**Google Sign in – front y back end**

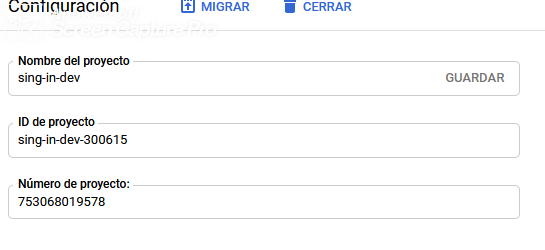
**Temas de la sección:**

1. Generar API Key de Google
2. Generar API Secret
3. Usar librerías de Google para la validación de tokens
4. Tips importantes en PostMan
5. Despliegues a Heroku
6. Uso del Google SignIn en el Front-End
7. Crear usuarios personalizados en base a respuestas de Google

Enlace con información para integrar Google Sign-in en nuestra web app.

https://developers.google.com/identity/sign-in/web/sign-in

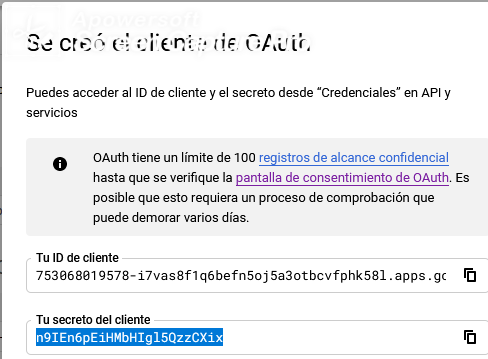
Iniciamos sesión, aceptamos términos y condiciones, en la página principal del dashboard creamos un nuevo proyecto.

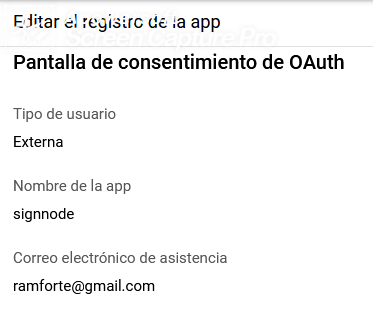
En credenciales, crear nuevas credencias 0auth, en caso de tener que llenar la pantalla de consentimiento, recuerda no usar mayúsculas y tampoco guiones porque no pasa. Llenar nombre y correo todo lo demás se puede dejar en blanco.

Al terminar click en crear credenciales, colocar un nombre y en orígenes autorizados ponemos para pruebas <http://localhost>:3000 y creamos.

ID CLIENT: **753068019578-i7vas8f1q6befn5oj5a3otbcvfphk58l.apps.googleusercontent.com**

Secreto del cliente: **n9IEn6pEiHMbHIgl5QzzCXix**





**Creando la página principal**

Creamos 07-restserver/public/index.html

Agregamos el script con la librería:

<script src="https://apis.google.com/js/platform.js" async defer></script>

(en el head, no importa)

<meta name="google-signin-client\_id" content="YOUR\_CLIENT\_ID.apps.googleusercontent.com"> //ponemos nuestro client ID

copiamos el código para poder el botón sign in y log out De la página con la documentación.

<div class="g-signin2" data-onsuccess="onSignIn"></div>

Botón y función logout

<a href="#" onclick="signOut();">Sign out</a>  
<script>  
  function signOut() {  
    var auth2 = gapi.auth2.getAuthInstance();  
    auth2.signOut().then(function () {  
      console.log('User signed out.');  
    });  
  }  
</script>

Función que obtiene la información

function onSignIn(googleUser) {  
  var profile = googleUser.getBasicProfile();  
  console.log('ID: ' + profile.getId()); // Do not send to your backend! Use an ID token instead.  
  console.log('Name: ' + profile.getName());  
  console.log('Image URL: ' + profile.getImageUrl());  
  console.log('Email: ' + profile.getEmail()); // This is null if the 'email' scope is not present.  
}

Debemos agregar localhost:3000 a la lista blanca, (donde dice orígenes javascript permitidos en credenciales)

**Validar Token de googlee back end**

En documentación: <https://developers.google.com/identity/sign-in/web/backend-auth>

    var id\_token = googleUser.getAuthResponse().id\_token; //Con esto conseguimos el tokenb de google

Then, send the ID token to your server with an HTTPS POST request, es Ajax, podemos usar React, Angular, etc. Para reemplazarlo, va en la función que hace sign in:

var xhr = new XMLHttpRequest();  
xhr.open('POST', 'https://yourbackend.example.com/tokensignin');  
xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');  
xhr.onload = function() {  
  console.log('Signed in as: ' + xhr.responseText);  
};  
xhr.send('idtoken=' + id\_token);

*Crearemos un servicio donde se recibirá el token y reemplazaremos ese valor en el xhr.open()*

*En el login(con lo anterior mandamos el token al servidor y este lo proporcional con un post a /google):*

app.post('/google', (req,res)=>{

    let token = req.body.idtoken // Este idtoken se envia desde la función que hace signin en google

    res.json({

        token

    });

});

***Validad el token de Google***

Para esto usaremos unas librería que nos proporciona Google siempre en la documentación authenticate with a back server, bajamos y clickeamos en nodejs , también existe para java.

npm install google-auth-library –save

Agregamos la función para obtener los datos a partir del token y adecuamos

async function verify(token) {  //La función la utillizaremos dentro del post, agregamos el token como argumento. Permite obtener los valores del usuario a partir del token

    const ticket = await client.verifyIdToken({

        idToken: token,

        audience: process.env.CLIENT\_ID,  // Specify the CLIENT\_ID of the app that accesses the backend

        // Or, if multiple clients access the backend:

        //[CLIENT\_ID\_1, CLIENT\_ID\_2, CLIENT\_ID\_3]

    });

    const payload = ticket.getPayload();

    //    console.log(payload.name)

    //    console.log(payload.email)

    //    console.log(payload.picture)

    return {

        nombre:  payload.name,

        email: payload.email,

        img: payload.picture,

        google: true

    }

    //const userid = payload['sub']; De la funcion original, no nos sirve

 }

Agregamos en el código post para que envié el token al servidor, en el servidor agregamos el código en el login que obtiene los datos a partir del token y los retorna:

En el index:

function onSignIn(googleUser) {

    let profile = googleUser.getBasicProfile();

    let id\_token = googleUser.getAuthResponse().id\_token; //Con esto conseguimos el token de google

   try

    {

        let xhr = new XMLHttpRequest();

        xhr.open('POST', '/google');  //   Porque estoy en el mismo servidor, y este es el index por eso /google

        xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');

        xhr.onload = function() {

        console.log('Signed in as: ' + xhr.responseText);

        };

        xhr.send('idtoken=' + id\_token);

    }catch(err)

    {

        console.log('Sucedio un error');

        console.error(err);

    }

  }

El post:

app.post('/google', async (req,res)=>{

    let token = req.body.idtoken; // Este idtoken se envia desde la función que hace signin en google

    let googleUser = await verify(token) //Recuerda poner el await o de lo contrario no espera la respuesta

                    .catch( e => {

                       res.status(403).json({

                           ok: false,

                            err: e

                        });

                    });  //catch

    Usuario.findOne( { email: googleUser.email }, (err, usuarioBD) => {

        if(err)

        {

            return res.status(500).json({ //500 error interno del servidor

                ok: false,

                errerr: {

                    message:"Error al buscar el usuario en la BD"

                }

             });

        }

        if(usuarioBD)  //Si existe. puede estar ya registrado de forma normal o por google

        {

            if(usuarioBD.google === false)

            {

                console.log('El usuario No se creo con google');

                return res.status(400).json({

                    ok: false,

                    err: {

                        message: "Sus datos ya cuentan con una cuenta tradicional, inicie sesión de forma tradicional"

                    }

                });

            }

            else{ // SI previamente se había autenticado con google, le generamos un token de nuestra app

                let token = jwt.sign(

                    { usuario: usuarioBD },

                       process.env.SEED,

                       {expiresIn: 60\*60\*24\*30}  //seg\*min\*hrs\*days

                       //    {expiresIn: process.env.CADUCIDAD\_TOKEN}  //seg\*min\*hrs\*days

                   );

                return res.json({

                   ok: true,

                   usuario:usuarioBD,

                   token, //token: token

                 })

            }

        }else{ //SI es usuario no existe en la BD, se creará un nuevo usario

            console.log('El usuario No existe, se crea uno nuevo');

            let usuario = new Usuario ();

            usuario.nombre = googleUser.nombre;

            usuario.email = googleUser.email;

            usuario.img = googleUser.img;

            usuario.google = true;

            usuario.password = ':)'; //para pasar la restricción de obligatoriedad. Porque no la estamos encriptando, al querer hacer login se va a incriptar y nunca va a corresponder

            usuario.save(usuario, (err, usuarioBD) => {

                if(err)

                {

                    return res.status(500).json({ //500 error interno del servidor                   ok: false,

                         err: {

                            message:"El ususario no existe pero se produjo un error al crearlo"

                        }

                     });

                }

                let token = jwt.sign(

                    { usuario: usuarioBD },

                       process.env.SEED,

                       {expiresIn: 60\*60\*24\*30}  //seg\*min\*hrs\*days

                       //    {expiresIn: process.env.CADUCIDAD\_TOKEN}  //seg\*min\*hrs\*days

                   );

               return res.json({

                   ok: true,

                   usuario: usuarioBD,

                   token, //token: token

               });

            });

        } //else

    }); //Buscamos si es usario ya esta en la base de datos o al menos su email

});

Después de subir, en este caso al heroku, debemos actualizar las credenciales en Google

Hacemos heroku open para abrirlo en el explorador, vamos a la consola y copiamos la url que nos marca el erro, posteriormente copiamos la url y vamos a Google:

***https://console.developers.google.com/apis/credentials?project=sing-in-dev-300615&folder=&organizationId=***

en orígenes java permitido lo pegamos.

Para esto ya habriamos subido a heroku y a git, ahora subiremos el tag.