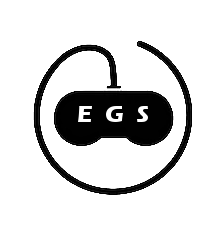
**Easy Game Searcher (EGS)**



Arquitectura e Integración de Sistemas Software

Grado de Ingeniería del Software

Curso ISW-G3

Eduardo Ignacio Vázquez Vázquez(edui1212@gmail.com)

Javier Pérez Martín (javierperezmartin.espartano98@gmail.com)

Pablo Muñoz Del Bot (pabmundel@gmail.com)

Tutor: Javier Troya Castilla

Número de grupo: 3

Enlace de la aplicación: <https://easy-game-searcher.appspot.com/>

Enlace de proyecto en projETSII, GitHub o similar: https://github.com/pabmundel/AISS/

Historial de versiones

|  |  |  |  |
| --- | --- | --- | --- |
| Fecha | Versión | Detalles | Participantes |
| 10/03/2014 | 1.0 | - Incluye introducción, prototipos de las interfaces de usuario y diagramas UML de componentes y despliegue. | -Eduardo Ignacio Vázquez Vázquez  -Javier Pérez Martín  -Pablo Muñoz Del Bot |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Índice

[1 Introducción 4](#_Toc471899224)

[1.1 Aplicaciones integradas 4](#_Toc471899225)

[1.2 Evolución del proyecto 5](#_Toc471899226)

[2 Prototipos de interfaz de usuario 5](#_Toc471899227)

[2.1 Vista X 5](#_Toc471899228)

[2.2 Vista Y 6](#_Toc471899229)

[3 Arquitectura 7](#_Toc471899230)

[3.1 Diagrama de componentes 7](#_Toc471899231)

[3.2 Diagrama de despliegue 8](#_Toc471899232)

[3.3 Diagrama de secuencia de alto nivel 9](#_Toc471899233)

[3.4 Diagrama de clases 9](#_Toc471899234)

[3.5 Diagramas de secuencia 10](#_Toc471899235)

[4 Implementación 11](#_Toc471899236)

[5 Pruebas 12](#_Toc471899237)

[6 Manual de usuario 13](#_Toc471899238)

[6.1 Mashup 13](#_Toc471899239)

[6.2 API REST 13](#_Toc471899240)

[Referencias 14](#_Toc471899241)

# Introducción

Con esta aplicación se busca solventar que cuando se busca información de un videojuego, tengas que acceder en otra pestaña si es posible, a Youtube para poder ver una review o algún video no oficial sobre el videojuego y que esto no te permita seguir leyendo información sobre el videojuego. Además de tener que buscar por otra parte si en Spotify esta la BSO de dicho juego o no.

El objetivo de esta aplicación es poder acceder a una búsqueda completa de los videojuegos que te interesen y conocer todos sus detalles, noticias y temas relacionados con ellos. También se pretende que a la vez que estas leyendo la información sobre el videojuego, puedas estar escuchando o viendo una review o algún gameplay si dicho juego ya ha salido a la venta.

Otra opción posible que te permitiría esta aplicación es escuchar la banda sonora de dicho videojuego y si esta está en Spotify, poder añadirla a alguna lista de reproducción que ya tengas o crear una nueva para que siempre la tengas ahí.

## Aplicaciones integradas

La API de RAWG es la que nos permitirá acceder a una gran base de datos de videojuegos con toda la información necesaria sobre ellos, como por ejemplo, donde comprarlo con link directo a la tienda, noticias, nombre de los desarrolladores, publishers etc.

La API de Youtube será la que nos permitirá acceder a videos sobre el videojuego que busquemos o a la banda sonora de este.

La API de Spotify nos permitirá buscar las bandas sonoras de dichos videojuegos y si estas están en Spotify, podremos añadirlas a listas de reproducción ya existentes que tengamos o crear listas de reproducción para poder añadirlas.

|  |  |
| --- | --- |
| Nombre aplicación | URL documentación API |
| RAWG | <https://api.rawg.io/docs/> |
| Youtube | <https://developers.google.com/youtube/v3/docs?hl=es> |
| Spotify | <https://developer.spotify.com/documentation/web-api/> |

Tabla 1. Aplicación integradas

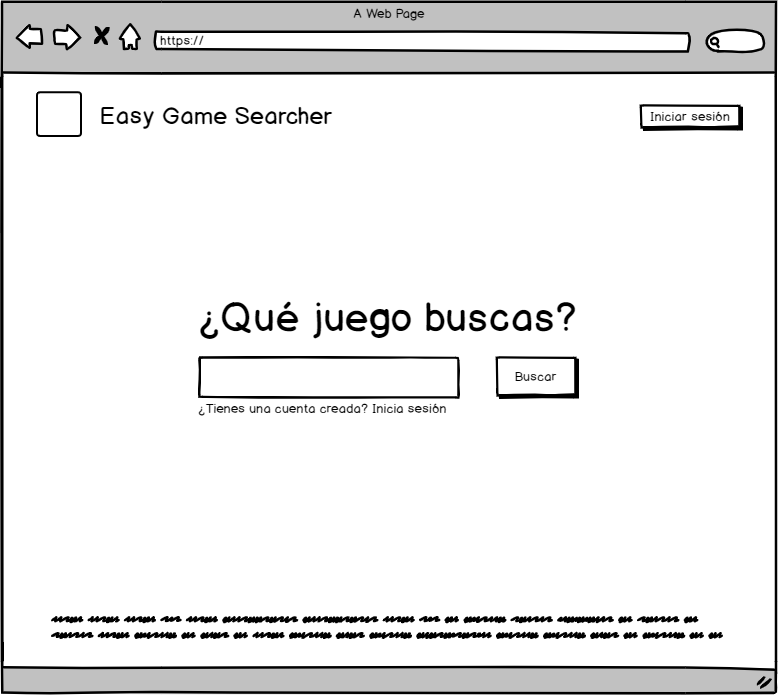
## Evolución del proyecto

1ª Entrega - Hemos comenzado con la implementación de una API bastante completa sobre videojuegos a la cual hemos decidido ampliar con acceso a Youtube y Spotify para poder acceder a campos que no se pueden acceder con la API principal.

# Prototipos de interfaz de usuario

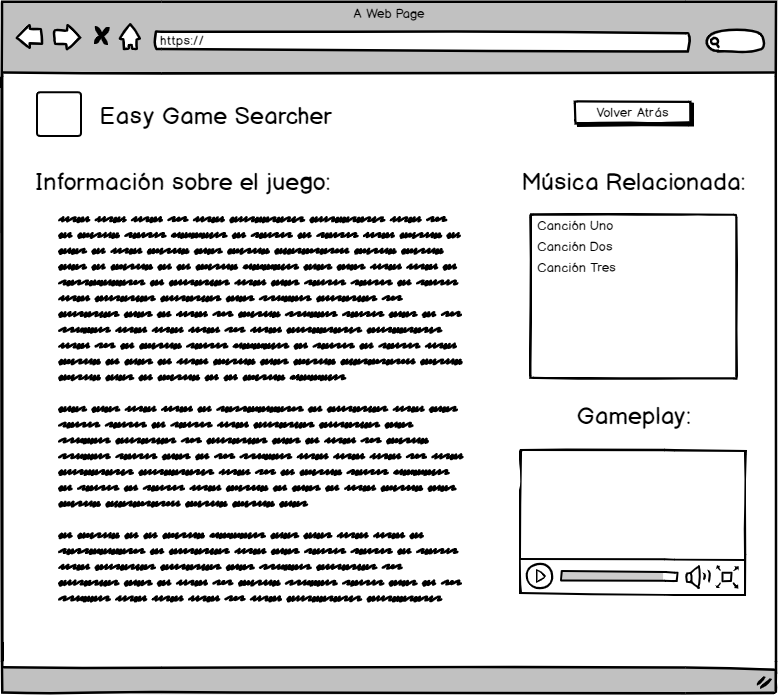
## Vista 1 – Login y buscador

En esta vista, se podrá hacer login y será la primera vista donde se accederá. Contiene un buscador, donde se pondrá el título del juego y te llevará a la segunda vista.



## Vista 2 – Vista Principal

En esta vista, podremos ver toda la información obtenida de las Api’s. Estadísticas relevantes sobre el juego, música relacionada o banda sonora si está disponible, y algún “gameplay” del mismo. También se incluirá un botón para volver a la vista anterior y poder buscar otro juego distinto.



# Arquitectura

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

## Diagrama de componentes

Captura de pantalla de un celular

Descripción generada automáticamente

Diagrama UML de componentes de alto nivel. Debe incluir las aplicaciones integradas y nuestra propia aplicación como un componente independiente.

## Diagrama de despliegue

Captura de pantalla de un celular con letras

Descripción generada automáticamente

Diagrama UML de despliegue de la aplicación.

## Diagrama de secuencia de alto nivel

Captura de pantalla de un celular con letras

Descripción generada automáticamente

Diagrama UML de secuencia indicando el flujo de mensajes entre las distintas aplicaciones integradas.

## Diagrama de clases

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

## 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 Spotify usando servicios RESTful. |
| Entrada | Se hace uso de la librería XXX para invocar al servicio usando la URI YYY 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 |

# Manual de usuario

## Mashup

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

## API REST

Incluir enlace a 1) fichero YAML de la API en OAS (Swagger) y 2) documentación interactiva generada a partir de la especificación OAS.

# 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.