UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Uso de las API de Dropbox y Google Drive para la integración de datos en la nube

Autor: Domingo Casarrubio Prieto

Tutor: Miguel Sánchez López

Grado en Ingeniería Informática

Curso 2013/2014

Resumen

El principal objetivo de este proyecto es desarrollar un sistema (API) que sea capaz de conectar varios servicios de almacenamiento en la nube (Dropbox, Google Drive…) para facilitar la administración de los archivos conjuntamente en vez de por separado para cada servicio.

La realización del sistema se llevará a cabo con el *framework* CakePHP. La principal tarea que realizará es la autenticación de los usuarios que deseen utilizar el sistema. Además, guardará los *tokens* necesarios de cada usuario para realizar las llamadas a la API de cada servicio que conecten.

Para la comunicación entre la API y el cliente se seguirá un derivado propio del protocolo OAuth, que servirá para poder autenticar a los usuarios en los servicios de terceros sin necesidad de guardar su usuario/contraseña. La única información que guardará la aplicación será el email con el que se registren los usuarios y el *token* que cada servicio asigne para realizar las llamadas a las APIs.

Palabras clave: Nube, API, Google, Drive, Dropbox, Almacenamiento, Integración, Archivos

Brief

The goal of this project is to develop a system (API) capable of connecting different cloud storage services (Dropbox, Google Drive...) to make easier the management of all the files together instead of using each service separately.

The development of the system will be done with CakePHP. The main task the system will have to do is to authenticate the users that desire to use the system, at the same time, it will save the tokens of each user needed to call each service API connected.

For the communication between the API and the client, OAuth protocol will be used, it will be used to authenticate the users in the services without the need to save their username/password. The only information that will be saved is the email that the users use to register with the system and the token of each service connected.

Keywords: Cloud, API, Google, Drive, Dropbox, Storage, Integration, Files

Índice

Índice de figuras y fotografías

Abreviaturas

API.....Application Programming Interface

HTTP .....Hypertext Transfer Protocol

HTTPS.....Hypertext Transfer Protocol Secure

JSON.....JavaScript Object Notation

RESTfull .....Representational State Transfer

SDK .....Software Developer Kit

SSL.....Secure Sockets Layer

Capítulo 1. Introducción

1.1. Motivación

En la era en la que vivimos cada vez más se hace un uso cotidiano extensivo de dispositivos móviles, tales como *smartphones*, *smartwatchs*, *tablets*… etc. Ya no solo se dispone de un ordenador de sobremesa con toda la información almacenada, si no que se quiere disponer de ella en cualquier momento y lugar.

A raíz de este problema surgieron los servicios de almacenamiento en la nube tales como Google Drive, Dropbox, Shared, Copy, etc, cuyo objetivo principal es disponer de todos tus archivos estés donde estés, no importa el lugar, ni el dispositivo, con tener acceso a internet podrás utilizarlos.

El inconveniente, es que no todos los servicios son gratuitos, y los que son ofrecen muy poco espacio de almacenamiento - Dropbox tan solo ofrece 2GB - lo que conlleva a utilizar varios servicios. El inconveniente es tener todos los archivos esparcidos entre varios servicios que no disponen de ninguna comunicación entre ellos.

La comunicación con los diferentes servicios se consigue con la utilización de las API (*Application Programming Interface*) que ofrece cada uno de ellos de manera distinta y con su propio formato de entrada/salida. Sin embargo, algo en lo que coinciden todas las API es en el uso del protocolo HTTP (*Hypertext Transfer Protocol*) con una arquitectura RESTfull (*Representational State Transfer*) y un estándar de intercomunicación JSON (*JavaScript Object Notation*).

En conclusión, estas serán las claves para realizar una integración de los servicios de almacenamiento en la nube.

1.2. Objetivos

El principal objetivo de este proyecto es conseguir un sistema agregador de servicios de almacenamiento en la nube para disponer en una única API una integración totalmente transparente.

El sistema (de ahora en adelante llamado UniCloud API) no debe tener ni mantener nunca estado, es decir, en cada llamada realizada a UniCloud API se realizará una autenticación, se realizará la acción, devolverá una respuesta, y finalizará la comunicación.

Añadir nuevos sistemas de servicios de almacenamiento debe ser sencillo y no debe modificar en absoluto el funcionamiento de los servicios que estén en ese momento implementados.

Todos los servicios deberán ser uniformemente integrados: todas las acciones que se desarrollen deberán estar disponibles en todos los servicios que se quieran integrar.

1.3. Organización de la memoria

La memoria constará de los siguientes capítulos:

- **Capítulo 1. Introducción:** Breve introducción al problema que se desea solucionar así como la motivación para el mismo además se mencionarán los objetivos que se han considerado más importantes para llegar a una solución correcta.
- **Capítulo 2. Servicios de almacenamiento en la nube:** Descripción y definición de los servicios de almacenamiento en la nube así como de los servicios más populares. Se diferenciarán entre los distintos tipos de servicios y se realizará la selección de los mejores servicios para el desarrollo de este proyecto.
- **Capítulo 3. Análisis de Google Drive:** Introducción al servicio así como sus características. Se analizará su API y destacaran sus ventajas e inconvenientes para la integración.
- **Capítulo 4. Análisis de Dropbox:** Introducción al servicio así como sus características. Se analizará su API y destacaran sus ventajas e inconvenientes para la integración.
- **Capítulo 5. Análisis de UniCloud:** Una vez realizado el análisis de los servicios a integrar en UniCloud en este capítulo se realizará el análisis de las características necesarias para la integración de ambos servicios.
- **Capítulo 6. Diseño y Desarrollo de UniCloud:** En el diseño del proyecto se hablará de las distintas tecnologías usadas y la arquitectura elegida para el sistema. En el desarrollo se describirá como se ha realizado la integración entre ambos servicios así como la preparación para la integración de futuros servicios adicionales.
- **Capítulo 7. Documentación de UniCloud:** Documentación técnica del proyecto. Descripción de cada una de las posibles llamadas a UniCloud y forma de uso.
- **Capítulo 8. Conclusiones:** Análisis y conclusión del trabajado realizado.

Capítulo 2. Servicios de almacenamiento en la nube

2.1. Introducción

Antiguamente se relacionaba el almacenamiento en la nube con una práctica de empresas, con grandes necesidades de espacio, sin embargo existen servicios que pueden ser utilizados como un usuario privado, algunos de ellos gratuitos (hasta cierta cantidad de datos), y que pueden ser útiles para respaldar la información, tenerla accesible desde cualquier lugar o, simplemente, para compartir archivos, como fotografías por ejemplo.

El almacenamiento en la nube nace debido a la necesidad cada vez mas creciente de disponer de los archivos en cualquier lugar, de hecho, con el crecimiento de los dispositivos móviles este factor se acentúa. De esta forma, las compañías que ofrecen almacenamiento en la nube como servicio disponen de grandes centros de procesamiento de datos y los ponen a disposición de los usuarios de forma gratuita o alquilando la capacidad de almacenamiento necesaria.

El almacenamiento en la nube es un modelo de servicio en el cual los datos se administran y se respaldan de forma remota, típicamente en grandes conjuntos de servidores donde se busca mantener las ventajas principales de un sistema en la nube, como pueden ser: elasticidad en el espacio que se puede utilizar y que sea un servicio por demanda, es decir que exista la capacidad de contratar 5GB, 10GB, 30GB o 100GB pero no intermedios.

2.2. Tipos de almacenamiento en la nube

Existen tres tipos de servicio de almacenamiento en la nube: público, orientado a cualquier persona, privado, orientado a empresas, e híbrido, una combinación de ambos.

- **Público:** Servicio en el que se requiere poco control administrativo y que se puede acceder en linea por cualquier persona que este autorizada. El almacenamiento en la nube pública utiliza un mismo conjunto de hardware para almacenar la información de varias personas, con medidas de seguridad y espacios virtuales para que cada usuario pueda ver específicamente la información que le corresponde. Este servicio es alojado externamente y se puede acceder mediante internet. Estos son algunos de los mas conocidos.

  - **Dropbox**, uno de los mas populares ya que fue el primero en hacer famoso el uso del almacenamiento en la nube.

  - **Google Drive**, servicio de almacenamiento de Google, originalmente denominado Google Docs.

- **Copy**, mucho mas reciente que los anteriores y que utiliza un método más agresivo para captar usuarios ofreciendo grandes cantidades sin coste.

- **Mega**, servicio creado por Kit Dotcom tras el cierre de Megaupload.

- **Privado:** Normalmente usado por empresas, es un sistema diseñado específicamente para cubrir las necesidades de la empresa o persona. En este modelo la empresa tiene el control administrativo y por lo tanto le es posible diseñar y operar el sistema de acuerdo a sus necesidades.

- **Híbrido:** Como su nombre indica, ofrece una combinación de nube pública y privada, de tal forma que les es posible a los usuarios el personalizar las funciones y que las aplicaciones se adapten mejor a sus necesidades. Un ejemplo típico de esta nube es que la información importante este alojada en la nube privada mientras que la información menos sensible se encuentre en la nube pública.

2.3. Servicios escogidos para el desarrollo de UniCloud

La primera característica que debe tener un servicio para poder ser integrado es que disponga de una API. Este aspecto resulta esencial ya que, sin dicha API, no se puede acceder a la información del usuario a través de terceros.

La segunda característica a tener en cuenta es que sea un servicio popular, ya que el hecho de integrar primero los servicios populares ayudará a atraer a usuarios que utilicen dichos servicios.

Después de analizar los siguientes servicios - Dropbox, Google Drive, Copy, Shared, Mega, Box - se llega a la conclusión de que Dropbox y Google Drive son los mejores candidatos para ser integrados.

Dropbox dispone de una gran popularidad debido a que fue el primero en extender el uso, además de que disponer de una API sencilla, actualizada y bien documentada.

Google Drive es el segundo más extendido después de Dropbox ya que pertenece al *gigante* de Google y además de alojar archivos permite también editarlos o crear nuevos directamente desde su interfaz web.

Capítulo 3. Análisis de Google Drive

3.1. Introducción

Google Drive es el servicio de almacenamiento en la nube que ofrece Google lanzado el 24 de Abril de 2012 y que fue el sucesor de Google Docs. Para disponer de él, tan solo hará falta acceder a la cuenta personal de Google y dentro de las aplicaciones disponibles se podrá encontrar Google Drive.

El espacio que ofrece inicialmente es de 15 GB compartidos entre todos los servicios de Google como Gmail y Google+ Photos. Además, Google Drive no solo ofrece espacio para almacenar archivos, si no que, también dispone de un procesador de texto, hojas de calculo, presentaciones y previsualización de archivos.

Respecto al espacio, cabe mencionar que todos los archivos en formato nativo de Google (.gdoc, .gslides, .gsheet), las fotos de una resolución menor a 2048x2048 y los videos que tengan una duración menor de 15 minutos no contarán de cara a los 15 GB que Google ofrece gratuitamente.

3.2. Características

Google ofrece un plan de precios para poder disponer de la capacidad de almacenamiento que mas se ajuste a las necesidades del usuario partiendo de 15 GB (gratis) hasta 30 TB ($299.99 al mes).

Figura 1: Plan de precios y almacenamiento en Google Drive

Almacenamiento	Precio
15 GB	Gratis
100 GB	$1.99 / mes
1 TB	$9.99 / mes
10 TB	$99.99 / mes
20 TB	$199.99 / mes
30 TB	$299.99 / mes

Fuente: Google Drive

Google Drive dispone de tres tipos de aplicaciones para acceder al contenido: web, escritorio y móvil.

En la aplicación web es posible crear, ver, editar, copiar, mover, borrar y subir cualquier tipo de archivo mientras tengamos conexión, También se puede seguir trabajando con los archivos

aunque no se disponga de conexión en ese momento, gracias a la implementación nativa que tiene en Google Chrome.

El listado de archivos que permite**n** ver la aplicación web es el siguiente:

- Google Docs
- Google Sheets
- Google Slides
- Google Forms
- Google Drawings
- Imagenes (.JPEG, .PNG, .GIF, .TIFF, .BMP)
- Videos (WebM, .MPEG4, 3GPP, .MOV, .AVI, .MPEGPS, .WMV, .FLV, .OGG)
- Texto (.TXT)
- Código (.CSS, .HTML, .PHP, .C, .CPP, .H, .HPP, .JS)
- Microsoft Word (.DOC and .DOCX)
- Microsoft Excel (.XLS and .XLSX)
- Microsoft PowerPoint (.PPT and .PPTX)
- Adobe Portable Document Format (.PDF)
- Apple Pages (.PAGES)
- Adobe Illustrator (.AI)
- Adobe Photoshop (.PSD)
- Autodesk AutoCad (.DXF)
- Scalable Vector Graphics (.SVG)
- PostScript (.EPS, .PS)
- Fuentes (.TTF, .OTF)
- XML Paper Specification (.XPS)
- Archivos comprimidos (.ZIP and .RAR)
- .MTS Files

Tras descargar e instalar la aplicación de escritorio, ésta creará una carpeta en el sistema, en la cual todo el contenido estará sincronizado con Google Drive, siendo capaz de disponer de cualquier archivo que se almacene en dicha carpeta en la nube.

Sin embargo, la aplicación para dispositivos móviles funciona de forma distinta ya que, en vez de descargar y mantener en todo momento sincronizados los archivos entre el dispositivo/nube, ésta muestra simplemente el contenido sin llegar a realizar en ningún momento la descarga de los archivos. Debido al poco espacio del que disponen en la actualidad los dispositivos móviles no sería posible almacenar todos los archivos.

3.3. API

La API de Google Drive se encuentra actualmente en su versión 2.0 con la última modificación realizada el 4 de abril de 2014. La documentación está dividida en trece bloques principales: *files, about, changes, children, parents, permissions, revisions, apps, comments, replies, properties, channels y realtime .* Dentro de cada uno de ellos están las distintas acciones que se pueden realizar respecto a su bloque.

La gestión de los archivos en Google Drive se realiza mediante punteros, es decir,no sigue la tradicional estructura jerárquica en árbol, donde cada archivo tiene un único padre y varios hijos. Los archivos pueden tener varios padres lo que significa que un mismo archivo puede estar bajo varias carpetas distintas sin ser una copia del mismo.

La comunicación con la API se realiza siguiendo el protocolo de llamadas o peticiones HTTP usando los *endpoints* especificados en la documentación dependiendo de la acción que se necesite realizar. Cada acción requiere de unas cabeceras y cuerpo distintos, usando el formato JSON como lenguaje común.

Para poder realizar peticiones se necesita de una *API KEY, client id* y *client secret* que se puede obtener dándose de alta como desarrollador en Google.

3.3.1. Autenticación

La autenticación en las llamadas se realiza con el protocolo OAuth 2.0, siguiendo cuatro pasos:

1. Obtener las credenciales de OAuth 2.0 de la consola de desarrolladores de Google, lo que incluye el *client id* y *client secret* ambas conocidas por Google y la aplicación. Los valores del par pueden cambiar dependiendo del tipo de aplicación que se esté desarrollando, una aplicación JavaScript no necesita el *client secret* pero una aplicación web si que lo necesita.

2. Obtener un *access token* de los servidores de autenticación de Google, antes de que la aplicación pueda acceder a la información privada usando la API, por lo que se necesita obtener un *access token* que le conceda acceso a la API. Un *access token* dispone de un alcance que controla las acciones que se pueden realizar. Para obtenerlo, se redirige ~~a~~ al usuario a una página de Google, donde se especifica la aplicación y los permisos que se están solicitando (página de consentimiento del usuario). En el caso de que éste autorice a la aplicación, la aplicación recibirá una respuesta de la API con un código de uso temporal que podrá intercambiar por un *access token*.

3. Realizar la autenticación con los servidores de Google, así se incluirá el *access token* como una cabecera HTTP

Figura 2: Cabecera *Authorization* en una petición HTTP

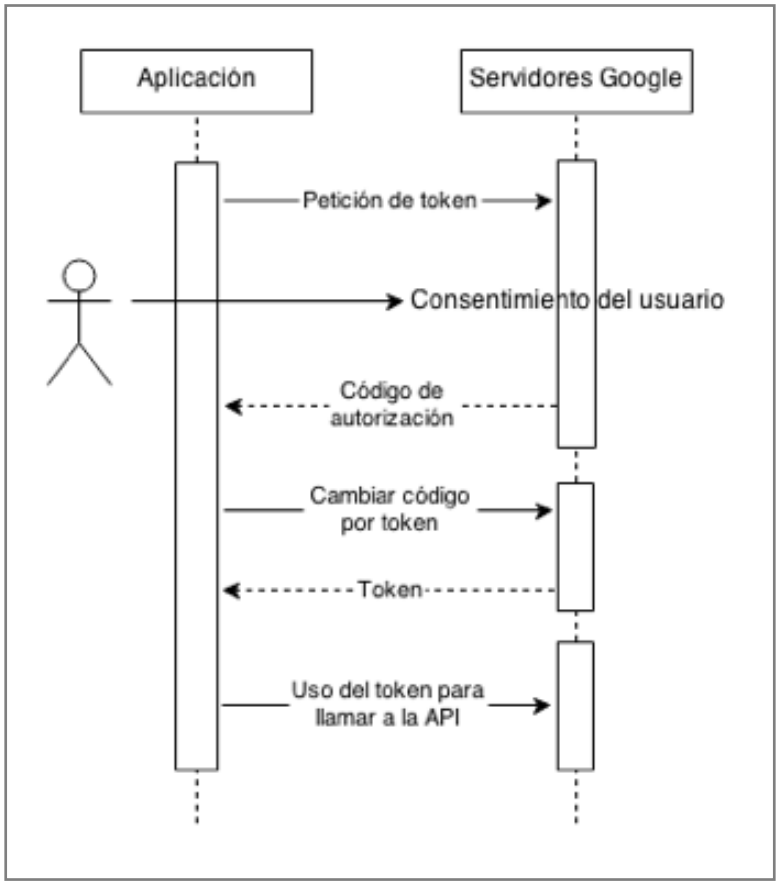
```
POST https://www.googleapis.com/drive/v2/files?key={YOUR_API_KEY}

Content-Type:  application/json
Authorization:  Bearer ya29.KgDpr1AGZxXeFTAAAABmol2lGSsFdy_v3ErFtv-
n7TZjddmP3EB__ciK4ePUzzGTnbMGK8OASR-Xggp_6_o
X-JavaScript-User-Agent:  Google APIs Explorer
```

Fuente: Elaboración propia

4. Renovar el *access token* si fuera necesario ya que los *access token* tienen una vida limitada. Si el *access token* estuviera caducado, Google respondería con una código HTTP que debería ser capturado para solicitar un nuevo *access token*.

Figura 3: Proceso de autenticación



Fuente: Elaboración propia

3.3.2. Listar archivos

Google Drive no ofrece una forma de navegar por los archivos típica en forma de árbol (todo son punteros) por lo que el parámetro 'q' tendrá un papel muy importante. Para cada búsqueda que se quiera realizar sobre los archivos se deberá empezar desde la raíz, y concatenar varias peticiones variando en cada una de ellas el parámetro 'q' ya que no podemos obtener el identificador de una carpeta o archivo de otra forma.

Esto es un factor a tener en cuenta para cada una de las acciones que se quiera realizar con Google Drive por que para realizar cualquier acción sobre un archivo o carpeta se necesita su identificador único. La única forma de obtenerlo es recorriendo todos los archivos hasta llegar al deseado.

Recorrer los punteros desde la raíz sera la base para todas las demás acciones que se realicen sobre la API de Google Drive.

Petición HTTP

```
GET https://www.googleapis.com/drive/v2/files
```

Parámetros

Nombre del parámetro	Tipo	Descripción
maxResults	integer	Número máximo de resultados que devolverá la petición. Valores aceptados entre 0 y 1000 incluidos. (Valor por defecto: 100)
pageToken	string	*Token* para la paginación de los archivos.
q	string	Cadena para realizar búsquedas o filtrados en los resultados.

Cuerpo

No se suministra un cuerpo para esta petición.

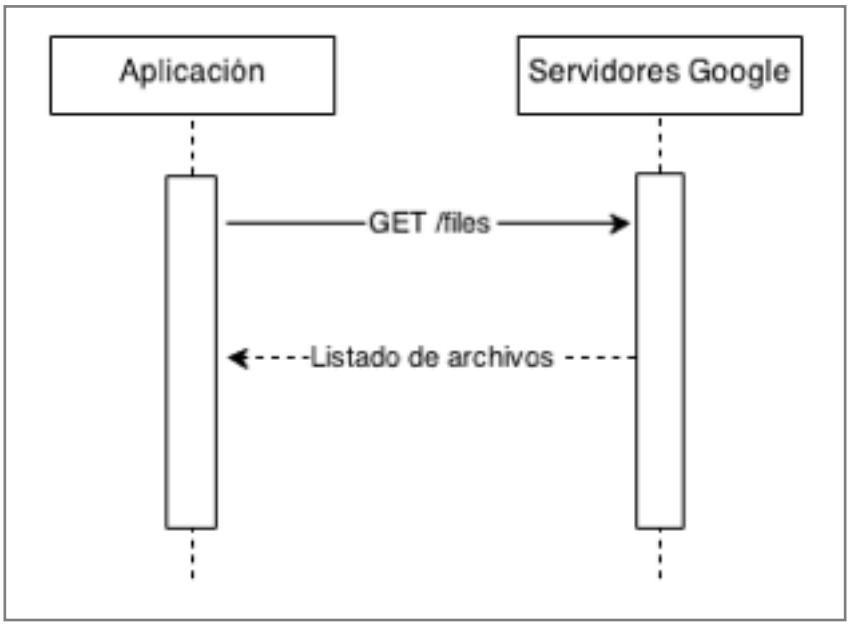
Respuesta

```
{
  "kind": "drive#fileList",
  "etag": etag,
  "selfLink": string,
  "nextPageToken": string,
  "nextLink": string,
  "items": [
    files Resource
  ]
}
```

Propiedad	Tipo	Descripción
kind	string	Tipo de archivo, siempre será drive#fileList.
selfLink	string	Enlace a la lista.
nextPageToken	string	Token para la siguiente paginación de la lista.
nextLink	string	Enlace a la siguiente lista. (En caso de paginación)
items[]	list	La lista con los archivos.

Esquema

Figura 4: Listar archivos en Google Drive



Fuente: Elaboración propia

3.3.3. Copiar archivos

Cuando se realiza una copia de un archivo Google esta duplicando el archivo que se le indique, dando la posibilidad de cambiar sus propiedades especificando en el cuerpo de la petición las propiedades que se deseen modificar.

Petición HTTP

```
POST https://www.googleapis.com/drive/v2/files/{fileId}/copy
```

Parámetros

Nombre del parámetro	Tipo	Descripción
Obligatorios		
fileId	string	Identificador del archivo a copiar.
Opcionales		
convert	boolean	Indica si debe convertirse el formato del archivo al formato Google Doc. (por defecto: false)
ocr	boolean	Indica si debe hacer OCR en los archivos .jpg, .png, .gif o .pdf. (por defecto: false)

Nombre del parámetro	Tipo	Descripción
ocrLanguage	string	Si ocr es true, indica el lenguaje a usar. Valores validos: códigos ISO 639-1
visibility	string	Indica la privacidad del nuevo archivo.

Cuerpo

Se debe indicar un *file Resource* especificado por Google, de la siguiente forma:

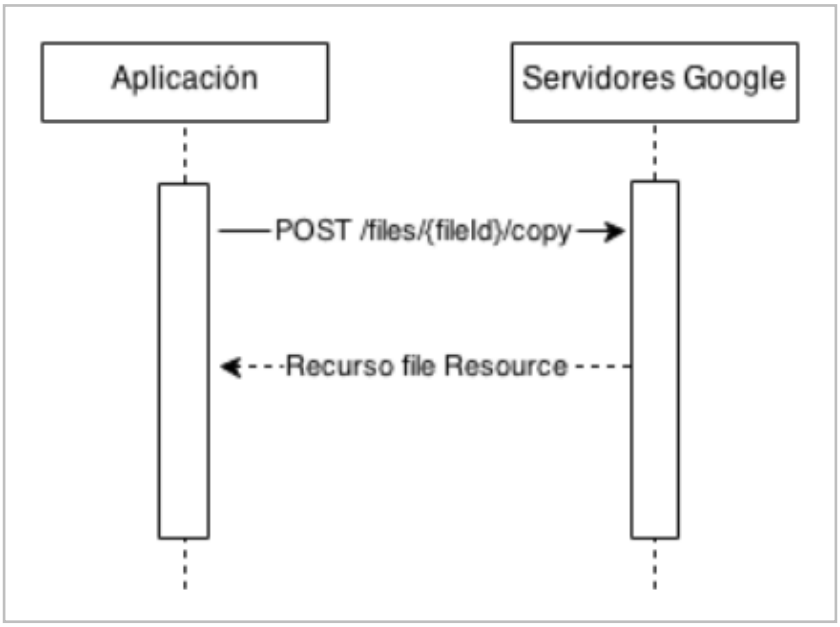
Nombre del parámetro	Tipo	Descripción
Opcionales		
description	string	Descripción breve del archivo.
indexableText.text	string	Texto para indexar el archivo.
labels.restricted	boolean	Indica si se puede descargar el archivo.
labels.starred	boolean	Indica si el archivo ha sido marcado como favorito por el usuario.
labels.trashed	boolean	Indica si el archivo ha sido borrado.
labels.viewed	boolean	Indica si el archivo ha sido visto por el usuario.
lastViewedDateByMe	datetime	La última vez que el usuario vio el archivo.
markedViewedDateByMe	datetime	Fecha especifica marcada por el usuario como visto.
modifiedDate	datetime	La última vez que se modifico el archivo.
parents[]	list	Colección de padres que contienen este archivo.
properties[]	list	Lista de propiedades del archivo.
title	string	Título del archivo.
writersCanShare	boolean	Indica si el archivo puede compartirse con los usuarios con acceso a escritura.

Respuesta

```
{
 "kind": "drive#file",
 "id": string
 "etag": etag
 "selfLink": string
 "alternateLink": string
 "embedLink": string
 "openWithLinks": {
  "889782162350": string},
 “defaultOpenWithLink": string,
 “iconLink": string,
 "title": string,
 “mimeType": string,
 "labels": {
  “starred": boolean,
  "trashed": boolean,
  "restricted": boolean,
  "viewed": boolean
 }}
```

Esquema

Figura 5: Copiar archivo en Google Drive



Fuente: Elaboración propia

3.3.4. Borrar archivos

Existen dos formas diferentes de borrar los archivos, una es usando *delete*, que borrará de forma permanente el archivo, y otra es usar *trash*, que enviará a la papelera dicho archivo pudiendo recuperarlo en cualquier momento.

```
DELETE https://www.googleapis.com/drive/v2/files/{fileId}
```

```
POST https://www.googleapis.com/drive/v2/files/{fileId}/trash
```

Petición HTTP

```
POST https://www.googleapis.com/drive/v2/files/{fileId}/trash
```

Parámetros

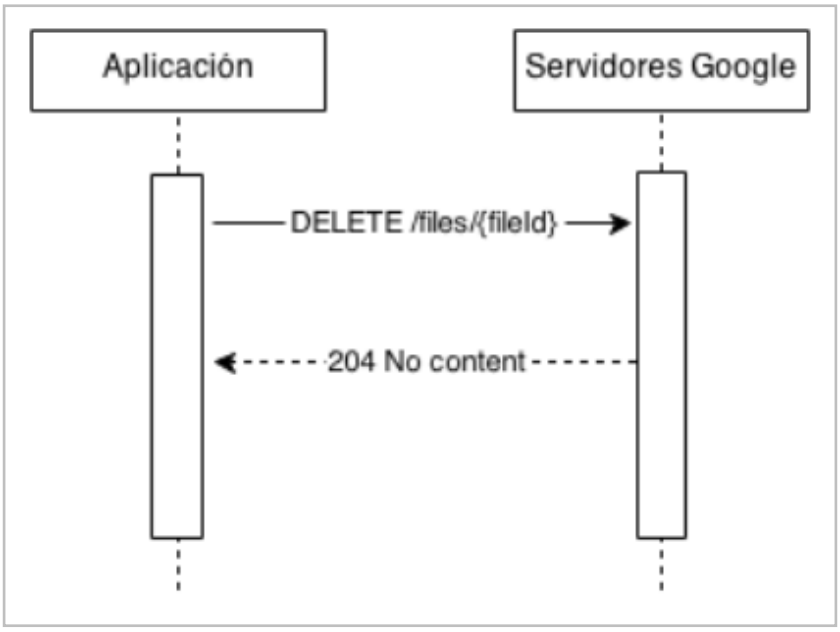
Nombre del parámetro	Tipo	Descripción
Obligatorios		
fileId	string	Identificador del archivo a copiar.

Cuerpo

No se suministra un cuerpo para esta petición.

Esquema

Figura 6: Borrar archivo en Google Drive



Fuente: Elaboración propia

Respuesta

```
 “mimeType": string,
 "labels": {
  "starred": boolean,
  "hidden": boolean,
  "trashed": boolean,
  "restricted": boolean,
  "viewed": boolean
 },
 "createdDate": datetime,
 “modifiedDate”: datetime,
 "modifiedByMeDate": datetime,
 "lastViewedByMeDate": datetime,
 “version”: double,
 "parents": [
  {
   files[] Resource
  }
 ],
 "exportLinks": {
   string[]
 },
 "userPermission": {
  "kind": "drive#permission",
  "etag": etag,
  "id": string,
  "selfLink": string,
  "role": string,
  “type": string
 },
 "quotaBytesUsed": double,
 "ownerNames": string[],
 "owners": [
  {
   "kind": "drive#user",
   "displayName": string,
   "picture": {
    “url": string
   },
   "isAuthenticatedUser": boolean,
   "permissionId": string,
   "emailAddress": string
  }
 ],
 "editable": boolean,
 "copyable": boolean,
 "writersCanShare": boolean,
 "shared": boolean,
}
```

Capítulo 4. Análisis de Dropbox

4.1. Introducción

Dropbox es un servicio de almacenamiento de archivos en la nube fundado en 2007 ofreciendo la posibilidad de sincronizar los archivos entre varios dispositivos haciendo parecer que es la misma carpeta independientemente del dispositivo que se este usando.

Inicialmente se dispone de 2 GB gratuitos con la posibilidad de poder aumentarlo realizando una serie de promociones ofrecidas por Dropbox. Por cada persona que se registre en Dropbox usando tu enlace, se añadirán 250 MB para siempre hasta un máximo de 10 GB.

Se puede almacenar cualquier tipo de archivo pudiendo compartirlo a través de un enlace teniendo un máximo de 20 GB de transferencia al día para las cuentas gratuitas y 200 GB para las cuentas Pro y de empresa.

Dropbox se anuncia diciendo que ni si quiera los empleados tienen acceso a los datos guardados (todos los archivos se almacenan mediante el protocolo de cifrado AES-256) aunque en el 2011 se pudo comprobar que un hacker obtuvo acceso a cualquier cuenta lo que puso en duda la seguridad. Aun así, a día de hoy, Dropbox es considerado el servicio más popular para guardar datos en la nube.

4.2. Características

Dropbox ofrece tres planes orientados a distintos usuarios finales partiendo de los 2 GB con la cuenta gratuita hasta 1 TB para las cuentas de empresa.

Figura 7: Plan de precios de Dropbox y almacenamiento

Almacenamiento	Precio
2 GB	Gratis
100 GB	9.99€ / mes
200 GB	19.99€ / mes
500 GB	49.99€ / mes
1 TB	699.99€ / mes / 5 usuarios*

* cada usuario extra añade 99.99€

Fuente: Dropbox

Por otro lado, los clientes oficiales están disponibles para Android, Windows Phone, Blackberry e iOS facilitando el acceso entre dispositivos y plataformas. Como pasaba con Google Drive cabe diferenciar la funcionalidad entre las aplicaciones de escritorio y las aplicaciones móvil. Mientras

que en el escritorio, tendremos todos los archivos almacenados en Dropbox disponibles sin acceso a internet (de forma física en el ordenador) en las aplicaciones móvil no pasa lo mismo, ya que los archivos solo están disponibles con conexión a internet o mediante descarga previa.

Como se explico en el capitulo de Google Drive, esto es debido a causa del poco espacio de almacenamiento en los móviles ya que la mayoría de móviles no suelen tener mas de 8-16 GB de espacio y los archivos almacenados en la nube pueden superarlo fácilmente.

4.3. API

El punto más fuerte de Dropbox es su API y sus distintos SDK (*Software Developer Kit*). Con la explosión de los dispositivos móviles, disponer de una aplicación es prácticamente obligatorio para toda las empresas, y gracias a la API, integrar almacenamiento en la nube es una solución con un coste prácticamente nulo para la empresa.

Cabe destacar que la API de Dropbox es mucho más sencilla de usar en comparación con Google Drive. Acciones como copiar y mover reciben dos parámetros, origen y destino, llevándose a cabo todo el proceso en los servidores de Dropbox y recibiendo una respuesta de confirmación en casi de que se haya realizado satisfactoriamente la acción.

Como pasaba con Google Drive, la comunicación con la API se realiza siguiendo el protocolo de llamadas o peticiones HTTP usando los *endpoints* especificados en la documentación dependiendo de la acción que se necesite realizar. Cada acción requiere de unas cabeceras y cuerpo distintos, usando el formato JSON como lenguaje común.

Dropbox ofrece dos tipos distintos de acceso, *Drop-ins* y *Core API, Drop-ins* es una forma rápida de integrar almacenamiento en la nube con Dropbox, pero con acceso limitado, mientras que *Core API* ofrece un acceso completo siendo mucho más flexible.

Además, Dropbox también ofrece varios SDK para distintos lenguajes, pero para la realización de este proyecto se usará *Core API siguiendo la documentación para HTTP* ya que además de ofrecer acceso completo se ajusta a las necesidades del mismo y ofrece mas versatilidad.

4.3.1. Autenticación

Como pasaba con Google Drive, Dropbox realiza su proceso de autenticación mediante el protocolo OAuth 2.0:

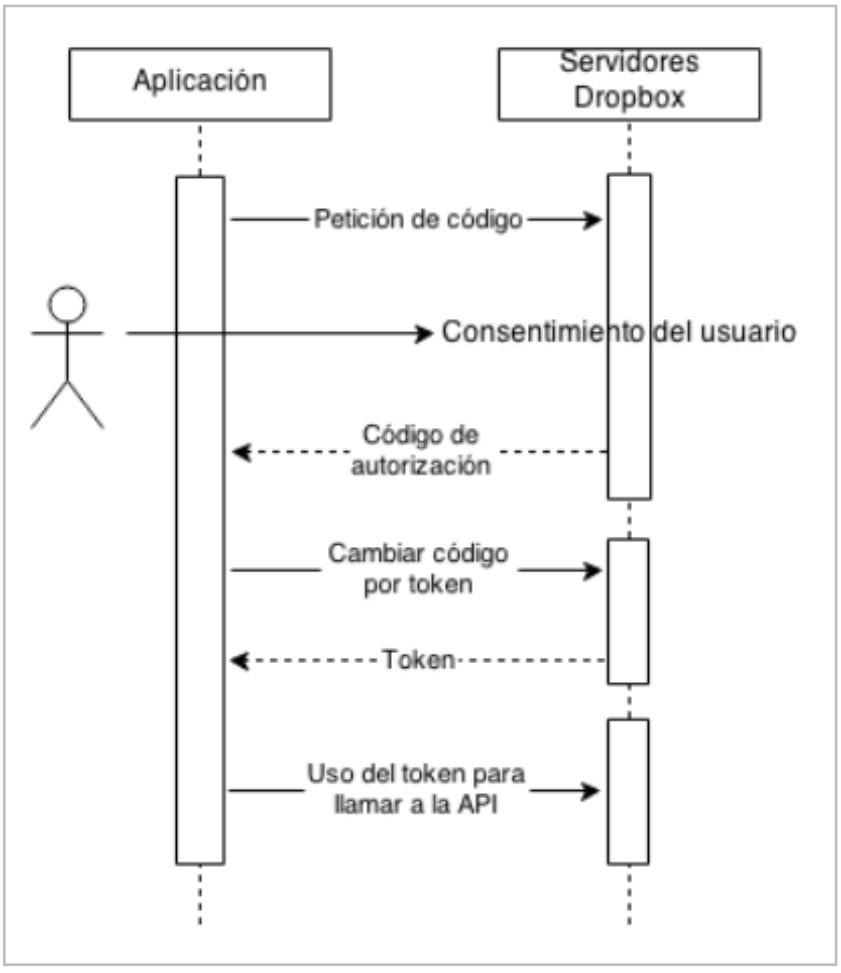
1. Para poder trabajar con la API de Dropbox, hay que disponer de una cuenta registrada - basta con que sea una cuenta gratuita - y acceder a la consola de desarrolladores. Una vez aquí, se registra una nueva aplicación seleccionado *Core API* con acceso completo

como tipo de aplicación, esto dará acceso al *app key* y *app secret* que será necesario para la autenticación más tarde.

2. La primera llamada para realizar la autenticación con la API de Dropbox y que el usuario dé su consentimiento se realiza directamente sobre una re-dirección hacía dropbox.com/1/oauth2/authorize y, una vez el usuario autorice a la aplicación, Dropbox re-direccionará a la URL que esté especificada en la cuenta de desarrollador.

3. Una vez recibido el código por parte de Dropbox confirmando que el usuario ha autorizado a la aplicación, se realizará una llamada a dropbox.com/oauth2/token donde se intercambiará el código recibido por el token final para realizar llamadas a la API de Dropbox.

Figura 8: Proceso de autenticación en Dropbox



Fuente: Elaboración propia

4.3.2. Listar archivos

Dropbox sigue una arquitectura en forma de árbol, donde todos los archivos disponen de un único padre y pueden tener varios hijos, simplificando mucho más los procesos de búsqueda que en Google Drive.

Petición HTTP

```
GET https://www.api-content.dropbox.com/1/files/<root>/<path>
```

Parámetros

Nombre del parámetro	Tipo	Descripción
root	string	La raíz relativa con respecto al parámetro path. Los valores admitidos son sandbox y dropbox.
path	string	La ruta al archivo que se desea listar.
rev	string	Especifica la revisión del archivo que se quiere obtener, por defecto se obtiene la más nueva.

Cuerpo

No se suministra un cuerpo para esta petición.

Respuesta

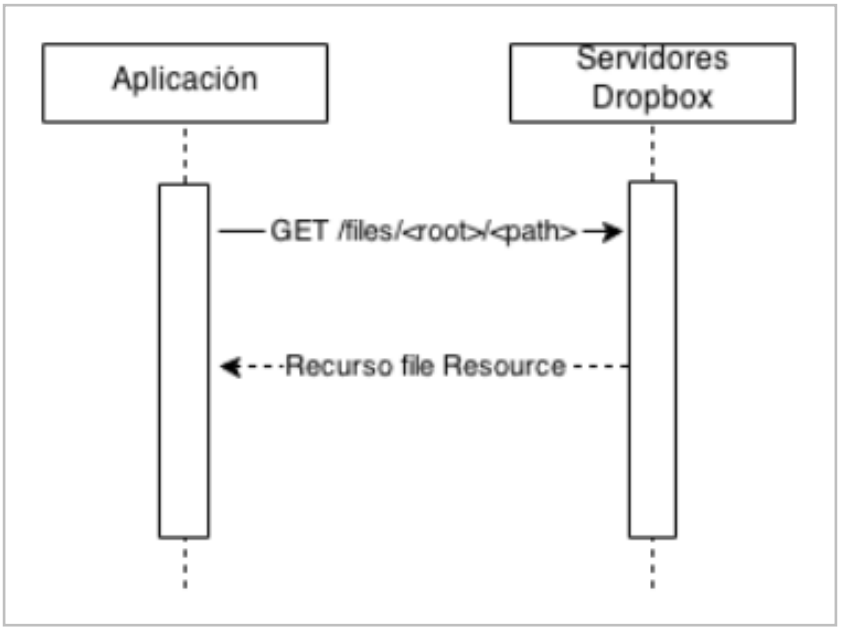
```
{
  "size": string,
  "hash": string,
  "bytes": double,
  “thumb_exists”: boolean,
  "rev": string,
  "modified": string,
  "path": string,
  "is_dir": boolean,
  "icon": string,
  "root": dropbox/sandbox,
  "contents": [
    files Resource
  ]
}
```

Propiedad	Tipo	Descripción
size	string	Tamaño del archivo expresado en forma de cadena.
hash	string	El hash se puede utilizar para saber si han habido cambios, muy parecido a rev.

Propiedad	Tipo	Descripción
bytes	double	Tamaño del archivo.
thumb_exists	boolean	Verdadero si el archivo es una imagen que puede ser convertida a miniatura.
rev	string	Identificador único para la versión actual del archivo.
modified	string	Última vez que fue modificado el archivo.
path	string	Ruta del archivo.
is_dir	boolean	Verdadero si es una carpeta, falso si no.
icon	string	Nombre del icono usado para ilustrar el tipo de archivo en Dropbox.
root	string	La raíz *top-level* del archivo.

Esquema

Figura 9: Listar archivos en Dropbox



Fuente: Elaboración propia

4.3.3. Copiar archivos

A diferencia de Google Drive, el proceso de copiar un archivo en Dropbox es mucho más sencillo, ya que solo será necesario especificar en la petición HTTP el origen del archivo original y el destino de la copia, realizando los servidores de Dropbox todo el proceso del mismo.

Petición HTTP

```
POST https://www.api.dropbox.com/1/fileops/copy
```

Parámetros

Nombre del parámetro	Tipo	Descripción
root	string	La raíz relativa con respecto al parámetro path. Los valores admitidos son sandbox y dropbox.
from_path	string	Origen del archivo a copiar.
to_path	string	Destino de la copia del archivo.

Cuerpo

No se suministra un cuerpo para esta petición.

Respuesta

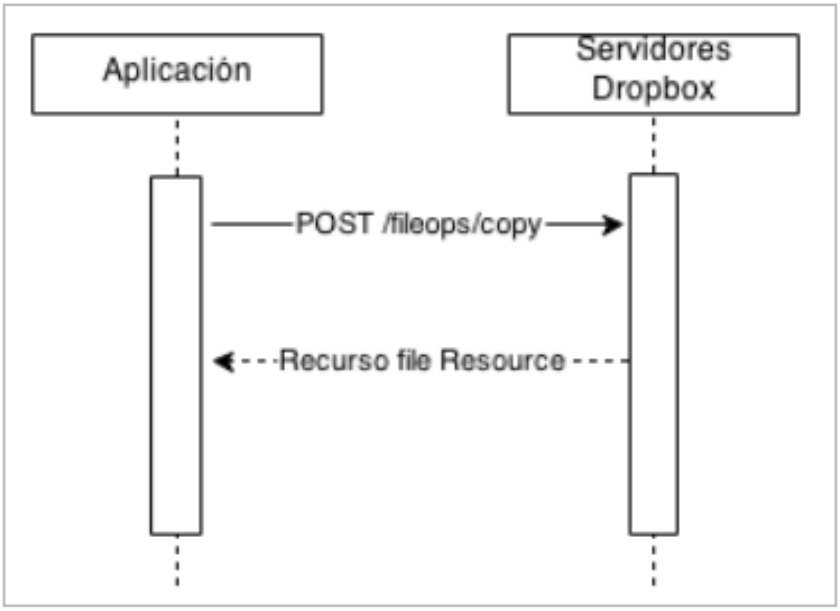
```
{
  "size": string,
  "rev": string,
  "bytes": double,
  “thumb_exists”: boolean,
  "rev": string,
  "modified": string,
  "path": string,
  "is_dir": boolean,
  "icon": string,
  "root": dropbox/sandbox,
  “mime_type": string,
  "revision": int,
}
```

Propiedad	Tipo	Descripción
size	string	Tamaño del archivo expresado en forma de cadena.
hash	string	El hash se puede utilizar para saber si han habido cambios, muy parecido a rev.
bytes	double	Tamaño del archivo.
thumb_exists	boolean	Verdadero si el archivo es una imagen que puede ser convertida a miniatura.
rev	string	Identificador único para la versión actual del archivo.
modified	string	Última vez que fue modificado el archivo.
path	string	Ruta del archivo.
is_dir	boolean	Verdadero si es una carpeta, falso si no.
icon	string	Nombre del icono usado para ilustrar el tipo de archivo en Dropbox.
root	string	La raíz *top-level* del archivo.

Propiedad	Tipo	Descripción
mime_type	string	Tipo del archivo.

Esquema

Figura 10: Copiar archivo en Dropbox



Fuente: Elaboración propia

4.3.4. Borrar archivos

Petición HTTP

```
POST https://www.api.dropbox.com/1/fileops/delete
```

Parámetros

Nombre del parámetro	Tipo	Descripción
root	string	La raíz relativa con respecto al parámetro path. Los valores admitidos son sandbox y dropbox.
path	string	Ruta del archivo a borrar.

Cuerpo

No se suministra un cuerpo para esta petición.

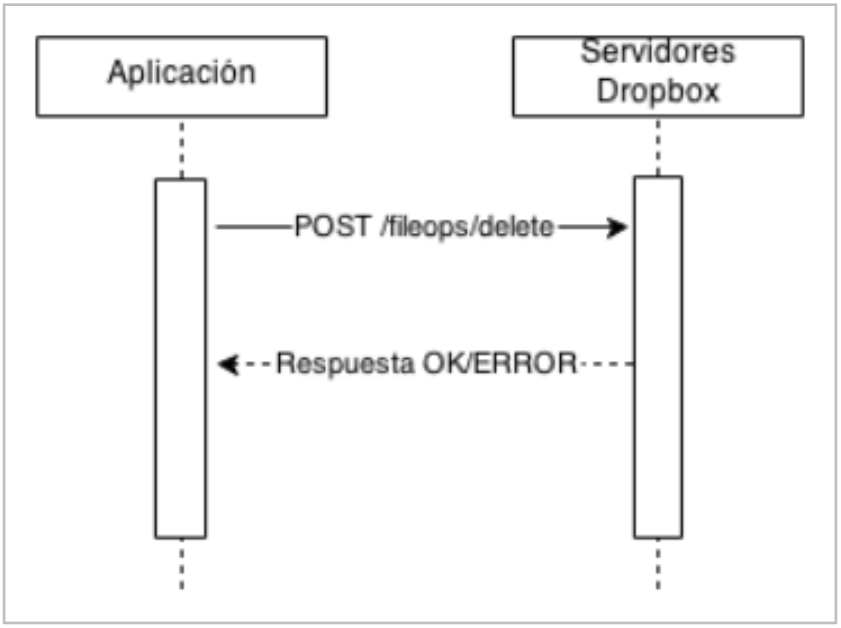
Respuesta

```
{
  "size": string,
  "rev": string,
  "bytes": double,
  “thumb_exists”: boolean,
  "rev": string,
  "modified": string,
  "path": string,
  "is_dir": boolean,
  "icon": string,
  "root": dropbox/sandbox,
  “mime_type": string,
  "revision": int,
}
```

Propiedad	Tipo	Descripción
size	string	Tamaño del archivo expresado en forma de cadena.
hash	string	El hash se puede utilizar para saber si han habido cambios, muy parecido a rev.
bytes	double	Tamaño del archivo.
thumb_exists	boolean	Verdadero si el archivo es una imagen que puede ser convertida a miniatura.
rev	string	Identificador único para la versión actual del archivo.
modified	string	Última vez que fue modificado el archivo.
path	string	Ruta del archivo.
is_dir	boolean	Verdadero si es una carpeta, falso si no.
icon	string	Nombre del icono usado para ilustrar el tipo de archivo en Dropbox.
root	string	La raíz *top-level* del archivo.
mime_type	string	Tipo del archivo.

Esquema

Figura 11: Borrar archivo en Dropbox



Fuente: Elaboración propia

Capítulo 5. Análisis de UniCloud

5.1. Introducción

Después de analizar las API de Google Drive y Dropbox, en este capítulo se analizará como se planteará el desarrollo de UniCloud para integrar ambos servicios en uno uniforme.

Para poder llevar a cabo la uniformidad entre ambos servicios tan solo se podrán integrar las acciones que estén disponibles en ambas API, ya que existen acciones que se pueden realizar en Google Drive y no en Dropbox y viceversa.

El objetivo principal es coger estas acciones que en cada servicio son realizadas de una forma diferente y unificarlas en UniCloud de forma que, tanto para Google Drive como Dropbox, se realicen de la misma manera llevando a cabo las diferencias en la parte del servidor, siendo totalmente transparente para los usuarios.

5.2. Características y API

UniCloud sigue el mismo protocolo y características que Google Drive y Dropbox para realizar las comunicaciones entre usuarios y servidor. Eligiendo de esta forma OAuth en su segunda versión como protocolo para las autenticaciones y JSON como lenguaje común.

La característica más importante es como se interacciona con la API para poder conectar los servicios, ya que, tal y como se observó previamente en ambos casos, tanto en Google Drive como en Dropbox el primer paso de la autorización es redirigir a sus puntos de acceso para que puedan validar al usuario y entregar el código para solicitar el token a la aplicación.

Cuando se desea conectar un servicio a un determinado usuario, UniCloud contestará a la petición con una URL a la que el cliente debe redirigir para poder autorizar y validar el usuario en Google Drive o Dropbox, obteniendo el código para solicitud de token. El cliente entonces enviará en la siguiente petición a UniCloud el código que cambiará por un token y ya estará preparado para realizar cualquier acción.

5.2.1. Autenticación

UniCloud es una API que se alimenta de otras API, por este motivo, el único momento en que se solicita un usuario/contraseña es cuando el usuario que vaya a utilizar UniCloud inicie sesión, obteniendo un token que se utilizará para todas las comunicaciones posteriores. Los tokens tienen un tiempo de vida determinado, siendo actualizado con cada petición que se realice sobre UniCloud.

Como se describía anteriormente, cuando se conecta un servicio, se redirige directamente a Google Drive o Dropbox siendo esto muy importante, ya que el usuario introducirá sus credenciales directamente con estos servicios no siendo necesario que una tercera parte (en este caso UniCloud) necesite saber el usuario/contraseña consiguiendo mayor seguridad y credibilidad por parte de los usuarios.

A continuación se describirán las acciones desde el punto de vista analítico. Para un mayor detalle de su implementación o uso ir al capítulo 7 y 8 respectivamente.

5.2.2. Listar archivos

La integración de varios servicios es el propósito principal de este proyecto. Por ello, cuando se realiza un listado de archivos es muy importante ser lo más transparente posible para el usuario, ya que, si ya dispone de varios servicios conectados, pueda obtener el listado de todos con tan solo realizar una petición a UniCloud.

Además, con motivo de poder diferenciar que archivo pertenece a qué servicio para posibles clientes que quieran poder mostrarlo gráficamente, también se incluye el nombre del servicio al que pertenece en todas las respuestas, así como la posibilidad de poder listar sólo de un servicio determinado.

5.2.3. Copiar archivos

En Dropbox, copiar no tiene mayor dificultad que realizar la llamada a su API con el origen y destino del archivo y ellos se encargan de hacer la copia dando una respuesta con éxito o error en caso de que ocurra.

En Google Drive, copiar tiene mayor dificultad, ya que desde la API de Google Drive cuando se realiza una copia de un archivo éste es duplicado en la carpeta donde esté ubicado actualmente. En el caso en el que la ruta de origen y la ruta de destino no coincidan, el archivo deberá ser movido a su carpeta correspondiente. Como se explicó en el capítulo anterior de Google Drive, navegar por los archivos a través de la API no es tarea fácil, ya que para realizar cualquier acción sobre cualquier archivo se necesita su identificador y la única forma de obtenerlo es navegar desde la raíz hasta el archivo deseado haciendo más costoso el tiempo de respuesta cuanto mayor sea la profundidad del archivo.

5.2.4. Borrar archivos

En este caso, borrar archivos es prácticamente igual en ambas API y bastará una única llamada para realizar el borrado de un archivo, con la única diferencia de que en Google Drive, como se

exponía anteriormente, se necesitará navegar hasta el archivo que se desee borrar desde la raíz para obtener su identificador.

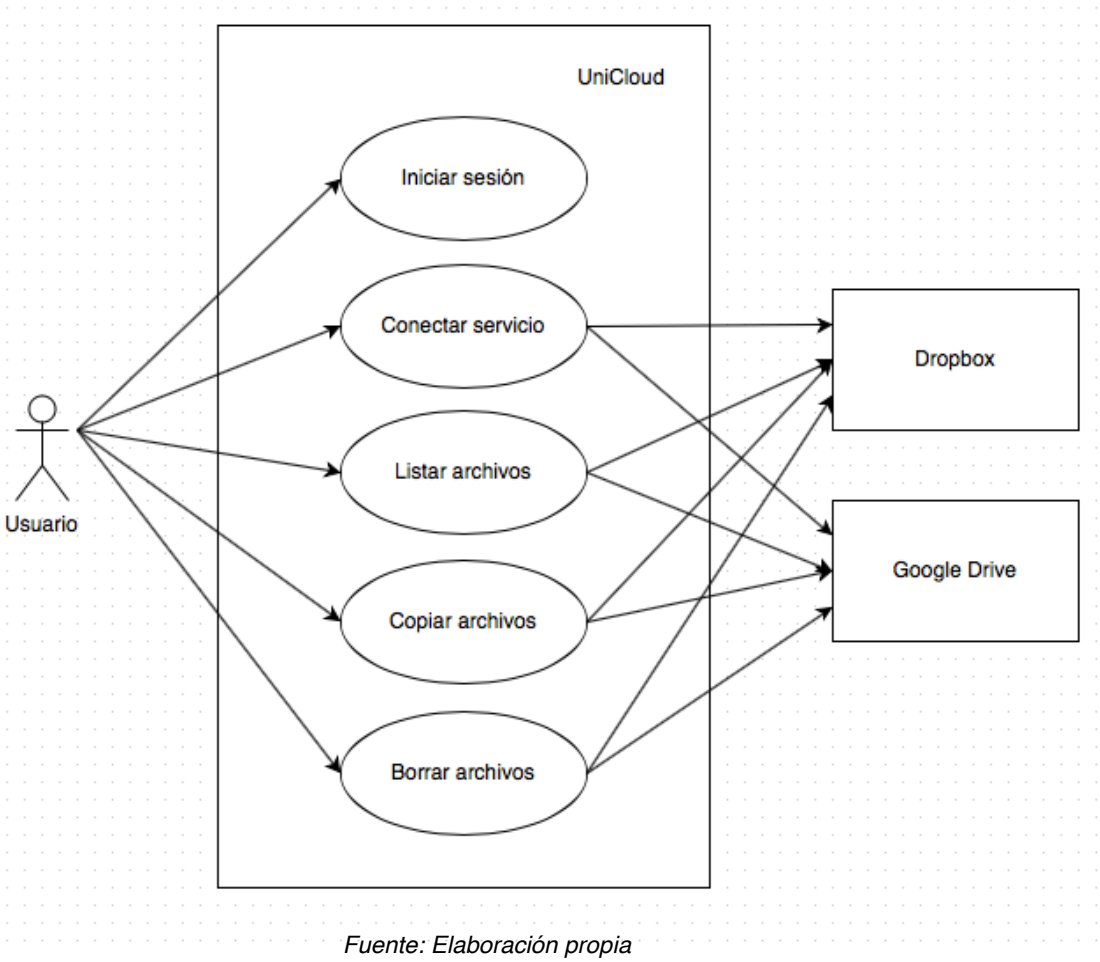
Capítulo 6. Diseño y Desarrollo de UniCloud

6.1. Tecnologías

Las tecnologías elegidas para este proyecto han sido Apache como servidor web, PHP como lenguaje de servidor (CakePHP como *framework* sobre PHP) y MySQL como sistema de gestión de base de datos relacional. La principal razón para escoger estas tecnologías sobre otras es la experiencia con ellas y la facilidad para encontrar respuestas a posibles problemas, ya que son muy populares para proyectos pequeños o que están empezando.

Una de las principales ventajas de Apache es su alta modularidad. Teniendo la posibilidad de desactivar mucho de los módulos que no serán útiles para este proyecto hace que el servidor esté mucho más simplificado y por consecuencia más eficaz.

Figura 12: Arquitectura UniCloud API



Fuente: Elaboración propia

El modulo más importante de Apache para este proyecto es *mod_ssl* ya que tanto la API de Google Drive como Dropbox exigen conexiones seguras mediante SSL (*Secure Sockets Layer*)

que permitirá realizar peticiones HTTPS (*Hypertext Transfer Protocol Secure*). La principal diferencia entre HTTP y HTTPS es que este último añade un canal seguro que dificulte ataques man-in-the-middle. Esto es imprescindible, todas las comunicaciones entre cliente-servidor y servidor-servidor intercambian información en texto plano, cualquier tercer parte que pudiera escuchar una comunicación podría obtener el token que se este usando para la autenticación del usuario y suplantar la identidad de éste pero, gracias a HTTPS resulta más complicado, debido a que las comunicaciones pasarán a estar encriptadas. Otra ventaja de HTTPS es que es necesario el uso de certificados, siendo muy importantes para que los clientes puedan tener confianza sabiendo que están accediendo a quién dice ser.

6.2. Arquitectura

La arquitectura escogida para este proyecto es MVC (*Model-View-Controller*). MVC es un patrón de arquitectura muy extendido y usado en desarrollo web, divide el sistema en tres grandes componentes delegando distintas responsabilidades en cada uno de ellos.

El modelo es el componente central de MVC, en el se especifican las características del problema, por ejemplo, en un coche, la matrícula, la marca, el color, etc, serían las características del modelo (un coche), además, los modelos se comunican directamente con la base de datos y son los encargados de mantener la lógica entre las características que se han definido con los distintos atributos en las tablas correspondientes.

Las vistas se encargan de representar la información de los modelos pero estas no saben nada respecto a la lógica de los mismos. En este sentido se podría decir que las vistas solo reciben la información de los modelos y se la muestran al usuario sin realizar ningún paso intermedio ni lógica.

Si en las vistas el usuario recibe la información, en los controladores es donde el usuario le solicita la información al sistema. Un controlador recibe una solicitud y este se encarga de usar o llamar al modelo correspondiente para que este obtenga la información de la base de datos y a su vez se la pase a la vista para ser mostrada al usuario.

En este proyecto, las relaciones previamente nombradas entre los distintos componentes de la arquitectura son muy importantes, ya que nos permitirán separar distintos elementos del sistema y delegar responsabilidades en cada uno de ellos.

CakePHP, el *framework* utilizado en este proyecto, se desarrolló teniendo en cuenta MVC en mente y como tal da soporté total a dicha arquitectura. En CakePHP, todas las llamadas son procesadas de la misma forma y siguen una serie de pasos antes de llegar al controlador correspondiente. Para la realización de este proyecto algunos de estos pasos han sido omitidos o modificados para obtener como resultado una API RESTful, por lo que en este caso, el último

paso (las vistas) no tendrán una gran importancia más allá de darle formato al JSON que se devolverá.

El primer paso que se realiza cuando UniCloud recibe una petición o llamada es detectar el tipo de petición HTTP para de esta forma poder obtener los parámetros y comprobar si contiene un *token* de acceso, esto es sumamente importante, ya que sin *token* de acceso no se pueden realizar llamadas. El *token* se utiliza para dos cosas, la primera para realizar la autenticación del usuario y la segunda saber de que usuario se trata. Las únicas llamadas que no realizarán dicha comprobación serán las necesarias para el registro de un usuario.

Una vez se ha obtenido el *token* y se ha validado el usuario se le pasará el control al controlador de enrutamiento y esté a su vez al controlador encargado de resolver la petición del usuario.

Para este proyecto se ha definido un modelo para los servicios, de forma que cada servicio nuevo que se necesitará integrar en UniCloud heredaría de este modelo. A su vez, existe un controlador genérico para las acciones realizadas en cualquier servicio y que dependiendo de los servicios que el usuario haya solicitado se haría uso del controlador específico para cada servicio.

Esta última parte es la más importante de todo el proyecto, ya que es donde se cumple el objetivo de la flexibilidad a la hora de añadir nuevos servicios a UniCloud sin qué afecte a los servicios que ya estén funcionando correctamente.

Capítulo 7. Documentación de UniCloud

7.1. Usuario

7.1.1. Registro

Petición HTTP

```
POST https://www.api.unicloud.com/1/user/register
```

Parámetros

Nombre del parámetro	Tipo	Descripción
Obligatorios		
email	string	Email del usuario a registrar.
password	string	Contraseña para la nueva cuenta.
Opcionales		
username	string	Nombre de usuario opcional al correo electrónico.

Respuestas

```
- 200: Successful
- 400: Invalid or missing token
- 401: Empty or invalid parameters
- 406: Duplicated email
```

Ejemplo OK

```
{
    "registered" : true,
    "message" : "User registered successful. "
}
```

Ejemplo ERROR

```
{
    "error" : "Email can not be empty. "
}
```

7.1.3. Confirmación

Petición HTTP

```
GET https://www.api.unicloud.com/1/user/confirm
```

Parámetros

Nombre del parámetro	Tipo	Descripción
Obligatorios		
confirmation_token	string	Token que recibe el usuario en su correo electrónico.

Respuestas

```
- 200: Successful
- 400: Invalid or missing token
- 401: Invalid or missing params
- 406: Error validating the user
```

Ejemplo OK

```
{
    "confirmed" : true,
    "message" : "User confirmed successful."
}
```

Ejemplo ERROR

```
{
    "error" : "Missing confirmation token."
}
```

7.1.3. Recuperación

Petición HTTP

```
POST https://www.api.unicloud.com/1/user/recover
```

Parámetros

Nombre del parámetro	Tipo	Descripción
Obligatorios		
email	string	Email del usuario para recuperar la contraseña.

Respuestas

```
- 200: Successful
- 400: Invalid or missing token
- 401: Invalid or missing params
- 404: User not found
- 406: Error generating the recover token
```

Ejemplo OK

```
{
    "message" : "Email with recover instructions sent
                 successfully."
}
```

Ejemplo ERROR

```
{
    "error" : "Error generating the recover token"
}
```

7.1.4. Login

Petición HTTP

```
GET https://www.api.unicloud.com/1/user/login
```

Parámetros

Nombre del parámetro	Tipo	Descripción
Obligatorios		
email	string	Email del usuario.
password	string	Contraseña del usuario.

Respuestas

```
- 200: Successful
- 400: Invalid or missing token
- 401: Invalid or missing params
- 404: User not found
- 406: Error generating the recover token
```

Ejemplo OK

```
{
    "message" : "Email with recover instructions sent
                 successfully."
}
```

Ejemplo ERROR

```
{
    "error" : "Error generating the recover token"
}
```

7.1.2. Información

Petición HTTP

```
PUT https://www.api.unicloud.com/1/user/info
```

Parámetros

Nombre del parámetro	Tipo	Descripción
Obligatorios		
token	string	Token OAuth usado para la autorización.
Opcionales		
email	string	Email nuevo.
password	string	Contraseña nueva.
username	string	Nombre de usuario nuevo.

Respuestas

```
- 200: Successful
- 400: Invalid or missing token
- 406: Duplicated email/username
```

Ejemplo OK

```
{
    "edited" : true,
    "message" : "User info edited successful."
}
```

Ejemplo ERROR

```
{
    "error" : "Invalid or missing token",
}

{
    "error" : "This  email already exists. "
}
```

7.2. Servicios

7.2.1. Conectar Dropbox

Petición HTTP

```
GET https://www.api.unicloud.com/1/dropbox/connect
```

Parámetros

Nombre del parámetro	Tipo	Descripción
Obligatorios		
token	string	Token OAuth usado para la autorización.

Respuestas

```
- 200: Successful
- 400: Invalid or missing token
```

Ejemplo OK

```
          {
              "url" : "https://www.dropbox.com/1/oauth2/authorize?
client_id=y29k3bcon7w6p3m&response_type=token&redirect_uri=http://
localhost/services/dropbox/authresponse"
          }
```

7.2.2. Conectar Dropbox - Token

Petición HTTP

```
POST https://www.api.unicloud.com/1/dropbox/token
```

Parámetros

Nombre del parámetro	Tipo	Descripción
Obligatorios		
token	string	Token OAuth usado para la autorización.

Nombre del parámetro	Tipo	Descripción
service_code	string	Código recibido cuando el usuario da el consentimiento en la página de autorización de Dropbox.

Respuestas

```
- 200: Successful
- 400: Invalid or missing token
- 403: Failed to retrieve access token
- 406: Failed to save access token
```

Ejemplo OK

```
{
    "message" : "Dropbox connected successful"
}
```

Ejemplo ERROR

```
{
    "message" : "Failed to save access token"
}
```

7.2.3. Conectar Google Drive

Petición HTTP

```
GET https://www.api.unicloud.com/1/gdrive/connect
```

Parámetros

Nombre del parámetro	Tipo	Descripción
Obligatorios		
token	string	Token OAuth usado para la autorización.

Respuestas

```
- 200: Successful
- 400: Invalid or missing token
```

Ejemplo OK

```
{
    "url" : "https://accounts.google.com/o/oauth2/auth?
scope=https://www.googleapis.com/auth/drive&redirect_uri=http%3A%2F
%2Flocalhost%2Fservices%2Fgdrive
%2Fauthresponse&response_type=code&client_id=962554299694-9g68neckc
gj0c6gfkpmob72a2v66nf2r.apps.googleusercontent.com&approval_prompt=
force"
}
```

7.2.4. Conectar Google Drive - Token

Petición HTTP

```
POST https://www.api.unicloud.com/1/dropbox/token
```

Parámetros

Nombre del parámetro	Tipo	Descripción
Obligatorios		
token	string	Token OAuth usado para la autorización.
service_code	string	Código recibido cuando el usuario da el consentimiento en la página de autorización de Dropbox.

Respuestas

```
- 200: Successful
- 400: Invalid or missing token
- 403: Failed to retrieve access token
- 406: Failed to save access token
```

Ejemplo OK

```
{
    "message" : "GDrive connected successful"
}
```

Ejemplo ERROR

```
{
    "message" : "Failed to save access token"
}
```

7.3. Archivos

7.3.1. Listar

Petición HTTP

```
GET https://www.api.unicloud.com/1/files/list
```

Parámetros

Nombre del parámetro	Tipo	Descripción
Obligatorios		
token	string	Token OAuth usado para la autorización.
service	string	Servicio del que se quieren obtener los archivos. Posibles valores: all, dropbox, gdrive.
path	string	Ruta del archivo que se desea obtener.

Respuestas

```
- 200: Successful
- 400: Invalid or missing token
- 401: Invalid or missing params
```

Ejemplo OK (service = all, path = “/“)

```
            {
              "service" : "all",
              "path" : "/",
              "files" : [
                {
                  "service" : "dropbox",
                  "mime" : "folder",
                  "path" : "/Books",
                  "size" : "0 bytes",
                  "modified" : "Thu, 23 Jan 2014 12:23:13 +0000"
                },
                {
                  "service" : "gdrive",
                  "mime" : "folder",
                  "path" : "/Drive prueba”
                  "size" : "0 bytes",,
                  "modified" : "2014-03-20T11:41:12.637Z"
                },
              ]
```

Ejemplo ERROR

```
            {
                "error" : "Invalid path."
            }
```

7.3.2. Ver

Petición HTTP

```
GET https://www.api.unicloud.com/1/files/view
```

Parámetros

Nombre del parámetro	Tipo	Descripción
Obligatorios		
token	string	Token OAuth usado para la autorización.
service	string	Servicio del que se quieren obtener los archivos. Posibles valores: all, dropbox, gdrive.
path	string	Ruta del archivo que se desea obtener.

Respuestas

```
- 200: Successful
- 400: Invalid or missing token
- 401: Invalid or missing params
- 402: Invalid or missing path
- 403: Invalid or missing service
```

Ejemplo OK (service = gdrive, path = “/Pruebas“)

```
            {
              "service" : "dropbox",
              "mime" : "folder",
              "size" : "0 bytes",
              "path" : "/Pruebas",
              "modified" : "Tue, 11 Mar 2014 09:28:53 +0000",
              ["url" : <url>]
            }
```

Ejemplo ERROR

```
            {
                "error" : "Invalid path."
            }
```

7.3.3. Buscar

Petición HTTP

```
GET https://www.api.unicloud.com/1/files/search
```

Parámetros

Nombre del parámetro	Tipo	Descripción
Obligatorios		
token	string	Token OAuth usado para la autorización.
query	string	Cadena para realizar la búsqueda.
Opcionales		
service	string	Servicio del que se quieren obtener los archivos. Posibles valores: all, dropbox, gdrive.
path	string	Ruta del archivo que se desea obtener.

Respuestas

```
- 200: Successful
- 400: Invalid or missing token
- 401: Invalid or missing params
- 402: Invalid or missing path
- 403: Invalid or missing service
```

Ejemplo OK (query = “arancha”)

```
            {
              "query" : "d",
              "path" : "/",
              "files" : [
                {
                  "service" : "dropbox",
                  "mime" : "folder",
                  "path" : "/Personal/Shared",
                  "size" : "0 bytes",
                  "modified" : "Tue, 12 Nov 2013 17:10:59 +0000"
                }
              ]
            }
```

Ejemplo ERROR

```
            {
                "error" : "Missing query."
            }
```

7.3.4. Crear carpeta

Petición HTTP

```
GET https://www.api.unicloud.com/1/files/create_folder
```

Parámetros

Nombre del parámetro	Tipo	Descripción
Obligatorios		
token	string	Token OAuth usado para la autorización.
service	string	Servicio del que se quieren obtener los archivos. Posibles valores: dropbox, gdrive.
path	string	Ruta del archivo que se desea obtener.

Respuestas

```
- 200: Successful
- 400: Invalid or missing token
- 401: Invalid or missing params
- 402: Invalid or missing path
- 403: Invalid or missing service
- 406: There is already a folder at the given destination
```

Ejemplo OK (query = “dropbox”, path = “/Pruebas/prueba3”)

```
{
  "service" : "dropbox",
  "path" : "/Pruebas/prueba3"
}
```

Ejemplo ERROR

```
{
    "error" : "Path must not be empty."
}
```

7.3.5. Descargar

Petición HTTP

```
GET https://www.api.unicloud.com/1/files/download
```

Parámetros

Nombre del parámetro	Tipo	Descripción
Obligatorios		
token	string	Token OAuth usado para la autorización.
service	string	Servicio del que se quieren obtener los archivos. Posibles valores: dropbox, gdrive.
path	string	Ruta del archivo que se desea obtener.

Respuestas

```
- 200: Successful
- 400: Invalid or missing token
- 401: Invalid or missing params
- 402: Invalid or missing path
- 403: Invalid or missing service
```

Ejemplo OK (query = “dropbox”, path = “/Pruebas/prueba3”)

```
{
    "url": <url>,
}
```

Ejemplo ERROR

```
{
    "error" : "Invalid or missing service."
}
```

7.3.6. Copiar

Petición HTTP

```
POST https://www.api.unicloud.com/1/files/copy
```

Parámetros

Nombre del parámetro	Tipo	Descripción
Obligatorios		
token	string	Token OAuth usado para la autorización.
service	string	Servicio del que se quieren obtener los archivos. Posibles valores: dropbox, gdrive.
from_path	string	Ruta del archivo origen.
to_path	string	Ruta para la copia del archivo.

Respuestas

```
- 200: Successful
- 400: Invalid or missing token
- 401: Invalid or missing params
- 402: Invalid or missing source/destinaton path
- 403: Invalid or missing service
- 404: The source file wasn't found at the specified path
- 405: An invalid copy operation was attempted (e.g. there is
already a file at the given destination, or trying to copy a shared
folder)
- 406: Too many files would be involved in the operation for it to
complete successfully. The limit is currently 10,000 files and
folders
```

Ejemplo OK (service = dropbox, from_path = /Pruebas, to_path = /Otra/pruebas)

```
            {
              "service" : "dropbox",
              "path" : "/Otra/pruebas",
              "size" : "0 bytes",
              "modified" : "Thu, 27 Mar 2014 10:36:54 +0000"
            }
```

Ejemplo ERROR

```
          {
              "error" : "An invalid copy operation was attempted."
          }
```

7.3.7. Mover

Petición HTTP

```
POST https://www.api.unicloud.com/1/files/move
```

Parámetros

Nombre del parámetro	Tipo	Descripción
Obligatorios		
token	string	Token OAuth usado para la autorización.
service	string	Servicio del que se quieren obtener los archivos. Posibles valores: dropbox, gdrive.
from_path	string	Ruta del archivo.
to_path	string	La nueva ruta del archivo.

Respuestas

```
- 200: Successful
- 400: Invalid or missing token
- 401: Invalid or missing params
- 402: Invalid or missing source/destinaton path
- 403: Invalid or missing service
- 404: The source file wasn't found at the specified path
- 405: An invalid move operation was attempted (e.g. there is
already a file at the given destination, or trying to move a shared
folder)
- 406: Too many files would be involved in the operation for it to
complete successfully. The limit is currently 10,000 files and
folders
```

Ejemplo OK (service = dropbox, from_path = /Pruebas, to_path = /Otra/pruebas)

```
            {
              "service" : "dropbox",
              "path" : "/Otra/pruebas",
              "size" : "0 bytes",
              "modified" : "Thu, 27 Mar 2014 10:36:54 +0000"
            }
```

Ejemplo ERROR

```
          {
              "error" : "An invalid move operation was attempted."
          }
```

7.3.8. Borrar

Petición HTTP

```
POST https://www.api.unicloud.com/1/files/delete
```

Parámetros

Nombre del parámetro	Tipo	Descripción
Obligatorios		
token	string	Token OAuth usado para la autorización.
service	string	Servicio del que se quieren obtener los archivos. Posibles valores: dropbox, gdrive.
path	string	Ruta del archivo.

Respuestas

```
- 200: Successful
- 400: Invalid or missing token
- 401: Invalid or missing params
- 402: Invalid or missing path
- 403: Invalid or missing service
- 404: The source file wasn't found at the specified path
- 406: Too many files would be involved in the operation for it to
complete successfully. The limit is currently 10,000 files and
folders
```

Ejemplo OK (service = dropbox, path = /Pruebas)

```
            {
              "service" : "dropbox",
              "path" : "/Pruebas",
              "size" : "0 bytes",
              "modified" : "Thu, 27 Mar 2014 10:36:54 +0000"
            }
```

Ejemplo ERROR

```
            {
                "error" : "No file was found at the specified
                           path."
            }
```

7.3.9. Subir

Petición HTTP

```
POST https://www.api.unicloud.com/1/files/upload
```

Parámetros

Nombre del parámetro	Tipo	Descripción
Obligatorios		
token	string	Token OAuth usado para la autorización.
service	string	Servicio del que se quieren obtener los archivos. Posibles valores: dropbox, gdrive.
path	string	Ruta del archivo.

Respuestas

```
- 200: Successful
- 400: Invalid or missing token
- 401: Invalid or missing params
- 402: Invalid or missing path
- 403: Invalid or missing service
- 404: The source file wasn't found at the specified path
- 500: Error uploading to Dropbox (Content-Length).
```

Ejemplo OK (service = dropbox, path = /Pruebas)

```
            {
              "service" : "dropbox",
              "path" : "/Pruebas",
              "size" : "0 bytes",
              "modified" : "Thu, 27 Mar 2014 10:36:54 +0000"
            }
```

Ejemplo ERROR

```
            {
                "error" : "Error uploading to Dropbox (Content-
Length)."
            }
```

Capítulo 8. Conclusiones

Partiendo del estudio realizado y del cada vez uso más extendido de los servicios de almacenamiento en la nube se puede llegar a la conclusión de que un servicio como el desarrollado en este proyecto puede ser muy útil para la integración de varios servicios facilitando y acercando más al usuario dichos servicios.

La migración del almacenamiento tradicional al almacenamiento en la nube es una realidad y la mayoría de usuarios que migran a la nube no se decantan por un solo servicio ya que normalmente no suele ser suficiente. Esto les lleva a la necesidad de tener varios servicios y tener separados los archivos, haciendo más difícil la búsqueda y organización de los mismos. Es aquí donde una herramienta como UniCloud pueda llegar a ser de una gran utilidad.

Se han logrado todos los objetivos marcados con especial hincapié en la flexibilidad del sistema. Como se expuso en la introducción del proyecto, la flexibilidad es muy importante de cara al futuro del proyecto, por que cuanto más fácil sea añadir nuevos servicios, más uso tendrá.

Respecto a la parte técnica, este proyecto es tan sólo la punta del iceberg de lo que puede llegar a ser. A corto plazo, cumple con su cometido, facilitando el uso de varios servicios pero a largo plazo, UniCloud puede llegar a tener tanto potencial como en su día tuvo Google.

El mayor motivo de esto se basa en el algoritmo del que se podría dotar a UniCloud para hacer mucho más eficiente la gestión, no sólo de los archivos para el usuario, si no que, también el uso de las API de los servicios.

Por lo que se podría concluir que el futuro de este proyecto sería la dotación de inteligencia artificial a los algoritmos para la gestión de los archivos, pasando de un control por parte del usuario a un sistema que toma decisiones por él.

Bibliografía

Capítulo 1. What exactly is RESTful programming?, http://stackoverflow.com/questions/671118/what-exactly-is-restful-programming

Capítulo 1. What are RESTful services?, http://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html

Capítulo 1. JSON: What is, How it works, How to use it, http://www.copterlabs.com/blog/json-what-it-is-how-it-works-how-to-use-it/

Capítulo 3. Google Drive, http://en.wikipedia.org/wiki/Google_Drive

Capítulo 3. Google Drive API Documentation, https://developers.google.com/drive/v2/reference/

Capítulo 4. Dropbox, http://en.wikipedia.org/wiki/Dropbox_(service)

Capítulo 4. Dropbox API Documentation, https://www.dropbox.com/developers/core/docs

Capítulo 6. Arquitectura MVC, http://en.wikipedia.org/wiki/Model–view–controller

Capítulo 6. Qué es MVC, http://www.desarrolloweb.com/articulos/que-es-mvc.html

Capítulo 6. Arquitectura en 3 capas, Arquitectura MVC y POO, http://www.eduardoaf.com/blog-frameworks/frameworks-mvc/arquitectura-en-3-capas-arquitectura-mvc-y-la-poo-1/