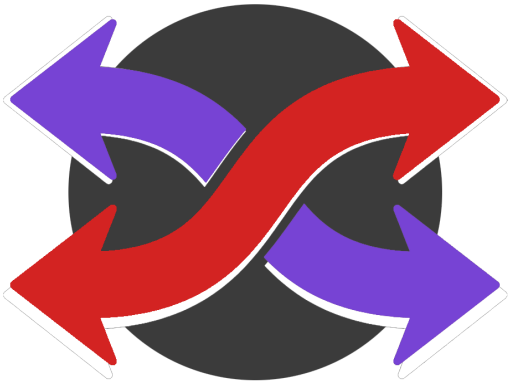
StreaMix



Arquitectura e Integración de Sistemas Software

Grado de Ingeniería del Software

Curso 2019/2020

José Montiel Nieves -- *esoj.nieves@gmail.com*
Pedro Alonso Pontiga -- *pedropontiga@gmail.com*
Rafael Ángel Jiménez Fernández -- *rafaelangeljimenezfer@gmail.com*

Tutor: Alfonso Márquez
Número de grupo: AML Grupo 4

Enlace de la aplicación: https://streamix-aiss.appspot.com/
Enlace de proyecto GitHub: https://github.com/JSnow11/streamix

HISTORIAL DE VERSIONES

Fecha	Versión	Detalles	Participantes
06/03/2020	1.0	- Incluye introducción, prototipos de las interfaces de usuario y diagramas UML de componentes y despliegue.	Antonio José Suárez José Montiel Nieves Pedro Alonso Pontiga Rafael Ángel Jiménez Fernández
03/05/2020	2.0	- Discord se descarta como una de las apis a implementar. - Se han integrado las otras 4 aplicaciones con éxito. - Se ha realizado la documentación sobre la estructura de la aplicación - Se ha implementado la API NotesUp.	José Montiel Nieves Pedro Alonso Pontiga Rafael Ángel Jiménez Fernández
24/05/2020	3.0	-Se ha implementado una funcionalidad que permite visualizar y postear comentarios en youtube. - Está disponible el chat de twitch y es posible comentar en el mismo. - Se han corregido los diagramas de secuencia. - Se han añadido las pruebas y la documentación de las mismas. - Se ha llegado a una versión final de la aplicación.	José Montiel Nieves Pedro Alonso Pontiga Rafael Ángel Jiménez Fernández

Índice

1 Introducción

StreaMix busca facilitar la interacción con las diferentes comunidades del tema del streaming o vídeo que el usuario haya seleccionado. Esto evita que el usuario tenga que abrir diferentes redes sociales y aplicaciones y buscar en ellas el tema en el que está interesado. Además permite al usuario informarse y mantenerse actualizado en el tema desde diferentes fuentes.

StreaMix permite al usuario seguir retransmisiones en directo desde Twitch o YouTube a la vez que consulta opiniones y noticias en directo o interactúa con otros usuarios de la comunidad seleccionada en las diferentes plataformas indicadas: Twitter, Reddit...

1.1 Aplicaciones integradas

- **Twitch** es una plataforma que ofrece un servicio de Streaming de video en vivo, siendo una de sus principales funciones la retransmisión de videojuegos en directo.
- **Twitter** es una red social que se describe como microblogging.
- **YouTube** es un sitio web dedicado a compartir videos,musica etc.
- **Reddit** es un sitio web de marcadores sociales y agregador de noticias donde los usuarios pueden añadir texto, imágenes, vídeos o enlaces donde otros usuarios pueden votar a favor o en contra del contenido, haciendo que aparezcan más o menos destacados.

Nombre aplicación	URL documentación API
Twitch	https://dev.twitch.tv/docs/api
Twitter	https://developer.twitter.com/en/docs/api-reference-index
Youtube	https://developers.google.com/youtube/v3
Reddit	https://www.reddit.com/dev/api/

TABLA 1. APLICACIONES INTEGRADAS

1.2 Evolución del proyecto

- Debido a un error en la matriculación en la asignatura, Antonio García Suárez deja de pertenecer al grupo de desarrollo.

- Se descarta Discord. La API no ofrece la funcionalidad que esperábamos, no nos permite la búsqueda de servidores y canales ni la integración de conexión de voz o bien participación en canales de mensajes de texto.

- Se tienen diversos errores con las respuestas de twitter y se decide parsear los archivos json recibidos con funciones elaboradas manualmente. Se añaden métodos para decodificar unicode ya que estas respuestas se devuelven mal codificadas a pesar de las especificaciones en cuanto a Content-Type y encoding.

- Se tienen problemas con la API v5 (Kraken) de Twitch por lo que se decide usar la nueva API (Helix) a pesar de complicaciones por actualizaciones futuras durante el desarrollo del mashup. (1 Mayo uso de Oauth).

- Hasta el momento se han integrado las 4 aplicaciones restantes y se han conseguido datos de todas ellas aunque aún no procesados de una forma definitiva. Se tienen problemas con la visualización de los posts de Reddit. Al desplegar se obtienen problemas al acceder a la vista /view, se devuelve un error de too many requests, debido al tiempo entre las mismas.

- Se ha implementado la funcionalidad de posteo de comentarios en Youtube.

- Se implementa la posibilidad de mirar el chat de twitch mientras ves un directo, también tienes la capacidad de comentar en directo por el chat.

- Las pruebas automáticas y manuales son exitosas en local.

- En el despliegue se tienen (en ocasiones) errores de too may request respecto a reddit, probablemente por la velocidad a la que se realizan las peticiones.
- Se llega a una versión final del proyecto totalmente funcional.

2 Prototipos de interfaz de usuario

Se proponen las siguientes vistas:

2.1 Vista Inicio

Pantalla inicial al entrar en la web. Tan solo dura unos segundos.

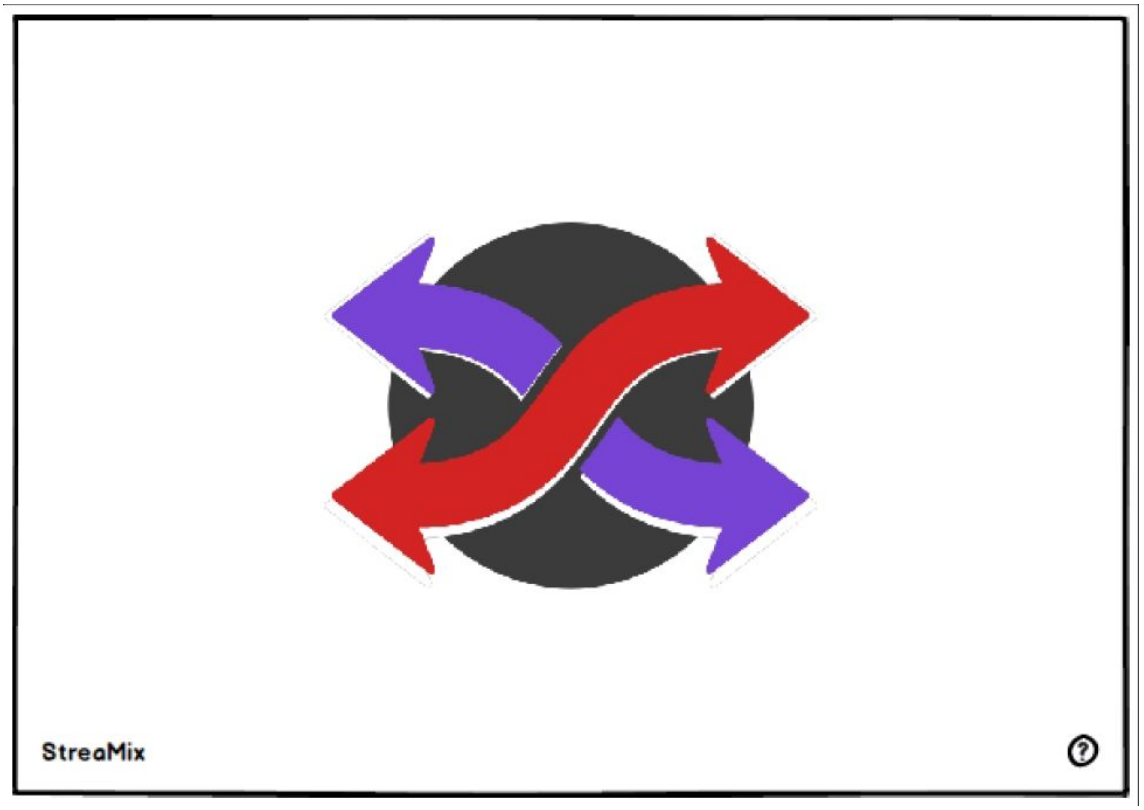


FIGURA 1. PROTOTIPO DE INTERFAZ DE USUARIO DE LA VISTA INICIO

2.2 Vista Página principal

Página inicial. Contiene el logo, una barra de búsqueda y las principales tendencias en Twitch y Twitter.

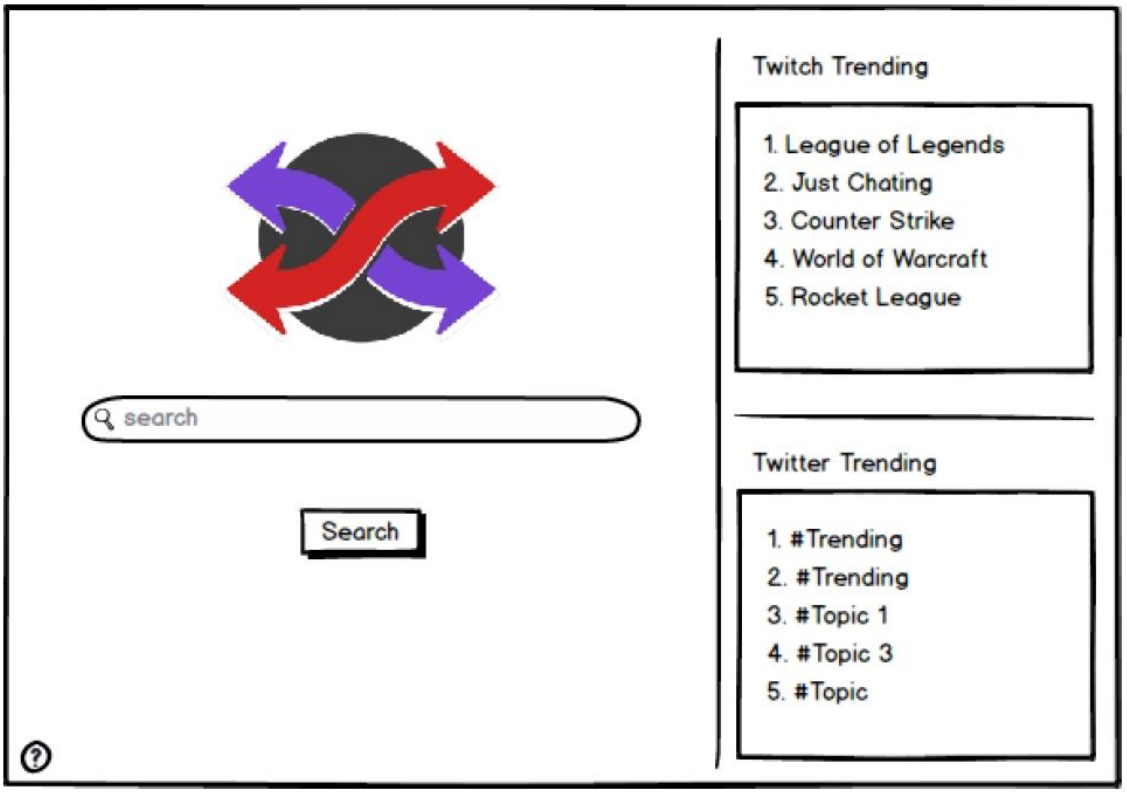


FIGURA 2. PROTOTIPO DE INTERFAZ DE USUARIO DE LA VISTA BUSCADOR

2.3 Vista Búsqueda

Tras realizar una búsqueda o elegir alguna de las tendencias, la aplicación busca en Youtube y en Twitch emisiones en directo relacionadas.

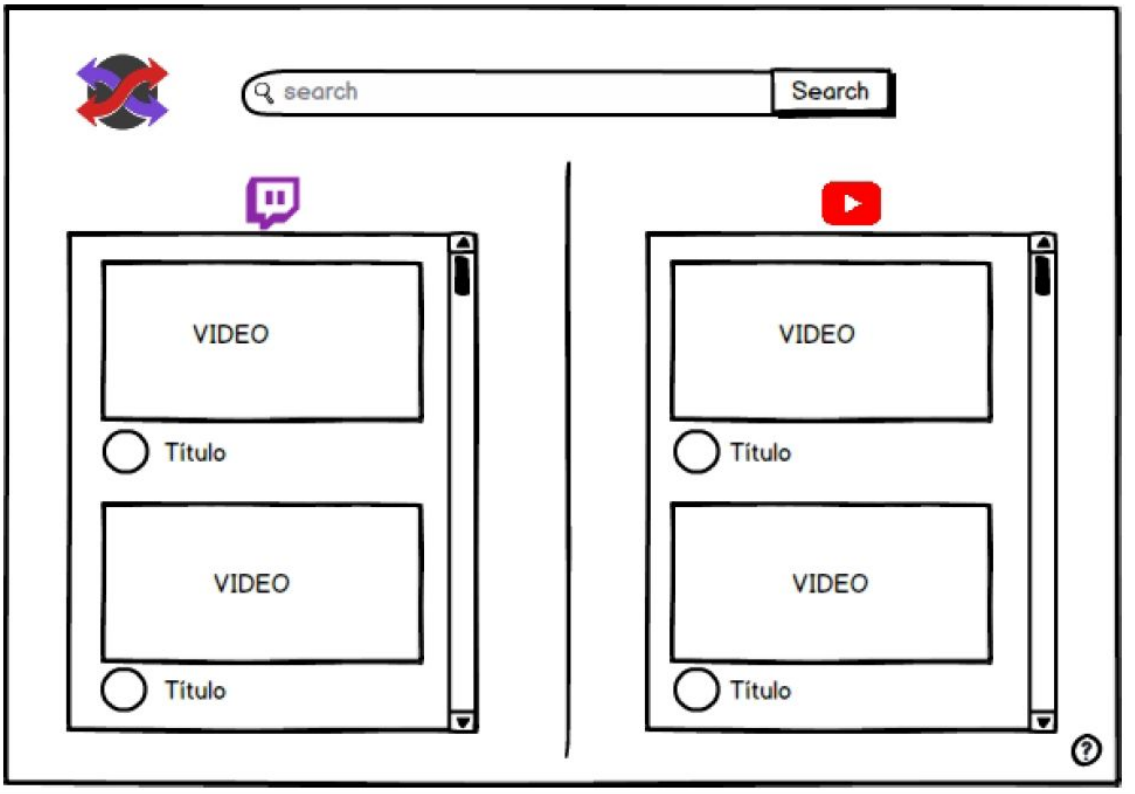


FIGURA 2. PROTOTIPO DE INTERFAZ DE USUARIO DE LA VISTA BÚSQUEDA

2.4 Vista Chat

Una vez el usuario ha escogido un retransmisión es redirigido a una página que contiene una barra de búsqueda, la retransmisión y el chat de la misma ya sea Twitch o Youtube, además de otros tres botones que le permiten cambiar entre las distintas redes sociales.

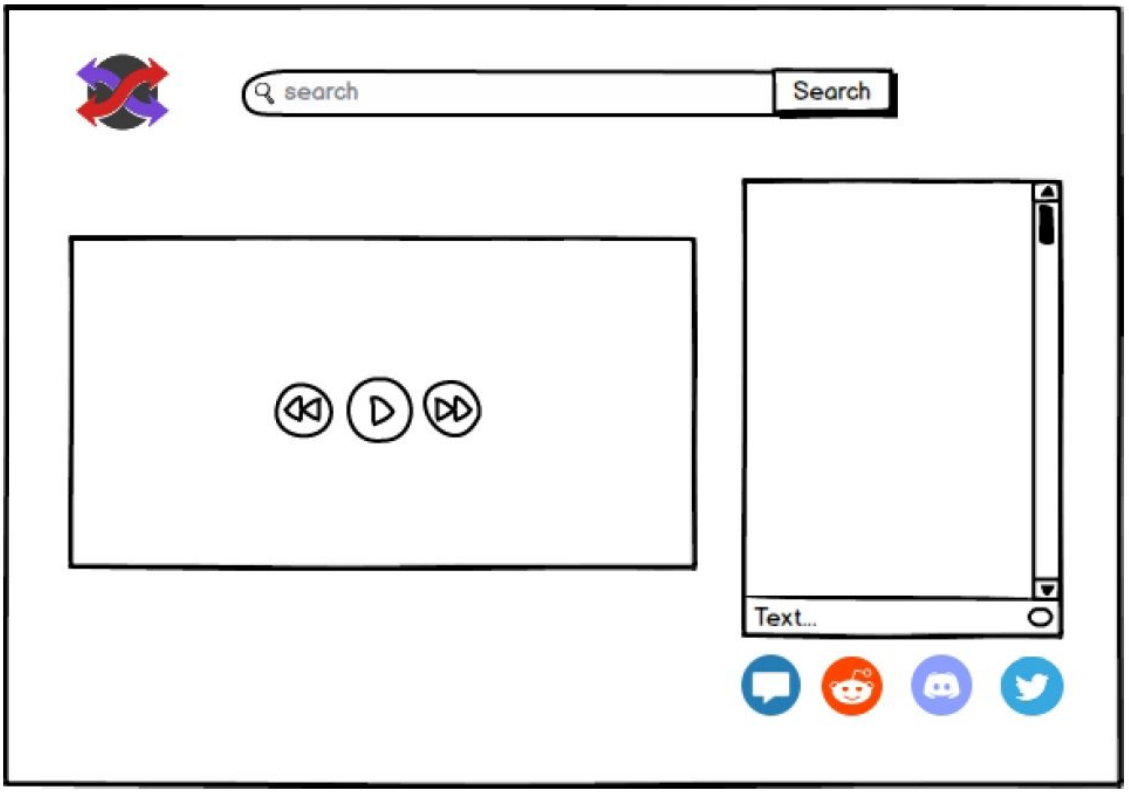


FIGURA 3. PROTOTIPO DE INTERFAZ DE USUARIO DE LA VISTA CHAT

2.5 Vista Reddit

Al igual que la anterior contiene la barra de búsqueda y la retransmisión. Esta vez el chat es substituido por una búsqueda del subreddit más relacionado al tema de la retransmisión.

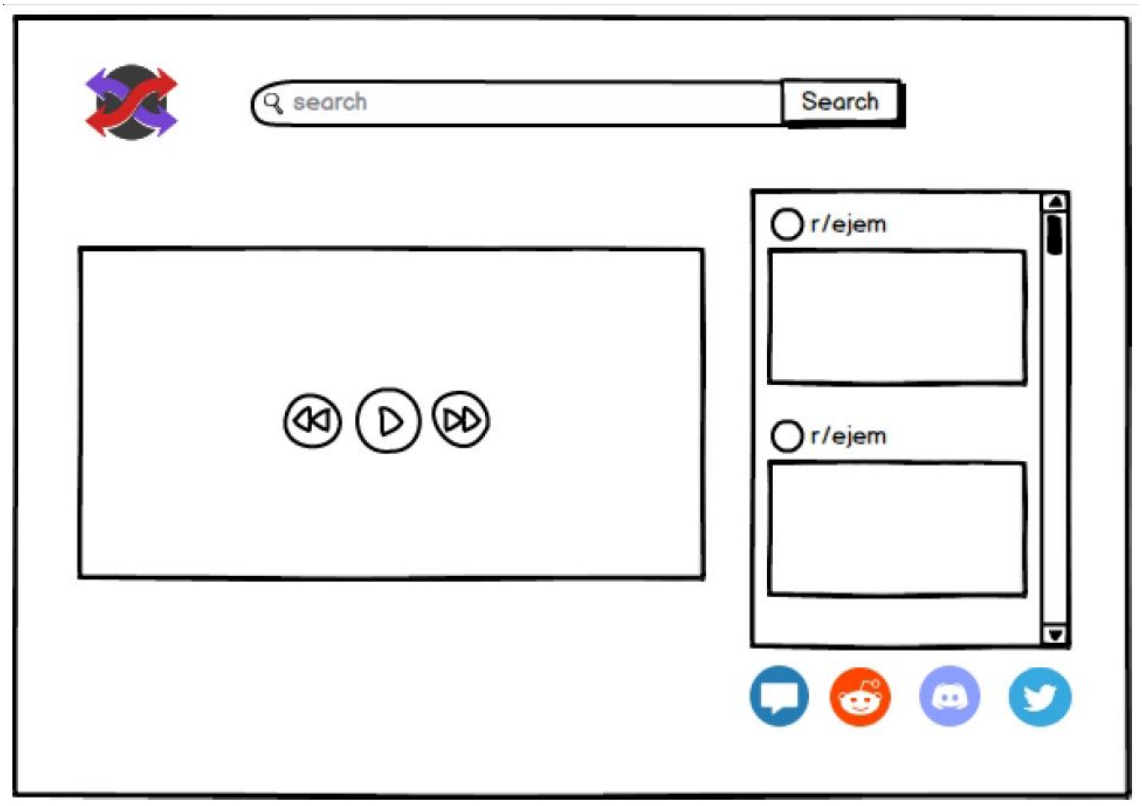


FIGURA 4. PROTOTIPO DE INTERFAZ DE USUARIO DE LA VISTA REDDIT

2.6 Vista Twitter

Al igual que la anterior contiene la barra de búsqueda y la retransmisión. Esta vez el chat es substituido por una búsqueda en Twitter relacionada al tema de la retransmisión.

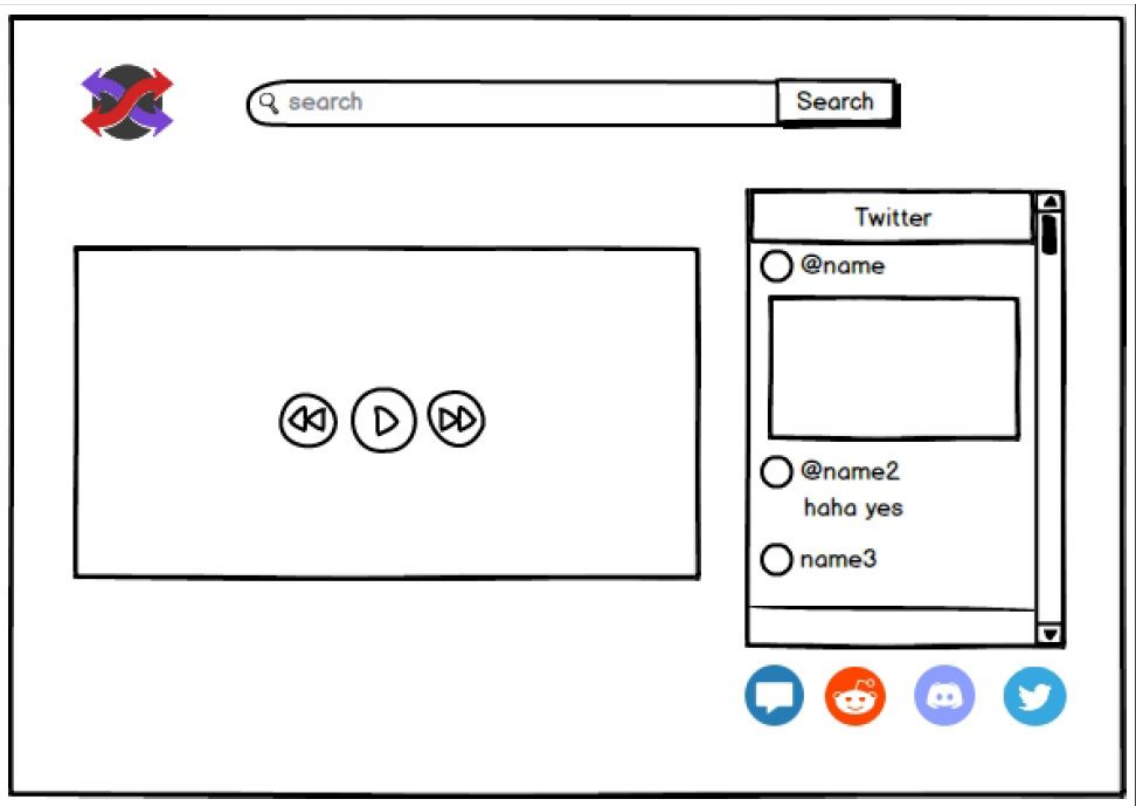


FIGURA 5. PROTOTIPO DE INTERFAZ DE USUARIO DE LA VISTA TWITTE

3 Arquitectura

Insertar los diagramas UML de componentes y de despliegue de la aplicación. Describir textualmente

3.1 Diagrama de componentes

Diagrama UML de componentes de alto nivel.

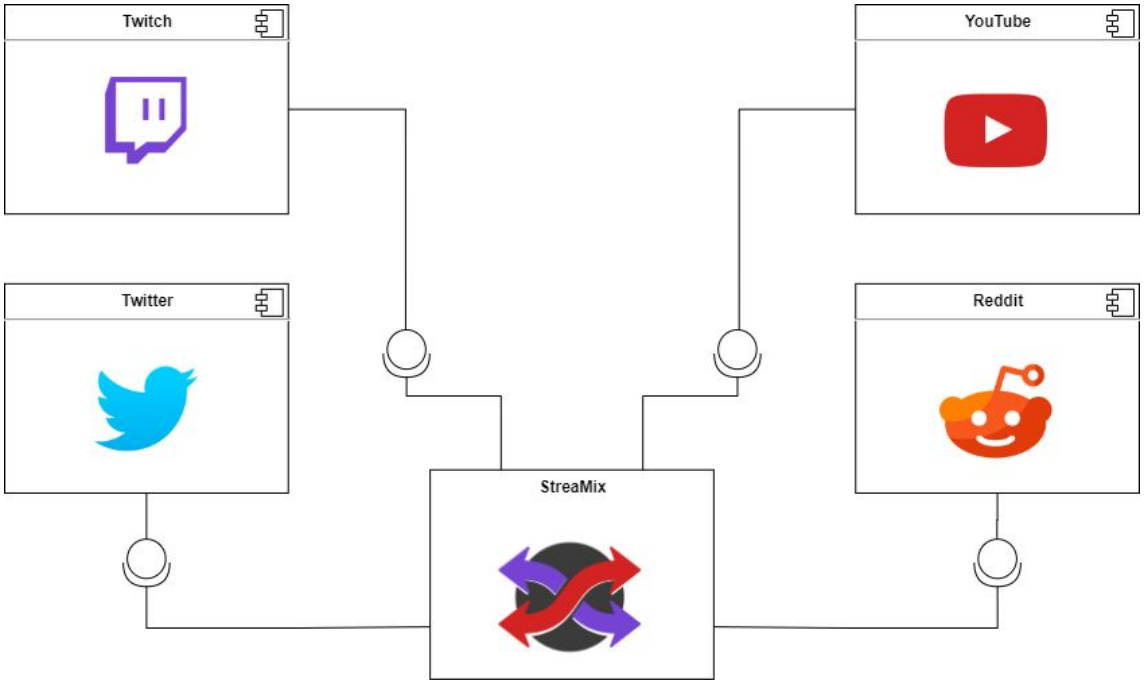


FIGURA 7. UML DE COMPONENTES

3.2 Diagrama de despliegue

Diagrama UML de despliegue de la aplicación.

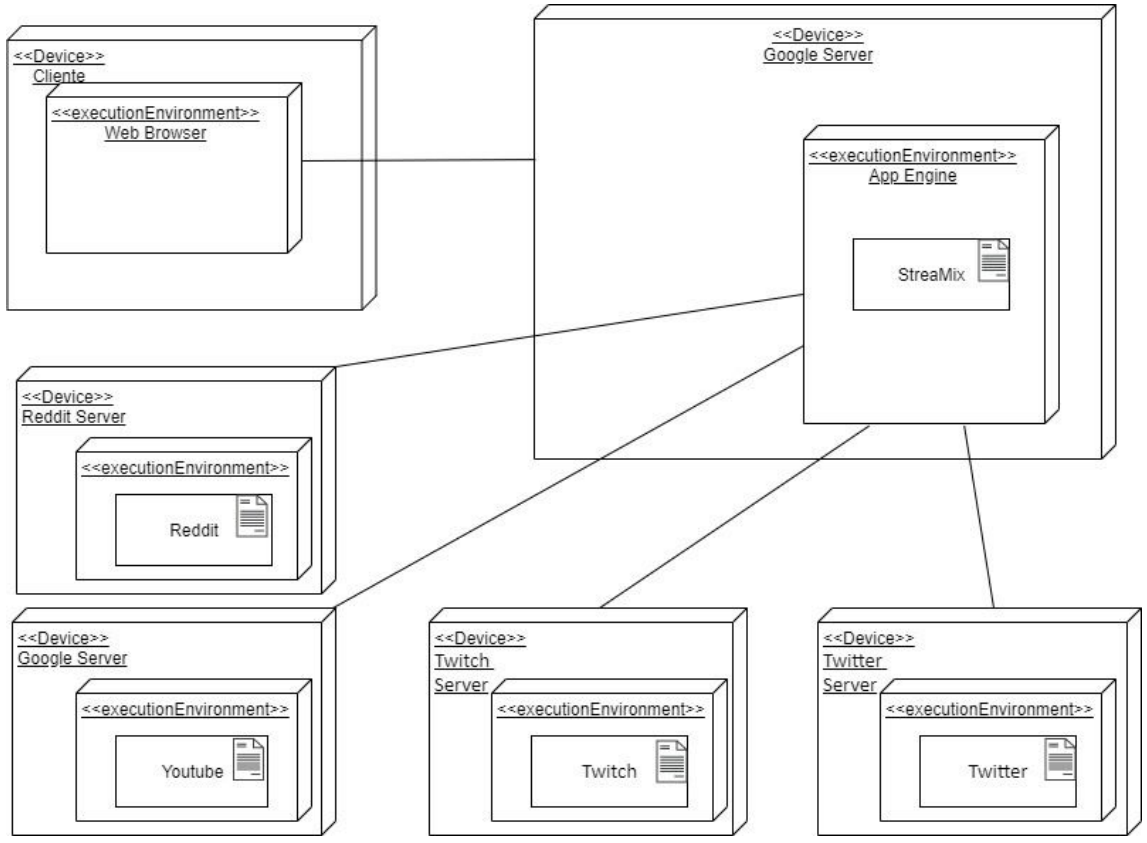


FIGURA 8. UML DE DESPLIEGUE

3.3 Diagrama de secuencia de alto nivel

Diagrama UML de despliegue de la aplicación.

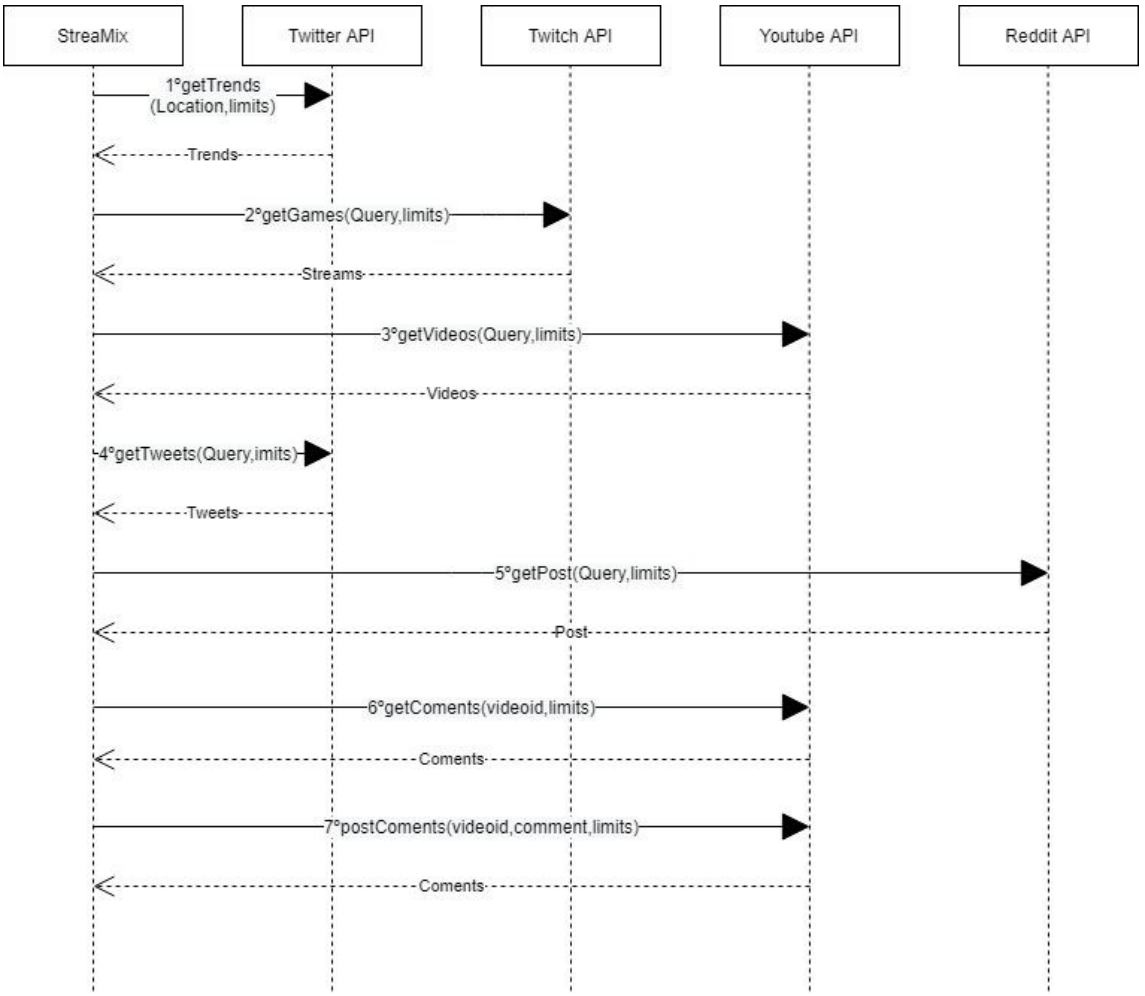


FIGURA 9. UML DE SECUENCIA DE ALTO NIVEL

3.4 Diagrama de clases

Diagrama UML de clases indicando la distribución de las clases entre las distintas capas, según el patrón MVC.

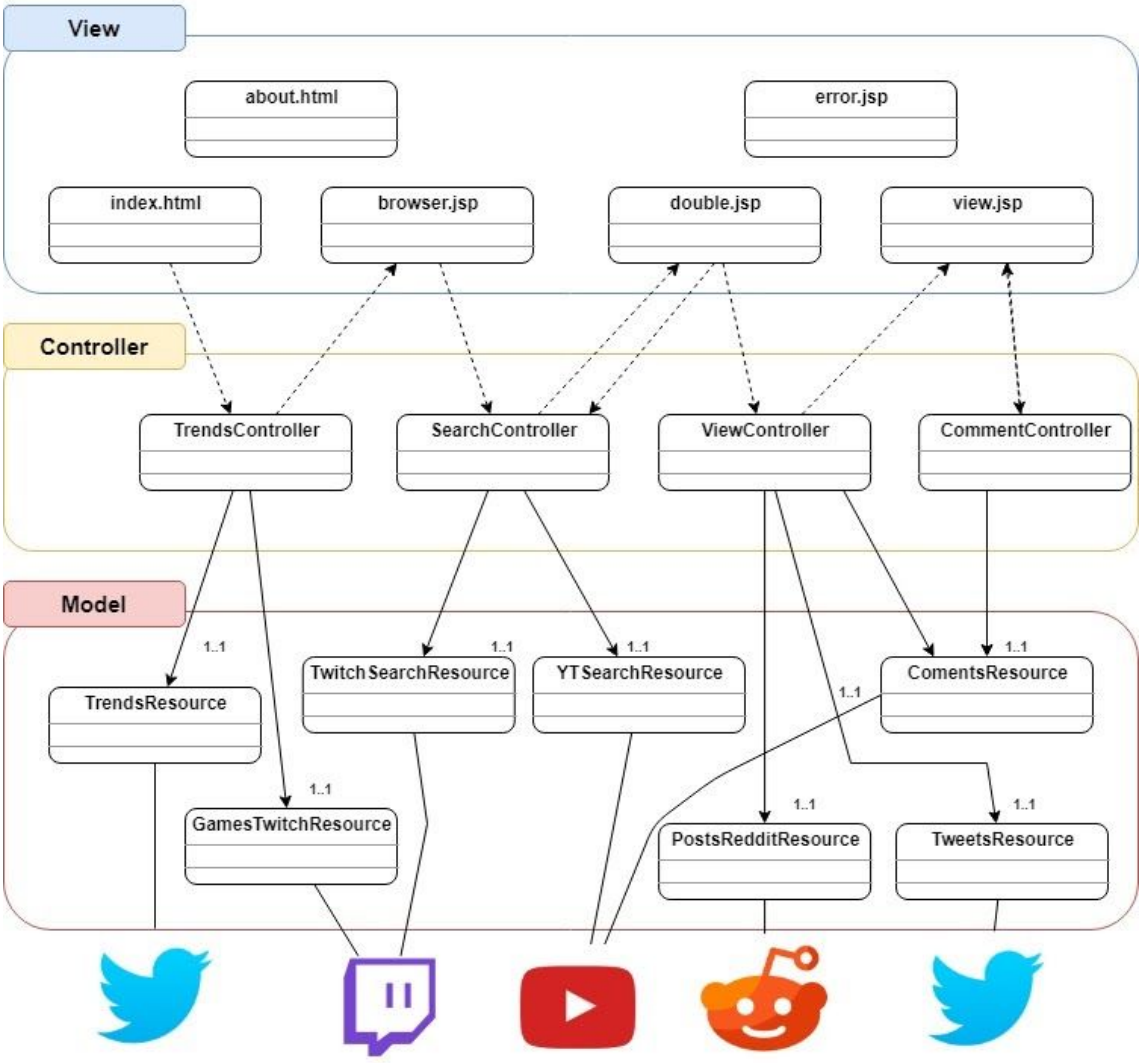
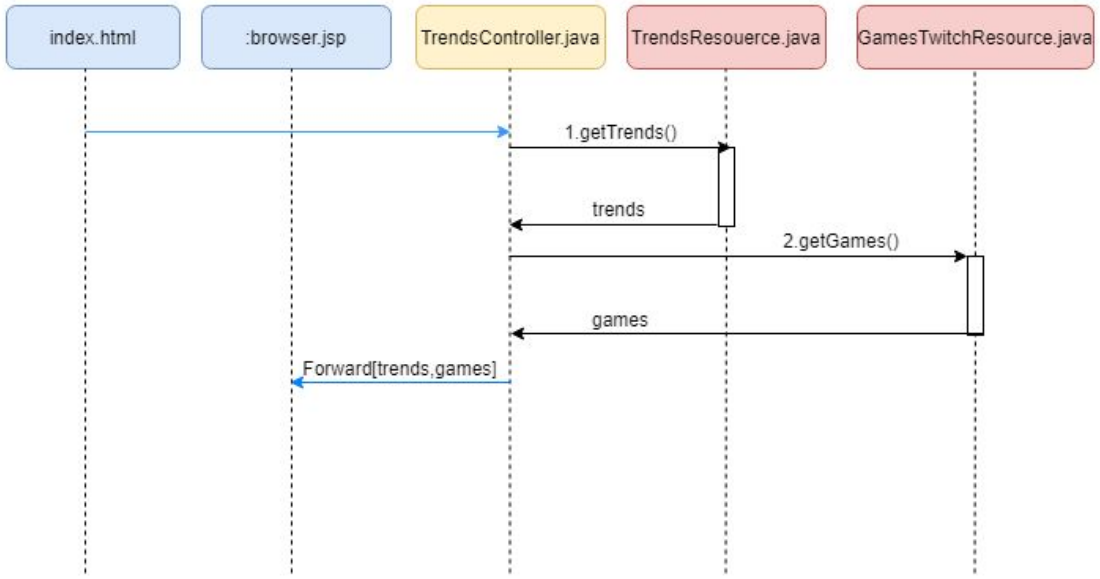


FIGURA 10. UML DE CLASES

3.5 Diagramas de secuencia

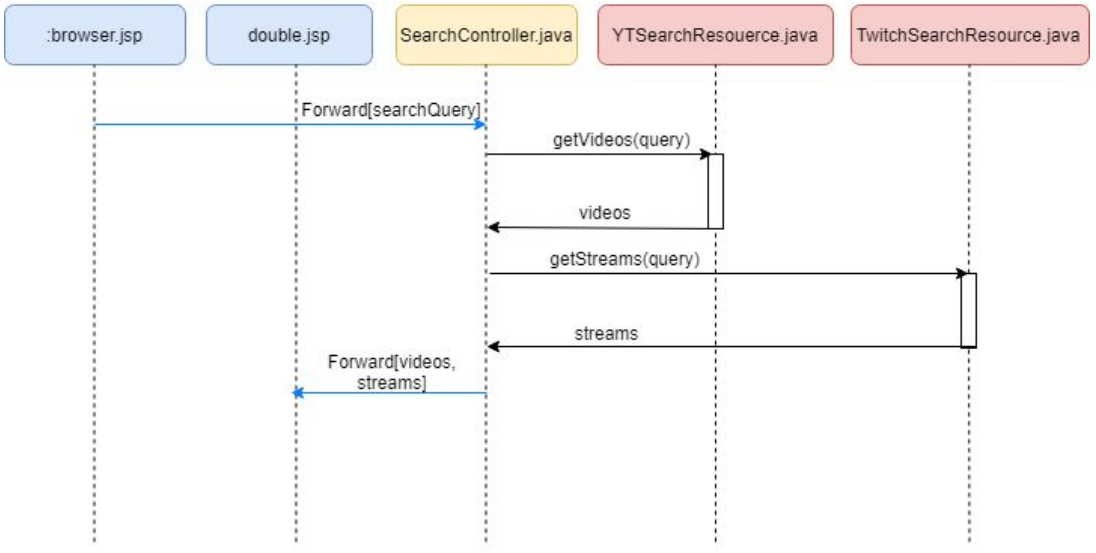
Diagramas UML de secuencia ilustrando la comunicación entre vistas, controladores y clases del modelo.

Petición de trends y games:



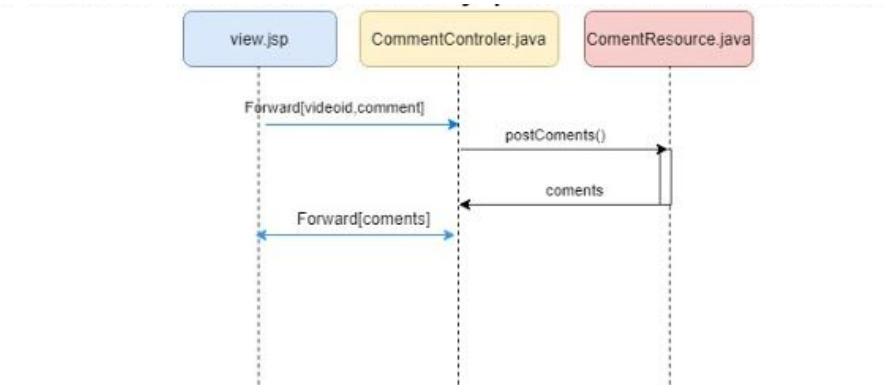
Se hacen las peticiones de tendencias tanto a twitter como a twitch y se muestran en browser.jsp.

Petición videos y streams:



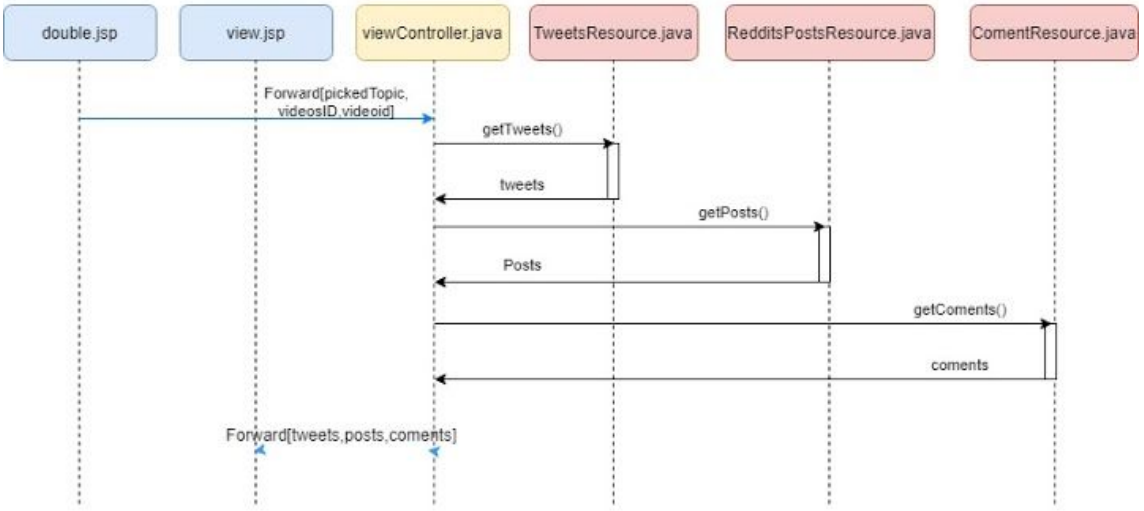
Se hace la petición de videos a Youtube y de stream a Twitch y se muestran en double.jsp.

Postear comentarios



Se postea un comentario.

Petición de Tweets, comentarios y Posts + Embedding



Se solicitan tweets, posts y comentarios y se llevan a view.jsp.

4 Implementación

En la implementación se destaca:

- Los problemas encontrados en cuanto al parseo de recursos de twitter a causa del formato recibido (unicode) han sido solventados con un reemplazamiento de dichos caracteres mediante un algoritmo de decodificación.
- La búsqueda de posts de reddit son tres peticiones anidadas con sus correspondientes procesados de datos.
- Se ha usado Oauth2 mediante bearer tokens en twitch, twitter y youtube.
- Se han elaborado pruebas en JUnit para los diferentes resources.
- Hemos usado variables de sesión para mantener parámetros y consultas tras una redirección para autenticarse.
- Se ha diseñado una botonera de selección mediante javascript.

5 Pruebas

Hemos seguido una estrategia basada en pruebas **sandwich**.

En el nivel del **modelo** comprobamos con pruebas unitarias automatizadas el correcto funcionamiento de nuestras peticiones a apis por separado y, por tanto, el correcto procesado de datos de forma independiente de todos los recursos que necesita nuestra aplicación.

Este tipo de pruebas se identificaran con el nombre de la app integrada seguido de un índice, todas son automáticas.

En el nivel del **controlador** hemos realizado un seguimiento paso a paso mediante logs del correcto funcionamiento de las obtenciones de los recursos del modelo y del recorrido de los mismos por nuestra aplicación hasta llegar a una vista.

En el nivel de **vista** hemos comprobado visualmente que los datos obtenidos, procesados y enrutados en los niveles predecesores ha sido realmente el esperado mostrando todos los datos que nos interesaban y comprobando que son correctos.

Hemos seguido esta estrategia ya que nuestro flujo de trabajo se ha basado en la integración aplicación por aplicación teniendo que pasar de pruebas unitarias a pruebas del controlador y visualizaciones en la vista constantemente. Siempre que un error era encontrado en la vista, requiriendo un enfoque inverso, se ha cambiado la dirección de las pruebas.

Identificación de pruebas:

- Pruebas JUnit : **JUx-X** (**x = índice** y **X = aplicación integrada**).
- Pruebas de flujo mediante logs: **PFx-X** (**x = índice** y **X = controller encargado**).
- Pruebas visuales: **PVx-X** (**x = índice** y **X** = vista jsp).

Resumen	
Número total de pruebas realizadas	15
Número de pruebas automatizadas	9 (60%)

ID	JU1-YT
Descripción	Prueba para la detección de errores al implementar búsquedas de vídeos en Youtube. Se testea una petición get de videos.
Entrada	(Void), se usa un String predefinido para el testeo (búsqueda por palabra).
Salida esperada	Un objeto json con los videos encontrados que es automáticamente parseado a la clase java YTSearch.
Resultado	EXITO
Automatizada	Sí

ID	JU2-YT
Descripción	Prueba para la detección de errores al implementar búsquedas de comentarios de un video de Youtube.
Entrada	(Void),videoID predefinida para el testeo(Comentarios de un video)
Salida esperada	Un objeto JSON con los comentarios del video que es automáticamente parseada a una clase Java YoutubeComents.
Resultado	EXITO
Automatizada	Sí

ID	JU3-YT
Descripción	Prueba para la detección de errores al implementar el post de comentarios en un video de Youtube.
Entrada	(Void), videoID, comment predefinida para el testeo (Postear comentarios en un video).
Salida esperada	Se espera un error ya que la autenticación no está realizada y debe hacerse de forma manual.
Resultado	EXITO
Automatizada	Sí

ID	JU4-TWITTER
Descripción	Prueba para la detección de errores al implementar búsquedas de tendencias en Twitter.
Entrada	(Void)
Salida esperada	Un objeto json con las tendencias de Twitter que es automáticamente parseado a la clase java Trending.
Resultado	EXITO
Automatizada	Sí

ID	JU5-TWITTER
Descripción	Prueba para la detección de errores al implementar búsquedas de tweets dada una cadena de caracteres de entrada.
Entrada	(Void), se usa un String query predefinido para el testeo, buscando tweets que contenga la palabra de entrada.
Salida esperada	Los datos devueltos en formato JSON son mapeados a una clase Java TweetsList y a continuación se muestran por pantalla.
Resultado	EXITO
Automatizada	Sí

ID	JU6-TWITCH
Descripción	Prueba para la detección de errores al implementar la búsqueda de juegos top en Twitch.
Entrada	Void
Salida esperada	Un objeto json con los juegos más populares de Twitch que es automáticamente parseado a la clase java Games.
Resultado	EXITO
Automatizada	Sí

ID	JU7-TWITCH
Descripción	Prueba para la detección de errores al implementar la búsqueda de streams en Twitch.
Entrada	(Void), se usa una game id predefinida para el testeo, buscando streams de ese juego
Salida esperada	Un objeto json con los streams de Twitch de ese juego que es automáticamente parseado a la clase java Streams.
Resultado	EXITO
Automatizada	Sí

ID	JU8-REDDIT
Descripción	Prueba para la detección de errores al implementar la búsqueda de posts en Reddit dada una cadena de caracteres de entrada.
Entrada	(Void), se usa una String query predefinida como entrada para el testeo.
Salida esperada	Los datos devueltos en formato JSON son mapeados a una clase Java Post y a continuación se muestran por pantalla.
Resultado	EXITO
Automatizada	Sí

ID	JU9-REDDIT
Descripción	Prueba para la detección de errores al implementar la búsqueda de subreddit dada una cadena de caracteres de entrada.
Entrada	(Void), se usa una String query predefinida como entrada para el testeo.
Salida esperada	Los datos devueltos en formato JSON son mapeados a una clase Java Post.
Resultado	EXITO
Automatizada	Sí

ID	**PF1-TRENDSCONTROLLER**
Descripción	Seguimiento manual mediante logs del correcto flujo de recursos de la búsqueda de trending topics y top games.
Entrada	Llamada a la URI \trends sin parámetros.
Salida esperada	Puntos clave en el flujo con logs por consola: ● Inicio de búsqueda. ● Respuesta de petición de trends. ● Respuesta de petición games. ● Envío de los recursos. ● Posibles errores. Trending topics y top games.
Resultado	**EXITO**
Automatizada	No

ID	**PF2-SEARCHCONTROLLER**
Descripción	Seguimiento manual mediante logs del correcto flujo de recursos de la búsqueda de videos y streams.
Entrada	Llamada a la URI \search con una String searchQuery.
Salida esperada	Puntos clave en el flujo con logs por consola: ● Inicio de búsqueda. ● Procesado de la query. ● Respuesta de petición de videos. ● Respuesta de petición streams. ● Envío de los recursos. ● Posibles errores. Videos y Streams.
Resultado	**EXITO**
Automatizada	No

ID	**PF3-VIEWCONTROLER**
Descripción	Seguimiento manual mediante logs del correcto flujo de recursos de la búsqueda de tweets, post y coments además del enrutamiento del video/stream.
Entrada	Llamada a la URI \view con una String videoID,String streamID y String pickedTopic.
Salida esperada	Puntos clave en el flujo con logs por consola: ● Inicio de búsqueda. ● Procesado del pickedTopic y el videoID/streamID. ● Respuesta de petición de tweets. ● Respuesta de petición de post. ● Respuesta de petición de comments. ● Envío de los recursos. ● Posibles errores. Tweets, Posts, videoID, streamID, Comments.
Resultado	**EXITO**
Automatizada	No

ID	**PF4-COMMENTCONTROLLER**
Descripción	Seguimiento manual mediante logs del correcto flujo de recursos del post de comentarios de un video de Youtube.
Entrada	Llamada a la URI \comment con un String videoID y un String comment.
Salida esperada	Puntos clave en el flujo con logs por consola: ● Inicio de búsqueda. ● Procesado del videoID. ● Procesado del accessToken. ● Post del comentario. ● Autentificación. ● Envío de los recursos. ● Posibles errores. Comentarios.
Resultado	**EXITO**
Automatizada	No

ID	**PV1-BROWSER**
Descripción	Visualización de la salida del TrendsController.
Entrada	Llamada a la URI \trends.
Salida esperada	Presentación visual: ● Trending topics. ● Top games.
Resultado	**EXITO**
Automatizada	No

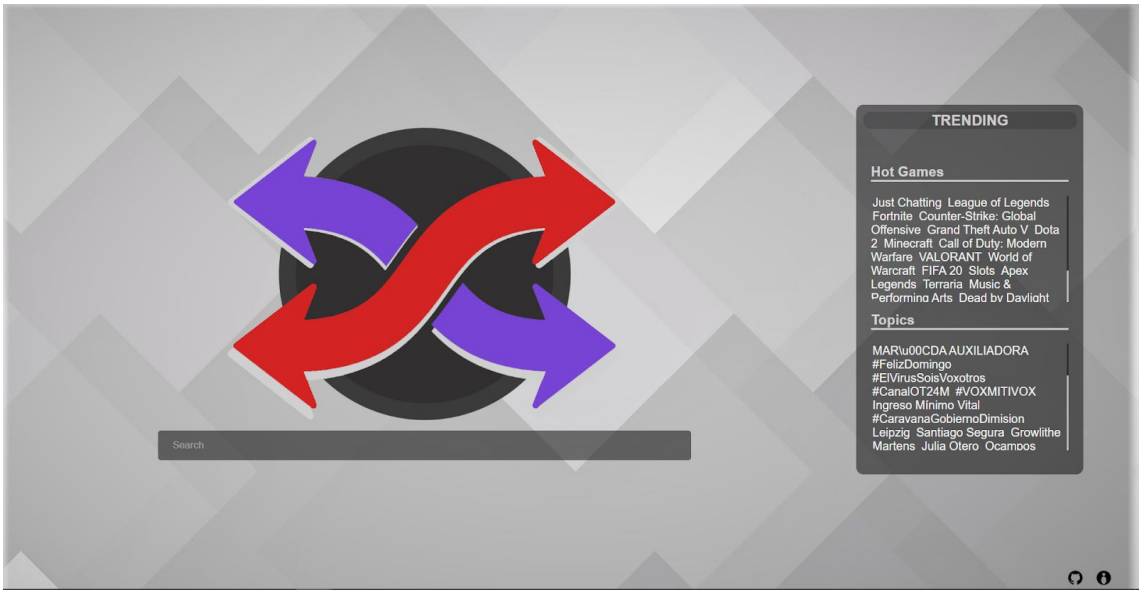
ID	**PV2-DOUBLE**
Descripción	Visualización de la salida del searchControler.
Entrada	Llamada a la URI \search con una String query.
Salida esperada	Presentación visual: ● Videos de youtube. ● Streams de twitch. ● Posibles errores.
Resultado	**EXITO**
Automatizada	No

ID	**PV3-VIEW**
Descripción	Visualización de la salida del ViewController.
Entrada	Llamada a la URI \view con un videoID o streamID.
Salida esperada	Presentación visual: ● Video o Stream embedded. ● Tweets. ● Posts. ● Comments.
Resultado	**EXITO**
Automatizada	No

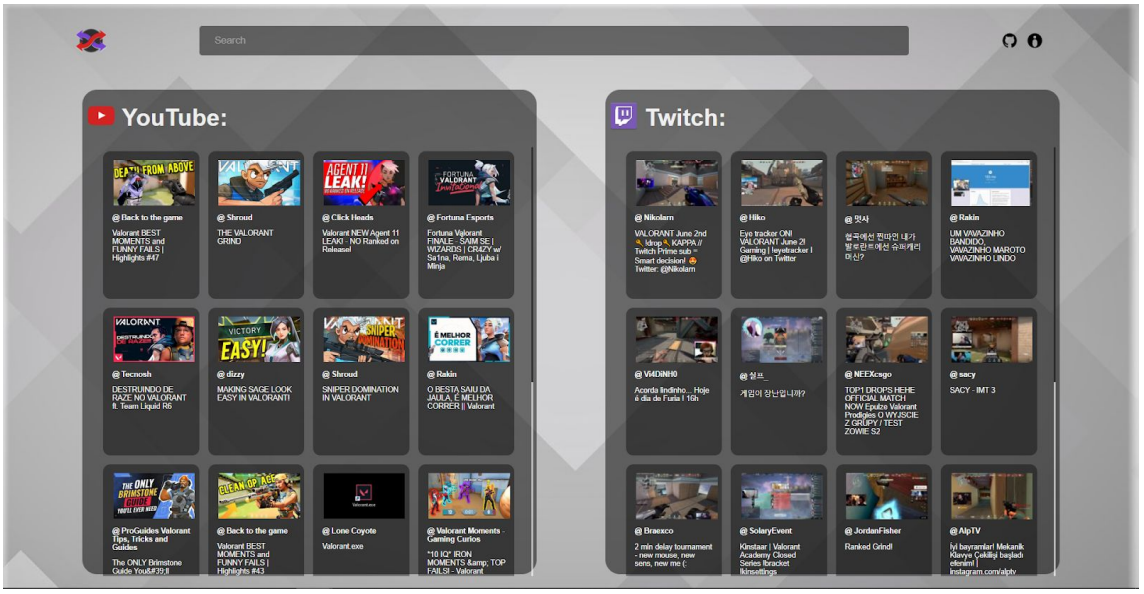
6 Manual de usuario

6.1 Mashup

Se pulsa en el logo para ir al buscador.



A la derecha de la pantalla podemos ver en la parte superior los juegos top de Twitch y en la inferior los Topics de Twitter, pueden ser usados como enlaces directos de búsqueda. Debajo del logo se encuentra la barra de búsqueda en la que introducimos lo que queremos buscar en Youtube y Twitch (por ahora solo se muestra YT).



Una vez introducido lo que queremos buscar no lleva a esta vista, donde nos ofrecerá una lista de videos de Youtube y de streams de Twitch relacionados con lo que hemos buscado. Elegiremos entre Youtube o Twitch.

Una vez elegida la plataforma donde se desea ver el contenido nos lleva a esta vista, donde además de poder ver el video, a la derecha se podrán consultar tweets y posts de Reddit relacionados con el tema que se ha buscado hace dos vistas. Se puede navegar entre los tweets, posts y comments mediante los botones blancos. En el botón de comentarios se puede postear un comentario.

6.2 API REST

Se ha desarrollado la API de NotesUP, una aplicación de notas categorizadas por labels.

Se ofrecen los siguientes enlaces:

- **Documentación desplegada:**

https://notesup-276114.ew.r.appspot.com/docs/

- **Fichero yaml:**

https://notesup-276114.ew.r.appspot.com/docs/StreaMix-NotesUpApplication-1.0.0-swagger.yaml

- **Desarrollo de la documentación en swaggerhub:**

https://app.swaggerhub.com/apis/StreaMix/NotesUpApplication/1.0.0

Referencias

[1] *Balsamiq*. http://balsamiq.com/. Accedido en Enero 2014.

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

[3] Myers, G.J. The art of software testing . Wiley. 2ª edición . 2004. Capítulo 2).