UNIVERSIDADE  LUSÓFONA

SISTEMAS DE GESTÃO DE IDENTIDADE
OAUTH2
DOCENTE: DOUTOR JOSÉ ROGADO

DANIEL LEANDRO N#21800566

MESTRADO DE ENGENHARIA DE INFORMÁTICA E SISTEMAS DE INFORMAÇÃO

08 DE JUNHO DE 2021

#AGENDA

- #Introdução
- #Desenvolvimento
- #OAuth2
- #Fluxo Padrão
- #Fluxo da Aplicação
- #oauth2.js
- #Base de Dados
- #Servidor
- #Passport-Oauth2
- #Conclusão

#INTRODUÇÃO

Este trabalho é produzido no âmbito da cadeira de Sistemas de Gestão de Identidade tendo como objetivo de criar uma aplicação com o sistema de autenticação OAuth2 e usar ferramentas e conhecimentos dados ao longo do semestre na cadeira. O trabalho foi elaborado com base em JavaScript e NodeJS.

A aplicação é um sistema de criação e edição de textos que para criar, editar ou eliminar é preciso estar logado. Para efetuar login pode ser feito localmente através do site, ou então usar o OAuth2 de sistemas conhecidos como o Gmail, Facebook ou Github.

#DESENVOLVIMENTO

Todo o código da aplicação encontra-se documentada, estando disponível no Github através de https://github.com/DASLCODE/OAuth2.git.

O desenvolvimento teve como principal base de apoio, as aulas de Sistemas de Gestão de Identidade onde foi possível retirar conteúdo e conhecimento para poder criar esta aplicação, realçar o JavaScript, Node.JS, Passport.JS, Express.JS, mongoDB, Oauth2.

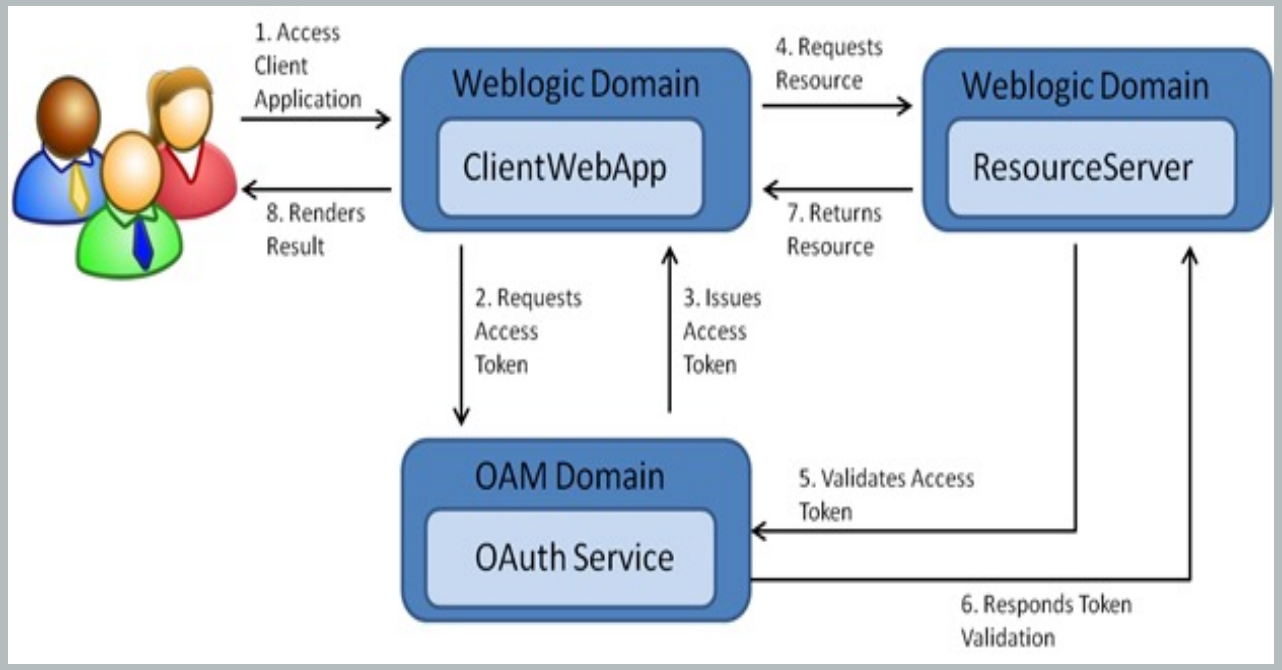
Através da plataforma de ensino Udemy, consegui também fortalecer esse conhecimento através do curso "Complete back end development with NodeJS with projects" e assim enriquecer o projeto.

#DESENVOLVIMENTO

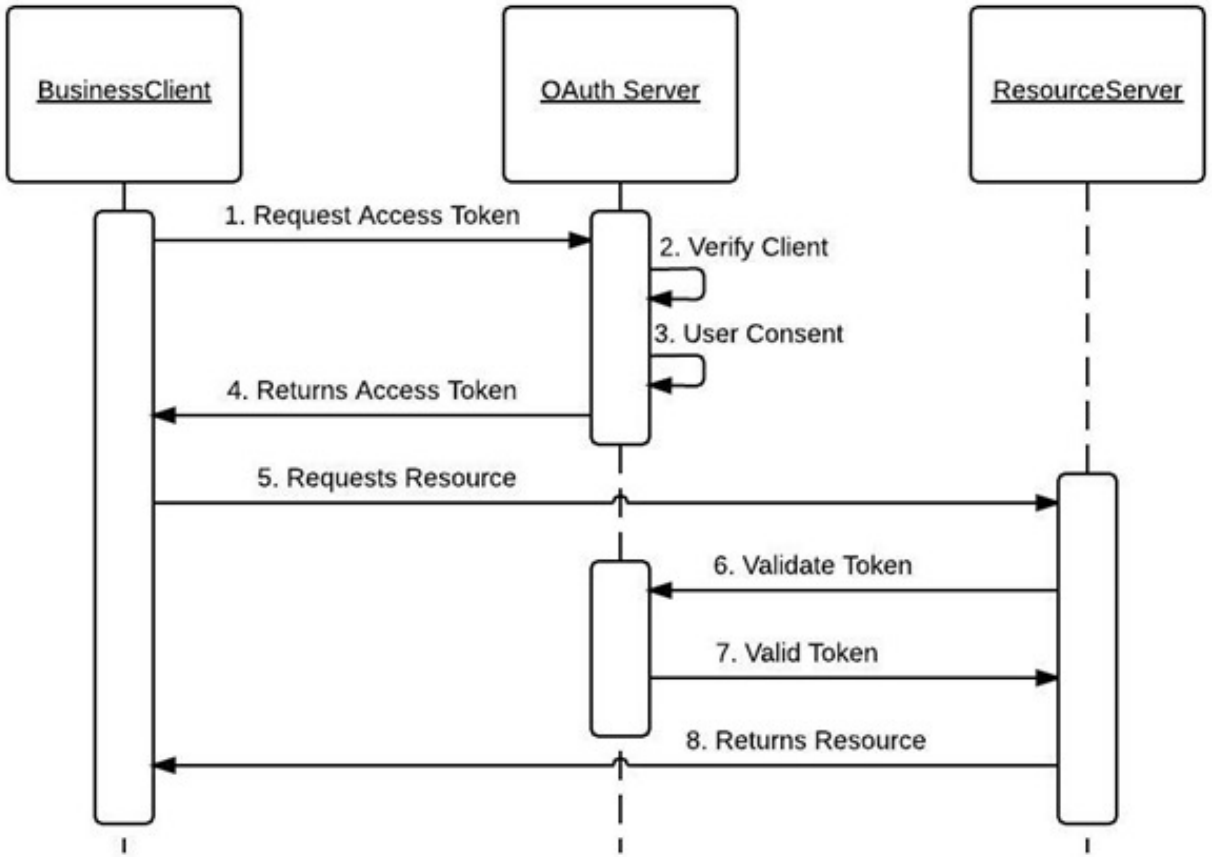
- JavaScript é uma linguagem de programação de script de alto nível com tipagem dinâmica fraca e multiparadigma mas de fácil utilização. A Aplicação foi desenvolvida em em JS.
- Node.js irá ter a função de correr como servidor que fará ligação da aplicação Express.js a uma base de dados MongoDB.
- Todo interface e frontend foi criado através do compilador Pug.js.
- As Libraries são geridas através do Bower que gere e certifica que os pacotes estão todos atualizados.

#OAUTH2

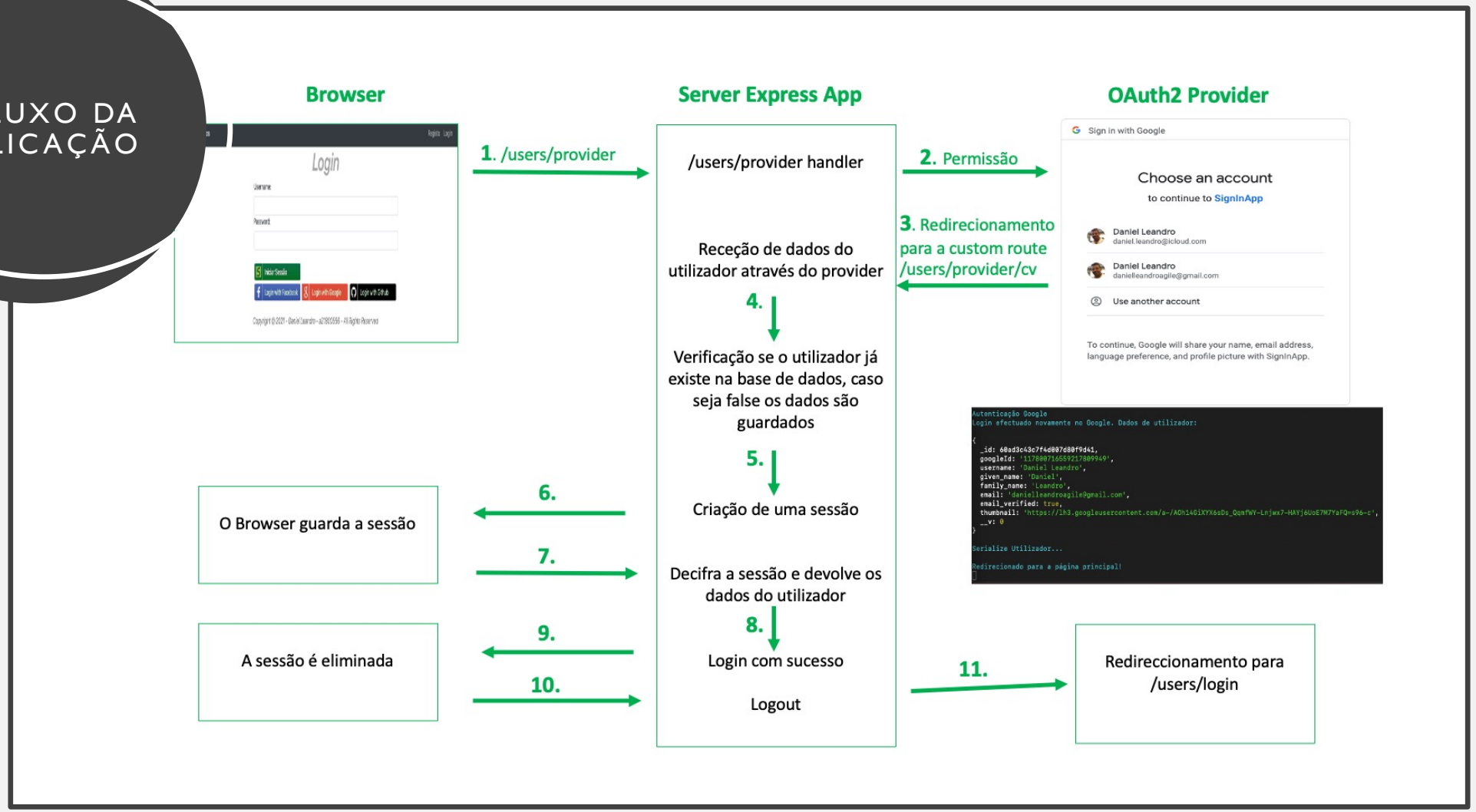
A estrutura de autorização OAuth2 é um protocolo que permite que um utilizador conceda a uma web-page ou um aplicação de terceiros acessos aos recursos protegidos pelo utilizador, sem que haja um comprometimento das credenciais ou mesmo da sua identidade.



#FLUXO PADRÃO



#FLUXO DA APLICAÇÃO



#OAUTH2.JS

- O ficheiro a ser executado é o oauth2.js. Podendo ser chamado como o ficheiro principal. Nos próximos slides vai estar todo desenvolvimento que este ficheiro tem.

#OAUTH2.JS

- Aqui é onde é criada a configuração da ligação à base de dados e reencaminha para o ficheiro keys.js que tem a dbURI que é fornecida pelo MongoDB.
- O mongoose é uma biblioteca que serve para modelar o mongoDB no node.js, assim conseguimos modelar os nossos dados todos diretamente no código.

```
const mongoose = require('mongoose');

const db = mongoose.connect(keys.mongodb.dbURI, {useNewUrlParser: true}, function(error){

console.log('A ligar à base de dados...'.yellow);
if (mongoose.connection.readyState == 0 || error){
    console.log('Não é possível ligar à base de dados.\n'.bgRed);
}else {
    console.log('\nLigação com sucesso!'.bgGreen);
}
});
```

```
mongodb: {
    dbURI: 'mongodb+srv://trabalhofinal:12345@cluster0.kl637.mongodb.net/myFirstDatabase?retryWrites=true&w=majority'
},
```

#OAUTH2.JS

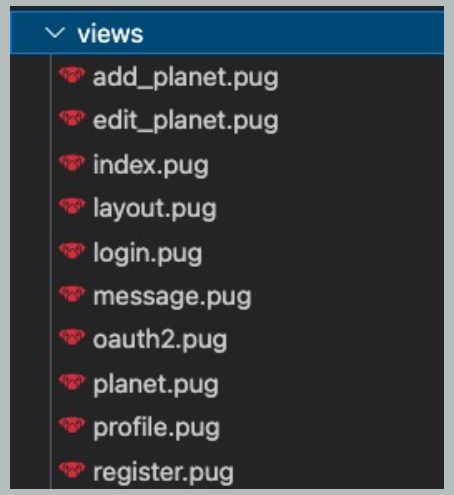
- No oauth2.js também é instanciado a aplicação express. Onde se encontra definido todos acessos para a pasta public.
- O express.static middleware serve para reconhecer ficheiros estáticos como imagens, css, js. O path que o express.static cria é relativamente à directoria do processo do node. No entanto há uma directoria criada apenas para o efeitos de views e outra para efeitos de bibliotecas de frontend.

```
const app = express();

const path = require('path');
app.use(express.static(path.join(__dirname, 'public')));
app.set('views', path.join(__dirname, 'views'));
```

#OAUTH2.JS

- Pug.js é um mecanismo de modelagem HTML. Permite gerar templates.
- No oauth2.js o Pug.js é configurado como sendo o mecanismo de visualização da nossa Express application.

```
app.set('view engine', 'pug');
```



#OAUTH2.JS

- Body Parser middleware é responsável por analisar os corpos da solicitação de entrada em um middleware antes de manipulá-lo.
- O body-parser extrai a parte inteira do corpo de um fluxo de solicitação de entrada e expõe no corpo de req.
- O app.use (bodyParser.json ()) transmite ao sistema informação para que o Json seja utilizado.
- O bodyParser.urlencoded ({extended: ...}) transmite ao sistema se usa um algoritmo simples para análise (false) ou um algoritmo complexo para uma análise mais profunda (true).

```
const bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: false }))
app.use(bodyParser.json())
```

#OAUTH2.JS

- Entre os pedidos de HTTP, tem que se armazenar os dados do utilizador.
- Os cookies e os parâmetros de URL são formas de transportar dados entre o servidor e o cliente, através das sessões.
- É atribuido um ID ao cliente e todos solicitações tem esse ID associado que é armazendo no servidor com esse ID.

```
const session = require('express-session');
app.use(session({
    secret: 'keyboard daniel',
    resave: true,
    saveUninitialized: true
}));
```

#OAUTH2.JS

- O connect-flash é uma biblioteca que permite enviar mensagens flash sempre que você for redirecionado para outra página da web.
- Normalmente, quando você faz login em um site, uma mensagem 'pisca' no topo da tela indicando o sucesso ou falha no login.

```
app.use(require('connect-flash')());
    app.use(function (req, res, next) {
    res.locals.messages = require('express-messages')(req, res);
    next();
});
```

#OAUTH2.JS

- Passport é middleware de autenticação para Node.js .
- Um conjunto abrangente de estratégias oferece suporte à autenticação usando um nome de usuário e senha , Facebook , Twitter e muito mais .
- Numa Express-based app, o passport.initialize() middleware é necessário para iniciar o Passport na app e o passport.session() middleware é necessário caso seja necessário sessões de login (nest caso, por cookie).

```
const passport = require('passport');
require('./config/passport')(passport);
app.use(passport.initialize());
app.use(passport.session());
```

#OAUTH2.JS

- Passport é middleware de autenticação para Node.js .
- Um conjunto abrangente de estratégias oferece suporte à autenticação usando um nome de usuário e senha , Facebook , Twitter e muito mais .
- Numa Express-based app, o passport.initialize() middleware é necessário para iniciar o Passport na app e o passport.session() middleware é necessário caso seja necessário sessões de login (nest caso, por cookie).

```
const passport = require('passport');
require('./config/passport')(passport);
app.use(passport.initialize());
app.use(passport.session());
```

#OAUTH2.JS

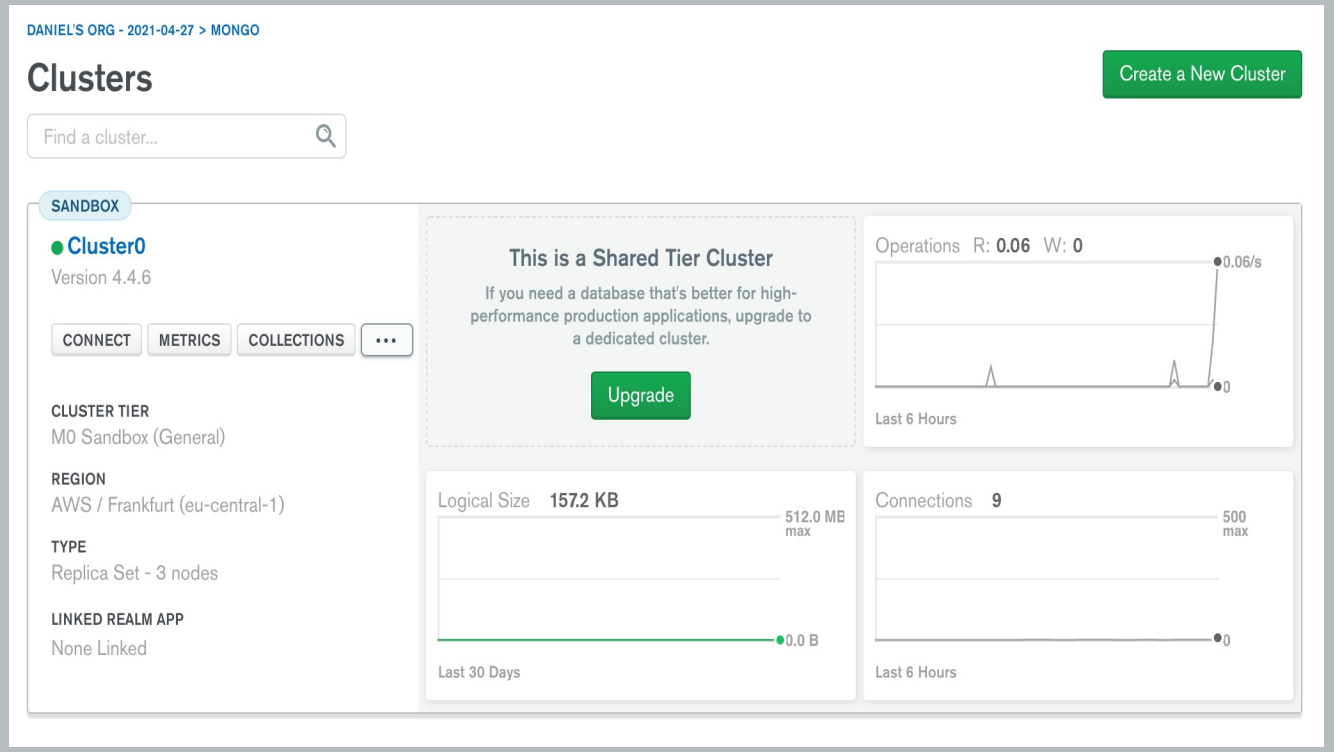
- Ainda no ficheiro principal estão definidos os paths para as outras páginas.
- Também é definido a Porta da Aplicação que neste caso é a 3000.

```
const planets = require('./routes/planets');
app.use('/planets', planets);

const users = require('./routes/users');
app.use('/users', users);

const profile = require('./routes/profile');
app.use('/profile', profile);

const oauth2 = require('./routes/oauth2');
app.use('/oauth2', oauth2);
```

```
const port = process.env.PORT || 3000;
app.listen(port, () => {
    console.log('\nAguarde pelo servidor...'.yellow);
    console.log('\nAVISO: A App está a correr na porta: '.red + port + '\n');
});
```

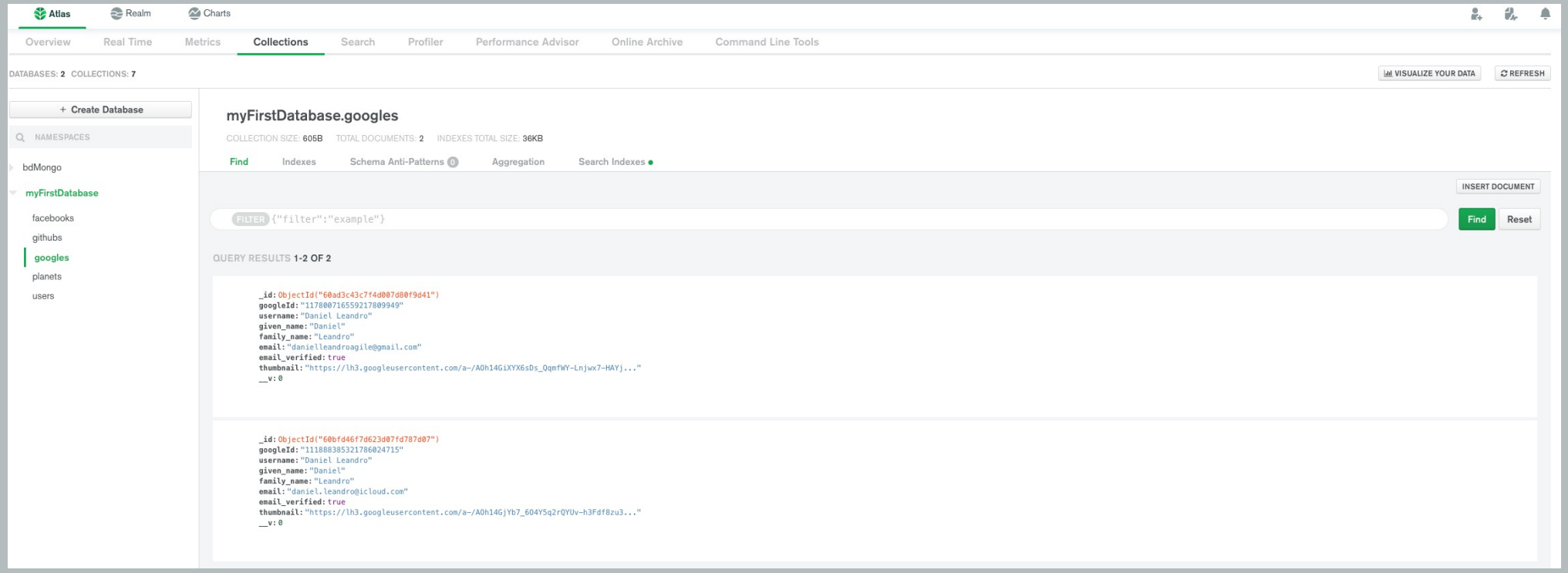
#BASE DE DADOS

- O mongoDB.atlas permite a criação de um cluster onde podemos alojar a nossa base de dados, com enorme facilidade.
- Oferecendo enorme variabilidade de métricas, funções e segurança.



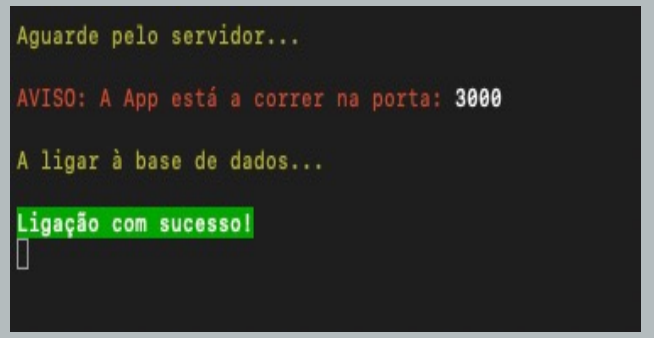
#BASE DE DADOS

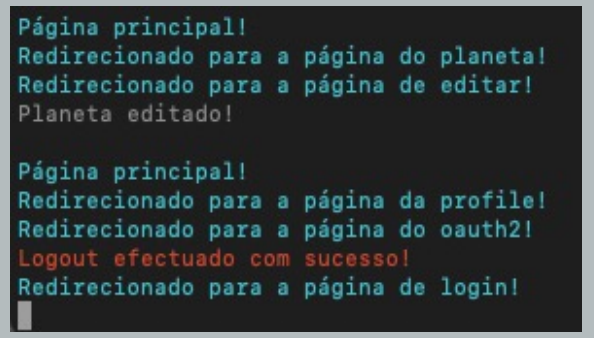
- Exemplo de utilizadores logados através da Google.



#SERVIDOR

- No lado do servidor, é oferecido um código informativo e apelativo, usando cores para o status da comunicação.
- É comunicado toda atividade que é efetuada na aplicação.





#PASSPORT-OAUTH2

- O Passport é um middleware que lida com a autenticação (registo e login) e autorização (acesso a rotas protegidas) para seu aplicativo.
- o Passport é um middleware que lida com a autenticação (registro e login) e autorização (acesso a rotas protegidas) para seu aplicativo. Além dessas duas funções, o Passport não interfere nas funcionalidades do seu aplicativo - em outras palavras, é "desobstrutivo".

#PASSPORT-OAUTH2

- Instalar o módulo bcryptjs para manter nossas senhas seguras no banco de dados. Armazenar hashs das senhas ao invés de texto plano no banco de dados é uma política mínima visando segurança das informações.
- Definir estratégias local e através de APIs conhecidas.

```
const keys = require('./keys');
const bcrypt = require('bcryptjs');
const colors = require('colors');

const FacebookStrategy = require('passport-facebook').Strategy;
const Facebook = require('../models/facebook');

const GoogleStrategy = require('passport-google-oauth20').Strategy;
const Google = require('../models/google');

const GitHubStrategy = require('passport-github2').Strategy;
const Github = require('../models/github');

const LocalStrategy = require('passport-local').Strategy;
const User = require('../models/user');
```

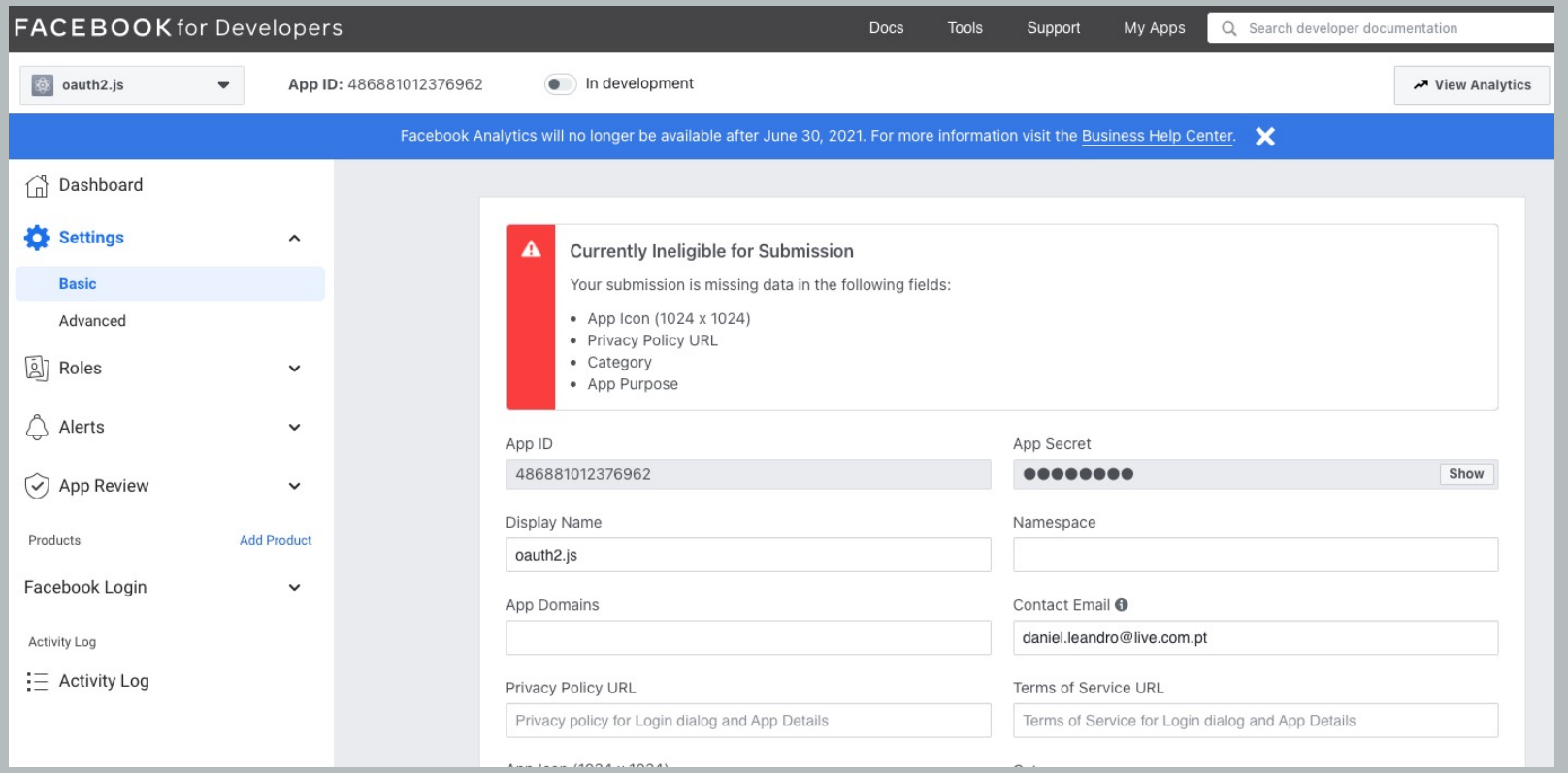
#PASSPORT-OAUTH2

- Parte da implementação de uma estratégia para autenticação através do Facebook.

```
passport.use(new FacebookStrategy({
    clientID: keys.facebook.clientID,
    clientSecret: keys.facebook.clientSecret,
    callbackURL: "/users/facebook/callback"

}, function(accessToken, refreshToken, profile, done) {

    // Verificar se o utilizador já se autenticou anteriormente pelo Facebook na nossa aplicação
    console.log('\nAutenticação Facebook'.cyan);
    Facebook.findOne({facebookId: profile.id}).then((currentUser) => {

        if(currentUser){
            console.log('Login efectuado novamente no Facebook. Dados de utilizador: \n'.cyan);
            console.log(currentUser);
            done(null, currentUser);

        // Primeira autenticação pelo Facebook, é necessário guardar a sua informação na base de dados
        } else {

            new Facebook({
                facebookId: profile.id,
                username: profile.displayName

            }).save().then((newFacebook) => {

                console.log('Login efectuado no Facebook por um novo utilizador: \n'.cyan);
                console.log(newFacebook);
                done(null, newFacebook);
            });
        }
    });
}));
```

#PASSPORT-OAUTH2

- Ainda no exemplo do Facebook, é necessário configuar e obter os tokens através do Facebook for DevelopersKeys



#PASSPORT-OAUTH2

- Os tokens obtidos é introduzido no ficheiro keys.js

```
facebook: {
    clientID: '486881012376962',
    clientSecret: 'e03a77ff431c0d7afac90d3482fb8b73'
```

- Exemplo de mensagem no servidor

```
Autenticação Facebook
Login efectuado novamente no Facebook. Dados de utilizador:

{
  _id: 60c1374133724d091c0c756f,
  facebookId: '10225858874289237',
  username: 'Daniel Leandro',
  __v: 0
}
```

#CONCLUSÃO

O OAuth2 é uma estrutura de autorização que permite que as aplicações obtenham um acesso limitado e fácil às contas do utilizador em um serviço HTTP.

Desta forma só é possível devido a três elementos, o utilizador, o cliente e o servidor. Traz bastantes benefícios e presentemente este mecanismo é usado com muita frequência.

Se é um processo seguro, segurança no ciberespaço nunca é garantida a 100%, principalmente quando entra o fator de "tempo", o oauth2 pode levar alguma polémica porque providers como Facebook ou Google, até que certo nível de controlo detalhado temos sobre os dados. Mas devido apenas passar um token e nunca compartilhar a senha acaba por ser um processo seguro em relação a algumas alternativas.

UNIVERSIDADE  LUSÓFONA

DANIEL LEANDRO N#21800566

MESTRADO DE ENGENHARIA DE INFORMÁTICA E SISTEMAS DE INFORMAÇÃO

08 DE JUNHO DE 2021