó con la fase de análisis de funcionalidades y requisitos. A continuación se realizó el diseño de la aplicación. Después de esto se implementó la aplicación y finalmente se hicieron pruebas y la documentación pertinente del proyecto. Durante todo el desarrollo, se ha tenido que volver al análisis de funcionalidades para realizar cambios, así como cambios en el diseño, provocados por dificultades encontradas durante la implementación de la aplicación.</p> <p>El resultado obtenido ha sido el esperado. Pese a las dificultades encontradas se ha conseguido desarrollar una aplicación web parecida a lo que se había pensado al inicio.</p> <p>Como conclusión, puedo decir que realizar el TFG me ha hecho aprender muchos conocimientos nuevos así como ampliar otros. Por otra parte, he tenido la oportunidad de desempeñar funciones que no había hecho antes como puede ser el requisito de funcionalidades o la planificación, es decir, la parte previa al desarrollo y he podido comprobar que es imprescindible para implementar la aplicación de forma correcta.</p>	

Abstract (in English, 250 words or less):

The application that I has been created allows users to watch movies online and benefit from discounts or promotions in cinemas. This application has been developed mainly with Spring Boot and Angular. This documentation serves to explain the phases of the project that have been done to develop the web application.

The memory first explains the purpose of the project and shows the planning. To continius, it details aspects of the project specification. You can read about the functionalities that the application has, the different roles it has and there is information about functional and non-functional requirements. You can also find a diagram of use cases with his specification. And the conceptual model.

On the other hand, the document specifies the design and architecture of the application as well as the technologies used. Finally, appears a conclusion of the project and a small manual about how run and use the web application.

In conclusion, users could watch a lot of movies from their homes and enjoy numerous discounts in cinemas. I think this could attract more people to the cinemas. So, the cinemas and the users could obtain benefit.

Índice

1. Introducción.	5
1.1. Objetivos del TFG.	6
1.2. Métodos seguidos.	7
1.3. Planificación.	8
1.4. Resultado final.	9
1.5. Contenido.	10
2. Especificaciones del proyecto.	11
2.1. Introducción.	11
2.2. Funcionalidades principales.	12
2.2.1. Funcionalidades principales de usuarios sin registrar.	12
2.2.2. Funcionalidades principales de usuarios registrados.	12
2.2.3. Funcionalidades principales de los cines registrados.	12
2.2.4. Funcionalidades principales de los administradores.	12
2.3. Observaciones.	13
2.4. Roles del sistema.	14
2.5. Requerimientos funcionales.	15
2.6. Requerimientos no funcionales.	17
2.7. Diagrama de casos de uso.	18
2.8. Especificación de casos de uso.	19
2.8.1. Casos de uso: gestión de usuarios.	19
2.8.2. Casos de uso: gestión de login.	23
2.8.3. Casos de uso: gestión de ofertas.	24
2.8.4. Casos de uso: gestión de películas.	28
2.8.5. Casos de uso: gestión de otros campos.	32
2.9. Modelo conceptual.	34
3. Diseño.	35
3.1. Arquitectura JPA.	37
3.2. Diseño de la REST.	39
3.3. Capa de presentación.	41
4. Entorno de trabajo.	47
5. Mejoras de versiones futuras.	48
6. Conclusiones.	49
7. Manual aplicación.	50
8. Bibliografía.	58

1. Introducción.

En los últimos años, se ha podido observar que cada vez se utiliza más internet para ver películas o series, ya que se puede elegir entre muchas opciones y en el momento que uno quiera. Además, aunque sigue habiendo muchas personas que van habitualmente al cine, el número va disminuyendo debido a estos nuevos medios para ver películas y a que cada vez es más caro ir al cine.

Por tanto, he querido crear una aplicación web con la que los usuarios puedan ver películas online y beneficiarse de descuentos o promociones en los cines. De esta forma, los cines, que también podrán utilizar la aplicación, conseguirán atraer más gente.

La aplicación ha sido desarrollada sobre tecnologías de Java EE con diferentes frameworks que actualmente se están utilizando más para este tipo de aplicaciones, como Spring Boot o AngularJS. La aplicación sigue un modelo de tres capas: presentación, servicio y datos.

Decidí utilizar estas tecnologías, ya que no he tenido la oportunidad de trabajar a fondo con ellas en el ámbito profesional, solo trabajos puntuales, y me parecieron interesantes. También, hay que decir que nunca había hecho una aplicación desde el inicio, con una planificación, un estudio de requerimientos funcionales, crear la base de datos, etc. Gracias al TFG he podido realizar todo esto.

Pese a los problemas que me han surgido durante el desarrollo, creo que sido capaz de ir adaptando y aprendiendo nuevos conocimientos a fin de cumplir, más o menos, con las especificaciones iniciales. Algunas partes se han tenido que modificar o suprimir por dificultad aunque puedo decir que estoy satisfecho del resultado final y con todo lo que he aprendido durante estos meses sobre el desarrollo propio de la aplicación, las tecnologías utilizadas y las tareas necesarias que se realizan antes, durante y después del desarrollo.

1.1. Objetivos del TFG.

El objetivo principal del trabajo era aprender a realizar el análisis, diseño y desarrollo de una aplicación web mediante tecnologías Java EE.

Una vez decidido el tema del TFG, los nuevos objetivos que surgieron fueron aprender a utilizar las tecnologías Spring Boot y AngularJS e implementar una aplicación que permitiese a los usuarios ver películas por internet y obtener ventajas en los cines.

1.2. Métodos seguidos.

Para el desarrollo del TFG se han realizado diferentes fases. Lo primero de todo fue pensar sobre qué hacer la aplicación. A partir de ahí, se han seguido las fases habituales. Se pensó una planificación y se empezó con una primera fase de análisis de funcionalidades y requisitos. A continuación se realizó el diseño de la aplicación. Después de esto se implementó la aplicación y finalmente se hicieron pruebas.

Durante todo el desarrollo, se ha tenido que volver al análisis de funcionalidades para realizar cambios, así como cambios en el diseño, provocados por dificultades encontradas durante la implementación de la aplicación.

Por otra parte, todo esto se ha tenido que compaginar con el aprendizaje de las tecnologías utilizadas en el proyecto, lo que ha complicado y retrasado el desarrollo de la aplicación.

1.3. Planificación.

La planificación ha sido pensada para cumplir con las fechas de entrega de las diferentes partes. El tiempo dedicado a cada parte se ha modificado respecto a lo planificado inicialmente debido a dificultades encontradas.

- PAC 1:
 - Del 26/09/2016 al 05/10/2016
 - Duración total: 9 días.
 - Tareas:
 - Elaboración del plan de trabajo. (9 días)
- PAC 2:
 - Del 06/10/2016 al 09/11/2016
 - Duración total: 33 días.
 - Tareas:
 - Análisis del proyecto. (7 días)
 - Especificaciones del proyecto. (8 días)
 - Diseño del proyecto. (14 días)
 - Instalación y preparación del entorno de trabajo. (4 días)
- PAC 3:
 - Del 10/11/2016 al 23/12/2016
 - Duración total: 43 días.
 - Tareas:
 - Implementación. (36)
 - Pruebas. (7)
- PAC 4:
 - Del 24/12/2016 al 12/01/2017
 - Duración total: 19 días.
 - Tareas:
 - Revisión. (12 días)
 - Memoria. (7 días)

1.4. Resultado final.

El resultado final ha sido una aplicación web creada principalmente con Spring Boot y Angular. El proyecto final incluye diferentes elementos:

1. La propia aplicación: todos los ficheros necesarios para poder arrancarla.
2. Un archivo que contiene la estructura de la base de datos y los propios datos para importarla en otro sistema.
3. La memoria del proyecto: donde se explica todas las fases realizadas durante el proyecto y un pequeño manual.
4. Un video explicativo sobre el proyecto y su funcionamiento.

1.5. Contenido.

El contenido de la memoria que viene a continuación es referente al desarrollo de la aplicación.

Se explicarán las especificaciones del proyecto, es decir, los requisitos funcionales, los diferentes actores que utilizan la aplicación, los casos de uso y el modelo conceptual.

Por otra parte, se hará referencia al diseño de la aplicación. Se podrá ver la estructura del proyecto así como explicar la arquitectura de JPA, la REST y el funcionamiento de la capa de presentación.

Por último, se comenta el entorno de trabajo utilizado, posibles mejoras y las conclusiones.

También se ha introducido un pequeño manual para el arranque de la aplicación y un breve comentario sobre la funcionalidad de cada pantalla.

2. Especificaciones del proyecto.

2.1. Introducción.

Como se ha comentado anteriormente, el objetivo era crear una aplicación web con la que los usuarios puedan ver películas online y beneficiarse de descuentos o promociones en los cines. De esta forma, los cines, que también pueden utilizar la aplicación, conseguirán atraer más gente.

Cualquier persona puede acceder a la aplicación para ver la información de las películas que hay disponibles. Hay la posibilidad de realizar una búsqueda de la película que se desea mediante filtros.

Si se quiere ver películas y disfrutar de los descuentos, se deberán registrar pagando una cuota mensual. Un usuario registrado, obtendrá diferente cantidad de puntos cada vez que vaya al cine dependiendo del día y hora, número de entradas y película. Estos puntos los podrá gastar para los descuentos y promociones disponibles.

Los cines registrados en la web, podrán añadir descuentos o promociones que consideren oportunos, así como eliminarlos. Por otra parte, serán los encargados de dar y quitar puntos a los usuarios.

2.2. Funcionalidades principales.

La aplicación tendrá diferentes funcionalidades que se describirán a continuación. Para ellos, se ha decidido dividirlos según los permisos de los usuarios.

2.2.1. Funcionalidades principales de usuarios sin registrar.

Estos usuarios pueden:

- Ver todas las películas disponibles.
- Pueden realizar una búsqueda mediante filtros para las películas.
- Pueden ver la información de las películas.
- Pueden registrarse para disfrutar de las diferentes funcionalidades.

2.2.2. Funcionalidades principales de usuarios registrados.

Una vez registrados, además de las funcionalidades anteriores, pueden:

- Ver las películas disponibles.
- Aprovechar los descuentos y promociones disponibles.
- Modificar sus datos de usuarios.
- Darse de baja.

2.2.3. Funcionalidades principales de los cines registrados.

Los cines registrados tienen las mismas funcionalidades que los usuarios sin registrar y además:

- Aplicar los descuentos y promociones a los usuarios.
- Dar los puntos correspondientes a los usuarios.
- Añadir y eliminar descuentos y promociones.

2.2.4. Funcionalidades principales de los administradores.

Los administradores tienen acceso a todas las funcionalidades de los usuarios registrados. Y también tienen acceso:

- Al mantenimiento completo de las películas.
- Al mantenimiento completo de los directores, actores y categorías.
- Al mantenimiento de los usuarios.
- Al mantenimiento de los cines.

2.3. Observaciones.

Hay que decir, que la funcionalidad de registro, no se ha realizado completamente. Ya que para registrarse hay que realizar el pago correspondiente. Por lo que en este caso, simplemente se elige una de las diferentes tarifas y se hace el alta del usuario. La funcionalidad del pago en sí, no se ha implementado.

Por otro lado, la funcionalidad de ver la película, tampoco se ha hecho por motivos obvios. Aunque sí aparece el botón para ello.

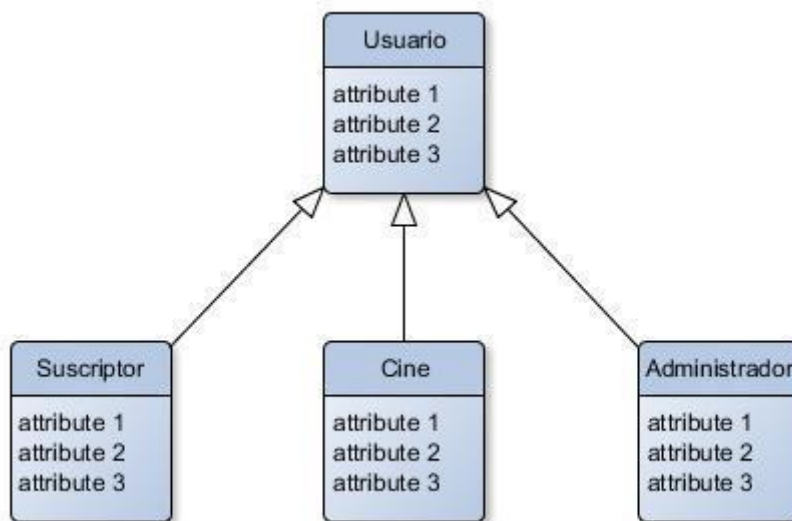
2.4. Roles del sistema.

Como se puede ver en el apartado anterior, los diferentes usuarios no pueden acceder a todas las funcionalidades y ni tampoco deberían ver datos personales de otros usuarios, a menos que sean administradores.

Por tanto, podemos definir como usuario, cualquier persona registrada o no que acceda a la aplicación.

Los usuarios registrados pueden tener tres roles diferentes: suscriptor, cine o administrador.

De esta forma, el diagrama de roles será el siguiente:



2.5. Requerimientos funcionales.

En este punto definimos los requerimientos que tiene el proyecto. A partir de las funcionalidades descritas anteriormente, podemos agrupar los requisitos según su función principal.

- Gestión usuarios:
 - Alta de usuarios:
 - Los usuarios pueden registrarse como nuevos suscriptores mediante un formulario.
 - El administrador, puede dar de alta a un cine o a otro administrador.
 - Los cines no pueden darse de alta por sí solos.
 - Baja de usuarios:
 - Cada usuario puede darse de baja.
 - Los cines solo pueden ser dados de baja por un administrador.
 - Un administrador, también puede dar de baja a un suscriptor.
 - Modificación de datos de usuario:
 - Cada usuario puede ver y modificar sus datos.
 - Un administrador puede ver los datos de cualquier usuario.
- Sistema de login:
 - Login:
 - Un usuario puede acceder a las funcionalidades propias de su rol mediante su nombre de usuario y contraseña, siempre que esté registrado.
 - Logout:
 - Un usuario logeado, puede salir de su cuenta, impidiendo que una persona acceda a la aplicación sin credenciales.
- Gestión de ofertas:
 - Dar de alta una oferta:
 - Un cine puede dar de alta un descuento o promoción.
 - Dar de baja una oferta:
 - Un cine puede dar de baja un descuento o promoción.
 - Aplicar ofertas:
 - Un cine puede aplicar una promoción o descuento a un suscriptor.
 - Los puntos de los suscriptores se suman o se restan directamente al aplicar la oferta.
 - Ver las ofertas publicadas.
 - Un usuario puede ver las ofertas publicadas.

- Gestión de películas:
 - Dar de alta una película:
 - Los administradores pueden dar de alta una película.
 - Dar de baja una película:
 - Los administradores pueden dar de baja una película.
 - Modificar datos de una película:
 - Los administradores pueden modificar los datos de una película.
 - Los usuarios pueden ver los datos de una película.
 - Ver la lista de películas disponibles:
 - Los usuarios pueden ver todas las películas disponibles en la aplicación.
 - Ver películas:
 - Los suscriptores y administradores pueden ver las películas disponibles tantas veces quieran.

- Gestión de otros campos:
 - Dar de alta datos necesarios de la descripción de una película:
 - Un administrador puede añadir un dato necesario como director, actor o categoría.
 - Dar de baja datos necesarios de la descripción de una película:
 - Un administrador puede eliminar un dato necesario como director, actor o categoría.

2.6. Requerimientos no funcionales.

Estos requerimientos también son muy importantes ya que se han de cumplir para que las funcionalidades se pueden utilizar de forma correcta y rápida para que el usuario pueda manejar la aplicación sin problemas.

- Eficiencia:

La aplicación debe poder ejecutarse sin ningún tipo de problemas. Debe realizar las funciones de forma ágil ya que si el usuario ve que va lento, dejará de usarla. Además, los cines deben poder aplicar las ofertas rápidamente para no formar colas en las taquillas.

- Usabilidad e interfaz de usuario:

Los usuarios deben poder desplazarse por las diferentes funcionalidades de forma rápida e intuitiva. La interfaz debe ser clara para no tener que ir probando los diferentes menús hasta encontrar lo que se quiere.

- Seguridad:

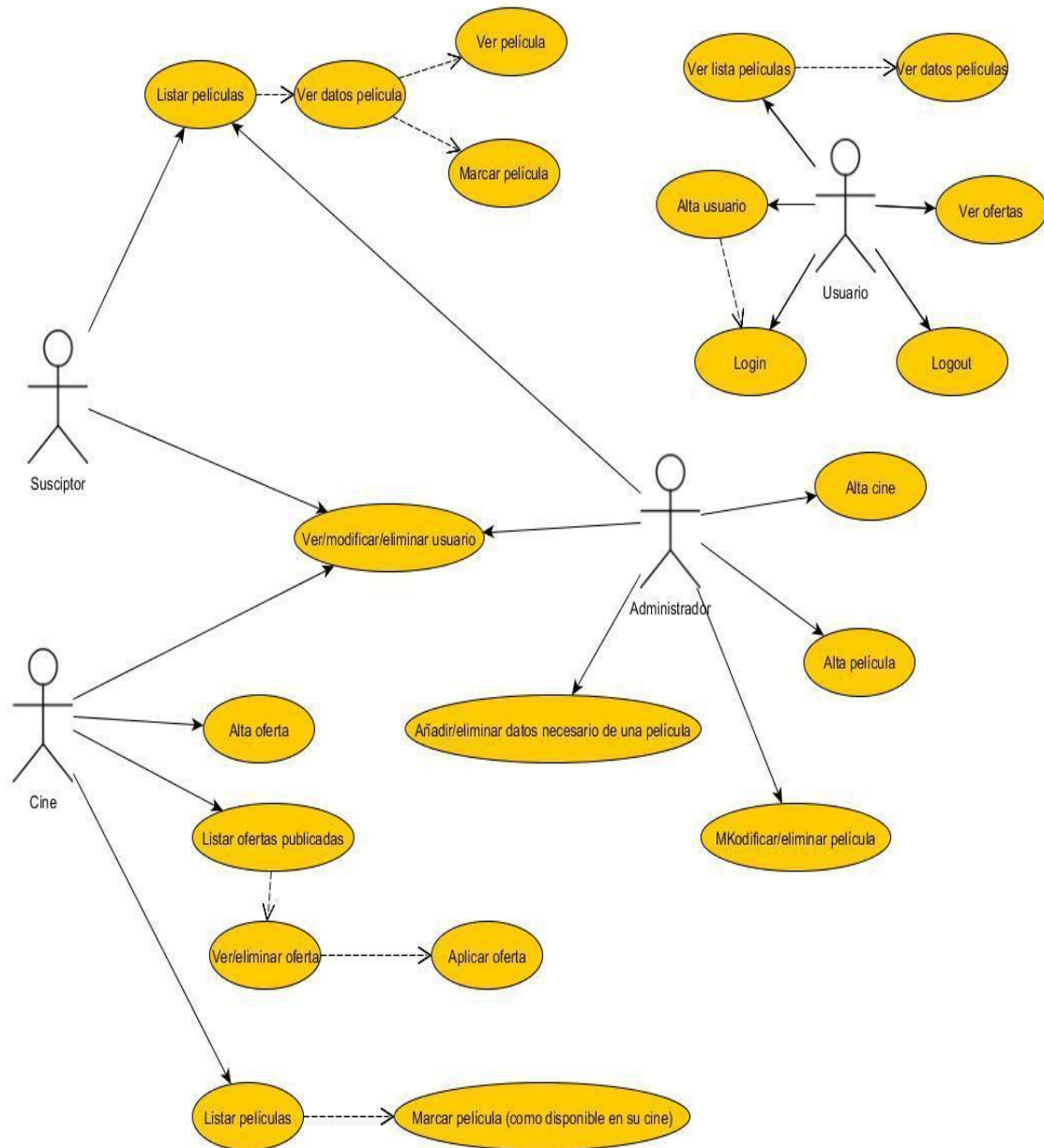
Como en todas las aplicaciones, la seguridad es muy importante. Cada usuario, excepto el administrador, solo debe poder acceder a sus datos personales y a sus funciones. Ya que de otra forma un usuario podría aplicarse ofertas a sí mismo o un usuario no registrado, ver películas de forma gratuita. Por eso se implementará un sistema de login y se dará un tipo de rol a cada usuario para ofrecer las funcionalidades adecuadas a cada uno.

- Escalabilidad:

Es conveniente que la aplicación tenga una buena escalabilidad para poder ampliar las funcionalidades o migrar a otro servidor de forma sencilla, sin que se tengan que cambiar toda la aplicación.

2.7. Diagrama de casos de uso.

En el siguiente diagrama se pueden ver los casos de uso.



2.8. Especificación de casos de uso.

Aquí se explica los diferentes casos de uso de forma más amplia.

2.8.1. Casos de uso: gestión de usuarios.

<i>Alta de un suscriptor al sistema</i>		
<i>Roles</i>	Usuario sin registrar	
<i>Descripción</i>	El usuario accede al sistema para crear una cuenta personal de la aplicación.	
<i>Condición previa</i>	Un usuario sin registrar solo se puede registrar como suscriptor.	
<i>Proceso</i>	Rol	Sistema
	<ol style="list-style-type: none">1. El usuario accede a la aplicación.1. El usuario clica sobre "Registrar".1. El usuario rellena los campos y aprieta "finalizar".	<ol style="list-style-type: none">1. El sistema muestra la página inicial.1. La aplicación muestra un formulario que debe ser rellenado por el usuario.1. El sistema valida los datos y si no hay problema, le redirige a la pantalla inicial de su cuenta.
<i>Alternativas</i>	<ol style="list-style-type: none">1. El sistema detecta un error y muestra un mensaje.	

Alta de un cine al sistema		
<i>Roles</i>	Administrador	
<i>Descripción</i>	El administrador accede al sistema para dar de alta a un cine en la aplicación.	
<i>Condición previa</i>	Solo el administrador puede registrar a un cine y debe haberse logeado.	
<i>Proceso</i>	Rol	Sistema
	<ol style="list-style-type: none"> 1. El administrador accede a la pantalla de mantenimiento de cines. 1. El administrador clicla sobre "Nuevo cine". 1. El administrador rellena los campos y aprieta "finalizar". 	<ol style="list-style-type: none"> 1. El sistema muestra un menú de opciones. 1. La aplicación muestra un formulario que debe ser rellenado por el administrador. 1. El sistema valida los datos y si no hay problema, muestra un mensaje de que el cine se ha creado.
<i>Alternativas</i>	<ol style="list-style-type: none"> 1. El sistema detecta un error y muestra un mensaje. 	

Baja de un usuario del sistema		
<i>Roles</i>	Suscriptor y administrador	
<i>Descripción</i>	El suscriptor o el administrador entra en el sistema para darse de baja.	
<i>Condición previa</i>	Se necesita haberse logeado en el sistema	
<i>Proceso</i>	Rol	Sistema
	<ol style="list-style-type: none"> 1. El usuario accede a su apartado personal. 1. El usuario clica sobre la opción de "Darse de baja". 1. El usuario continúa con el proceso. 	<ol style="list-style-type: none"> 1. El sistema muestra una serie de opciones. 1. El sistema muestra un mensaje de confirmación. 1. El sistema muestra un mensaje y le redirige a la página inicial.
<i>Alternativas</i>	<ol style="list-style-type: none"> 1. El usuario cancela el proceso. 	

Modificación datos de un usuario		
<i>Roles</i>	Cualquier usuario registrado.	
<i>Descripción</i>	Un usuario accede a sus datos personales para realizar cambios.	
<i>Condición previa</i>	El usuario debe estar logeado.	
<i>Proceso</i>	Rol	Sistema
	<ol style="list-style-type: none"> 1. El usuario accede a la pantalla de datos personales. 1. El usuario clica sobre cambiar datos. 1. El usuario cambia los datos que quiera y clica en guardar. 	<ol style="list-style-type: none"> 1. El sistema muestra sus datos. 1. El sistema muestra el formulario con sus datos.
<i>Alternativas</i>		

2.8.2. Casos de uso: gestión de login.

Login de un usuario		
<i>Roles</i>	Todos los usuarios registrados.	
<i>Descripción</i>	Un usuario se logea para acceder a su cuenta.	
<i>Condición previa</i>	El usuario debe estar registrado.	
<i>Proceso</i>	Rol	Sistema
	<ol style="list-style-type: none"> Un usuario entra en la aplicación y aprieta el botón "Login". El usuario rellena el formulario y clicla. 	<ol style="list-style-type: none"> El sistema muestra un formulario. El sistema muestra la pantalla inicial del usuario.
<i>Alternativas</i>	<ol style="list-style-type: none"> El sistema muestra un error de autenticación y vuelve a pedir las credenciales. 	

Logout de un usuario		
<i>Roles</i>	Todos los usuarios registrados.	
<i>Descripción</i>	Un usuario sale de la su cuenta de la aplicación.	
<i>Condición previa</i>	Estar logeado.	
<i>Proceso</i>	Rol	Sistema
	<ol style="list-style-type: none"> El usuario aprieta el botón "Salir". 	<ol style="list-style-type: none"> El sistema le redirige a la pantalla inicial.
<i>Alternativas</i>		

2.8.3. Casos de uso: gestión de ofertas.

<i>Dar de alta una oferta</i>		
<i>Roles</i>	Cine	
<i>Descripción</i>	Un cine da de alta una oferta.	
<i>Condición previa</i>	Estar logeado.	
<i>Proceso</i>	Rol	Sistema
	1. El cine accede a la pantalla de nueva oferta. 1. El cine rellena el formulario y clicla en guardar.	1. El sistema muestra un formulario. 1. El sistema confirma la nueva oferta y le redirige a la pantalla de ofertas.
<i>Alternativas</i>	1. El sistema informa de un error al crear la nueva oferta.	

<i>Dar de baja una oferta</i>		
<i>Roles</i>	Cine	
<i>Descripción</i>	Un cine da de baja una oferta.	
<i>Condición previa</i>	Estar logeado.	
<i>Proceso</i>	Rol	Sistema
	<ol style="list-style-type: none"> El cine accede a la pantalla de ofertas. El cine elimina una oferta haciendo clic en el botón "Eliminar" al lado de la oferta. 	<ol style="list-style-type: none"> El sistema muestra la lista de sus ofertas. El sistema elimina la oferta.
<i>Alternativas</i>	<ol style="list-style-type: none"> El sistema muestra un error al eliminar la oferta y le vuelve a mostrar la lista. 	

Aplicar oferta		
<i>Roles</i>	Cine	
<i>Descripción</i>	Un cine aplica una oferta a un suscriptor.	
<i>Condición previa</i>	Estar logeado.	
<i>Proceso</i>	Rol	Sistema
	<ol style="list-style-type: none"> 1. El cine accede a la pantalla de ofertas. 1. El cine clics sobre una oferta haciendo clic en el botón “Ver” al lado de la oferta. 1. El cine rellena el formulario y clics sobre “aplicar oferta”. 	<ol style="list-style-type: none"> 1. El sistema muestra la lista de sus ofertas. 1. El sistema muestra una pantalla de la oferta con un formulario. 1. El sistema muestra un mensaje de confirmación.
<i>Alternativas</i>	<ol style="list-style-type: none"> 1. El Sistema muestra un mensaje de error al aplicar la oferta y le redirige a la lista de ofertas. 	

Ver ofertas publicadas							
<i>Roles</i>	Todos los usuarios						
<i>Descripción</i>	Un usuario ve todas las ofertas que hay publicadas.						
<i>Condición previa</i>	No hay.						
<i>Proceso</i>	<table border="1"> <thead> <tr> <th>Rol</th> <th>Sistema</th> </tr> </thead> <tbody> <tr> <td>1. Un usuario accede a la aplicación.</td> <td>1. El sistema muestra la pantalla inicial.</td> </tr> <tr> <td>1. El usuario clica sobre "Ofertas".</td> <td>1. El sistema muestra la lista de ofertas.</td> </tr> </tbody> </table>	Rol	Sistema	1. Un usuario accede a la aplicación.	1. El sistema muestra la pantalla inicial.	1. El usuario clica sobre "Ofertas".	1. El sistema muestra la lista de ofertas.
	Rol	Sistema					
1. Un usuario accede a la aplicación.	1. El sistema muestra la pantalla inicial.						
1. El usuario clica sobre "Ofertas".	1. El sistema muestra la lista de ofertas.						
<i>Alternativas</i>							

2.8.4. Casos de uso: gestión de películas.

<i>Alta de una película</i>		
<i>Roles</i>	Administrador	
<i>Descripción</i>	Un administrador da de alta una película en el sistema.	
<i>Condición previa</i>	El administrador debe estar logeado.	
<i>Proceso</i>	Rol	Sistema
	<ol style="list-style-type: none">1. El administrador clica sobre "Añadir película".1. El administrador rellena el formulario y aprieta el botón de crear.	<ol style="list-style-type: none">1. El sistema muestra un formulario.1. El sistema guarda la película y muestra un mensaje de confirmación.
<i>Alternativas</i>	<ol style="list-style-type: none">1. El sistema no puede guardar la película y muestra un mensaje de error.	

Baja de una película		
<i>Roles</i>	Administrador	
<i>Descripción</i>	Un administrador da de baja una película del sistema.	
<i>Condición previa</i>	El administrador debe estar logeado.	
<i>Proceso</i>	Rol	Sistema
	<ol style="list-style-type: none"> 1. El administrador clica sobre "Listar películas". 1. El administrador clica sobre el botón eliminar que aparece al lado de cada película. 1. El administrador confirma la eliminación. 	<ol style="list-style-type: none"> 1. El sistema muestra un la lista de películas. 1. El sistema pregunta si se quiere eliminar. 1. El sistema elimina la película y muestra un mensaje de confirmación.
<i>Alternativas</i>	<ol style="list-style-type: none"> 1. El sistema muestra un mensaje de error al intentar eliminar la película. 	

Modificar datos de una película		
<i>Roles</i>	Administrador	
<i>Descripción</i>	Un administrador modifica los datos de una película.	
<i>Condición previa</i>	El administrador debe estar logeado.	
<i>Proceso</i>	Rol	Sistema
	<ol style="list-style-type: none"> 1. El administrador clic sobre "Listar películas". 1. El administrador clic sobre el botón modificar datos que aparece al lado de cada película. 1. El administrador cambia lo que desea y clic sobre "Guardar". 	<ol style="list-style-type: none"> 1. El sistema muestra un la lista de películas. 1. El sistema muestra el formulario de crear nueva película con los datos rellenos. 1. El sistema guarda los datos y muestra un mensaje de confirmación.
<i>Alternativas</i>	<ol style="list-style-type: none"> 1. El sistema muestra un mensaje de error al intentar guardar los cambios. 	

Ver lista de películas		
<i>Roles</i>	Cualquier usuario	
<i>Descripción</i>	Un usuario ve la lista de películas que hay en el sistema.	
<i>Condición previa</i>	No hay.	
<i>Proceso</i>	Rol	Sistema
	1. Un usuario clica sobre "Listar películas".	1. El sistema muestra todas las películas.
<i>Alternativas</i>		

Ver película		
<i>Roles</i>	Suscriptor y administrador	
<i>Descripción</i>	Un suscriptor o administrador reproduce la película que desea.	
<i>Condición previa</i>	Estar logeado.	
<i>Proceso</i>	Rol	Sistema
	1. El suscriptor o administrador clica sobre "Listar películas". 1. El suscriptor o administrador clica sobre el botón "Ver película".	1. El sistema muestra un la lista de películas. 1. El sistema reproduce la película.
<i>Alternativas</i>	1. El sistema no puede reproducir la película y muestra un mensaje de error.	

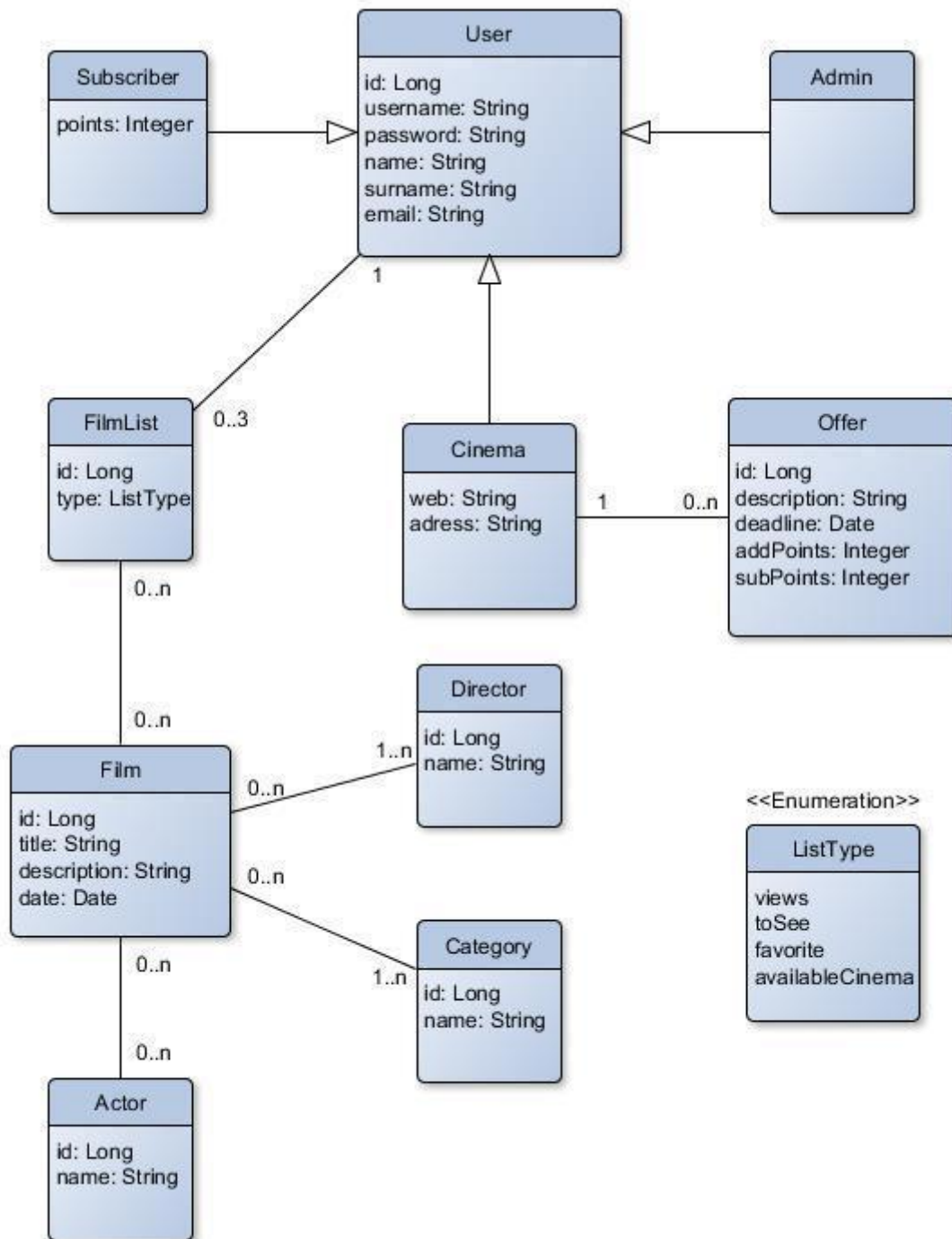
2.8.5. Casos de uso: gestión de otros campos.

<i>Alta de datos necesarios de las películas</i>		
<i>Roles</i>	Administrador	
<i>Descripción</i>	Un administrador añade un dato necesario para la descripción de las películas, como actores, director o categoría.	
<i>Condición previa</i>	Estar logeado.	
<i>Proceso</i>	Rol	Sistema
	<ol style="list-style-type: none"> El administrador accede a la pantalla de mantenimiento de los datos necesarios. El administrador rellena el campo del dato que quiere introducir y clic en "Añadir". 	<ol style="list-style-type: none"> El sistema muestra la pantalla del mantenimiento. El sistema lo añade y muestra un mensaje de confirmación.
<i>Alternativas</i>	<ol style="list-style-type: none"> El sistema muestra un mensaje de error al añadirlo. 	

Baja de datos necesarios de las películas		
<i>Roles</i>	Administrador	
<i>Descripción</i>	Un administrador elimina un dato necesario para la descripción de las películas, como actores, director o categoría.	
<i>Condición previa</i>	Estar logeado.	
<i>Proceso</i>	Rol	Sistema
	<ol style="list-style-type: none"> El administrador accede a la pantalla de mantenimiento de los datos necesarios. El administrador selecciona un elemento de la lista de uno de los datos necesarios y clic en "Eliminar". 	<ol style="list-style-type: none"> El sistema muestra la pantalla del mantenimiento. El sistema lo elimina y muestra un mensaje de confirmación.
<i>Alternativas</i>	<ol style="list-style-type: none"> El sistema muestra un mensaje de error al eliminarlo. 	

2.9. Modelo conceptual.

Aquí se puede ver un diagrama UML del modelo conceptual de la aplicación.



3. Diseño.

En este punto se explicará la arquitectura utilizada en el proyecto y el diseño de la misma.

Como se ha comentado en apartados anteriores, la aplicación se ha creado mediante spring boot y angularJS. Se ha seguido un modelo de tres capas, la capa de presentación, la capa de servicio y la capa de datos.

1. La capa de presentación: en esta parte de la aplicación, se interactúa directamente con el cliente. Aquí se visualizan las funciones de la aplicación y se recogen y envían los datos. Para esta capa se utilizará los frameworks Bootstrap para el diseño de la propia aplicación, ya que es una manera relativamente sencilla de crear una interfaz clara para el usuario, y AngularJS para comunicarse con la capa de servicio y alguna funcionalidad más.
2. La capa de servicio: En esta parte se reciben los datos, se interactúa con la base de datos, se procesan los datos y se mandan los resultados. En esta capa se utiliza el framework Spring Boot.

Este framework permite crear aplicaciones de forma rápida y sencilla. Se ha utilizado para crear una REST para obtener y almacenar los datos de los usuarios, películas y demás, en la base de datos.

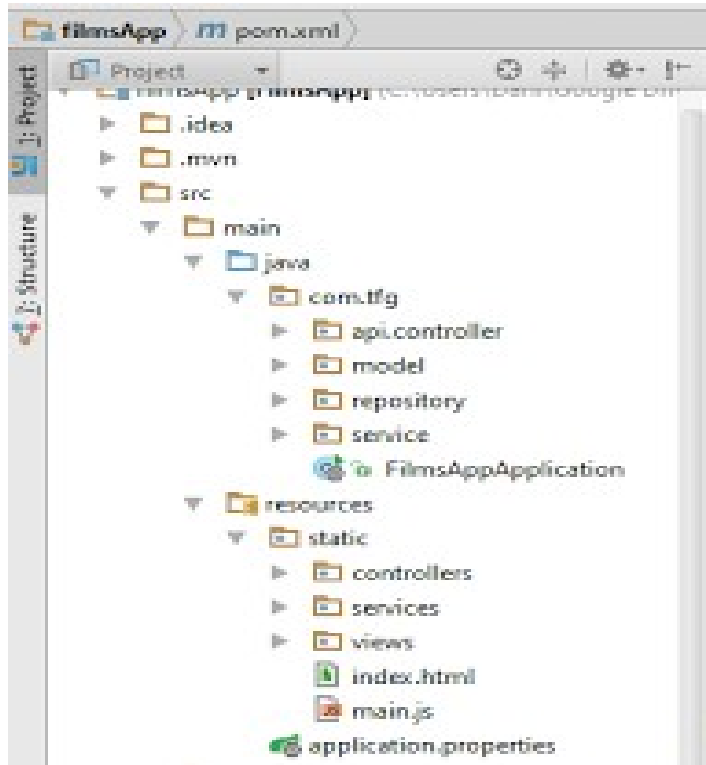
Para la persistencia de datos, finalmente se utilizará el framework JPA que se explicará a continuación.

3. La capa de datos: Es donde se almacenen los datos. Se ha utilizado una base de datos MySQL.

En esta imagen se puede ver una estructura de 3 capas parecida a la de mi aplicación. La parte de la seguridad no la he implementado y mi base de datos es MySQL.



En mi aplicación los ficheros se dividen en dos partes. Por una parte está la capa de servicio dentro de una carpeta llamada “Java” y la capa de presentación en una carpeta llamada “resources”. Dentro de esta última carpeta, también hay un archivo properties en donde se encuentra la configuración necesaria para la conexión con la base de datos.



La aplicación utiliza maven para añadir todas las dependencias necesarias para la implementación. Por ejemplo para poder utilizar JPA o para poder realizar la conexión con la base de datos.

Estas dependencias se añaden a un archivo pom.xml.

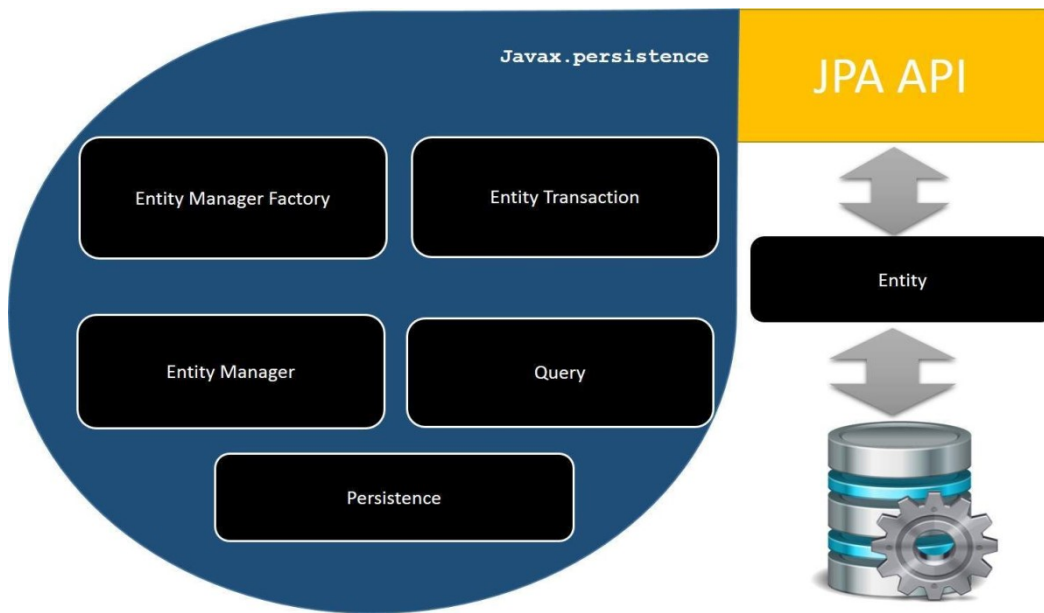
3.1. Arquitectura JPA

Como he dicho antes, para la persistencia de datos se ha utilizado el framework JPA. Se ha decidido esto porque quita mucho trabajo, ya que no es necesario realizar toda la codificación de los daos para comunicarse con la base de datos.

JPA (Java Persistence API) proporciona un modelo basado en POJO's para mapear bases de datos relacionales. El mapeo-relacional se realiza mediante anotaciones en las clases de entidad.

Este framework nos permite desarrollar más rápido, trabajar con la base de datos por medio de entidades en vez de querys y mejora el mantenimiento de la aplicación.

También hay que decir que requiere una curva de aprendizaje más alta que otros frameworks.



Podemos ver un ejemplo en la aplicación. Las entidades tienen las anotaciones necesarias.

```

package com.tfg.model;

import ...

@Entity
@Data
@NoArgsConstructor
public class User extends ModelItem {

    public User(String username, String password, String name, String surname, String nif, String email) {

        this.username = username;
        this.password = password;
        this.name = name;
        this.surname = surname;
        this.nif = nif;
        this.email = email;
    }

    @Id
    @GeneratedValue
    private Integer id_user;

    @Column(nullable = false)
    private String username;

    @Column(nullable = false)
    private String password;

    @Column(nullable = false)
    private String name;

    @Column(nullable = false)
}
    
```

En la carpeta “repository” se implementan las operaciones con la base de datos. Las consultas básicas (leer, añadir, modificar y eliminar) ya vienen implementadas, por lo que no hay que tocar nada. Si se quiere realizar otro tipo de operación se debe indicar.

```

package com.tfg.repository;

import ...

@Repository
public interface Film_actorRepository extends JpaRepository<Film_actor, Integer> {

    Film_actor findByIdFilmAndIdActor(Integer idFilm, Integer idActor);

    List<Film_actor> findByIdActor(Integer idActor);

    List<Film_actor> findByIdFilm(Integer idFilm);

    void deleteByIdFilmAndIdActor(Integer idFilm, Integer idActor);
}
    
```

3.2. Diseño de la REST

Una REST es una interfaz entre sistemas que sirven para obtener datos o generar operaciones sobre ellos en todos los formatos posibles como JSON o XML.

Las operaciones más importantes que se realizan con los datos son GET, POST, PUT y DELETE (consultar, crear, editar y eliminar).

Unas ventajas de REST es que existe una separación entre el cliente y el servidor. Separa totalmente la interfaz de usuario del servidor y el almacenamiento de datos. Esto mejora la portabilidad de la interfaz a otras plataformas.

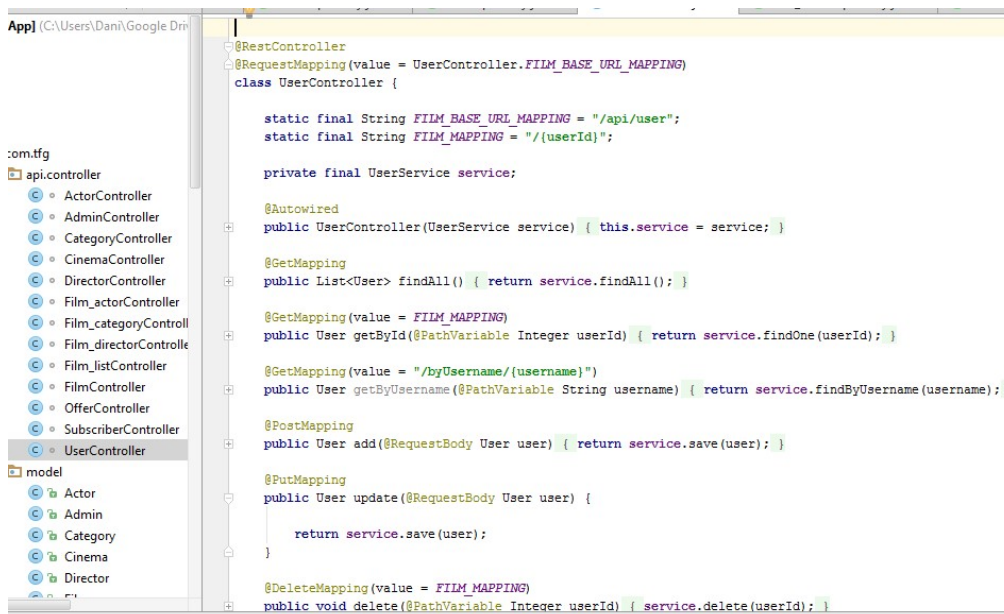
Otras ventajas son la visibilidad, fiabilidad y escalabilidad. La separación entre cliente y servidor permite escalar el producto sin excesivos problemas. Se puede migrar a otros servidores o realizar todo tipo de cambios en la base de datos, siempre y cuando los datos de cada una de las peticiones se envíen de forma correcta. Esta separación facilita tener en servidores distintos el front y el back y eso convierte a las aplicaciones en productos más flexibles a la hora de trabajar.

Por último, destacar que una REST es independiente del tipo de lenguaje o plataforma. Lo importante es que los datos se envíen con el formato adecuado, en nuestro caso JSON.

Para crear la REST de la aplicación se ha utilizado Spring Boot. Las partes principales que tiene esta parte, son el modelo de cada entidad y un controlador de cada entidad que llama a los servicios necesarios para cumplir las funcionalidades.

Para crear la REST, se ha creado un controller por cada entidad, estos controllers, llaman a su servicio correspondiente y desde este servicio se llama al repository JPA para realizar la operación.

En los controllers se ha creado un método por cada operación, añadiendo la anotación necesaria para poder acceder a las operaciones de la REST desde la capa de presentación.



```
@RestController
@RequestMapping(value = UserController.FILM_BASE_URL_MAPPING)
class UserController {

    static final String FILM_BASE_URL_MAPPING = "/api/user";
    static final String FILM_MAPPING =("/{userId}";

    private final UserService service;

    @Autowired
    public UserController(UserService service) { this.service = service; }

    @GetMapping
    public List<User> findAll() { return service.findAll(); }

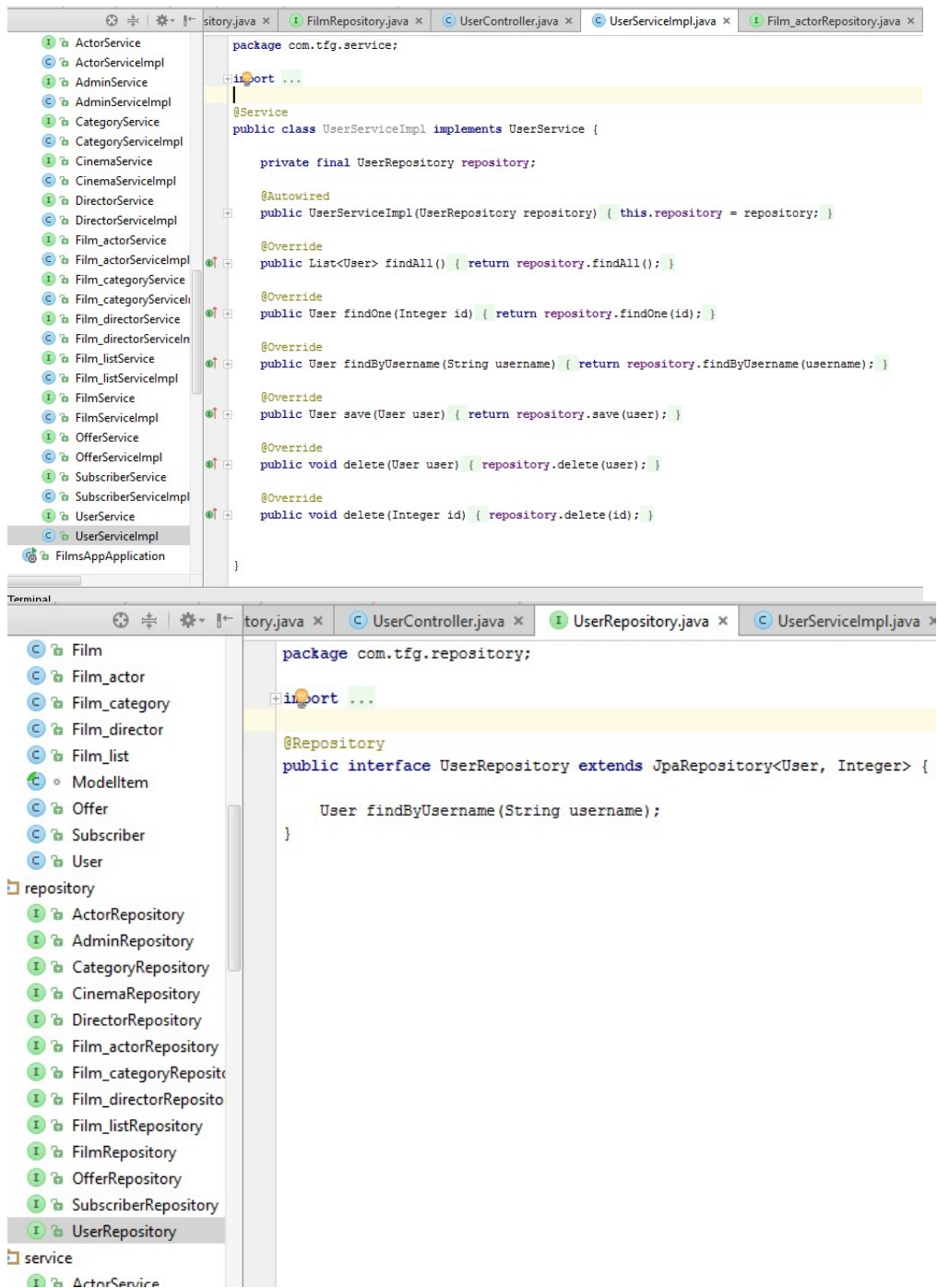
    @GetMapping(value = FILM_MAPPING)
    public User getById(@PathVariable Integer userId) { return service.findOne(userId); }

    @GetMapping(value = "/byUsername/{username}")
    public User getByUsername(@PathVariable String username) { return service.findByUsername(username); }

    @PostMapping
    public User add(@RequestBody User user) { return service.save(user); }

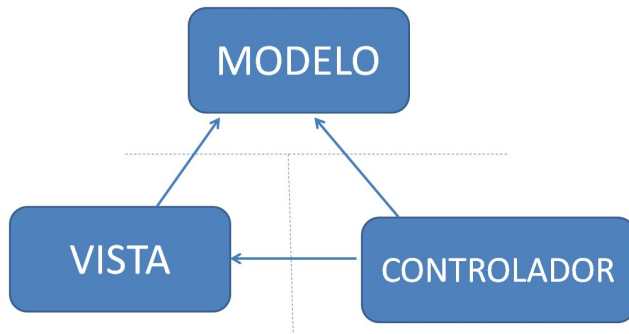
    @PutMapping
    public User update(@RequestBody User user) {
        return service.save(user);
    }

    @DeleteMapping(value = FILM_MAPPING)
    public void delete(@PathVariable Integer userId) { service.delete(userId); }
```

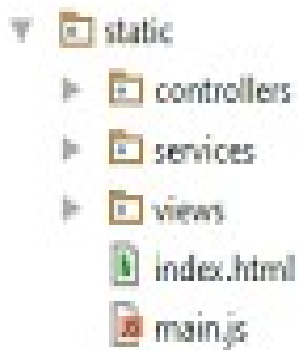



3.3. Capa de presentación

El servicio de presentación se ha realizado mediante AngularJS, un framework de JavaScript, y siguiendo una estructura MVC (modelo-vista-controlador).



Esta capa se compone de diferentes partes. Por una parte está el archivo main.js que vendría a ser el archivo de configuración en donde se inyectan los módulos angular que se van a utilizar entre otras cosas. Por otra parte, tenemos las vistas que corresponden a las diferentes funcionalidades. Cada vista tiene su controlador en donde se implementan las funcionalidades. Por último, se han creado servicios en donde se desarrollan funciones comunes para no repetir código en diferentes controladores.



La página html principal contiene la cabecera y el pie de página común para las diferentes funcionalidades de la aplicación. En esta página, se van inyectando las diferentes vistas de las funcionalidades.

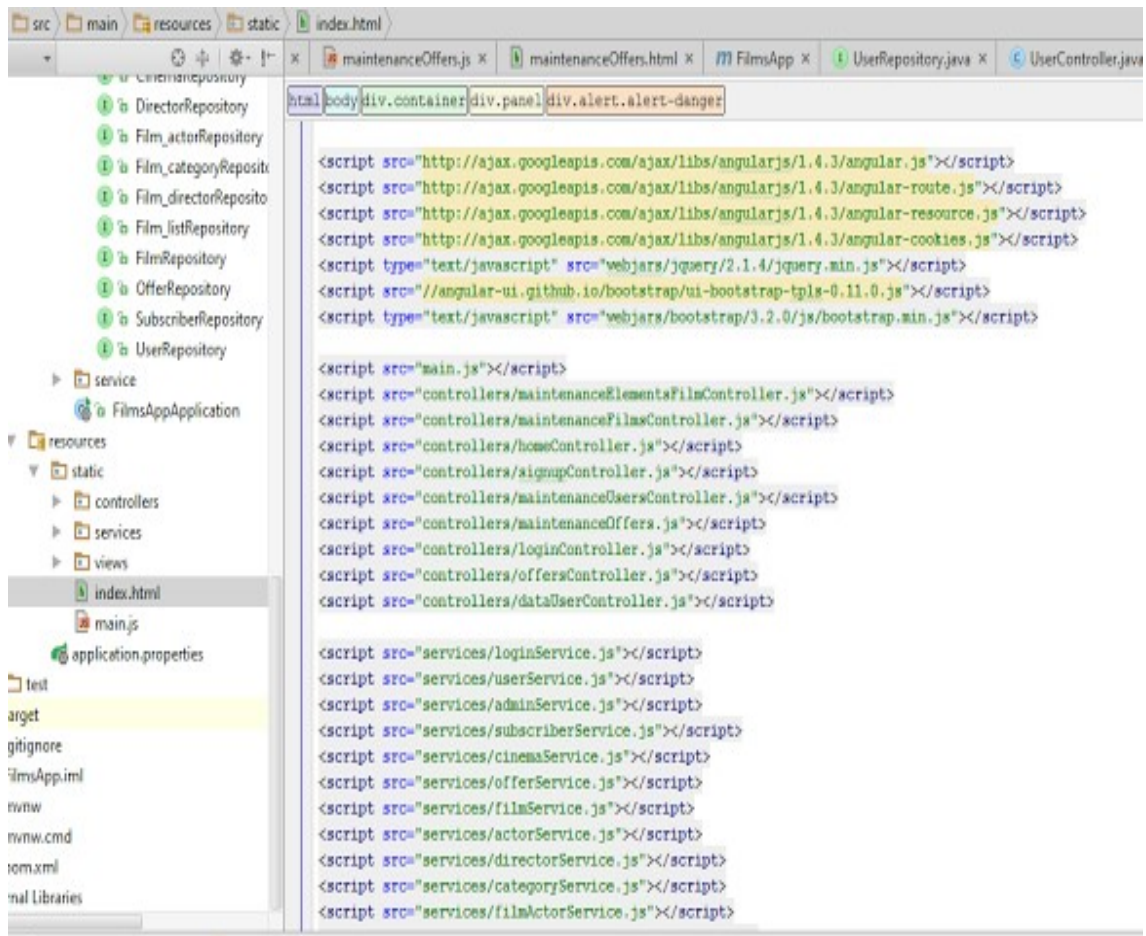
Esto es la página principal:



```
html body div.container div.panel div.alert.alert-danger
  <li><a href="#/signup"><span class="glyphicon glyphicon-user"></span> Alta usuario</a></li>
  <li><a href="#/login"><span class="glyphicon glyphicon-log-in"></span> Login</a></li>
</ul>
<ul class="nav navbar-nav navbar-right" ng-show="role != 'user'">
  <li><a href="#/dataUser">Datos usuario</a></li>
  <li><a ng-click="mainCtrl.logout()"><span class="glyphicon glyphicon-log-out"></span> Logout</a></li>
</ul>
</div>
</nav>
</div>
<div ng-show="error" class="alert alert-danger">
  {{error}}
</div>
<div ng-show="msg" class="alert alert-success">
  {{msg}}
</div>
<div id="page-content-wrapper">
  <div class="panel">
    <div ng-view>
    </div>
  </div>
</div>
<div class="container-fluid">
  <div class="row" style="...">
    <div ng-show="role != 'user'">
      <h4>Usuario: {{username}}</h4>
      <h4>Conectado como: {{role}}</h4>
    </div>
  </div>
</div>
```

En el “ng-view” es donde se introducen las vistas de cada funcionalidad.

Aquí también se añaden todos los controllers y servicios utilizados, además de las librerías necesarias.

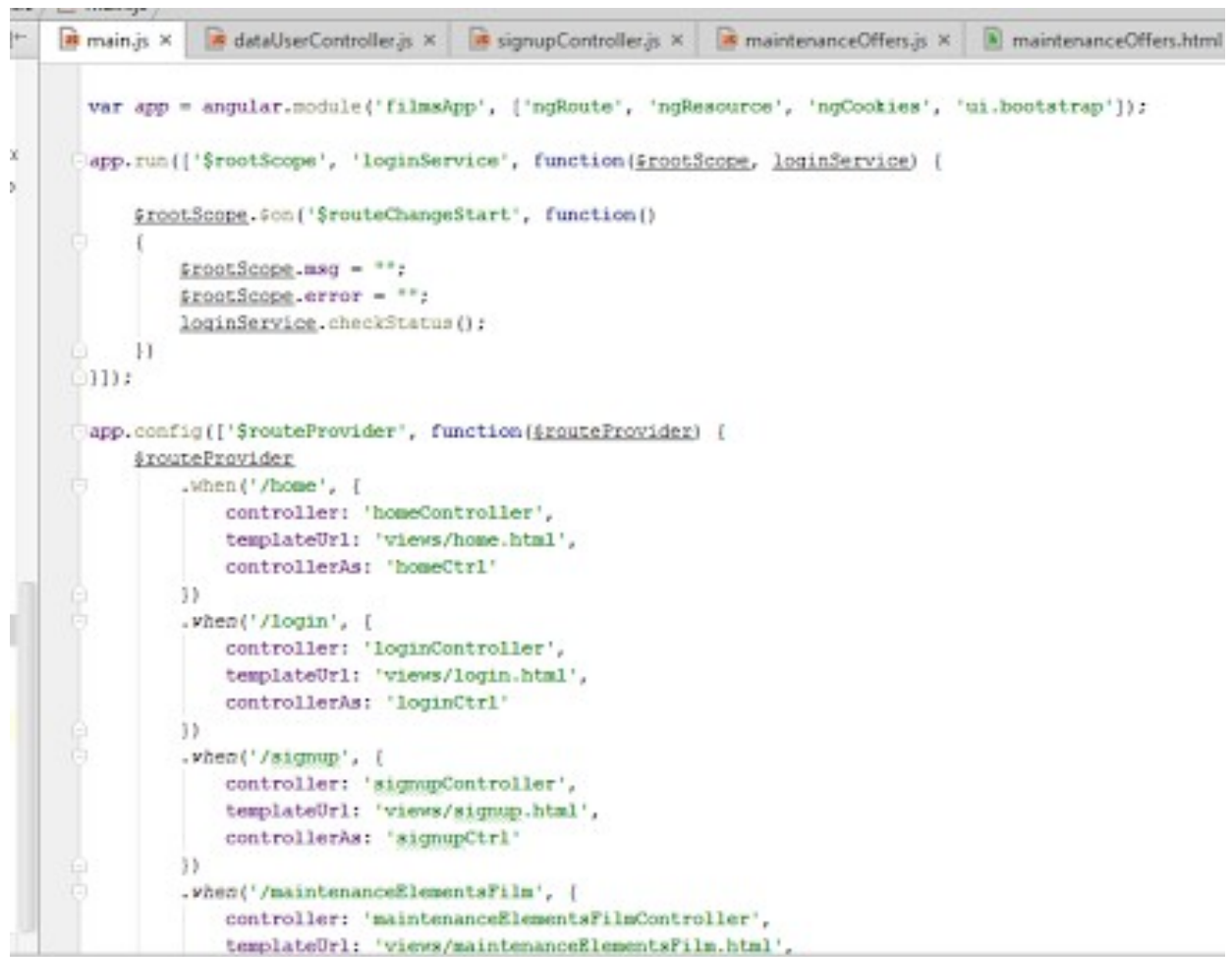


```
html body div.container div.panel div.alert.alert-danger<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.3/angular.js"></script><script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.3/angular-route.js"></script><script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.3/angular-resource.js"></script><script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.3/angular-cookies.js"></script><script type="text/javascript" src="webjars/jquery/2.1.4/jquery.min.js"></script><script src="//angular-ui.github.io/bootstrap/ui-bootstrap-tpls-0.11.0.js"></script><script type="text/javascript" src="webjars/bootstrap/3.2.0/js/bootstrap.min.js"></script><script src="main.js"></script><script src="controllers/maintenanceElementsFilmController.js"></script><script src="controllers/maintenanceFilmsController.js"></script><script src="controllers/homeController.js"></script><script src="controllers/signupController.js"></script><script src="controllers/maintenanceUsersController.js"></script><script src="controllers/maintenanceOffers.js"></script><script src="controllers/loginController.js"></script><script src="controllers/offersController.js"></script><script src="controllers/dataUserController.js"></script><script src="services/loginService.js"></script><script src="services/userService.js"></script><script src="services/adminService.js"></script><script src="services/subscriberService.js"></script><script src="services/cinemaService.js"></script><script src="services/offerService.js"></script><script src="services/filmService.js"></script><script src="services/actorService.js"></script><script src="services/directorService.js"></script><script src="services/categoryService.js"></script><script src="services/filmActorService.js"></script>
```

También es posible descargarse las librerías y sustituir estas dependencias por las rutas donde se encuentran.

Las vistas son páginas html creadas con la ayuda de bootstrap, un framework que permite diseñar las vistas de una forma relativamente fácil.

El archivo main.js contiene los módulos necesarios para el correcto funcionamiento como utilizar cookies o para definir las rutas de las vistas y asignar sus controladores.



```
main.js x  dataUserController.js x  signupController.js x  maintenanceOffers.js x  maintenanceOffers.html

var app = angular.module('filmsApp', ['ngRoute', 'ngResource', 'ngCookies', 'ui.bootstrap']);

app.run(['$rootScope', 'loginService', function($rootScope, loginService) {

    $rootScope.$on('$routeChangeStart', function()
    {
        $rootScope.msg = '';
        $rootScope.error = '';
        loginService.checkStatus();
    })
}]);

app.config(['$routeProvider', function($routeProvider) {
    $routeProvider
        .when('/home', [
            controller: 'homeController',
            templateUrl: 'views/home.html',
            controllerAs: 'homeCtrl'
        ])
        .when('/login', [
            controller: 'loginController',
            templateUrl: 'views/login.html',
            controllerAs: 'loginCtrl'
        ])
        .when('/signup', [
            controller: 'signupController',
            templateUrl: 'views/signup.html',
            controllerAs: 'signupCtrl'
        ])
        .when('/maintenanceElementsFilm', [
            controller: 'maintenanceElementsFilmController',
            templateUrl: 'views/maintenanceElementsFilm.html',
```

Los servicios contienen las funciones para llamar a la REST. Un ejemplo sería:



```
services > userService.js
main.js x  userService.js x  signupController.js x  maintenanceOffers.js x  maintenanc

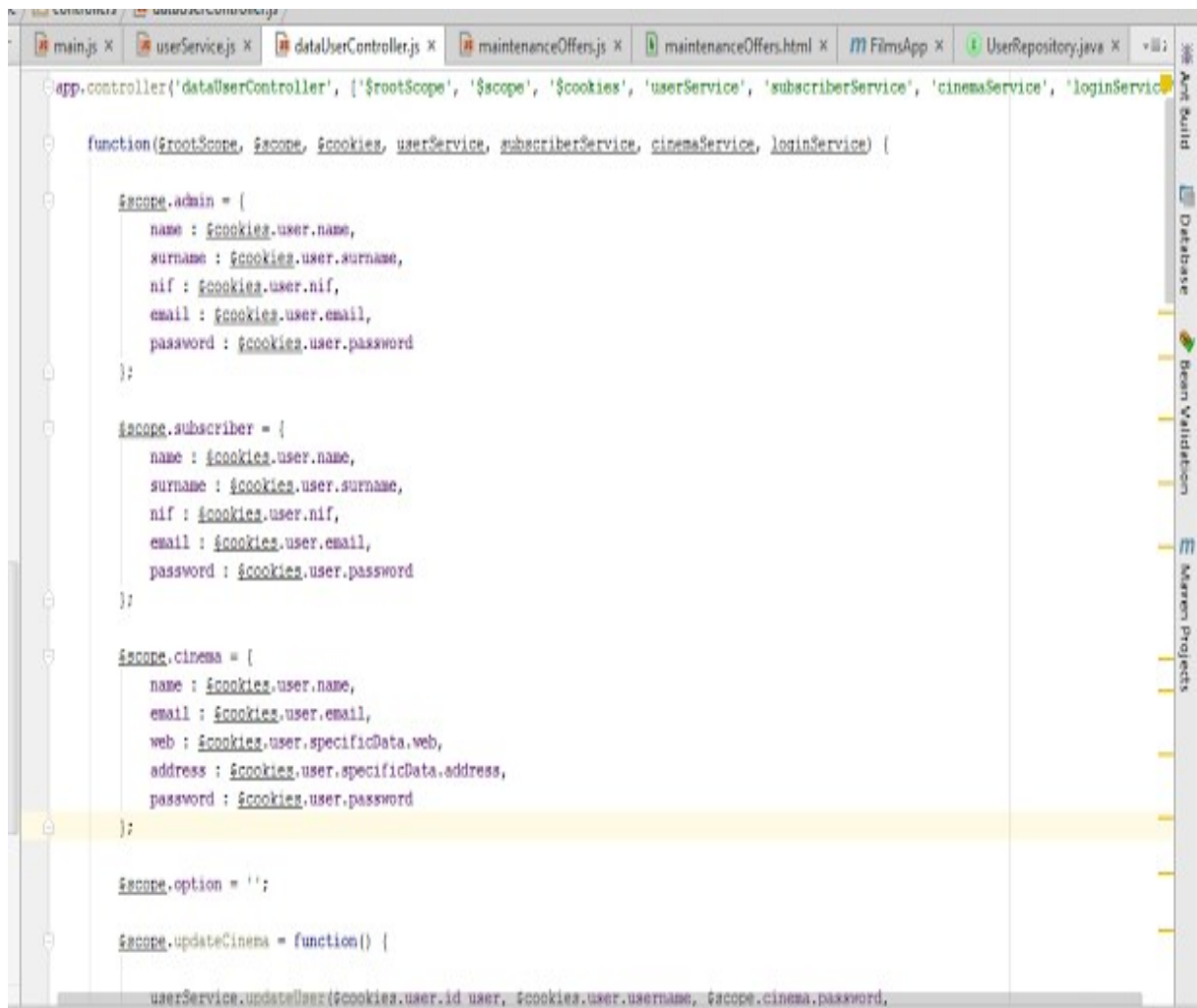
|:
  this.getUser = function(username) {
    var deferred = $q.defer();
    var promise = deferred.promise;

    $http.get('/api/user/byUsername/'+username)
      .success(function(data) {
        deferred.resolve(data);
      })
      .error(function(error) {
        deferred.reject(error);
      });
    return promise;
  };

  this.addUser = function(username, password, name, surname, nif, email) {
    var deferred = $q.defer();
    var promise = deferred.promise;

    $http.post('/api/user',
      {
        username : username,
        password : password,
        name : name,
        surname : surname,
        nif : nif,
        email : email
      })
      .success(function(data) {
        deferred.resolve(data);
      }).error(function(error) {
        deferred.reject(error);
      });
  };
}
```

Un ejemplo de un controlador:



```
app.controller('dataUserController', ['$rootScope', '$scope', '$cookies', 'userService', 'subscriberService', 'cinemaService', 'loginService']  
  
function($rootScope, $scope, $cookies, userService, subscriberService, cinemaService, loginService) {  
  
    $scope.admin = {  
        name : $cookies.user.name,  
        surname : $cookies.user.surname,  
        nif : $cookies.user.nif,  
        email : $cookies.user.email,  
        pasavord : $cookies.user.password  
    };  
  
    $scope.subscriber = {  
        name : $cookies.user.name,  
        surname : $cookies.user.surname,  
        nif : $cookies.user.nif,  
        email : $cookies.user.email,  
        password : $cookies.user.password  
    };  
  
    $scope.cinema = {  
        name : $cookies.user.name,  
        email : $cookies.user.email,  
        web : $cookies.user.specificData.web,  
        address : $cookies.user.specificData.address,  
        password : $cookies.user.password  
    };  
  
    $scope.option = '';  
  
    $scope.updateCinema = function() {  
  
        userService.updateUser($cookies.user.id user, $cookies.user.username, $scope.cinema.password,
```

4. Entorno de trabajo.

Para desarrollar la aplicación se ha utilizado el siguiente entorno de trabajo con las siguientes características:

- Windows 10.
- Java 7.
- Angular 1.4.3
- Spring Boot
- JPA
- Bootstrap 3.2.0
- IntelliJ IDEA: se ha utilizado este ide para realizar la codificación.
- MySQL Workbench 6.3: para gestionar la base de datos.
- Apache Maven 3.3: se ha utilizado maven para construir el proyecto ya que, como he mencionado, permite inyectar las dependencias necesarias de forma sencilla.
- GITExtensions: he utilizado esta herramienta para controlar las versiones del proyecto pero de forma local, es decir, el repositorio se encuentra en una carpeta en mi pc sincronizada con GoogleDrive, por lo que me permite llevar un control de los cambios y versiones y al mismo tiempo tener una copia de seguridad en el Drive. Pero no puedo acceder al repositorio desde otro ordenador.
- yEd Graph Editor: Para realizar los diagramas.
- Icecream Screen Recorder y Aimersoft Video Editor: para el video de presentación.

5. Mejoras de versiones futuras.

En este punto explicaré algunas de las funcionalidades que no he podido desarrollar por falta de tiempo o por dificultad.

Una próxima versión podría dar la posibilidad a los usuarios registrados marcar las películas como favoritas, vistas o pendientes de ver y a los cines marcar como disponibles aquellas que se puedan ver en ellos. Esta posibilidad, implicaría añadir otra funcionalidad para permitir listar las películas por favoritas, vistas o pendientes de ver y ver las películas disponibles en los cines.

Otra cosa que se podría mejorar es la seguridad de la REST, ya que el login actual impide acceder a ciertas funcionalidades pero no impide utilizar la REST desde otro medio si se conocen las peticiones http.

También se podrían añadir la disponibilidad ver series de televisión.

Añadir más información sobre los actores y directores podría ser otra mejora interesante.

6. Conclusiones.

Realizar el TFG me ha hecho aprender muchos conocimientos nuevos así como ampliar otros. Elegí hacer TFG relacionado con Java EE porque quería aprender a desarrollar una aplicación web desde cero y creo que ese objetivo lo he cumplido, ya que pienso el resultado obtenido, aunque se puede mejorar mucho, es bueno.

La elección de las tecnologías creo que han sido acertadas. Son tecnologías relativamente nuevas que cada vez se utilizan más. En el caso de Angular y Spring Boot, ya tenía algún conocimiento que he podido ampliar, pero el framework JPA no lo conocía y la curva de aprendizaje ha sido mayor.

Por otra parte, he tenido la oportunidad de desempeñar funciones que no había hecho antes como puede ser el requisito de funcionalidades o la planificación, es decir, la parte previa al desarrollo, y, aunque es la parte que menos me ha gustado, he podido comprobar que es imprescindible para implementar la aplicación de forma correcta. Finalmente, decir que la curva de aprendizaje, en general, ha sido más grande de lo esperado y no he podido implementar algunas funcionalidades que tenía pensadas.

Pese a ello, pienso que ha sido una buena experiencia para conocer todas las fases de desarrollo de una aplicación web.

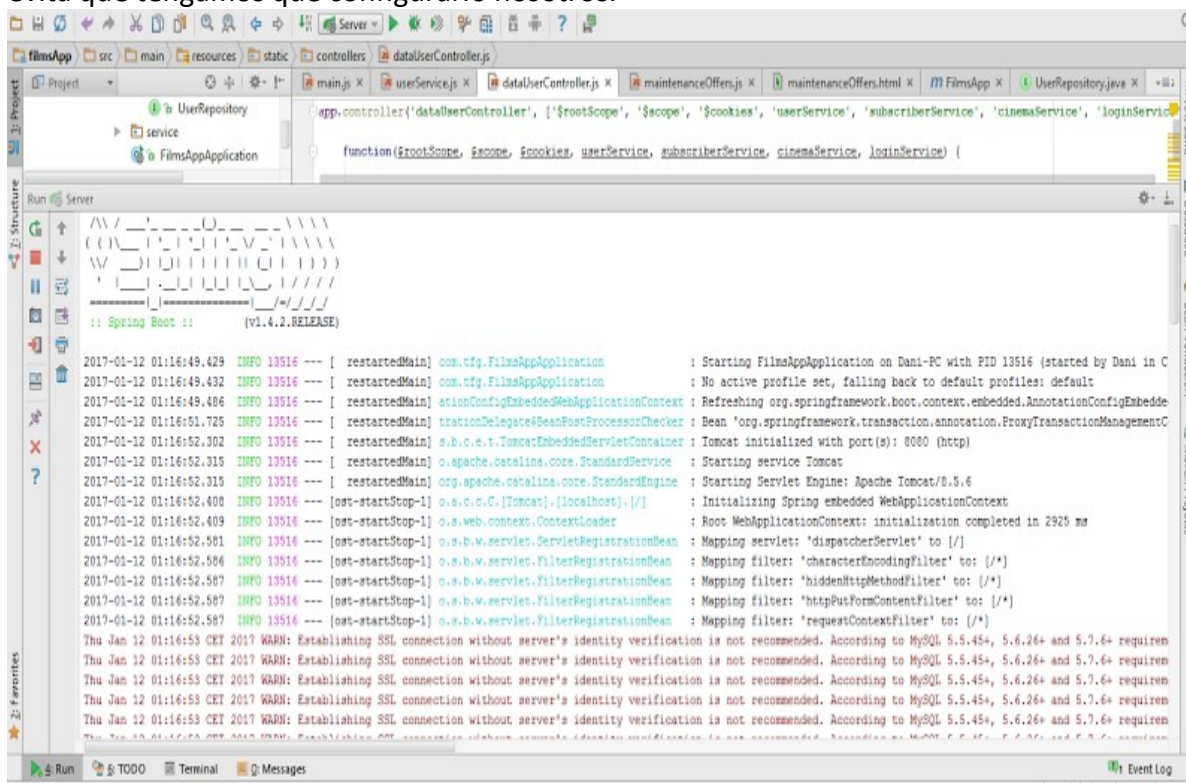
7. Manual aplicación.

En este punto explicaré cómo arrancar la aplicación y las funcionalidades de las diferentes pantallas.

Para utilizar la aplicación hay que seguir los siguientes pasos:

1. Lo primero es abrir el proyecto con un IDE, en mi caso IntelliJ IDEA.
2. A continuación, editar la configuración y añadir spring Boot, indicando la clase principal, para arrancar la aplicación.
3. Realizar un clean install, ya que es un proyecto maven.
4. Arrancar la aplicación.

Spring Boot arranca un servidor Tomcat para el despliegue de la aplicación, lo que evita que tengamos que configurarlo nosotros.



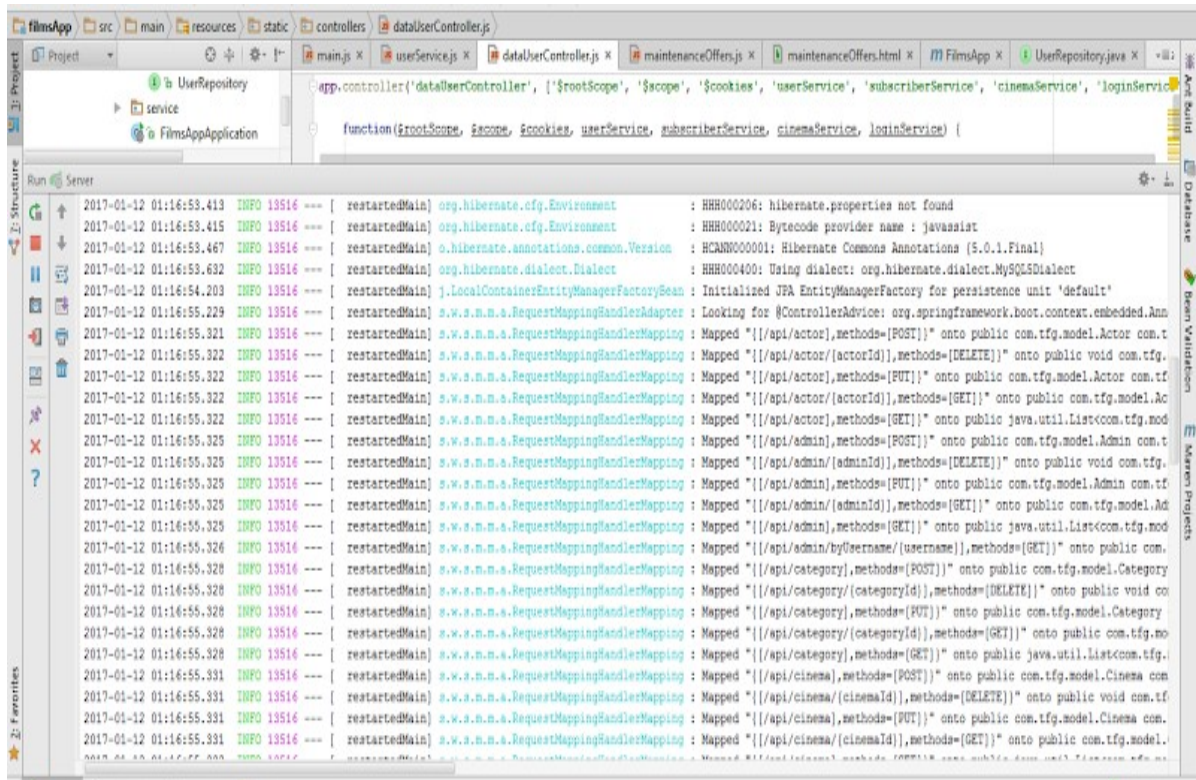
```

  ____  __  _  / 
 / ___/ /_/_/  / 
/ /   /_/_/  / 
/ ___/ /_/_/  / 
/_/   /_/_/  / 
:: Spring Boot :: (v1.4.2.RELEASE)

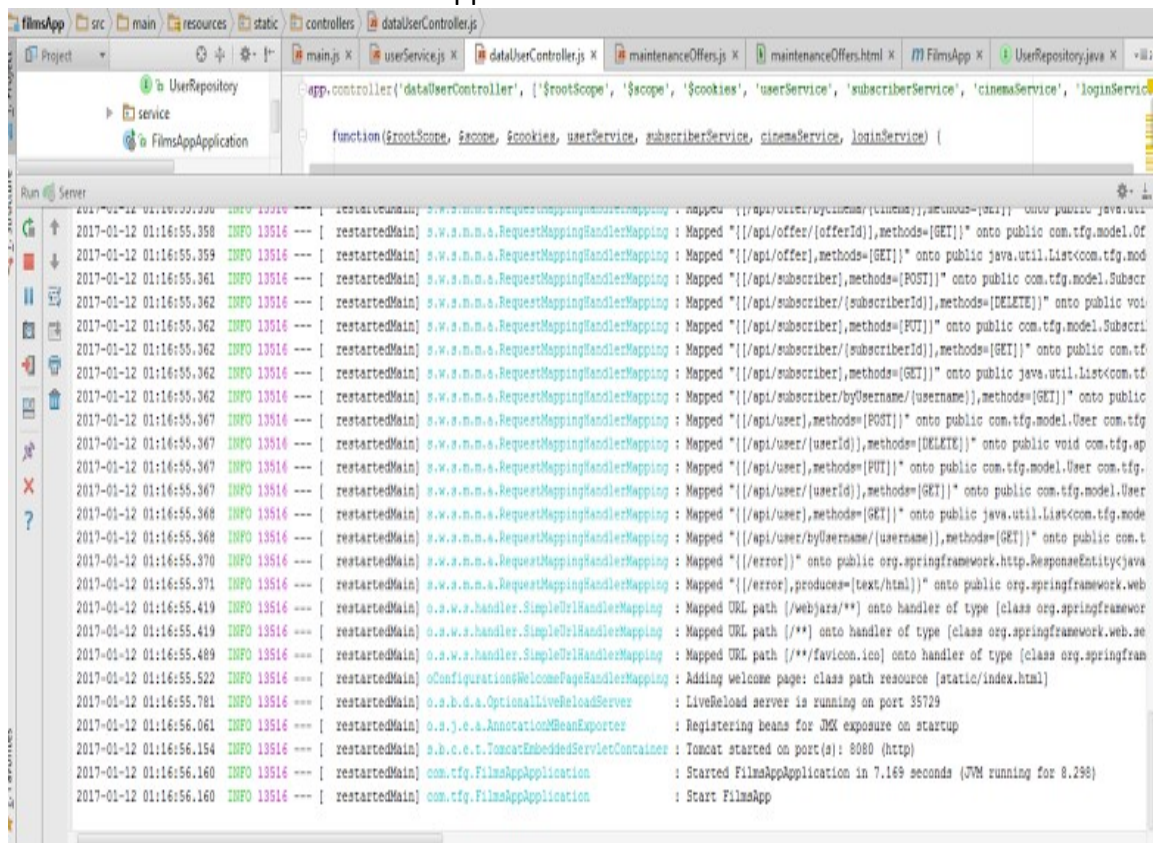
2017-01-12 01:16:49.429 INFO 13516 --- [ restartedMain] com.cfg.FilmsAppApplication : Starting FilmsAppApplication on Dani-PC with PID 13516 (started by Dani in C
2017-01-12 01:16:49.432 INFO 13516 --- [ restartedMain] com.cfg.FilmsAppApplication : No active profile set, falling back to default profiles: default
2017-01-12 01:16:49.486 INFO 13516 --- [ restartedMain] ationConfigEmbeddedWebApplicationContext : Refreshing org.springframework.boot.context.embedded.AnnotationConfigEmbedde
2017-01-12 01:16:51.725 INFO 13516 --- [ restartedMain] trationDelegateBeanPostProcessorChecker : Bean 'org.springframework.transaction.annotation.ProxyTransactionManagementC
2017-01-12 01:16:52.302 INFO 13516 --- [ restartedMain] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat initialized with port(s): 8080 (http)
2017-01-12 01:16:52.315 INFO 13516 --- [ restartedMain] o.apache.catalina.core.StandardService : Starting service Tomcat
2017-01-12 01:16:52.315 INFO 13516 --- [ restartedMain] org.apache.catalina.core.StandardEngine : Starting Servlet Engine: Apache Tomcat/8.5.6
2017-01-12 01:16:52.408 INFO 13516 --- [ost-startStop-1] o.s.c.o.c.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2017-01-12 01:16:52.409 INFO 13516 --- [ost-startStop-1] o.s.web.context.ContextLoader : Root WebApplicationContext: initialization completed in 2925 ms
2017-01-12 01:16:52.581 INFO 13516 --- [ost-startStop-1] o.s.b.w.servlet.ServletRegistrationBean : Mapping servlet: 'dispatcherServlet' to [/]
2017-01-12 01:16:52.586 INFO 13516 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'characterEncodingFilter' to: [/]*
2017-01-12 01:16:52.587 INFO 13516 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'hiddenHttpMethodFilter' to: [/]*
2017-01-12 01:16:52.587 INFO 13516 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'httpPutFormContentFilter' to: [/]*
2017-01-12 01:16:52.587 INFO 13516 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'requestContextFilter' to: [/]*
Thu Jan 12 01:16:53 CET 2017 WARN: Establishing SSL connection without server's identity verification is not recommended. According to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ require
Thu Jan 12 01:16:53 CET 2017 WARN: Establishing SSL connection without server's identity verification is not recommended. According to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ require
Thu Jan 12 01:16:53 CET 2017 WARN: Establishing SSL connection without server's identity verification is not recommended. According to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ require
Thu Jan 12 01:16:53 CET 2017 WARN: Establishing SSL connection without server's identity verification is not recommended. According to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ require
Thu Jan 12 01:16:53 CET 2017 WARN: Establishing SSL connection without server's identity verification is not recommended. According to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ require

```

Se puede ver que Spring Boot mapea todos los métodos de la REST sin tener que configurar ningún archivo.

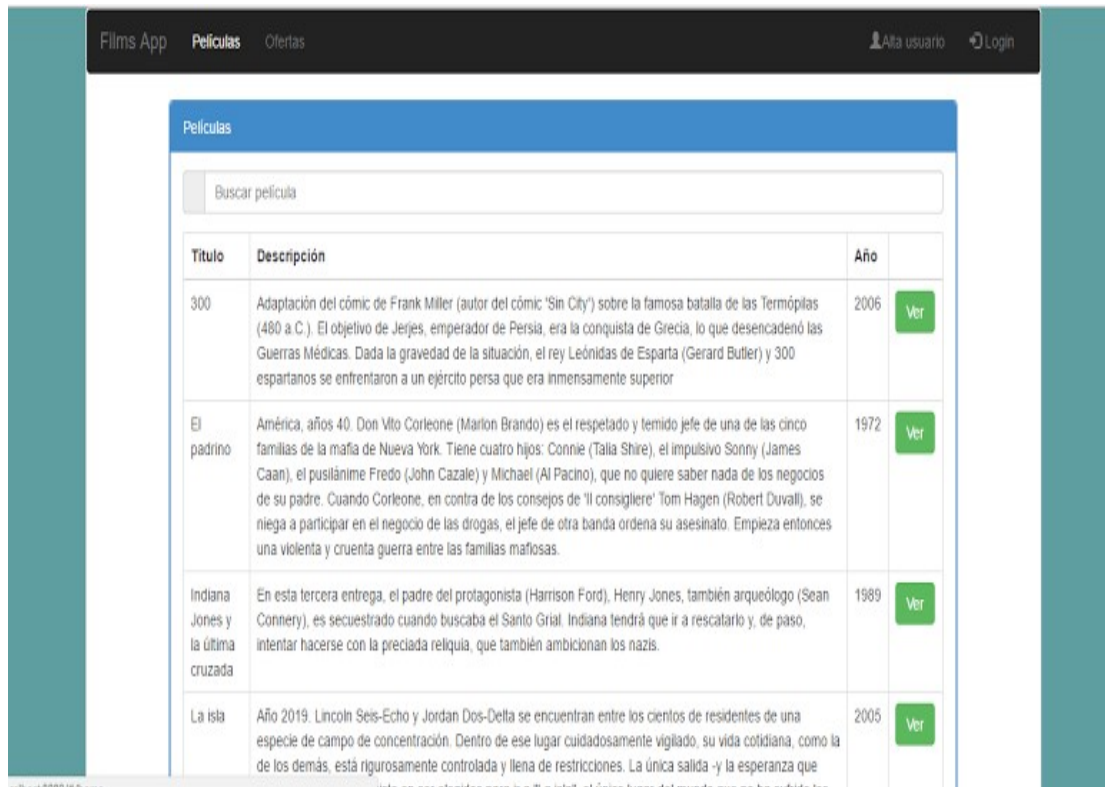


Una vez desplegado todo, nos muestra el mensaje de la clase principal que habíamos indicado. En mi caso: "Start filmsApp".

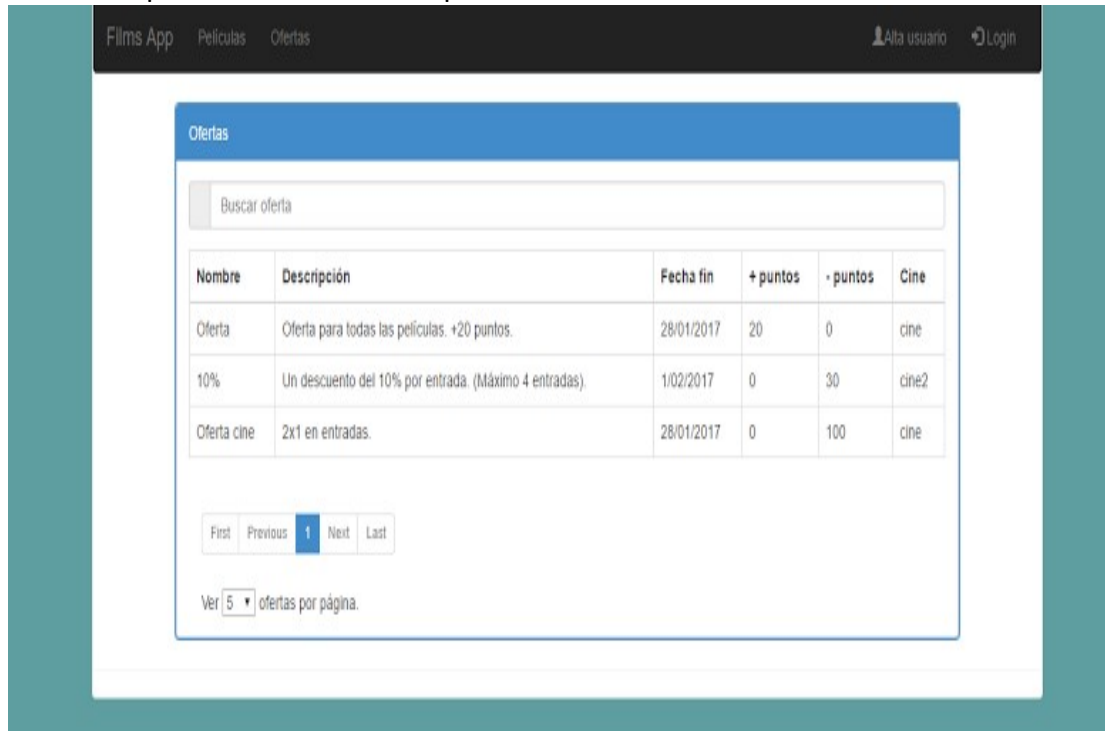


Las funciones de cada pantalla se explican a continuación.


Esta es la pantalla inicial, donde se pueden ver la lista de películas y ver sus datos o disfrutar de ella.



En esta se pueden ver las ofertas publicadas de todos los cines.

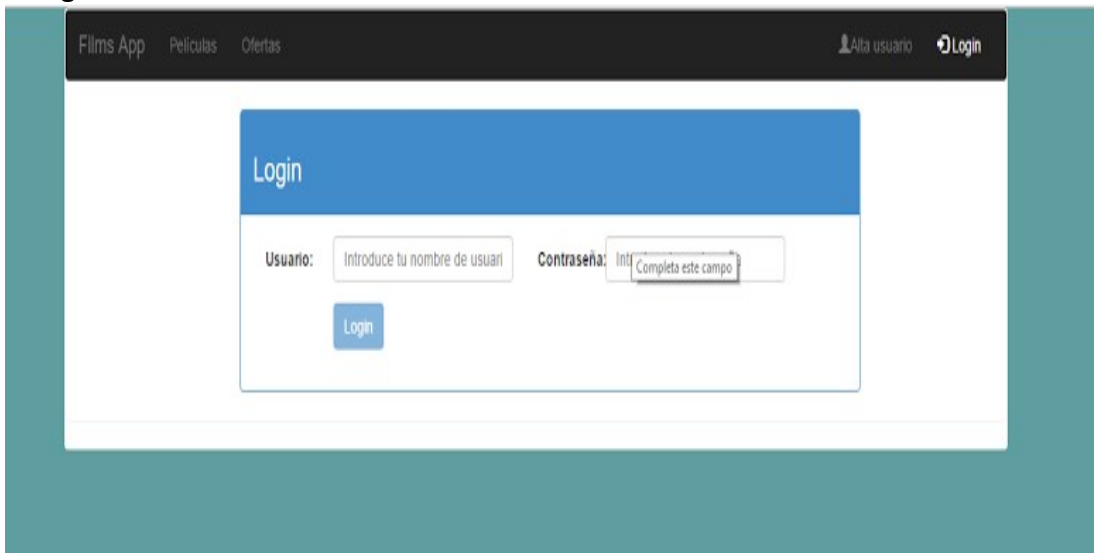


La página de registro de un nuevos suscriptor.



The screenshot shows a registration form titled "Formulario de registro" on a website. The header includes "Films App", "Películas", "Ofertas", and user options "Alta usuario" and "Login". The form fields are arranged in two columns: "Nombre:" (Introduce tu nombre), "Apellidos:" (Introduce tus apellidos), "NIF:" (Introduce tu NIF), "Email:" (Introduce tu email), "Usuario:" (Introduce tu nombre de usuario), and "Contraseña:" (Introduce tu contraseña). Below the fields are three radio button options: "1 mes/10€", "3 mes/17€", and "6 mes/24€". A blue "Submit" button is located at the bottom right of the form area.

El login.



The screenshot shows a login form titled "Login" on a website. The header includes "Films App", "Películas", "Ofertas", and user options "Alta usuario" and "Login". The form has two input fields: "Usuario:" (Introduce tu nombre de usuari) and "Contraseña:" (Introduce tu contraseña). A blue "Login" button is positioned below the "Usuario:" field. A tooltip with the text "Completa este campo" is visible over the "Contraseña:" field.

Esta página es común a todos los usuarios. Aquí se pueden modificar los datos personales o darse de baja.

Esta página sólo es accesible para los cines. Es donde se añaden, eliminan y se aplican las ofertas.

Id	Nombre	Descripción	Fecha fin	+ puntos	- puntos	Aplicar oferta	Eliminar oferta
10	Oferta	Oferta para todas las películas. +20 puntos.	28/01/2017	20	0	Aplicar	Eliminar
13	Oferta cine	2x1 en entradas.	28/01/2017	0	100	Aplicar	Eliminar

Este es el mantenimiento de directores, actores y categorías. Accesible desde un administrador.

			Eliminar
14	John	Dahl	Eliminar
15	Lasse	Hallström	Eliminar
16	Francis	Ford Coppola	Eliminar
17	John	Lasseter	Eliminar
18	Michael	Bay	Eliminar

First Previous 1 Next Last

Ver 20 directores por página.

Nombre: Apellido:

Actores

Buscar actor

Id	Nombre	Apellidos	Eliminar Actor
4	Natalie	Portman	Eliminar

Accesible desde un administrador. Es el mantenimiento de películas.

Films App Películas Ofertas Elementos de películas **Mantenimiento películas** Mantenimiento usuarios Datos usuario Logout

Añadir nueva película

Título: Año: Descripción:

Completar/modificar datos de las películas

Selecciona una película:

Eliminar película

Y estas dos son el mantenimiento de usuarios. Solo accesible por administradores.

Films App Películas Ofertas Elementos de películas Mantenimiento películas **Mantenimiento usuarios** Datos usuario Logout

Cines

Buscar cine

Id	Usuario	Contraseña	Nombre	Email	Web	Dirección	Eliminar
54	cine	cine	Cine 1	cine@filmsapp.com	www.cine.com	calle Del cine nº2	Eliminar
56	cine2	1	cine 2	cine2@filmsapp.com	www.cine2.com	dirección del cine	Eliminar

First Previous 1 Next Last

Ver cines por página.

Suscriptores

Buscar suscriptor

Id	Usuario	Nombre	Apellidos	NIF	Email	Puntos	Fin suscripción	Eliminar
58	Eliminar

Ver administradores por página.

Alta cine

Nombre: Email:

Web: Dirección:

Usuario: Contraseña:

Submit

Alta administrador

Nombre: Apellidos:

NIF: Email:

Usuario: Contraseña:

Submit

8. Bibliografía.