|  |
| --- |
| ‘I.e.s tetuan de las victorias |
| PROYECTO FIN DE GRADO |
|  |
|  |
| **Raquel Sadornil Garzón y Diego Joaquín Chipana Reyna** |
| **07/06/2019** |



|  |
| --- |
|  |

Contenido

[Introducción 2](#_Toc11247717)

[Descripción 3](#_Toc11247718)

[Estructura de aplicación web 3](#_Toc11247719)

[Fundamentos 5](#_Toc11247720)

[Segmentación de clientes 5](#_Toc11247721)

[Tipos de usuarios 5](#_Toc11247722)

[Tecnologías usadas 6](#_Toc11247723)

[Manual de Spring 6](#_Toc11247724)

[Manual de Angular 11](#_Toc11247725)

[Otras tecnologías usadas 15](#_Toc11247726)

[Tareas 16](#_Toc11247727)

[Diagramas 17](#_Toc11247728)

[E/R 17](#_Toc11247729)

[De Casos de uso 18](#_Toc11247730)

[De Navegación 26](#_Toc11247731)

[De Archivos 27](#_Toc11247732)

[Capturas de código llamativo 27](#_Toc11247733)

[Mejoras del proyecto 34](#_Toc11247734)

[Conclusiones 35](#_Toc11247735)

[Bibliografía 36](#_Toc11247736)

# Introducción

Este trabajo va a consistir en la elaboración y explicación de una aplicación web en la cual vamos a usar tecnologías vistas en el periodo de formación en el centro de trabajo y tecnologías vistas durante los dos cursos del ciclo formativo de Desarrollo de Aplicaciones Web.

En la primera parte del documento expondremos de qué trata nuestra aplicación web, a qué tipo de clientes-usuarios está destinado y cuáles son las tecnologías que hemos usado para el desarrollo.

Después mostraremos un diagrama con las tareas que hemos realizado y el tiempo dedicado a cada tarea y las dificultades que hemos ido encontrando al realizar estas tareas.

En la tercera parte del documento estarán unas capturas de pantalla con el código que nos ha parecido más relevante junto a una breve explicación de la funcionalidad del código.

En la siguiente parte se encontraran los distintos diagramas de la aplicación web como son el diagrama de entidad relación, diagrama de casos de uso, diagrama de navegación…

En la última parte del documento se encontrará una conclusión y la bibliografía.

## Descripción

La aplicación web que hemos programado sirve para consultar información sobre las películas que estén guardadas en una base de datos, esta información contendrá tanto información sobre las películas en sí mismas como de los actores principales y directores que estén relacionados con esas películas.

A través de la aplicación podremos ver también las críticas que hagan los usuarios que estén registrados en la base de datos, pero únicamente podrán añadir películas y actores los usuarios que tengan el rol de administrador ya que este rol tiene más privilegios. El resto de usuarios tendrán la oportunidad de proponer películas que no estén registradas en nuestra base de datos y que sean interesantes.

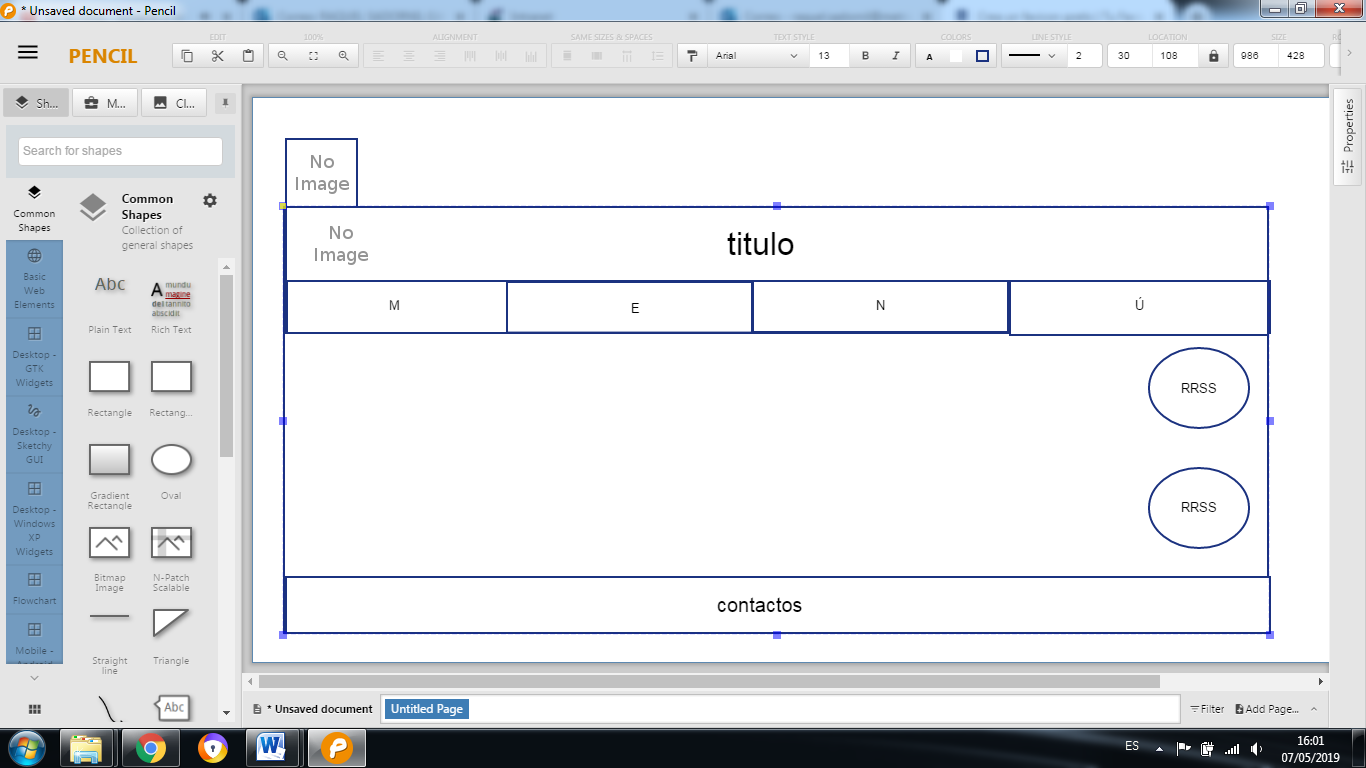
La información que se puede consultar de las películas es el título, la portada, el género o los géneros a los que pertenezca la película, la duración, el director, los actores principales, los premios que ha conseguido la película, la fecha de estreno, el presupuesto y las reseñas anteriormente mencionadas.

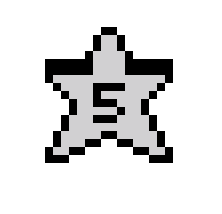
La información de los actores y directores que se puede consultar es el nombre y apellidos de la persona, fecha de nacimiento y defunción (si la hay), lugar de nacimiento y una fotografía de la persona más o menos actual.

### Estructura de aplicación web

Todas las páginas que componen la aplicación mantienen una estructura similar ya que tienen de componentes en común:

* En la parte superior izquierda se encuentra el icono que cuando se pulsa sobre él te redirige a la página principal, en el centro de la parte superior se encuentra el título de la aplicación.
* Debajo del título y el icono se encuentra el menú que te redirige a las distintas partes de la aplicación.
* En el cuerpo de la aplicación en la parte derecha se encuentra unos iconos de las redes sociales.
* En el píe de cada página se encuentran el correo electrónico de los administradores para poder contactar con ellos si surge algún problema durante la utilización de la aplicación web.



El favicon es un elemento importante en las aplicaciones web porque sirve para identificar más fácilmente las páginas web por eso este es el favicon que tiene nuestra aplicación web, lo hemos diseñado nosotros mismos aunque hay en páginas web donde puedes comprarlos. El favicon es una estrella con un 5 dentro ya que nuestra aplicación se llama “ Five stars” que significa “cinco estrellas”.

 El icono que hemos colocado en el título es el de la empresa en la cual hemos hecho las prácticas, ya que hemos usado en la aplicación las tecnologías aprendidas en el periodo que hemos estado.

## Fundamentos

Nuestra idea de desarrollar una aplicación web, que nos permita consultar datos sobre películas, surgió porque nos dimos cuenta que el cine es una forma de ocio muy extendida y por tanto para la gente será útil tener en una misma página los datos más relevantes de las películas y de las personas que participan en ella. Además damos la oportunidad de que los usuarios den su opinión de las películas y sugieran otras películas que no estén en la aplicación web que les parezcan interesantes haciéndoles partícipes de nuestra aplicación a los usuarios.

## Segmentación de clientes

La aplicación web que hemos diseñado está orientada a todo tipo de usuarios, ya que es algo muy general, porque tanto las personas jóvenes como las adultas ven películas para entretenerse y actualmente la tecnología está siendo usada por todas las generaciones.

Esta aplicación la hemos diseñado pensando en que sea fácil de usar para que la mayoría de las personas puedan darle utilidad ya que si diseñas una aplicación de gran complejidad en su uso, la gente decide buscar otra aplicación más sencilla en el mercado y actualmente existe mucha competencia porque hay muchas aplicaciones que hacen la misma funcionalidad cada una de ellas con unas ventajas y desventajas. También en el diseño de la aplicación hemos tenido en cuenta que sea fluida y no tarde mucho en cargar para que los usuarios no se impacienten porque eso sería una desventaja.

## Tipos de usuarios

En nuestra aplicación va haber dos tipos de roles tipo usuario normal o usuario administrador, según el tipo de usuario que se sea se tiene unos privilegios u otros (por ejemplo, el administrador es el único que puede añadir películas).

## Tecnologías usadas

### Manual de Spring

#### - Java

Java es un lenguaje de programación orientada a objetos que surgió en los años noventa.

El lenguaje java se creó con cinco objetivos principales:

1. Usar el paradigma de la programación orientada a objetos.
2. Permitir la ejecución en distintos sistemas operativos.
3. Soporte para trabajo en red.
4. Ejecutar código en sistemas remotos de forma segura.
5. Que sea sencillo de usar.

#### - Spring

Para empezar a crear un proyecto spring MVC primero debemos conocer Spring Core, el cual es el motor de spring boot, ya que utilizaremos inyección de dependencias, el manejo de beans, inyección automática que es la base de este framework.

Vamos a aprender a construir de un proyecto MVC con Spring. Para esto hay que primero crear un proyecto Spring y tener claro el modelo MVC.

El Modelo Vista Controlador (MVC) es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

Se trata de un modelo muy maduro y que ha demostrado su validez a lo largo de los años en todo tipo de aplicaciones, y sobre multitud de lenguajes y plataformas de desarrollo.

El Modelo que contiene una representación de los datos que maneja el sistema, su lógica de negocio, y sus mecanismos de persistencia. Se hace en Spring.

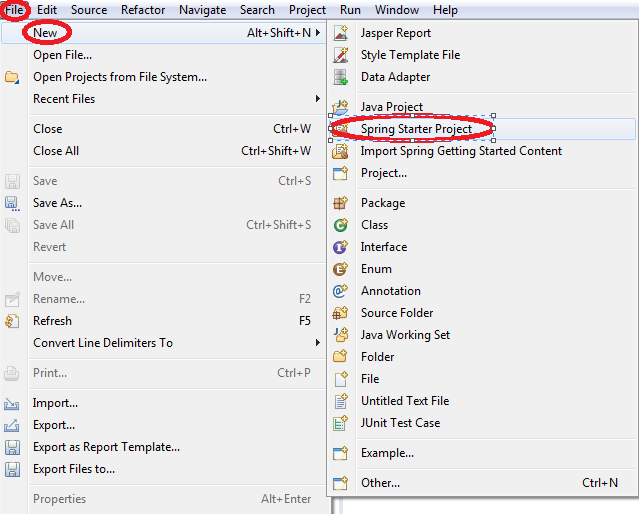
La Vista, o interfaz de usuario, que compone la información que se envía al cliente y los mecanismos interacción con éste. Se hace en Angular.

El Controlador, que actúa como intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno. Se hace en Spring.

***Cómo crear un proyecto Spring***

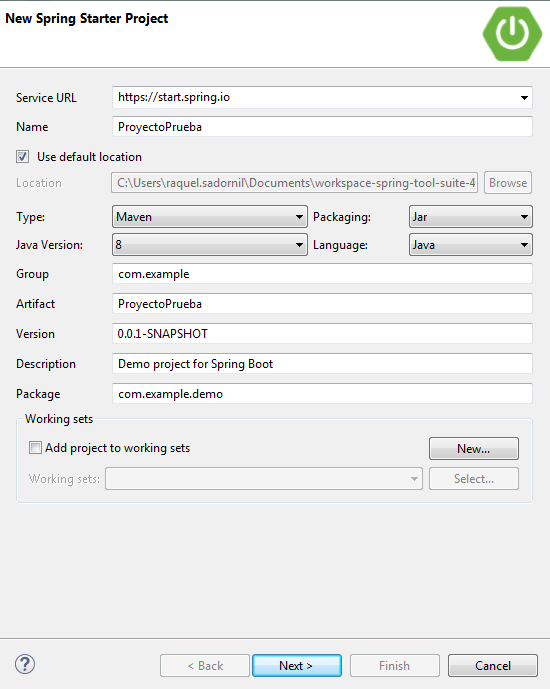


1. Creamos un nuevo proyecto



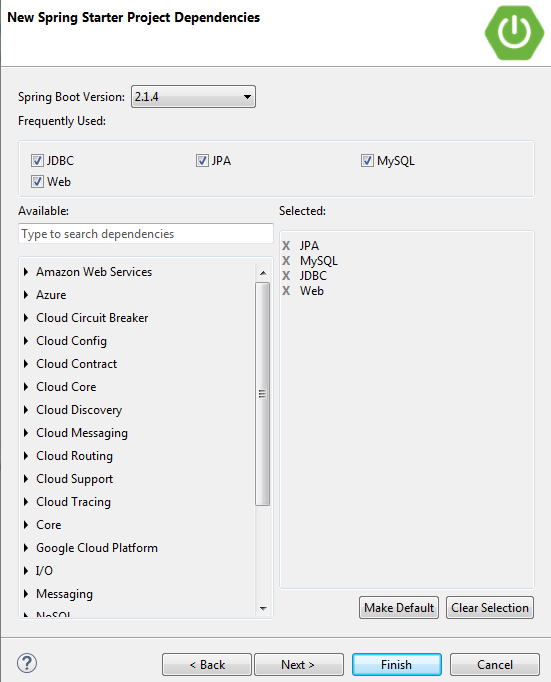
Vamos a File, pinchamos a New y elegimos la opción de Spring Starter Project.

2. Elegimos las opciones que necesitemos



Elegimos el nombre del proyecto y elegimos si queremos un proyecto tipo Maven o Gradle.

En este caso elegimos Maven ya que nos permite descargar las dependencias que necesitemos directamente.



Elegimos las dependencias que queramos. Nosotros cogeremos JDBC, JPA, MySQL y WEB.

Con esto finalizaríamos nuestra creación, ahora solo falta codificar nuestros controladores y modelos.

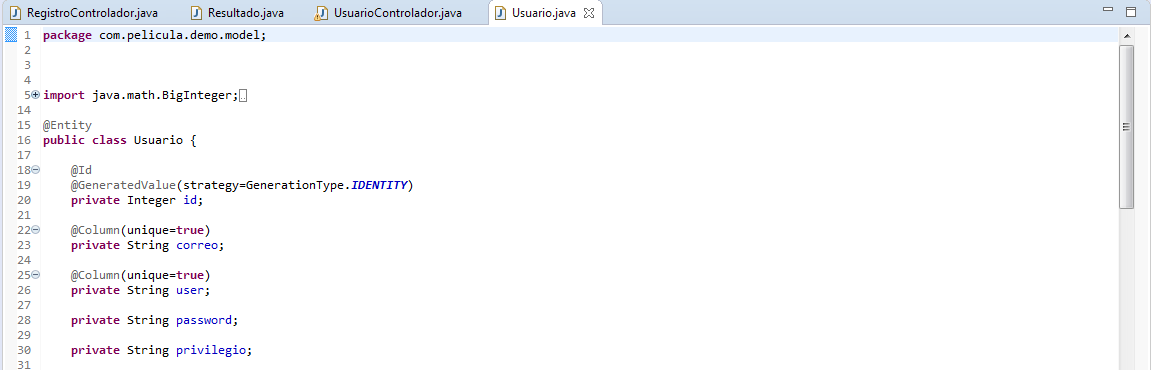
Una vez hecho esto, se genera una clase java el cual permite ejecutar la aplicación y aparte generamos un pom.xml en el cual está nuestras dependencias y podemos modificarlo para poner más.

***Estructura de una API***

Nuestra API consistirá en tres partes: controladores, modelos y repositorios.

Es muy aconsejable empezar por el modelo, ya que tienes que saber la estructura de los objetos para poder hacer peticiones con este en el controlador.

Para crear los objetos es necesario conocer las anotaciones de hibernate, ya que, si nos olvidamos alguna anotación, nuestra aplicación no funcionará si intentamos utilizar un objeto que no tenga las anotaciones correctas.

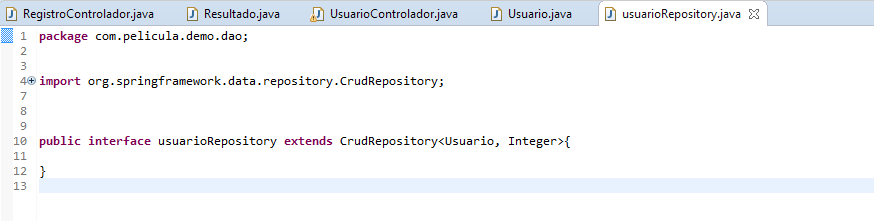


Como se puede observar es un objeto normal pero con unas cuantas anotaciones(las cuales pueden ser más complejas) para poder crear correctamente una entidad.

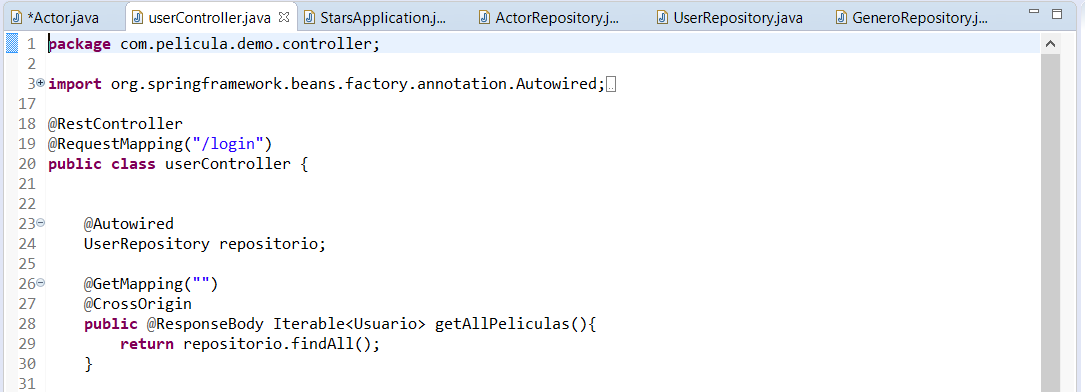
Una vez creado los modelos, pasaremos a crear repositorios.

Hay que destacar que los repositorios son interfaces y no clases, no confundir.

Extenderemos la interface de CrudRepository o JpaRepository dependiendo de lo que queramos, con CrudRepository nos será suficiente esta vez.



Una vez creado el repositorio y el modelo es hora de ponernos en marcha con los controladores, un controlador básico sería este:



Como podemos observar le añadimos a la clase dos anotaciones. @RestController(para indicar que este es nuestro controlador de nuestra API) y @RequestMapping(Con esta anotación se le indica la ruta con la que se va a hacer las peticiones).

Tenemos que abrir repositorios con la anotación @Autowired.

Y por último creamos un método sencillo que te devuelve todas las películas, se le añade dos anotaciones: @GetMapping(para indicarle que va a ser una petición GET y también la ruta) y @CrossOrigin(para que se accesible desde angular).

### Manual de Angular



Angular es un **framework** multiplataforma para crear aplicaciones del lado cliente en HTML y JavaScript.

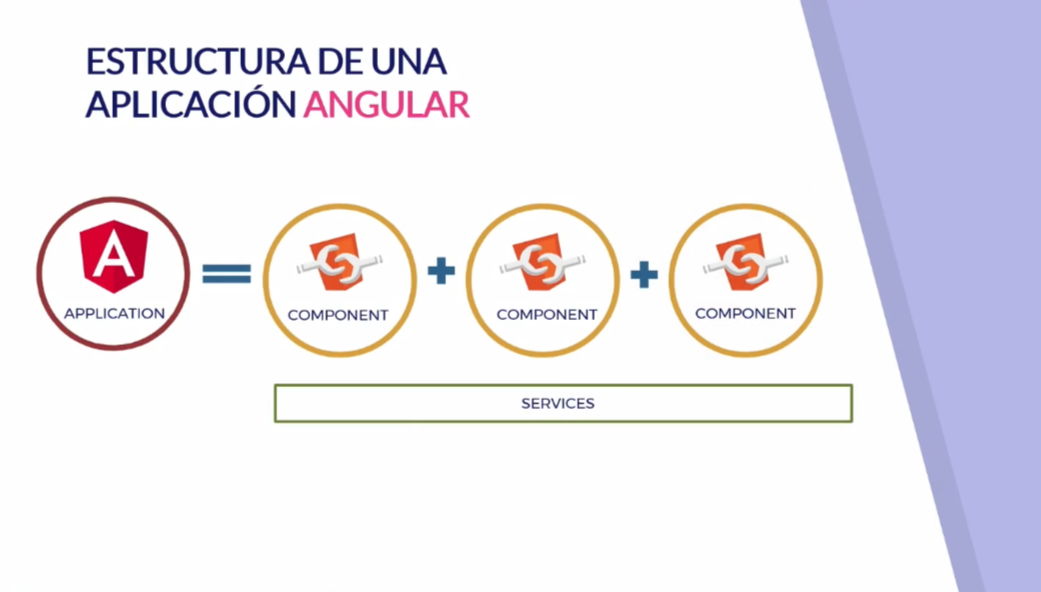
Las características principales son:

**Angular es dogmático**: Toma muchas decisiones por defecto como puede ser la conectividad de red, la elección del idioma, herramientas de compilación,…

**Angular es escalable**: Permite la participación de muchos desarrolladores en la construcción de una aplicación.

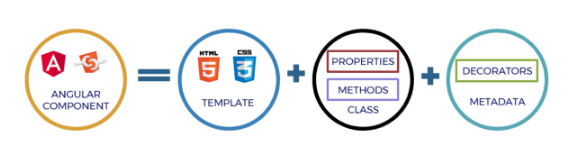
**Angular tiene un fuerte ecosistema**: Es Open Source pero es mantenido y desarrollado por Google. Existen multitud de herramientas y bibliotecas reutilizables en internet.

***Estructura de una aplicación angular***



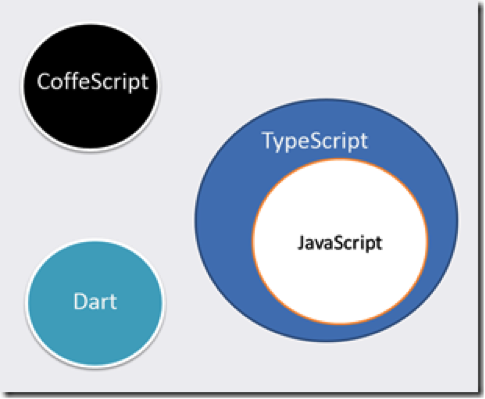
**¿Qué es un componente?**

Un componente es una etiqueta HTML personalizada, reutilizable y encapsulada. A su vez, un componente se divide en:

1. **Template**: HTML y CSS.
2. **Class**: Clase que define funcionalidad con propiedades y métodos.
3. **Metadaos**: Se indica a Angular qué características y propiedades tiene el componente.

El **lenguaje** que usa **Angular** es **TypeScript**.

TypeScript es un lenguaje de programación libre y de código abierto desarrollado y mantenido por Microsoft. Lo que más identifica a este lenguaje de programación es que es un superconjunto de JavaScript.



Puede hacer lo mismo que JavaScript y aún más. Puedes picar código JavaScript en TypeScript sin ningún problema pero no estaríamos sacando el máximo provecho porque lo que se distingue es que es un lenguaje tipado.

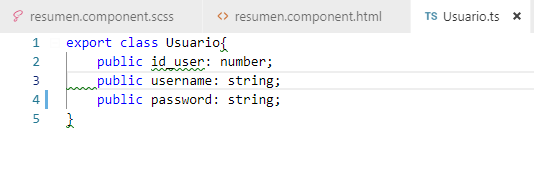
***Cómo conectar un servicio RESTful***

Para conectar Angular a tu servicio en Spring lo primero que tienes que hacer es crear un componente con el comando “ng g componente [nombreDelComponente]”. Nuestro componente estará en la ruta “src/app”.

Lo segundo que hay que hacer es crear un servicio con el comando “ng g service [nombreDelServicio]”(preferiblemente dentro de cada componente y no un servicio para todos los componentes de nuestro proyecto).

Una vez hecho esto, en nuestro módulo principal tenemos que importar todo lo que necesitemos, en nuestro caso FormsModule y HttpClientModule.

También crearemos una carpeta que sea los modelos y crearemos archivos typescript que van a ser objetos que usaremos en nuestro proyecto.



Una vez creado el modelo de nuestros objetos, tenemos que completar el servicio de cada componente, aquí es donde se envía la información a tu servicio RESTful.



Para hacer esto tenemos que crear una cabecera si es necesario y crear una variable **ruta** para poder reutilizarlo. En el constructor definimos una variable **http** de tipo **HttpClient** que es lo que te permite llamar al servicio.

Después de esto, hay que hacer los métodos que sean necesario en tu página web, en este caso un **SignIn** que te devuelve el objeto Usuario\_Role(previamente creado en nuestra carpeta modelo) y a nuestra llamada le pasaremos un objeto **Usuario** para que nuestro servicio compruebe si este usuario existe.

Después de completar nuestro servicio, tenemos que completar nuestro componente(principalmente HTML y TS).

El HTML en nuestro caso será un simple login.



El typescript de nuestro componente quedaría así.



En el método de validar, se hace llamada a nuestro servicio el cual tiene la conexión al servicio. También se hace las comprobaciones en caso de que sea incorrecto algún dato o esté vacío. Si todo está bien se harán uso de sesiones en angular.

En nuestro componente también tendremos un archivo CSS, SCSS o SASS. Podremos elegir entre estos tres tipos de archivos para poder modificar el estilo de nuestra aplicación web.

### Otras tecnologías usadas

#### - SQL



**SQL** es un lenguaje de programación desarrollado por la empresa multinacional estadunidense “IBM”, diseñado para administrar, y recuperar información de [sistemas de gestión de bases de datos relacionales](https://es.wikipedia.org/wiki/Sistema_de_gesti%C3%B3n_de_bases_de_datos_relacionales).

#### - GitHub

****

**GitHub** es una plataforma de desarrollo colaborativo para alojar proyectos utilizando el sistema de [control de versiones](https://es.wikipedia.org/wiki/Control_de_versiones) [Git](https://es.wikipedia.org/wiki/Git). El código de los proyectos alojados en GitHub se almacena típicamente de forma [pública](https://es.wikipedia.org/wiki/C%C3%B3digo_abierto), aunque utilizando una cuenta de pago, también permite hospedar repositorios privados.

#### - Postman



**Postman** consiste en una herramienta dirigida a desarrolladores web que permite realizar peticiones HTTP a cualquier API. Postman es muy útil a la hora de programar y hacer pruebas, ya que nos ofrece la posibilidad de comprobar el correcto funcionamiento de nuestra aplicación.

#### - Bootstrap

****

**Bootstrap** es un framework CSS y Javascript diseñado para la creación de interfaces limpias y con un diseño responsive de manera que los usuarios pueden crear prácticamente cualquier tipo de sitio web fácilmente.

# Tareas

|  |  |  |  |
| --- | --- | --- | --- |
| **Descripción** | | **Tiempo aproximado** | **Tiempo real** |
| **Análisis/Memoria** | | | |
| **Redactar la idea** | | 1H | 1H 30min |
| **Segmentación de los clientes y tipos de usuario** | | 1H | 1H |
| **División de tareas** | | 1H | 1H |
| **Manuales de Spring y Angular** | | 5H | 7H |
| **Diagrama** | **E/R** | 2H | 2H |
| **Casos de usos** | 2H | 3H |
| **Navegación** | 1H | 30min |
| **Componentes** | 30min | 30min |
| **Archivos** | 15min | 5min |
|  | | | |
| **Código** | | | |
| **Aprendizaje spring** | | 10H | 20H |
| **Aprendizaje angular** | | 15H | 30H |
| **Base de dato** | | 1H | 2H |
| **Código Spring/Servicio** | | 5H | 5H |
| **Código Angular** | | 7H | 10H |
|  | | | |
| **Total: 83H 35min** | | | |

En esta tabla hemos expresado las distintas tareas que hemos tenido que hacer para el proyecto y hemos decidido poner una columna con el tiempo que hemos aproximado para cada tarea, ya que en los proyecto reales de las empresas la estimación de tiempo juega un papel fundamental, porque si las estimaciones se alejan mucho de la realidad te encuentras con diversos problemas como por ejemplo que te comprometas con un cliente en la entrega de un proyecto y luego te falte tiempo.

# Diagramas

Los diagramas son una herramienta muy importante en el mundo del desarrollo de aplicaciones porque sirven para representar gráficamente las bases de ellas y así facilitan la organización y el entendimiento de las aplicaciones.

## E/R

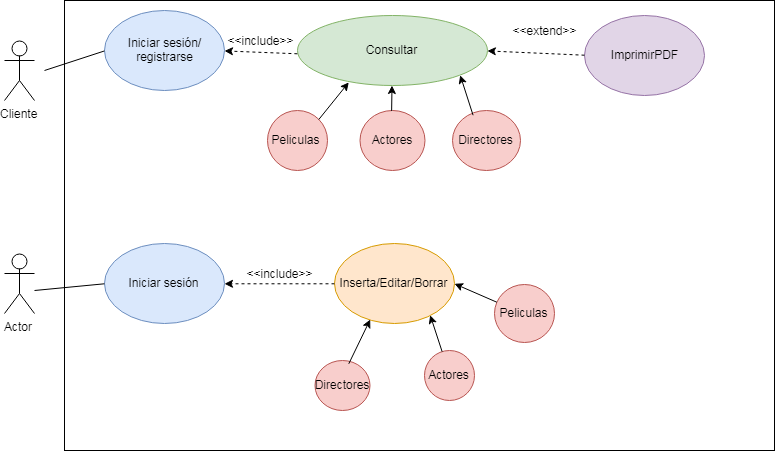
El diagrama entidad relación es un tipo de diagrama que describen la información que trata un sistema de información ilustrando cómo las "entidades", las personas, objetos o conceptos, se relacionan entre sí dentro de un sistema.

Imagen que contiene texto, mapa

Descripción generada automáticamente

## De Casos de uso

Un diagrama de casos de uso representa a un sistema o subsistema como un conjunto de interacciones que se desarrollarán entre casos de uso y sus actores (que son los personajes o entidades que participarán en un diagrama de caso de uso) en respuesta a un evento que inicia un actor principal. Un caso de uso es la descripción de una acción o actividad.Representa una funcionalidad del sistema, algo que reporta un resultado de valor al actor que inicia el caso de uso.



|  |
| --- |
| **Nombre: RegistrarCliente**  **ID: CU-1** |
| **Descripción:**  El cliente se da de alta para poder usar la aplicación web. |
| **Actores:** Cliente. |
| **Precondiciones:**  El cliente no está registrado en el sistema. |
| **Curso normal del caso de uso:**   1. Acceder al sistema. 2. Acceder al enlace de registrar usuario. 3. Rellenar los campos del formulario correctamente. 4. Enviar los datos al servidor. |
| **Postcondiciones:**  El cliente es dado de alta. |
| **Alternativas:**  3. El usuario rellena mal los campos.  3.1 Salta una alerta.  4. El usuario vuelve a introducir los datos pero de forma correcta.  5. Enviar los datos al servidor. |

|  |
| --- |
| **Nombre: ImprimirPDF**  **ID: CU-2** |
| **Descripción:**  El cliente solicita la descarga del pdf, pulsando a un botón que aparecerá en la parte inferior derecha, que contiene los datos de la búsqueda que ha realizado. |
| **Actores:** Cliente. |
| **Precondiciones:**  Que el cliente esté registrado y que hay consultado alguna película o algún actor o algún director. |
| **Curso normal del caso de uso:**   1. Iniciar sesión. 2. El usuario consultará una película/actor/director. 3. Pulsar en el botón que ponga “descargar PDF”. 4. Se descarga el PDF. |
| **Postcondiciones:**  El cliente ya tiene en su dispositivo el PDF. |
| **Alternativas:**  2. El buscador que esté usando el cliente tenga bloqueada la opción de pantallas emergentes o la opción de descargar PDF. |

|  |
| --- |
| **Nombre: ConsultarPelículas**  **ID: CU-3** |
| **Descripción:**  El usuario que ya ha iniciado sesión busca una película en la lista que tenemos. |
| **Actores:** Cliente. |
| **Precondiciones:**  El usuario tiene que estar dado de alta y haber iniciado sesión. |
| **Curso normal del caso de uso:**   1. El usuario inicia sesión. 2. Introduce una película en el buscador. 3. Hace click en la película. |
| **Postcondiciones:**  El usuario accede a la información de la película. |
| **Alternativas:**  2. Introduce un título que no está registrado.  3. El servicio no muestra ninguna película.  4. Volver al paso 2. |

|  |
| --- |
| **Nombre: ConsultarActoresYDirectores**  **ID: CU-4** |
| **Descripción:**  El usuario que ya ha iniciado sesión busca un actor o director en la sección de Actores y directores. |
| **Actores:** Cliente. |
| **Precondiciones:**  El usuario tiene que estar dado de alta y haber iniciado sesión. |
| **Curso normal del caso de uso:**   1. El usuario inicia sesión. 2. Accede a la sección de Actores y Directores. 3. Tiene la opción de filtrar por el buscador. 4. Hace click en el director o actor que deseas. |
| **Postcondiciones:**  El usuario accede a la información de la persona. |
| **Alternativas:**  2. Introduce un nombre que no está registrado.  3. El servicio no muestra ningún resultado.  4. Volver al paso 2. |

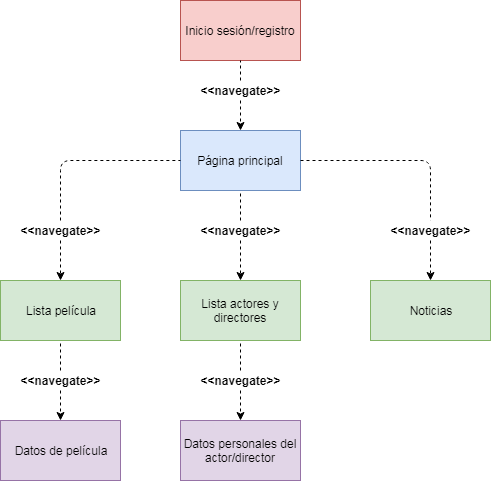
|  |
| --- |
| **Nombre: ConsultarNoticias**  **ID: CU-5** |
| **Descripción:**  El cliente dispondrá de esta sección en la cual se mostraran noticias actuales relacionadas con el cine. |
| **Actores:** Cliente. |
| **Precondiciones:**  Se requiere que el cliente esté registrado. |
| **Curso normal del caso de uso:**   1. Iniciar sesión. 2. El usuario accede a la sección de noticias. |
| **Postcondiciones:**  El usuario verá las noticias que estén disponibles en esa sección. |

|  |
| --- |
| **Nombre: InsertarPeliculas**  **ID: CU-7** |
| **Descripción:**  El administrador inserta películas en la base de datos para que los clientes puedan consultarlas. |
| **Actores:** Administrador. |
| **Precondiciones:**  Estar registrado como administrador y que la película no esté ya registrada. |
| **Curso normal del caso de uso:**   1. Iniciar sesión como administrador. 2. Rellenar formulario para insertar la película. |
| **Postcondiciones:**  La película se inserta en la base de datos y los usuarios pueden consultarla. |
| **Alternativas:**  2. Algún campo no esté rellenado adecuadamente. |

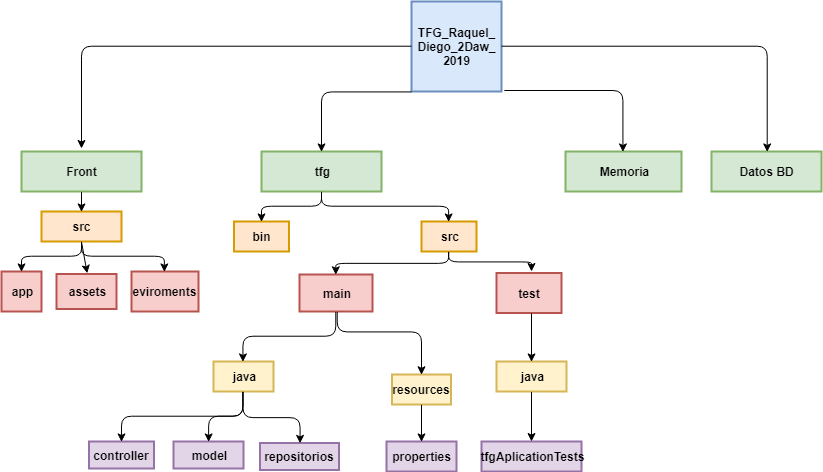
|  |
| --- |
| **Nombre: InsertarNoticias**  **ID: CU-8** |
| **Descripción:**  El administrador inserta noticias a la base de datos para poder publicarlas. |
| **Actores:** Administrador. |
| **Precondiciones:**  Ser administrador de la web. |
| **Curso normal del caso de uso:**   1. Iniciar sesión como administrador. 2. Ir a la sección de noticias. 3. Acceder al formulario de añadir película. 4. Rellenar los campos. 5. Enviar a la base de datos |
| **Postcondiciones:**  La noticia se publica en la página y es visible para todos. |

|  |
| --- |
| **Nombre: InsertarActor/Director**  **ID: CU-9** |
| **Descripción:**  El administrador inserta actores/directores en la base de datos para que los clientes puedan consultarlos. |
| **Actores:** Administrador. |
| **Precondiciones:**  Estar registrado como administrador y que el actor/director no esté ya registrada. |
| **Curso normal del caso de uso:**   1. Iniciar sesión como administrador. 2. Acceder a la sección de Actores y Directores. 3. Rellenar formulario para insertar el actor/director. |
| **Postcondiciones:**  El actor/director se inserta en la base de datos y los usuarios pueden consultarlo. |
| **Alternativas:**  2. Algún campo no esté rellenado adecuadamente. |

## De Navegación

Los mapas de navegación son una representación esquemática de la estructura del hipertexto, indicando los principales conceptos de la información y las interrelaciones que existen entre ellos, cuyo objetivo es orientar al usuario durante el uso de la aplicación o para facilitarle un acceso directo al lugar que le interese. Reflejará la estructura de la *web* por medio de [enlaces](http://www.hipertexto.info/documentos/enlaces.htm) a los [nodos](http://www.hipertexto.info/documentos/nodos.htm) principales, y éstos también pueden desarrollarse para mostrar los subnodos.

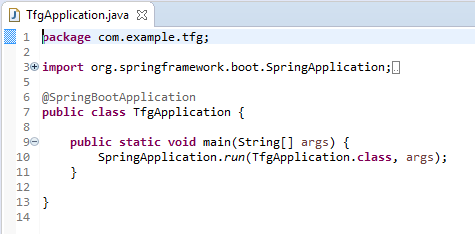
## De Archivos

Un diagrama de archivos es la representación de las carpetas y archivos que componen de un proyecto gracias a esto es más fácil ver la organización de la aplicación.

# Capturas de código llamativo

Empezaremos mostrando capturas del código de Spring(BackEnd) para mostrar cómo hemos hecho el controlador de nuestro servicio RESTful y el DAO. Más adelante mostraremos capturas del controlador frontal y de la vista (FrontEnd).

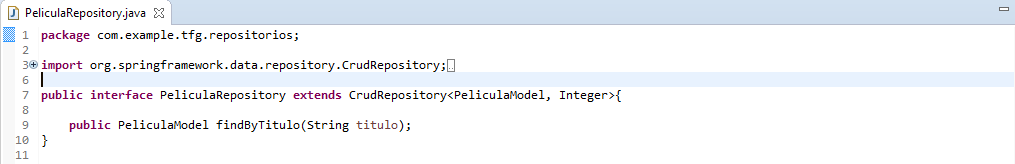
**1.** La primera captura de código que vamos a poner va al TfgApplication que se crea automáticamente cuando creas el proyecto. Esto sirve para poder ejecutar la aplicación spring boot App.



Esta anotación indica que es una aplicación Spring. Puede haber más anotaciones entre las cuales se encuentran @EntityScan, @ComponentScan y @EnableJpaRepositories los cuales son obligatorios si los paquetes de modelo, DAO y controladores no extienden del paquete en el que está TfgApplication.

**2.** La segunda captura que ponemos es la del repositorio del objeto película, esta interfaz te permite realizar acciones en la base de datos mediante dos tipos de repositorios: CRUD y JPA, da igual cuál elijas pero en nuestro caso solo necesitamos el repositorio de CRUD.

Esta interfaz nos permite crear métodos los cuales pueden estar vacíos ya que crea la sentencia SQL automáticamente.

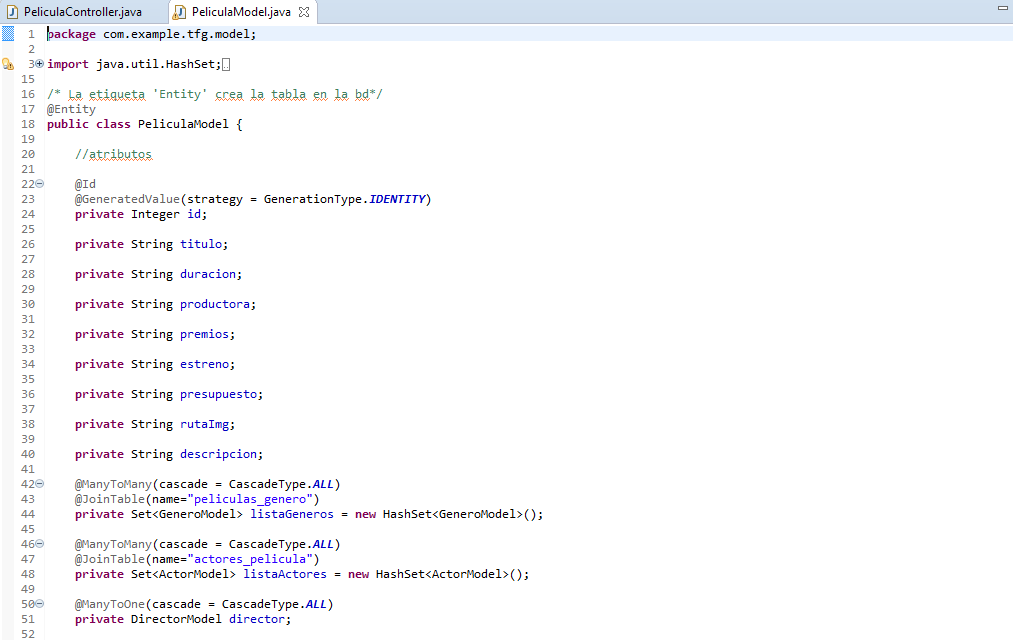


Aquí se pone la entidad al que va a afectar los métodos de nuestro repositorio.

Aquí va el tipo de dato del que es el Id de la entidad.

Como podemos ver aquí, creamos el método vacío. Solo tenemos que poner “findBy”+ campo por el que quieres buscar.

**3.** Esta vez vamos a poner la captura de la entidad película. Hemos decidido poner esta ya que es el que más nos costó hacer debido a que tiene varias relaciones las cuales pueden dar problemas.



Una relación ManyToMany crea una tabla relación en la base de datos

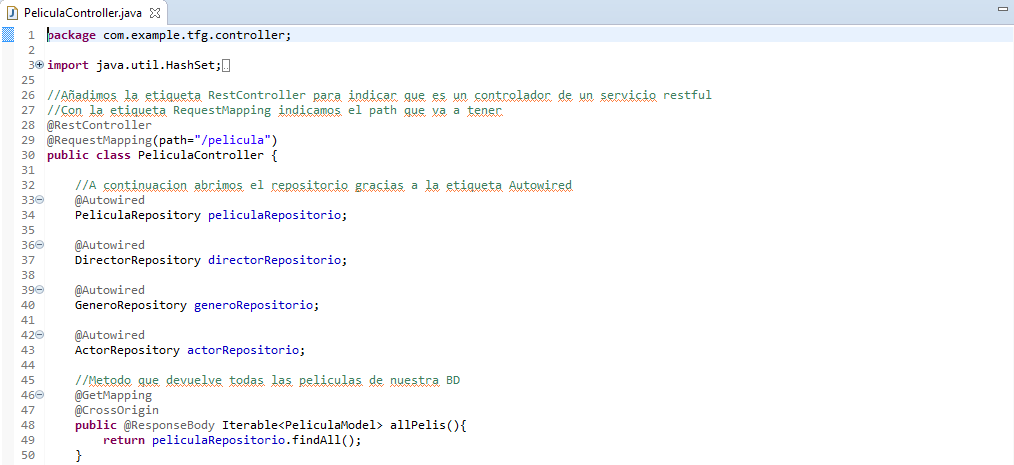
La relación ManyToOne crea un campo en la propia tabla con el id de la entidad que hace referencia.

Las anotaciones indican que es el Id y que es un autoincrement dentro de la tabla

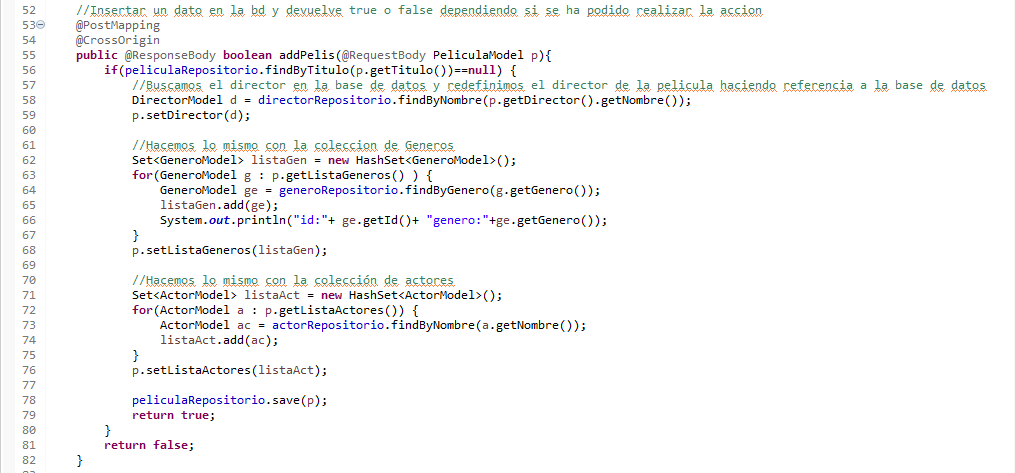


Por último generamos los getters y setter por si nos hace falta en algún momento.

4. Por último mostraremos el controlador de películas con sus correspondientes loggers para imprimir en consola lo que va sucediendo en mi servicio RESTful.



Con la anotación de @Autowired abres el repositorio y puedes trabajar con él.



Este método es algo complejo, para empezar ponemos @PostMapping y @CrossOrigin para indicar que es un método Post y que puedes acceder a este método desde angular. Buscamos si existe la película y en caso de que aún no exista esa película, buscamos los actores, directores y géneros en las otras entidades para luego relacionarlas con películas.

Ahora vamos a ver capturas de **Angular**.

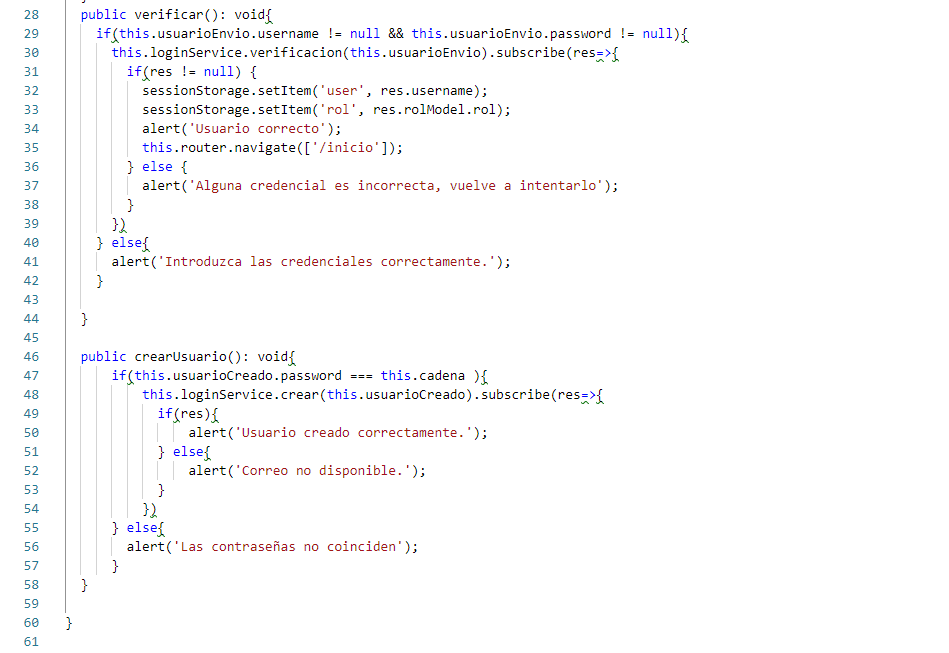
**1.** Primero vamos a ver un servicio, de momento no está acabado pero lo estará para el día del proyecto. Lo importante de esta captura es ver como se conecta con el servidor.



Creamos un método en el cual llamamos a la url del servidor, accedemos a la ruta /create y le enviamos un objeto usuario en forma de JSON.

**2.** Ahora vamos a ver el typescript del componente de login. En este hacemos llamada al servicio anterior para poder recibir datos del servidor. También instanciamos objetos para poder obtener los datos del HTML.





Hace lo mismo pero para crear un usuario. Comprueba si el usuario o correo ya existen y si la contraseña está bien introducida.

Método que verifica si el usuario es correcto y te crea sesiones. Si esto no se cumple, tienes que volver a introducir los datos.

1. Por último veremos el HTML para ver algunos atributos característicos de Angular. Con esto se transmiten valores de formularios o establecerlos.



Establece el valor a nuestro objeto

Clases bootstraps para crear modales

# Mejoras del proyecto

Tras analizar nuestra aplicación hemos pensado como posibles mejoras son:

- **Encriptación MD5** para las contraseñas de los usuarios, ya que en la base de datos las hemos guardado sin codificación y eso es poco seguro porque si no es muy fácil obtener las contraseñas. Nos hemos informado sobre este método de encriptación para implementarlo pero no nos dio tiempo. Para encriptar tenemos que crear un método que reciba por parámetro un String, y mediante una librería transforma la cadena que hemos pasado a una cadena encriptada de 32 caracteres.

- También creemos que sería interesante que los usuarios pudieran **mandar un correo electrónico** directamente desde la aplicación al administrador por si tienen algún problema o da algún fallo la aplicación. Esto se podría implementar con un método en Spring el cual envíe correos con una cuenta predefinida para todos los usuarios y que en el asunto ponga el username del usuario y el contenido sea las dudas o las incidencias que tenga.

- Para la autentificación del usuario que se introduce en el inicio de la aplicación nosotros hemos creado un método pero si implementamos **Spring Security** ya estaría directamente ese método y otros métodos ya que pero para conectarlo a Angular provocaba muchos problemas y por eso no lo hemos implementado.

- Reconocemos que **JUnit** es muy importante para las pruebas unitarias en el proceso de desarrollo de un proyecto pero por falta de tiempo hemos ido probando los módulos a través de Postman que es una herramienta más sencilla de utilizar que JUnit.

# Conclusiones

Este trabajo ha sido muy útil para acostumbrarnos a trabajar en grupo que es como se trabaja en nuestro sector, por lo tanto es un requisito imprescindible para las empresas, y para nuestro desarrollo como programadores web ya que hemos podidos practicar con el proyecto las tecnologías que hemos aprendido en nuestra formación en el centro de trabajo, las cuales son muy demandadas actualmente en el mercado laboral.

También hemos usado, para el proyecto, herramientas aprendidas durante los dos cursos del ciclo como son los diagramas y nos hemos podido dar cuenta de la importancia que tienen en el desarrollo de proyectos como pueden ser aplicaciones web,

Gran parte del tiempo que hemos dedicado al proyecto ha sido para buscar información y esto nos ha ayudado a que seamos más autodidactas que también es una característica muy importante en nuestro perfil profesional.

Entre las tecnologías que hemos usado en el proyecto la que más nos costó aprender e implementar ha sido Angular ya que fue difícil encontrar información y la conexión entre Angular y Spring, además teníamos que hacer muchas pruebas para que la interfaz quedase como queríamos, aunque conseguimos solucionar todos los problemas que iban surgiendo hasta llegar al resultado. En cuanto a Spring no nos costó tanto ya que se basa en Java y teníamos conocimientos de ello.

En general estamos muy satisfechos y contentos con nuestro proyecto por los conocimientos adquiridos mencionados anteriormente, la metodología de trabajo usada (SCRUM) y por el resultado obtenido finalmente.

Por último, gracias al proyecto hemos visto la importancia de lo aprendido durante estos dos años y la relevancia que tiene a la hora de trabajar.

# Bibliografía

**Diagramas:**

Libro: “Entornos de Desarrollo”, ISBN:978-84-1545-297-3 (20/05/2019).

**Tecnologías:**

Typescript:

<https://es.wikipedia.org/wiki/TypeScript>  (29/05/2019)

<https://openwebinars.net/academia/aprende/angular/3541/> (29/05/2019)

Spring:

<https://spring.io/guides> (28/03/2019)

<https://openwebinars.net/academia/aprende/spring-core/> (31/03/2019)

<https://openwebinars.net/academia/aprende/spring-boot/> (01/04/2019)

GITHUB

<https://www.youtube.com/watch?v=H03E8_HImAU> (25/04/2019)

STACKOVERFLOW (Para dudas)

https://stackoverflow.com/

ANGULAR+SPRING

<https://github.com/eugenp/tutorials/tree/master/spring-security-angular> (05/04/2019)

Logback

<http://chuwiki.chuidiang.org/index.php?title=Primeros_pasos_con_logback> (28/03/2019)

Logger.

<https://blog.elhacker.net/2013/01/log-binario-en-mysql.html> (28/03/2019)

ENCRIPTAR MD5

<https://www.yoelprogramador.com/como-encriptar-contrasenas-en-md5-en-java/> (15/05/2019)

Junit

<http://www.jtech.ua.es/j2ee/publico/lja-2012-13/sesion04-apuntes.html>  (29/05/2019)