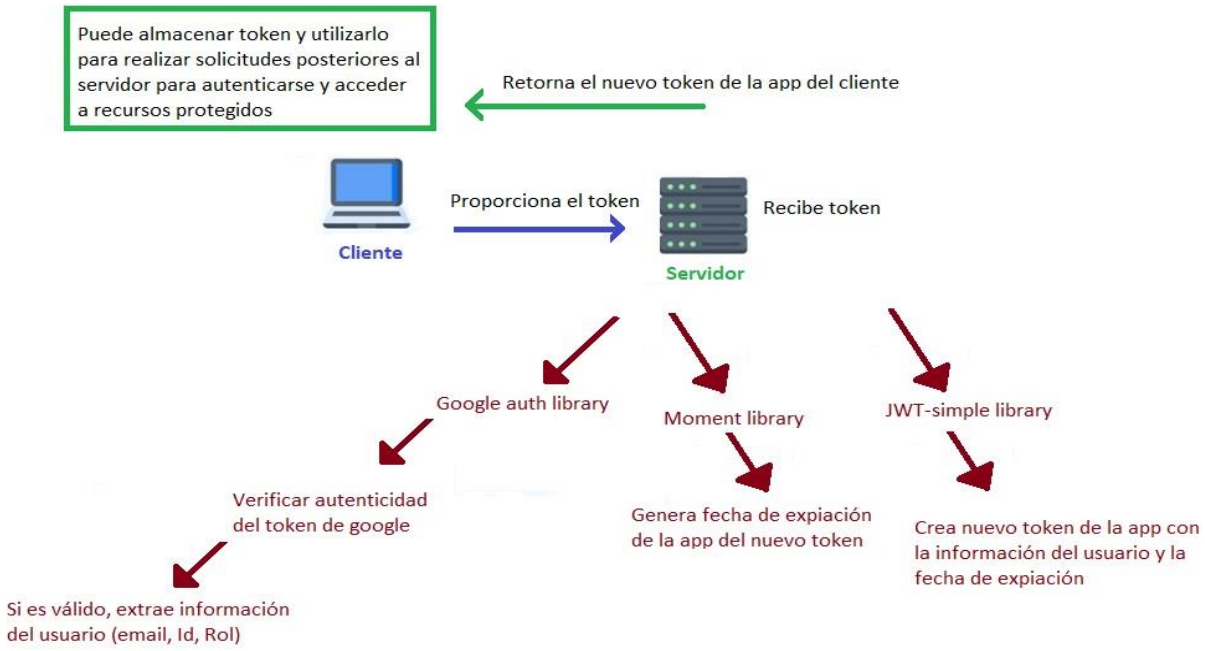
Implementación de inicio de sesión con Google en el api rest de la plataforma web Dory

Objetivos:

- Permitir que los usuarios inicien sesión en la aplicación usando sus cuentas de google.
- Verificar el token de Google recibido y autenticar al usuario.
- Generar un nuevo token de aplicación utilizando la información del usuario autenticado.



Recursos necesarios:

- Node.js versión 16.14.1 + npm versión 8.5.0 instaladas en el sistema.
- Express versión 4.17.1
- Conocimientos básicos de node.js, Express y manejo de paquetes npm.

Piezas de software:

- Librería Google-auth-library versión 7.14.1
- Librería JWT-simple versión 0.5.6
- Librería moment versión 2.29.1

Pasos:

1. Retomando la receta " integración del servicio SMTP de Google en el api rest de la plataforma web Dory, en las variables de entorno asignar las credenciales de Google: El id de cliente de la aplicación web (OAUTH_CLIENT_ID_WEB) y el id del cliente de la aplicación móvil (OAUTH_CLIENT_ID_MOBILE).

2. En el archivo principal "index.js" se debe tener la configuración del servidor de Express (ver imagen 1).

```
const express = require('express');
const app = express();
const bodyParser = require('body-parser');
const port = process.env.PORT || 3000;
const {verifyToken, validarToken} = require ('./middelware/auth');
require('dotenv').config();

/*Socket.io con express*/
const http = require('http');
const server = http.createServer(app);
const { Server } = require("socket.io");
const io = new Server(server,{
  cors: {
    origin:"*.*"
  }
});

server.listen(port, () => {
  console.log('listening on *:'+port);
});
```

Imagen 1. *Configuración del servidor Express*

3. *Se debe tener implementado la ruta para iniciar sesión con Google en el archivo principal "index.js" (ver imagen 2).*

```
const loginRouter = require('./routes/login');
app.use('/api/login', loginRouter)
```

Imagen 2. *Implementación de la ruta para iniciar sesión con Google en "index.js"*

4. *Se debe tener definido la función "createToken" para generar un nuevo token de aplicación en el módulo "helper.js" (ver imagen 3).*

```
function createToken (user,minutes) {
  var payload = {
  email:user.email,
  sub: user.id,
  rol: user.nombre_tipo_usuario,
  iat: moment().unix(),
  exp: moment().add(minutes, "minutes").unix(),
};
return  jwt.encode(payload, config.TOKEN_SECRET);
};
```

Imagen 3. Función "createToken" en módulo "helper.js"

5. *Asegúrate de tener las variables de clientes necesarias configuradas en el archivo de implementación del servicio de google, como TOKEN_SECRET, CLIENT_ID_1 y CLIENT_ID_2 (Ver imagen 4).*

```
const {OAuth2Client} = require('google-auth-library');
const CLIENT_ID_1 = [redacted];
const CLIENT_ID_2 = [redacted];
const client = new OAuth2Client(CLIENT_ID_1);
```

Imagen 4. *Declaración de variables de clientes*

6. En el servicio de inicio de sesión con Google se recibe la solicitud y en el cuerpo de la solicitud el token de Google con el nombre "token"(ver imagen 5).

```
async function loginWithGoogle(req){        Solicitud del usuario
            if(!req.body.token){
              throw createError(400,'Debe proporcionar un token.');
            }
```

Imagen 5. *Servicio de inicio de sesión con Google recibiendo la solicitud del usuario*

7. Si el usuario está verificado por Google y existe en la base de datos Dory, se le crea un nuevo token de aplicación. (ver imagen 6).

```
let creadoCon =  'Google';
const newToken = await helper.createToken(rows[0],ONE_YEAR_MILLISECONDS);
return {
  token: newToken,
  authenticated:true,
  creadoCon
};
```

Imagen 6. *Crearción del token de usuario en el servicio de login de google*

8. Prueba la implementación iniciando tu servidor Node.js y enviando una solicitud HTTP con el verbo POST a la ruta /api/login/google con el token de Google en el cuerpo de la solicitud con el nombre “token”. Deberías recibir un nuevo token de aplicación en la respuesta. (ver imagen 7).

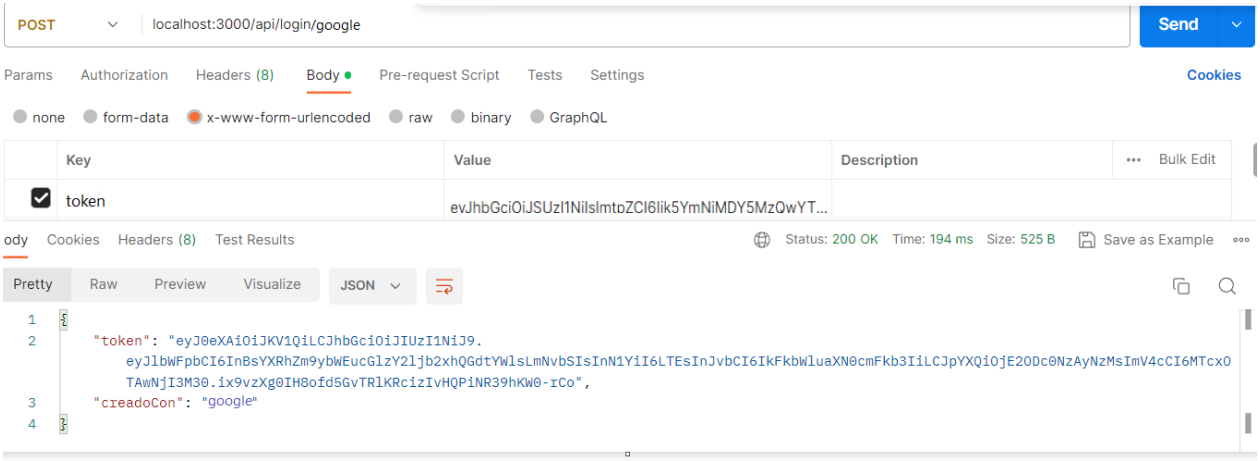


Imagen 7. *Prueba en postman del servicio de login de google*