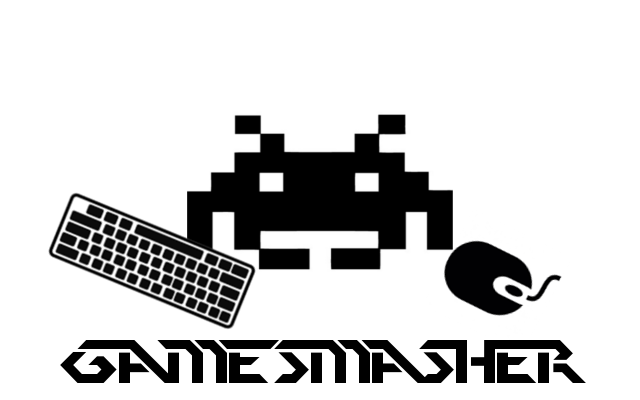
**GameSmasher**



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 number:

App link: vgsmasher-aiss.appspot.com

Project link en projETSII, GitHub o similar: <Enlace proyecto> (opcional)

Historial de versiones

|  |  |  |  |
| --- | --- | --- | --- |
| Fecha | Versión | Detalles | Participantes |
| 21/03/2017 | 1.0 | - Incluye introducción, prototipos de las interfaces de usuario y diagramas UML de componentes y despliegue. | Antonio Montaño  Pablo Martínez  Fernando Ruiz  Pablo Vázquez |
|  |  | <Mencionar los cambios más significativos con respecto a la versión anterior> |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Índice

[1 Introducción 5](#_Toc471899224)

[1.1 Aplicaciones integradas 5](#_Toc471899225)

[1.2 Evolución del proyecto 5](#_Toc471899226)

[2 Prototipos de interfaz de usuario 6](#_Toc471899227)

[2.1 Vista X 6](#_Toc471899228)

[2.2 Vista Y 6](#_Toc471899229)

[3 Arquitectura 7](#_Toc471899230)

[3.1 Diagrama de componentes 7](#_Toc471899231)

[3.2 Diagrama de despliegue 7](#_Toc471899232)

[3.3 Diagrama de secuencia de alto nivel 7](#_Toc471899233)

[3.4 Diagrama de clases 7](#_Toc471899234)

[3.5 Diagramas de secuencia 7](#_Toc471899235)

[4 Implementación 8](#_Toc471899236)

[5 Pruebas 9](#_Toc471899237)

[6 Manual de usuario 10](#_Toc471899238)

[6.1 Mashup 10](#_Toc471899239)

[6.2 API REST 10](#_Toc471899240)

[Referencias 11](#_Toc471899241)

# Introducción

Como jugadores habituales, en ocasiones tenemos el problema de que no podemos obtener la información sobre nuestra cuenta personal en el videojuego. Para ello tenemos intención de crear esta página con el objetivo de obtener esta información de manera sencilla ya que muchas otras páginas son complejas y difíciles de entender debido a la cantidad de información mostrada en la página.

El mashup consiste en una aplicación que da la opción de consultar las estadísticas en dos juegos conocidos mundialmente (Overwatch y League of Legends) y, con objetivo de hacer la aplicación más completa, hemos decidido añadir una biblioteca de videojuegos en la cual consultar cualquier videojuego que el usuario esté buscando. Esta biblioteca, además de contener información sobre el videojuego que se está buscando, proporcionará enlaces que redirijan al usuario a distintas páginas para obtener información extra sobre el videojuego.

## Aplicaciones integradas

|  |  |
| --- | --- |
| Nombre aplicación | URL documentación API |
| Facebook | https://developers.facebook.com/docs/graph-api/using-graph-api/ |
| League of Legends | https://developer.riotgames.com/api-methods/ |
| Overwatch\* | https://api.lootbox.eu/documentation |
| Twitch | https://dev.twitch.tv/docs |
| Wikia(?)\*\* | http://www.wikia.com/api/v1/#!/Wikis |
| Steam | https://developer.valvesoftware.com/wiki/Steam\_Web\_API |
| Google Drive | https://developers.google.com/drive/ |

Tabla 1. Aplicación integradas

\*La API de Overwatch no es oficial, pero la hemos testeado y funciona a la perfección, además de proporcionarnos lo necesario para nuestra aplicación.

\*\*No es segura su implementación en la aplicación ya que tenemos la API de Steam que es muy completa.

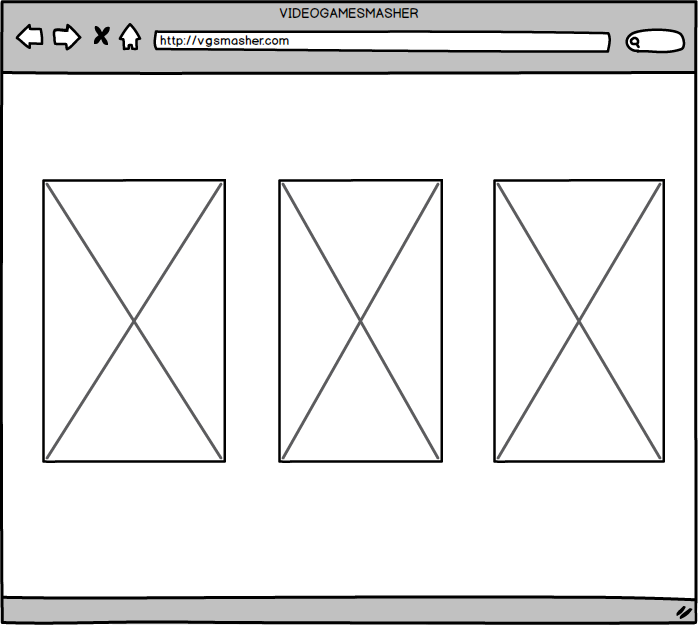
## Evolución del proyecto

Es habitual que la aplicación final diste mucho de la idea inicial. Puede que la idea fuese muy compleja, no haya sido posible integrar alguna de las aplicaciones o alguno de los miembros del grupo haya abandonado. Explicar en esta sección cuál ha sido la evolución del proyecto, problemas, cambios, decisiones, etc.

# Prototipos de interfaz de usuario

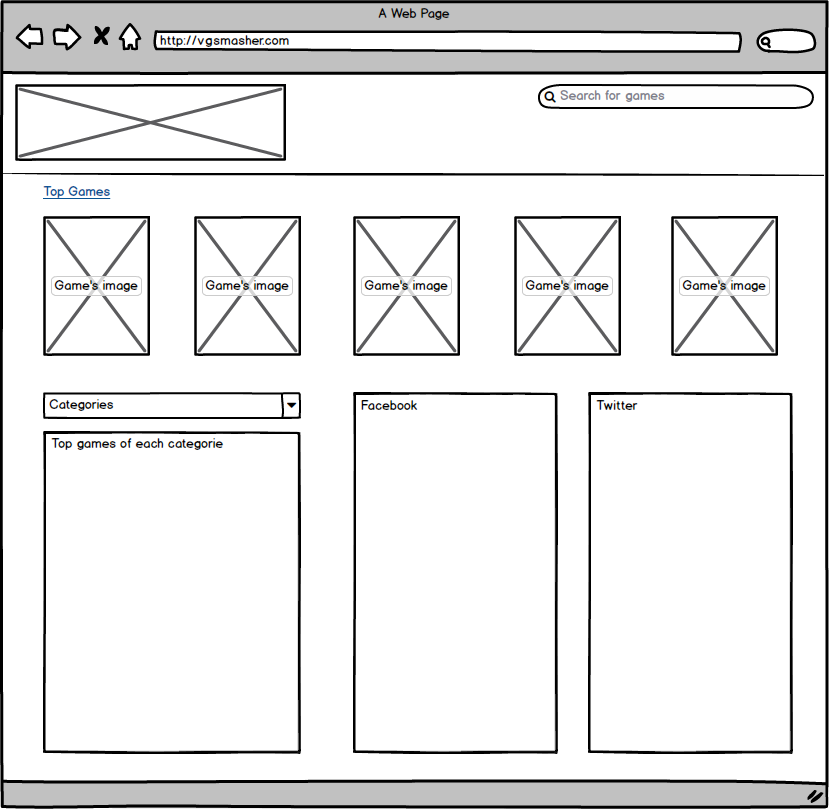
Insertar las imágenes de todos los prototipos desarrollados. Añadir para cada prototipo una breve descripción textual. Se recomienda hacer prototipos simples y realistas. Para los prototipos pueden usarse aplicaciones como Balsamiq [1]

1.MAIN PAGE:



En esta página se nos muestra tres imágenes diferentes, cada una redirigiendo a la página correspondiente(biblioteca de videojuegos , Overwatch o League of Legends).

2.PÁGINA PRINCIPAL DE LA BIBLIOTECA DE VIDEOJUEGOS:

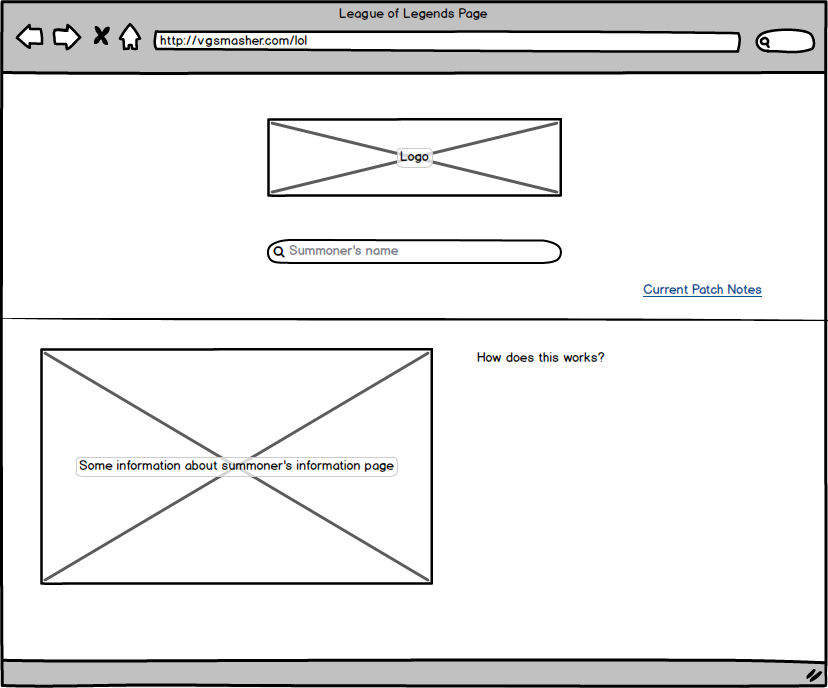


Es un menú principal donde podemos encontrar los videojuegos más populares, así como los más populares ordenados por categorías y menciones sobre dichos videojuegos tanto en Twitter como en Facebook.

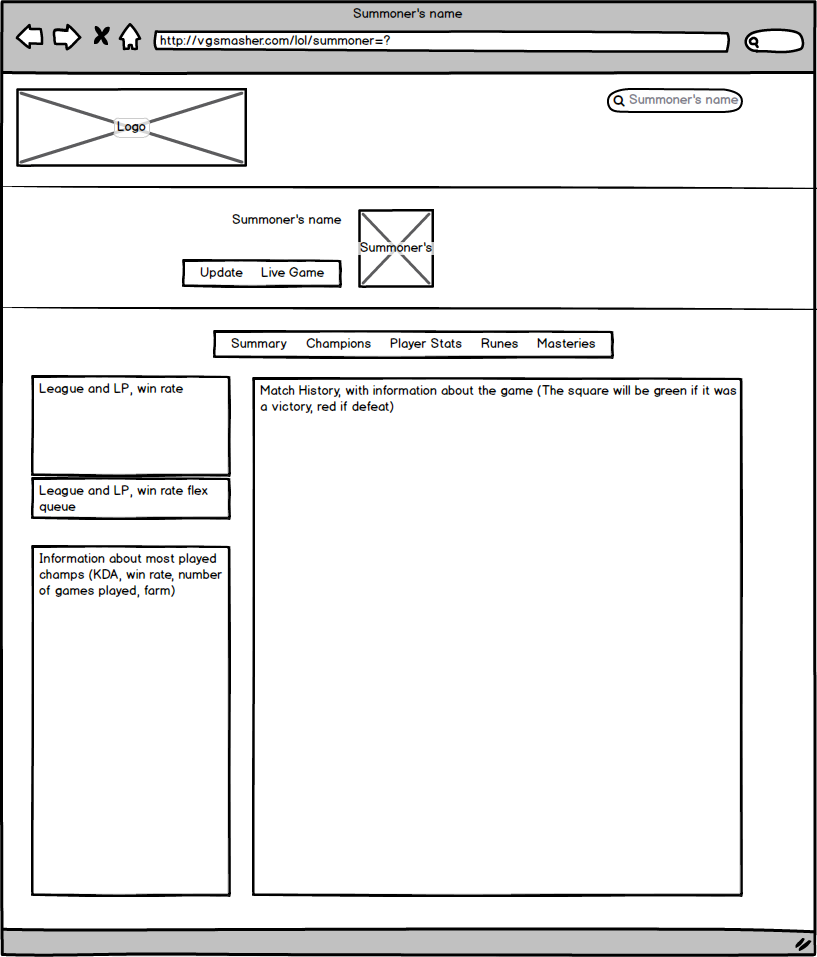
3. VISTA DE UN VIDEOJUEGO EN LA BIBLIOTECA: **(VISTA PENDIENTE DE HACER)**

4.VISTA DE LAS DISTNTAS ESTADÍSTICAS DE LEAGUE OF LEGENDS:

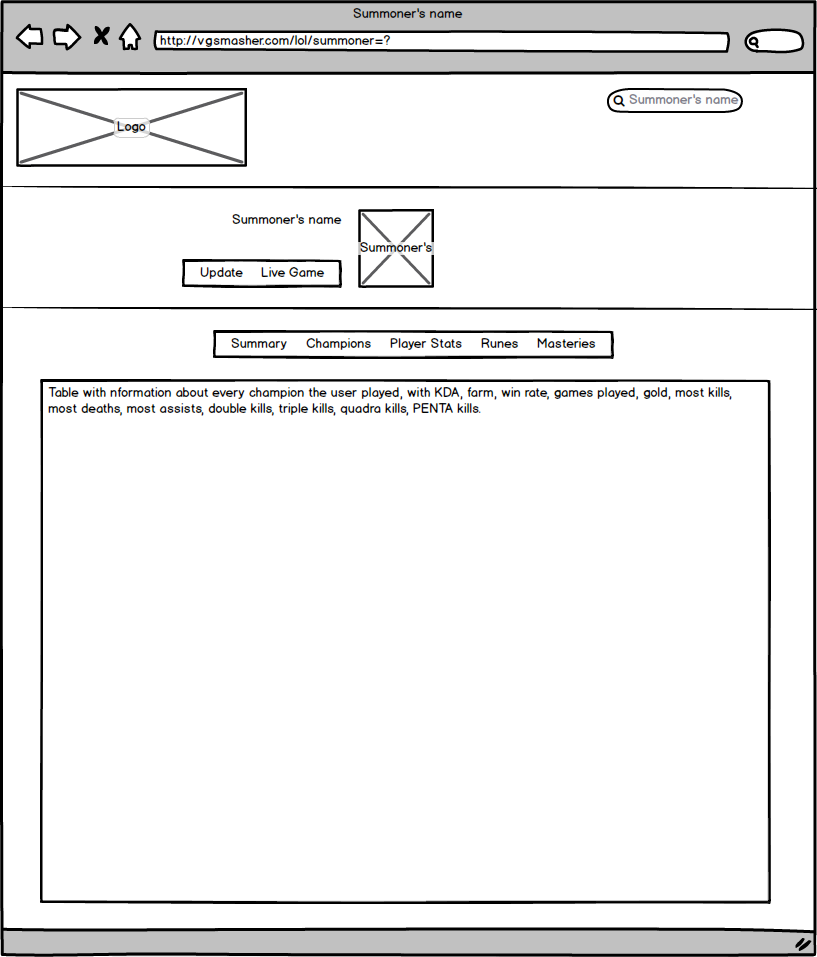
4.1. BUSCADOR DE INVOCADORES:



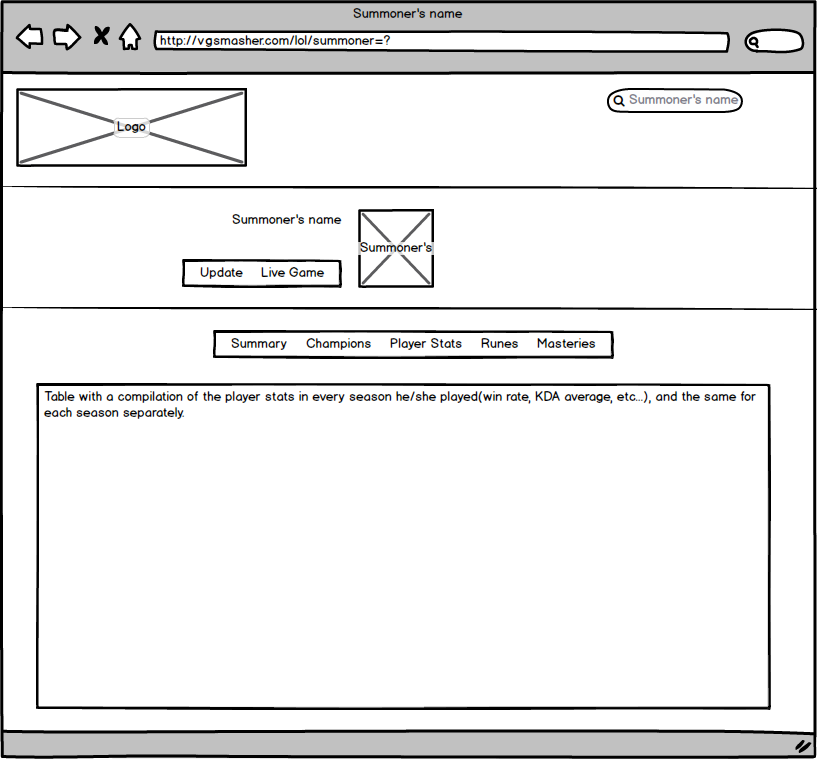
4.2. RESUMEN DE LAS ESTADÍSTICAS:



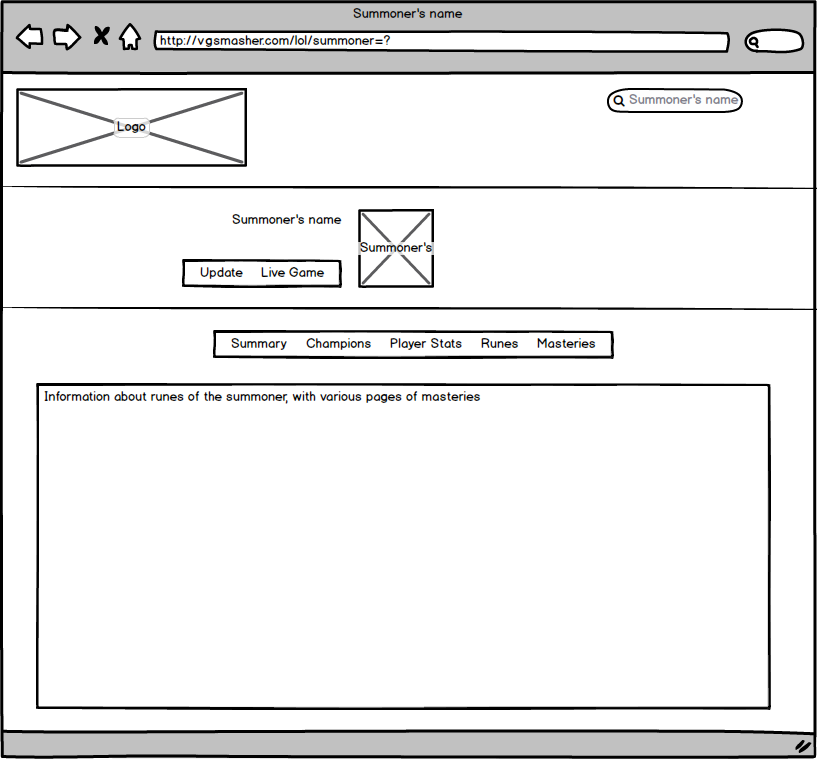
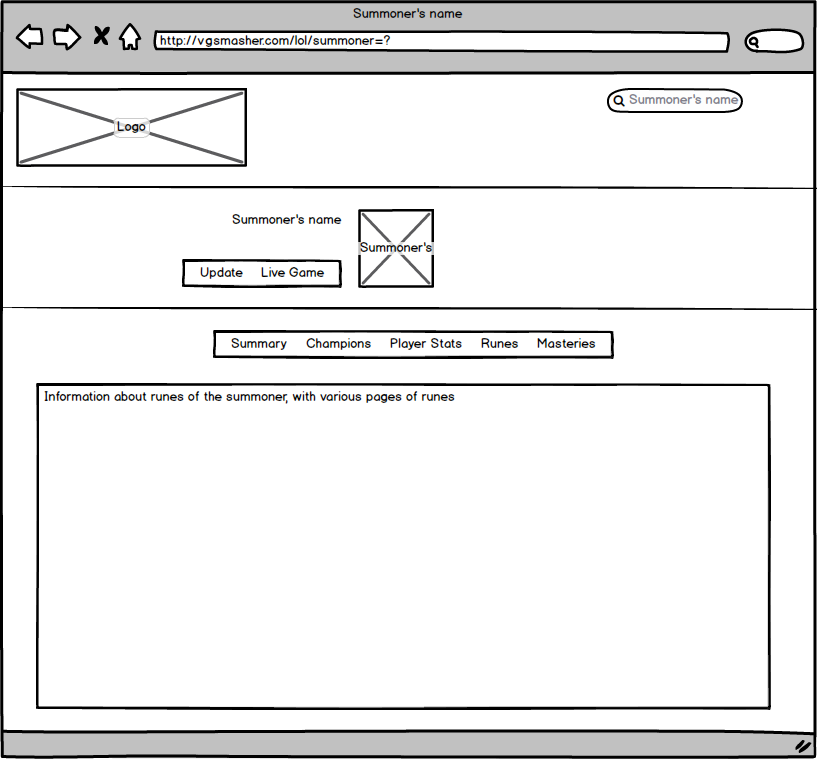
4.3. CAMPEONES JUGADOS:



4.4. ESTADÍSTICAS DE JUGADOR:



4.5. RUNAS Y MAESTRÍAS:



# Arquitectura

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

## Diagrama de componentes

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

## Diagrama de despliegue

Diagrama UML de despliegue de la aplicación.

## Diagrama de secuencia de alto nivel

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

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.