Info-Game Mashup

Arquitectura e Integración de Sistemas Software

Grado de Ingeniería del Software

Curso <2018-2019>

Francisco Alé Palacios (fraalepal@alum.us.es)
Pedro Biedma Fresno (pedbiefre@alum.us.es)
Enrique Merino Verde (enrmerver@alum.us.es)
José Manuel Gata Fernández (josgatfer@alum.us.es)

Tutor: Alfonso Eduardo Márquez Chamorro
Número de grupo: 1

Enlace de la aplicación: http://info-game-mashup.appspot.com/
Enlace de proyecto en GitHub:
https://github.com/blackylyzard/Info-Game

HISTORIAL DE VERSIONES

Fecha	Versión	Detalles	Participantes
11/03/2019	1.0	- Incluye introducción de la aplicación a desarrollar, prototipos de las interfaces de usuario (vista inicial y vista de la aplicación) y diagramas UML de componentes, de despliegue y de secuencia de alto nivel.	-Francisco Alé Palacios -Pedro Biedma Fresno -Enrique Merino Verde -José Manuel Gata Fernández
25/04/2019	1.1	-Incluye descripciones de los diagramas realizados, así como la implementación de los diagramas de clases y de secuencia -Modificaciones en los diagramas de secuencia de alto nivel y componentes.	-Francisco Alé Palacios -Pedro Biedma Fresno -Enrique Merino Verde -José Manuel Gata Fernández
25/05/2019	1.2	-Incluye descripciones de la totalidad de los diagramas, la implementación de todos ellos y diversas modificaciones en el campo de evolución. -Se han incluido las pruebas -Se ha incluido capturas de funcionamiento en el manual de usuario -Se ha incluido la documentación de la API -Se ha modificado el apartado de implementación.	-Francisco Alé Palacios -Pedro Biedma Fresno -Enrique Merino Verde -José Manuel Gata Fernández

Índice

1 Introducción

Desde hace varios años, la industria de los videojuegos ha experimentado un desarrollo abrumador y son cada vez más las personas que invierten su tiempo libre en jugar, incrementándose así, el número de jugadores interesados en ampliar sus conocimientos sobre el mundo de los videojuegos.

El problema que surge debido a esta gran evolución es que cada vez son más frecuentes los lanzamientos de grandes videojuegos y el tiempo que una persona puede dedicar a cada uno de ellos es bastante más limitado en comparación con años anteriores.

Así pues, cuando los jugadores se ven interesados en comenzar con un nuevo título surgen dudas del estilo ¿me gustará? ¿será este mi tipo de juego?... Con objeto de resolver estas dudas surge Info-Game Mashup, una aplicación que proporcionará al jugador información básica acerca del título deseado. Encontraremos desde una descripción básica del juego, enlaces a las bandas sonoras oficiales, vistas de los streamings más vistos sobre el mismo e incluso acceso a foros donde poder debatir y buscar información con el resto de la comunidad.

Todo esto para poder evitar intensas y complejas tardes de búsqueda de información y centrarse cuanto más rápido en la verdadera pasión del usuario, jugar.

1.1 Aplicaciones integradas

Wikipedia: con la API de Wikipedia podremos extraer información de los juegos.

Twitch: con esta API podremos obtener los streamings más vistos del momento.

Spotify: con esta API podremos obtener la lista de reproducción correspondiente a la banda sonora del videojuego.

YouTube: con esta API extraeremos videos acerca del videojuego buscado.

Nombre aplicación	URL documentación API
Wikipedia	https://es.wikipedia.org/w/api.php
Twitch	https://dev.twitch.tv/docs/v5/
YouTube	https://developers.google.com/youtube/
Spotify	https://developer.spotify.com/documentation/web-api/

TABLA 1. APLICACIÓN INTEGRADAS

1.2 Evolución del proyecto

Partimos de la idea de crear un sitio web donde los jugadores habituales de videojuegos pudieran acceder con más facilidad a comunidades de jugadores con el mismo juego en común y a servicios de streaming como los de Twitch, esta idea fue abandonada debido a que las API de la mayoría de los servicios tales como Steam, Origin… no ofrecían lo suficiente como para poder hacerlo.

Más tarde surgió la idea de recopilar la información básica acerca de un videojuego en concreto y, mediante servicios tales como Twitch o YouTube, poder acerca a un jugador nuevo hacia una comunidad con mayor experiencia y conocimiento.

En un primer momento pensamos en utilizar la API de alguna enciclopedia en línea, para obtener la mayor información posible; después de una búsqueda en Internet encontramos que Wikipedia utiliza Mediawiki, por lo que pensamos en usarla.

En lugar de Mediawiki, escogimos IGDB, una base de datos de videojuegos, mucho más fiable que nuestra anterior opción.

Ante las dificultades que plantea la implementación de la API de Reddit, hemos tomado la decisión de "sustituirla" por la API de YouTube, de la cual, extraeremos videos y realizaremos las operaciones de post.

Sin embargo, hemos decidido no descartar por el momento el uso de la API de Reddit hasta comprobar su viabilidad y decidir su implementación o no. Así pues, hemos decidido modificar los diagramas y el mockup acorde a la implementación de la API de YouTube pero manteniendo las operaciones que debería realizar Reddit.

Finalmente hemos concluido que no haremos uso de la API de Reddit, la cual será sustituida por la de YouTube, la operación de Post que proponíamos con la API de Reddit va a ser sustituida por varias operaciones de post haciendo uso de la API de YouTube, en concreto postear un comentario desde el Mashup, dar like al video o dar dislike.

La API de IGBD también ha sido sustituida por la de Wikipedia, contactamos con IGDB los cuales nos informaron de una de sus versiones iba a ser eliminada pronto, sin embargo ante la duda y la tardanza en responder de IGDB decidimos sustituir su API por la de Wikipedia y así evitar futuros problemas.

La vista de la aplicación ha sufrido bastantes alteraciones a lo largo del desarrollo debido a los constantes cambios, los cuales han provocado que las modificaciones en las diferentes vistas se tuvieran que realizar para una mejor implementación visual.

El desarrollo se ha completado a la perfección pudiendo aplicar los cambios que eran necesarios sin ningún tipo de problema de funcionamiento.

2 Prototipos de interfaz de usuario

2.1 Vista Inicial

Esta vista se correspondería con el primer contacto del usuario y la aplicación. En ella encontramos una barra de búsqueda donde el usuario deberá escribir el nombre del videojuego sobre el que quiere obtener información.

FIGURA 1. PROTOTIPO DE INTERFAZ DE USUARIO DE LA VISTA INICIAL

2.2 Vista de la Aplicación

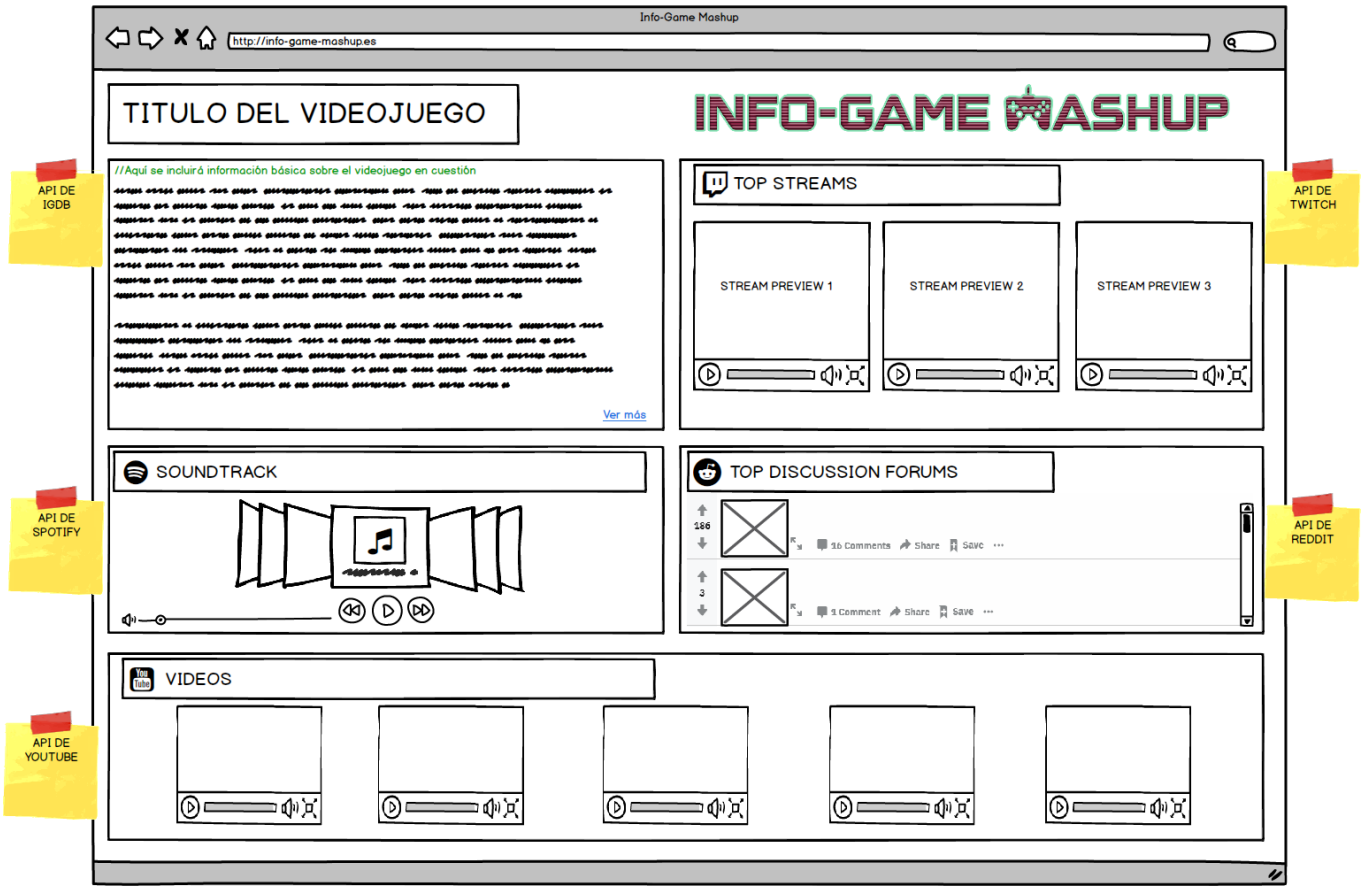


FIGURA 2. PROTOTIPO DE INTERFAZ DE USUARIO DE LA VISTA DE LA APLICACIÓN

Esta vista se correspondería con la respuesta que el usuario recibe a su petición de información, en ella podemos encontrar de forma ordenada información muy variada acerca del videojuego consultado.

-En la parte superior izquierda encontramos el titulo así como la descripción del juego en cuestión.

-En la parte superior derecha encontramos los Streams más vistos de ese juego y una posible vista previa de cada uno.

-En la parte central izquierda encontramos la lista de reproducción correspondiente a la banda sonora del videojuego.

-En la parte central derecha encontramos los foros de discusión más populares.

-En la parte inferior, encontramos los videos relacionados con el videojuego buscado.

3 Arquitectura

3.1 Diagrama de componentes

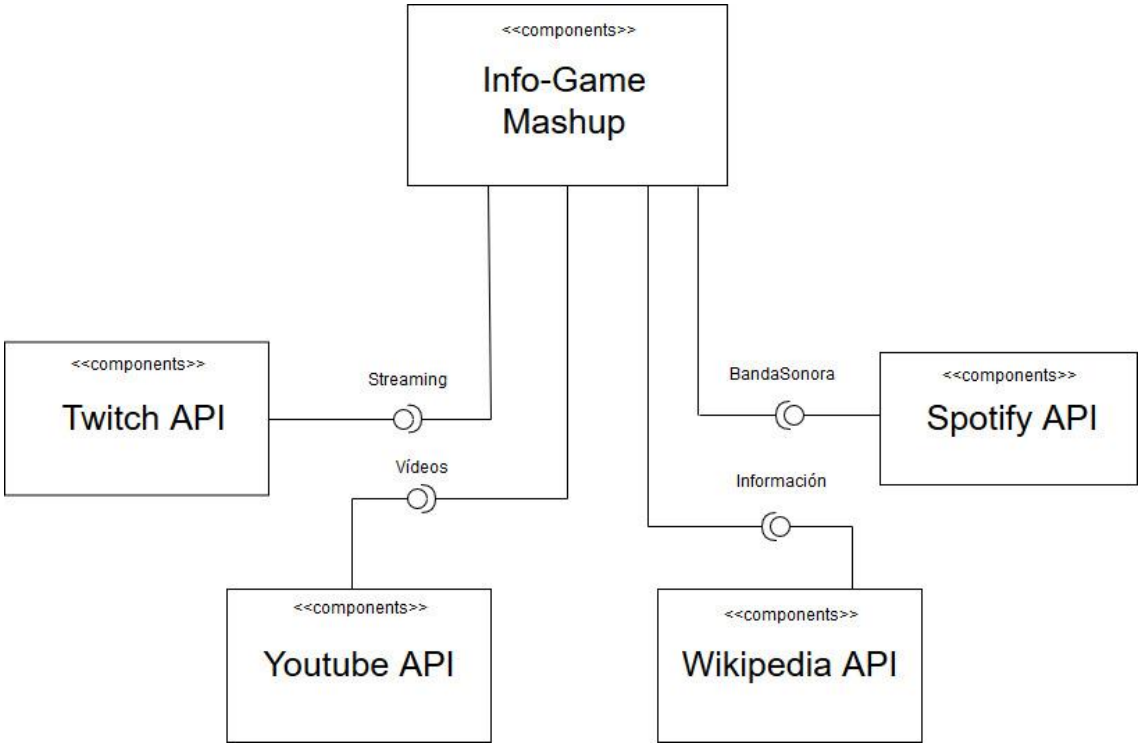


FIGURA 3. DIAGRAMA DE COMPONENTES.

3.2 Diagrama de despliegue

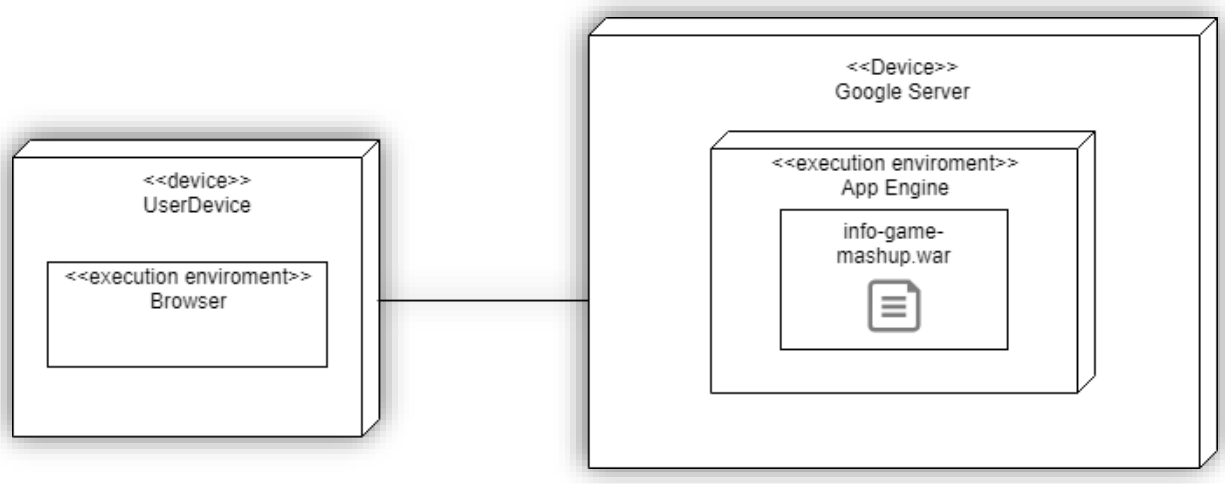


FIGURA 4. DIAGRAMA DE DESPLIEGUE.

3.3 Diagrama de secuencia de alto nivel

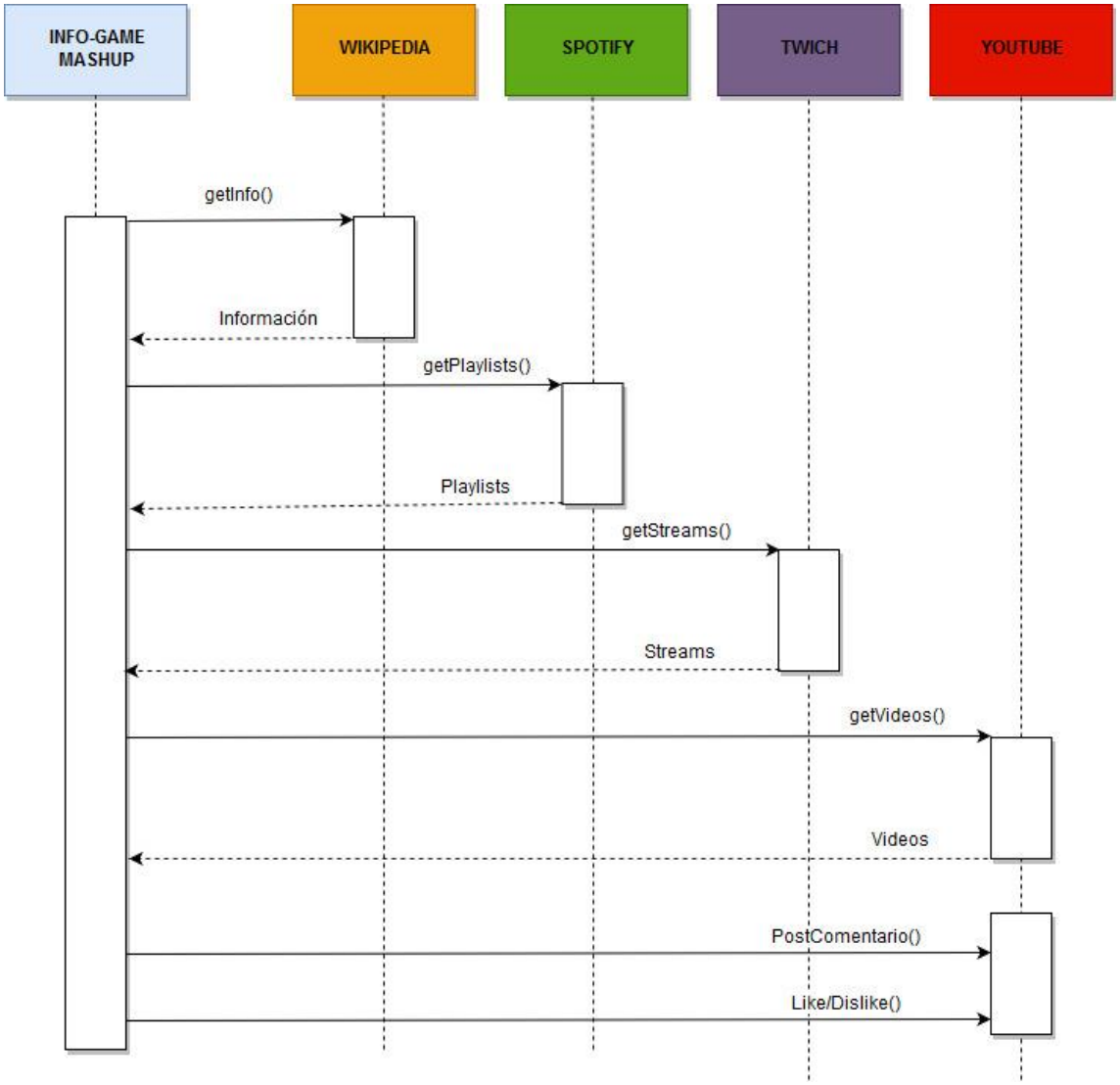


FIGURA 5. DIAGRAMA DE SECUENCIA.

DIAGRAMA DE SECUENCIA QUE EXPLICA DE FORMA ESQUEMÁTICA LAS DIFERENTES GESTIONES QUE SE SIGUEN PARA LA OBTENCIÓN DE LA INFORMACIÓN Y SU POSTERIOR DEVOLUCIÓN EN LA VISTA DE LA APLICACIÓN.

3.4 Diagrama de clases

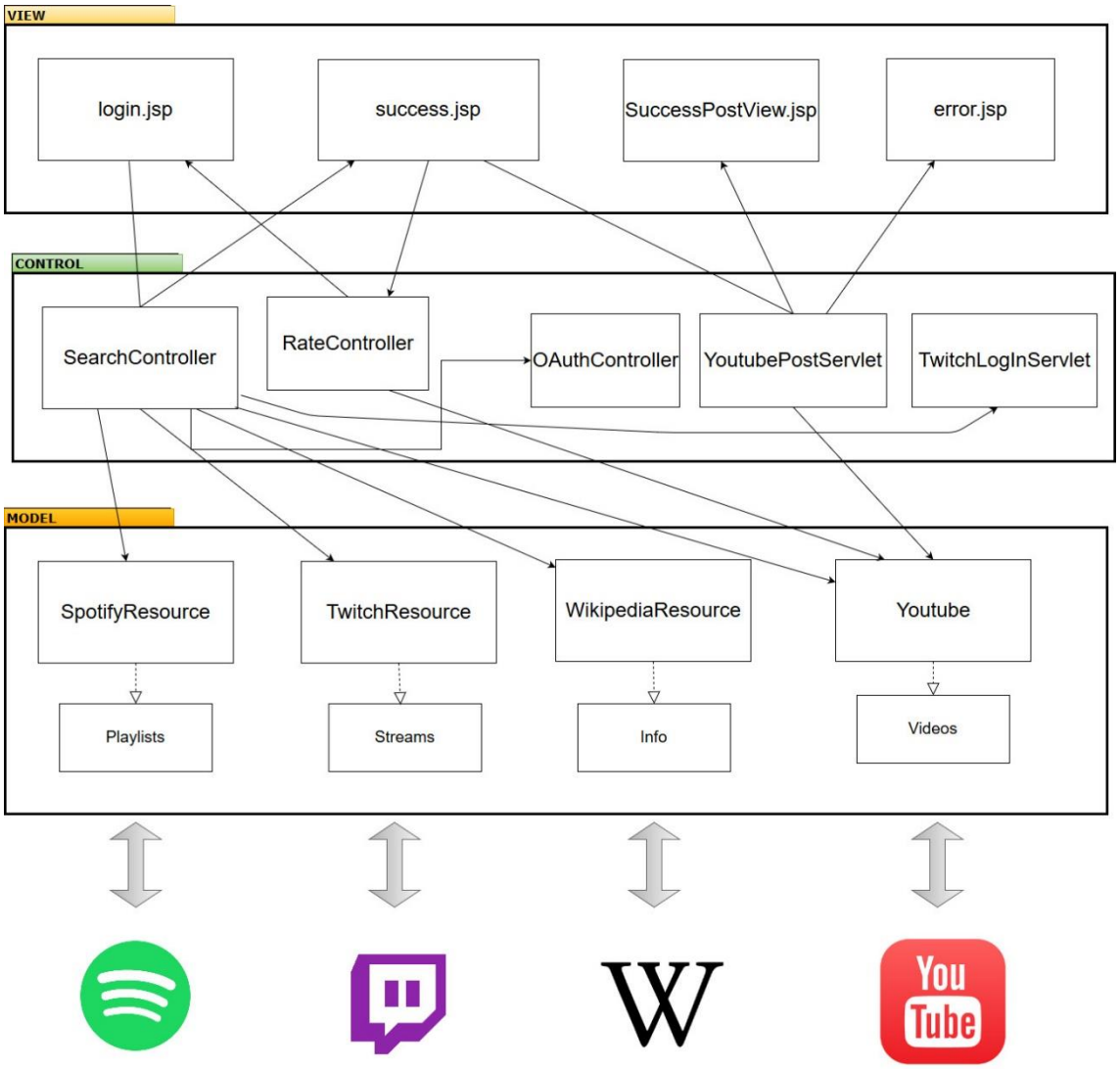


FIGURA 6. DIAGRAMA DE CLASES.

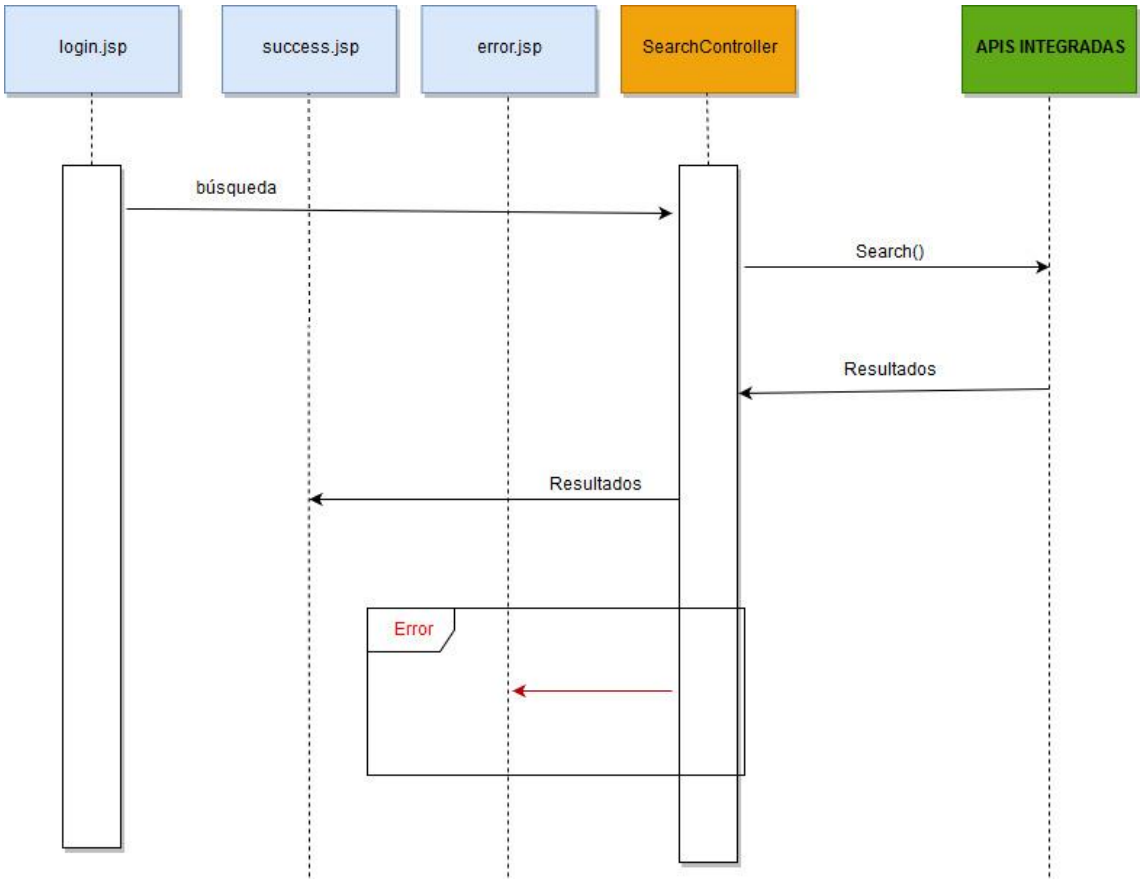
DIAGRAMA DE CLASES QUE MUESTRA LA ESTRUCTURA DE LA APLICACIÓN BASADA EN EL MODELO VISTA CONTROLADOR.

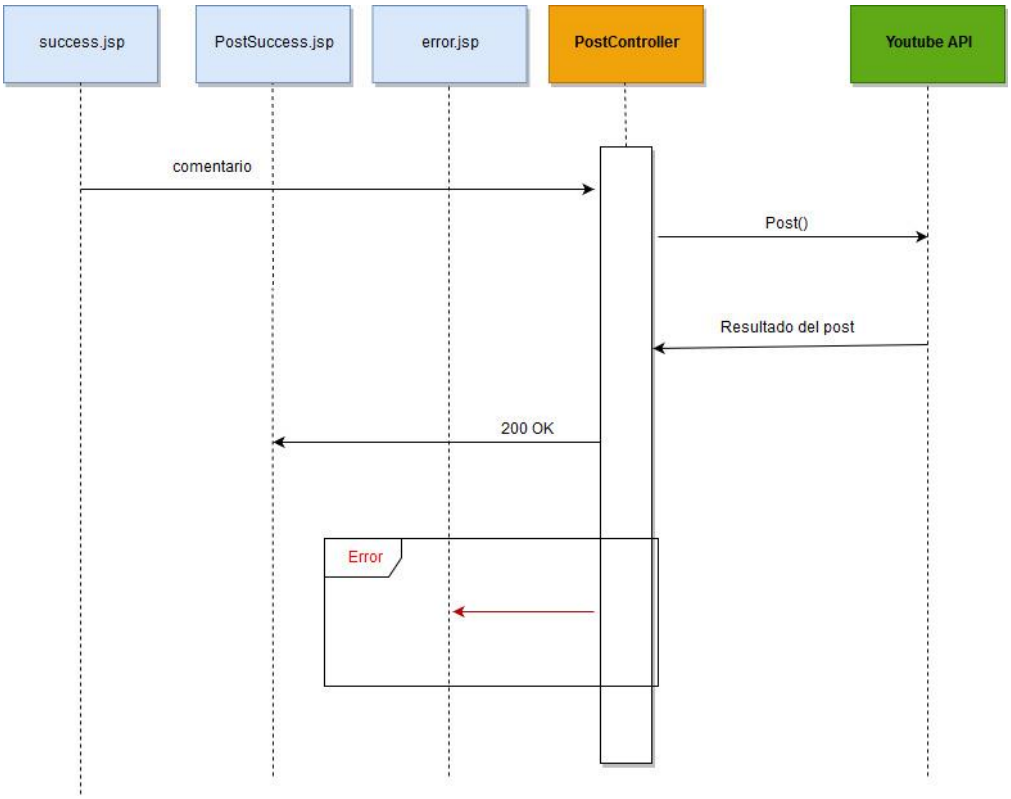
EN EL PODEMOS VER LOS SUCESIVOS PROCEDIMIENTOS REALIZADOS POR LA APLICACIÓN DESDE EL MOMENTO EN EL QUE USUARIO INTRODUCE EL CAMPO A BUSCAR HASTA SU MUESTRA POR PANTALLA, HACIENDO ALUSIÓN TAMBIÉN A LA OPERACIÓN DE POST QUE EL USUARIO PUEDE REALIZAR.

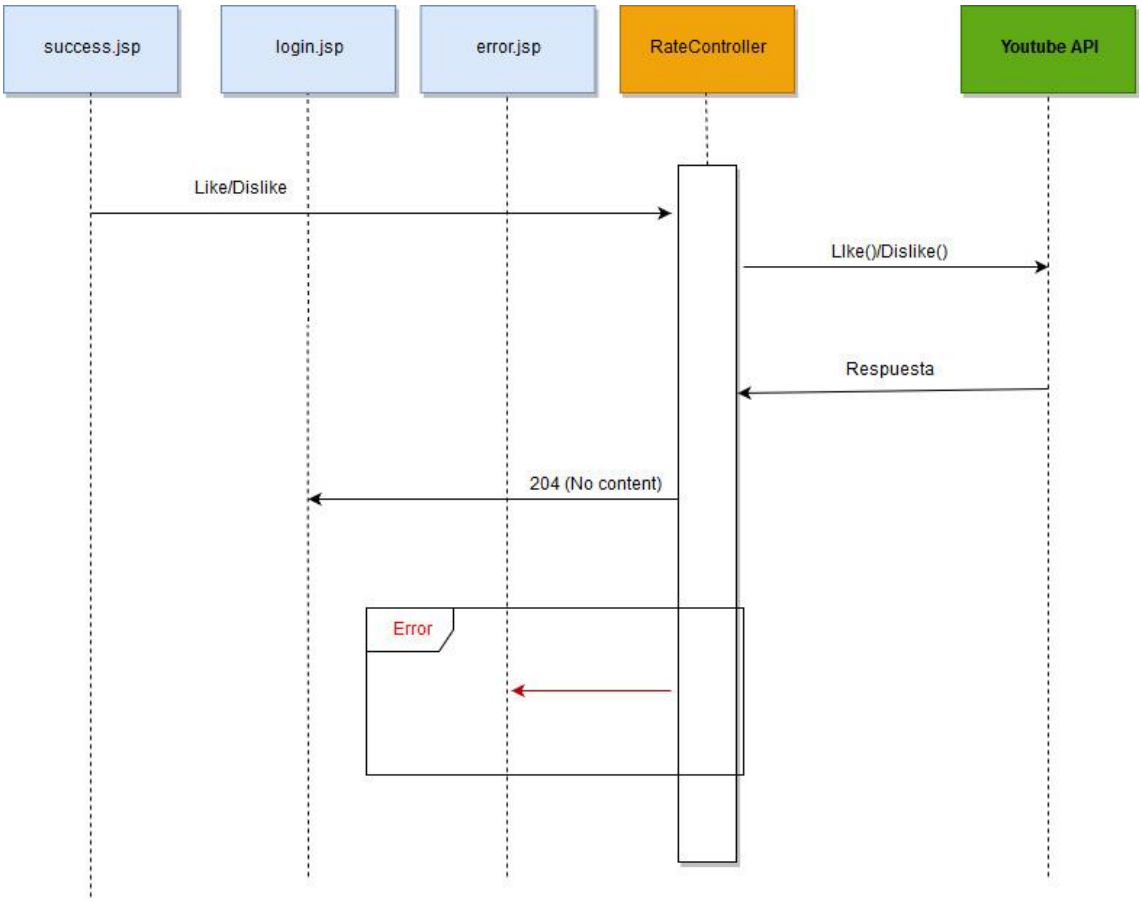
3.5 Diagramas de secuencia

DIAGRAMA DE SECUENCIA QUE EXPLICA DE FORMA PRECISA LOS DIFERENTES PASOS SE QUE SIGUEN PARA LA OBTENCION DE LA INFORMACIÓN Y LA POSTERIOR DEVOLUCION DE LOS RESULTADOS EN LA VISTA DE LA APLICACIÓN.

SE MUESTRA DE UNA FORMA DETALLADA EL PROCEDIMIENTO QUE SIGUE LA APLICACIÓN CUANDO SE PRODUCEN LAS DIFERENTES OPERACIONES DE POST.




success.jsp
PostSuccess.jsp
error.jsp
PostController
Youtube API
comentario
Post()
Resultado del post
200 OK
Error


success.jsp
login.jsp
error.jsp
RateController
Youtube API
Like/Dislike
LIke()/Dislike()
Respuesta
204 (No content)
Error

4 Implementación

Algunos datos sobre la implementación son:

- La vista de los resultados de la búsqueda implementa recursos iframes y embeds de html para poder visualizar numerosos directos de Twitch desde la misma vista, los thumbnails de los videos de YouTube y reproducción directa de las listas de reproducción de Spotify.

- El texto de Wikipedia ha sido procesado para ser más limpio mediante una función.

- El recurso de Twitch y las operaciones de like/dislike se han implementado mediante el objeto de Java HTTPClient.

- La búsqueda en Wikipedia se compone de una búsqueda y se captura el id del primer resultado y se piden con parámetros especiales para poder cargar el texto adecuadamente

- En caso de que la búsqueda no encuentre resultados para alguno de los recursos se muestra una imagen para explicar que no se ha encontrado nada.

5 Pruebas

Documentar las pruebas realizadas a la aplicación. Justificar textualmente la estrategia de pruebas seguida y por qué (ej. pruebas incrementales ascendentes).

Indicar el número total de pruebas realizadas y cuáles de ellas han sido automatizadas mediante JUnit.

Resumen	
Número total de pruebas realizadas	8
Número de pruebas automatizadas	8 (100%)

ID	**Prueba 1**
Descripción	Prueba para la detección de errores al implementar búsquedas en Wikipedia usando servicios de su API.
Entrada	Se hace uso de la API para invocar al servicio usando la URI 'https://es.wikipedia.org/w/api.php?action=parse&format=json&page=title &prop=wikitext' desde nuestra aplicación.
Salida esperada	Los datos devueltos en formato JSON son mapeados a una clase Java correctamente
Resultado	**EXITO**
Automatizada	Sí

ID	**Prueba 2**
Descripción	Prueba para la detección de errores al implementar búsquedas en Wikipedia usando servicios de su API.
Entrada	Se hace uso de la API para invocar al servicio usando la URI 'https://es.wikipedia.org/w/api.php?action=parse&format=json&page=title &prop=wikitext' desde nuestra aplicación.
Salida esperada	La clase que envuelve los datos recibidos poseen contenido y son legibles.
Resultado	**EXITO**
Automatizada	Sí

ID	**Prueba 3**
Descripción	Prueba para la detección de errores al implementar búsquedas en Spotify usando servicios RESTful.
Entrada	Se hace uso de la API para invocar al servicio usando la URI 'https://api.spotify.com/v1/search?q="Doom+soundtrack"&type=playlist&limit=4' desde nuestra aplicación.
Salida esperada	Los datos devueltos en formato JSON son mapeados a una clase Java y a continuación se muestran por pantalla.
Resultado	**EXITO**
Automatizada	Sí

ID	**Prueba 4**
Descripción	Prueba para la detección de errores al implementar búsquedas en Spotify usando servicios RESTful.
Entrada	Se hace uso de la API para invocar al servicio usando la URI 'https://api.spotify.com/v1/search?q="Doom+soundtrack"&type=playlist&limit=4' desde nuestra aplicación.
Salida esperada	El objeto POJO que recibe la información no está vacío
Resultado	**ÉXITO**
Automatizada	Sí

ID	**Prueba 5**
Descripción	Prueba para la detección de errores al implementar búsquedas en Spotify usando servicios RESTful.
Entrada	Se hace uso de la API para invocar al servicio usando la URI 'https://www.googleapis.com/youtube/v3/search?part=snippet&maxResults=3&q=query&type=video' desde nuestra aplicación.
Salida esperada	Los datos devueltos en formato JSON son mapeados a una clase Java correctamente.
Resultado	**EXITO**
Automatizada	Sí

ID	**Prueba 6**
Descripción	Prueba para la detección de errores al implementar búsquedas en Youtube usando servicios RESTful.
Entrada	Se hace uso de la API para invocar al servicio usando la URI 'https://www.googleapis.com/youtube/v3/search?part=snippet&maxResults=3&q=query&type=video' desde nuestra aplicación.
Salida esperada	El objeto POJO creado con los resultados de la petición no están vacios.
Resultado	**EXITO**
Automatizada	Sí

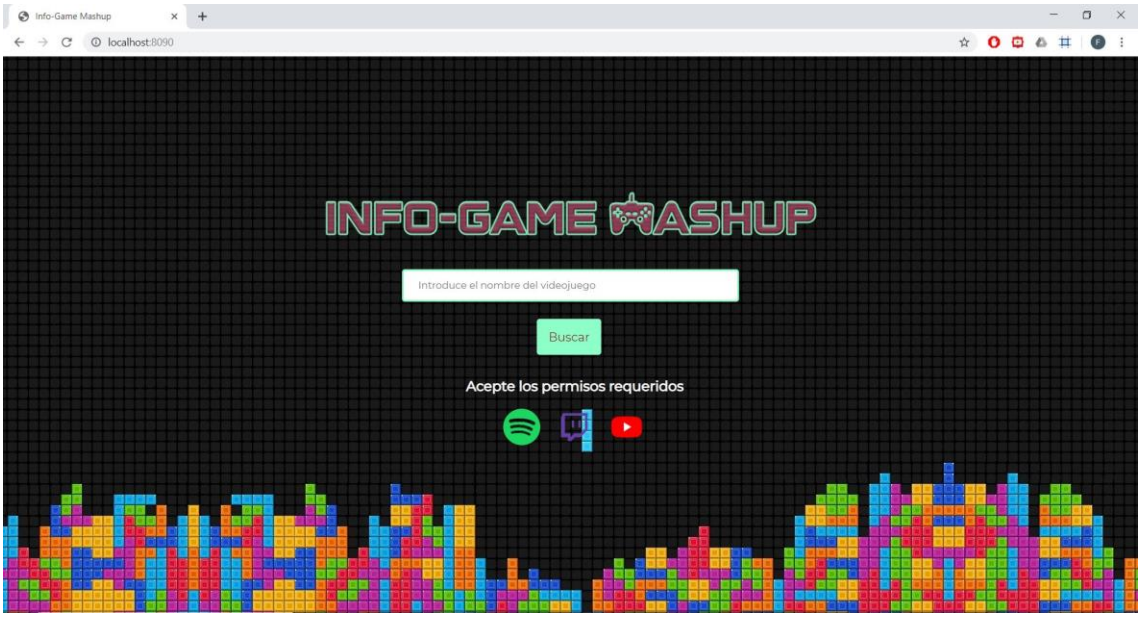
ID	**Prueba 7**
Descripción	Prueba para la detección de errores al implementar búsquedas en Twitch usando servicios RESTful.
Entrada	Se hace uso de la API para invocar al servicio usando la URI 'https://api.twitch.tv/kraken/search/streams?query=doom&limit=3' desde nuestra aplicación.
Salida esperada	La información recobida no es null
Resultado	**ÉXITO**
Automatizada	Sí

ID	**Prueba 8**
Descripción	Prueba para la detección de errores al implementar búsquedas en Twitch usando servicios RESTful.
Entrada	Se hace uso de la API para invocar al servicio usando la URI 'https://api.twitch.tv/kraken/search/streams?query=doom&limit=desde nuestra aplicación.
Salida esperada	El objeto POJO que recibe la información no está vacío
Resultado	**ÉXITO**
Automatizada	Sí

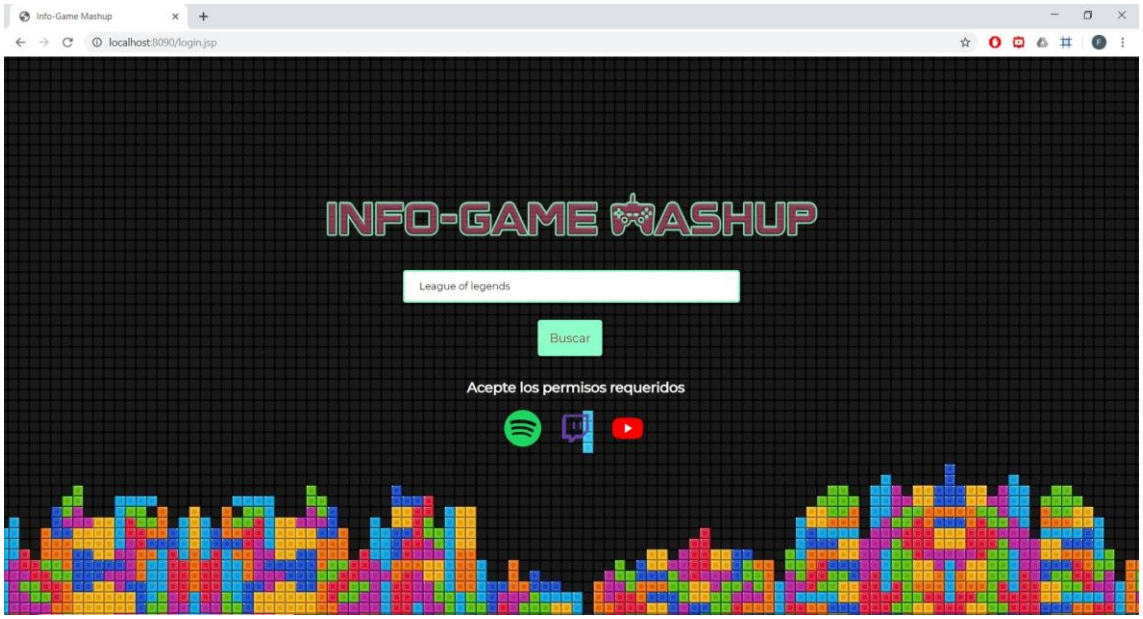
6 Manual de usuario

6.1 Mashup

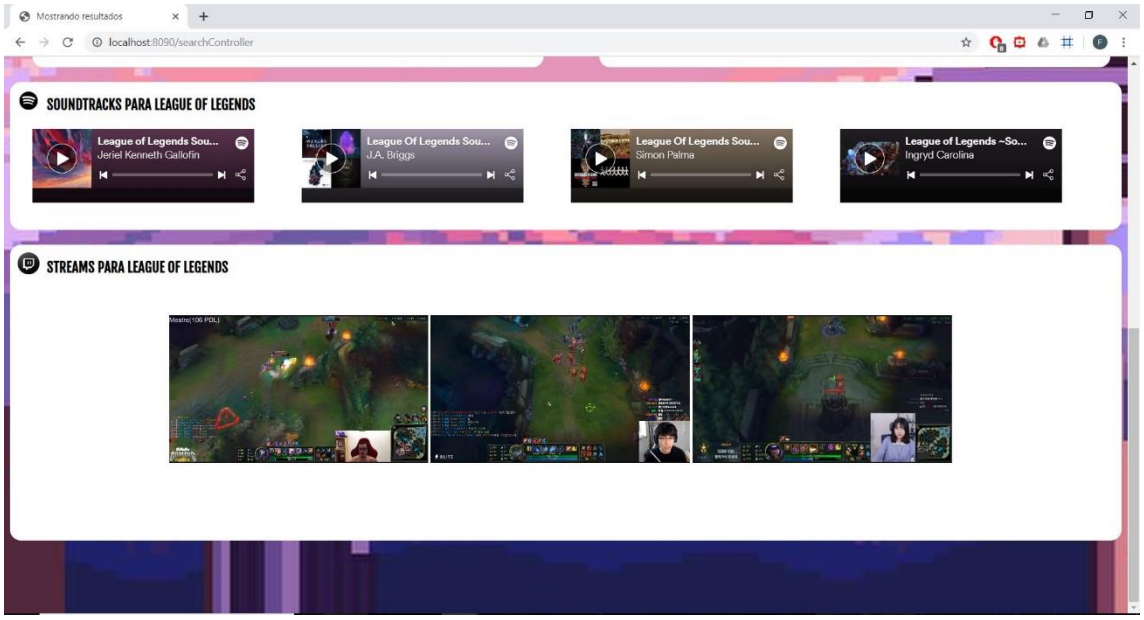
1) Accedemos a la ventana de login mediante la url y aceptamos los permisos requeridos para poder usar la aplicación.



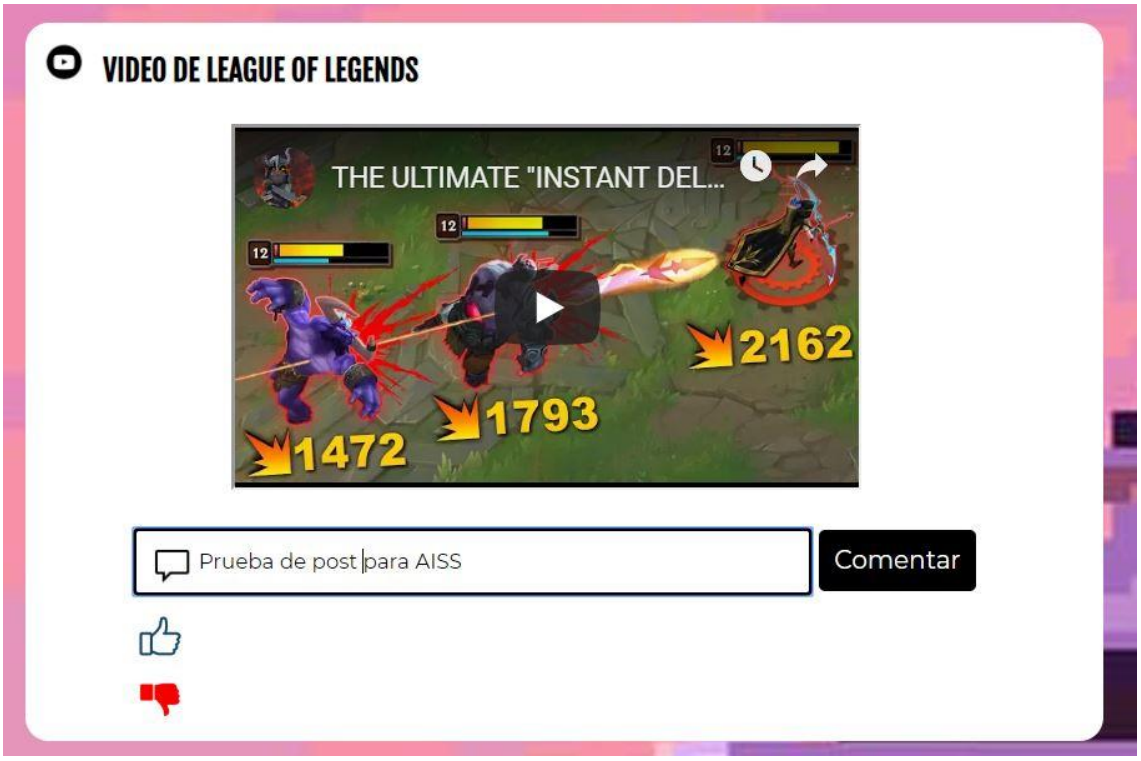
2) En ella introducimos el videojuego a buscar.

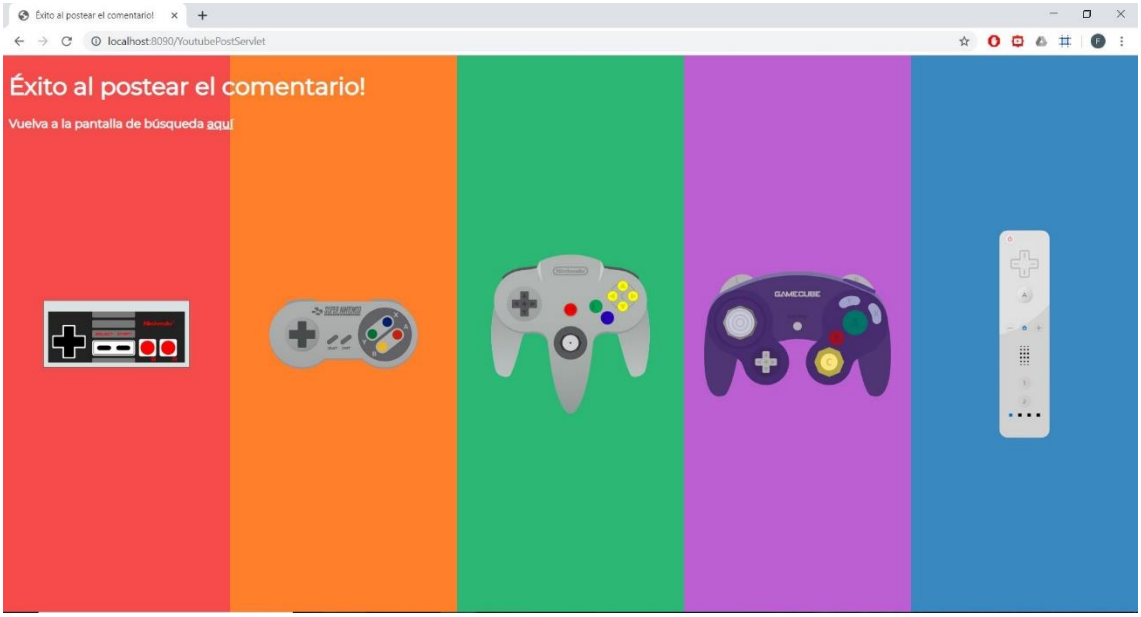


3) SE NOS RETORNARÁ LA VISTA DE RESULTADOS.

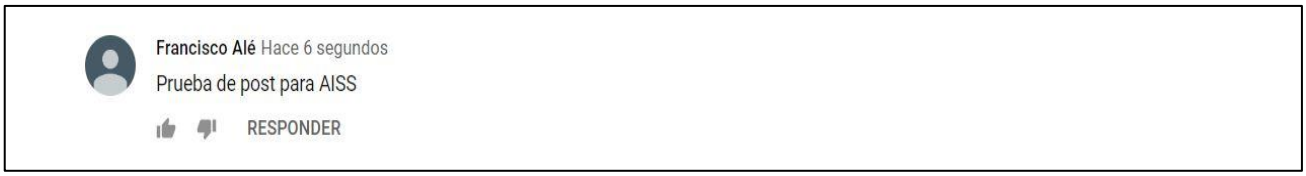


4) Si queremos hacer un post, introducimos nuestro comentario en input de la ventana de YouTube y le damos al botón de comentar. Se nos redirigirá a una ventana de éxito

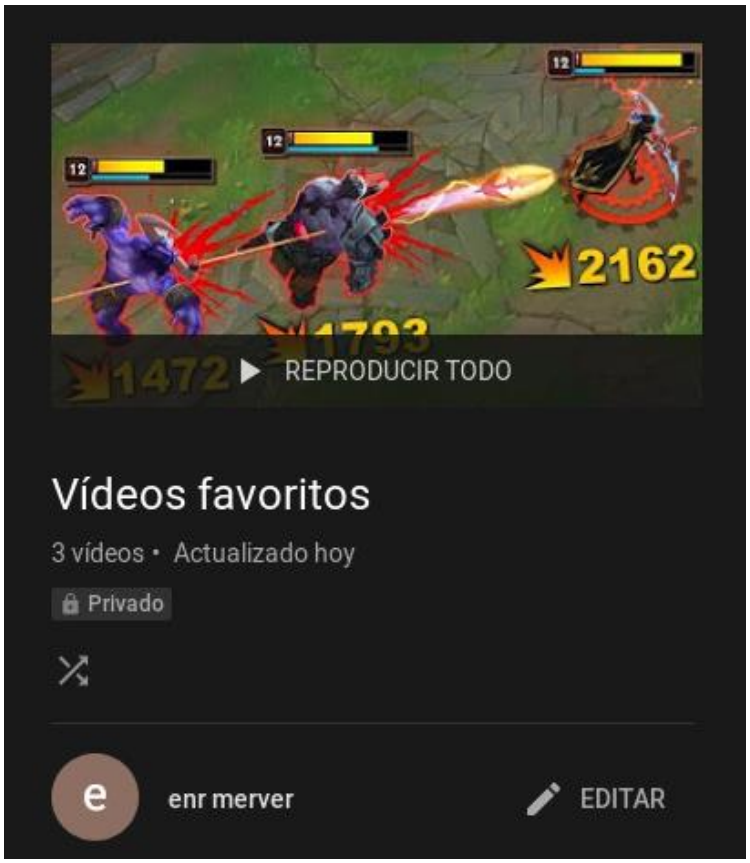




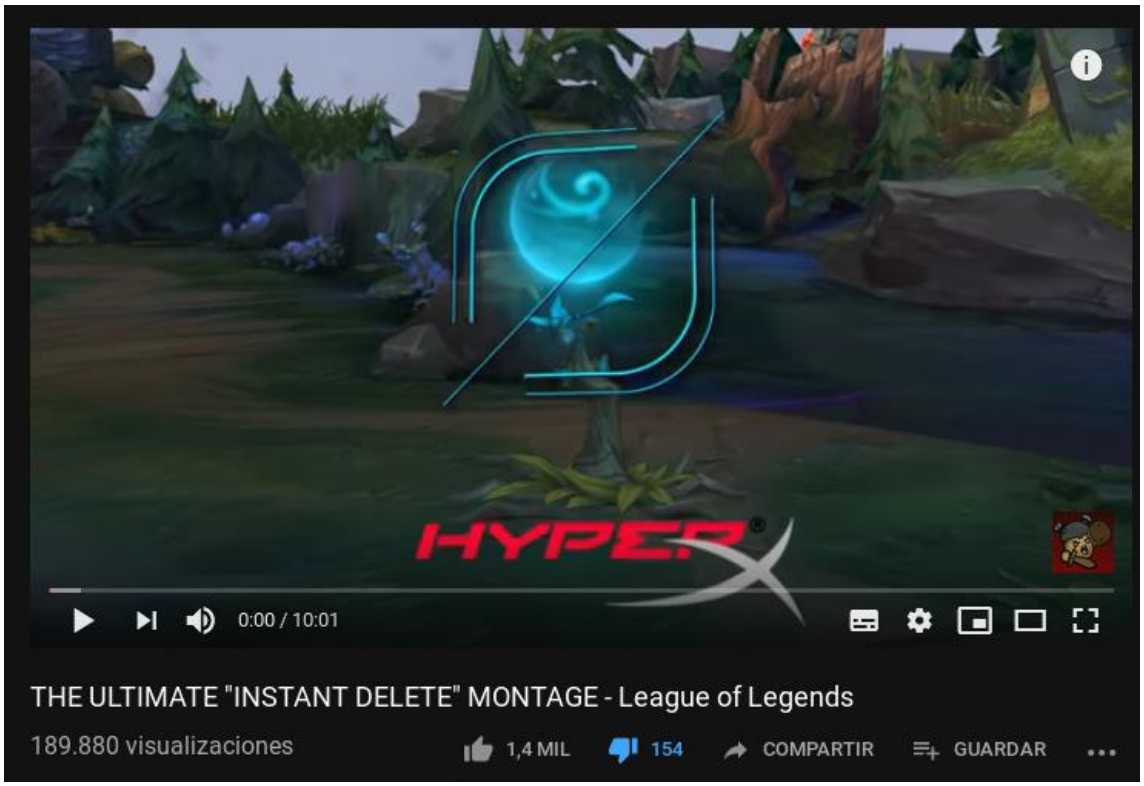
5) Podemos comprobar que se ha realizado el comentario correctamente



6) Si queremos valorar el video solo tenemos que pulsar sobre los iconos de los pulgares y si entramos en YouTube veremos el resultado:



Resultado de la operación like(), se añade a la lista de reproducción de videos que nos ha gustado



Resultado de la operación dislike(), se puede comprobar el dislike en el video seleccionado

6.2 API REST

La documentación viene registrada en el siguiente enlace:

https://app.swaggerhub.com/apis/Info-Game/Info-Game/2.0

Referencias

[1] *Balsamiq*. http://balsamiq.com/.

[2] J. Webber, S. Parastatidis y I. Robinson. *REST in Practice: Hypermedia and Systems Architecture.* O'Reilly Media. 2010.

[3] Swagger. https://swagger.io/.