**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 projETSII, GitHub o similar:

<https://projetsii.informatica.us.es/projects/3hjhkcggclvfvjlfn6j>

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 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Índice

[1 Introducción 4](#_Toc3535431)

[1.1 Aplicaciones integradas 4](#_Toc3535432)

[1.2 Evolución del proyecto 5](#_Toc3535433)

[2 Prototipos de interfaz de usuario 6](#_Toc3535434)

[2.1 Vista Inicial 6](#_Toc3535435)

[2.2 Vista de la Aplicación 7](#_Toc3535436)

[3 Arquitectura 8](#_Toc3535437)

[3.1 Diagrama de componentes 8](#_Toc3535438)

[3.2 Diagrama de despliegue 8](#_Toc3535439)

[3.3 Diagrama de secuencia de alto nivel 9](#_Toc3535440)

[3.4 Diagrama de clases 9](#_Toc3535441)

[3.5 Diagramas de secuencia 9](#_Toc3535442)

[4 Implementación 10](#_Toc3535443)

[5 Pruebas 11](#_Toc3535444)

[6 Manual de usuario 12](#_Toc3535445)

[6.1 Mashup 12](#_Toc3535446)

[6.2 API REST 12](#_Toc3535447)

[Referencias 13](#_Toc3535448)

# 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.

## Aplicaciones integradas

**IGDB**: con la API de IGDB podremos extraer información de los juegos.

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

**Reddit:** con esta API podremos acceder a foros relacionados con el videojuego buscado.

**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 |
| IGDB | <https://api-docs.igdb.com/> |
| Twitch | <https://dev.twitch.tv/docs/v5/> |
| Reddit | <https://www.reddit.com/dev/api> |
| Spotify | <https://developer.spotify.com/documentation/web-api/> |
| YouTube | <https://developers.google.com/youtube/> |

Tabla 1. Aplicación integradas

## 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 Reddit, 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. Hubo que descartar esta opción, ya que es una API que contiene muy poca información de videojuegos.

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.

# Prototipos de interfaz de usuario

## 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

## 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.

# Arquitectura

## Diagrama de componentes

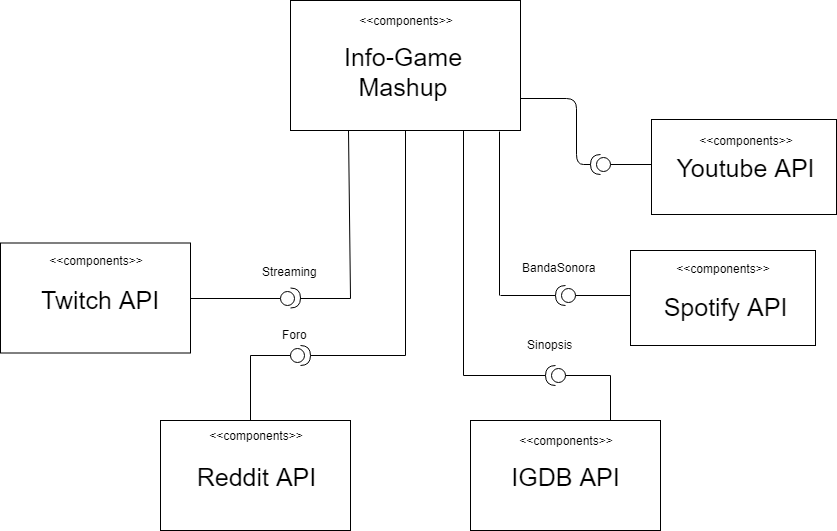


Figura 3. Diagrama de componentes.

## Diagrama de despliegue

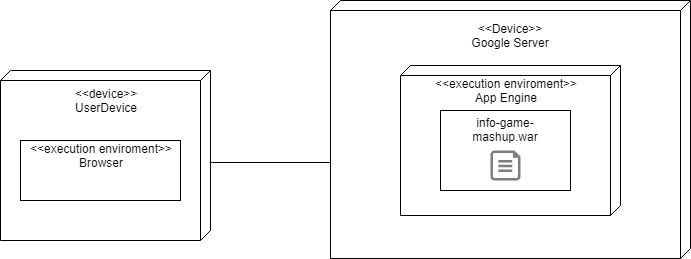


Figura 4. Diagrama de despliegue.

## Diagrama de secuencia de alto nivel

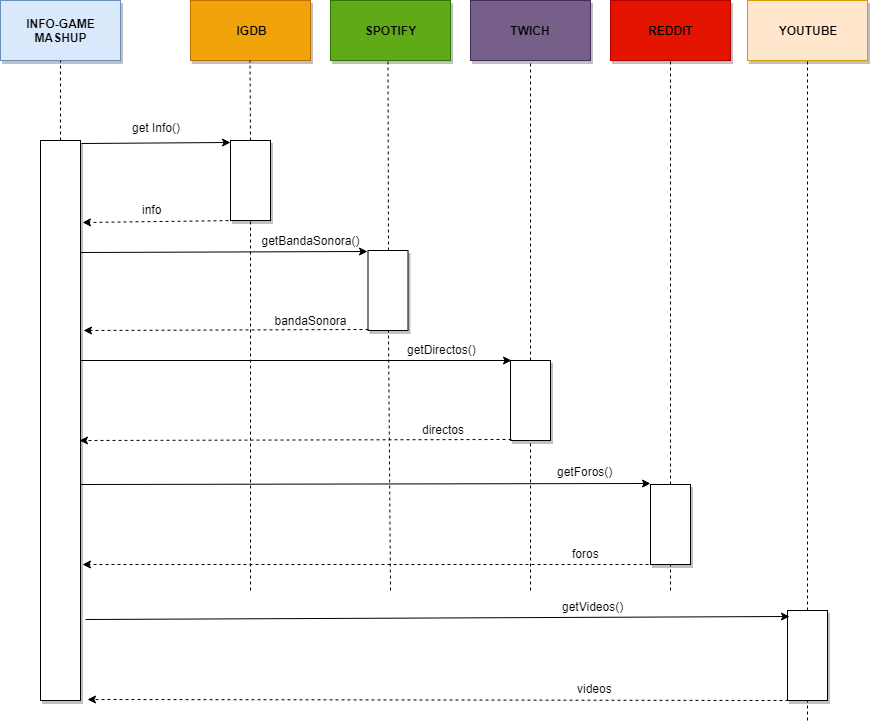


FIGURA 5. DIAGRAMA DE SECUENCIA.

## Diagrama de clases

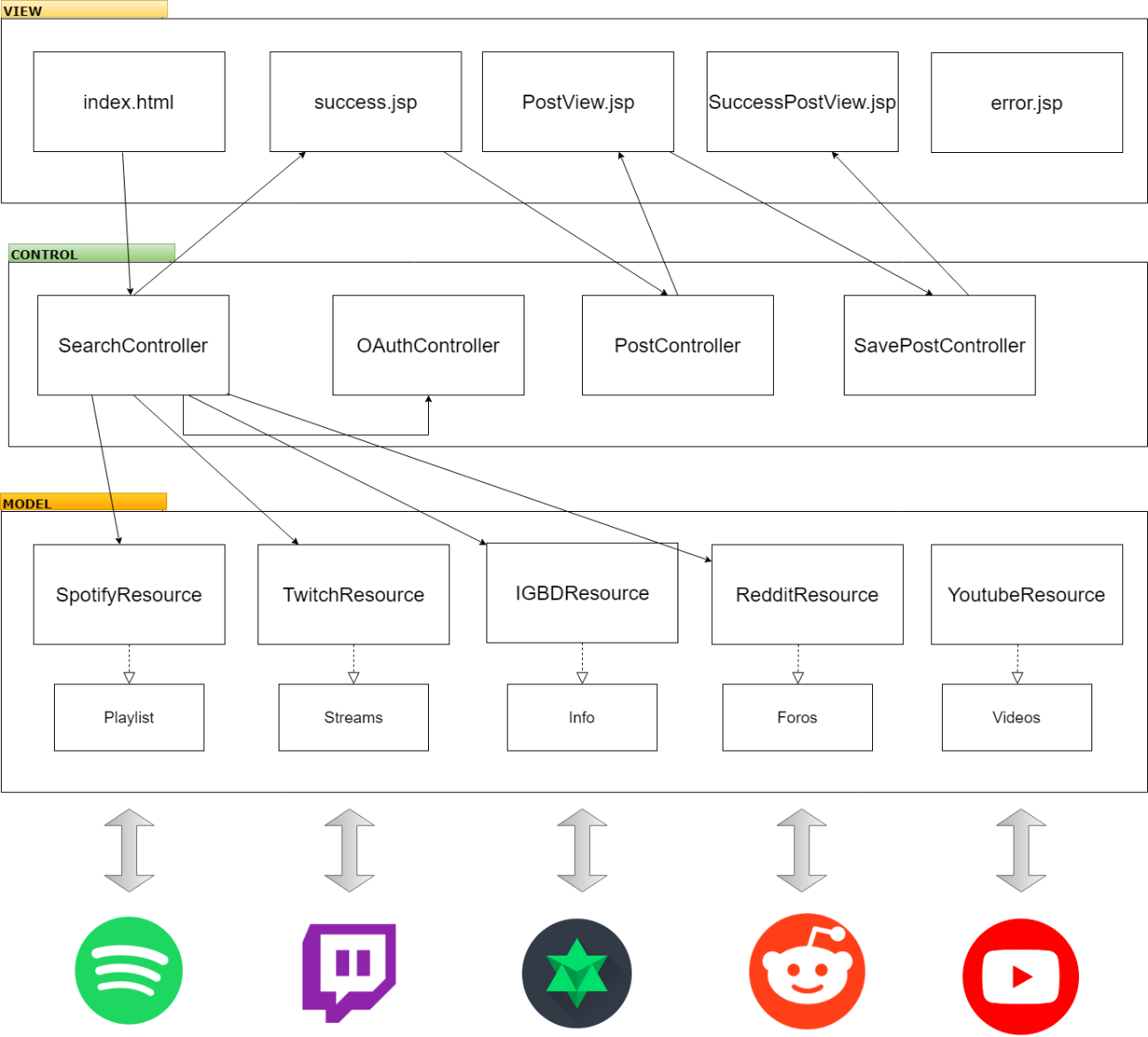


FIGURA 6. DIAGRAMA DE clases.

## Diagramas de secuencia

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

# Implementación

Describir brevemente los aspectos de la implementación que creen da más mérito al trabajo. Añadir algún fragmento de código si se considera oportuno.

# 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 | 25 |
| Número de pruebas automatizadas | 20 (80%) |

|  |  |
| --- | --- |
| 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 Spotify usando servicios de su API. |
| 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 | 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 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 | El objeto POJO creado con los resultados de la petición no están vacios. |
| Resultado | **EXITO** |
| Automatizada | Sí |

# Manual de usuario

## Mashup

Indique textualmente e **incluyendo capturas de pantalla** el manual de uso del mashup.

## API REST

La documentación viene registrada en el siguiente enlace:

<https://app.swaggerhub.com/apis/Info-Game/Info-Game/1.0.1>

# 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.