

PROGRAMACIÓN

FRAMEWORK SPRING

PROYECTO FINAL

ESCENARIO

Desde Tokio School te proponemos la construcción de un portal dirigido a las personas aficionadas al cine donde daremos de alta nuestras películas favoritas y solicitaremos la colaboración de otros cinéfilos y cinéfilas para que las puntúen.

Pondréis en práctica todo lo que hemos aprendido sobre **Spring**. Podréis utilizar una serie de recursos que os proporcionaremos para facilitar la realización del proyecto; en este documento encontraréis las especificaciones del sistema a desarrollar y algunos "*hints*" que os resultaran útiles para la implementación.

ESPECIFICACIONES

Os recomendamos que utilicéis las **últimas versiones disponibles** de Spring y de todas las librerías que tengáis planteado utilizar.

A modo de ejemplo, en el momento de preparar esta documentación, las versiones que se han utilizado para las dependencias principales son las siguientes.

Versiones	
<i>Java</i>	21
<i>Spring Boot</i>	3.1.5
<i>Maven</i>	3.9.5
<i>MySQL</i>	8.0
<i>Bootstrap</i>	5.3.2
<i>Lombok</i>	1.18.30

DESARROLLO

El proyecto constará de tres partes, todas relacionadas entre sí:

1. **Aplicación WEB:** Una web que permita a los usuarios interactuar con ella mediante una interface gráfica.
2. **Servicio REST:** Una API Rest de prestación de servicios basada en interacciones JSON y securizada.
3. **Proceso BATCH:** Un proceso *batch* que se ejecute de manera periódica para realizar algunas acciones.



En los apartados siguientes haremos una descripción de los servicios que cada aplicación tiene que prestar. Todas las aplicaciones van a necesitar base de datos, bien sea MySQL o MariaDB. Por simplicidad puedes usar el mismo esquema de base de datos para todas.

HINTS

- *Desarrolla el proyecto en un proyecto **multi module de Maven** usando tu IDE escogido. Tendrás un **parent pom** en el que puedes definir las dependencias comunes y luego un módulo para cada aplicación. Te será más sencillo desarrollar las tres aplicaciones de esta manera que tener tres proyectos separados.*
- *Desarrolla el proyecto usando archivos de propiedades desde el principio. Hará que el proyecto sea más fácilmente configurable. Puedes usar el formato **yaml** o el formato **properties**, el que prefieras.*

Aplicación WEB

Para crear el módulo de la aplicación web puedes usar el servicio de Spring **initializr** (<https://start.spring.io/>) o bien crearlo desde cero. Si lo haces vía web, no olvides indicar que quieres utilizar:

- Lombok.
- Spring Web.
- Spring Data JPA
- Spring Data JDBC
- Mysql (o MariaDB)
- Spring thymeleaf

Ya irás añadiendo el resto de las librerías a medida que vayas avanzando con los requisitos (*security, validation, etc.*).

Requisitos Aplicación WEB

La aplicación web es la encargada de presentar la información al usuario y de proponer una interface gráfica que permita interactuar con el sistema. Se espera que la interface gráfica sea amigable, y con una navegación a partir de menús en una barra superior. Puedes poner también un pie de página si lo deseas, de manera que quedaran tres espacios bien definidos:

- Una barra superior con menús.

- Un espacio central donde mostrar la información.
- Una barra inferior con un pie de página opcional.

A continuación, os presentamos los requisitos que queremos desarrollar en forma de *historias de usuario*.

"Como usuario anónimo necesito poderme registrar en la plataforma para ser un usuario registrado"

Vamos a tener **dos tipos de usuarios** registrados en el sistema: **ADMIN** y **USER**. Los usuarios ADMIN tendrán permiso para acceder a alguna funcionalidad que los USER tendrán restringida.

Los usuarios se podrán registrar ellos mismos mediante un formulario en el que se pedirán los datos:

- **Username:** *string*, obligatorio y tiene que ser único en el sistema.
- **Password:** *string*, obligatorio.
- **Password Bis:** *string*, obligatorio, tiene que coincidir con el *password* anterior.
- **Email:** *string*, obligatorio, un email válido sintácticamente y único en el sistema.
- **Name:** *string*, obligatorio.
- **Surname:** *string*, obligatorio.
- **Date of birth:** fecha de nacimiento, obligatorio, con el formato dd/MM/yyyy.

Por defecto todos los usuarios serán USER. El sistema les asignará un id único de manera automática y se almacenará en base de datos. Además, también registraremos la fecha de creación del usuario.

Por simplicidad, ahora no se plantea la opción de definir un usuario ADMIN a través de la interface gráfica. Eso sí, en tiempo de arranque del sistema os recomendamos crear un usuario ADMIN vía *script* de base de datos, juntamente con la creación de todas las tablas del sistema, de la que os proporcionaremos el modelo de clases.

"Como usuario anónimo necesito poderme identificar en la plataforma para acceder a sus contenidos"

Un usuario podrá acceder a un formulario de *login* para identificarse en el sistema:

- **Username o Email:** el usuario puede indicar su nombre de usuario (username) o bien el email; cualquiera de los dos valores puede ser usado para identificarse en el sistema

- **Password:** el password correspondiente

Si los datos proporcionados son correctos, el usuario es redirigido al interior de la plataforma, en la que podrá ver un menú superior con las opciones disponibles.

Cuando un usuario hace *login* satisfactoriamente, necesitamos registrar la fecha y hora en que se produce este *login*.

"Como usuario identificado necesito poder hacer *logout* para dejar de estar identificado"

Un usuario podrá acceder a una opción de menú en la que podrá hacer *logout* y dejar de estar identificado en la sesión que tenía en marcha.

"Como usuario registrado necesito poder crear un artista para poder referenciarlo posteriormente"

Como hemos dicho, la web permite registrar películas a los usuarios. Una parte de los datos de estas películas son los denominados Artistas. Para **crear un artista** el sistema solicitará:

- **Name:** *string*, obligatorio
- **Surname:** *string*, obligatorio
- **Type:** desplegable, con dos valores: **ACTOR** o **DIRECTOR**

Al lanzar el formulario, el sistema registra el artista en base de datos asignándole un identificador único.

"Como usuario registrado necesito poder crear una película para poder registrarla en el sistema"

El caso de uso principal consiste en **dar de alta una película** en el sistema. Para ello hay que completar el siguiente formulario:

- **Title:** *string*, obligatorio
- **Release year:** obligatorio, numérico de 4 dígitos, mayor que cero
- **Director:** obligatorio, un desplegable en el que aparecen los artistas (nombre y apellidos) disponibles que son de tipo DIRECTOR, solo se puede escoger una única entrada
- **Actores:** obligatorio, un desplegable múltiple en el que aparecen los artistas (nombre y apellidos) disponibles que son de tipo ACTOR, se pueden escoger todos los que se desee.
- **Poster:** obligatorio, un fichero que represente la película que se está dando de alta.

Si el formulario es correcto, el sistema creará una nueva película con los datos proporcionados y asignará un identificador único a la misma.

Respecto la imagen, se persistirá en un directorio en la máquina en la que se ejecuta la aplicación. Y el identificador de la imagen, quedará registrado junto los datos de la película para que podamos acceder a ella cuando sea necesario.

"Como usuario registrado necesito poder buscar películas para ver su detalle posteriormente"

La aplicación tiene que **mostrar un menú para buscar las películas** que se han creado en el sistema. La búsqueda se podrá realizar a través de un único campo de búsqueda que permitirá buscar por título. La búsqueda tiene que ser tipo *"like"*, es decir, buscar parcialmente en el registro de películas por título.

El resultado de la búsqueda será la presentación de un listado en el que se mostrará el poster de la película, el título y el año de estreno para cada película que cumpla el criterio de búsqueda.

"Como usuario registrado y ADMIN necesito poder editar una película para ajustar sus datos"

Desde el listado de películas, en el caso que el usuario registrado sea un usuario con rol ADMIN, tendrá la opción de **editar una película**.

El proceso de edición de película es el mismo que la creación a excepción que el poster no es obligatorio, ya que tendrá ya uno existente. En el caso que se suministre, la nueva imagen será considerada la imagen de la película a partir de ahora. El resto de los campos siguen siendo los mismos que en la creación y con las mismas validaciones.

"Como usuario registrado necesito poder ver el detalle de una película para ajustar sus datos"

Desde el listado de películas cualquier usuario registrado podrá acceder a **ver el detalle** de una película.

El detalle de una película consiste en mostrar el título, el año de estreno, el director, los actores y el poster.

"Como usuario registrado necesito poder puntuar una película para valorarla"

Desde la vista de una película, el usuario registrado puede **valorar la película** indicando una puntuación de entre 1 a 5. La idea es mostrar un desplegable con la siguiente información:

- Valoración: 5 opciones de 1 a 5, obligatorio.

Un usuario sólo puede valorar una película una única vez, de manera que, si ya ha votado, el formulario no se tiene que mostrar. En caso de que el usuario enviase igualmente el formulario el sistema tiene que controlar que ya existe la valoración y denegarla.

"Como usuario registrado necesito poder ver la puntuación media de una película"

Desde la vista de una película, y en caso de que el usuario registrado haya valorado la película en cuestión, tendrá que poder **ver la valoración** que asignó. No la podrá modificar, solo ver la valoración.

"Como usuario registrado necesito poder ver la puntuación media de una película"

Des de la vista de una película, el usuario registrado puede **ver la media de las valoraciones** al lado del título de la película.

La media de las valoraciones de 1 a 5 tiene que presentarse como un número decimal de dos decimales, redondeado al alza. En el caso que no haya valoraciones, para no mostrar un 0, la interface tendrá que mostrar "--", (*dos guiones*).

HINTS:

- *Spring proporciona mecanismos para validar los datos de una request HTTP (spring-boot-starter-validation); haz uso de ellos.*
- *No esperamos que haya validaciones en la capa de navegador, no es necesario. Es suficiente que implementéis las validaciones a nivel de servidor.*
- *Recordad que el patrón GET-POST-Redirect es el patrón adecuado para implementar el envío y gestión de formularios.*
- *Recordad que queremos que nuestro código sea simple, clean code y mantenible, con una clara división entre capas, en general: controladores, servicios y gestión de la persistencia.*
- *Para la interface dispones de thymeleaf, el mecanismo de templates que permite la presentación de html integrada con Java y Spring.*
- *Para los elementos gráficos puedes optar por el uso de Bootstrap, la librería de componentes html responsive.*
- *Los casos de uso relacionados con la valoración de las películas tendrán que hacer uso del servicio REST; esto se presenta a continuación.*

Servicio REST

Para crear el módulo del servicio REST puedes usar el servicio de Spring *initializr* (<https://start.spring.io/>) o bien crearlo desde cero. Si lo haces vía web, no olvides indicar que quieres utilizar:

- Lombok.
- Spring Web.
- Spring Data JPA
- Mysql (o MariaDB)

Ya irás añadiendo el resto de las librerías a medida que vayas avanzando con los requisitos.

Requisitos Servicio REST

El servicio **REST** se encargará de **almacenar las valoraciones** sobre una película. En concreto, para cada valoración almacenará la siguiente información:

- **Id**: identificador único autogenerado por el sistema (base de datos)
- **createdAt**: el instante de creación
- **score**: el valor de la puntuación
- **filmId**: el identificador de la película valorada
- **userId**: el identificador del usuario autor de la valoración

El principal usuario del servicio REST es la aplicación web, de manera que los casos de uso descritos anteriormente que hacen referencia a las valoraciones tienen que usar los elementos que describiremos a continuación.

Para esto, el servicio REST **propondrá 2 endpoints** basados en HTTP y en JSON. El servicio estará **segurizado** mediante el mecanismo más simple basado en *Client Credentials Flow* (basados en el intercambio de tokens y generación de JWT). Aquí ya nunca hablaremos de usuarios tipo ADMIN o USER como en la aplicación web. Aquí solo hablaremos de un usuario si se comunica con el servicio REST mediante el mecanismo de seguridad definido y con un token válido.

Os presentamos los *endpoints* y el mecanismo de seguridad a continuación.

"Como usuario del servicio REST necesito poder crear una valoración de una película para que quede registrada"

El usuario de la aplicación deberá enviar una petición HTTP como la siguiente:

- *POST /ratings*, con un *request body* formado con los campos:
 - *userId*, numérico, obligatorio
 - *filmId*, numérico, obligatorio
 - *score*, numérico, obligatorio
- Con el header *Authentication Bearer* con un JWT válido.

El servicio tendrá que comprobar que los datos son válidos y proceder con la creación de la valoración.

- Si la petición no se realiza mediante un token JWT válido (en formato y no expirado), el servicio devolverá una respuesta con status 401.
- Si alguno de los datos no es válido, el servicio devolverá una respuesta con status 400 (*bad request*).
- Si ya existe una valoración para el usuario y el film indicados, devolverá una respuesta con status 400 (*bad request*)
- Si se procede a crear la valoración correctamente, devolverá una respuesta con estado 201 (*created*).

"Como usuario del servicio REST necesito recuperar la valoración que un usuario ha dado a una película"

El usuario de la aplicación deberá enviar una petición HTTP como la siguiente:

- *GET /ratings/films/{filmId}/users/{userId}*, donde
 - *{filmId}* es el identificador de la película
 - *{userId}* es el identificador del usuario
- Con el header *Authentication Bearer* con un JWT válido

El sistema buscará el registro correspondiente a la valoración del usuario *userId* para la película *filmId*.

- Si la petición no se realiza mediante un token JWT válido (en formato y no expirado), el servicio devolverá una respuesta con status 401.
- Si el registro no existe, el servicio devolverá una respuesta con status 404 (*not found*)
- Si el registro existe, el servicio devolverá una respuesta con status 200 (*ok*) y una *response body* con:
 - *score*, el valor de la valoración registrado
 - *createdAt*, la fecha y hora de creación del registro en formato ISO 8861 ("yyyy-MM-dd'T'HH:mmZ")

"Como usuario del servicio REST necesito recuperar la media de las valoraciones para un film"

El usuario de la aplicación deberá enviar una petición HTTP como la siguiente:

- *GET /ratings-average/films/{filmId}*, donde
 - *{filmId}* es el identificador de la película
- Con el header *Authentication Bearer* con un JWT válido

El servicio recuperará todas las valoraciones para la película *filmId* y calculará la media, en dos decimales y redondeada al alza. Si la película no tiene valoraciones registradas, la media será 0.

- Si la petición no se realiza mediante un token JWT válido (en formato y no expirado), el servicio devolverá una respuesta con status 401.
- El servicio siempre devolverá una respuesta con status 200 (*ok*) y con una response body con:
 - *average*: valor de la media calculada
 - *ratings*: número total de valoraciones registradas para la película

“Como usuario no identificado quiere poder usar el servicio REST”

Uno de los mecanismos más extendidos de seguridad para usar servicios REST entre dos aplicaciones es el flujo *Client Credentials Grant*. Este flujo es muy sencillo, pero hay que entender bien los pasos para implementarlo correctamente. Para entender los pasos, vamos a definir que la aplicación que quiere usar el servicio REST es el *client* y el servicio REST es el *provider*.

Los pasos son:

1. El *client* envía una petición POST al endpoint */authenticate* con las siguientes características:
 - a. Header *Content-Type*: *application/x-www-form-urlencoded*
 - b. Header *Authorization* con una *authentication Basic*, esto quiere decir un usuario y password en base64
 - c. Body con el campo *grant_type* con el valor *client_credentials*
2. El *provider* genera una respuesta en función del resultado:
 - a. Si el user y password indicados en la cabecera Basic no son correctos, devuelve un estado 401
 - b. Si la petición no contiene el campo *grant_type* con el valor esperado, devuelve un estado 400
 - c. Si ambas condiciones anteriores no se cumplen:
 - i. El servicio genera un token JWT con la siguiente información:
 1. Con una fecha de expiración de una hora (a poder ser configurable mediante *properties*)
 2. Signado con una *secret* (también configurable mediante *properties*)



3. Asignando el valor del usuario usado en el *subject* del JWT
- ii. El servicio genera una respuesta con status 200 (*ok*) y tres campos:
 1. `accessToken`: valor del token JWT generado
 2. `tokenType`: con el valor "bearer"
 3. `expiresIn`: con el valor en segundos del tiempo de expiración del token

Tenéis todo el proceso de seguridad documentado oficialmente en:
<https://datatracker.ietf.org/doc/html/rfc6749#section-4.4>

Es importante saber leer estos rfc, ya que son de uso común para implantar estándares en el desarrollo de protocolos.

HINTS

- *Recuerda que gran parte de la gestión de la seguridad se realiza mediante spring-security*
- *Ten presente que tendrás algunos endpoints securizados con basic authentication y algunos endpoints securizados con jwt, con lo que serán necesarias definir dos configuraciones de seguridad.*
- *El usuario y password habilitado por parte del provider tienen que ser conocidos también por el client, de manera que esta información es ideal que también esté configurada mediante properties en los dos.*

Proceso BATCH

Para crear el módulo del proceso *batch* puedes usar el servicio de Spring *initializr* (<https://start.spring.io/>) o bien crearlo desde cero. Si lo haces vía web, no olvides indicar que quieres utilizar:

- Lombok
- Spring Batch
- Spring JPA
- Mysql (o MariaDB)

Ya irás añadiendo el resto de las librerías a medida que vayas avanzando con los requisitos.

Requisitos Proceso BATCH

El proceso Batch consiste en exportar las películas de manera incremental. Es decir, el proceso leerá las películas registradas en la base de datos y para aquellas que no se hayan nunca exportado se procederá a generar una entrada en un fichero CSV.

Vamos a definir los tres pasos del proceso Batch.

"Primer paso: seleccionar las películas no exportadas"

El proceso Batch se iniciará leyendo las películas que no se hayan exportado. El mecanismo para saber si una película se ha exportado o no es sencillo. Para que el proceso Batch no altere el contenido de la tabla original (su gestión la hace la aplicación WEB), se creará una tabla en la que guardará un registro por película exportada. Esta tabla, idealmente, contendrá lo siguiente:

- `Id`, identificador del registro, generado automáticamente
- `jobId`, identificador del job (de la ejecución del batch) que ha generado este registro de exportación
- `filmId`, identificador de la película exportada
- `exportedAt`, fecha y hora de la exportación

Como esta tabla y la tabla de películas estarán en el mismo esquema de base de datos, podéis realizar una sentencia SQL que os de las películas que no se han exportado.

"Segundo paso: generar el archivo CSV"

La generación del archivo CSV consiste en generar una línea del CSV para cada película con el siguiente contenido:

- `{filmId},{title},{releaseYear}`
- Donde `filmId` es el identificador de la película, `title` es el título de la película, y `releaseYear` es el año de estreno.

"Tercer paso: notificación de finalización de escritura"

Tras la ejecución de los dos pasos anteriores, se puede notificar un servicio para indicar que la escritura ha finalizado. En este servicio, se creará un registro para la película que se ha exportado a CSV en la tabla de exportaciones y se insertará en la tabla con los datos antes mencionados (`id`, `jobId`, `filmId`, `exportedAt`).

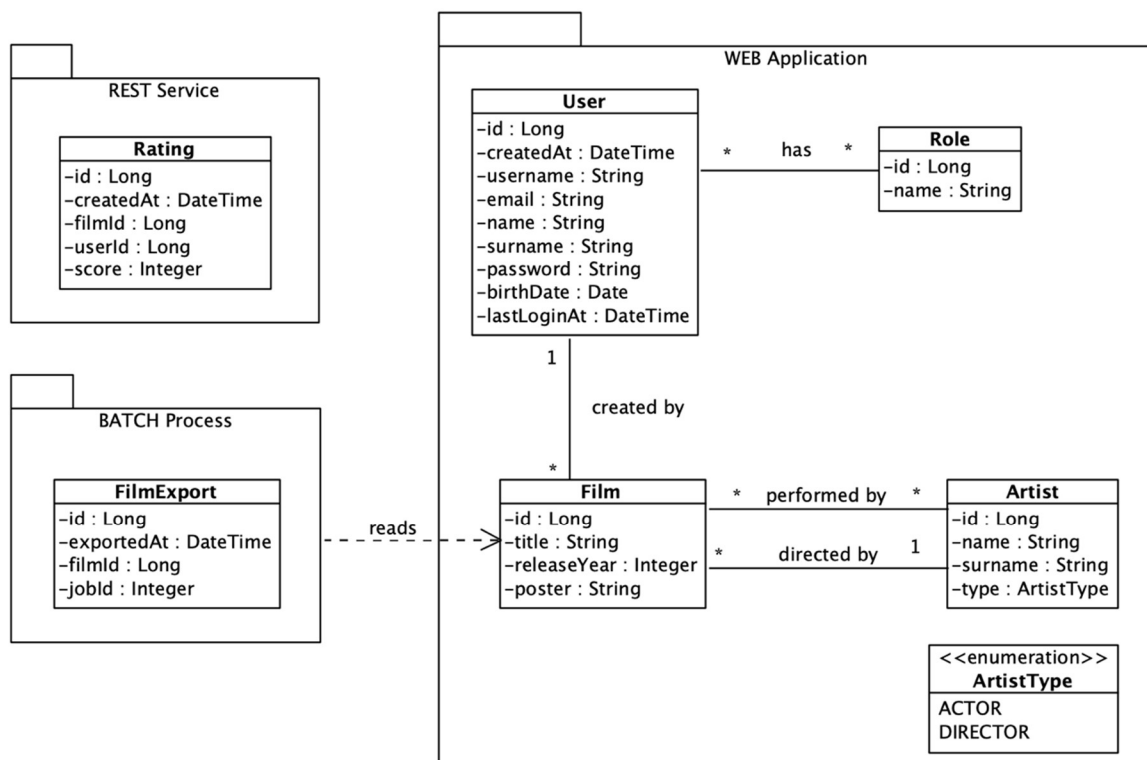
De esta manera, en la siguiente ejecución del proceso estas películas se descartarán de la exportación.

HINTS

- Recuerda que un proceso batch puede definir steps y listeners. En nuestro caso, estamos solicitando un step de reader, un step de writer y un listener.

Diagrama de clases y límites entre componentes

Os presentamos una propuesta de diagrama de clases que puede dar soporte a todos los servicios.



Como se observa, se han definido tres áreas diferenciadas. La aplicación WEB, el servicio REST y el proceso BATCH. Para simplificar, podéis utilizar el mismo esquema de base de datos para todos los componentes, aunque en un entorno real las restricciones serían diferentes.

Algunos comentarios:

- **No se están creando restricciones entre las tablas de un componente y las tablas de otro componente.** Por ejemplo, el servicio REST almacena un *filmId* y un *userId*, sin comprobar que estas entidades existen en ningún sitio. Esto es así, es correcto desde el punto de vista de diseño y nos facilita la gestión.

- **Hay una dependencia entre el proceso BATCH y la aplicación WEB**, ya que el proceso BATCH necesita leer las películas. Esto es así y es correcto, aunque en general, en un entorno real, el proceso BATCH operaría con una réplica de los datos de la aplicación WEB, No nos hace falta entrar en estos mecanismos en nuestro escenario.
- **Todos los id de las tablas son numéricos**, autogenerados por el propio mecanismo de base de datos. No es necesario complicar el modelo de generación de claves primarias con ningún otro mecanismo.
- **El poster se está almacenando como un string**. Este string representa el identificador de archivo que se ha guardado en el sistema. Es una manera de hacerlo, simple y que funciona. Lo podéis hacer así o con alguna alternativa que ya hayáis visto.
- **Tendréis que mapear las relaciones entre las clases de objetos a relaciones JPA**. Revisad las cardinalidades en cada extremo de cada relación para definir el mejor mecanismo en cada caso:
 - o User – Role
 - o Film – User
 - o Film – Artist (director)
 - o Film – Actor (actors)

Aspectos transversales

No se expresa como requisito, pero es necesario que penséis en los siguientes aspectos.

Logging

Podéis usar *logback* como vuestro mecanismo para generar trazas tanto en consola como, si lo estimáis oportuno, en fichero. Ya sabéis que la observabilidad, especialmente, de los errores y *exceptions* es fundamental en cualquier sistema.

Gestión de exceptions

Se recomienda la implementación de un servicio de gestión de *exceptions* que gestione los errores del sistema de una manera grácil para el usuario, bien mediante la presentación de páginas específicas de errores para la aplicación WEB (configurable vía *thymeleaf* y *spring*), bien mediante la generación de mensajes coherentes en el servicio REST.

Internacionalización

Podéis habilitar mecanismo multiidioma en la aplicación WEB, de manera que el usuario pueda escoger el idioma en el que ver la interface. No es requisito que toda la aplicación este en dos idiomas,



pero es interesante que aun dejando la mayoría de mensajes sin traducir, procuréis dejar la aplicación internacionalizada.

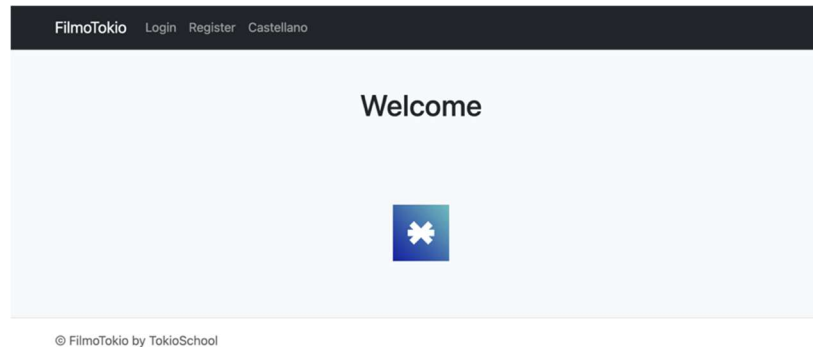
Testing

Se valorará muy positivamente que estéis testeando vuestro código mediante tests unitarios. Un grado elevado de cobertura de vuestro código os dará tranquilidad para saber que todo funciona correctamente en el backend.

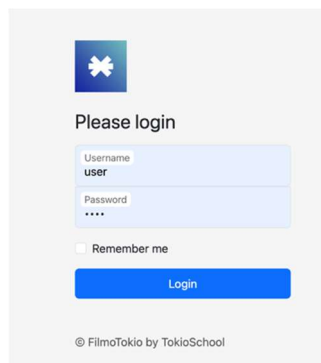
Mockups de ejemplo

Sin que lo que se presenta a continuación sea el ejemplo de lo que hay que implementar (dejamos que seáis vosotros los que decidáis como presentar la información al usuario), os dejamos algunos ejemplos de una posible interfaz de la aplicación WEB.

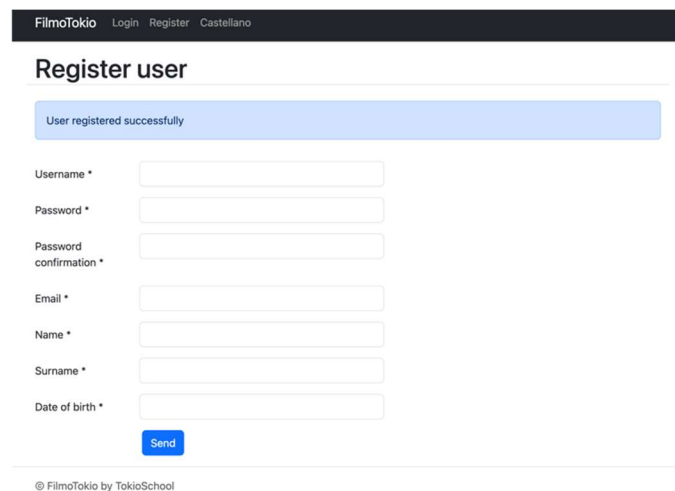
Bienvenida



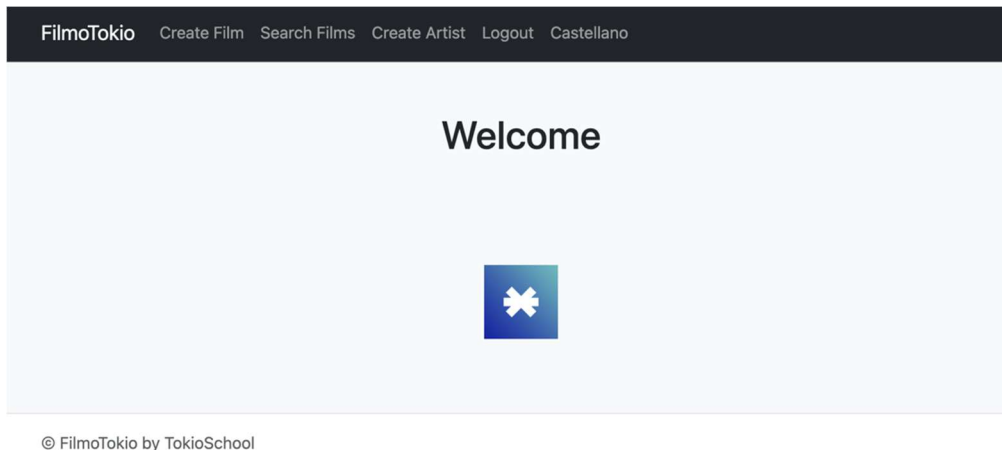
Login



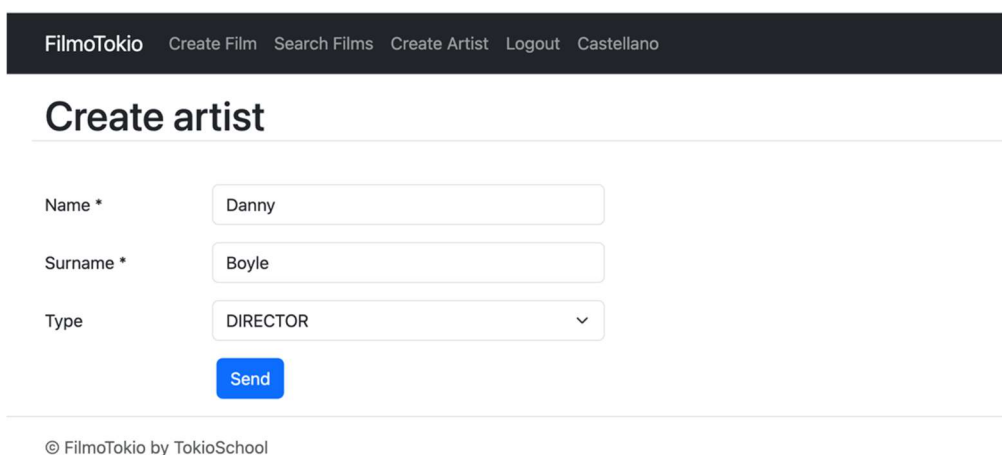
Registrarse



Bienvenida tras el login

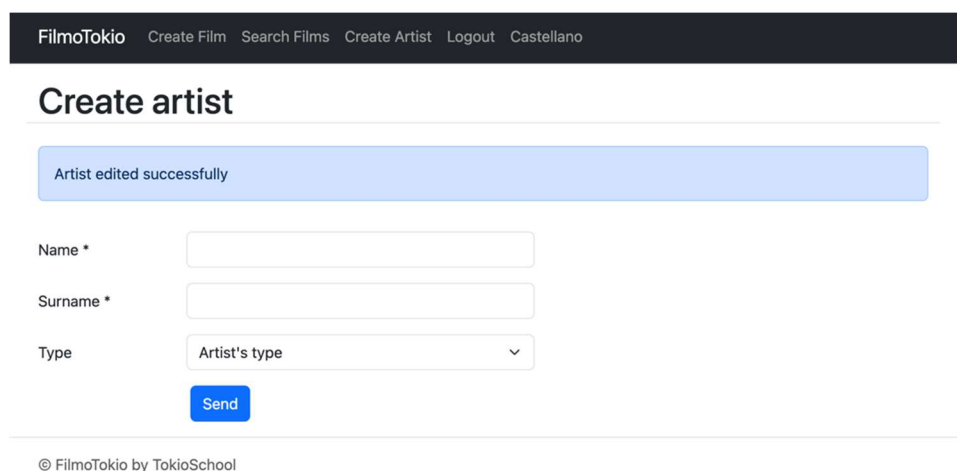


Creación de un artista



The screenshot displays the "Create artist" form in the FilmoTokio application. The dark navigation bar at the top is identical to the previous page. The title "Create artist" is prominently displayed. The form consists of three input fields: "Name *" with the value "Danny", "Surname *" with the value "Boyle", and "Type" with a dropdown menu currently showing "DIRECTOR". A blue "Send" button is positioned below the "Type" field. A copyright notice "© FilmoTokio by TokioSchool" is at the bottom.

Feedback tras la creación de un artista





This screenshot shows the "Create artist" form after a successful submission. A light blue success message box at the top states "Artist edited successfully". The form fields are now empty: "Name *" and "Surname *" are empty text boxes, and "Type" is a dropdown menu showing "Artist's type". The blue "Send" button remains at the bottom. The navigation bar and copyright notice "© FilmoTokio by TokioSchool" are consistent with the previous pages.


Mostrar errores siempre que sea posible

FilmoTokio Create Film Search Films Create Artist Logout Castellano

Create film


Title * 
must not be blank

Release year * 
must not be null

Director * 


Actors *

Daniel Craig
Ana de Armas
Michael Fassbender
Morgan Freeman


must not be empty

Poster

Tria un fitxer

No ...txer 

invalid poster

© FilmoTokio by TokioSchool

Buscador de películas

FilmoTokio Create Film Search Films Create Artist Logout Castellano


Search films

Title

© FilmoTokio by TokioSchool

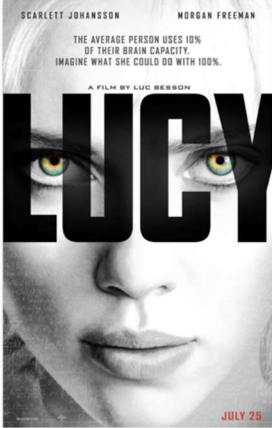
Resultados de una búsqueda

FilmoTokio Create Film Search Films Create Artist Logout Castellano




Ex Machina (2015)

[View](#)



Lucy (2014)

[View](#)



Steve Jobs (2015)

[View](#)

© FilmoTokio by TokioSchool

Ver el detalle de una película (el usuario no ha valorado la película, ni nadie lo ha hecho)

FilmoTokio Create Film Search Films Create Artist Logout Castellano

Steve Jobs (--)

Release year:
2015

Director:
Danny Boyle


Actors:
Seth Rogen
Michael Fassbender
Kate Winslet

Create by user:
user

Your rating:
Pending

--- Rate ▾

[Rate](#)



Ver el detalle de una película (el usuario ha valorado la película y aparece la media)

FilmoTokio Create Film Search Films Create Artist Logout Castellano

Steve Jobs (4.00)


Release year:
2015

Director:
Danny Boyle

Actors:
Kate Winslet
Seth Rogen
Michael Fassbender

Create by user:
user

Your rating:
4



michael
fassbender
kate
winslet
seth
rogen
jeff
daniels

steve jobs|

from director
danny boyle
and
screenwriter
aaron sorkin

coming soon

Ver el detalle de una película (el usuario ha intentado valorar la película, pero el servicio REST no está disponible)

FilmoTokio Create Film Search Films Create Artist Logout Castellano

Ex Machina (--)

Release year:
2015

Director:
Alex Garland

Actors:
Alicia Vikander

Create by user:
admin

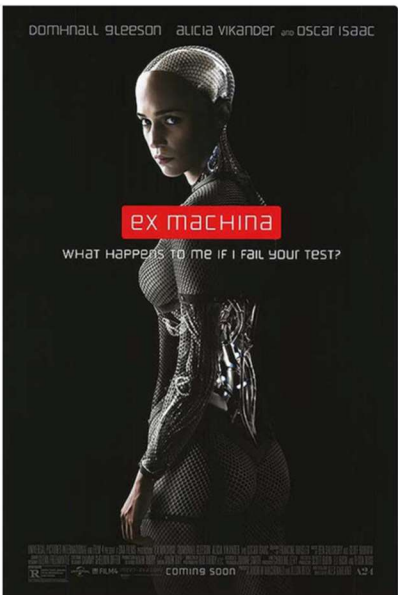
Your rating:
Pending

0

▼

service is not available

Rate



DOMMONALL GLEESON ALICIA VIKANDER OSCAR ISAAC

ex machina

WHAT HAPPENS TO ME IF I FAIL YOUR TEST?

coming soon



Página de error genérica

FilmoTokio Create Film Search Films Create Artist Logout Castellano

An error has occurred

Not Found

© FilmoTokio by TokioSchool

ENTREGA

El proyecto final debe estar compuesto **obligatoriamente por los 3 componentes descritos**:

- Aplicación WEB
- Servicio REST
- Proceso BATCH

Cualquier otra solución será valorada positivamente siempre y cuando esté dentro del marco de trabajo de Spring y aborde los elementos técnicos equivalentes.

Para la evaluación del proyecto, será necesario adjuntar en el apartado de “Entrega” de la plataforma:

1. El código en formato .ZIP o .RAR.
2. Un documento o documentos en PDF o Word con las explicaciones, argumentos y razonamientos de los puntos más relevantes y/o complejos del proyecto.
 - a. Podéis añadir en este documento todas las capturas de pantalla que consideréis necesarias para dar soporte a vuestras explicaciones.
3. El proyecto una vez descomprimido, debe poder ser importado como proyecto Maven en cualquier IDE de desarrollo.

Los plazos de corrección del proyecto serán de entre 7 y 15 días laborales desde la fecha de entrega. Junto con el feedback y la nota, el profesor te formulará tres preguntas sobre el proyecto, las cuales deben ser respondidas en un video y esto se considerará la defensa del proyecto.

La duración del video de la defensa no debe ser mayor a 10 minutos.