



UNIVERSIDAD DE EXTREMADURA

ESCUELA POLITÉCNICA

Ingeniería Informática en Ingeniería del Software

CURSO 2019/20

Arquitectura Orientada a Servicios
Instalación de pre-requisitos para OAuth2.0

Alberto Mangut Bustamante(amangutb@alumnos.unex.es)
Carlos Delgado Guiberteau(cdelgadow@alumnos.unex.es)

Introducción y conceptos previos

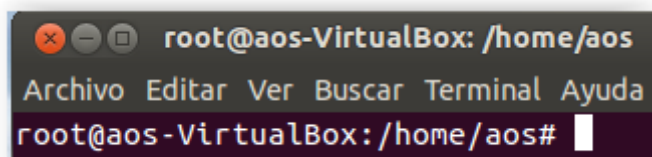
En esta guía procederemos a explicar la instalación de los distintos programas y librerías que serán necesarias para la clase teórico-práctica que tratará sobre **OAuth2.0**

Para la realización de los ejercicios usaremos **Python 3 y Flask**, este último nos servirá para la creación del servidor y despliegue efectivo de páginas web sobre el mismo.

Dado que disponemos de una máquina virtual ofrecida por los profesores de la asignatura, todas las instalaciones se realizarán sobre ella para evitar posibles problemas que puedan surgir durante las instalaciones, de esta forma todos trabajaremos sobre la misma distribución y la misma versión de Ubuntu.

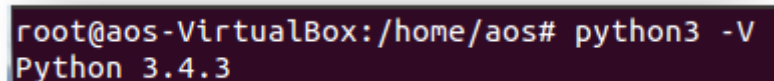
Instalación de Programas

Iniciamos la máquina virtual y abrimos una consola de comandos nueva. El primer paso será **logearnos** como **superusuarios** para evitar problemas derivados de la posible falta de permisos. Para ello ejecutamos el comando '**sudo su**' e introducimos la contraseña de la máquina virtual (por defecto '**aos**'). Para saber que estamos en modo **superusuario** debemos tener un **#** al final de la ruta en la que nos encontramos como sigue:



```
root@aos-VirtualBox: /home/aos
Archivo Editar Ver Buscar Terminal Ayuda
root@aos-VirtualBox: /home/aos#
```

Una vez que hemos accedido como usuario **root** procedemos a instalar los programas necesarios. Lo primero que debemos hacer será comprobar la instalación de **python3** en nuestra máquina virtual. Para ello ejecutamos el comando **python3 -V** y tenemos que obtener la siguiente salida:



```
root@aos-VirtualBox: /home/aos# python3 -V
Python 3.4.3
```

Como podemos observar de esta forma comprobamos que versión tenemos instalada de **Python** en nuestro dispositivo. Esta versión ya viene instalada en la máquina virtual por lo que todos deberíais de tener la misma versión.

El siguiente paso que debemos realizar será la instalación de **pip3**. Este programa nos permitirá instalar paquetes en nuestro entorno para poder usar correctamente los programas que vamos a realizar en **Python**.

Para instalarlo debemos ejecutar primeramente el comando **apt-get update** y esperar a que termine de ejecutarse.

```
root@aos-VirtualBox:/home/aos# apt-get update
Obj http://security.ubuntu.com trusty-security InRelease
Obj http://ppa.launchpad.net trusty InRelease
Ign http://es.archive.ubuntu.com trusty InRelease
Ign http://extras.ubuntu.com trusty InRelease
Obj http://es.archive.ubuntu.com trusty-updates InRelease
Obj http://security.ubuntu.com trusty-security/main Sources
Obj http://extras.ubuntu.com trusty Release.gpg
Obj http://ppa.launchpad.net trusty InRelease
Obj http://es.archive.ubuntu.com trusty-backports InRelease
Obj http://security.ubuntu.com trusty-security/restricted Sources
Obj http://extras.ubuntu.com trusty Release
Obj http://es.archive.ubuntu.com trusty Release.gpg
Obj http://security.ubuntu.com trusty-security/universe Sources
Obj http://extras.ubuntu.com trusty/main Sources
Obj http://ppa.launchpad.net trusty/main i386 Packages
Obj http://es.archive.ubuntu.com trusty-updates/main Sources
Obj http://security.ubuntu.com trusty-security/multiverse Sources
Obj http://extras.ubuntu.com trusty/main i386 Packages
Obj http://security.ubuntu.com trusty-security/main i386 Packages
Obj http://ppa.launchpad.net trusty/main Translation-en
Obj http://es.archive.ubuntu.com trusty-updates/restricted Sources
```

Una vez que volvemos a tener la consola lista para introducir comandos, el siguiente que debemos ejecutar es el siguiente: **apt-get -y install python3-pip**

En caso de que se muestre algún **mensaje de error** al ejecutar este comando, es posible que no tengamos conexión a internet en nuestra máquina virtual por lo que tendremos que ajustar esta configuración en la ventana de preferencias de nuestra máquina.

Si el comando se ha ejecutado correctamente podremos ejecutar el comando **pip3 -V** y obtener la siguiente salida:

```
root@aos-VirtualBox:/home/aos# pip3 -V
pip 1.5.4 from /usr/lib/python3/dist-packages (python 3.4)
```

De esta forma confirmaremos que la instalación ha sido satisfactoria. Ahora que tenemos instalado **pip** podremos proceder a instalar **Flask** para poder desplegar el servidor y probar nuestras páginas webs. Para instalarlo ejecutamos el comando **pip3 install Flask**

Dejamos que descargue todos los paquetes y deberíamos obtener el siguiente mensaje por consola:

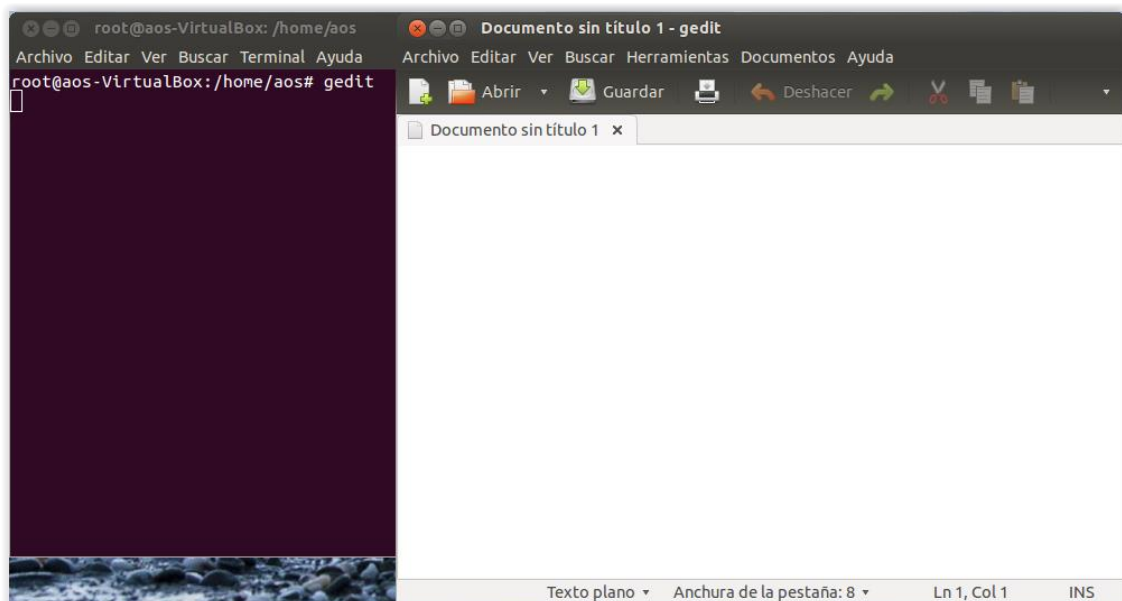
```

Running setup.py install for MarkupSafe
/usr/lib/python3.4/distutils/dist.py:260: UserWarning: Unknown distribution
option: 'python_requires'
  warnings.warn(msg)
/usr/lib/python3.4/distutils/dist.py:260: UserWarning: Unknown distribution
option: 'project_urls'
  warnings.warn(msg)

no previously-included directories found matching 'docs/_build'
warning: no previously-included files matching '*.py[co]' found anywhere in
distribution
building 'markupsafe._speedups' extension
i686-linux-gnu-gcc -pthread -DNDEBUG -g -fwrapv -O2 -Wall -Wstrict-prototype
s -g -fstack-protector --param=ssp-buffer-size=4 -Wformat -Werror=format-securit
y -D_FORTIFY_SOURCE=2 -fPIC -I/usr/include/python3.4m -c src/markupsafe/_speedup
s.c -o build/temp.linux-i686-3.4/src/markupsafe/_speedups.o
i686-linux-gnu-gcc -pthread -shared -Wl,-O1 -Wl,-Bsymbolic-functions -Wl,-Bs
ymbolic-functions -Wl,-z,relro -Wl,-Bsymbolic-functions -Wl,-z,relro -g -fstack-
protector --param=ssp-buffer-size=4 -Wformat -Werror=format-security -D_FORTIFY_
SOURCE=2 build/temp.linux-i686-3.4/src/markupsafe/_speedups.o -o build/lib.linux-
i686-3.4/markupsafe/_speedups.cpython-34m.so
Successfully installed flask Werkzeug Jinja2 itsdangerous click MarkupSafe
Cleaning up...
root@aos-VirtualBox:/home/aos#

```

Para comprobar la correcta instalación de **Flask**, vamos a crear una pequeña página web y desplegarla en el servidor. Ejecutamos el comando **gedit** para poder editar un nuevo archivo. Debemos poder ver el siguiente editor de texto en nuestra pantalla:



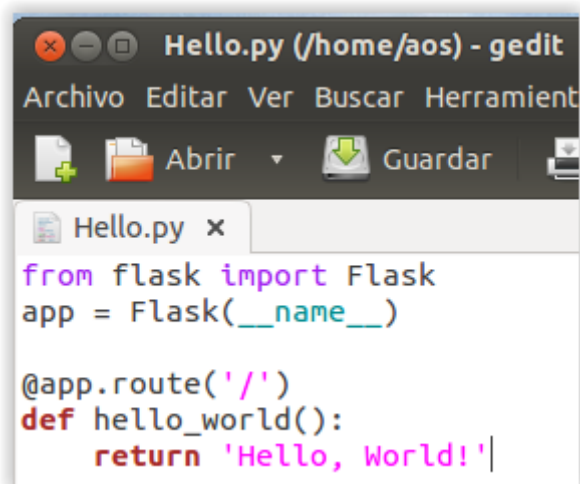
A continuación, copiamos el siguiente código en el nuevo documento y lo guardamos con el nombre **Hello.py**:

```

from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, World!'

```

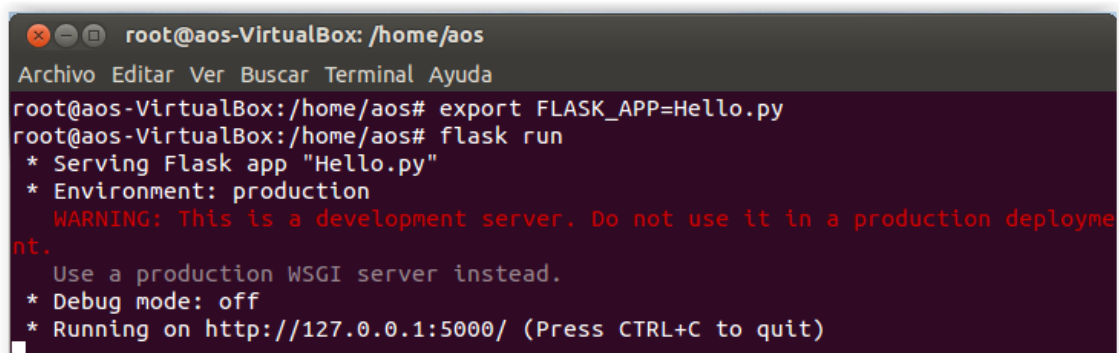


```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, World!'
```

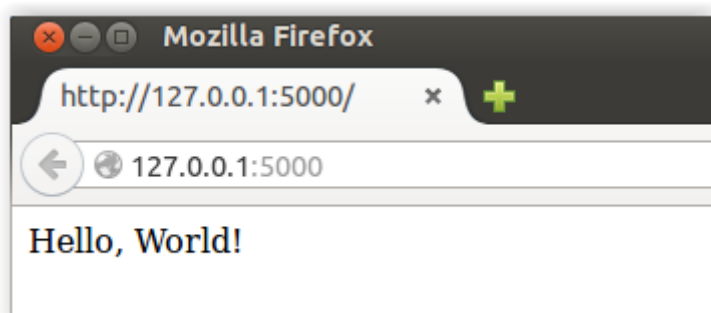
Este será nuestro primer programa, como podemos observar una vez que se despliegue el servidor y accedamos a la ruta **http://localhost:5000/** obtendremos el mensaje 'Hello, World!' por pantalla. Podemos cerrar la ventana y volver a la consola donde introduciremos el comando **export FLASK_APP=Hello.py** y finalmente ejecutamos el programa con el comando **flask run**.

Debemos obtener el siguiente mensaje por pantalla en el que se nos informa que el servidor está funcionando correctamente y podemos acceder a nuestra página web.



```
root@aos-VirtualBox: /home/aos
root@aos-VirtualBox:/home/aos# export FLASK_APP=Hello.py
root@aos-VirtualBox:/home/aos# flask run
* Serving Flask app "Hello.py"
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

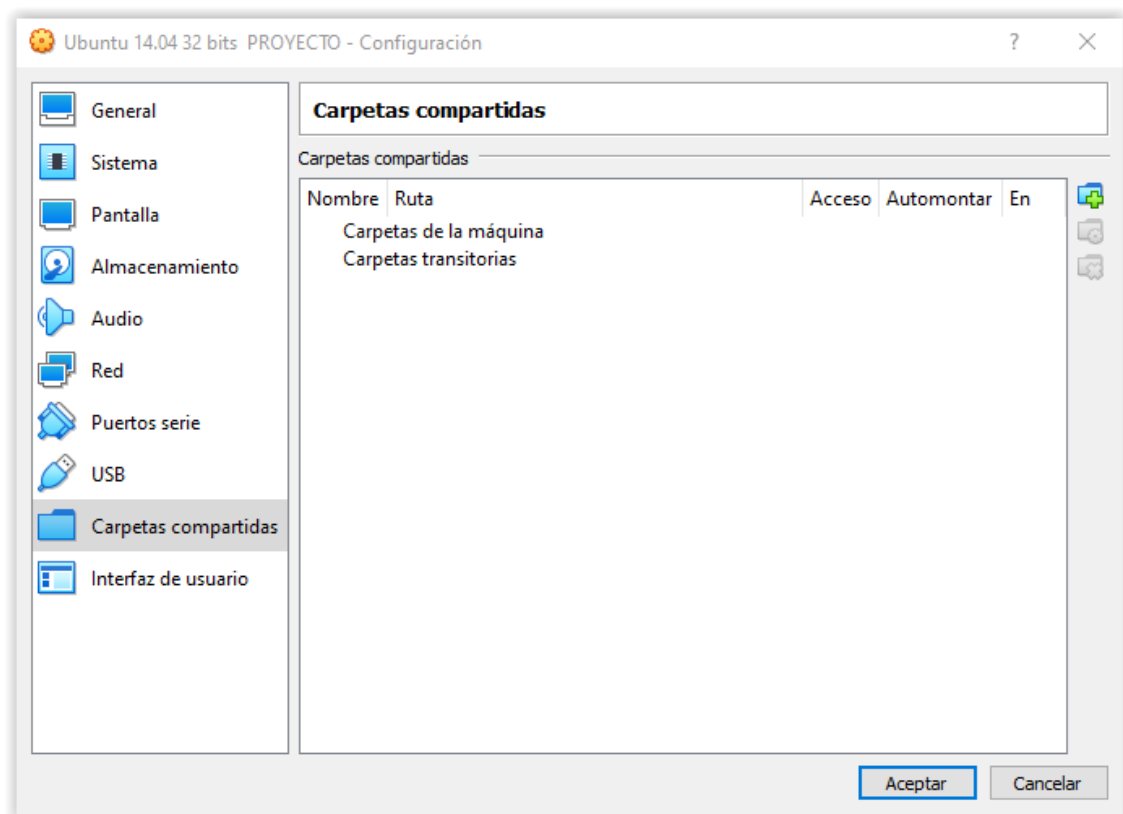
Podemos pulsar **CTRL+C** mientras hacemos **click** sobre la dirección **URL** para que se abra el navegador web y nos muestra la página web. Si todo ha ido correctamente debemos poder visualizar la siguiente página:



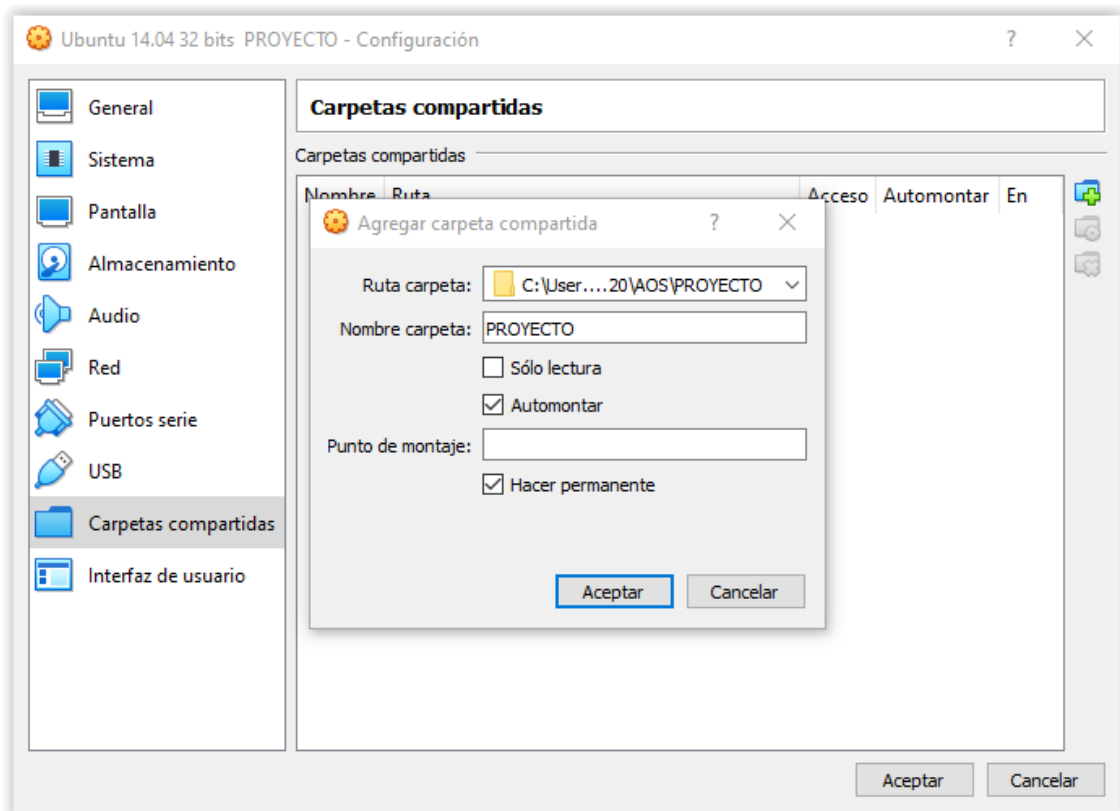
Con esto tendríamos configurado **Flask** para poder trabajar con el en clase. Es muy importante parar la ejecución del servidor haciendo uso de **CTRL + C** en la consola de comandos que tenemos abierta. De esta forma podremos ejecutar repetidas veces los programas que creamos sobre **Flask** y no tendremos la **URL** ocupada por otro programa.

Creación carpeta compartida

El siguiente paso será la creación de una carpeta compartida para poder usar nuestro sistema operativo e intercambiar ficheros con la máquina virtual de forma sencilla. Es un paso optativo pero recomendable para poder trabajar de forma ágil con los ficheros que descarguemos. Para ello nos situamos en la ruta del escritorio con `cd Escritorio`. Una vez que estamos en la ruta procedemos a crear un nuevo directorio llamado **carpeta_compartida**. Podemos hacerlo ejecutando el comando `mkdir carpeta_compartida`. Podremos ver esta carpeta en nuestro escritorio y acceder a ella. A continuación, accedemos a la configuración de la máquina virtual sobre la que estamos trabajando y nos desplazamos hasta la pestaña de Carpetas Compartidas



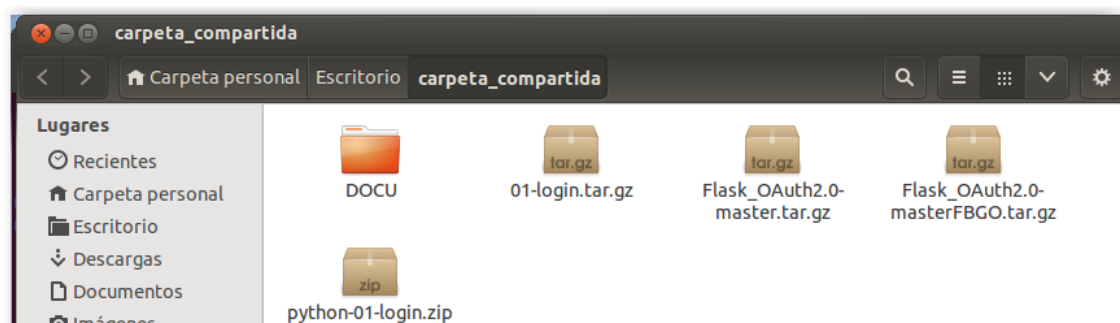
Una vez que nos encontramos esta pantalla procedemos a **crear una nueva carpeta** en nuestro **SO Host** que usaremos para intercambiar ficheros. Una vez que la hemos creado volvemos a la ventana de carpetas compartidas y pulsamos sobre el icono de añadir nueva carpeta e indicamos la ruta de la carpeta que acabamos de crear en nuestro host. Finalmente marcamos las casillas de **Automontar** y **Hacer permanente** y clickamos en Aceptar



Finalmente volvemos a nuestra máquina virtual y ejecutamos el comando **mount -t vboxsf nombre_carpeta_host carpeta_compartida**

```
root@aos-VirtualBox: /home/aos/Escritorio
Archivo Editar Ver Buscar Terminal Ayuda
root@aos-VirtualBox:/home/aos/Escritorio# mount -t vboxsf PROYECTO carpeta_compartida
```

Si todo ha ido correctamente, podríamos abrir la carpeta compartida, añadir algún archivo a ella y verlo en nuestra máquina virtual.



Instalación de Paquetes

Por último, vamos a proceder con la instalación de los paquetes que necesitaremos usar en las aplicaciones que vamos a crear. En Python podemos crear archivos de texto en la que incluimos los paquetes necesarios para ejecutar una aplicación concreta y se nos instalarán automáticamente ejecutando un comando por consola.

Hemos preparado unos ejemplos prácticos a realizar durante las prácticas que requieren de paquetes concretos por lo que para agilizar la clase vamos a descargarlos del repositorio que hemos creado. Este repositorio se encuentra en el siguiente enlace:

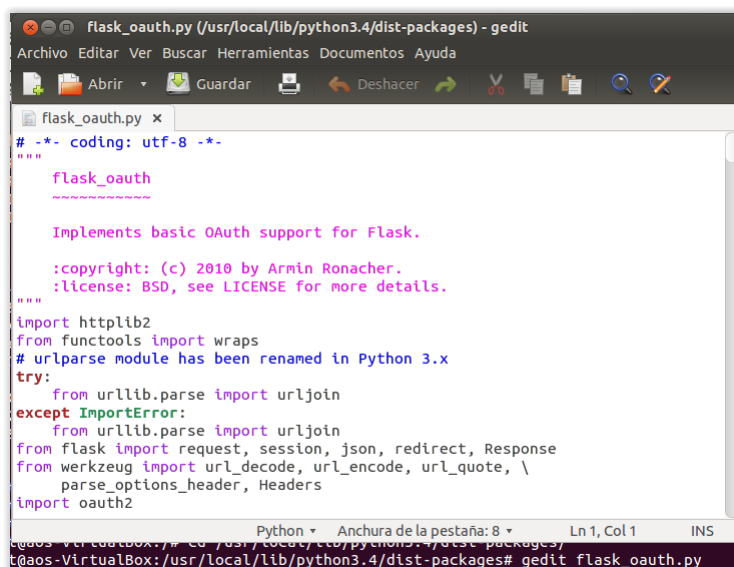
<https://github.com/albertomangut/instalacion>

Este repositorio incluye los archivos que serán necesarios en este pdf. Uno de los archivos es **requirements.txt** en el que incluimos los paquetes que vamos a usar por lo que procedemos a instalarlos en nuestro entorno y el otro es un archivo de Python que se explicará su uso en apartados posteriores de esta guía.

El fichero se encuentra en la carpeta raíz del proyecto, dentro de la carpeta **OAuth**. Debemos dirigirnos hasta este directorio en la consola y una vez que nos encontramos en el mismo directorio que el fichero ejecutamos el comando **pip3 install -r requirements.txt**

Dejamos que se instalen todas las librerías y ya tendríamos listos los paquetes necesarios para ejecutar los programas.

El último paso consiste en modificar un fichero de **Python3** dado que se encuentra deprecado con respecto a la versión en la que se ejecuta. Este fichero se ha incluido también en el repositorio y se llama **flask_oauth.py**. Debemos abrir este fichero y copiar todo su contenido, una vez que lo hemos hecho volvemos a la consola de comandos y nos aseguramos que estamos en modo **root**, ejecutamos el siguiente comando para dirigirnos a la carpeta en cuestión **cd /usr/local/lib/python3.4/dist-packages/** .Una vez que nos encontramos en esta carpeta escribimos **gedit flask_oauth.py** y se nos abrirá en el editor de texto el fichero que queremos modificar.



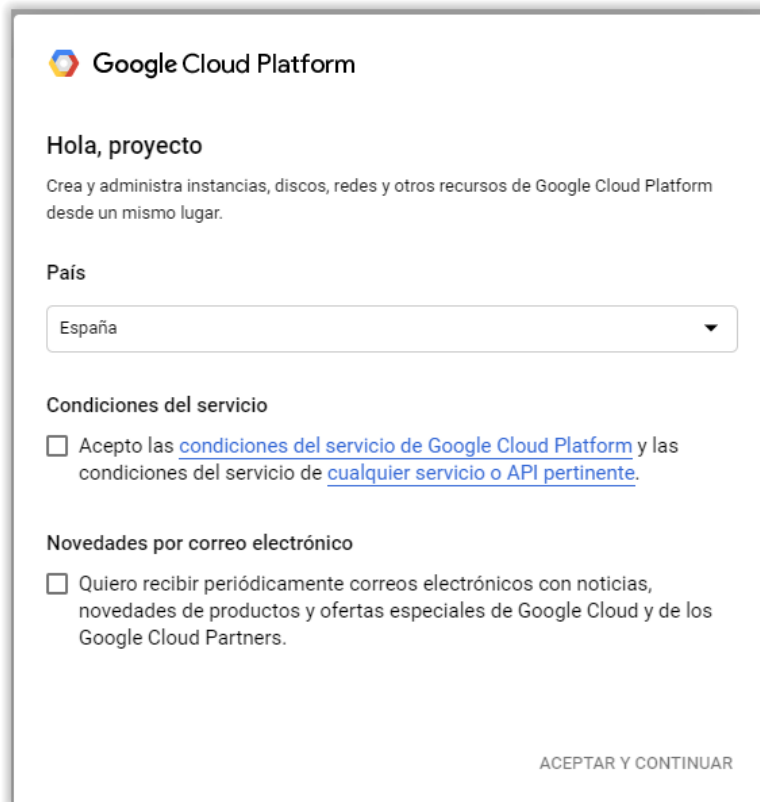
```
flask_oauth.py (/usr/local/lib/python3.4/dist-packages) - gedit
Archivo Editar Ver Buscar Herramientas Documentos Ayuda
flask_oauth.py x
# -*- coding: utf-8 -*-
"""
    flask_oauth
    Implements basic OAuth support for Flask.
    :copyright: (c) 2010 by Armin Ronacher.
    :license: BSD, see LICENSE for more details.
"""
import http-lib2
from functools import wraps
# url-lib module has been renamed in Python 3.x
try:
    from url-lib.parse import url-join
except ImportError:
    from url-lib.parse import url-join
from flask import request, session, json, redirect, Response
from werkzeug import url-decode, url-encode, url-quote, \
    parse_options_header, Headers
import oauth2

Python Python 3.4.3
Anchura de la pestaña: 8
Ln 1, Col 1
INS
@aos-VirtualBox: /usr/local/lib/python3.4/dist-packages# gedit flask_oauth.py
```


Seleccionamos todo el contenido y lo sustituimos por el contenido que hemos copiado anteriormente y guardamos. Llegados a este punto disponemos de todos los requisitos necesarios para llevar a cabo los ejercicios y poder ejecutar los scripts.

Creación de APIs

El último paso será la creación de las distintas cuentas en las plataformas que nos permitirán crear APIs. Empezaremos por la creación de una cuenta en Google developers. Para ello accedemos al siguiente [enlace](#) donde nos pedirá que introduzcamos el correo electrónico con el que queremos crear la nueva cuenta. Lo introducimos y nos mostrará la siguiente ventana



The screenshot shows the Google Cloud Platform account creation interface. At the top, the Google Cloud Platform logo is displayed. Below it, the text 'Hola, proyecto' is followed by a description: 'Crea y administra instancias, discos, redes y otros recursos de Google Cloud Platform desde un mismo lugar.' A 'País' (Country) dropdown menu is set to 'España'. Under 'Condiciones del servicio' (Service conditions), there is a checkbox for accepting the Google Cloud Platform terms and any relevant API terms, which is currently unchecked. Below this, under 'Novedades por correo electrónico' (Email newsletters), there is a checkbox for receiving periodic emails with news, product updates, and special offers from Google Cloud and its partners, also unchecked. At the bottom right, there is a button labeled 'ACEPTAR Y CONTINUAR'.

Leemos las condiciones y si estamos de acuerdo las aceptamos y le damos a continuar. Nos dirigimos hasta la pestaña de biblioteca en el menú lateral y pulsamos sobre crear.

Este botón nos redirigirá a una nueva pantalla en la que añadimos el nombre del proyecto, ejemplo, proyectoaas, y le damos a crear. Veremos que en la esquina superior izquierda ahora nos aparece un nuevo proyecto.

API APIs y servicios	Pantalla de consentimiento de OAuth
<div> <div>⚙️</div> <div>Panel de control</div> </div> <div> <div>📖</div> <div>Biblioteca</div> </div> <div> <div>🔑</div> <div>Credenciales</div> </div> <div> <div>🔗</div> <div>Pantalla de consentimiento ...</div> </div> <div> <div>✅</div> <div>Verificación del dominio</div> </div> <div> <div>⚙️</div> <div>Acuerdos de uso de páginas</div> </div>	<p>En la pantalla de consentimiento, los usuarios deciden, antes de autenticarse, si quieren conceder acceso a sus datos privados. En ella también se incluyen enlaces a las condiciones del servicio y a la política de privacidad. En esta página se configura la pantalla de consentimiento de todas las aplicaciones del proyecto.</p> <p>Estado de verificación Sin publicar</p> <p>Nombre de aplicación <small>?</small> Nombre de la aplicación que solicita el consentimiento</p> <input type="text" value="proyectoaoos"/>

El siguiente paso será dirigirnos a la pestaña 'pantalla de consentimiento' donde podemos definir el nombre de proyecto, dominios autorizados... e indicamos el nombre del proyecto como 'proyectoaoos' y guardamos.

Ahora nos dirigimos a la pestaña de credenciales y pulsamos sobre crear credenciales

API APIs y servicios	Credenciales
<div> <div>⚙️</div> <div>Panel de control</div> </div> <div> <div>📖</div> <div>Biblioteca</div> </div> <div> <div>🔑</div> <div>Credenciales</div> </div> <div> <div>🔗</div> <div>Pantalla de consentimiento ...</div> </div> <div> <div>✅</div> <div>Verificación del dominio</div> </div> <div> <div>⚙️</div> <div>Acuerdos de uso de páginas</div> </div>	<div> <div>APIs</div> <div>Credenciales</div> </div> <p>Necesitas credenciales para acceder a las API. Activa las API que quieras usar y, a continuación, crea las credenciales que exijan. Según la API, hacen falta una clave de API, una cuenta de servicio o un ID de cliente de OAuth 2.0. Para obtener más información, consulta la documentación de autenticación.</p> <div> <div>Crear credenciales</div> </div>

Se nos desplegará un menú en el que elegimos ID de cliente de OAuth y en la siguiente pantalla pulsamos sobre Web e indicamos el nombre, por ejemplo OAuth, y en Authorized JavaScript añadimos el dominio desde el que vamos a realizar las peticiones que en nuestro caso serán dos: <http://127.0.0.1:5000> y <http://localhost:5000>

En Authorized redirect URIs debemos añadir las URLs a los que se van a realizar los callbacks por lo que las añadimos, en nuestro caso serán <http://127.0.0.1:5000/oauth2callback>

← Crear ID de cliente de OAuth

Puedes usar un ID de cliente de OAuth 2.0 para generar tokens de acceso para las aplicaciones que utilicen el protocolo OAuth 2.0 para llamar a las API de Google. Los tokens contienen un identificador único. Para obtener más información, consulta [este artículo sobre cómo configurar OAuth 2.0](#).

Tipo de aplicación

☒ Web

☐ Android [Más información](#)

☐ Chrome [Más información](#)

☐ iOS [Más información](#)

☐ Otro

Nombre ?

OAuth

Restricciones

Introduce los orígenes de JavaScript, los URI de redirección o ambos. [Más información](#)

Debes añadir los dominios de origen y redirección a la lista Dominios autorizados de la [configuración de autorización de OAuth](#).

Orígenes de JavaScript autorizados

Para usarse en las solicitudes desde un navegador. Se trata del URI de origen de la aplicación cliente. No puede contener caracteres comodín (https://*.example.com) ni una ruta (<https://example.com/subdir>). Si utilizas un puerto que no sea estándar, deberás incluirlo en el URI de origen.

<http://127.0.0.1:5000>

<http://localhost:5000>

Introduce el dominio y pulsa Intro para añadirlo

URLs de redirección autorizadas

Para usarse con las solicitudes de un servidor web. Es la ruta de la aplicación a la que se redirecciona a los usuarios después de autenticarse en Google. A dicha ruta se añadirá el código de autorización de acceso. Debe tener un protocolo. No puede incluir fragmentos de URL ni rutas relativas. No puede ser una dirección IP pública.

<http://127.0.0.1:5000/oauth2callback>

<http://localhost:5000/callback/google>

Introduce el dominio y pulsa Intro para añadirlo

Crear **Cancelar**

Nos debe quedar tal y como se muestra en la imagen. Llegados a este punto pulsamos en crear y ya tendríamos la API de Google lista para recibir peticiones.

La siguiente API que debemos crear será en la siguiente [página](#) . Una vez que hemos accedido pulsamos sobre SIGN UP. Podemos registrarnos con nuestra de Google si queremos por lo que procedemos a realizar el registro con la opción que más nos interese. En la siguiente pantalla introducimos el nombre del proyecto, debe ser un nombre que no esté ya solicitado por lo que ponemos el que queramos y seleccionamos EU como región.

TENANT DOMAIN


proyectoaos19


✓


.eu.auth0.com

To help you easily explore our product, we've selected a tenant domain name for you. Although you can't rename a tenant, you can always add more tenants to your account (for staging or production environments) later.

REGION

 AU

 EU

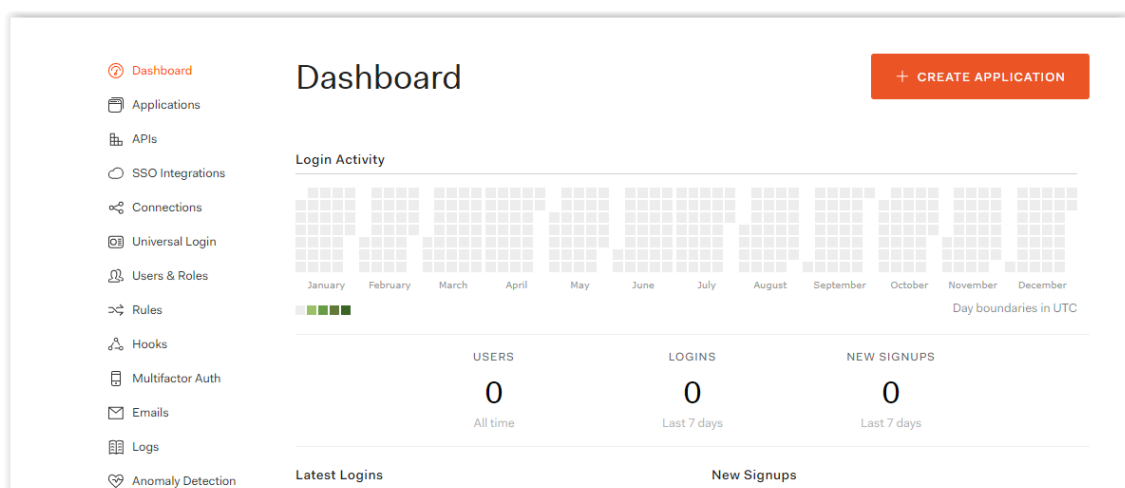
 US

We can host all of your data in any of these regions.
Useful if you need to comply with EU Data Protection Directive

NEXT

Pulsamos en siguiente y seleccionamos Personal, en role Developer y en main challenge Add Auth to my app y pulsamos en crear la cuenta.

Llegados a este punto ya tendremos nuestra cuenta de prueba en Auth0 para poder crear nuestras apis. A continuación, vamos a crear nuestra primera aplicación pulsando sobre el botón de CREATE APPLICATION



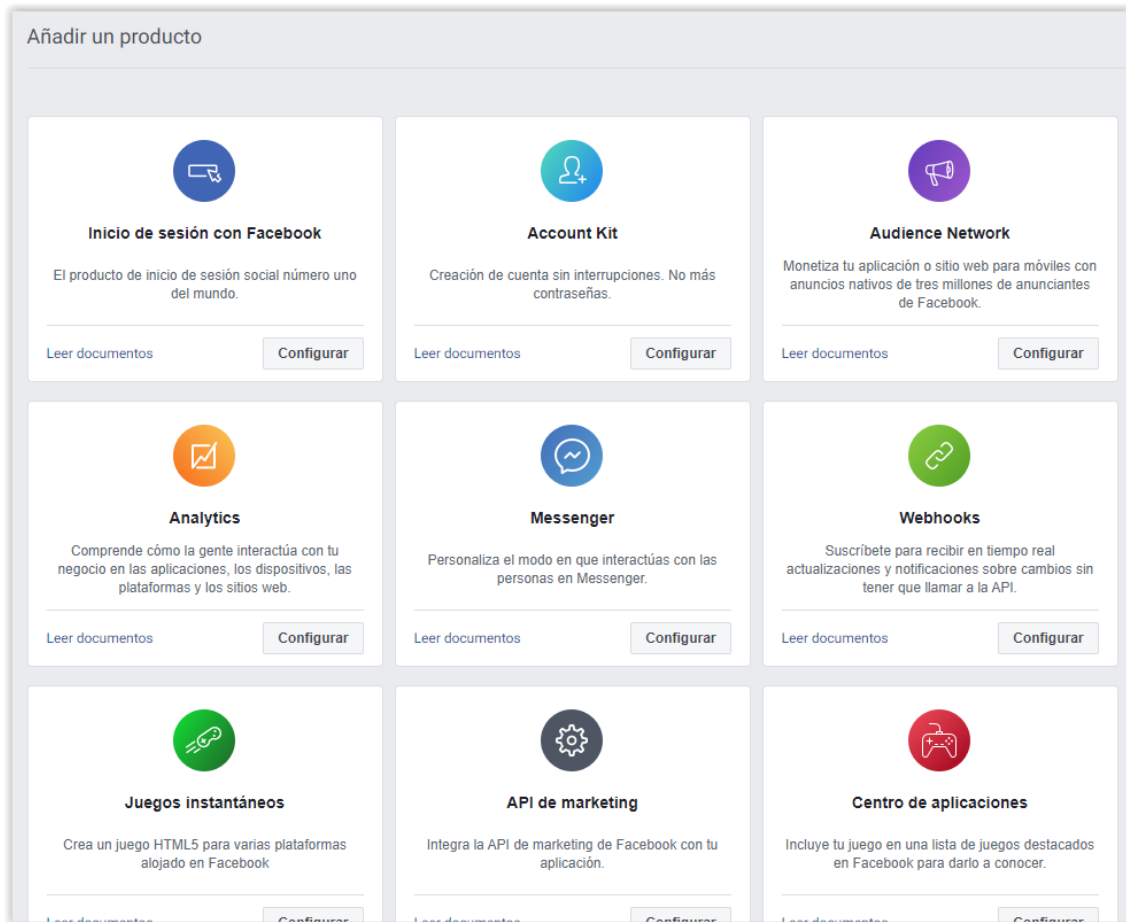
En el nombre del proyecto podemos poner el que queramos, por ejemplo, proyectoaos y el tipo de aplicación debe ser Regular Web Applications. Pulsamos en crear y ya tendremos nuestro primer proyecto creado. Por último, en la pestaña de Settings de la aplicación que hemos creado veremos el apartado Allowed Callback URLs en el que tenemos que añadir el siguiente: <http://127.0.0.1:5000/callback/auth0> y en Allowed Web Origins <http://127.0.0.1:5000> junto con Allowed Logout URLs <http://127.0.0.1:5000>. Guardamos los cambios y ya tendríamos Auth0 configurado para ser usado.

La última página en la que nos debemos crear una cuenta será en Facebook Developers, podemos acceder a esta página desde el siguiente [enlace](#). Debemos logearnos con nuestra cuenta de Facebook y una vez que estamos logeados nos redirigirá a la página principal de Facebook Developers. El siguiente paso será crear una nueva aplicación en la pestaña superior derecha.



Clickamos en crear una aplicación y se nos abrirá una ventana en la que debemos indicar el nombre del proyecto y un correo de contacto. Pulsamos en crear identificador de la aplicación

y se nos abrirá la consola de desarrollo de FB.



Pulsamos sobre Inicio de sesión con Facebook y seleccionamos WEB. Indicamos la URL del sitio web que será `http://localhost:5000/` y pulsamos en siguiente. Ya no tendríamos que configurar nada más en esta ventana. El resto de configuración necesaria será explicada en clase.

Cualquier problema que pueda encontrarse durante la realización de las instalaciones podéis enviarnos un correo a **Alberto**(amangutb@alumnos.unex.es) o **Carlos**(cdelgadow@alumnos.unex.es) y lo trataremos de resolver.