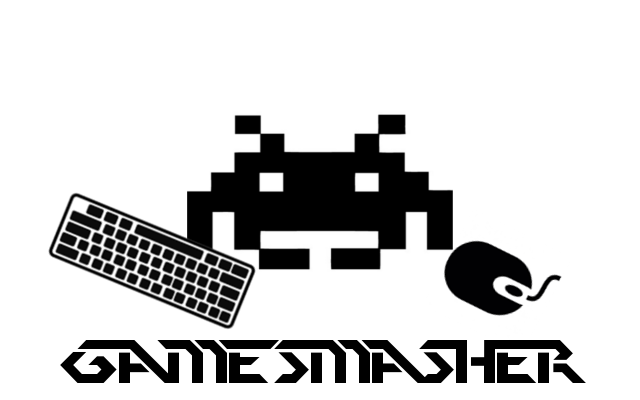
**GameSmasher**



Architecture and Integration of Software Systems

Degree in Software Engineering

2nd Course

Antonio Manuel Montaño Aguilera (antoniomma97@gmail.com)

Fernando Ruiz Robles (fernandoruizrobles@gmail.com)

Pablo Vázquez Zambrano (pabvazzam@gmail.com)

Pablo Martínez Figueroa (pablomf@hotmail.es)

Tutor: Adela del Río Ortega

Group name: ADR-GRUPO ING-VideoGameSmasher

App link: vgsmasher-aiss.appspot.com

Project link on projETSII, GitHub or similar: https://github.com/Silverath/VideoGameMasher

Versions history

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Details | Participants |
| 21/03/2017 | 1.0 | - It includes introduction, prototypes of user interface and UML diagrams of components, deployment and sequence (high level). | Antonio Montaño  Pablo Martínez  Fernando Ruiz  Pablo Vázquez |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Index

[1 Introduction 4](#_Toc478315668)

[1.1 APIs integrated 4](#_Toc478315669)

[1.2 Project evolution 5](#_Toc478315670)

[2 Prototypes of the user interface 5](#_Toc478315671)

[2.1 Main Page 5](#_Toc478315672)

[2.2 Videogame Library Main Page 6](#_Toc478315673)

[2.3 Game view in the library 7](#_Toc478315674)

[2.4 League of Legends and Overwatch statistics 8](#_Toc478315675)

[2.4.1 Summoner's and battletag search 8](#_Toc478315676)

[2.4.2 Statistics summary 9](#_Toc478315677)

[2.4.3 Played Champions (League of Legends)/Heroes (Overwatch) 10](#_Toc478315678)

[3 Architecture 11](#_Toc478315679)

[3.1 Component diagram 11](#_Toc478315680)

[3.2 Deployment diagram 11](#_Toc478315681)

[3.3 High level sequence diagram 12](#_Toc478315682)

[3.4 Diagrama de clases 12](#_Toc478315683)

[3.5 Diagramas de secuencia 12](#_Toc478315684)

[4 Implementación 12](#_Toc478315685)

[5 Pruebas 13](#_Toc478315686)

[6 Manual de usuario 14](#_Toc478315687)

[6.1 Mashup 14](#_Toc478315688)

[6.2 API REST 14](#_Toc478315689)

[Referencias 15](#_Toc478315690)

# Introduction

As usual players, sometimes we have the problem that we can’t obtain enough information of our personal profile in the videogame. For that, we want to create our application, with the aim of obtaining that information in a simple way, because other pages are so complex and difficult to understand due to the quantity of information shown in that page. The mashup consists in an application which gives the option to consult statistics of a game worldwide known: League of Legends.

## APIs integrated

|  |  |
| --- | --- |
| App name | URL API documentation |
| Facebook | https://developers.facebook.com/docs/graph-api/using-graph-api/ |
| League of Legends | https://developer.riotgames.com/api-methods/ |
| Twitch | https://dev.twitch.tv/docs |
| Google Drive | https://developers.google.com/drive/ |

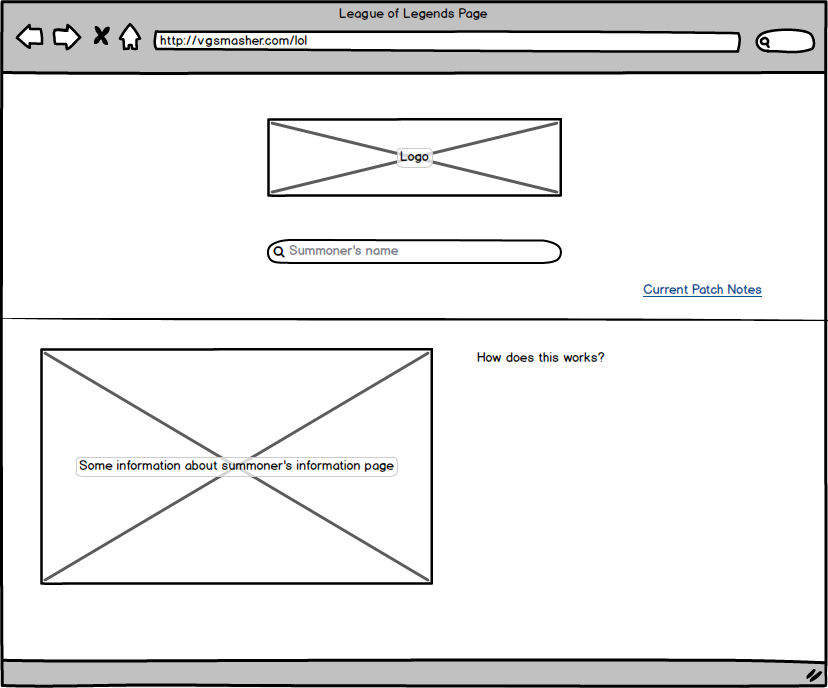
Table 1. APIs integrated

## Project evolution

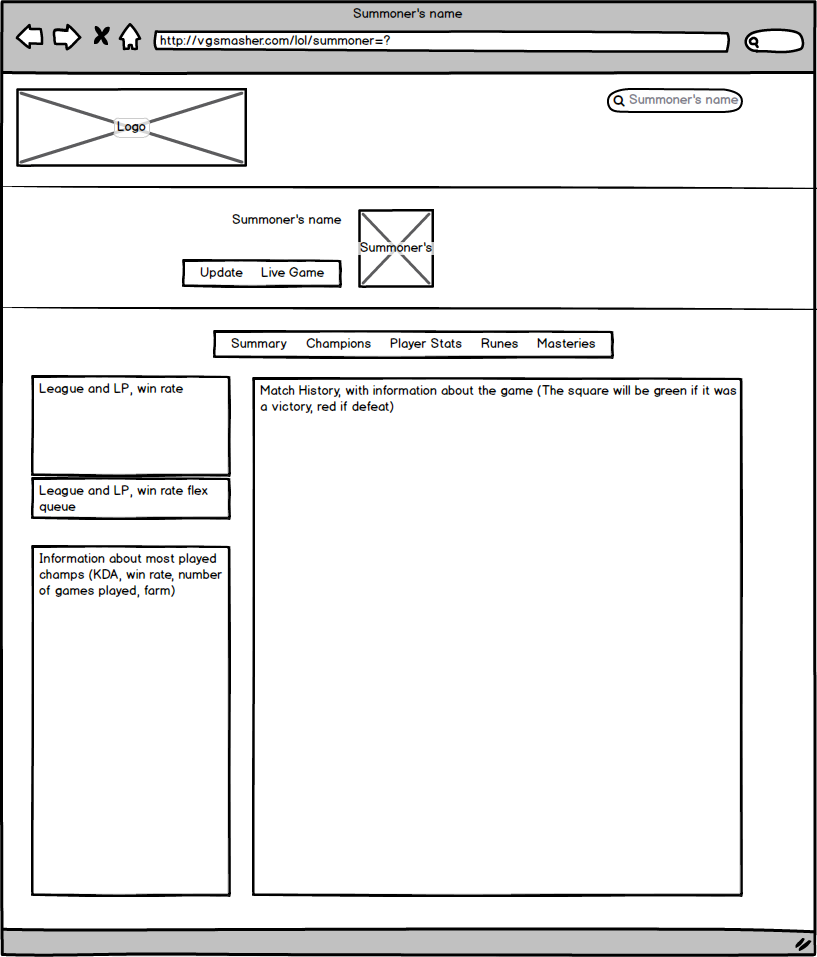
We have reconsidered the mashup and, after few discussions we have reached the conclusion that the mashup that we were planning to do is too big and we reduced it to just the stats page.

# Prototypes of the user interface

### Summoner's search

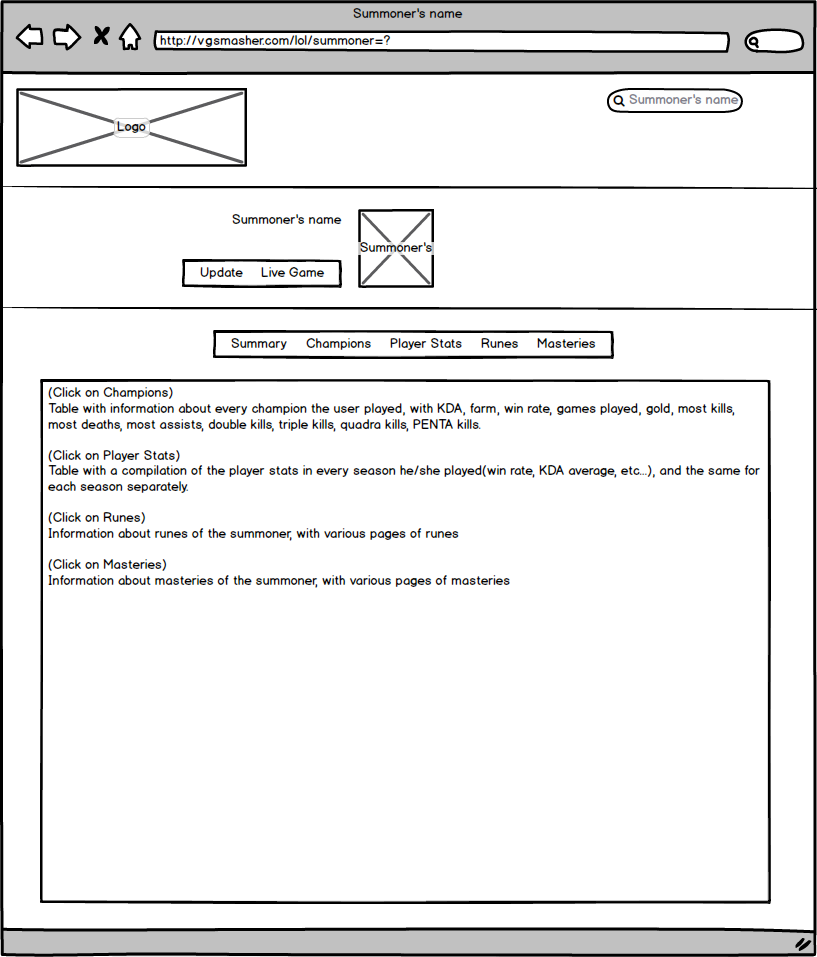
This one consists on a search box where the user types in his/her account name to find their statistics.

### Statistics summary



This view will show the general statistics of the user career in League of Legends/Overwatch.

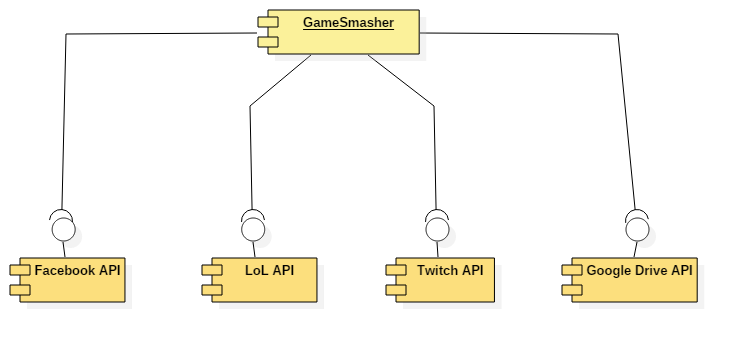
### Played Champions (League of Legends)/Heroes (Overwatch)



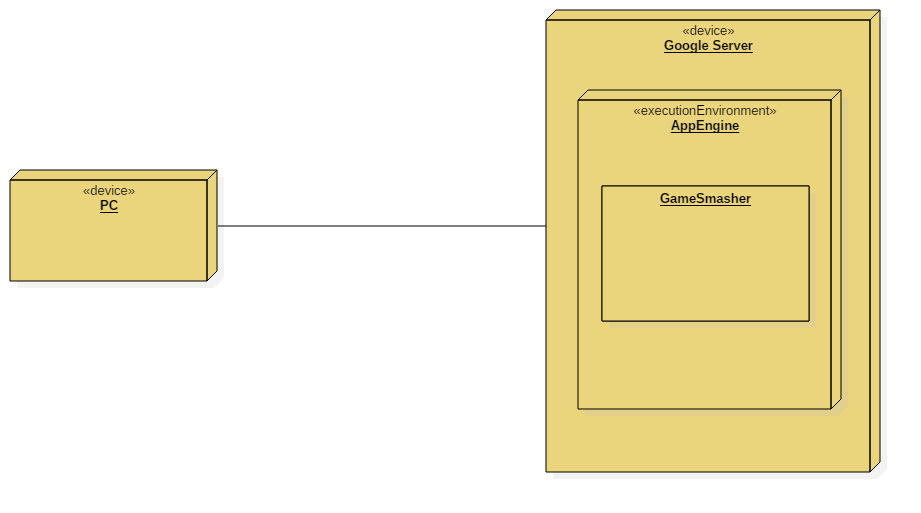
Depending on which tab you select, it will appear the information that is already written on the prototype.

# Architecture

## Component diagram

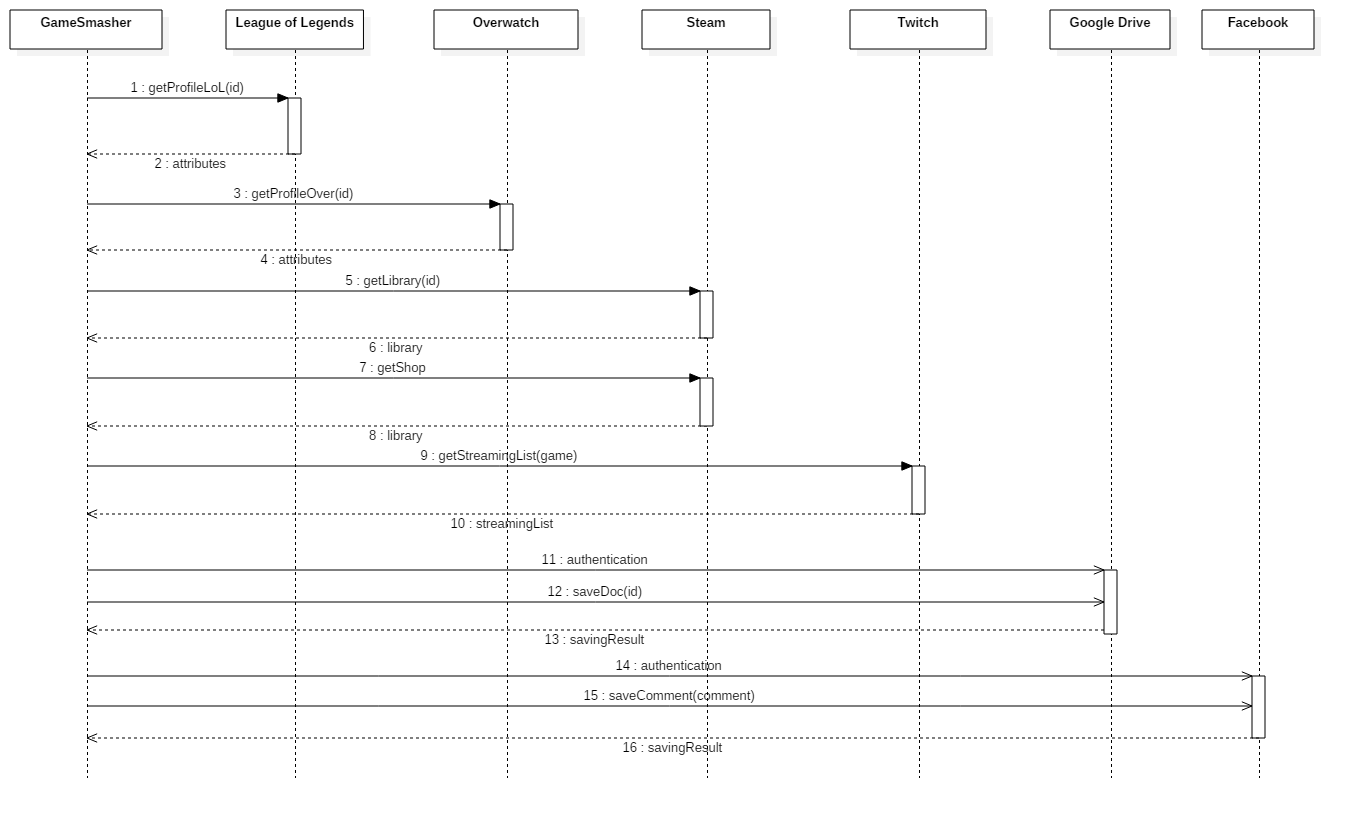


## Deployment diagram



## 

## High level sequence diagram



## 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

Indique la documentación de la API REST (contrato) implementada [2]. Cómo mínimo, la API debería incluir:

1. Protocolo de aplicación empleado por el servicio.
2. URIs para invocar a las operaciones del servicio.
3. Formato empleado para las representaciones de los recursos.
4. Códigos de estado empleados por el servicio.
5. Ejemplos de uso.

Está información también debe facilitarse en formato HTML como parte de la aplicación.

# Referencias

[1] *Balsamiq*. <http://balsamiq.com/>. Accedido en Marzo 2017.

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