**Proyecto MARC**



Arquitectura e Integración de Sistemas Software

Grado de Ingeniería del Software

Curso 2º

Bueno Menéndez, Rubén

Jiménez Navarro, Manuel Jesús

Monteseirín Puig, Alejandro

Reyes Aroca, Camila

Tutor: Alfonso Eduardo Márquez

Número de grupo: MARC

Enlace de la aplicación: <http://proyecto-marc-aiss.appspot.com/>

Enlace del proyecto en GitHub: <https://github.com/ruben077/ProyectoAISS>

Historial de versiones

|  |  |  |  |
| --- | --- | --- | --- |
| Fecha | Versión | Detalles | Participantes |
| 14/03/2017 | 1.0 | - Incluye introducción, prototipos de las interfaces de usuario y diagramas UML de componentes y despliegue. | Rubén  Manuel  Camila  Alejandro |
| 30/03/2017 | 1.1 | - Incluye el prototipo funcional de la aplicación utilizando las distintas APIs expuestas en los diagramas. Además se proporciona la documentación de la API de la propia aplicación. | Rubén  Manuel  Camila  Alejandro |
| 20/05/2017 | 2.0 | - Implementación de los servicios RESTful, pruebas unitarias y desarrollo de la API propia, previamente documentada en la anterior versión, con nuevas funcionalidades. Se añade, además, documentación en Swagger. | Rubén  Manuel  Camila  Alejandro |

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 **¡Error! Marcador no definido.**](#_Toc471899228)

[2.2 Vista Y **¡Error! Marcador no definido.**](#_Toc471899229)

[3 Arquitectura 11](#_Toc471899230)

[3.1 Diagrama de componentes 11](#_Toc471899231)

[3.2 Diagrama de despliegue 11](#_Toc471899232)

[3.3 Diagrama de secuencia de alto nivel 12](#_Toc471899233)

[3.4 Diagrama de clases 12](#_Toc471899234)

[3.5 Diagramas de secuencia 13](#_Toc471899235)

[4 Implementación 16](#_Toc471899236)

[5 Pruebas 17](#_Toc471899237)

[6 Manual de usuario 20](#_Toc471899238)

[6.1 Mashup 20](#_Toc471899239)

[6.2 API REST 21](#_Toc471899240)

# Introducción

La cultura japonesa ha ganado especial relevancia en los últimos años, sobre todo en el sector del manga y la animación japonesa (anime). A pesar de este incremento de popularidad, aunque existen varias aplicaciones que muestran informaciones varias sobre dichos sectores, existen muy pocas que reúnan dichas informaciones en una sola aplicación, lo cual ahorra mucho tiempo de navegación al usuario que no solo busca imágenes o vídeos, por ejemplo, sobre un anime.

Por ello queremos desarrollar una aplicación híbrida que solvente este problema, integrando varias APIs que ofrecen imágenes, vídeos, merchandising e información variada sobre un anime concreto, además de mantener un registro de los animes vistos o pendientes de ver.

## Aplicaciones integradas

Describir cada una de las aplicaciones integradas dando detalles sobre cada una de ellas

|  |  |
| --- | --- |
| Nombre aplicación | URL documentación API |
| MyAnimeList | https://myanimelist.net/modules.php?go=api |
| AnimeNewsNetwork | http://www.animenewsnetwork.com/encyclopedia/api.php |
| Youtube | https://www.youtube.com/yt/dev/es/api-resources.html |
| AliExpress | https://portals.aliexpress.com/help.htm?page=help\_center\_api |
| Tumblr | https://www.tumblr.com/docs/en/api/v2 |

Tabla 1. Aplicación integradas

## Evolución del proyecto

En un primer momento, nos planteamos integrar sólo 3 APIs (MyAnimeList, AnimeNewsNetwork y Tumblr). Las dos primeras trabajan con formato XML y no JSON, que es el visto en clase. Debido a ello nos planteamos no integrarlas ya que no conseguíamos hacer funcionar la aplicación. Sin embargo, tras mucha investigación conseguimos integrarlas junto con Tumblr.

En primer lugar, tomamos la decisión de integrar las APIs MyAnimeList y AnimeNewsNetwork obteniendo el contenido XML de los resultados obtenidos y pasándolo a formato de texto, de manera que después se dividiese mediante etiquetas las partes que queríamos obtener y las añadíamos a sus correspondientes atributos. De esta manera simulábamos el mismo resultado que queríamos obtener.

Como lo anterior no era del todo viable, con mucha más investigación conseguimos mediante la dependencia JAX-RS Client API (definida en el paquete javax.ws.rs.client) y combinando métodos, proporcionar una API de alto nivel para acceder a cualquier recurso REST. Aun así, para que no quedase en vano todo el trabajo anterior, ya que sólo íbamos a obtener resultados con AnimeNewsNetwork, decidimos dejar esta como ya lo teníamos y nos centramos en MyAnimeList.

También surgió el problema de que con MyAnimeList es obligatoria la autenticación para todo, por ello decidimos crear una clase independiente (en el paquete aiss.utility) que añadiese a la cabecera de la web la autenticación correspondiente y mediante otro método se comprobase si las credenciales son correctas o incorrectas. A la hora de implementar esto hemos decidido crear un formulario de autenticación.

Después decidimos integrar también Youtube y Aliexpress con el fin de suministrar aún más información sobre el anime buscado. Estas no supusieron mucha dificultad a la hora de integrarlas, ya que trabajan con formato JSON.

Para terminar, decidimos crear tests de pruebas unitarias JUnit que comprobase que funciona cada método implementado además de que también decidimos que toda la web tuviera un diseño responsivo que se adaptase a todos los dispositivos, es por ello que se ha trabajado también minuciosamente sobre los estilos CSS.

# Prototipos de interfaz de usuario

## Index

Vista principal donde se realizará la búsqueda del anime sobre el que queremos obtener información.

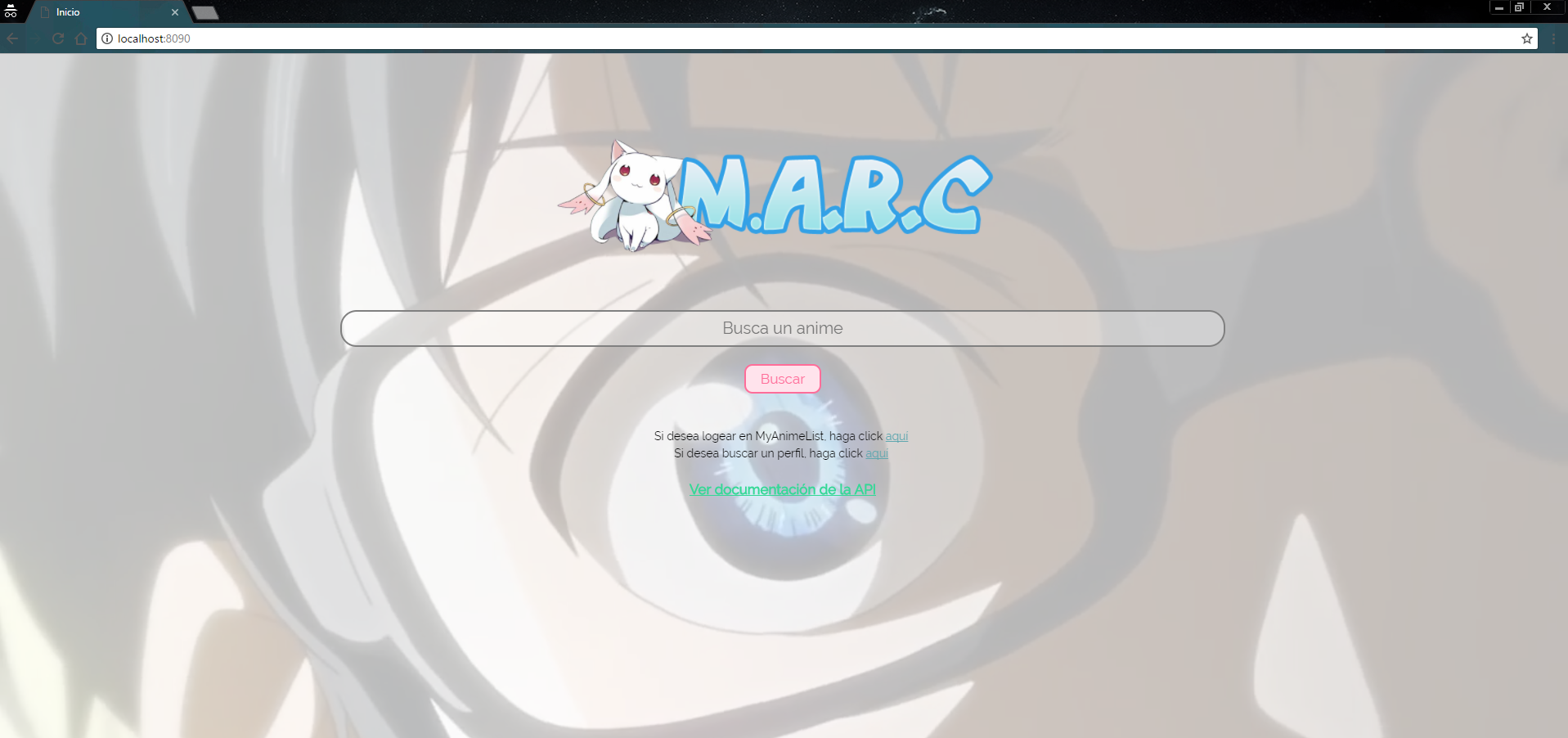
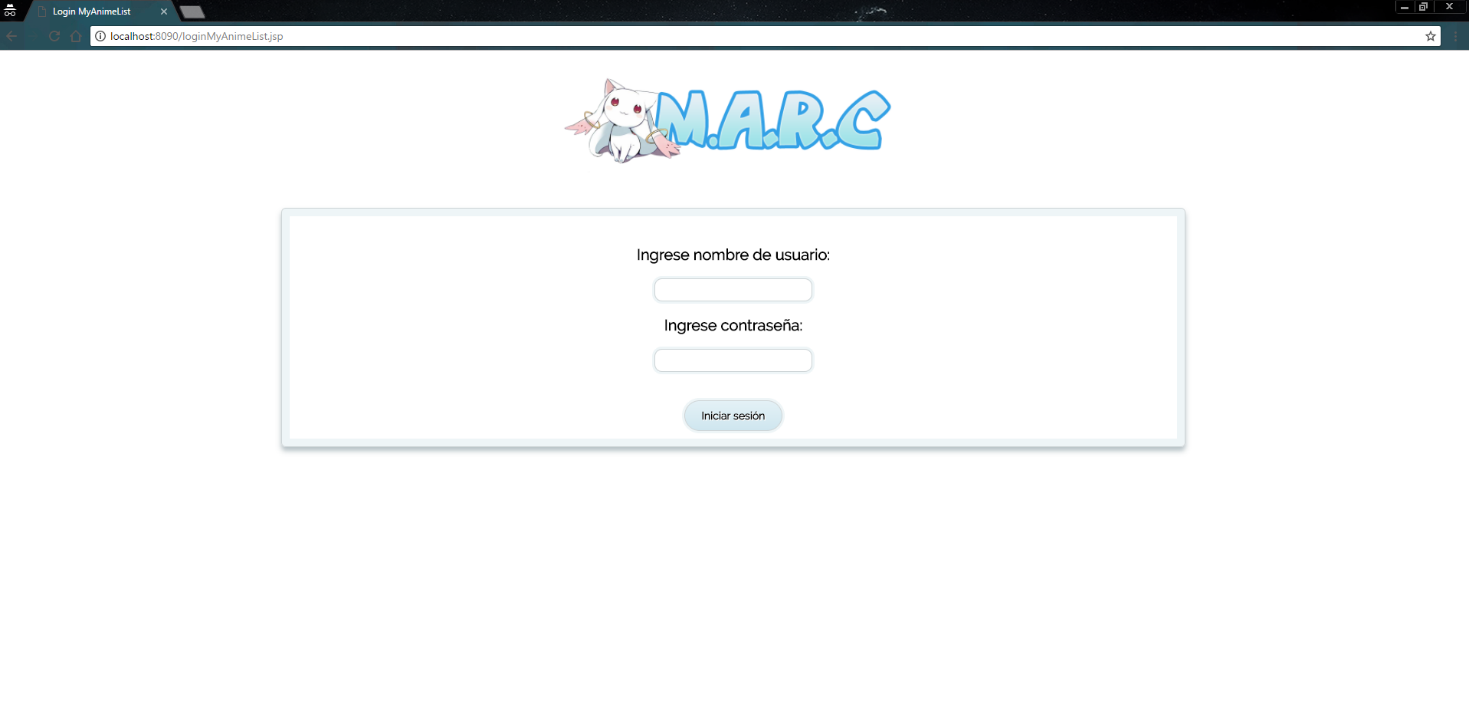


Figura 1. Prototipo de interfaz de usuario de la vista X

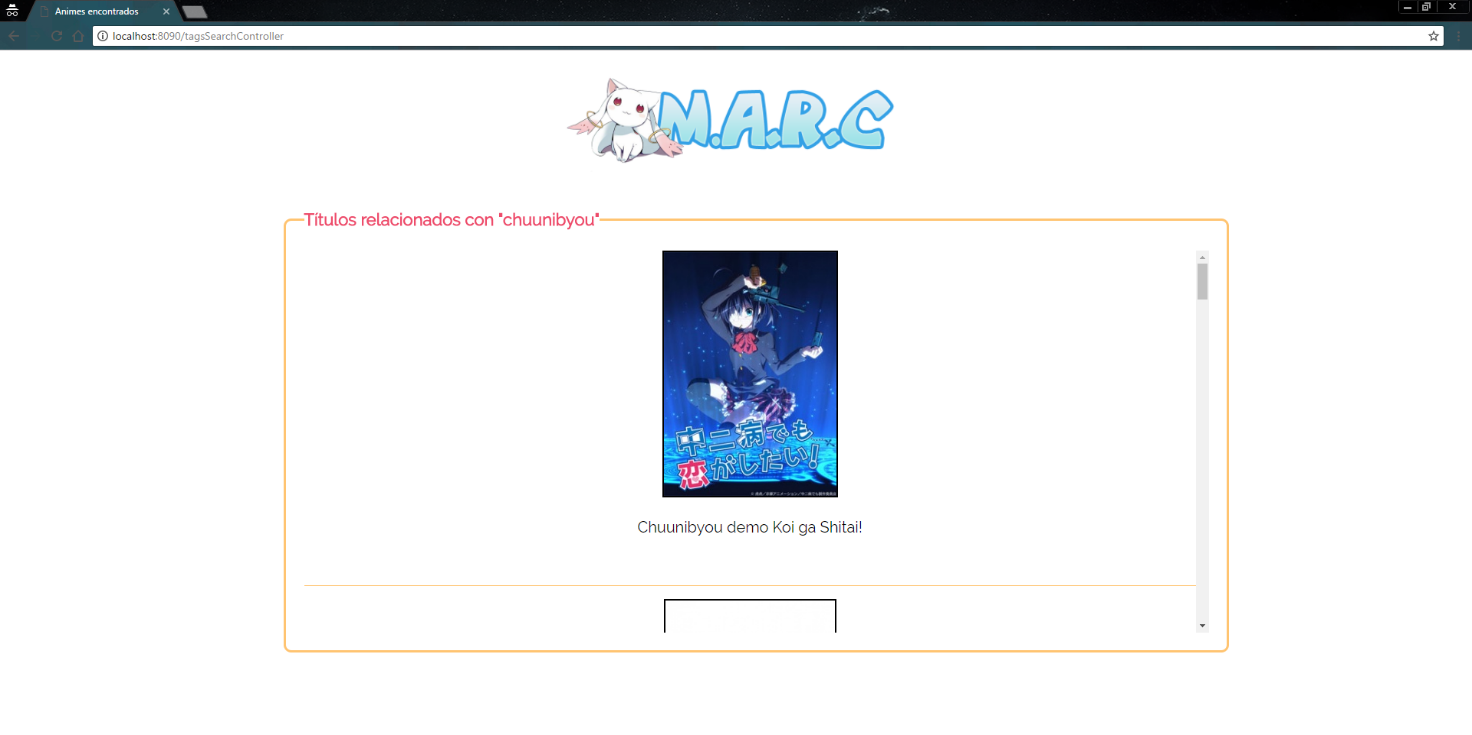
## loginMyAnimeList.jsp

Formulario para iniciar sesión en MyAnimeList.



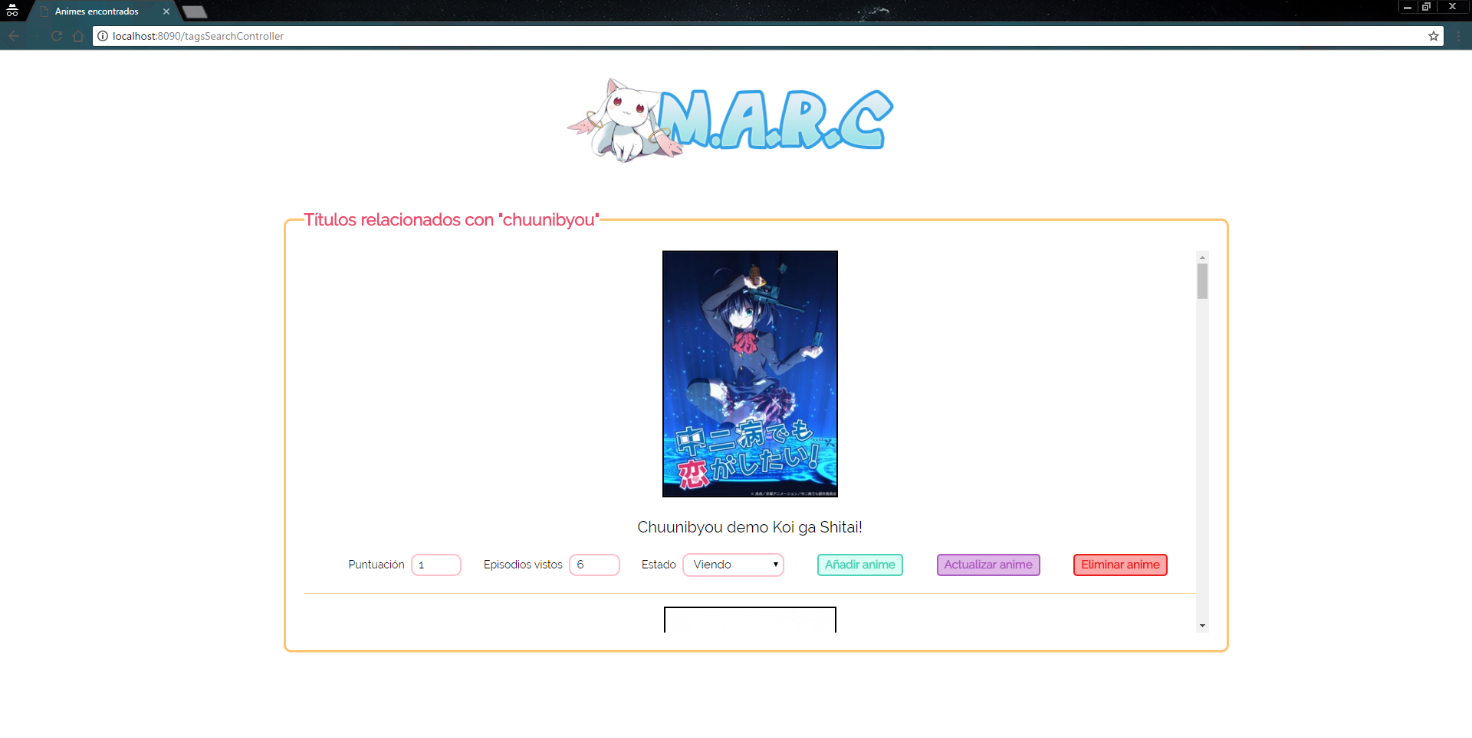
## listaAnimes.jsp (sin sesión iniciada)

Lista de los animes encontrados relacionados con la búsqueda.



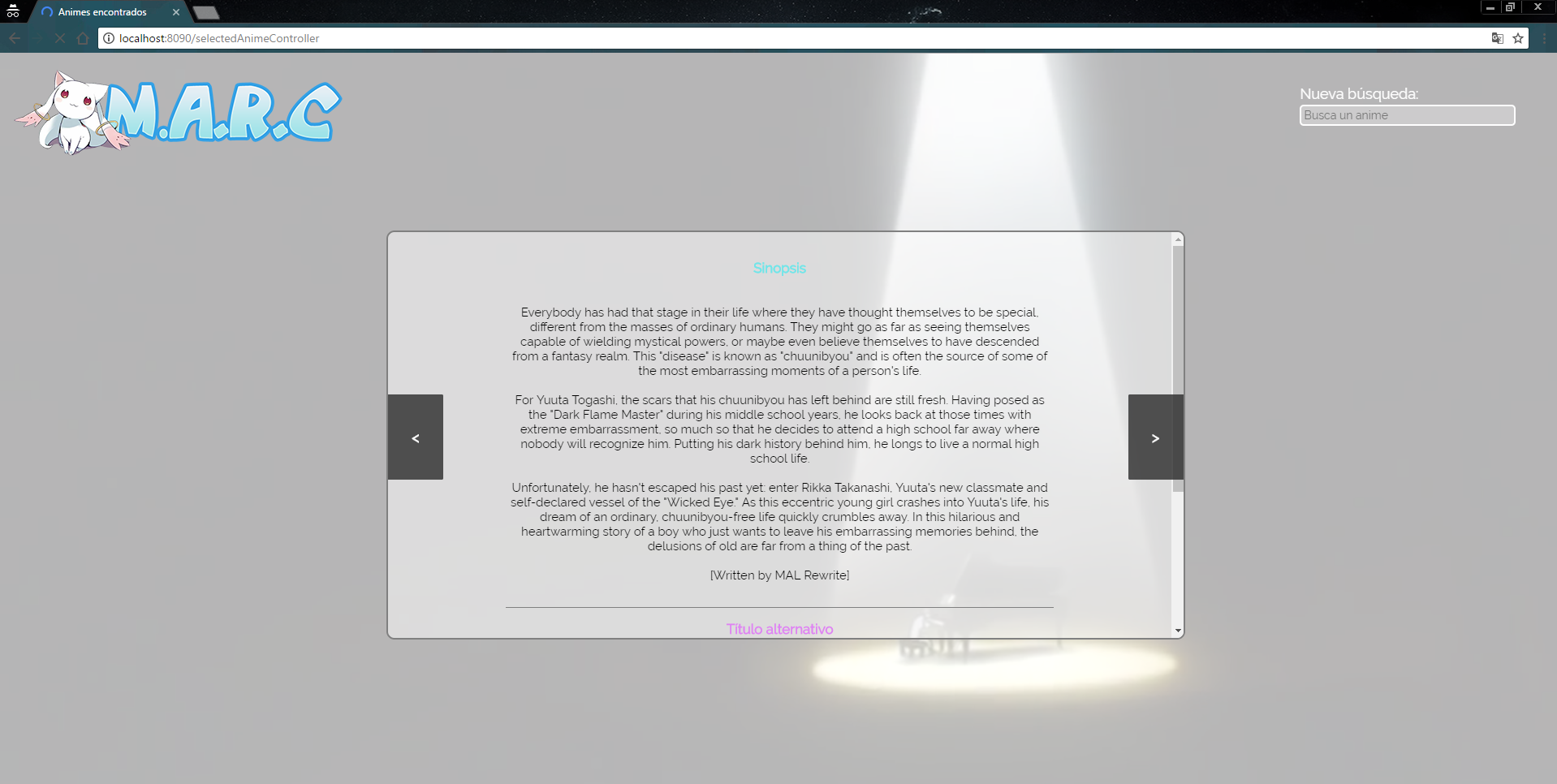
## listaAnimes.jsp (con sesión iniciada)

Lista de los animes encontrados relacionados con la búsqueda con posibles operaciones.



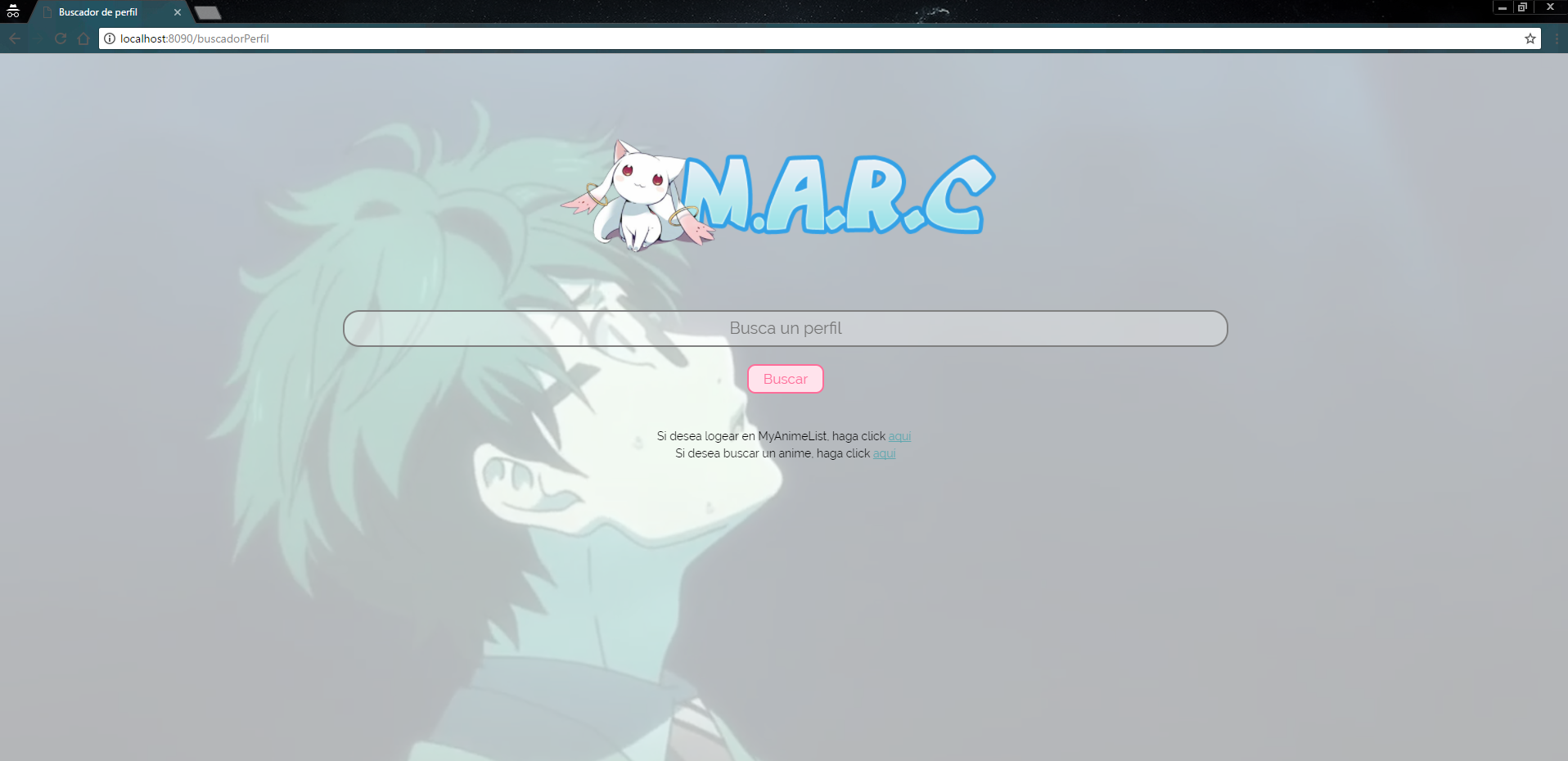
## resultados.jsp

Resultados del anime seleccionado en listaAnimes.jsp con su información, noticias, vídeos, imágenes y productos.



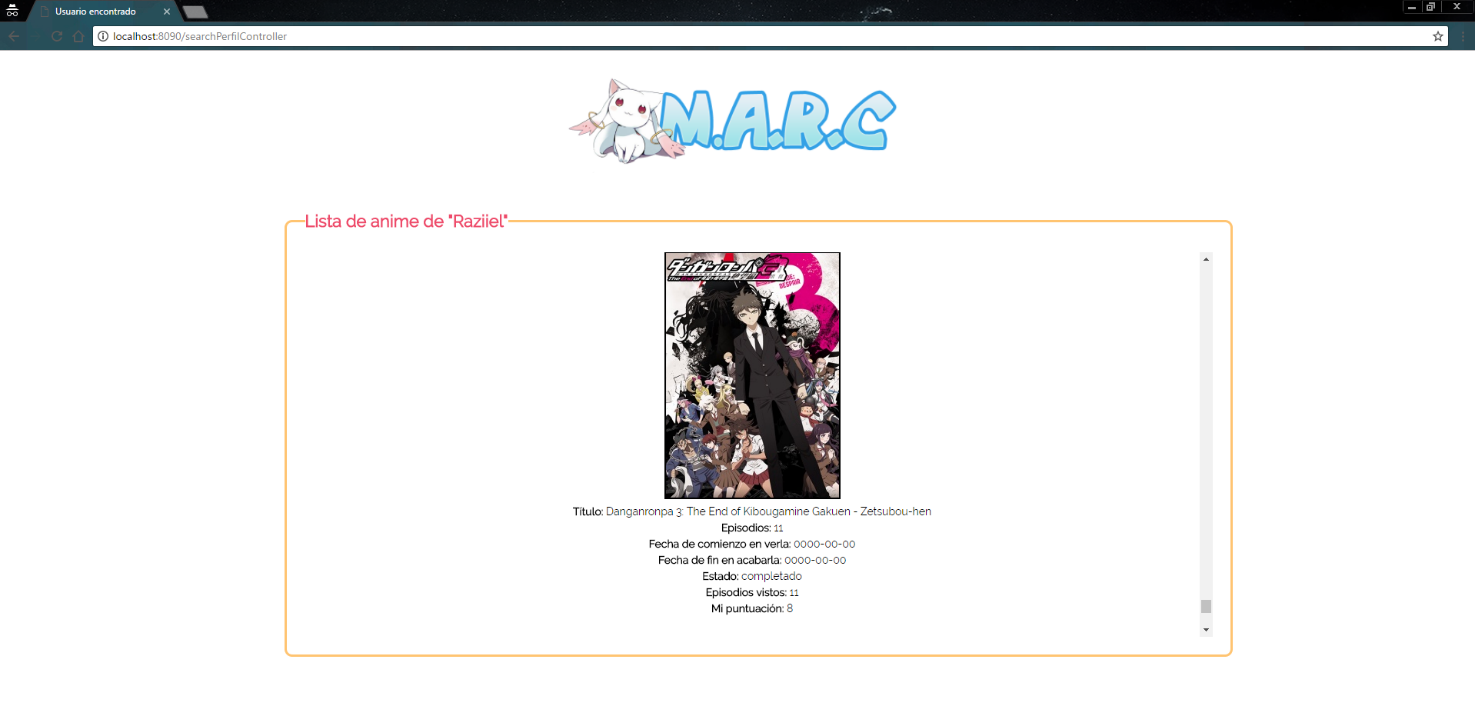
## BuscadorPerfil

Vista principal donde se realizará la búsqueda de un perfil registrado en MyAnimeList sobre el que queremos obtener información.



## perfil.jsp

Lista de los animes encontrados en el perfil relacionado con la búsqueda.



## error.jsp

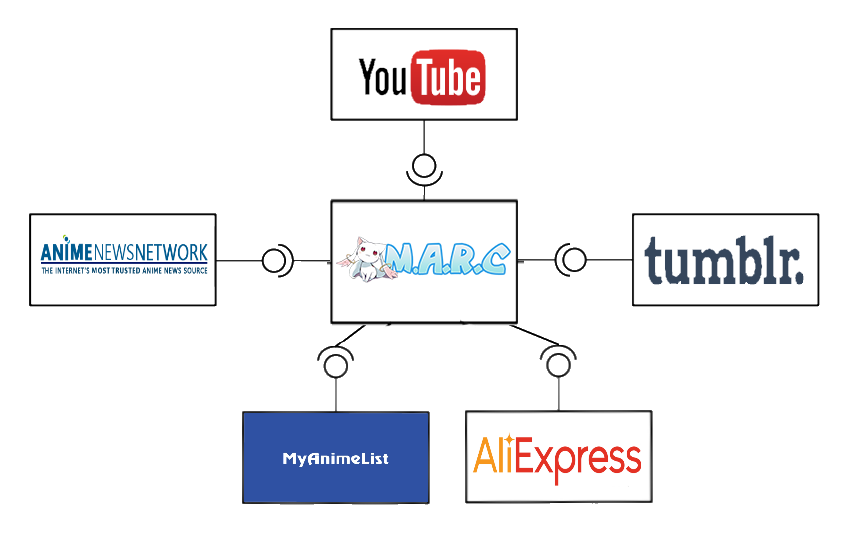
Página de error en el caso de que algo vaya mal en la web.



# Arquitectura

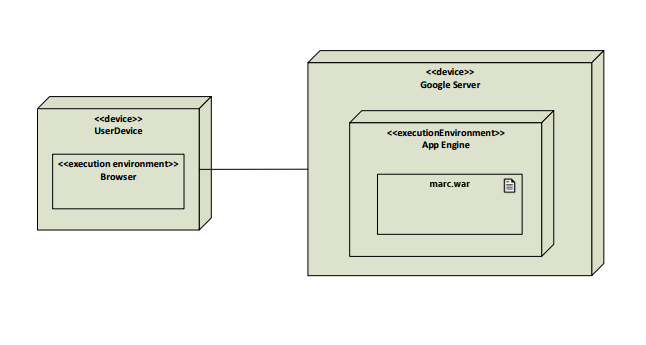
## Diagrama de componentes

En este diagrama se muestra los distintos componentes(Aplicaciones) que formarán a nuestra aplicación y que explotaremos a través de sus respectivas APIs.



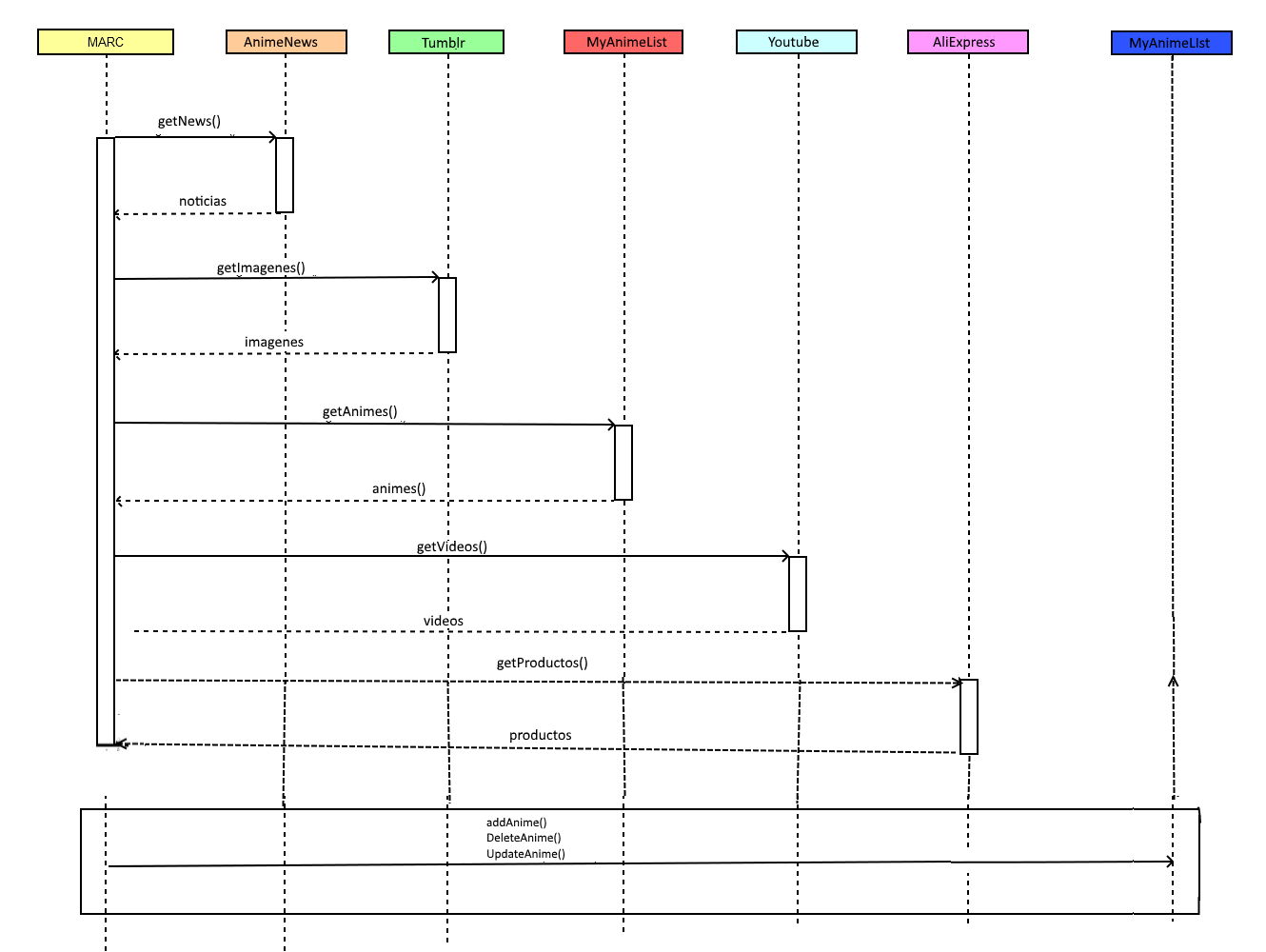
## Diagrama de despliegue

Diagrama que muestra como se ve desde fuera nuestra aplicación estando contenida en la plataforma appEngine que a su vez estará en un servidor de Google para que cualquier persona pueda utilizarla.



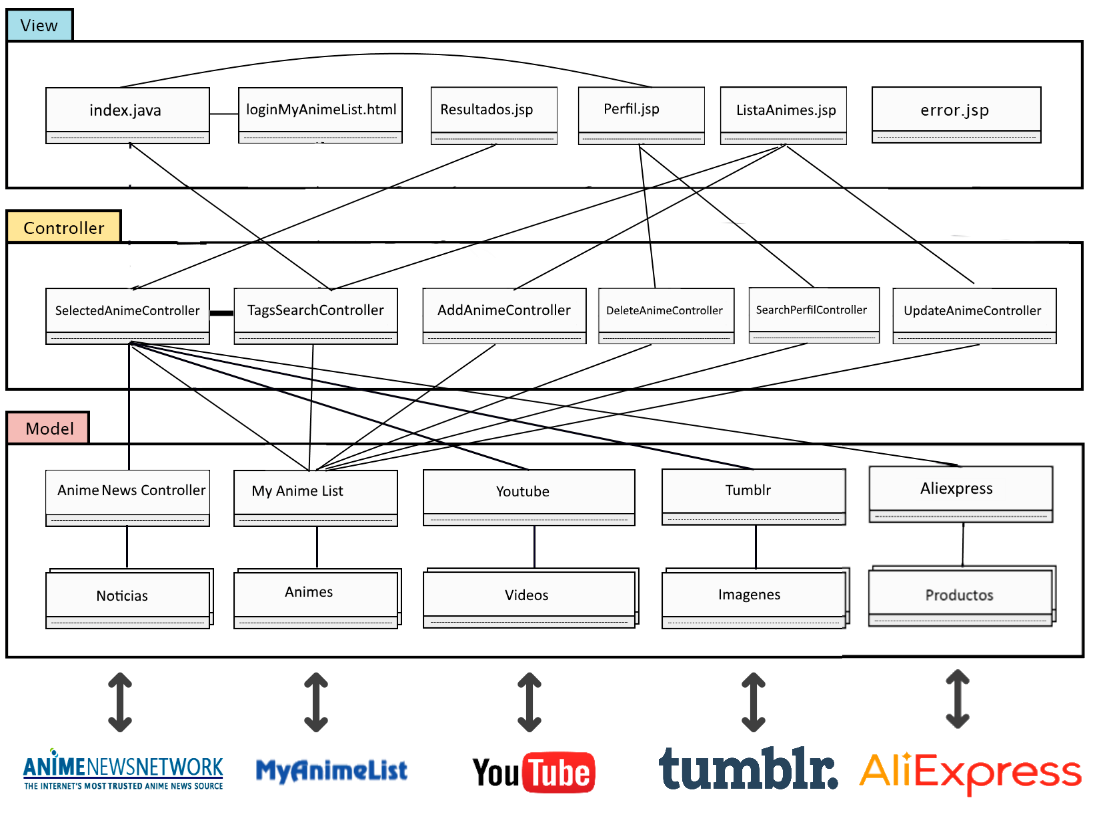
## Diagrama de secuencia de alto nivel

Muestra a grosso modo el funcionamiento de la aplicación.



## Diagrama de clases

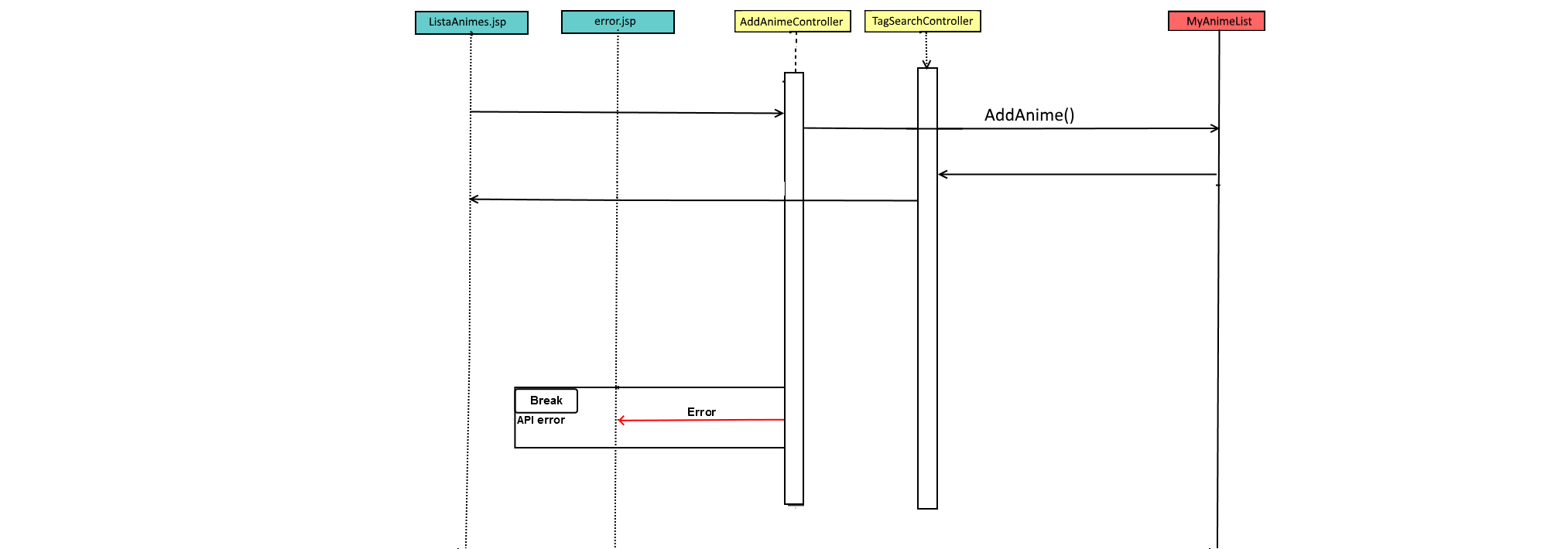
En este diagrama se muestra la estructura completa que tiene nuestra aplicación y las dependencias entre los distintos módulos de la aplicación dividido en niveles.



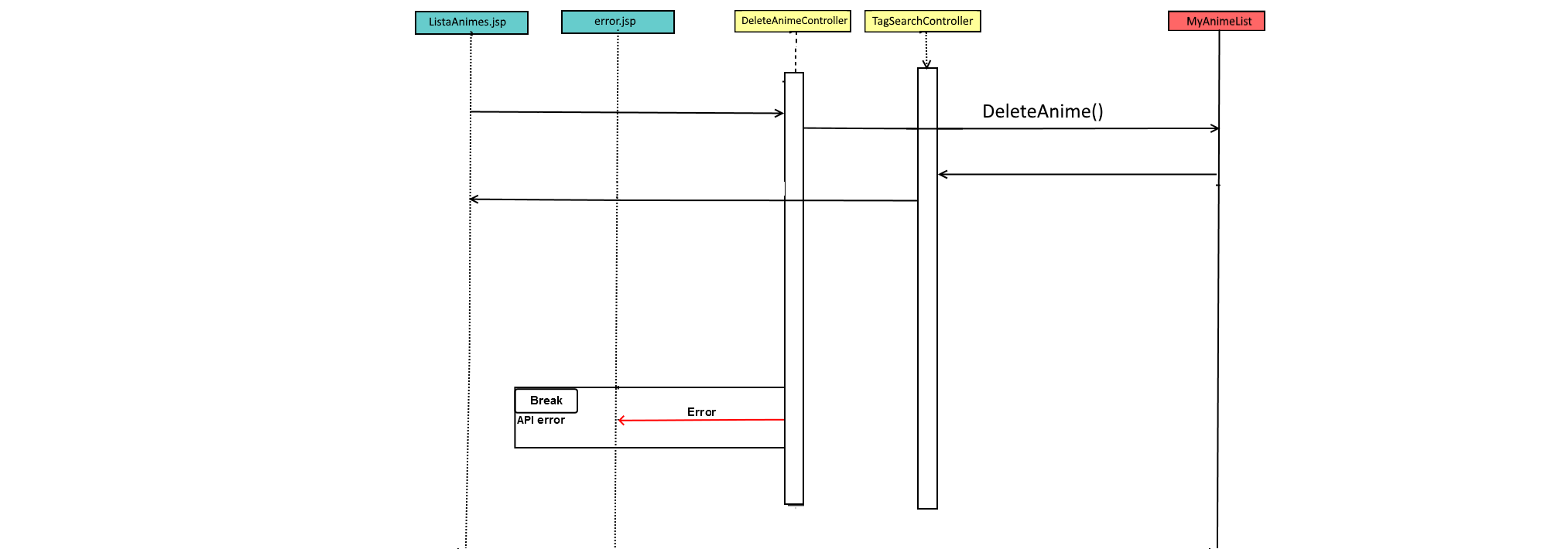
## Diagramas de secuencia de bajo nivel

Muestra en detalle la funcionalidad de nuestra API y cómo van a llevarse a cabo todas las funcionalidades.

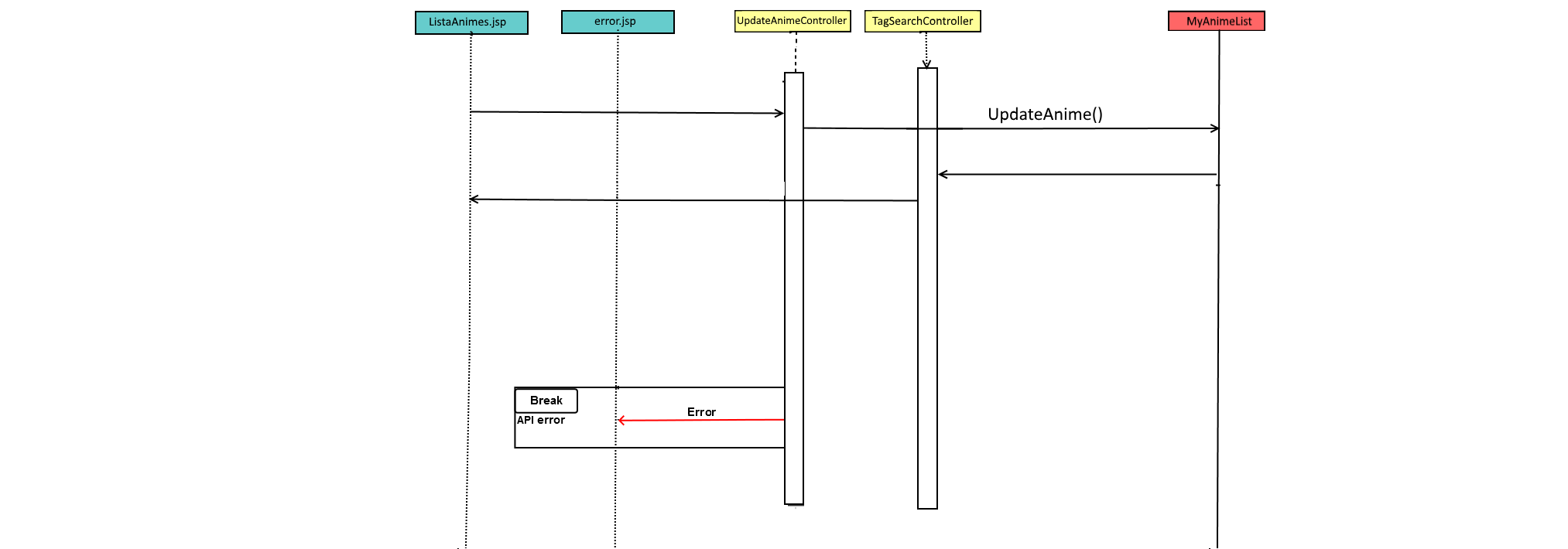
**Añadir anime**



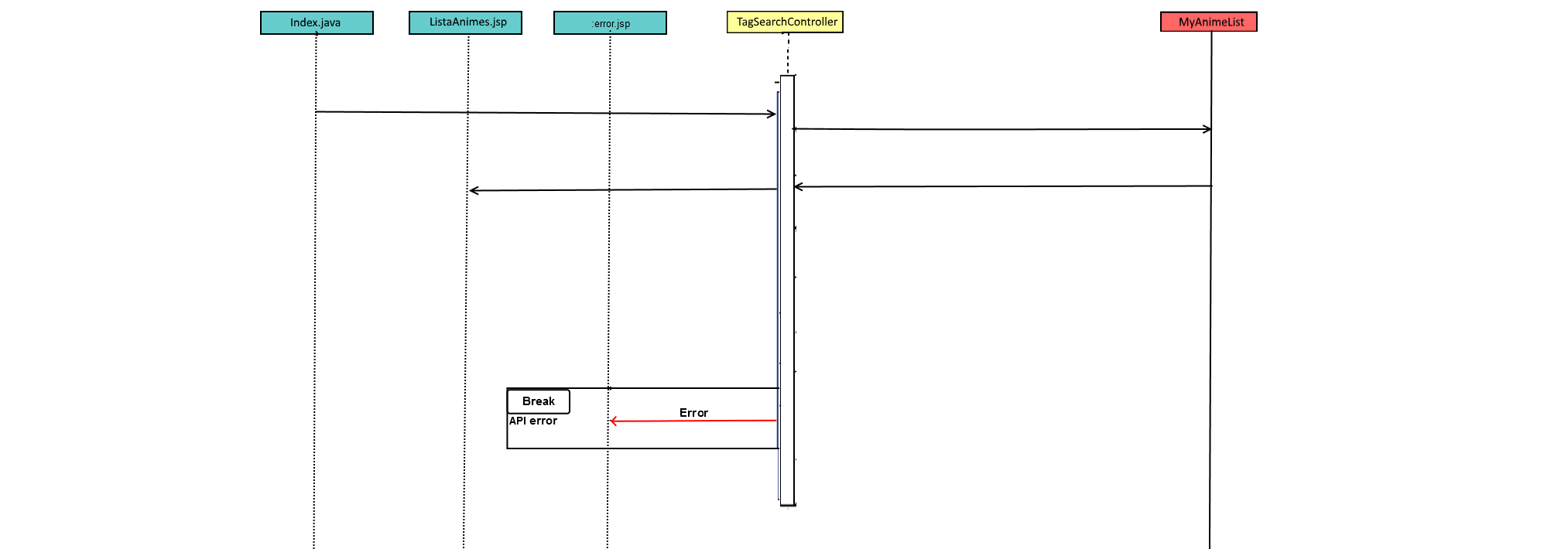
**Borrar anime**



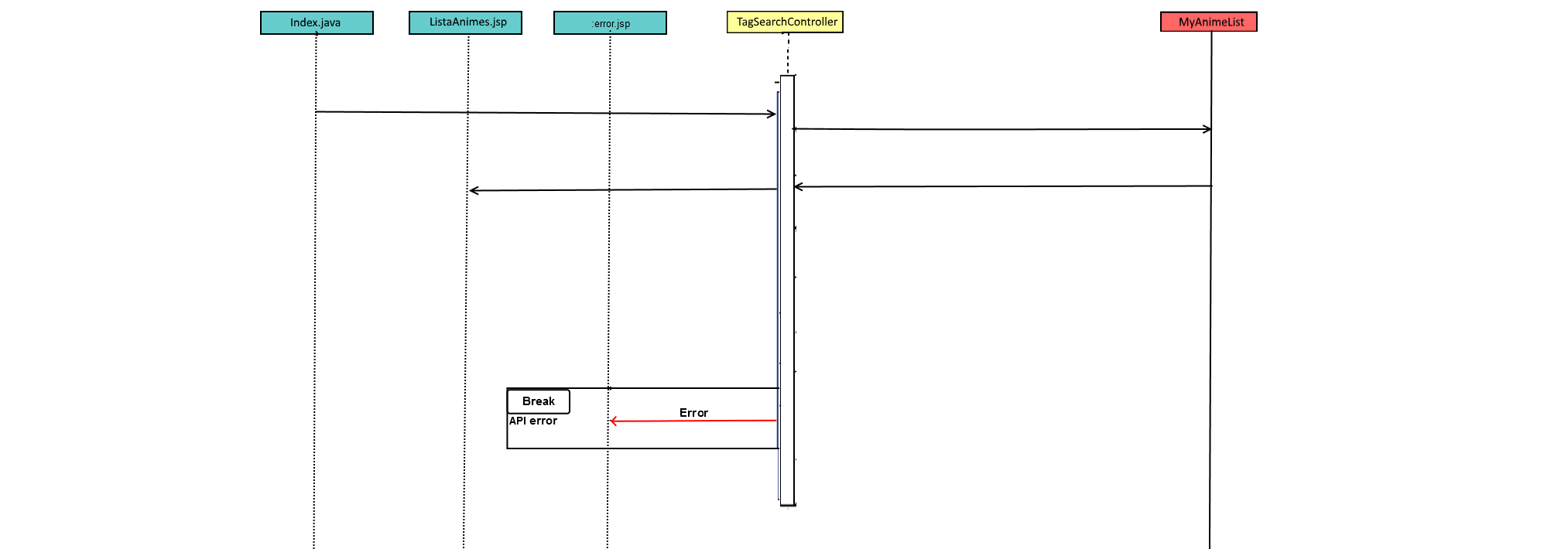
**Actualizar anime**



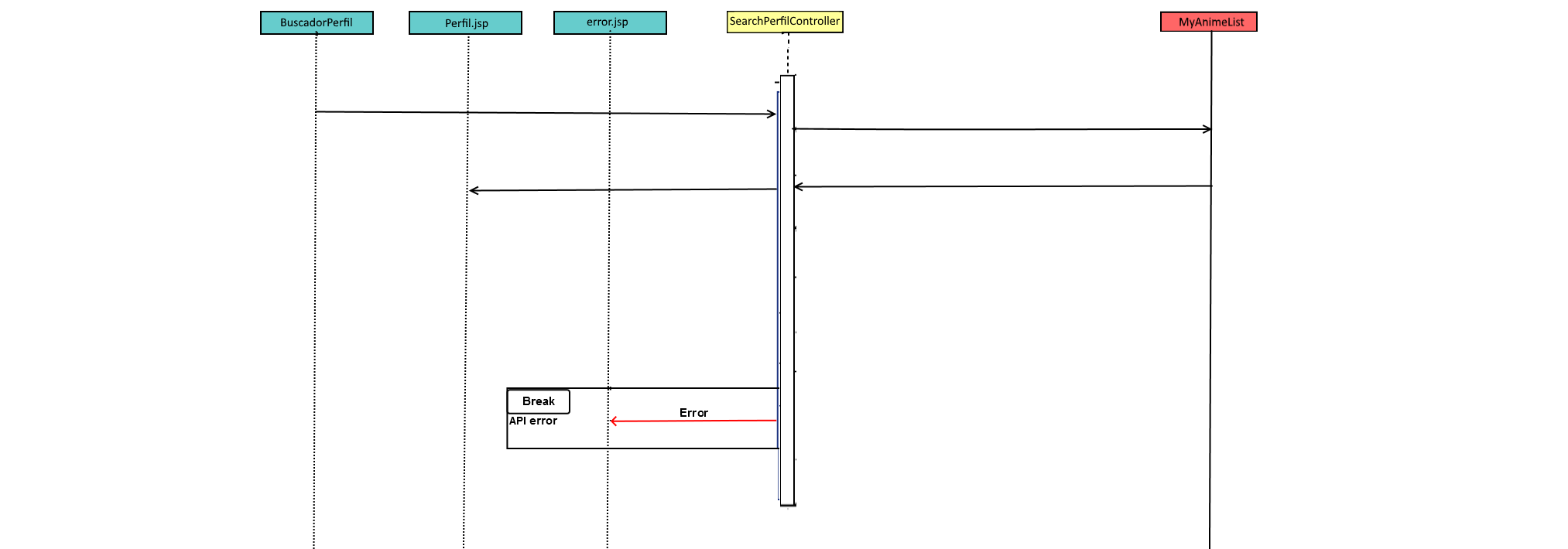
**Buscar anime**



**Seleccionar anime**



**Buscar Perfil**



# Implementación

La aplicación ofrece varias funciones en solo una búsqueda además de facilitar la utilización de cada una de las aplicaciones que se explotan en esta app. Además de eso el parseado de XML y la forma de autorización de la aplicación de MyAnimeList es un problema para aquellos que no conozcan las librerías como SAX que realizan esa conversión fácilmente, eso fue un punto de dificultad para nuestra aplicación además de la forma de estructuras la página de forma que sea vistosa y fácil de usar.

Además de ello completamos ciertas funcionalidades en las que por ejemplo la API de MyAnimeList no ofrecía y que tuvimos que utilizar medios como coger e código XML de un enlace de la página.

Por último, la dificultad de implementar las 5 APIs y además obtener la información de ellas de forma fácil y de forma que no resultara tedioso tener que buscar en cada página por separado es un gran punto a favor de esta aplicación.

# Pruebas

|  |  |
| --- | --- |
| Resumen |  |
| Número total de pruebas realizadas | 9 |
| Número de pruebas automatizadas | 9 (100%) |

|  |  |
| --- | --- |
| ID | **Prueba 1** |
| Descripción | Prueba para la detección de errores al implementar búsquedas en AnimeNewsNetworks usando servicios RESTful. |
| Salida esperada | Los datos devueltos en formato XML son mapeados a una clase Java y a continuación se muestra si ha habido éxito con JUnit. |
| Resultado | **EXITO** |
| Automatizada | Sí |

|  |  |
| --- | --- |
| ID | **Prueba 2** |
| Descripción | Prueba para la detección de errores al implementar búsquedas en Tumblr usando servicios RESTful. |
| Salida esperada | Los datos devueltos en formato JSON son mapeados a una clase Java y a continuación se muestra si ha habido éxito con JUnit. |
| Resultado | **EXITO** |
| Automatizada | Sí |

|  |  |
| --- | --- |
| ID | **Prueba 3** |
| Descripción | Prueba para la detección de errores al implementar búsquedas en YouTube usando servicios RESTful. |
| Salida esperada | Los datos devueltos en formato JSON son mapeados a una clase Java y a continuación se muestra si ha habido éxito con JUnit. |
| Resultado | **EXITO** |
| Automatizada | Sí |

|  |  |
| --- | --- |
| ID | **Prueba 4** |
| Descripción | Prueba para la detección de errores al implementar búsquedas en AliExpress usando servicios RESTful. |
| Salida esperada | Los datos devueltos en formato JSON son mapeados a una clase Java y a continuación se muestra si ha habido éxito con JUnit. |
| Resultado | **EXITO** |
| Automatizada | Sí |

|  |  |
| --- | --- |
| ID | **Prueba 5** |
| Descripción | Prueba para la detección de errores al implementar búsquedas de animes en MyAnimeList usando servicios RESTful. |
| Salida esperada | Los datos devueltos en formato XML son mapeados a una clase Java y a continuación se muestra si ha habido éxito con JUnit. |
| Resultado | **EXITO** |
| Automatizada | Sí |

|  |  |
| --- | --- |
| ID | **Prueba 6** |
| Descripción | Prueba para la detección de errores al implementar búsquedas perfiles en MyAnimeList usando servicios RESTful. |
| Salida esperada | Los datos devueltos en formato XML son mapeados a una clase Java y a continuación se muestra si ha habido éxito con JUnit. |
| Resultado | **EXITO** |
| Automatizada | Sí |

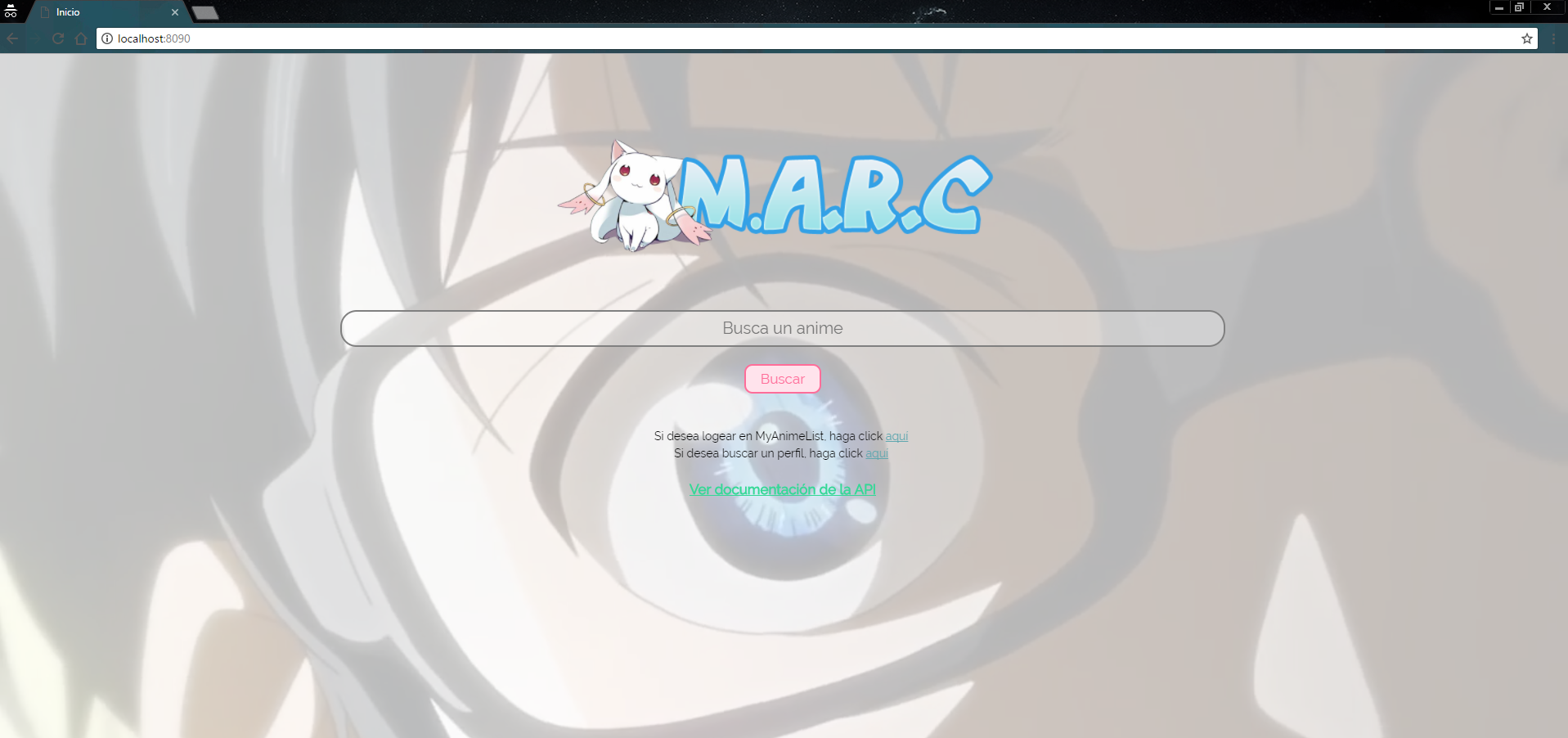
|  |  |
| --- | --- |
| ID | **Prueba 7** |
| Descripción | Prueba para la detección de errores al implementar adición de animes en MyAnimeList usando servicios RESTful. |
| Salida esperada | Los datos devueltos en formato XML son mapeados a una clase Java y a continuación se muestra si ha habido éxito con JUnit. |
| Resultado | **EXITO** |
| Automatizada | Sí |

|  |  |
| --- | --- |
| ID | **Prueba 8** |
| Descripción | Prueba para la detección de errores al implementar actualización de animes en MyAnimeList usando servicios RESTful. |
| Salida esperada | Los datos devueltos en formato XML son mapeados a una clase Java y a continuación se muestra si ha habido éxito con JUnit. |
| Resultado | **EXITO** |
| Automatizada | Sí |

|  |  |
| --- | --- |
| ID | **Prueba 9** |
| Descripción | Prueba para la detección de errores al implementar borrado de animes en MyAnimeList usando servicios RESTful. |
| Salida esperada | Los datos devueltos en formato XML son mapeados a una clase Java y a continuación se muestra si ha habido éxito con JUnit. |
| Resultado | **EXITO** |
| Automatizada | Sí |

# Manual de usuario

## Mashup



La idea es que el cliente realice una búsqueda en la barra de búsqueda que aparece justo arriba de este texto obteniendo información de las distintas aplicaciones que implementamos.

Incluye funciones para guardar una serie en tu cuenta de MyAnimeList de forma paralela a como si lo hicieras desde ella.

## API REST

Para toda la documentación de nuestra API, acceder aquí:

<https://proyecto-marc-aiss.appspot.com/docs/index.html>

