

## MEMORIA API VIRTUALDISKSD.

### Índice de Contenidos:

1. Justificación.
2. Objetivo.
3. Planificación del Proyecto.
  - a. Etapas.
  - b. Diagrama de Grantt.
  - c. Roles.
  - d. Reparto de tareas.
4. Características de los servicios
5. Requisitos Previos.
6. Problemas con las APIs.
7. Implementación de VIRTUALDISKSD
  - a. Funciones que Implementa la API
  - b. Persistencia con JSON
  - c. Problemas durante el Desarrollo.
8. Funcionalidades que ofrece la API
9. Instalación y Desplegado.
  - a. Local.
  - b. Servidor.
10. Uso y Pruebas de la Aplicación.
11. App Móvil y Ampliaciones futuras.
  - a. App Móvil.
  - b. Ampliaciones.
12. Histórico de Cambios.

## 1. Justificación.

VIRTUALDISKSD es un API implementado en Python (2.7) que viene a resolver una situación provocada por la oferta de almacenamiento gratuito en la nube. El hecho de registrarse en varios servicios gratuitos nos aporta una gran capacidad de almacenamiento, pero nos obliga a llevar un control de qué ficheros subimos y dónde los subimos, encontrándonos en ocasiones con situaciones en las que tenemos un servicio prácticamente en desuso y otro al borde del colapso. VIRTUALDISKSD nace con la finalidad de ofrecer un servicio centralizado de almacenamiento en la nube.

## 2. Objetivo.

Centralizar todos los servicios de almacenamiento en la nube del usuario en una única API que los gestione de forma transparente.

## 3. Planificación del proyecto.

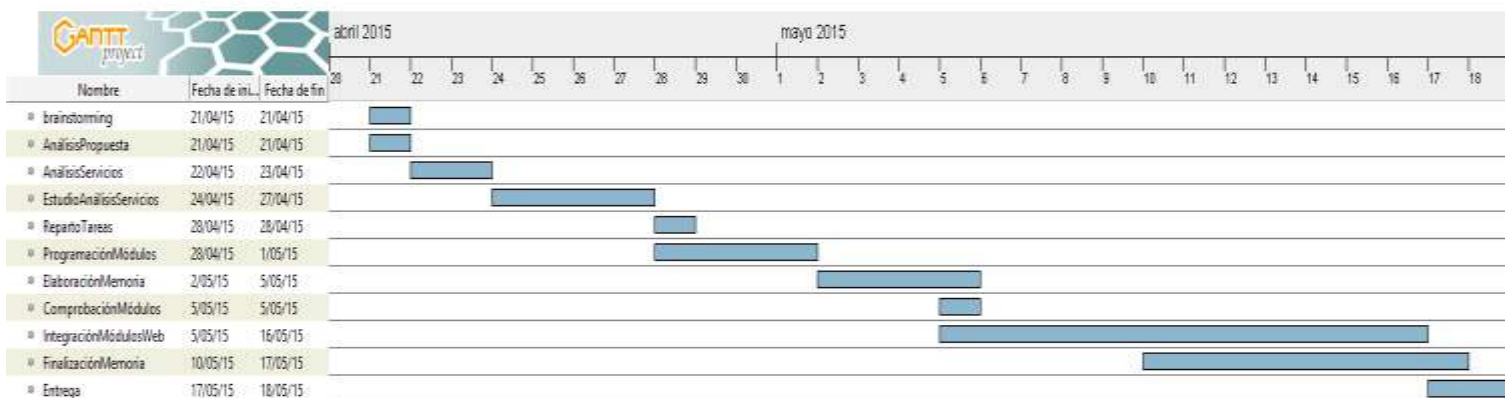
### a. Etapas:

El proyecto se ha dividido en 11 etapas, cuyo diagrama de Gantt se ha añadido como Anexo 1, siendo su descripción la siguiente:

- **Brainstorming:** Duración estimada de 2 horas. En horario de clase de la asignatura, todos los miembros del equipo aportarán ideas sobre la finalidad de la API, consensuándose al final varias propuestas para su estudio.
- **Análisis Propuestos:** Duración estimada de 1 día. Los miembros del grupo, de forma individual, sopesará los pros y contras de cada propuesta salida de la etapa anterior y las jerarquizará. Al final del día, se consensuará la API a diseñar.
- **Análisis Servicios:** Duración estimada de 2 días. Una vez consensuada la API a diseñar, cada miembro del grupo de forma individual analizará al menos un servicio disponible en la nube que satisfaga los requisitos para el desarrollo de la misma. Al finalizar dicho periodo, informará al resto del grupo de la información obtenida.
- **Estudio Análisis Servicios:** Duración estimada de 4 días. Cada miembro del grupo de forma individual, buscará información de las APIs disponibles de los servicios analizados en la anterior etapa. Se prestará especial atención a la información disponible sobre las APIs.
- **Reparto Tareas:** Duración estimada de 1 hora. En horario de clase de la asignatura, con la información obtenida en la etapa anterior, el grupo consensuará las dos APIs que se integrarán. Una vez seleccionadas, los 2 miembros del grupo que analizaron dichas APIs asumirán el rol de “responsable de programación” (cada uno de la que estudió) y los otros dos asumirán el rol de “ayudante de programación”, los cuales serán asociados a un responsable de programación.
- **Programación Módulos:** Duración estimada de 4 días. Para cada API seleccionada, el responsable de programación y el ayudante de programación elaborarán las funciones necesarias usando su API, así como las plantillas HTML vinculadas a dicha API.
- **Elaboración Memoria:** Duración estimada de 4 días. Simultáneamente a la tarea anterior y posteriormente a la misma se deberá ir redactando la memoria del proyecto.

- **Comprobación Módulos:** Duración estimada de 1 h. En horario de clase de la asignatura, con los módulos implementados en la etapa anterior, el grupo verificará el funcionamiento de los dos módulos y repartirá las tareas de integración de los mismos.
- **Integración Módulos Web:** Duración estimada de 11 días. En esta etapa se programará la API definitiva y su integración con un servicio web que permita comprobar su funcionamiento. Se designará un “responsable de integración”, quedando el resto del equipo a disposición del mismo para el diseño y programación de las secciones que dicho responsable estime oportuno.
- **Finalización Memoria:** Duración estimada de 7 días. Simultáneamente a la tarea anterior y posteriormente a la misma se deberá terminar de redactar la memoria en base al trabajo ejecutado.
- **Entrega:** Duración estimada de 1 hora. En horario de clase de la asignatura, se procederá a la entrega del trabajo al profesor responsable de la asignatura.

b. **Diagrama de Grantt:**



c. **Roles:**

- **Director del proyecto:** Se encargará de velar por el cumplimiento de la planificación, teniendo el poder de decisión en aquellas situaciones en las que el consenso no sea posible.
- **Responsable de la memoria:** Se encargará de velar por la correcta elaboración de la memoria, con especial énfasis en la coherencia de la misma, siendo responsabilidad suya la comprobación de la aportación de todos los miembros.
- **Responsable de programación:** Se encargará de la correcta programación de un módulo de la API, así como de las plantillas en HTML relacionadas con dicho módulo.
- **Ayudante de programación:** Se encargará de las tareas asignadas por el responsable de programación en el ámbito de la programación del módulo asignado y de las plantillas HTML asignadas.
- **Responsable de integración:** Se encargará de la correcta programación de la integración de los módulos y de la página web que permita su comprobación. Repartirá la carga de trabajo según su propio criterio entre los otros tres miembros del grupo.

d. **Reparto de tareas:**

- **Bello Villanueva, Emilio:** Responsable de la memoria y ayudante de programación.
- **De la Torre Macías, Juan Carlos:** Director del proyecto y ayudante de programación.
- **Francisco Aparicio, Manuel:** Responsable de programación y responsable de integración.
- **Vidal Jiménez, José Manuel:** Responsable de programación y responsable de integración.

#### 4. Características de los Servicios.

Para alojar la información, se requiere de servicios que admitan el almacenamiento de datos en la nube. Estos servicios serán los utilizados para alojar los datos que los usuarios suban de manera transparente, dando la sensación al usuario de que se encuentra ante un único servicio de almacenamiento.

Para elegir los servicios a investigar, se consensuaron los siguientes requisitos:

- a. **Prestigio:** Los usuarios deben de tener previamente cuenta en los servicios de alojamiento, por tanto es imprescindible que sean conocidos para facilitar la adopción.
- b. **Capacidad:** Los servicios deben de tener una capacidad considerable, no siendo óptimo un servicio que disponga de una capacidad exponencialmente inferior al resto.
- c. **API:** Los servicios deben de disponer de una API para desarrolladores, ya que esta es la esencia de poder trabajar con servicios ajenos de manera sencilla. Además el uso de la API debe de ser intuitivo, sencillo y estar bien documentado.
- d. **Autenticación OAuth 2:** Este protocolo de autenticación provee una forma segura de realizar autenticaciones a través de APIs, ya que creas una aplicación en el servicio, generando unas keys únicas y se facilita información de esta, no de la cuenta del usuario, de modo que esta información se envía y se recibe un token como respuesta, si este token es correcto la autenticación es correcta, sino no se permite el acceso.

Con estas características en mente y tras un filtro previo, los servicios de alojamiento elegidos son: Dropbox, Google Drive, Box y Microsoft OneDrive.

#### 5. Requisitos previos.

El objetivo de VIRTUALDISKSD no es otro que realizar una capa superior y transparente para los usuarios, donde utilizando una aplicación, se pueda interactuar con varios servicios de almacenamiento en la nube, de modo que será VIRTUALDISKSD el encargado de tratar con las APIs de los servicios para manejar la información sin que el usuario tenga consciencia de que por ejemplo dos archivos pueden estar alojados en diferentes servicios. De esta manera se propusieron los siguientes requisitos para la aplicación:

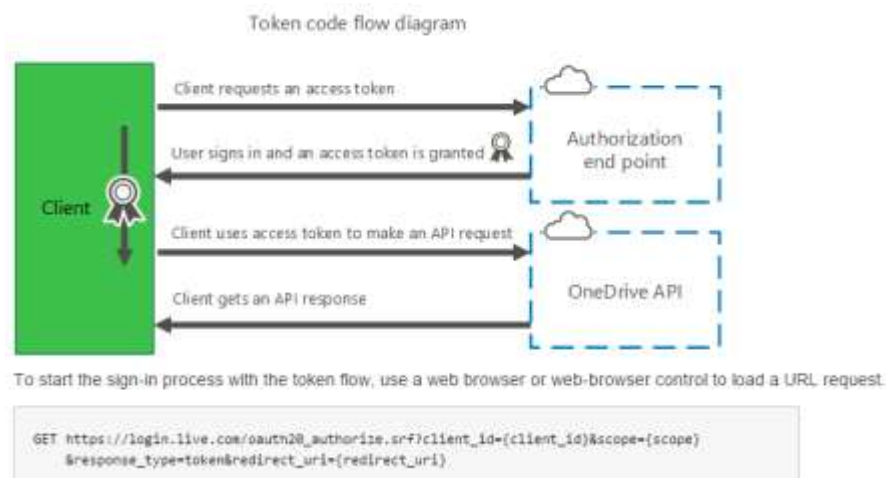
- **Subir fichero:** El objetivo de este requisito no es otro que poder realizar una subida a un servicio de almacenamiento en la nube, de manera transparente para el usuario, de modo que antes de realizar la subida se compruebe cual es el servicio que tiene mayor capacidad en ese momento y se procederá a realizar la subida a ese servicio, por lo que el usuario en ningún momento elegirá a que servicio realizar la subida.

- **Capacidad restante:** Para elegir a que servicio realizar la subida, se necesita saber cuál es la capacidad de la que dispone el servicio, por lo que es necesario conocer este dato devolviendo un entero con el que realizar la comparación.
- **Listar ficheros:** Este requisito mostrará por pantalla todos los ficheros almacenados en la nube, de manera que el usuario pueda ver la información previamente subida.
- **Bajar fichero:** Una vez subida la información, es necesario poder tener una manera de descargarla para poder acceder a ella.
- **Eliminar fichero:** Cuando un fichero deja de ser útil, se debe de proceder a su eliminación, por lo que se debe de disponer de un método que elimine un fichero aunque el usuario no sepa donde se encuentra almacenado dicho fichero.
- **Reorganizar ficheros:** Se sumará la capacidad disponible de todos los servicios y para ver si un archivo de gran tamaño que no cabría en ningún de ellos podría llegar a caber haciendo una reorganización de los datos, de manera que si el fichero cupiese se proceda al traslado de datos de un servicio al resto y luego se proceda a la subida al servicio con espacio disponible.
- **Autenticación:** Se necesitará un método transparente, previa disposición de una cuenta en el servicio de almacenamiento, para de este modo poder hacer uso del servicio desde VIRTUALDISKSD.

## 6. Problemas con las APIs.

- a. **Google Drive:** La API de Google Drive tiene buena documentación oficial, por lo que no nos hemos encontrado con problemas mayores (al contrario que la API de OneDrive). Sin embargo, ha habido dos detalles importantes que cabrá resaltar:
  - Para aplicaciones locales, la API de Google no devuelve los credenciales automáticamente, por lo que el usuario está obligado a copiar y pegar en un formulario habilitado para ello. Este problema es fácilmente solucionable si la aplicación es desplegada en un servidor (habrá que reconfigurar los tokens de la aplicación y el ámbito scope).
  - La función que realiza la subida del fichero a Google Drive recibe la ruta absoluta al fichero en el sistema. El principal problema que esto ocasiona es la necesidad de guardar el flujo de bytes enviado por el usuario mediante POST (recordemos que el usuario utiliza un formulario de HTML para subir el fichero) en un fichero temporal que será borrado una vez que se complete la subida a los servidores de Google Drive. Aunque la aplicación sea ejecutada en local, no podemos acceder a la ruta del fichero por directivas de seguridad de los navegadores, de ahí que nos veamos obligados a usar Python para ello.
  - No existen las carpetas. Google Drive almacena todo en un mismo directorio. Las carpetas son emuladas mediante etiquetas.
  - Al listar, Google Drive no devuelve la lista completa de ficheros, sino que va enviando un número limitado de ficheros y proporciona una referencia a la siguiente página. Esto implica tener que hacer varias peticiones (dentro de un bucle while) hasta tener la lista completa, con el consiguiente retardo que esto ocasiona.
- b. **Dropbox:** La API de Dropbox está bien documentada y es oficial. No hemos encontrado ningún problema al trabajar con ella, pero creemos conveniente resaltar lo siguiente:

- Dropbox devuelve los credenciales mediante GET a la aplicación. No es necesario que el usuario los copie a mano (al contrario que en Google Drive).
- La función que realiza la subida del fichero recibe un flujo de bytes, por lo que no es necesario guardar el fichero enviado mediante POST en un archivo temporal local.
- c. **Box:** La API de Box es la mejor documentada y con más códigos de ejemplo de los 4 servicios elegidos a priori, tanto es así que todo el código se encuentra ya realizado para su puesta en marcha con una serie de ejemplos que contiene el github oficial de la aplicación. Por esta razón no se han encontrado problemas durante el trabajo con esta API.
- d. **Microsoft OneDrive:**
  - Api Python (no oficial): <https://github.com/mk-fg/python-onedrive.git>
  - OneDrive API: <https://github.com/OneDrive/onedrive-api-docs.git>
  - Un gran problema que nos hemos encontrado con la api para Python es la falta de información en la documentación que ofrece. Ésta falta de información provocó un esfuerzo excesivo en el intento por entender dicha api mediante técnicas de acierto y error.
  - Sin embargo, nuestro talón de Aquiles con la API llegó en el proceso de autenticación de la misma.  
Como podemos ver en la imagen, OneDrive nos devuelve un token tras la solicitud de acceso, el cual usaremos posteriormente para realizar las distintas solicitudes al servicio.



([https://dev.onedrive.com/auth/msa\\_oauth.htm](https://dev.onedrive.com/auth/msa_oauth.htm))

Cuando solicitamos el token, se abre una nueva ventana en el navegador para autenticarnos y autorizar la conexión del servicio con la api, tras lo cual, se nos devuelve el token en la barra de direcciones. Y éste es el problema, cuando el servidor web es ejecutado de forma local, el token es devuelto en una ventana que no es de nuestro servidor web, ya que OneDrive no permite redireccionar a localhost. Esto nos impide recogerlo. Esta situación debería ser resuelta por la api de Python, pero no es el caso.

En las distintas pruebas que se hicieron, si utilizamos un redireccionamiento a una web en la nube, entonces el token si es devuelto a nuestra página.

- Tras demasiadas horas de trabajo dedicadas al aprendizaje de dicha api, se deshecha su uso por la restricción de no poderla usar en localhost.

Ante estos problemas, tras consultar al profesor y viendo que el enunciado del ejercicio tan solo propone el uso de 2 APIs diferentes, se decide centrar los esfuerzos en la utilización de las APIs Dropbox y Google Drive, desechando el uso de Box debido a la simplicidad de trabajar con esta API, no dando lugar a una investigación por nuestra parte y Microsoft OneDrive, debido al caso contrario tal como se indicó anteriormente. Se dejan estos servicios para una futura ampliación.

## 7. Implementación de VIRTUALDISKSD.

Como lenguaje de programación se decide utilizar Python, ya que es un lenguaje sencillo, se ve en la asignatura de Sistemas Distribuidos, tiene compatibilidad con multitud de APIs y además dispone de frameworks que dotan al lenguaje de capacidad web. Como framework para el uso en la web de la aplicación se elige Flask por las recomendaciones habidas en clase.

- **Funciones que utiliza API VIRTUALDISKSD:**
  - `automaticUpload()`: Compara que servicio tiene mayor capacidad y realiza la subida a de un fichero a este.
  - `getlist()`: Obtiene los json de todos los servicios y los junta en uno, mostrando la información en la web.
  - `getDriveAuth()`: Obtiene la url de autorización y la devuelve con un json.
  - `driveAuth()`: Recibe el código de la autorización y lo guarda en un fichero de texto.
  - `getDriveQuota()`: Devuelve un json con información sobre el espacio de Google Drive como la capacidad total o el espacio usado.
  - `uploadToDrive()`: Método para subir un archivo a Google Drive, devuelve el código 200 si todo es correcto.
  - `getDriveList()`: Método para obtener un json con los archivos alojados en Google Drive.
  - `removeDriveFile()`: Elimina un archivo de Google Drive, devuelve un código 200 para indicar que todo ha sido correcto.
  - `getDropboxAuth()`: Devuelve un json con la url de la autorización.
  - `dropboxAuth()`: Guarda la autorización en un fichero de texto.
  - `uploadToDropbox()`: Método usado para subir un fichero a Dropbox, devuelve el código 200 si todo es correcto.
  - `getDropboxList()`: Método para obtener un json con los archivos alojados Dropbox.
  - `getDropboxQuota()`: Devuelve un json con información sobre el espacio de Dropbox como la capacidad total o el espacio usado.
  - `removeDropboxFile()`: Elimina un archivo de Dropbox, devuelve un código 200 para indicar que todo ha sido correcto.
- **Persistencia con JSON:** En nuestra aplicación, hacemos uso de dos ficheros que contienen JSON, con la finalidad de almacenar los credenciales de autorización para las APIs de Google Drive y Dropbox. Estos ficheros son creados una vez que se generan los credenciales, y se mantienen mientras que sean válidos o el usuario no genere unos nuevos. En futuras versiones, convendrá guardar los credenciales en una base de datos, para mejorar el mantenimiento, seguridad y escalabilidad.
- **Problemas surgidos durante el desarrollo:**

A lo largo del desarrollo, nos hemos encontrado con algunos problemas, algunos de ellos han sido solventados fácilmente.

- **Flask soporta únicamente (por limitaciones de python) un hilo de ejecución:** Esto implica que no puede servirse contenido a varios usuarios de forma simultánea, y que hacer peticiones internas desde la API a la API (usando peticiones HTTP en vez de llamadas a procedimientos) cause interbloqueos.
- **Flask no soporta peticiones cross-domain:** La principal implicación que esto ha tenido sobre nuestra aplicación es la imposibilidad de separar el cliente web de la API (Error obtenido: Solicitud desde origen distinto bloqueada: la política de mismo origen impide leer el recurso remoto en <URL>. Esto se puede arreglar moviendo el recurso al mismo dominio).

## 8. Funcionalidades que ofrece la API.

El objetivo fundamental de este ejercicio no es otro que ofrecer una API que sirva como adaptador de otras APIs que se encontraran a más bajo nivel, de modo que utilizando el API VIRTUALDISKSD se pueda ofrecer el servicio descrito anteriormente. Los métodos que se proveen son los siguientes:

### 1. API Global:

- **Subida Automática:** Se sube de forma automático el archivo seleccionado al servicio que disponga de mayor capacidad libre en ese momento:
  - Ruta: /automatic\_upload
  - Tipo: Post
  - Devuelve: Llamada a la función uploadToDrive o uploadToDropbox
- **Obtener lista de ficheros de ambos servicios:** Obtener la lista de ficheros de todos los archivos alojados en los servicios utilizados.
  - Ruta: /get\_list
  - Tipo: Get
  - Devuelve: JSON = [{
    - 'id': path,
    - 'filename': path (without /)
    - 'link': path.url,
    - 'size': size (bytes)
 }];

### 2. API Drive:

- **Obtener URL para autorizar en Google Drive:** Google Drive necesita una autorización para poder conectarse con el servidor.
  - Ruta: /get\_drive\_auth
  - Tipo: Get
  - Devuelve: JSON = { 'url': url\_autorizacion };
- **Obtener URL para autorizar Drive:** Se necesita una url para autenticarse en google drive
  - Ruta: /get\_drive\_auth
  - Tipo: Get
  - Devuelve: JSON = { 'url': url\_autorizacion };
- **Recibir código de autorización:** Una vez se recibe el código de autenticación, este se guarda en un fichero txt
  - Ruta: /save\_drive\_auth
  - Tipo: Post (authcode -> Código de autorización)
  - Devuelve: 200 OK



- **Obtener cuotas de Drive:** Se recoge información sobre las capacidades del servicio.
  - Ruta: /get\_drive\_quota
  - Tipo: GET
  - Devuelve: {'used': long, 'total': long}
- **Subir fichero a Drive:** Se fuerza a subir un fichero al servicio Google Drive.
  - Ruta: /upload\_to\_drive
  - Tipo: Post (file -> Flujo de bytes del fichero)
  - Devuelve: 200 OK
- **Obtener lista de ficheros de Drive:** Se fuerza a obtener únicamente la lista de ficheros alojados en el servicio Google Drive.
  - Ruta: /get\_drive\_list
  - Tipo: GET
  - Devuelve JSON = [{
    - 'id': file.id,
    - 'filename': file.title,
    - 'link': file.alternateLink,
    - 'size': file.size (bytes)
 }];
- **Eliminar fichero Drive:** Se elimina el fichero seleccionado de drive.
  - Ruta: /remove\_drive\_file
  - Tipo: GET
  - Devuelve 200 OK

### 3. API Dropbox:

- **Obtener URL para autorizar Dropbox:** Se necesita una url para autenticarse en Dropbox
  - Ruta: /get\_dropbox\_auth
  - Tipo: GET
  - Devuelve: JSON = { 'url': url\_autorizacion };
- **Recibir código de autorización:** Una vez se recibe el código de autenticación, este se guarda en un fichero txt.
  - Ruta: /save\_dropbox\_auth
  - Tipo: Post (authcode -> Código de autorización)
  - Devuelve: 200 OK
- **Subir fichero a Dropbox:** Se fuerza a subir un fichero al servicio Dropbox.
  - Ruta: /upload\_to\_dropbox
  - Tipo: Post (file -> Flujo de bytes del fichero)
  - Devuelve: 200 OK
- **Obtener lista de ficheros de Dropbox:** Se fuerza a obtener únicamente la lista de ficheros alojados en el servicio Dropbox.
  - Ruta: /get\_dropbox\_list
  - Tipo: GET
  - Devuelve: JSON = [{
    - 'id': path,
    - 'filename': path (without /)
    - 'link': path.url,
    - 'size': size (bytes)
 }];
- **Obtener cuotas de Dropbox:** Se recoge información sobre las capacidades del servicio.
  - Ruta: /get\_dropbox\_list

- Tipo: GET  
Devuelve {'used': long, 'total': long}
- **Eliminar fichero Dropbox:** Se elimina el fichero seleccionado de Dropbox.
  - Ruta: /remove\_dropbox\_file
  - Tipo: GET
  - Devuelve 200 OK

#### 4. API Web Global:

- **Home:** Método que es llamado al inicio que es llamado al iniciar la web.
  - Ruta: /
  - Tipo: GET
  - Devuelve upload.html
- **Lista:** Método que es llamado para mostrar la plantilla html con la información con los archivos alojados en los servicios.
  - Ruta: /list
  - Tipo: GET
  - Devuelve list.html
- **Estilos:** Método que es llamado para obtener la hoja de estilos.
  - Ruta: /estilos
  - Tipo: GET
  - Devuelve estilos.css

#### 5. API Web Drive:

- **Configuración de Drive:** Este método es llamado para entrar en el apartado de configuración de Google Drive.
  - Ruta: /driveconfig
  - Tipo: GET
  - Devuelve auth\_drive.html
- **Subir ficheros a Drive:** Devuelve la plantilla html para hacer una subida a Google Drive.
  - Ruta: /driveupload
  - Tipo: GET
  - Devuelve driveupload.html
- **Listar ficheros a Drive:** Devuelve la plantilla html para mostrar los ficheros de Google Drive.
  - Ruta: /drivelist
  - Tipo: GET
  - Devuelve drivelist.html

#### 6. API Web Dropbox:

- **Configuración de Dropbox:** Este método es llamado para entrar en el apartado de configuración de Dropbox.
  - Ruta: /dropboxconfig
  - Tipo: GET
  - Devuelve auth\_dropbox.html
- **Subir ficheros a Dropbox:** Devuelve la plantilla html para hacer una subida a Dropbox.
  - Ruta: / dropboxupload
  - Tipo: GET
  - Devuelve dropboxupload.html
- **Listar ficheros a Dropbox:** Devuelve la plantilla html para mostrar los ficheros de Dropbox.
  - Ruta: / dropboxlist

- Tipo: GET
- Devuelve dropboxlist.html

## 9. Instalación y Desplegado.

**Url del proyecto:** <http://virtualdiskd-empiezaaprogramar.c9.io/>

El servidor elegido para desplegar la aplicación es cloud9, ya que más que un servidor, es un conjunto de herramientas donde se permite desarrollar de manera conjunta y en tiempo real a varias personas, además de proveer una serie de herramientas como servidor propio, de forma que al trabajar con c9 parece que estamos trabajando en local, pero podemos proveer a un tercero o a nosotros mismos para pruebas de una dirección web desde donde se puede acceder al resultado del proyecto.

Por tanto, para instalar el proyecto, ya sea en cloud9 o en local se necesitarán instalar las siguientes dependencias:

```
sudo pip install flask
```

```
sudo pip install google-api-python-client
```

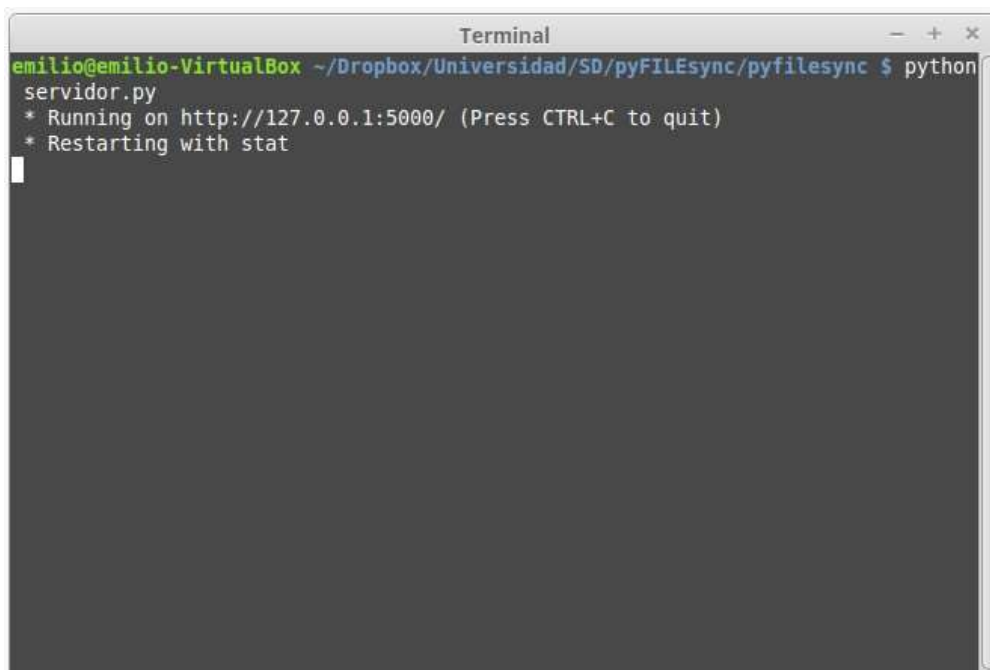
```
sudo pip install Dropbox
```

una vez instaladas las dependencias de manera satisfactoria, ya hay que comenzar a diferenciar en la instalación entre el servidor y el modo local:

- **Local:** Para ejecutar la aplicación en modo local, tan solo se deberá de ejecutar el archivo principal de la siguiente manera y aparecerá la siguiente información:

Python servidor.py

Y ya estaría corriendo la aplicación lista para abrir un enlace en el navegador:



```
Terminal
emilio@emilio-VirtualBox ~/Dropbox/Universidad/SD/pyFILESync/pyfilesync $ python
servidor.py
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
* Restarting with stat
```

- **Cloud9:** Al crear el proyecto en c9, debe de elegirse un proyecto de tipo Python/Django, por lo que el proyecto no se encuentra predefinido para trabajar con flask aunque se instalen las dependencias de Flask, así que con esto en mente se tienen que realizar los siguientes cambios:

- **Cambio en el código:** La siguiente línea:

```
if __name__ == "__main__":
```

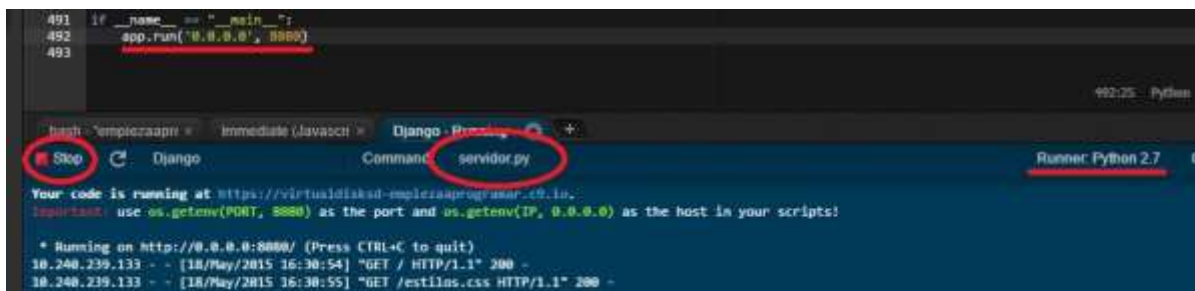
```
    app.run(debug=True)
```

Debe de ser reemplazada porque no se tiene acceso al PYTHONPATH:

```
if __name__ == "__main__":
```

```
    app.run('0.0.0.0', 8080)
```

- **Cambios en el servidor:** Para ejecutar Flask en cloud9, debe pulsarse Run Project y ocurrirá un error, entonces se cambian los siguientes parámetros tal como aparece en la imagen y ahora si, pulsar Run Project se iniciara el archivo .py seleccionado:



## 10. Uso y Prueba de la aplicación.

Url del proyecto: <http://virtualdisksd-empiezaaprogramar.c9.io/>

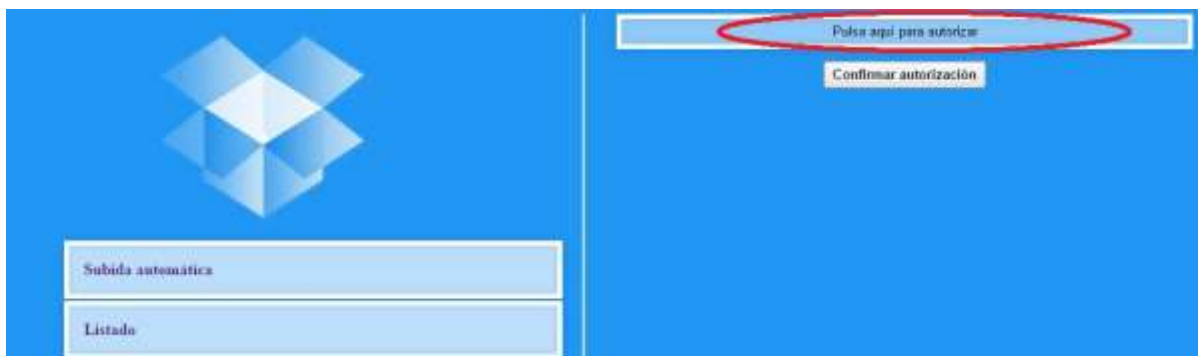
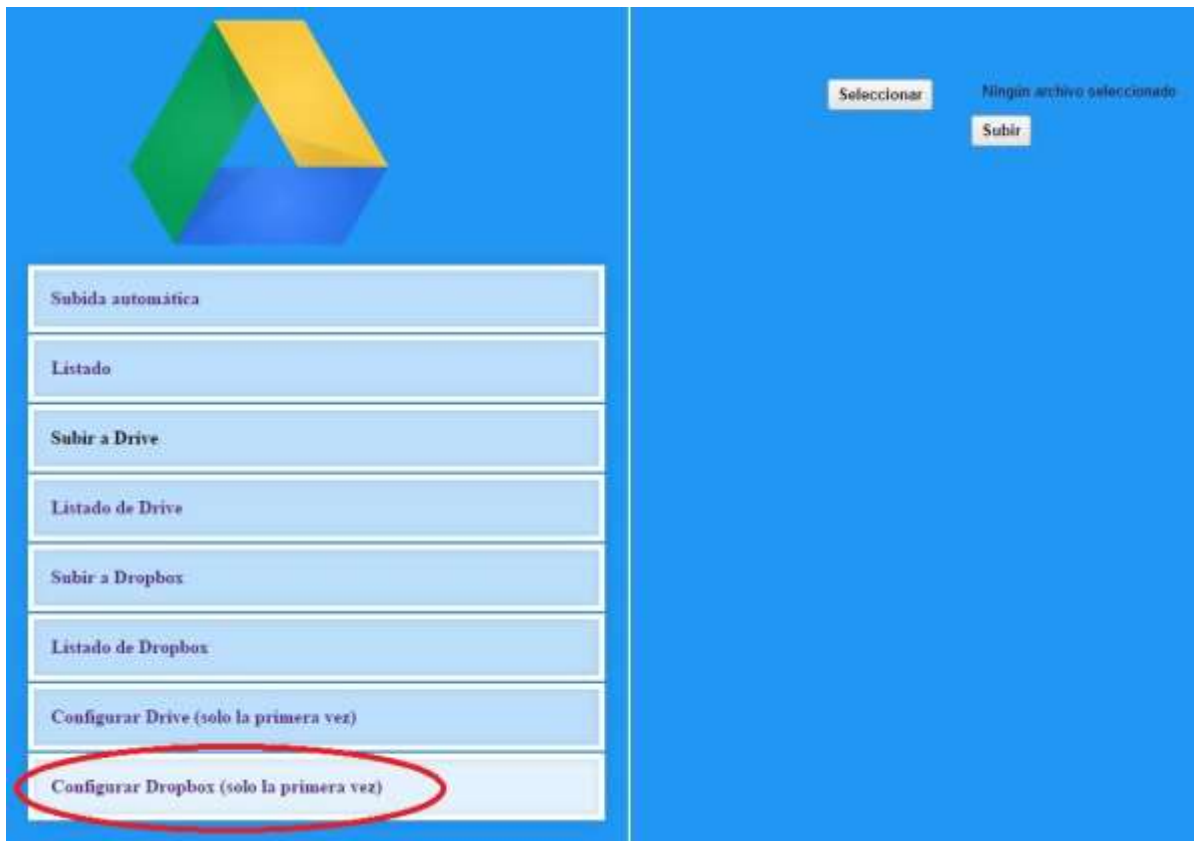
Usuario: sdgrupo1

Contraseña: basura123

La aplicación está orientada para el uso de un público masivo, lo que quiere decir que debe de ser lo más intuitiva posible, adaptándose a usuarios de todos los niveles. Se decidió desde el comienzo del proyecto dotar de un menú al apartado web, para así facilitar la navegación por la aplicación, sin tenerse que usar la barra de navegación en ningún momento.

- **Uso y prueba de la app:**

- **Autenticación de Dropbox:** Lo primero para usar la aplicación por primera vez, deberá de ser autenticarse en los servicios, vamos a ver como autenticarnos en Dropbox:



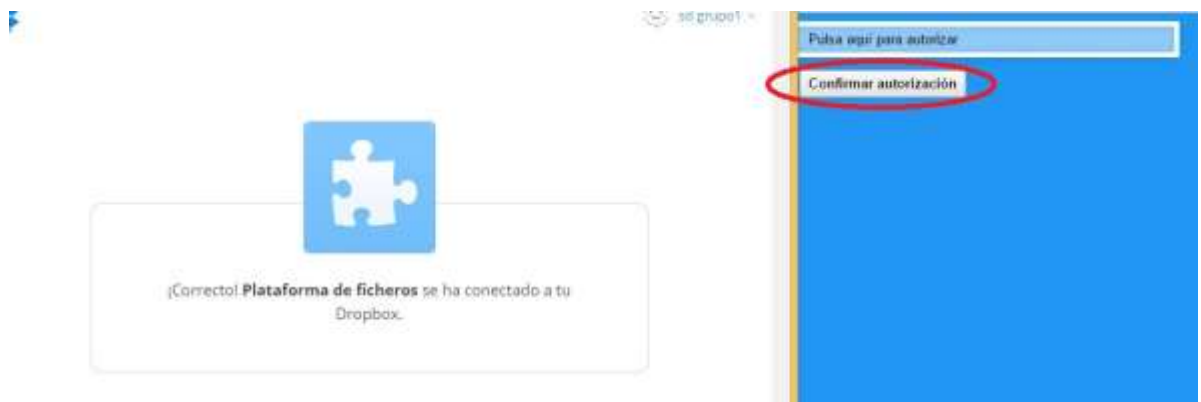
Una vez se pulsa en el botón arriba señalado, aparecerá una nueva ventana donde deberemos de introducir nuestros datos del servicio Dropbox:



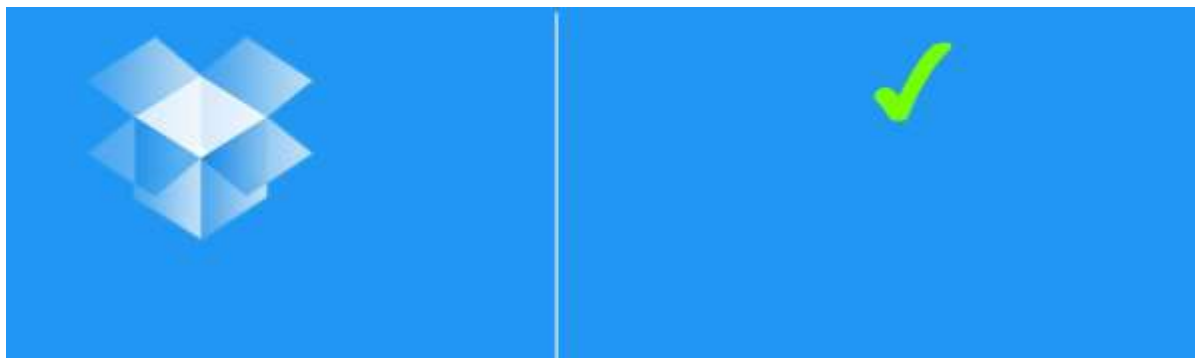
Tras esto se pulsa en iniciar sesión y aparecerá la siguiente pantalla:



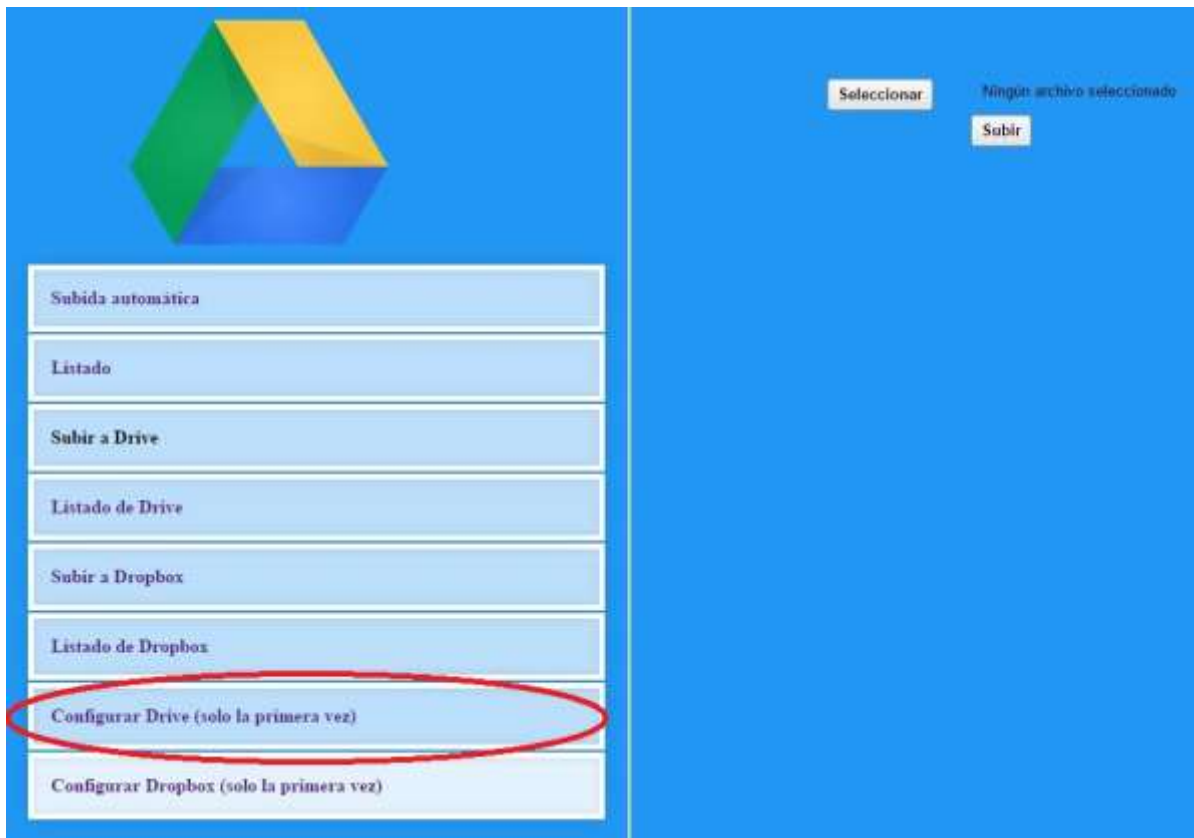
Donde se pulsa en permitir para crear los credenciales para crear la autenticación para la API, tras esto solo quedara confirmar la autorización:



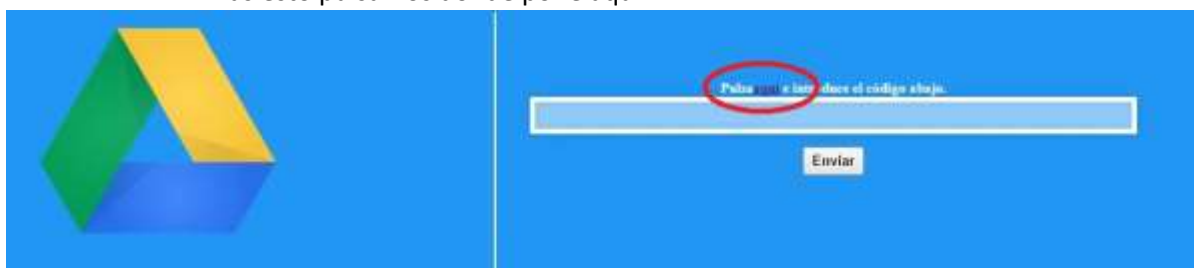
Y tras confirmar tal como indica la imagen aparecerá que todo ha sido correcto:



- **Autenticación de Google Drive:** Tal como antes nos autenticamos en el Dropbox, vamos a hacerlo ahora en Google Drive para tener activos ambos servicios, por lo que debemos de volver a la pantalla inicial:



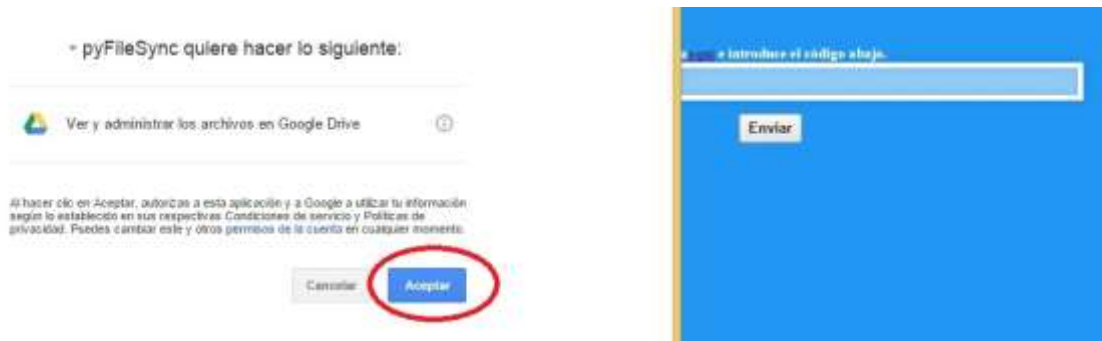
Tras esto pulsamos donde pone aquí:



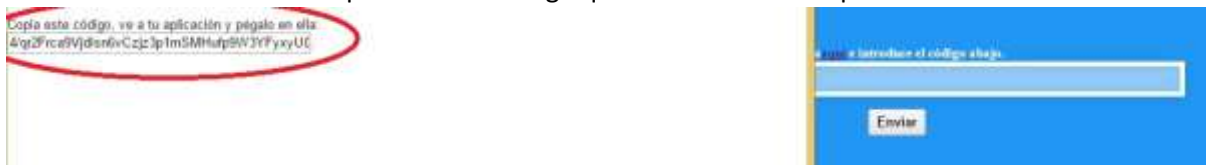
Ahora hacemos login con nuestro usuario de Google Drive e iniciamos sesión:



Aceptamos que se va a administrar la información de Google Drive desde un lugar externo:



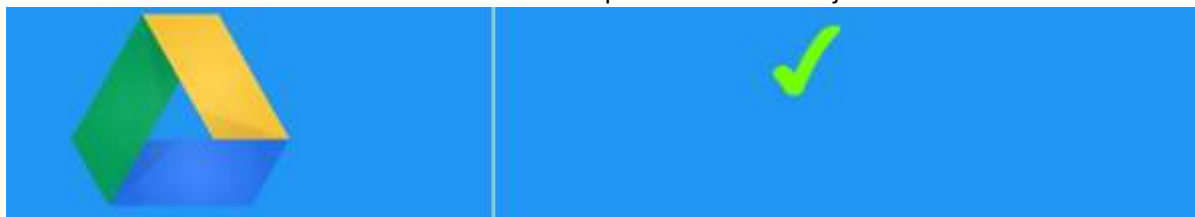
Ahora nos aparecerá un código que deberemos de copiar en su totalidad:



Este código debe de pegarse en el cuadro de texto preparado para esto y que se indica a continuación y pulsar en enviar:



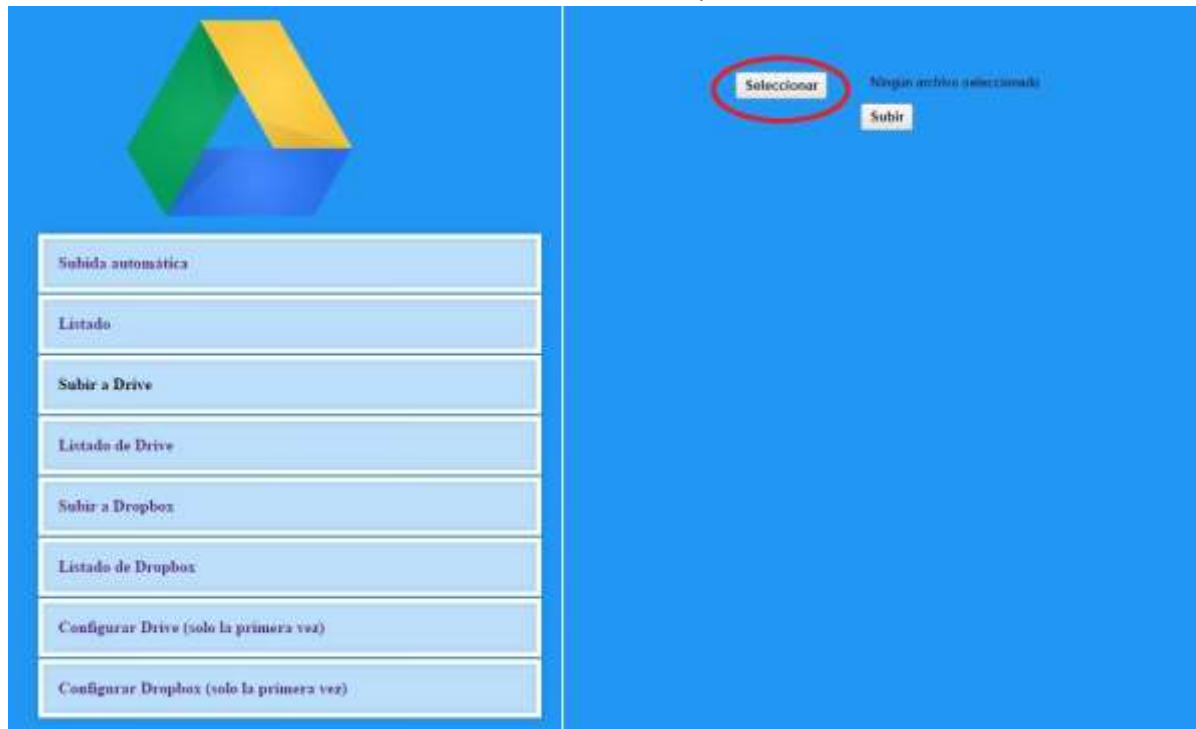
Si todo ha salido correctamente aparecerá un mensaje de éxito:



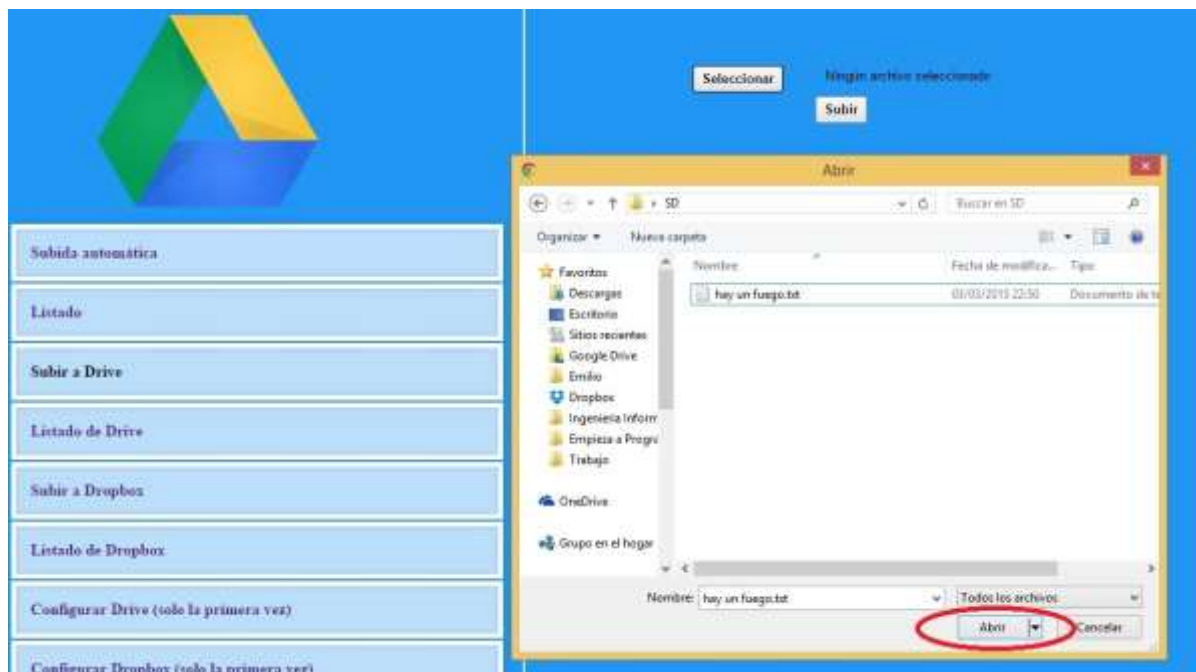
- **Subida automática:** Una vez realizada la autenticación, ya se puede hacer uso de la aplicación, siendo la subida automática la principal característica de esta, ya que con la subida



automática se subirá la información al servicio con más capacidad en ese momento:



Tras pulsar en seleccionar se selecciona un archivo cualquiera del PC local:



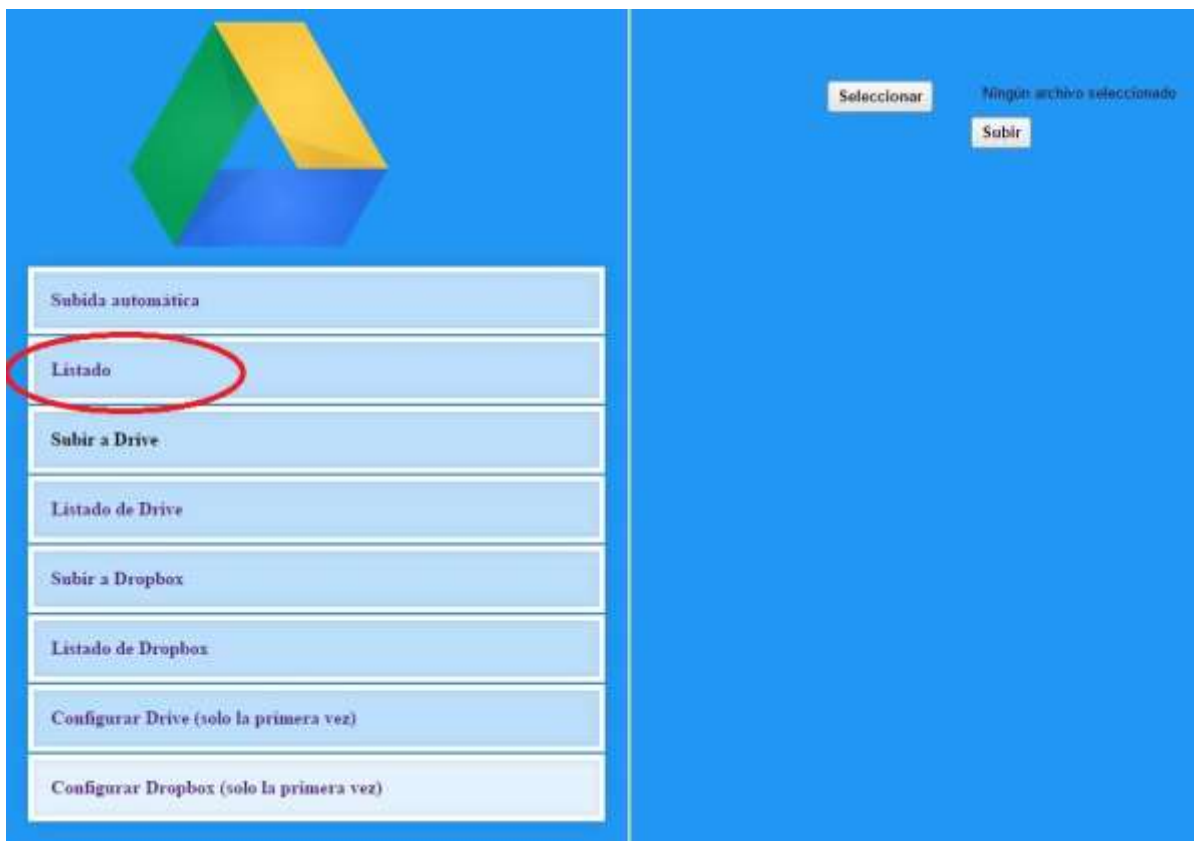
Tras esto se pulsa subir:



Y una vez realizado esto se sube el archivo y si todo ha salido correctamente aparecerá un mensaje confirmándolo.



- **Listado:** Para comprobar que es lo que se ha subido, está la opción listado. Con la opción listado podrá verse todo el contenido alojado tanto de Google Drive como de Dropbox, siendo completamente transparente para el usuario de donde es cada uno de los archivos



Una vez seleccionado el listado aparecerá en pantalla este:



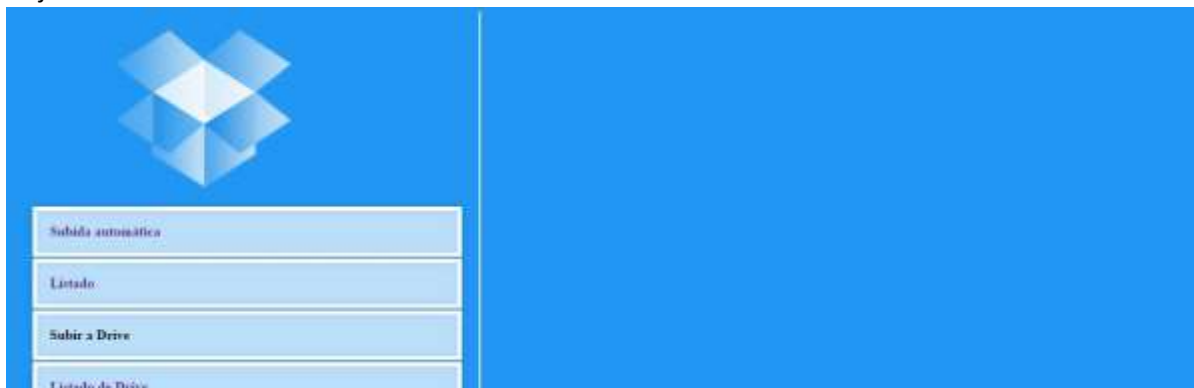
Si se pulsa sobre un elemento se realizara la misma opción que si se pulsara el archivo en el mismo servicio en el que se aloja:



Si se quiere eliminar un archivo, solo deberá de pulsarse sobre el icono de basura que tiene cada uno de los archivos en la lista:



Dejando en este caso en concreto la lista vacía:



- **Otras opciones:** El resto de opciones, son las mismas vistas arriba pero para un servicio en concreto, si por ejemplo solo nos interesa ver el listado de archivos alojados en Google Drive, pulsaremos sobre listado de Drive, igual que si queremos subir un archivo forzosamente en Google Drive pulsaremos en Subir a Drive.
- **Problemas conocidos:** Tras realizar varias pruebas de uso, existen dos problemas conocidos que son ajenos a este grupo de desarrollo:
  - **https:** En primer lugar, a la hora de realizar las pruebas, es muy importante no estar conectado con a través de https, ya que Flask tiene dificultades para trabajar con este protocolo dando muchas veces errores, como por ejemplo el mostrar la lista, que se queda cargando infinitamente. Para solucionar este problema, simplemente debe de usarse el protocolo http, funcionando a la perfección la aplicación.
  - **Cloud9:** El entorno de desarrollo en la nube cloud9, es muy potente, pero tiene un pequeño fallo y es que, el servidor requiere de “encenderse”, es decir, que cuando se pulsa en la pestaña Run Project dentro de Cloud9, todo funciona perfectamente, pero al cabo de un tiempo, el servidor se para automáticamente, no siendo la aplicación web funcional, por lo que debe de volver a pulsarse Run Project.

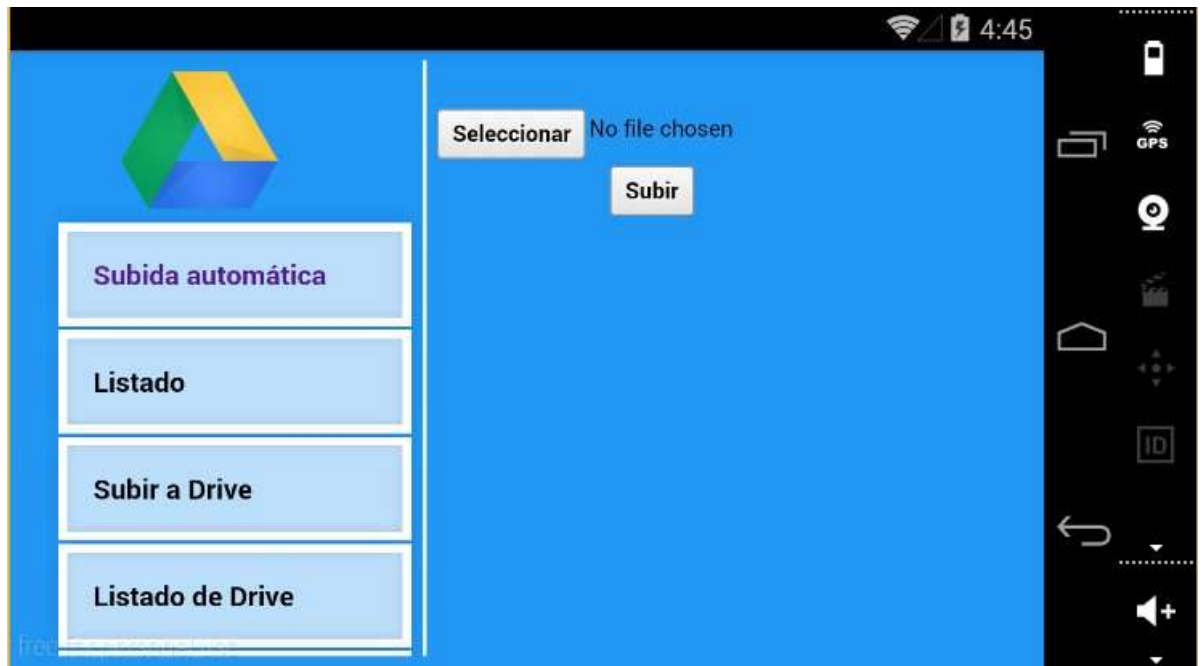
## 11. App Móvil y ampliaciones futuras.

Una vez que el desarrollo de la aplicación estaba avanzado, se procedió al despliegue de la web desde un servidor externo y fue entonces cuando se comenzaron a gestar nuevas ideas para ampliar la funcionalidad de la aplicación, habiendo sido posible implementar parte de ellas:

- **Aplicación para móviles:** Con los móviles que se disponen a día de hoy, siempre existe la posibilidad de que se quiera disponer de un archivo que este alojado en el móvil o en el PC y se quiera en el otro dispositivo. Además de esto, como toda la información de VIRTUALDISKSD se encuentra alojada en un servicio externo y nada de forma local, haría que los tan saturados a día de hoy móviles no necesitaran de ampliar su espacio con costosas memorias SD, por lo que nuestra aplicación podría ser perfecta para este tipo de usos.

Parte del equipo disponía de experiencia previa con el SO Android, por lo que esta plataforma fue la elegida para un primer acercamiento. La idea para portar la aplicación fue hacerlo todo lo más sencillo posible, teniendo en cuenta que no es un requisito para la asignatura y además de que se trataba de un trabajo de investigación, así que se decidió crear un WebView para Android, donde la aplicación se conecta a la web de cloud9 y hace uso de esta de forma nativa. Durante el desarrollo de la app, existió un problema, y es que se tuvo que implementar código nativo de Android para realizar la selección del archivo a subir. Una vez solucionado este problema, la aplicación quedo lista:

**Nota:** Para hacer capturas de pantalla se ha utilizado el emulador de Android Genymotion, siendo posible su utilización para probar el archivo APK alojado en Bitbucket para testear la aplicación.



- **Ampliaciones futuras:** Como toda aplicación, durante el desarrollo se ha priorizado terminar el núcleo del proyecto, dejándose atrás muchas características que nos hubiera gustado implementar, estas son algunas:
  - **Nuevos servicios:** Se han quedado fuera por cuestiones de tiempo servicios como Box, Mega, Amazon Cloud y otros, los cuales ampliarían el atractivo de la aplicación.
  - **Aplicación iOS:** El equipo priorizó la versión Android, puesto que los integrantes del mismo ya poseían experiencia programando en este SO, por lo que se dejó a un lado la versión de iOS, pero sería realmente atractivo el crear una app para este SO.

- **Reordenación:** Esta es quizás, la característica que más dolió dejar atrás, ya que se pretendía desde un primer momento que si un archivo era muy grande y no cabía en ninguno de los ficheros, se reordenaran estos para ver si existía alguna posibilidad de almacenar este archivo. Lamentablemente una vez más por cuestiones de tiempo, esta idea quedo en segundo plano, priorizando que todo quedase funcional y dejándola para una futura versión 2.0 de la aplicación.
- **Sustitución de Flask:** Debido a los numerosos problemas que ha originado este framework durante el desarrollo que se han descrito durante toda esta memoria, para un uso masivo en el futuro, sería más eficiente emplear otro framework para el uso de funcionalidades web como podría ser Django.

## 12. Histórico de Cambios.

Author	Commit	Message	Date
<a href="#">Emilio Bello Villanueva</a>		Version final de la memoria y apk Android.	
<a href="#">Juan Carlos de la Torre Macías</a>	<a href="#">e3e26eb</a>	Eliminación de README.MD	4 minutes ago
<a href="#">Juan Carlos de la Torre Macías</a>	<a href="#">0ae69ba</a>	README.md edited online with Bitbucket	7 minutes ago
<a href="#">Juan Carlos de la Torre Macías</a>	<a href="#">27b6db8</a>	README.md edited online with Bitbucket	9 minutes ago
<a href="#">Juan Carlos de la Torre Macías</a>	<a href="#">22272eb</a>	README.md edited online with Bitbucket	12 minutes ago
<a href="#">Juan Carlos de la Torre Macías</a>	<a href="#">3bf77a3</a>	README.md edited online with Bitbucket	13 minutes ago
<a href="#">Juan Carlos de la Torre Macías</a>	<a href="#">55df625</a>	README.MD deleted online with Bitbucket	16 minutes ago
<a href="#">Juan Carlos de la Torre Macías</a>	<a href="#">3998110</a>	Primera versión del README.MD	40 minutes ago
<a href="#">Juan Carlos de la Torre Macías</a>	<a href="#">8cb65cf</a>	Reorganización de ficheros en /docs	54 minutes ago
<a href="#">Manuel F.</a>	<a href="#">e7f0e1d</a> <a href="#">M</a>	<a href="#">Merge branch 'master' of https://bitbucket.org/filosofiayletras/pyfilesync</a>	9 hours ago
<a href="#">Manuel F.</a>	<a href="#">f839ebf</a>	Capturada la excepción para ficheros de tamaño 0 en Drive (ficheros compartidos).	9 hours ago
<a href="#">Juan Carlos de la Torre Macías</a>	<a href="#">3c0263e</a>	Revisión de la memoria	11 hours ago
<a href="#">Emilio Bello Villanueva</a>	<a href="#">1e12f5d</a> <a href="#">M</a>	<a href="#">Merge https://bitbucket.org/filosofiayletras/pyfilesync</a>	22 hours ago
<a href="#">Emilio Bello Villanueva</a>	<a href="#">5ca9524</a> <a href="#">M</a>	Version memoria 0.9 y añadiendo imagenes	22 hours ago
<a href="#">Juan Carlos de la Torre Macías</a>	<a href="#">b485cc7</a>	Comentarios sobre onedrive	23 hours ago
<a href="#">Emilio Bello Villanueva</a>	<a href="#">6466744</a>	Memoria version 0.2.1	2 days ago

<a href="#">Juan Carlos de la Torre Macías</a>	<a href="#">caba4b1</a>	Añadido problemas con OneDrive	3 days ago
<a href="#">Manuel F.</a>	<a href="#">e623ccc</a>	Idependizada la web de la API, añadido menú y configuración de servicios via AJAX.	4 days ago
<a href="#">Manuel F.</a>	<a href="#">bed58abM</a>	<a href="#">Merge branch 'master' of https://bitbucket.org/filosofiayletras/pyfilesync</a>	12/05/2015
<a href="#">Manuel F.</a>	<a href="#">09a5601</a>	Añadido botón de eliminar.	12/05/2015
<a href="#">Emilio Bello Villanueva</a>	<a href="#">136abd3</a>	Agregando nuevas secciones y completando informacion v0.2	12/05/2015
<a href="#">Emilio Bello Villanueva</a>	<a href="#">e2e61ea</a>	Actualizando la memoria v0.1	12/05/2015
<a href="#">Juan Carlos de la Torre Macías</a>	<a href="#">f369232</a>	Añadida definición de Roles	05/05/2015
<a href="#">Juan Carlos de la Torre Macías</a>	<a href="#">6,24E+11</a>	Finalizada la descripción de tareas	05/05/2015
<a href="#">Juan Carlos de la Torre Macías</a>	<a href="#">35f4d89</a>	Justificación y planificación en la memoria	05/05/2015
<a href="#">Juan Carlos de la Torre Macías</a>	<a href="#">da87746M</a>	<a href="#">Merge branch 'master' of https://bitbucket.org/filosofiayletras/pyfilesync</a>	05/05/2015
<a href="#">Juan Carlos de la Torre Macías</a>	<a href="#">aa33812</a>	Subida del diagrma de gantt	05/05/2015
<a href="#">jose manuel vidal jimenez</a>	<a href="#">7,84E+09</a>	Version 1 web total	05/05/2015
<a href="#">Manuel F.</a>	<a href="#">e144b92</a>	Añadido método global a la API para subida automática	05/05/2015
<a href="#">Manuel F.</a>	<a href="#">15d6f15</a>	Añadida carpeta uploads al tracking	03/05/2015
<a href="#">Manuel F.</a>	<a href="#">78247a4</a>	Arreglado el desastre del Jose. Dropbox y Drive funcionando	03/05/2015
<a href="#">jose manuel vidal jimenez</a>	<a href="#">3a10426</a>	Version Dropbox con error en la subida	03/05/2015
<a href="#">jose manuel vidal jimenez</a>	<a href="#">8f6d11a</a>	Añadido funcionamiento de Dropbox	02/05/2015
<a href="#">Manuel F.</a>	<a href="#">ff17ba8</a>	Mime-type automático para subida a Drive y añadida petición a la API para saber porcentaje de uso	02/05/2015
<a href="#">Manuel F.</a>	<a href="#">ab298d9</a>	Arreglado aspecto visual	02/05/2015
<a href="#">Manuel F.</a>	<a href="#">ba47674</a>	Commit inicial. Drive medio funcionando.	02/05/2015