PhotoLoad

Publica y descarga las fotos de cada día

Desplegada en: https://etsiiphotoload.appspot.com
Código fuente: https://github.com/ManuelLR/PhotoLoad

Arquitectura e Integración de Sistemas Software
Segundo Curso del Grado de Ingeniería del Software
Grupo 2
Universidad de Sevilla

Realizado por:

Manuel Francisco López Ruiz (manloprui1@alum.us.es)
Miguel Rodríguez Caballero (miguel.rc95@gmail.com)
David Tinoco Castillo (daviid.tc@gmail.com)

Tutorizado por:

Ana Belén Sánchez Jerez (anabsanchez@us.es)

Índice

1. Introducción

Hoy en día subir fotos a las redes sociales es una tarea tediosa y descargarlas es más complicado aun pero, si además queremos hacer esto con todas las fotos a la vez, ya se convierte en algo imposible.

Es ahí donde interviene PhotoLoad, un sencillo mashup ideado para facilitar estas tareas que todos realizamos alguna vez. Tras iniciar sesión mediante PhotoLoad en las aplicaciones que desees usar, podrás ver las fotos de todos tus dispositivos pudiendo publicarlas en Facebook o Flickr con un solo click. Pero la mayor funcionalidad de PhotoLoad radica de la posibilidad de descargar las fotos de Facebook y Flickr en Google Drive, Dropbox o a tu ordenador con un simple click, ya sean pocas o muchas, con PhotoLoad podrás hacerlo.

1.1. Aplicaciones Integradas

Como se ha mencionado anteriormente, PhotoLoad integrará 4 aplicaciones:

- **Facebook** (https://developers.facebook.com/docs/javascript). API de la cual se pueden publicar fotos en el muro y descargar fotos. Esta aplicación se ha implementado mediante OAuth2 y servicio Rest.
- **Flickr** (https://www.flickr.com/services/api/). API de la cual se pueden descargar fotos. Esta aplicación mediante OAuth1 y servicio Rest.
- **Google Drive** (https://developers.google.com/drive/), la cual integra la anteriormente denominada Picasa Web Album. De esta API se obtienen las fotos de los dispositivos gracias a la aplicación Android denominada "Fotos"la cual realiza copias de seguridad de ellas automáticamente. Esta API nos permitirá descargar fotos (de la copia de seguridad y de otras fuentes). Esta aplicación se ha implementado mediante OAuth2 y servicio Rest.
- **Dropbox** (https://www.dropbox.com/developers/core). De esta API se pueden exportar las fotos para publicarlas en las redes sociales o importarlas de las mismas. Esta aplicación se ha implementado mediante OAuth2 y servicio Rest.

1.2. Evolución del proyecto

1.2.1. Integración con Google Drive

Durante la exploración de la api de Google Fotos, vimos que habían realizado un cambio en el funcionamiento de la misma, lo que no nos permitía usarla como teníamos planeado. Por ello decidimos sustituirla por Google Drive en la cual podías incluir una carpeta con tus fotos de Google Fotos mediante una nueva función fruto del cambio de la api por parte de Google. En este punto nos encontramos con la dificultad, que aún no hemos subsanado, de navegar entre las carpetas de Google Drive lo que nos imposibilita acceder directamente a la carpeta de fotos para evitar molestias al usuario final de la

aplicación. A su vez, hemos conseguido descargar los archivos al ordenador pero no ha sido posible obtener el enlace de descarga de los mismos para poder subirlos directamente a los otros servicios que integra PhotoLoad.

Aparte de dichos problemas, también se han encontrado los problemas comunes detallados en el apartado 1.2.5.

1.2.2. Integración con Dropbox

Inicialmente, Dropbox se iba a usar mediante dos apis: Dropbox Chooser y Dropbox Saver. Estas apis se decidieron sustituir por los servicios Rest prestados mediante la Dropbox Core API (Rest) ya que, las otras apis, estaban preparadas para integrarlas directamente en una web impidiéndonos pasar los archivos entre los diferentes servicios sin descargarlos en el ordenador. Tras el cambio de aplicación, primero nos encontramos problemas con la autenticación debido a que Dropbox solo permite el envío y reenvío de peticiones mediante conexión cifrada (https), y después tuvimos algún que otro problema relacionado con los permisos asignados a la aplicación desde Dropbox.

La subida de archivos no se ha podido implementar hasta la fecha debido a la mala documentación por parte de Dropbox.

Aparte de dichos problemas, también se han encontrado los problemas comunes detallados en el apartado 1.2.5.

1.2.3. Integración con Facebook

Esta api fue la última api que implementamos ya que, esperamos hasta realizar la práctica en la que se realizarían peticiones a Facebook mediante OAuth2 (práctica 7). Aun así, nos encontramos con el problema de que no se podían hacer peticiones que implicarán datos del usuario, a pesar de tener el acces_token (ni en la misma práctica se pudieron realizar). Esto se debía al reciente cambio de políticas en los permisos para realizar las peticiones que había implementado Facebook (justamente la noche anterior a la realización de la práctica 7). Tras esto, decidimos pedir a Facebook los permisos necesarios, los cuales fueron rechazados dos veces (estamos esperando la respuesta de la tercera petición). Para solventar este problema, añadimos una caja de texto la cual se rellena automáticamente con el access_token pero que, debido a los permisos ofrecidos a nuestra aplicación por parte de Facebook, no es válido para realizar las peticiones. En su lugar se puede generar un access_token desde este link: `https://developers.facebook.com/tools/explorer/`. Necesitaremos seleccionar (mediante el botón Get Token) los permisos de user_photos (para obtener tus fotos) y, en la sección Extended Permission de esa misma ventana emergente, seleccionaremos publish_actions (para poder publicar las fotos). Tras esto, solo deberemos sustituir el access_token proporcionado por PhotoLoad por el obtenido desde Facebook manualmente.

La aplicación y los servicios Rest no dieron ningún tipo de problemas.

1.2.4. Integración con Flickr

Esta api ha sido la más complicada y problemática desde el inicio del proyecto. Primeramente se iba a usar Instagram, a pesar de que no dispusiera de librerías en java pero, al comprobar que solo se permitía la subida de fotos desde sus aplicaciones oficiales [1], decidimos reemplazarla por Flickr, la cual se parece a Intragram pero orientada a un ámbito más serio y profesional.

Flickr no mantiene ninguna librería para interactuar con la aplicación si no que muestra una lista de las librerías creadas por los usuarios. Debido a esta característica decidimos implementarlo como servicio Rest ya que de esta manera nos aseguraríamos el correcto funcionamiento. Además de todo esto, Flickr usa OAuth1 para la autenticación de los usuarios, lo que supuso un gran esfuerzo por el grupo para conseguir la integración y el login. En primer lugar buscamos algunas librerías y decidimos implementar la librería de oauth distribuida desde su página web oficial (`oauth.net`). Tras varios días intentando entenderla, implementarla y usarla, decidimos dejarla ya que no habíamos conseguido ningún tipo de avance. Posteriormente, decidimos volver a la web oficial y encontramos otra librería denominada OAuth-Signpost. Esta librería fue más complicada todavía ya que tuvimos que aprender a compilar con Maven desde terminal, instalando todos los complementos que requería para ser utilizada. Tras compilarla, encontrar errores de código, corregirlos y recompilar múltiples veces lo único que conseguimos tras tres días fue el mecanismo para firmar las peticiones. Una de las mayores dificultades de esta librería, era la gran modularidad que tenía, llegando a ser tal que no se sabía de que modulo se debían usar las clases.

Tras varios días de búsqueda, encontramos la librería denominada Scribe. Tras compilarla, en media tarde pudimos hacer un login a Flickr desde la terminal, consiguiendo el access_token y todo lo necesario para hacer la primera petición de prueba. Ahora debíamos implementarlo para que funcionara como aplicación GWT con todo lo que ello conlleva ya que, todos los login debían estar en la parte de servidor y solo se le debía dar al cliente los datos necesarios. Tras varios días, conseguimos implementarlo aunque unos días después tuvimos que hacerlo desde el principio ya que, los datos de la autenticación se guardaban en el servidor por lo que no era posible que mas de un usuario pudiera usar la aplicación al mismo tiempo. Tras las modificaciones oportunas, la creación de clases y los parches necesarios para usar esas clases con Scribe, conseguimos un login funcional en GWT desde la interfaz gráfica.

Posteriormente nos encontramos con el problema que conlleva la realización de peticiones mediante el método de OAuth1, la cual debe llevar una firma que sirve para comprobar que los datos hayan llegado correctamente. Por suerte no hubo problema en este apartado, pero no se puede decir lo mismo del parseo de los datos recibidos por las peticiones. La librería Scribe incluye los objetos JSONObject, los cuales sirven para parsear los datos pero, tras entender su funcionamiento, tuvimos que crear las clases de los objetos manualmente y, después de descubrir que Flickr añade a sus respuestas atributos que impiden el parseo por parte de JSONObject, conseguimos hacer funcionar

[1] Véanse las últimas líneas de la documentación oficial: `https://instagram.com/developer/endpoints/media/`

satisfactoriamente las peticiones en una aplicación java local. Al adaptarlo para GWT aparecieron errores, antes inexistentes, debido a problemas con la librería JSONObject por lo que nos vimos en la obligación de usar otras incluidas ya en GWT.

Tras todos estos problemas con Flickr, conseguimos implementar las visualización de las fotos en nuestra aplicación sin mayores problemas. La subida de fotos no se ha podido implementar todavía debido al tiempo consumido por esta api.

Aparte de dichos problemas, también se han encontrado los problemas comunes detallados en el apartado 1.2.5.

1.2.5. Problemas comunes encontrados

Contar el problema de la descarga de archivos eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee eeeeeeeeeeeeeeeee

2. Prototipos de Interfaz de Usuario

A continuación se definirán los prototipos de interfaz (mockup) de PhotoLoad. Se explicará cada vista por separado acompañada de un boceto. La iteración entre vistas se detallará en la sección 3.3 (página 13).

2.1. Interfaz de inicio

Esta vista (Figura 1) será la primera que vea el usuario al acceder a la aplicación. En ella el usuario podrá elegir si desea usar la aplicación o ir a la página de información.

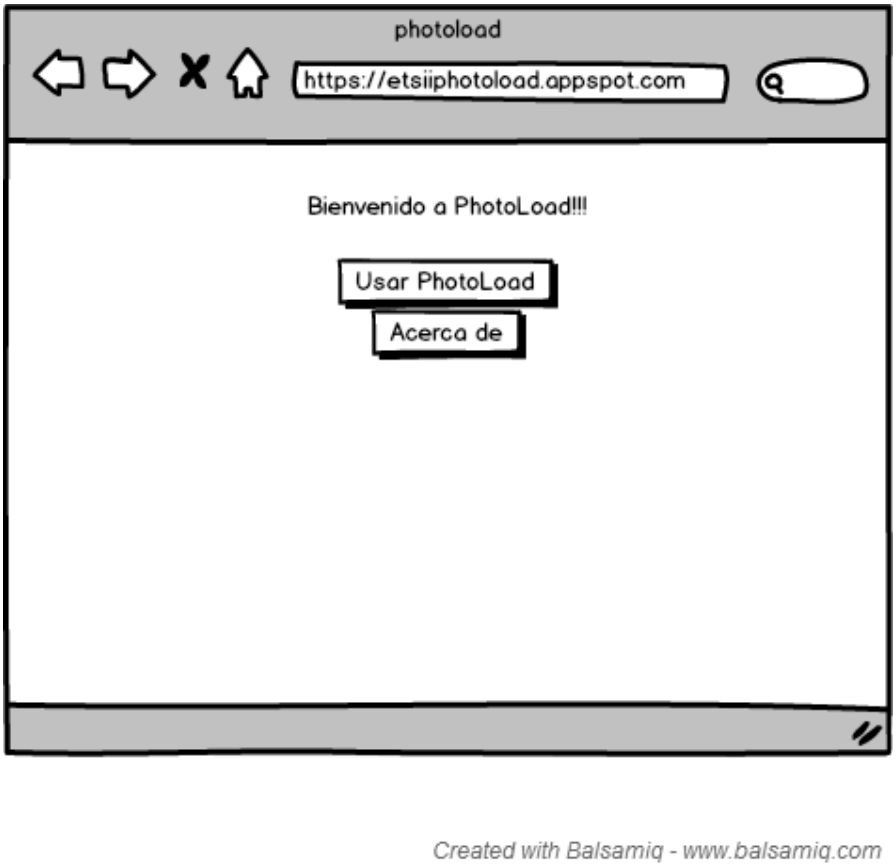

Figura 1: Interfaz de inicio

2.2. Interfaz de selección de descarga

Esta vista (Figura 2) permitirá al usuario seleccionar desde que aplicación desea descargar las fotos que posteriormente subirá. A su vez podrá ir a la página de Inicio (sección 2.1).

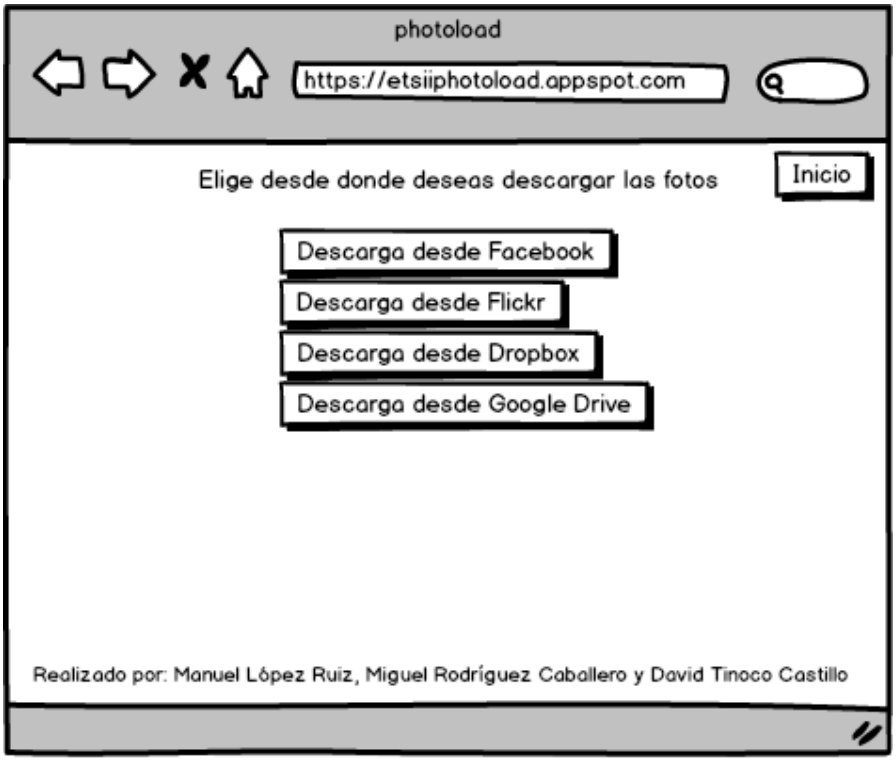

Figura 2: Interfaz de selección de descarga

2.3. Interfaz de cada aplicación

Esta vista dependerá de cada aplicación aunque podremos diferenciar claramente entre la de descarga (Ejemplo: Figura 3) y la de publicación (Ejemplo: Figura 4).

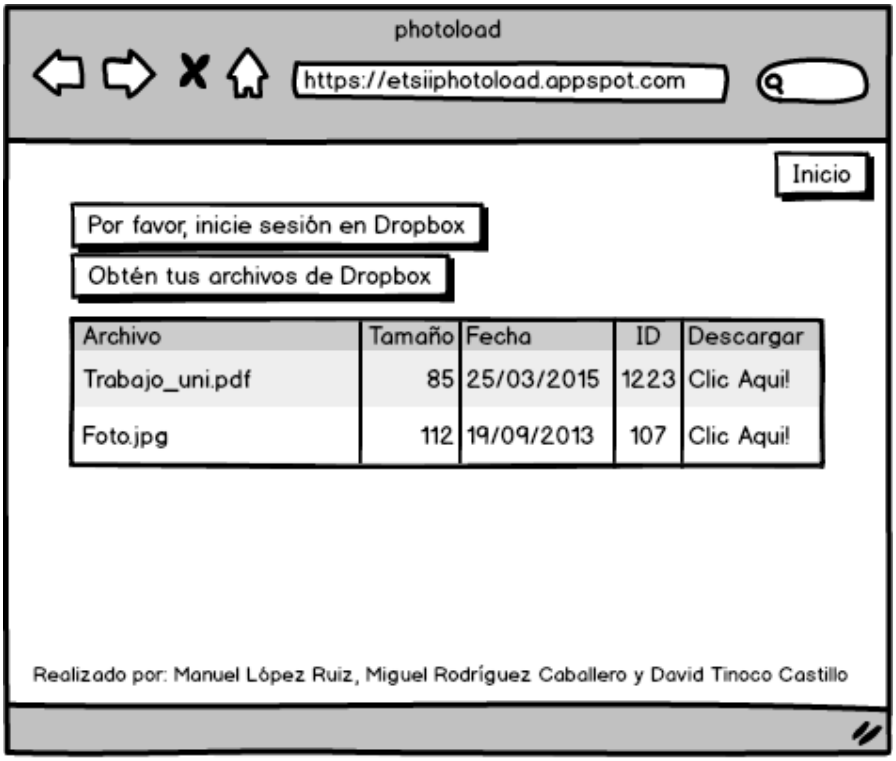

Figura 3: Interfaz de descarga de Dropbox

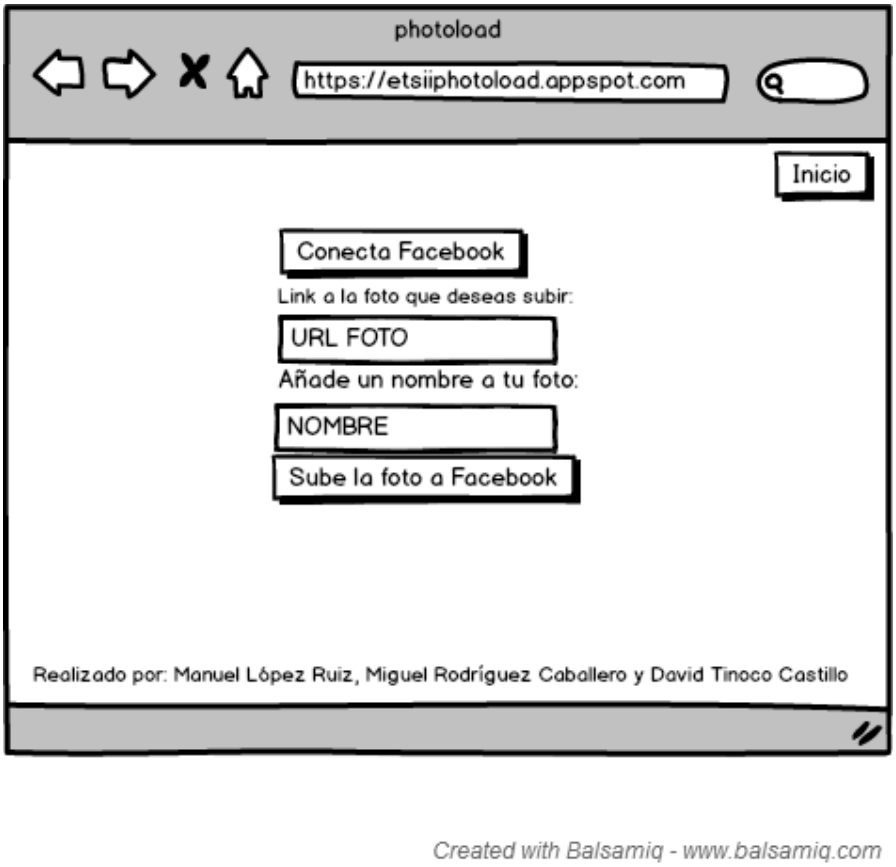

Figura 4: Interfaz de subida de Facebook

2.4. Interfaz de selección de publicación

Esta vista (Figura 5) permitirá al usuario seleccionar a donde desea subir las fotos.

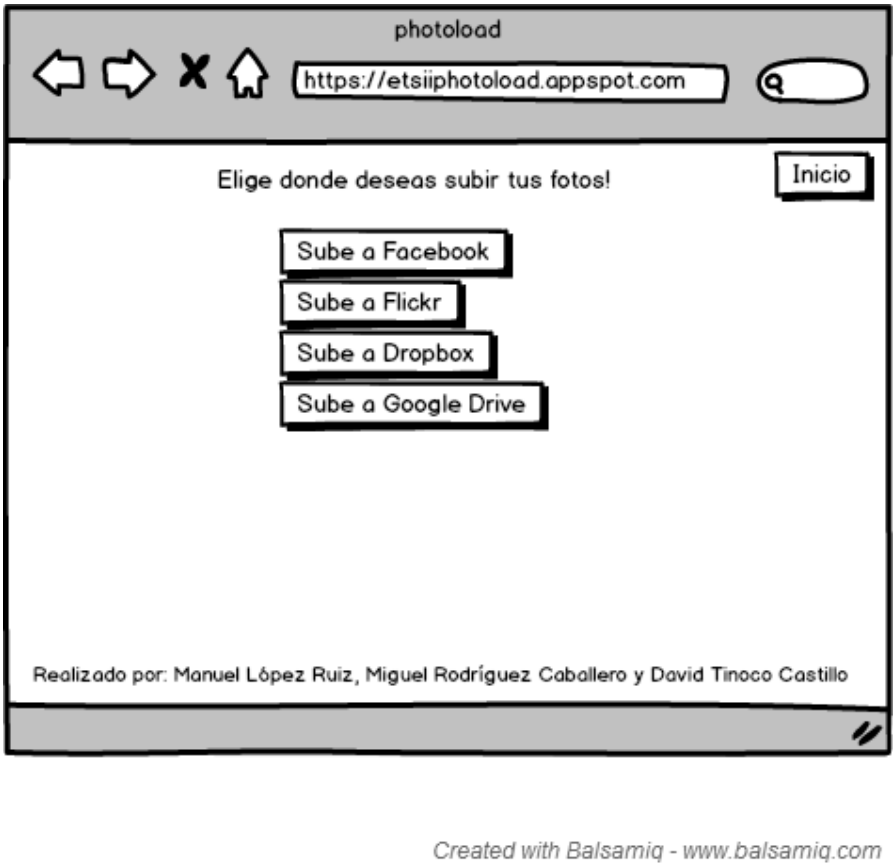

Figura 5: Interfaz de descarga

3. Arquitectura

3.1. Diagrama de Componentes

El diagrama de componentes (Figura 6) muestra las aplicaciones usadas en el mockup y como enlazarán con PhotoLoad. Se nutre de un total de 4 APIs.

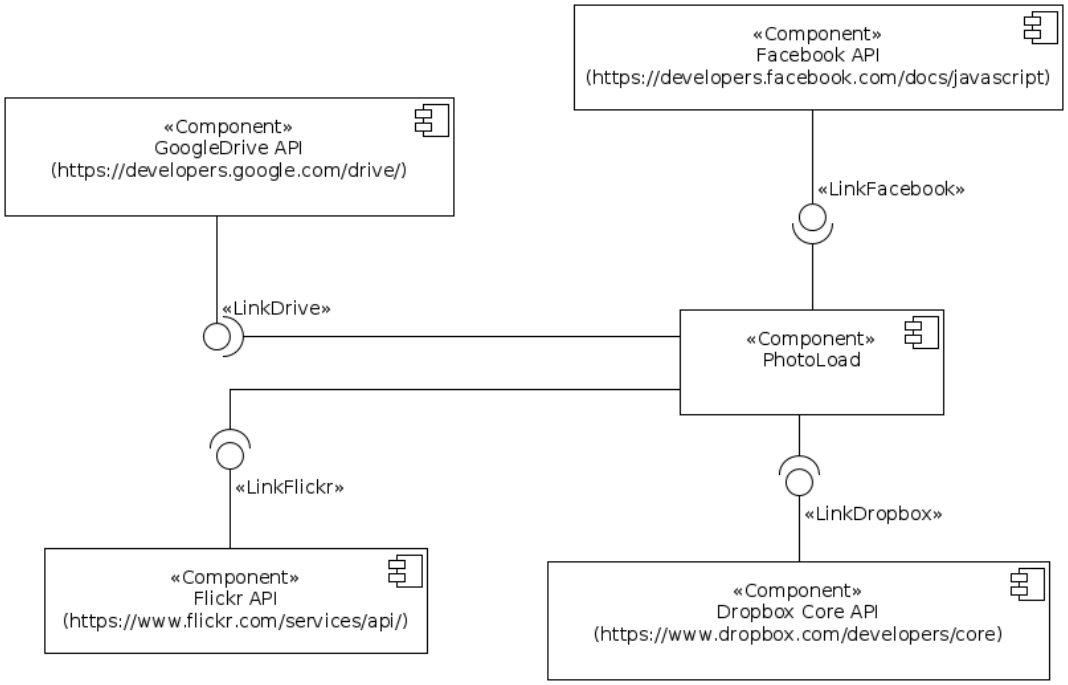

Figura 6: Diagrama de Componentes

3.2. Diagrama de Despliegue

El diagrama de despliegue (Figura 7) muestra como se desplegará PhotoLoad en el servidor (AppEngine) y como lo hará en el navegador web del cliente.

Figura 7: Diagrama de Despliegue

3.3. Diagrama de Vistas

El diagrama de vistas (Figura 8) muestra como se podrá ir navegando por las diferentes vistas de PhotoLoad.

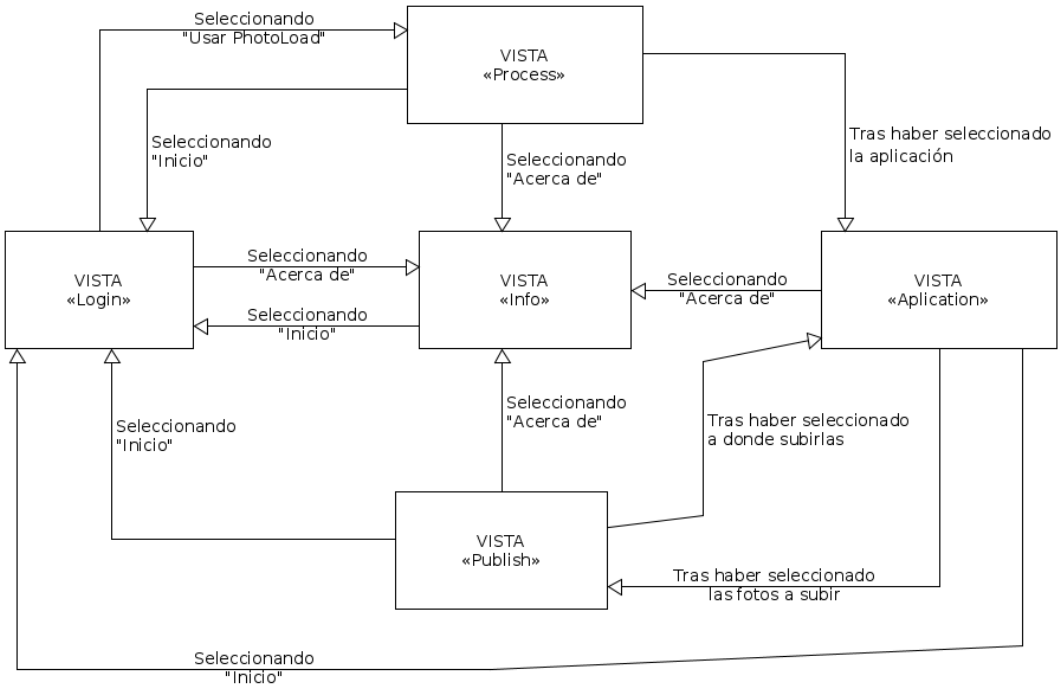

Figura 8: Diagrama de Vistas

3.4. Diagrama de Clases

En esta fase del proyecto no está suficientemente definido el diagrama de clases por lo que se pondrá en futuros entregables.

4. Pruebas

A continuación se expondrá la estructura de las pruebas realizadas en JUnit y separadas por Apis.

4.1. Facebook

ID	Prueba FB1
Descripción	Comprueba la correcta obtención de las fotos existentes en Facebook para el usuario
Entrada	Token
Salida esperada	Lista de Fotos
Resultado	Exito

ID	Prueba FB2
Descripción	Comprueba la correcta subida de fotos al tablon de Facebook
Entrada	Token y url de la foto a subir
Salida esperada	El id de la foto subida
Resultado	Exito

4.2. Flickr

ID	Prueba FL1
Descripción	Comprueba el correcto funcionamiento de las dos fases de login necesarios para Flickr
Entrada	No entra ningún dato
Salida esperada	Un objeto de tipo OAuth con una URL y un AccessToken relleno
Resultado	Exito

ID	Prueba FL2
Descripción	Comprueba la correcta obtención de las fotos existentes en Flickr para el usuario
Entrada	Token (OAuth)
Salida esperada	Lista de FlickrPhoto no null con los ids de las fotos pero sin los links de descarga
Resultado	Exito

ID	Prueba FL3
Descripción	Comprueba la correcta obtención de los links de las fotos existentes en Flickr para el usuario
Entrada	Token (OAuth) y lista de FlickrPhoto de las que queremos obtener el link
Salida esperada	Lista de FlickrPhoto con la información anterior además de links de descarga
Resultado	Exito

4.3. Dropbox

Estas pruebas son muy sencillas, ya que consisten en ver que los métodos implementados son capaces de obtener archivos (testGetAll), crear archivos en Dropbox (testInsert) y descargarlos (testDownload).

4.3.1. Test GetAll

El primero de ellos consiste en hacer una llamada al método getFolder, el cual nos dará la información de los archivos y las carpetas que pertenezcan a la ruta que les pasamos como id (vacía en este caso). Después lo que hacemos es comprobar que el archivo resultante no es nulo, y mostraremos todas las carpetas y documentos que guarda mediante un bucle for.

4.3.2. Test Insert

En este test lo que hacemos es lo siguiente. Primero, creamos un archivo el cual queremos subir a Dropbox, después le indicamos la ruta en la que se debe crear y con que nombre (en la ruta id con el nombre Fichero_a_insertar). Una vez creado el archivo, creamos una cadena de texto el cual será el contenido de ese archivo que vamos a subir y hacemos una llamada al método insertFile pasándole como parámetros la ruta, el archivo a subir y su contenido. Finalmente, comprovamos que el fichero no es nulo y mostramos por pantalla el id y la ruta del archivo.

4.3.3. TestDownload

Sin duda es el test mas simple de todos. Llamamos al método downloadFile y le pasamos la ruta del archivo a descargar. Este método devuelte una cadena de texto, la cual es la url del archivo. Comprobamos que la cadena de texto no es nula y la mostramos por pantalla.

4.4. Google Drive

ID	Prueba GD1
Descripción	Comprueba el correcto funcionamiento al solicitar la lista de archivos a la cuenta de Google Drive logeada.
Entrada	Token (OAuth)
Salida esperada	Lista de archivos del Google Drive del usuario
Resultado	Éxito

ID	Prueba GD2
Descripción	Comprueba el correcto funcionamiento al intentar descargar un archivo de la cuenta de Google Drive logeada.
Entrada	Token(OAuth) e índice del archivo a descargar en la lista
Salida esperada	URL para descargar dicho archivo
Resultado	Éxito

5. Api Rest

PhotoLoad también integra una Api Rest para que los usuarios más experimentados puedan intercambiar comentarios entre ellos. Dicha api esta alojada en la siguiente url: https://etsiiphotoload.appspot.com/api.

Las peticiones que se podrán realizar a la api vienen recogidas en la siguiente tabla:

HTTP	Plantilla URI	Descripción
GET	/comments	Obtiene una lista con todos los comentarios.
GET	/comments/{index}	Obtiene el comentario con dicho índice.
POST	/comments	Añade un nuevo comentario. Si se añade correctamente devuelve un mensaje “201 Uploaded”. Si ocurre algún error devuelve “400 Error”.
PUT	/comments/edit/{index}	Obtienes el comentario con dicho índice para que lo modifiques y se vuelva a publicar. Si se realiza correctamente, debe devolver un “204 No Content”. Si el comentario, debe devolver un “404 Not Found”.
DELETE	/comments/delete/{index}	Borra el comentario con dicho índice. Si se realiza correctamente, debe devolver un “204 No Content”. Si el comentario no existe debe devolver un “404 Not Found”.

Ahora pondremos algunos ejemplos de respuestas en formato JSON (el cual ofrece nuestra API):

GET: https://etsiiphotoload.appspot.com/api/comments

```
-0: {
        contenido:  "Que gran aplicacion!"
}
-1: {
        contenido:  " Por  que no quedamos el sabado?"
}
```

GET: https://etsiiphotoload.appspot.com/api/comments{id}

```
{
contenido:  "Que gran aplicacion!"
}
```

6. Historial de Versiones

Este apartado muestra el historial de versiones mediante la siguiente tabla:

Fecha	Versión	Detalles	Participantes
22-03-2015	1.0	Incluye introducción, prototipos de las interfaces de usuario y diagramas UML de componentes y despliegue.	López Ruiz, Manuel Francisco Rodríguez Caballero, Miguel Tinoco Castillo, David
10-05-2015	2.0	· **Mejora de la versión 1.0**: Cambios en la Figura 7 y otros cambios menores. · **Cambio de APIs** reflejado en el apartado 1.1 y todos los cambios que conlleva respecto al entregable anterior. También se ha explicado el tipo de servicio usado con cada api. · **Ampliación** del apartado 8 y del apartado 1.2 con los problemas encontrados en esta entrega .	López Ruiz, Manuel Francisco Rodríguez Caballero, Miguel Tinoco Castillo, David
31-05-2015	3.0	· **Cambio en las vistas** reflejado en el apartado 2 y en el 3.3. · **Ampliación** del apartado 1.2 con los problemas encontrados en esta entrega . · **Creado** el apartado 4 y 5.	López Ruiz, Manuel Francisco Rodríguez Caballero, Miguel Tinoco Castillo, David

7. Autoevaluación

Este apartado muestra, cuantitativamente, la dedicación prestada al proyecto por cada integrante del grupo. Para ello se usará la siguiente tabla la cual muestra el porcentaje de colaboración siendo un 0 % la no participación en el entregable y el 100 % la total implicación en él.

Alumno	Entregable 1	Entregable 2	Entregable 3
López Ruiz, Manuel Francisco	100 %	100 %	100 %
Rodríguez Caballero, Miguel	100 %	100 %	100 %
Tinoco Castillo, David	100 %	100 %	100 %

8. Herramientas utilizadas

En este apartado se enumerarán las herramientas usadas para realizar el proyecto. También se especificará en que sistema operativo se han usado.

- Elaboración del documento PDF: LaTeXmediante la herramienta TexStudio en ArchLinux.
- Elaboración de Diagrama de Componentes y de Despliegue: UMLet en ArchLinux.
- Despliegue online de la aplicación: AppEngine de Google.
- Herramientas utilizadas para programar la aplicación:
  - Eclipse Java Developer en Windows y ArchLinux.
  - Plugin para Eclipse de AppEngine provisto por Google en Windows y ArchLinux.
  - Notepad++ (en Windows), Vim (en ArchLinux) y Gedit (en Ubuntu y ArchLinux) para editar códigos en otros lenguajes distintos a Java.
- Control de versiones del código: GitHub en Windows, Ubuntu y ArchLinux.
- Elaboración de mockups: Balsamiq online (webdemo.balsamiq.com).
- Herramientas utilizadas para la gestión y construcción de proyectos:
  - Maven: para la compilación de librerías Java publicadas en GitHub.
- Librería para implementar OAuth1: Scribe la cual solo está presente en el servidor.
- Librería para implementar OAuth2: Gwt-oauth2 la cual se implementa a nivel de cliente y servidor.

9. Lenguajes utilizados

En este apartado se enumerarán los lenguajes que hemos necesitado para realizar el proyecto. Algunos se aprendieron con anterioridad, otros en la asignatura y otros tuvimos que aprenderlos para poder seguir adelante.

- LaTeX(Para realizar los entregables).
- Java
- HTML
- CSS