



U N I V E R S I D A D
COMPLUTENSE
M A D R I D

Autenticación delegada

Gestión de la Información en la Web
Enrique Martín - emartinm@ucm.es
Grados de la Fac. Informática

Gestión de claves

- Como hemos visto, **gestionar** de manera adecuada las **claves** que almacenamos en la BD requiere **cuidado**.
- Además, las leyes de protección de datos imponen limitaciones y exigencias sobre los ficheros que contienen datos personales → **más tiempo y esfuerzo**.
- *¿Cómo podemos evitarlo?*

Autenticación delegada

- Una solución es **delegar** el proceso de autenticación en una **tercera parte**: Google, Facebook, Twitter, etc.
- De esta manera es esa **tercera parte** la que tiene que **gestionar** de manera segura los datos y las claves, y **cumplir** la legislación sobre almacenamiento de datos.

Autenticación delegada

- Usando autenticación delegada **facilitamos** a los potenciales usuarios el **acceso** a nuestra aplicación web:
 - No se tienen que registrar (o deben dar muy pocos datos).
 - Acceden a través de una red social de confianza (Google, Facebook, etc.).
- Adicionalmente podremos **integrarnos** con esa red social: publicar logros, obtener amigos, obtener imagen de perfil, almacenar en la nube, etc.

Autenticación delegada

- En principio **todos ganan** (*win-win*):
 - La red social consigue visibilidad y potenciales nuevos usuarios.
 - Nuestra aplicación se “desentiende” de la autenticación, aprovecha la confianza de la red social y se puede integrar con ella.
- **Ojo:** la autenticación delegada es un servicio, y como tal la red social podría cobrarlo, limitar el número de autenticaciones mensuales, etc.

Protocolos

- El principal protocolo de autenticación delegada es **OpenID Connect**.
- *OpenID Connect* es la **tercera evolución** de OpenID, tras pasar también por OpenID 2.0.
- Las versiones anteriores **no** gozaron de mucha **aceptación** debido a ser **complicadas** de utilizar.
 - Entre otros puntos requerían un uso intensivo de la firma de mensajes, al poder ser usado sobre HTTP no seguro.

Protocolos

- OpenID Connect **se despliega sobre OAuth 2.0**, el protocolo de autorización delegada con más éxito.
- Por ello, la **autenticación** con *OpenID Connect* y la **autorización** con *OAuth 2.0* son procesos muy similares, que comparten muchos de los pasos.
- Todo el proceso se basa en el envío y obtención de **tokens**: uno de **identificación** (*id_token*) y uno de **acceso** (*access_token*).

Pasos a realizar

- 1) Tu aplicación redirige al usuario al servicio autenticador (p.ej. Google). Allí el usuario rellena todo lo necesario para autenticarse.
- 2) Una vez autenticado, el servicio autenticador lo redirige a tu aplicación con un **código alfanumérico**.
- 3) Tu aplicación conecta con el servicio autenticador para **canjear** el **código** por un **token de identificación**. El **token de identificación** contiene la **información básica del usuario**: email, nombre de usuario, foto, etc.
- 4) Adicionalmente se puede canjear el código por un ***token de acceso*** que permite que tu aplicación realice acciones como si fuera el usuario (p.ej. publicar un texto en su tablón de mensajes).

Autenticación usando Google

- Veamos un **ejemplo** de autenticación utilizando OpenID Connect y la API de Google.
(Usar otra red social será muy parecido)
- Mostraremos las **ideas principales** del proceso y de las peticiones HTTP involucradas, pero debéis consultar los **detalles** concretos en la documentación oficial de Google:
 - <https://developers.google.com/identity/openid-connect/openid-connect>

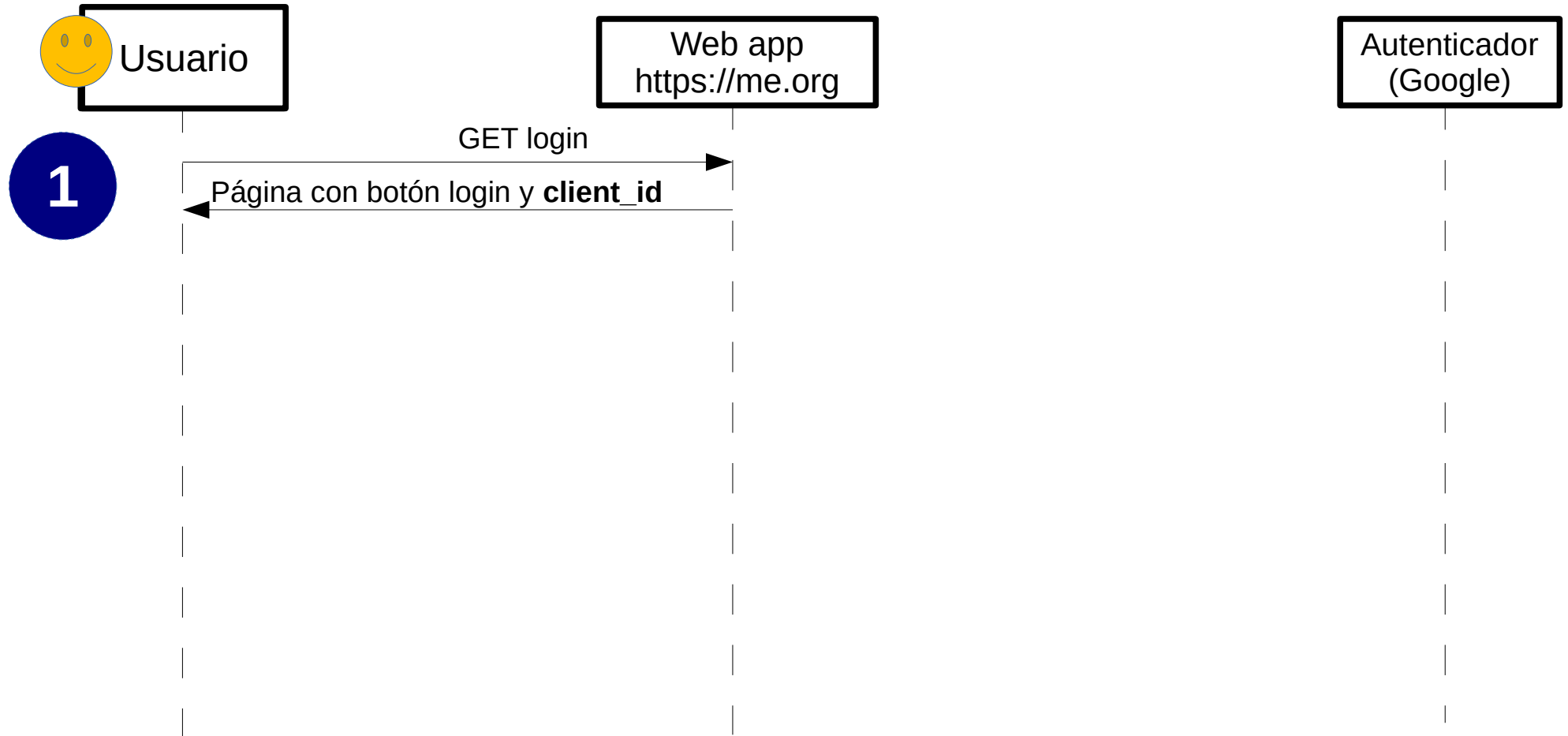
Autenticación usando Google

- Lo primero que hay que hacer es darse de **alta** como **desarrollador** en Google y dar de alta un proyecto.
- Todo esto se hace desde la **consola de desarrolladores** de Google:
<https://console.developers.google.com>
- Podéis usar vuestra cuenta UCM.
- ***En el enunciado de la práctica de este tema explico con detalle los pasos.***

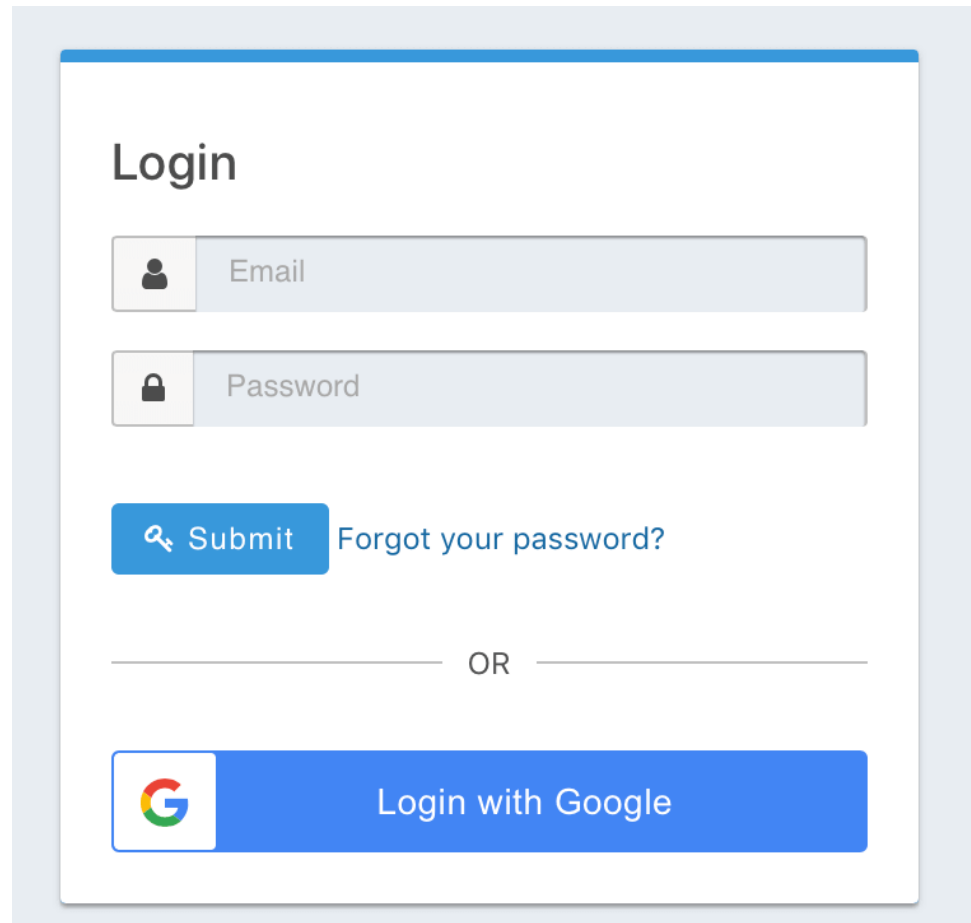
Autenticación usando Google

- También se debe generar los **credenciales** para OAuth. Esto creará (entre otros):
 - **client_id**: identificador único de mi aplicación web de cara a Google
 - **secret**: clave que solo conocen mi aplicación web y Google
 - **redirect_uri**: lista de URL de mi aplicación web a las que es legítimo que Google redirija al usuario tras un *login* con éxito

Autenticación: paso 1



Autorización: paso 1



The image shows a login form with a light blue border. At the top, the word "Login" is displayed in a bold, dark grey font. Below this, there are two input fields. The first field has a user icon on the left and the placeholder text "Email". The second field has a lock icon on the left and the placeholder text "Password". Below these fields, there is a blue button with a magnifying glass icon and the text "Submit", followed by a link that says "Forgot your password?". A horizontal line with the word "OR" in the center separates this from the Google login option. The Google login option consists of the Google "G" logo on the left and a blue button with the text "Login with Google" on the right.


Login

Email

Password

Submit Forgot your password?

OR

 Login with Google

(Fuente: <https://fusionauth.io/docs/v1/tech/identity-providers/google/>)

Autenticación: paso 1

- Las páginas de la aplicación web que permiten autenticar contendrán un enlace HTML al **punto de autorización** (*authorization endpoint*) de Google.
- La URL de este punto se obtiene del **documento de descubrimiento**. Es un JSON con las distintas rutas y configuraciones necesarias para autenticar:

<https://accounts.google.com/.well-known/openid-configuration>

(Supondremos que el *authorization endpoint* es <https://accounts.google.com/o/oauth2/v2/auth>)

Autenticación: paso 1

- Ese **enlace HTML** al punto de autorización contendrá varios parámetros:
 - **client_id**: para que Google sepa qué aplicación web es la que quiere solicitar la autenticación de un usuario.
 - **redirect_uri**: una de las dadas de alta en la consola del desarrollador, y a la que Google redirigirá al usuario tras la autenticación.
 - **scope**: para expresar qué datos se solicitan del usuario, en nuestro caso **openid email** (solo e-mail).
Dependiendo del *scope* Google mostrará diversos mensajes de advertencia sobre privacidad.

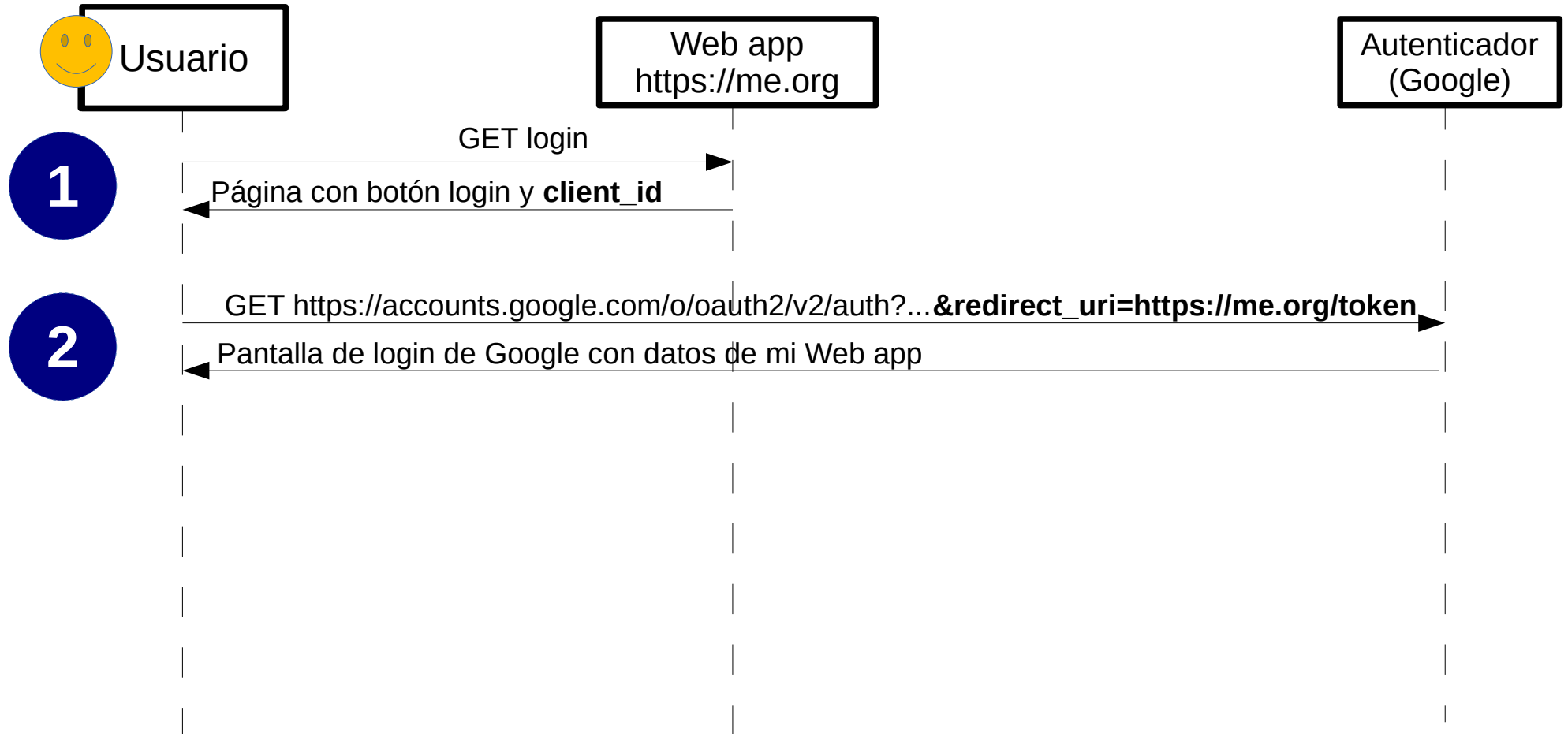
Autenticación: paso 1

- Suponiendo que `client_id` es 1234 y que `redirect_uri` es <https://me.org/token>, la URL completa del enlace quedaría como:

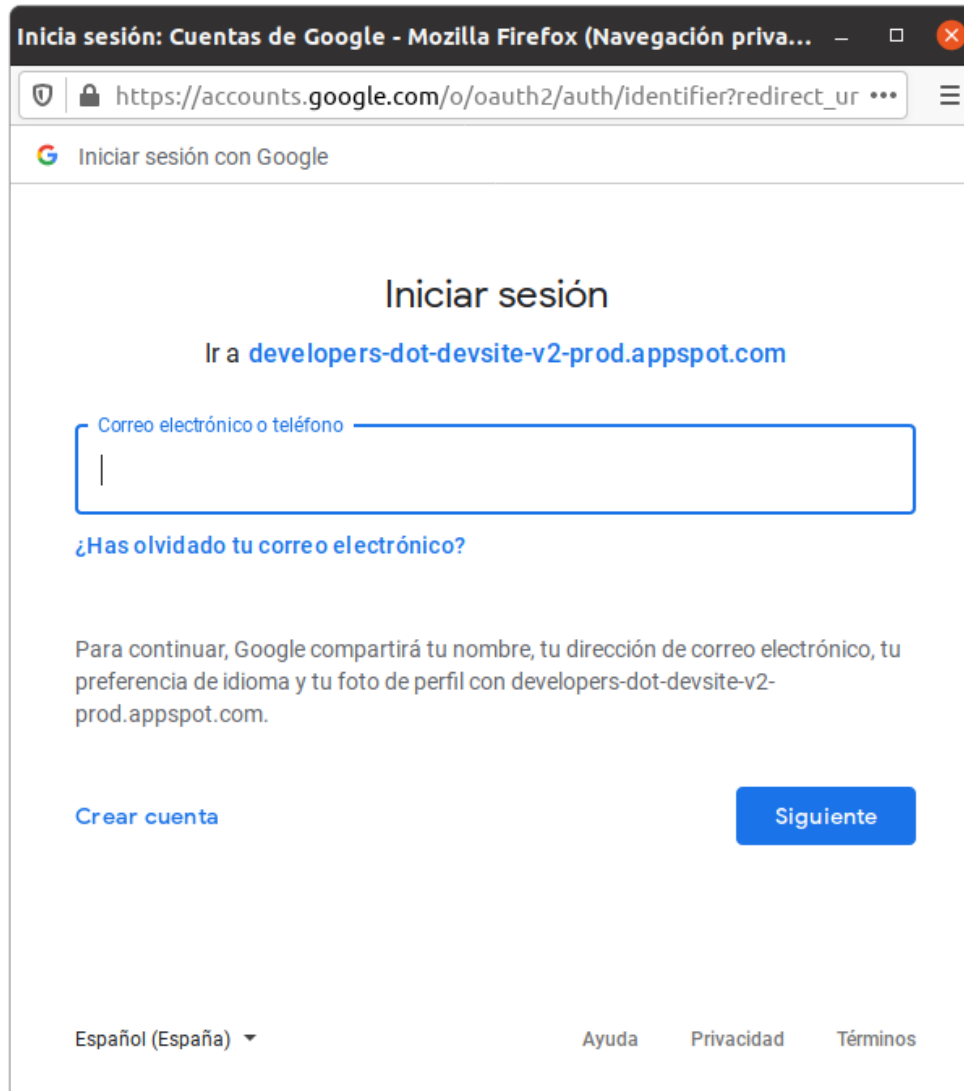
```
https://accounts.google.com/o/oauth2/v2/auth?  
  client_id=1234&  
  response_type=code&  
  scope=openid%20email&  
  redirect_uri=https://me.org/token
```

- Podéis encontrar detalles de los distintos argumentos en <https://developers.google.com/identity/openid-connect/openid-connect#sendauthrequest>

Autenticación: paso 2



Autenticación: paso 2



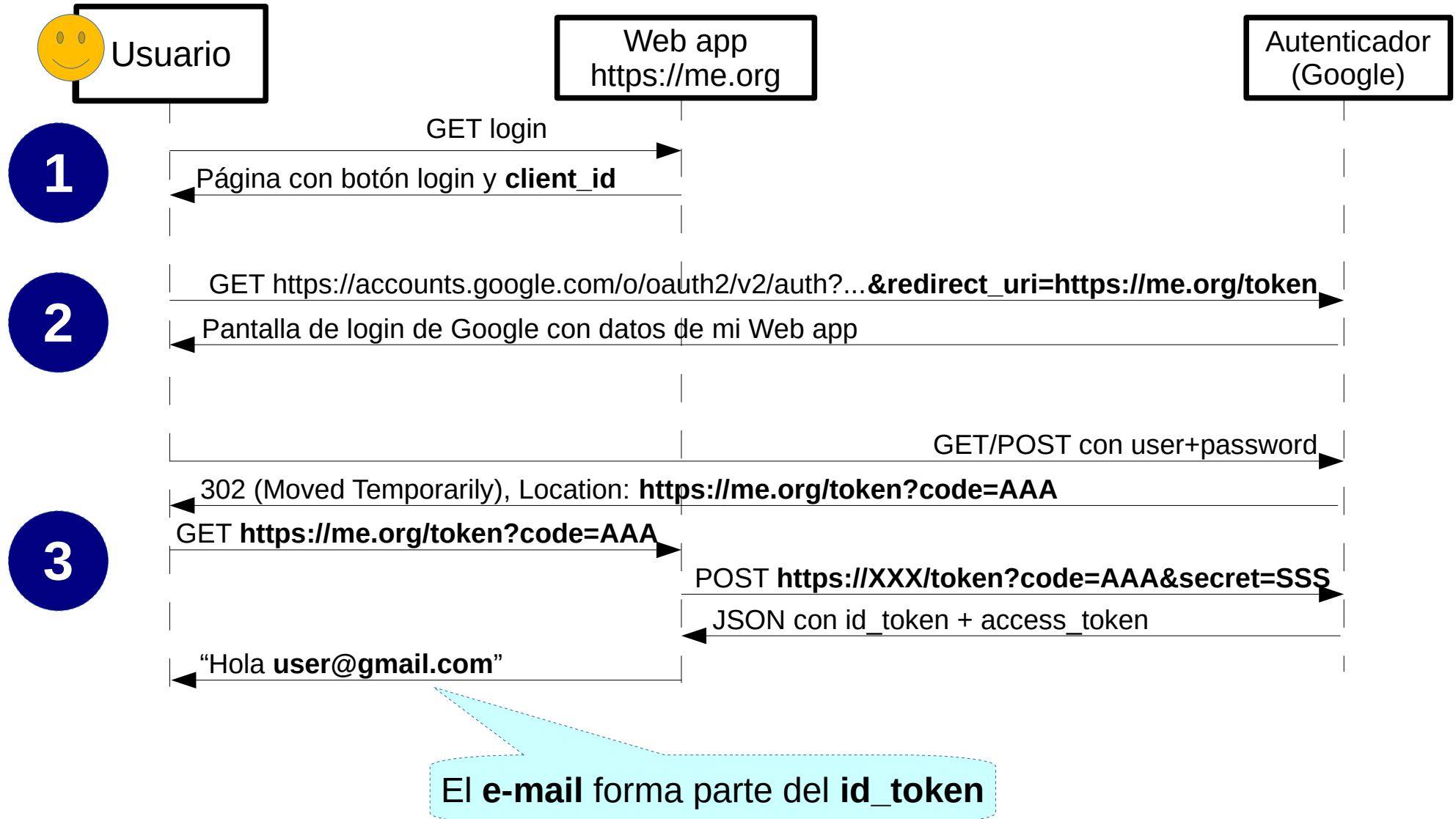
The screenshot shows a web browser window with the title "Inicia sesión: Cuentas de Google - Mozilla Firefox (Navegación priva...". The address bar displays the URL "https://accounts.google.com/o/oauth2/auth/identifier?redirect_ur...". The page content includes the Google logo and the text "Iniciar sesión con Google". The main heading is "Iniciar sesión", followed by the instruction "Ir a developers-dot-devsite-v2-prod.appspot.com". Below this is a text input field labeled "Correo electrónico o teléfono" with a cursor inside. A link "¿Has olvidado tu correo electrónico?" is positioned below the input field. A paragraph explains that Google will share the user's name, email address, language preference, and profile picture with the specified application. At the bottom, there are two buttons: "Crear cuenta" (a text link) and "Siguiente" (a blue button). The footer contains the language selection "Español (España)" with a dropdown arrow, and links for "Ayuda", "Privacidad", and "Términos".

(Fuente: <https://developers.google.com/identity/sign-in/web/sign-in>)

Autenticación: paso 2

- El usuario pincha en el enlace, que le lleva a una página de Google
- Aquí tiene que introducir sus credenciales y dar permiso a la aplicación web para que acceda a los datos solicitados en el parámetro *scope*.
- Gracias al *client_id* Google incluye información sobre tu aplicación web en la página de autenticación: título, logotipo, enlace a términos y condiciones, etc.

Autenticación: paso 3



Autenticación: paso 3

- La tercera etapa de autenticación involucra más pasos:
 - 1) El usuario envía sus credenciales a Google
 - 2) Google genera un código temporal **AAAA** y redirige al navegador del usuario a *redirect_uri* con dicho código **AAAA** como argumento.
 - 3) La aplicación web recibe el código **AAAA** y contacta con Google para canjearlo por los dos tokens: *id_token* y *access_token*
 - 4) La aplicación extrae la información sobre el usuario del *id_token*

Autenticación: paso 3

- Es importante darse cuenta de que el **código temporal solo sirve para mi aplicación web**
 - Para intercambiar el código por los tokens se necesita proporcionar el secreto que únicamente mi aplicación web y Google conocen.
- El **id_token** es un objeto JSON **firmado** con la clave privada de Google (JSON Web Token, JWT).
 - Para **validar** que el JWT de verdad ha sido producido por Google, es necesario usar el certificado digital público de Google. Este certificado se puede encontrar a través del *documento de descubrimiento*.

JSON Web Token (JWT)

- Un JWT es una cadena de texto formada por 3 fragmentos separados por puntos:
 - 1) Una **cabecera** que indica el tipo de token y el algoritmo usado para firmarlo
 - 2) Un **cuerpo** con los campos del JWT. Suele incluir información del usuario, del emisor, de la aplicación destino, etc.
 - 3) Una **firma** de las partes 1) y 2) usando un algoritmo de firma y un certificado público (RSA) o secreto.

```
RSASHA256(  
    base64UrlEncode(header) + "." +  
    base64UrlEncode(payload),  
    -----BEGIN PUBLIC KEY-----
```


Autenticación: paso 4

- Para depuración y prototipos, podemos solicitar a Google que valide el JWT a través de una petición a su endpoint de depuración
- Consultad los detalles en la documentación de Google:
<https://developers.google.com/identity/openid-connect/openid-connect#validatinganidtoken>
- Requiere conectar con:
<https://oauth2.googleapis.com/tokeninfo>

Autenticación: paso 4

- En una aplicación en producción validaremos los JWT en el propio servidor usando los certificados públicos de Google. Bibliotecas que servirían para este propósito:
 - <https://github.com/jpadilla/pyjwt>
 - <https://github.com/mpdavis/python-jose>
- Los certificados públicos de Google aparecen en el campo “**jwks_uri**” del documento de descubrimiento.

Autenticación: paso 4

- La validación del JWT en el servidor no es un proceso muy complejo si se usan bibliotecas como *pyjwt*. Ver el código de ejemplo en:

<https://pyjwt.readthedocs.io/en/stable/usage.html#retrieve-rsa-signing-keys-from-a-jwks-endpoint>

- 1) Obtener la URL con los certificados a partir del documento de descubrimiento
- 2) Crear un objeto **PyJWKClient** asociado a la URL de certificados
- 3) Usar el PyJWKClient para obtener la clave pública necesaria para validar el id_token recibido usando el método **get_signing_key_from_jwt()**
- 4) Decodificar y validar el id_token usando **jwt.decode()**

Autenticación: paso 4

- En lugar de descifrar el `id_token`, también se puede usar el `access_token` para obtener los datos del usuario.
- Requiere una conexión a **`userinfo_endpoint`** del documento de descubrimiento, pasando el `access_token` como **`cabecera`** de autenticación:

`Authorization: Bearer access_token`

Referencias

Referencias

- Documentación oficial sobre autenticación con Google:
<https://developers.google.com/identity/openid-connect/openid-connect>
- Explicaciones paso a paso sobre las fases de la autenticación con Google (rutas, peticiones y parámetros involucrados):
<https://developers.google.com/identity/openid-connect/openid-connect#server-flow>
- Consola del desarrollador de Google:
<https://console.developers.google.com>

Referencias

- Descifrado de JWTs de Google:
<https://developers.google.com/identity/openid-connect/openid-connect#validatinganidtoken>
- Documento de descubrimiento de Google:
<https://accounts.google.com/.well-known/openid-configuration>