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

Índice

1 Introducción

Esta aplicación reúne cinco aplicaciones de publicación de contenido fotográfico con el objetivo de hacer más práctico el hecho de subir imágenes, comentar y dar 'like' a publicaciones y recoger estadísticas del usuario.

El objetivo final es facilitar a profesionales y cualquier usuario el poder compartir su trabajo o hobbie, ya que no sería necesario cambiar de plataforma, sino que se podría hacer todo desde un mismo sitio web.

1.1 Aplicaciones integradas

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

De nuevo, las cinco 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 online, donde las fotos de los usuarios pueden ser sometidas a las críticas y opiniones de otros usuarios.

Por especificar un poco, DeviantArt y Unsplash podrían clasificarse como las más profesionales: la primera en diseño gráfico y la segunda en fotografía profesional.

Nombre aplicación	URL documentación API
DeviantArt	https://www.deviantart.com/developers/
Unsplash	https://unsplash.com/documentation
Imgur	https://api.imgur.com/
Flicker	https://www.flickr.com/services/api/
Google Photos	https://developers.google.com/photos/

TABLA 1. APLICACIÓN INTEGRADAS

1.2 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.

Hasta ahora, hemos conseguido implementar la primera funcionalidad de la aplicación: búsqueda simultánea en cuatro de las cinco aplicaciones (DeviantArt, Unsplash, Imgur y Flickr), la siguiente funcionalidad sería la publicación de imágenes en DeviantArt, Unsplash, Imgur y Google Photos, pero no está operativa aún.

2 Prototipos de interfaz de usuario

2.1 Vista inicio de sesión

Vista del inicio de sesión en las cuatro redes sociales.

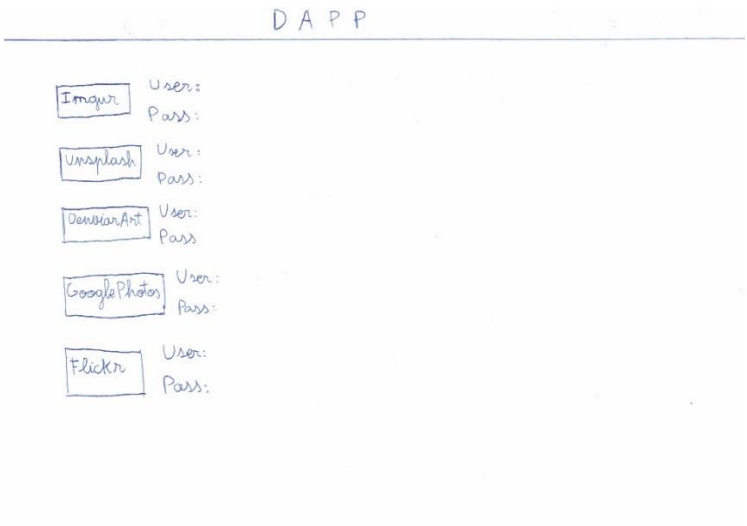


FIGURA 1. PROTOTIPO DE INTERFAZ DE USUARIO DE LA VISTA DE INICIO DE SESIÓN

2.2 Vista de la publicación

Vista de la interfaz para realizar la publicación simultánea.

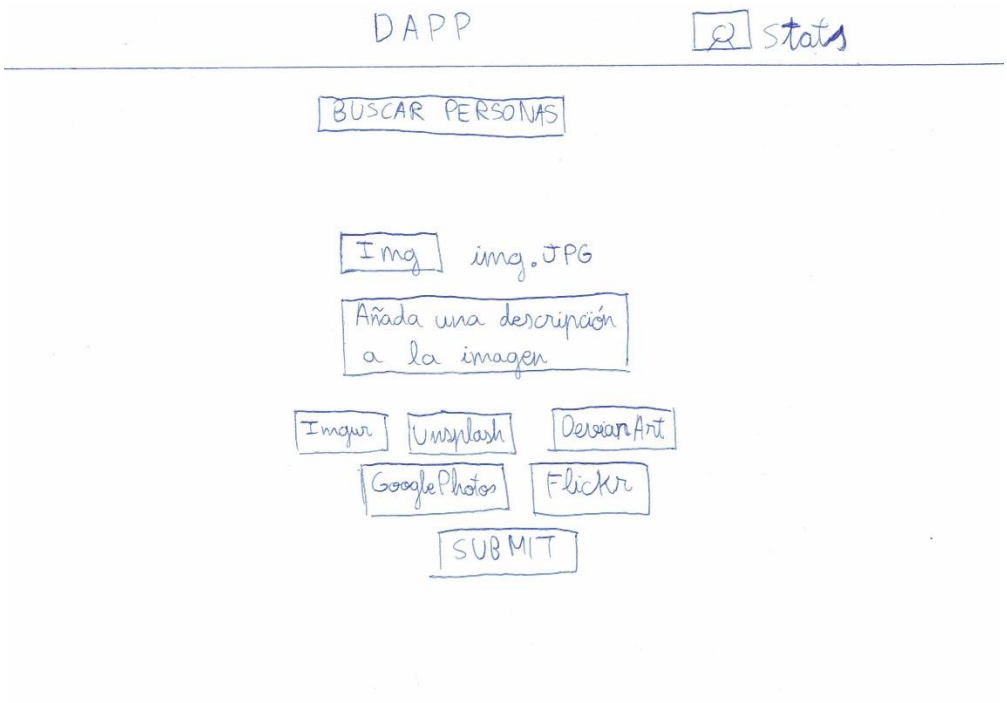


FIGURA 2. PROTOTIPO DE INTERFAZ DE USUARIO DE LA VISTA DE PUBLICACIÓN

2.3 Vista del filtro de búsqueda

Vista del filtro de búsqueda que permite buscar a un usuario en las diferentes redes sociales al mismo tiempo.

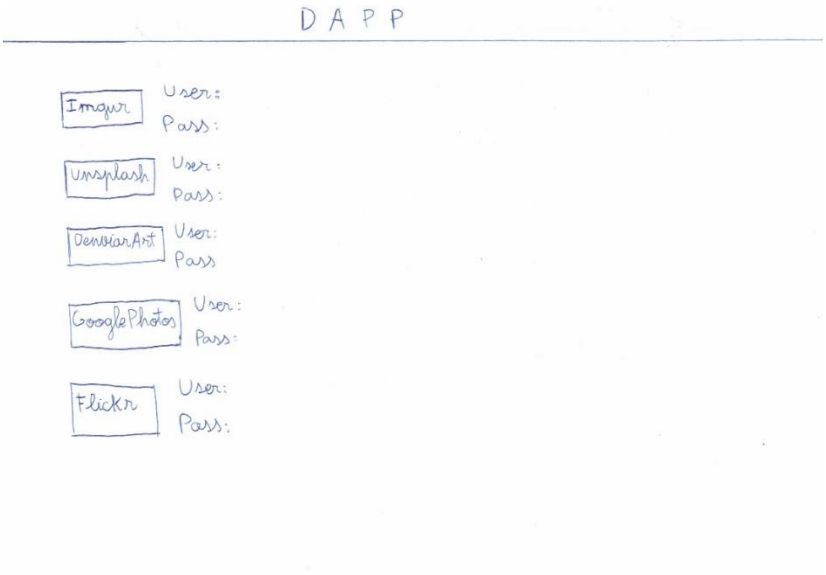


FIGURA 3. PROTOTIPO DE INTERFAZ DE USUARIO DE LA VISTA DEL FILTRO DE BÚSQUEDA

2.4 Vista de las estadísticas

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

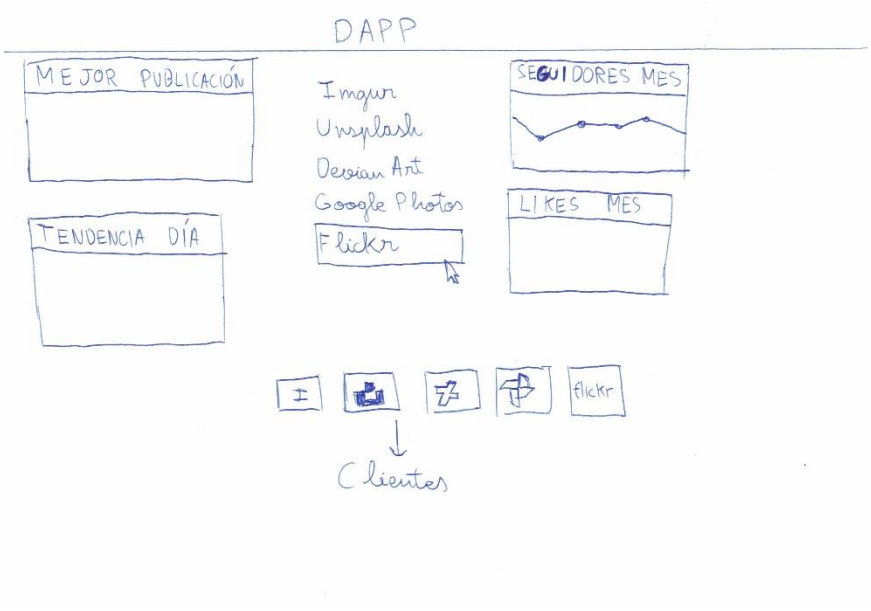
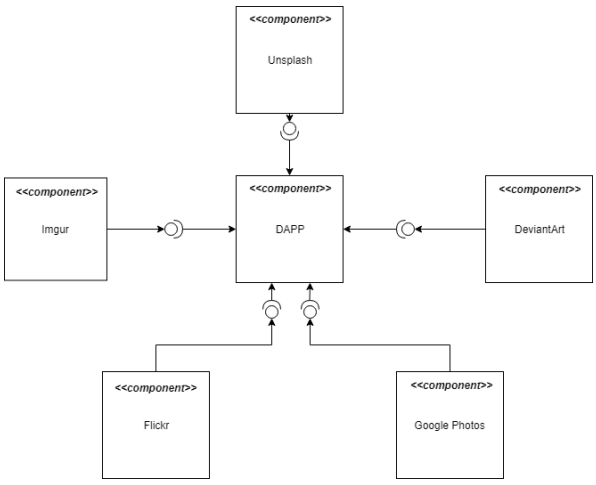


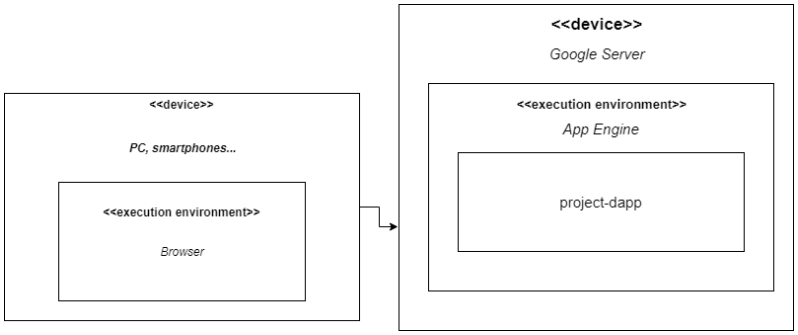
FIGURA 4. PROTOTIPO DE INTERFAZ DE USUARIO DE LA VISTA DE LAS ESTADÍSTICAS

3 Arquitectura

3.1 Diagrama de componentes



3.2 Diagrama de despliegue



3.3 Diagrama de secuencia de alto nivel

Diagrama de publicación en todas las plataformas.

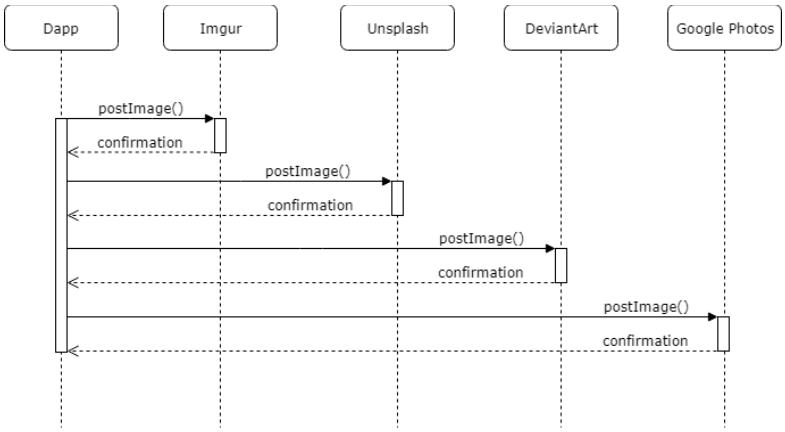


Diagrama de búsqueda de fotos simultánea en todas las plataformas.

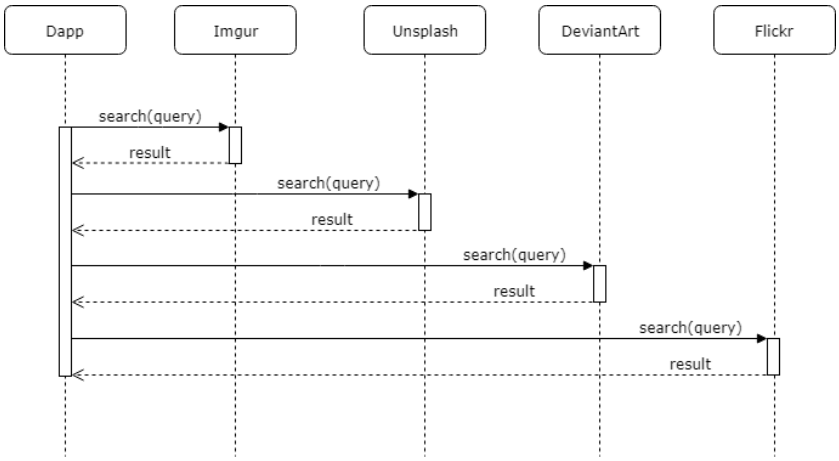
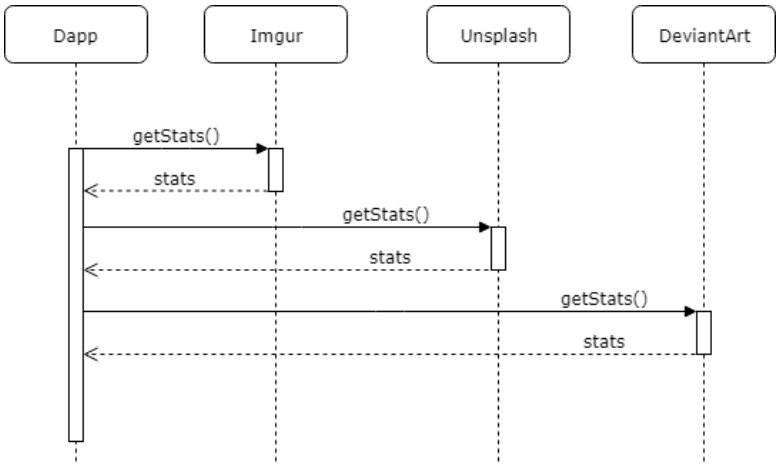


Diagrama de petición de las estadísticas de cada plataforma.



3.4 Diagrama de clases

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

Diagrama MVC SearchImages

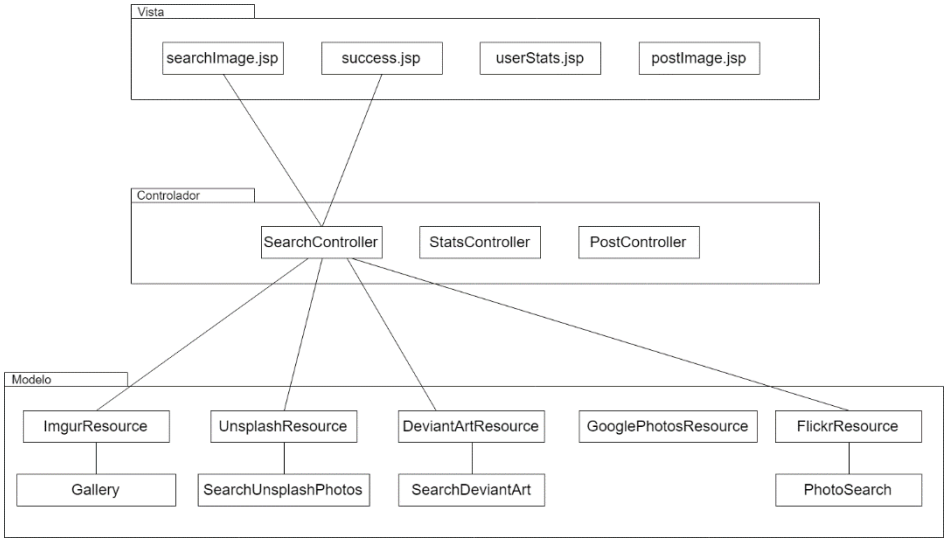
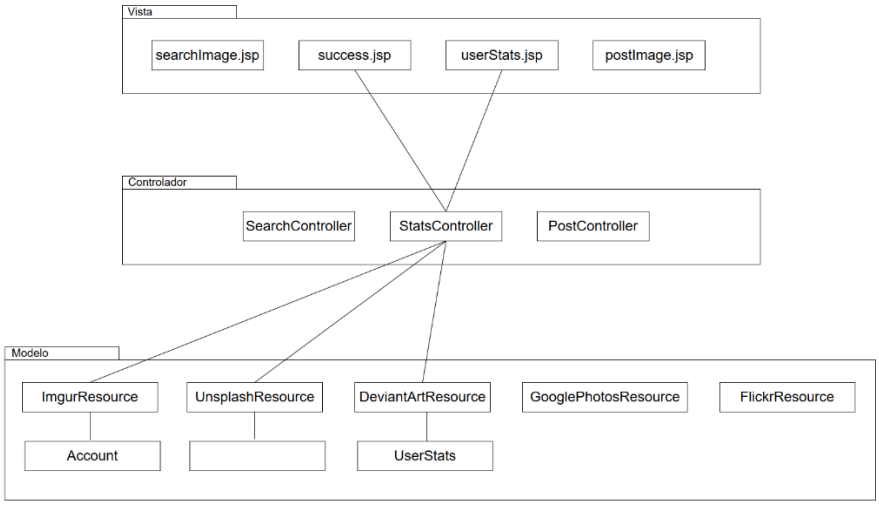
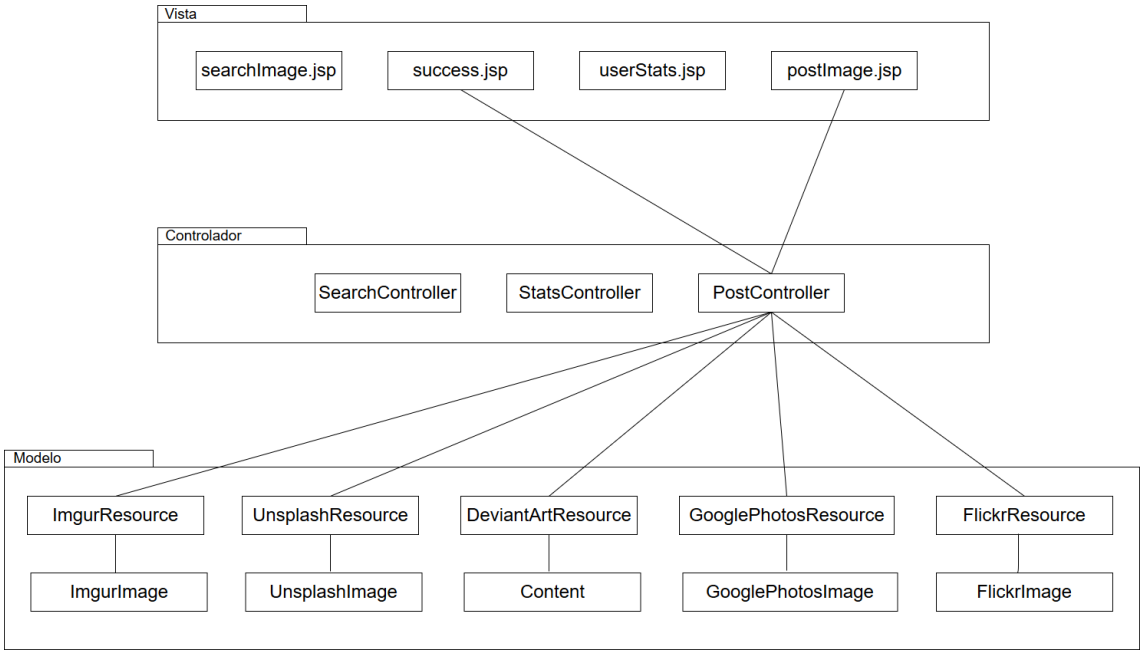


Diagrama MVC StatsImages



Text

Diagrama MVC PostImages



3.5 Diagramas de secuencia

Diagramas UML de secuencia ilustrando la comunicación entre vistas, controladores y clases del modelo.

Diagrama de secuencia MVC de búsqueda

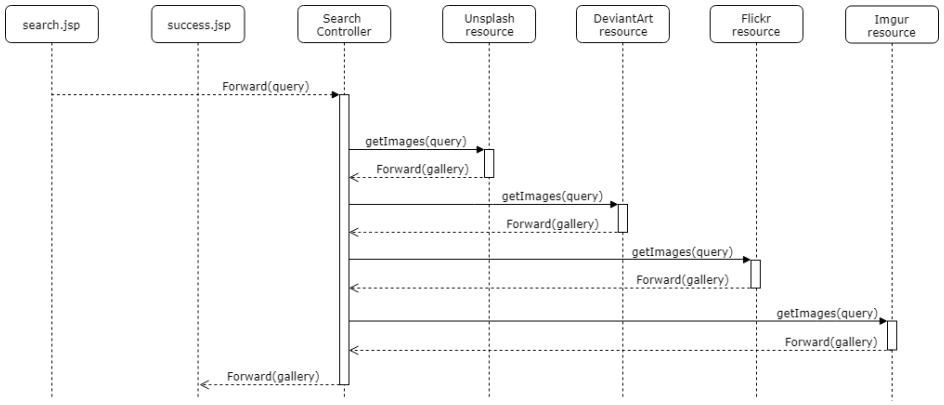
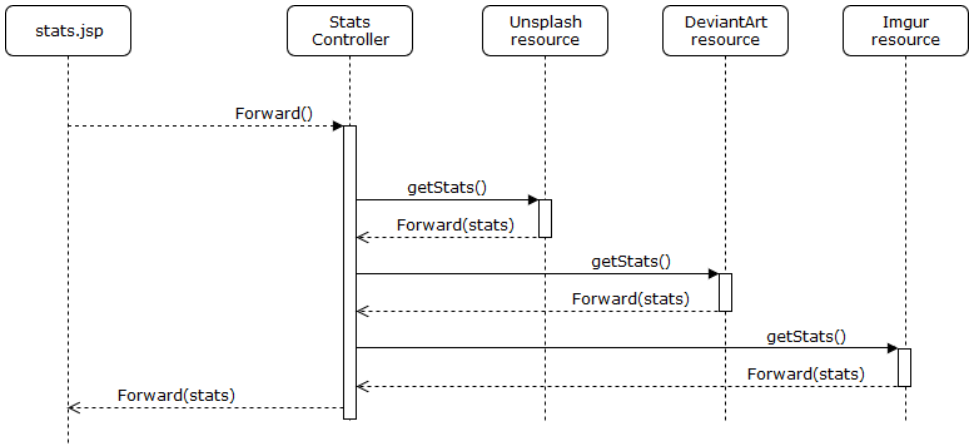


Diagrama de secuencia MVC de estadísticas de usuario



4 Implementación

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

```
public String getAccessToken() throws UnsupportedEncodingException {

    String url = "https://unsplash.com/oauth/token?client_id=f2bf65c4a4fdb6a286ba98495ef14b36607d81f2305783f0ebe97f0aa28d8cf0&"
            + "client_secret=0fc50688e19388e6f6ca71c849676e7d52fb7c4b663cb25d43cfe2487954f4a1&redirect_uri=urn:ietf:wg:oauth:2.0:oob&code="
            + access_token + "&grant_type=authorization_code";
    Log.log(Level.FINE, "El access viene de :" + url);

    String result = "";
    ClientResource cr = null;

    try {
        cr = new ClientResource(url);
        cr.setEntityBuffering(true);
        result = cr.post("", AccessToken.class).getAccessToken();

    } catch (ResourceException re) {
        // ToDo: print useful log information before returning
        Log.warning("Error when creatin a Access Token: " + cr.getResponse().getStatus());
    }
    return result;
}
```

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.

5 Pruebas

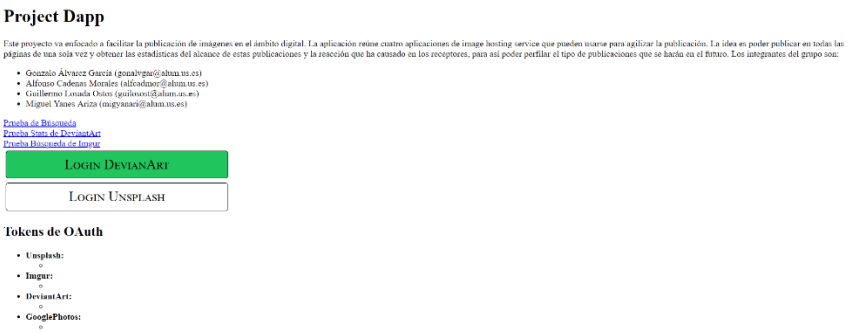
Resumen	
Número total de pruebas realizadas	1
Número de pruebas automatizadas	0 (0%)

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	Sí

6 Manual de usuario

6.1 Mashup

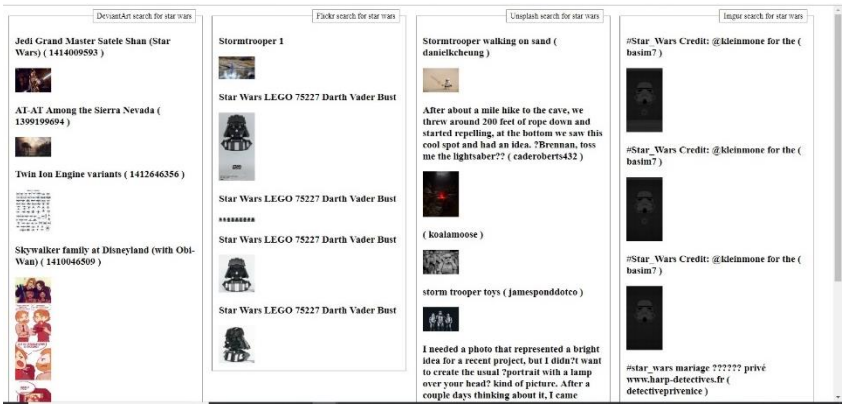
La página principal de la aplicación actualmente es la siguiente:



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

Si entramos a la prueba de búsqueda, la página será la siguiente:

Podremos realizar la búsqueda especificando el tema del cuál queremos las fotos, pasando a la siguiente interfaz:



6.2 API REST

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

Los recursos serán devueltos en formato JSON.

GET	**/search?query={q}**	Realiza una búsqueda relacionada con la query enviada en todas las plataformas al mismo tiempo.
GET	**/stats**	Recopila información sobre las estadísticas del usuario.
POST		

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.