**Proyecto DAPP**



Arquitectura e Integración de Sistemas Software

Grado de Ingeniería del Software

Curso 2

Gonzalo Álvarez García (gonalvgar@alum.us.es)

Alfonso Cadenas Morales (alfcadmor@alum.us.es)

Guillermo Losada Ostos (guilosost@alum.us.es)

Miguel Yanes Ariza (migyanari@alum.us.es)

Tutor: Javier Troya Castilla

Número de grupo:

Enlace de la aplicación: <https://project-dapp.appspot.com/>

Historial de versiones

|  |  |  |  |
| --- | --- | --- | --- |
| Fecha | Versión | Detalles | Participantes |
| 17/03/2019 | 1.0 | - Incluye introducción, prototipos de las interfaces de usuario y diagramas UML de componentes y despliegue. | Gonzalo Álvarez García  Alfonso Cadenas Morales  Guillermo Losada Ostos  Miguel Yanes Ariza |
| 28/04/2019 | 2.0 | - Incluye todo el contenido del anterior entregable actualizado a las nuevas APIs que se han implementado. Además, se añade la API y detalles de la implementación. | Gonzalo Álvarez García  Alfonso Cadenas Morales  Guillermo Losada Ostos  Miguel Yanes Ariza |
| 20/05/2019 | 3.0 | -Incluye todo el contenido del anterior entregable actualizando a las nuevas APIs que se han implementado. Se actualiza nuestra API Rest, amplían las pruebas y detalles de implementación. | Gonzalo Álvarez García  Alfonso Cadenas Morales  Guillermo Losada Ostos  Miguel Yanes Ariza |

Índice

[1 Introducción 4](#_Toc9509788)

[1.1 Aplicaciones integradas 4](#_Toc9509789)

[1.2 Evolución del proyecto 4](#_Toc9509790)

[2 Prototipos de interfaz de usuario 5](#_Toc9509791)

[2.1 Vista inicio de sesión 5](#_Toc9509792)

[2.2 Vista de búsqueda 6](#_Toc9509793)

[2.3 Vista de las estadísticas 6](#_Toc9509794)

[3 Arquitectura 7](#_Toc9509795)

[3.1 Diagrama de componentes 7](#_Toc9509796)

[3.2 Diagrama de despliegue 7](#_Toc9509797)

[3.3 Diagrama de secuencia de alto nivel 7](#_Toc9509798)

[3.4 Diagrama de clases 9](#_Toc9509799)

[3.5 Diagramas de secuencia 10](#_Toc9509800)

[4 Implementación 11](#_Toc9509801)

[5 Pruebas 12](#_Toc9509802)

[6 Manual de usuario 12](#_Toc9509803)

[6.1 Mashup 12](#_Toc9509804)

[6.2 API REST 14](#_Toc9509805)

[Referencias 15](#_Toc9509806)

# Introducción

Esta aplicación tiene como objetivo realizar búsquedas simultáneas de imágenes y vídeos en el ámbito digital, reuniendo tres aplicaciones de image/video hosting service (Youtube, DeviantArt y Dailymotion) en un mashup. Otra de las funcionalidades que ofrece es la facilidad para ofrecer feedback a las publicaciones recibidas tras la búsqueda mediante valoraciones y comentarios.

Por último, proporcionará las estadísticas de la cuenta con la que hayamos iniciado sesión como el número de visitas, la publicación más vista, valoraciones totales, etc.

## Aplicaciones integradas

Las tres aplicaciones que conforman el mash-up ofrecen servicios de redes sociales en las que se puede publicar de forma inmediata en forma de imágenes y vídeos.

De nuevo, las tres aplicaciones tienen un formato muy similar, por lo que explicar una a una estas sería repetirse sin razón. Básicamente, son plataformas de subida e intercambio de imágenes y/o vídeos online, donde las fotos y vídeos de los usuarios pueden ser sometidas a las críticas y opiniones de otros usuarios.

|  |  |
| --- | --- |
| Nombre aplicación | URL documentación API |
| DeviantArt | <https://www.deviantart.com/developers/> |
| Youtube | <https://developers.google.com/youtube/v3/> |
| DailyMotion | <https://developer.dailymotion.com/api> |

Tabla 1. Aplicación integradas

## Evolución del proyecto

La idea inicial consistía en implementar los clientes de Twitter, Tumblr e Instagram dentro de nuestra aplicación y poder publicar tanto en todas las redes a la vez como en cada una individualmente. Por incompatibilidades en los servicios de autenticación, tuvimos que cambiar dos de las tres aplicaciones y añadir otra más, que son las definitivas: Twitter, Facebook, Reddit y Pinterest – todas coinciden en el uso de OAuth2.

Además, descartamos implementar el cliente completo de todas las aplicaciones ya que es algo muy complejo y no mejoraría el uso de la aplicación. Por ello, finalmente decidimos que la aplicación solo publicaría en todas las redes e informaría de las estadísticas de feedback de las publicaciones realizadas.

Tras el cambio de aplicaciones del primer entregable, volvimos a encontrarnos con más problemas. En este caso, la problemática radicaba en los permisos que precisa Facebook para poder usar sus servicios de publicación y recogida de datos y la obtención del access token de Reddit, por lo que tuvimos que deshacer completamente el primer proyecto y empezar de cero otro nuevo.

Para este segundo proyecto intentamos implementar una aplicación de funcionalidad muy parecida a la anterior solo que cambiando el tipo de publicación que se hace, que pasa de ser en formato de texto a formato imagen, ya que solo utilizaremos plataformas donde subir imágenes – DeviantArt, Unsplash, Imgur, Flicker y Google Photos.

Sin embargo, nos volvimos a encontrar con problemas en las APIs seleccionadas previamente, viéndonos forzados a cambiar las aplicaciones de nuevo por las plataformas de publicación de imagen y vídeo Youtube, DeviantArt y Dailymotion. Aún así, sin alejarse mucho de la idea inicial: poder hacer consultas y POSTs en estas plataformas. Además de obtener ciertas estadísticas según los resultados que nos proporcione cada búsqueda.

# Prototipos de interfaz de usuario

## Vista inicio de sesión

Vista del inicio de sesión en las cuatro redes sociales.

Imagen que contiene captura de pantalla

Descripción generada automáticamente

Figura 1. Prototipo de interfaz de usuario de la vista de inicio de sesión

## Vista de búsqueda

Vista de la interfaz al realizar una búsqueda.

Imagen que contiene captura de pantalla

Descripción generada automáticamente

Figura 2. Prototipo de interfaz de usuario de la vista de búsqueda

## Vista de las estadísticas

Vista de las estadísticas de las publicaciones que se han hecho desde las diferentes redes sociales para conocer el feedback.

Imagen que contiene captura de pantalla

Descripción generada automáticamente

Figura 3. Prototipo de interfaz de usuario de la vista de las estadísticas

# Arquitectura

## Diagrama de componentes

Imagen que contiene captura de pantalla

Descripción generada automáticamente

## Diagrama de despliegue



## Diagrama de secuencia de alto nivel

Diagrama de publicación de comentarios y like en todas las plataformas.

Imagen que contiene captura de pantalla

Descripción generada automáticamente

Diagrama de búsqueda de fotos/vídeos simultánea en todas las plataformas.

Imagen que contiene captura de pantalla

Descripción generada automáticamente

Diagrama de petición de las estadísticas de cada plataforma.

Imagen que contiene captura de pantalla

Descripción generada automáticamente

## Diagrama de clases

Diagrama MVC SearchImages

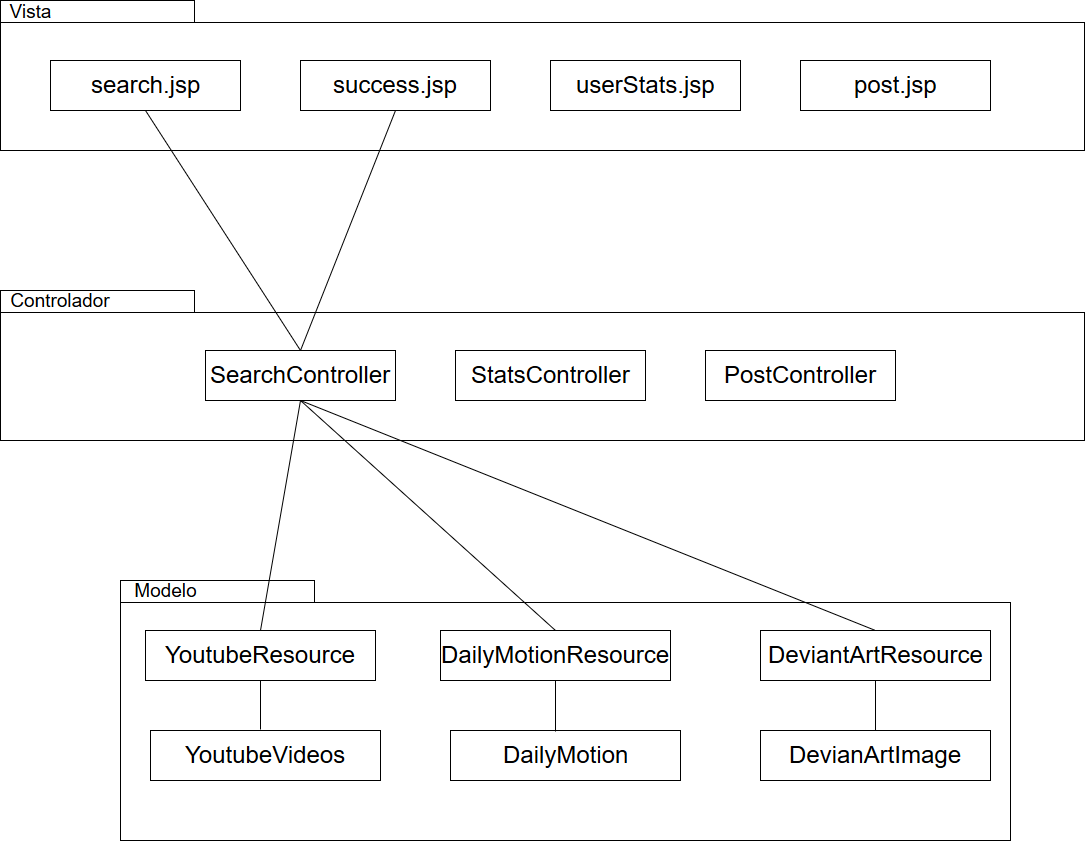


Diagrama MVC StatsImages

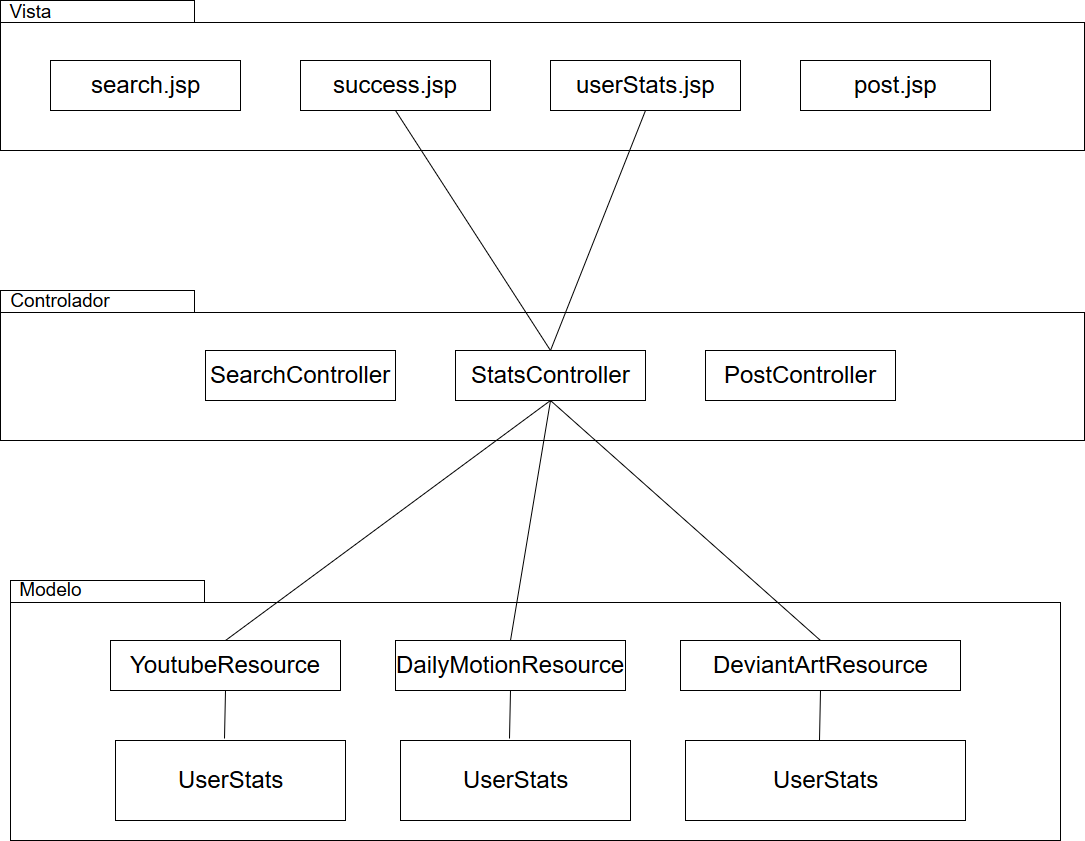
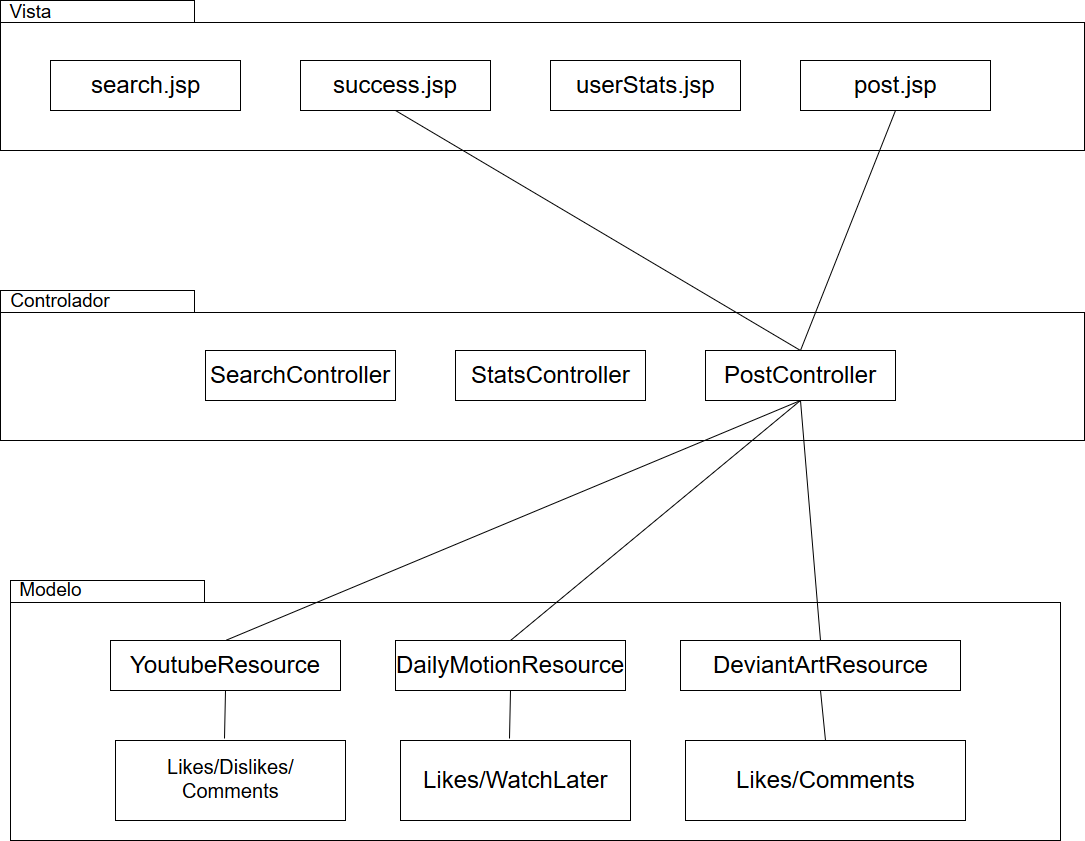


Diagrama MVC PostImages



## Diagramas de secuencia

Diagrama de secuencia MVC de búsqueda

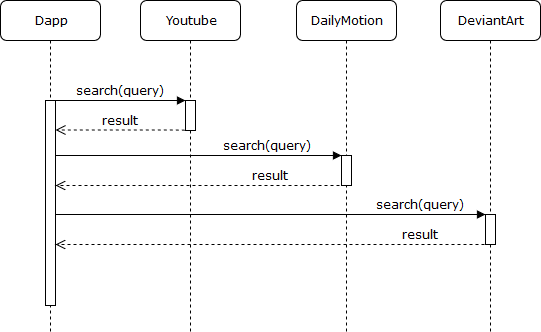
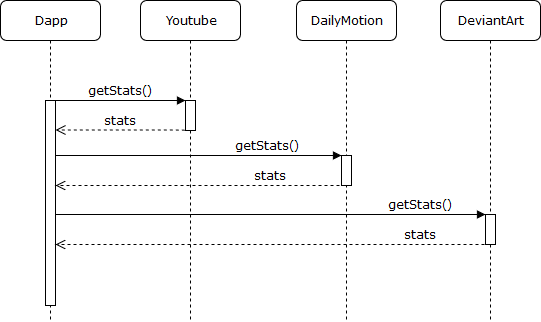
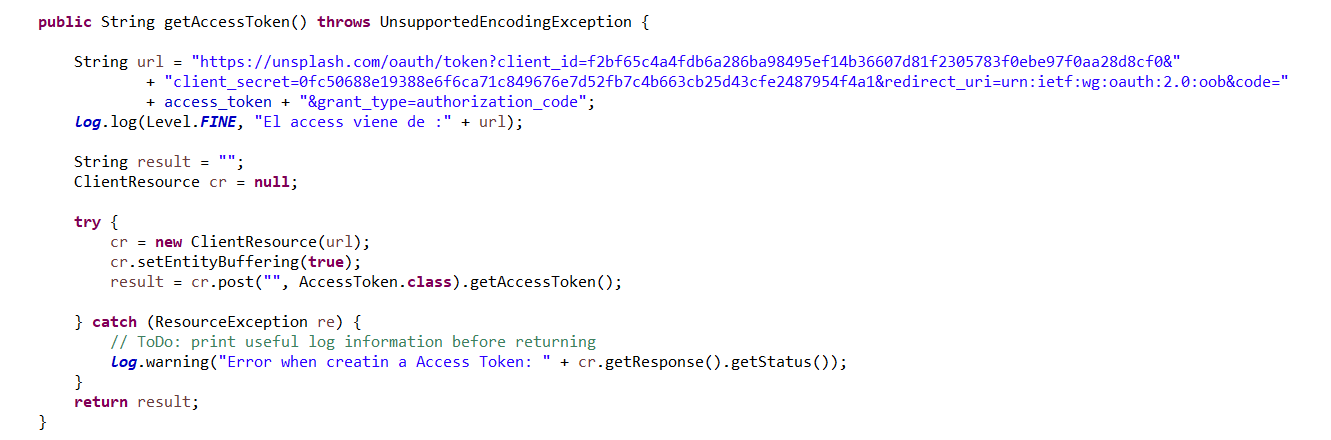


Diagrama de secuencia MVC de estadísticas de usuario



# Implementación

El fragmento de código que más problemática nos ha generado ha sido el siguiente:



No por su complejidad en sí mismo sino por el proceso hasta llegar a implementarlo, ya que el proceso de autenticación de Unsplash es un poco peculiar.

Para obtener el access token, primero debemos hacernos con un código de autenticación que será utilizado posteriormente en la URL de la que obtendremos el access token. El problema que se nos presentaba es que conseguíamos sacar el código de autenticación con el código proporcionado por uno de los laboratorios realizados en clase, pero no sabíamos cómo sacar el access token de un POST a una URL. Finalmente, usamos un RESTlet Client para conocer el formato del recurso del access token y crear una clase Java con él, conseguimos ejecutar un POST desde Java a la URL con el código de autenticación correspondiente y le sacamos el access token al objeto resultante haciendo uso de la clase que creamos antes.

# Pruebas

|  |  |
| --- | --- |
| Resumen |  |
| Número total de pruebas realizadas | 1 |
| Número de pruebas automatizadas | 0 (0%) |

|  |  |
| --- | --- |
| ID | **Prueba 1** |
| Descripción | Prueba para la detección de errores al implementar búsquedas en todas las plataformas. |
| Entrada | Se hace uso de la librería XXX para invocar al servicio usando la URI /search?query=nature 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 | No |

# Manual de usuario

## Mashup

La página principal de la aplicación actualmente es la siguiente:

Imagen que contiene captura de pantalla

Descripción generada automáticamente

Si entramos en cualquiera de los botones de inicio de sesión, entraremos en la aplicación otorgándole los derechos especificados en el scope.

Si hacemos la búsqueda con el tema especificado, la página será la siguiente:

Imagen que contiene armario, mobiliario, marcador, carretera

Descripción generada automáticamente

Si entramos en el botón de estadísticas pasaremos a la siguiente vista:



Por último, la vista de la página de información sobre el proyecto:



## API REST

La URL es la siguiente: <http://project-dapp.appspot.com/>

Los recursos serán devueltos en formato JSON.

|  |  |  |
| --- | --- | --- |
| GET | **/images/allImages** | Recopila todas las imágenes. |
| GET | /images/image/{query} | Recopila las imágenes asociadas a un query determinado. |
| GET | /images/image/{id} | Devuelve una imagen con un id determinado. |
| GET | /m/all | Recopila todos los archivos multimedia. |
| GET | /m/all/{query} | Recopila todos los archivos multimedia con que esten asociados a un query determinado. |
| GET | /videos/allVideos | Devuelve todos los videos. |
| GET | /videos/video/{query} | Devuelve los videos asociados a una query determinada. |
| GET | /videos/videoId/{videoId} | Devuelve un video con una id determinada. |
| POST | /image/addImage | Postea una imagen. |
| POST | /videos/addVideo | Añade un video. |
| PUT | /image/updateImage | Actualiza una imagen antigua. |
| PUT | /videos/updateImage | Actualiza un video antiguo. |
| DELETE | /image/imageId/{imageId} | Elimina una imagen con un id determinado. |
| DELETE | /videos/videoId/{videoId} | Elimina un video con una id especifica. |

Códigos de estado del servicio:

|  |  |  |
| --- | --- | --- |
| 200 | **OK** | La solicitud ha sido procesada correctamente. |
| 403 | **FORBIDDEN** | Los datos a los que se quiere acceder están protegidos y se ha denegado el acceso al cliente por falta de autorización. |
| 404 | **NOT FOUND** | No fue posible encontrar los datos requeridos. |
| 500 | **INTERNAL SERVER ERROR** | Error inesperado en el servidor. |

# 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.