# Gestión de la Información en la Web
# Curso 2024-25
# Práctica 10 – Autenticación delegada

**Fecha de entrega: domingo 15 de diciembre de 2024, 23:55h**

**Entrega de la práctica**
La entrega de la práctica se realizará a través del Campus Virtual de la asignatura mediante un fichero **grupoXX.zip** donde **XX** es el número de grupo. Este ZIP contendrá un fichero de nombre `autenticacion_delegada.py` con el código del servidor web, cuyo esqueleto se puede descargar del Campus Virtual. Además del servidor web, el fichero ZIP contendrá las plantillas necesarias para mostrar los datos adecuadamente (si las habéis utilizado).

**Lenguaje de programación**
**Python 3.11** o superior.

**Evaluación**
Además de la corrección funcional, se valorará la **calidad, concisión y claridad del código**, la incorporación de **comentarios** explicativos, su **eficiencia** tanto en tiempo como en memoria y la puntuación obtenida en Pylint.

**Declaración de autoría e integridad**
Todos los ficheros entregados contendrán una cabecera en la que se indique la asignatura, la práctica, el grupo y los autores. Esta cabecera también contendrá la siguiente declaración de integridad:

> *Declaramos que esta solución es fruto exclusivamente de nuestro trabajo personal. No hemos sido ayudados por ninguna otra persona o sistema automático ni hemos obtenido la solución de fuentes externas, y tampoco hemos compartido nuestra solución con otras personas de manera directa o indirecta. Declaramos además que no hemos realizado de manera deshonesta ninguna otra actividad que pueda mejorar nuestros resultados ni perjudicar los resultados de los demás.*

**No se corregirá ningún fichero que no venga acompañado de dicha cabecera**.

---

## Autenticación delegada utilizando Google

En esta práctica vamos a utilizar el API de Google para autenticar usuarios. Únicamente vamos a implementar una prueba de concepto, por lo que los pasos a seguir serán:

1. Darse de alta en la Consola de Google Cloud para configurar la aplicación y obtener los credenciales de la aplicación web. Podéis encontrar más detalles en la sección «Crear el proyecto y los credenciales» más adelante.
2. A la hora de autenticar un usuario, redirigirlo a Google con los parámetros adecuados. **Importante**: debéis usar `response_type=code`.
3. Recibir la petición del usuario con el código temporal generado por Google y canjearlo por un id_token.
4. Extraer el e-mail del usuario incluido en el id_token (JWT en Base64 firmado por Google) y devolver una página de bienvenida. Como el id_token proviene directamente de Google mediante HTTPS no es necesario que validéis la firma del JWT, es decir:
   - podéis acceder directamente al *payload* del JWT y decodificarlo usando Base64. Tened cuidado porque es posible que debáis rellenar el *padding* (`https://en.wikipedia.org/wiki/Base64#Output_padding`) antes de invocar a las funciones Python que decodifican Base64
   - podéis utilizar el extremo `tokeninfo` de Google para validar y decodificar el JWT

Aunque existen distintas bibliotecas con funciones que facilitan la autenticación delegada con distintas redes sociales, el objetivo de esta práctica es realizar todas las fases de manera *manual*, sin utilizar APIs ni funciones externas sino realizando las peticiones HTTP adecuadas. Seguiremos el esqueleto que podéis descargar del Campus Virtual, llamado `pr10_skel.py`. Este fichero contiene un servidor web que responde a peticiones **GET** en dos rutas:

- /login_google
  Genera una página HTML con un enlace o un botón que redirige al usuario a la página de autenticación delegada de Google. Esta petición debe contener las credenciales necesarias de nuestra aplicación web.
- /token
  Recibe la petición del usuario redirigida desde Google con el código temporal que deberemos canjear por un id_token. Por tanto, en las credenciales de nuestra aplicación deberemos configurar el redirect_uri a `http://localhost:5000/token` tal y como aparece en el esqueleto. El id_token que obtendremos será un JWT firmado por Google con la información del usuario. Como este JWT lo hemos obtenido a través de una conexión HTTPS con Google no es necesario validar la firma, únicamente extraer la dirección de e-mail. Finalmente se generará una página HTML de bienvenida con el mensaje `Bienvenido <e-mail>`.

## Crear el proyecto y los credenciales

Para crear un proyecto que nos permita autenticar usuarios a través de Google debemos seguir los siguientes pasos:

1. Acceder a la consola de Google Cloud (`https://console.cloud.google.com` con la **cuenta la UCM**. Llegaréis a una página similar a la que aparece en la figura 1.

2. Crear un proyecto dentro de la organización `ucm.es`, rellenando los campos que se ven en la figura 2. Podéis elegir cualquier otro nombre de proyecto que queráis.
3. En el apartado de «APIs y servicios → Credenciales» configurar la «pantalla de consentimiento». En el tipo de usuario elegiremos **Interno** para poder autenticar únicamente usuarios UCM. En la ventana «Información de la aplicación» introducís un nombre para la aplicación (puede ser el mismo que el nombre del proyecto) y ponéis vuestro e-mail UCM tanto en «Correo electrónico de asistencia del usuario» como en «Información de contacto del desarrollador», dejando el **resto de campos en blanco**. Al llegar a la ventana de «Permisos» debéis agregar únicamente el permiso no sensible **openid**, tal y como se ve en la figura 3. En la ventana final de «Información opcional» podéis dejar todos los campos en blanco.
4. Crear los **credenciales** de la aplicación. Para ello pulsad el botón «Crear credenciales» y elegid «ID de cliente de OAuth». En la ventana de configuración elegid «Aplicación web» como tipo de aplicación y poned un nombre para el cliente de OAuth (puede ser el mismo que el proyecto y la apliación), tal y como se ve en la figura 4. **MUY IMPORTANTE: en el apartado «URI de redireccionamiento autorizados» debéis añadir la URI** `http://localhost:5000/token`. Si no la añadís u os equivocáis al escribirla, todo el proceso de autenticación fallará.
5. Una vez creados los credenciales, se os mostrará el «ID de cliente» y el «Secreto de cliente» que debéis anotar y copiar en vuestra aplicación Flask. Si en algún momento las perdéis siempre podréis recuperarlas descargando un fichero JSON con todos los datos desde la ventana de «Credenciales».

## Importante

- Para obtener la **máxima calificación** en esta práctica es necesario:
  1. utilizar el **documento de descubrimiento de Google** durante la ejecución de vuestro servidor para obtener las rutas actualizadas que están involucradas en el proceso de autenticación (es decir, esas rutas no pueden estar «atornilladas» en el código Python).
  2. crear un token antifalsificación (CSRF) y validarlo al recibir la petición del usuario tras autenticarse en Google
- Para probar que la autenticación delegada tiene éxito debéis conectaros con vuestro navegador a `http://localhost:5000/login_google` e iniciar el proceso. **No lo hagáis desde `http://127.0.0.1:5000` porque aunque esa es la IP de `localhost`, el navegador los tratará como *hosts* diferentes de cara a una sesión de navegación.**

## Referencias

- Detalles paso a paso de las fases de autenticación delegada en Google usando *OpenID Connect* (rutas, peticiones, parámetros involucrados):
  `https://developers.google.com/identity/openid-connect/openid-connect#server-flow`
- Consola del desarrollador de Google para dar de alta la aplicación, configurar los parámetros básicos y obtener los credenciales:
  `https://console.cloud.google.com`
- Documento de descubrimiento de Google para conocer las rutas involucradas en la autenticación delegada:
  `https://accounts.google.com/.well-known/openid-configuration`

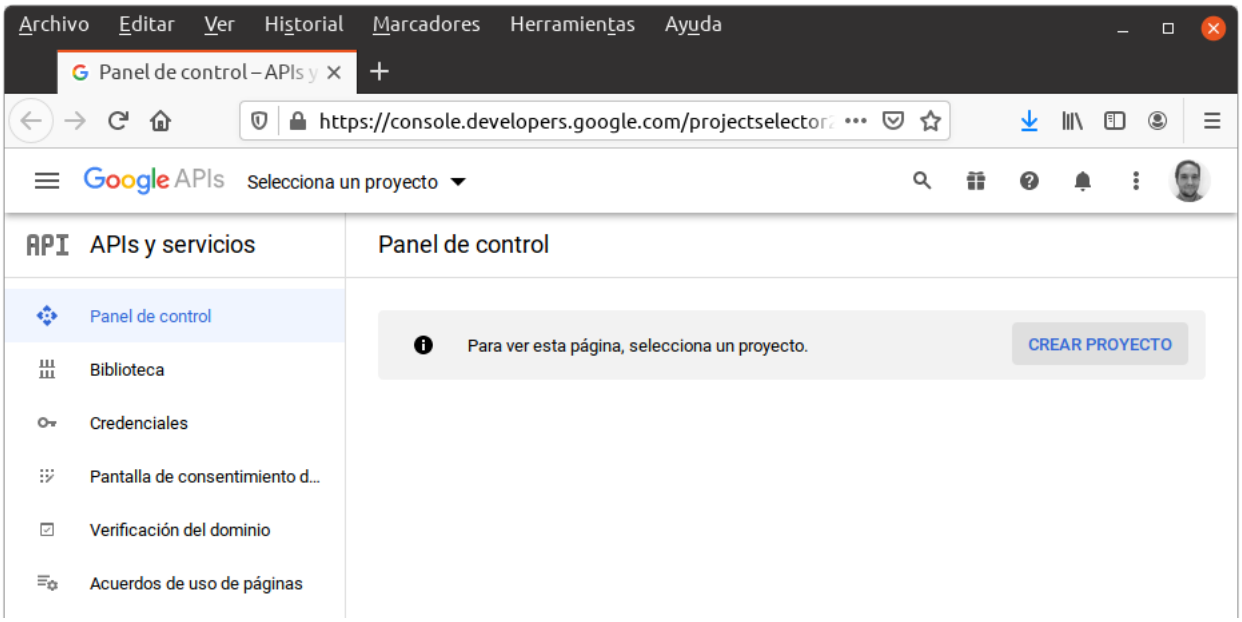


Figura 1: Consola de desarrolladores Google

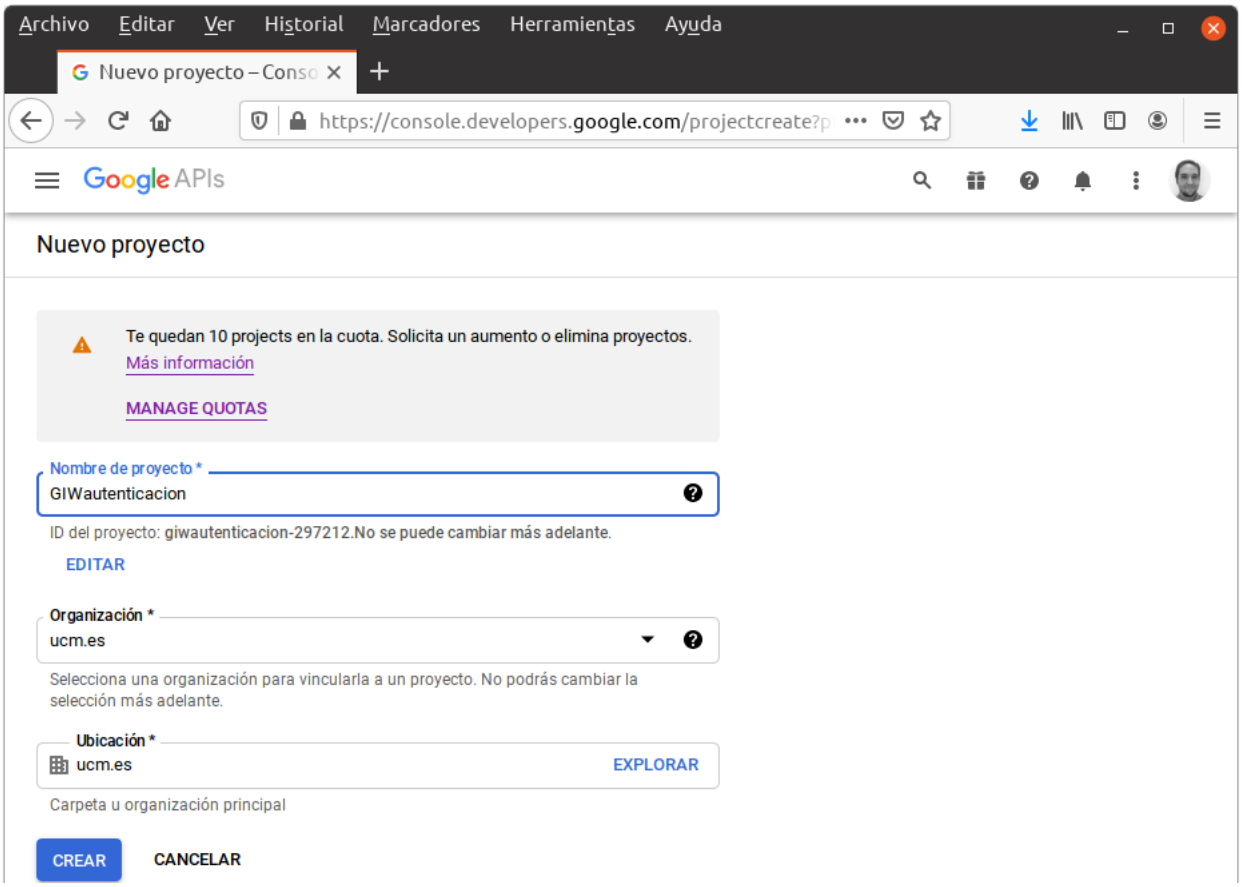


Figura 2: Creación de proyecto

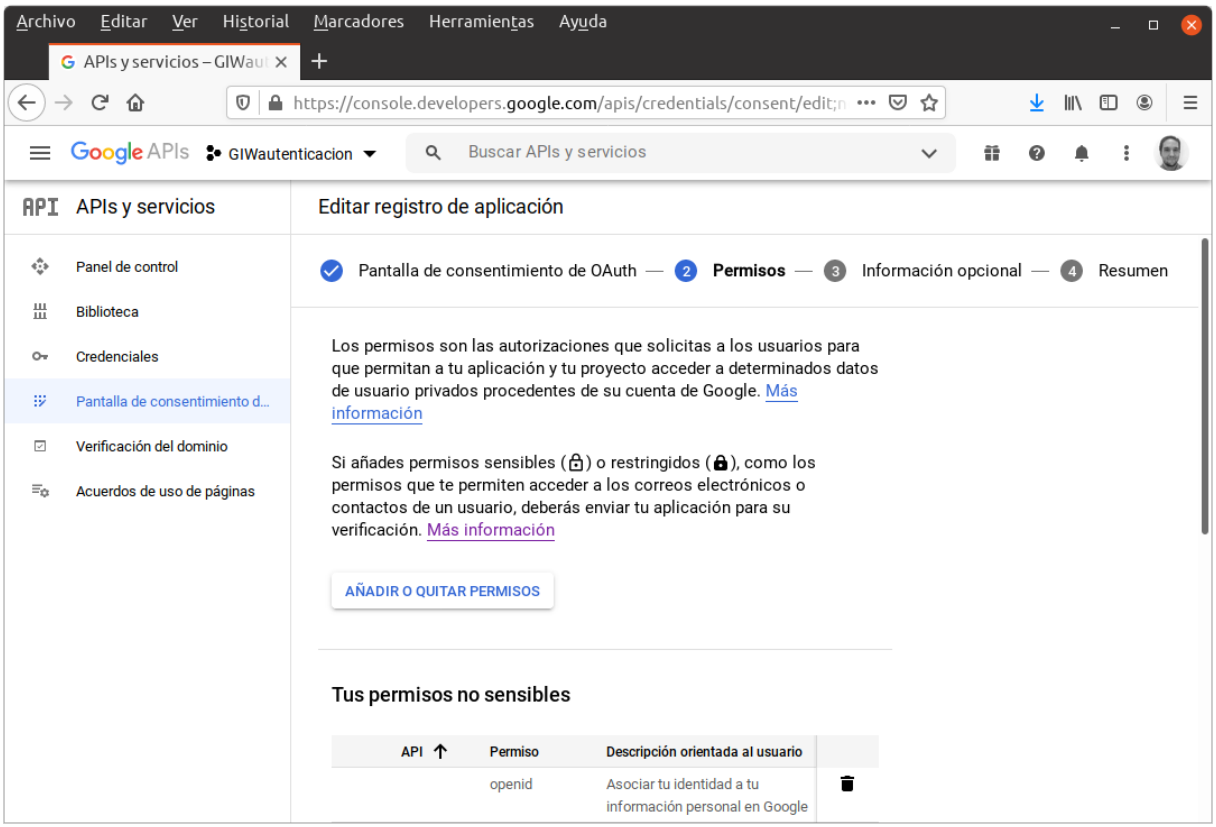


Figura 3: Permisos del proyecto

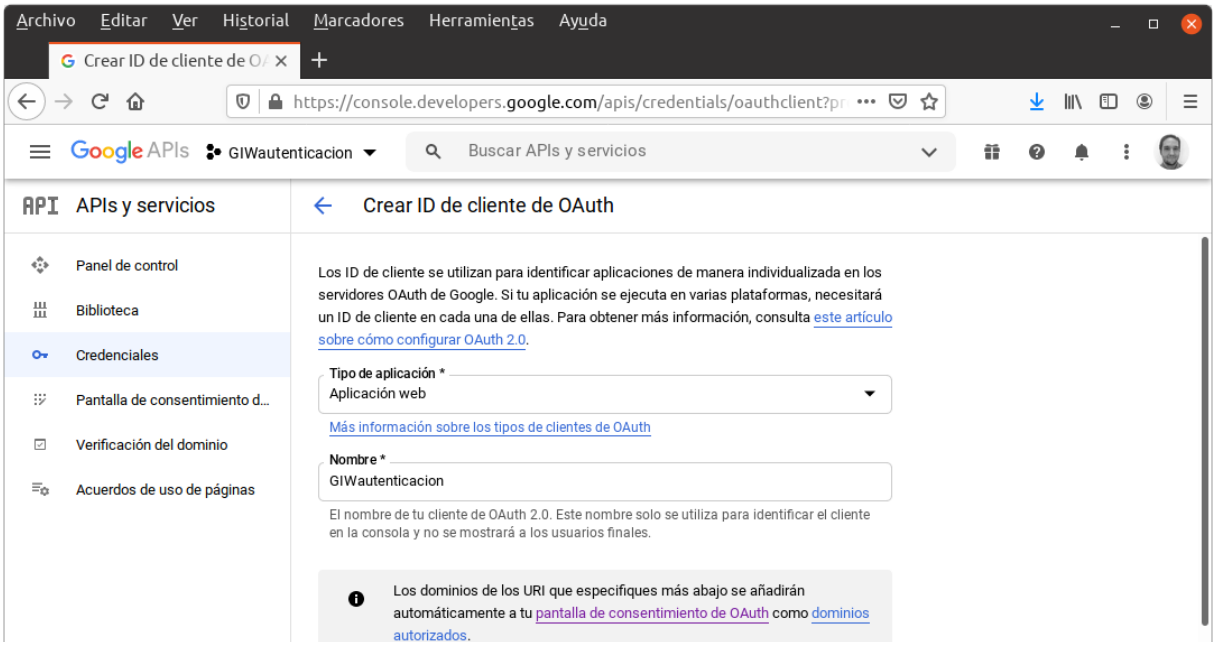


Figura 4: Creación de credenciales