# Oath2

Julian Felipe Latorre Ochoa

August 2024

## 1 Introducción

Este documento proporciona una guía paso a paso para implementar OAuth2 en una aplicación Express.js.

## 2 Requisitos previos

- Node.js y npm instalados
- Conocimientos básicos de Express.js
- Una cuenta en un proveedor de OAuth2 (por ejemplo, Google, GitHub)

## 3 Pasos de implementación

1. **Configuración del proyecto**
   a. Crear un nuevo directorio para el proyecto
   b. Inicializar un nuevo proyecto npm:

```
npm init -y
```

   c. Instalar las dependencias necesarias:

```
npm install express passport passport-oauth2
```

2. **Configuración del proveedor OAuth2**
   a. Registrar tu aplicación con el proveedor OAuth2
   b. Obtener el Client ID y Client Secret
   c. Configurar la URL de redirección
3. **Configuración de Express y Passport**
   a. Crear un archivo `app.js`

b. Importar las dependencias necesarias:

```
const express = require('express');
const passport = require('passport');
const OAuth2Strategy = require('passport-oauth2');
```

c. Configurar la estrategia OAuth2:

```
passport.use(new OAuth2Strategy({
    authorizationURL: 'https://provider.com/oauth2/authorize',
    tokenURL: 'https://provider.com/oauth2/token',
    clientID: CLIENTE_ID,
    clientSecret: CLIENTE_SECRET,
    callbackURL: "http://localhost:3000/auth/callback"
},
function(accessToken, refreshToken, profile, cb) {
    // Manejar la autenticación exitosa
}));
```

4. **Configuración de rutas**

a. Configurar la ruta de inicio de sesión:

```
app.get('/auth/login', passport.authenticate('oauth2'));
```

b. Configurar la ruta de callback:

```
app.get('/auth/callback',
    passport.authenticate('oauth2', { failureRedirect: '/login' }),
    function(req, res) {
        // Autenticación exitosa, redirigir
        res.redirect('/');
    });
```

5. **Manejo de sesiones**

a. Instalar express-session:

```
npm install express-session
```

b. Configurar la sesión en `app.js`:

```
const session = require('express-session');
app.use(session({ secret: 'tu secreto', resave: false, saveUninitialized: f
app.use(passport.initialize());
app.use(passport.session());
```

6. **Protección de rutas**

   a. Crear un middleware para verificar la autenticación:

```
function ensureAuthenticated(req, res, next) {
    if (req.isAuthenticated()) { return next(); }
    res.redirect('/auth/login');
}
```

   b. Usar el middleware en rutas protegidas:

```
app.get('/perfil', ensureAuthenticated, function(req, res){
    res.send('Perfil del usuario');
});
```

7. **Iniciar la aplicación**

   a. Agregar el código para iniciar el servidor:

```
const PORT = process.env.PORT || 3000;
app.listen(PORT, () => console.log(`Servidor corriendo en puerto ${PORT}`))
```

   b. Ejecutar la aplicación:

```
node app.js
```

# 4 Implementación con Google como servidor de autenticación

Para usar Google como servidor de autenticación, seguiremos estos pasos adicionales:

1. **Configuración en Google Cloud Console**

   a. Ir a `https://console.cloud.google.com/`
   b. Crear un nuevo proyecto o seleccionar uno existente
   c. Navegar a "APIs & Services" ¿ "Credentials"
   d. Hacer clic en "Create Credentials" ¿ "OAuth client ID"
   e. Seleccionar "Web application" como tipo de aplicación
   f. Configurar "Authorized JavaScript origins" con `http://localhost:3000`
   g. Configurar "Authorized redirect URIs" con `http://localhost:3000/auth/google/callback`
   h. Anotar el Client ID y Client Secret proporcionados

2. **Instalación de dependencias adicionales**

```
npm install passport-google-oauth20
```

3. **Configuración de Passport para Google OAuth**

```
const GoogleStrategy = require('passport-google-oauth20').Strategy;

passport.use(new GoogleStrategy({
    clientID: GOOGLE_CLIENT_ID,
    clientSecret: GOOGLE_CLIENT_SECRET,
    callbackURL: "http://localhost:3000/auth/google/callback"
},
function(accessToken, refreshToken, profile, cb) {
    // Aquí manejas el perfil del usuario
    // Típicamente, buscarías o crearías un usuario en tu base de datos
    return cb(null, profile);
}));

passport.serializeUser((user, done) => {
    done(null, user);
});

passport.deserializeUser((user, done) => {
    done(null, user);
});
```

4. **Configuración de rutas para autenticación Google**

```
app.get('/auth/google',
    passport.authenticate('google', { scope: ['profile', 'email'] }));

app.get('/auth/google/callback',
    passport.authenticate('google', { failureRedirect: '/login' }),
    function(req, res) {
        // Autenticación exitosa, redirigir al home
        res.redirect('/');
    });
```

5. **Manejo de la información del usuario**

   a. En la función de callback de GoogleStrategy, puedes acceder a la información del usuario a través del objeto `profile`

   b. Típicamente, aquí crearías o actualizarías un registro de usuario en tu base de datos

c. Ejemplo de manejo básico:

```
function(accessToken, refreshToken, profile, cb) {
    console.log('Google profile:', profile);
    // Aquí normalmente buscarías o crearías un usuario en tu BD
    // Por ahora, solo pasamos el perfil completo
    return cb(null, profile);
}
```

6. **Protección de rutas y acceso a la información del usuario**

```
app.get('/profile', ensureAuthenticated, (req, res) => {
    res.json(req.user);
});

function ensureAuthenticated(req, res, next) {
    if (req.isAuthenticated()) {
        return next();
    }
    res.redirect('/auth/google');
}
```