# OpenId connect (with auth0)

## Auth0

* Signup on auth0
* Create a new application

## Front end

* Add the library npm install auth0-js –-save
* Update your authentication services to make it work with auth0.

auth0 = new auth0.WebAuth({

clientID: 'YOUR-AUTH0-CLIENT-ID',

domain: 'YOUR-AUTH0-DOMAIN', // e.g., you.auth0.com

audience: 'YOUR-AUTH0-API-IDENTIFIER', // e.g, 'https://sprint- shcool.auth0.com/api/v2/

redirectUri: 'http://localhost:4200/callback,

responseType: 'token id\_token',

params: {scope: 'openid profile email user\_metadata app\_metadata picture user\_id'}

});

* Addapt your function to maked them work with auth0.

login() {

// Auth0 authorize request

this.auth0.authorize();

}

handleLoginCallback() {

// When Auth0 hash parsed, get profile

this.auth0.parseHash((err, authResult) => {

if (authResult && authResult.accessToken) {

window.location.hash = '';

this.getUserInfo(authResult);

} else if (err) {

console.error(`Error: ${err.error}`);

}

this.router.navigate(['/']);

});

}

getAccessToken() {

this.auth0.checkSession({}, (err, authResult) => {

if (authResult && authResult.accessToken) {

this.getUserInfo(authResult);

}

});

}

getUserInfo(authResult) {

// Use access token to retrieve user's profile and set session

this.auth0.client.userInfo(authResult.accessToken, (err, profile) => {

if (profile) {

this.\_setSession(authResult, profile);

}

});

}

private \_setSession(authResult, profile) {

// Save authentication data and update login status subject

this.expiresAt = authResult.expiresIn \* 1000 + Date.now();

this.accessToken = authResult.accessToken;

this.userProfile = profile;

this.authenticated = true;

}

logout() {

// Log out of Auth0 session

// Ensure that returnTo URL is specified in Auth0

// Application settings for Allowed Logout URLs

this.auth0.logout({

returnTo: 'http://localhost:4200',

clientID: environment.auth.clientID

});

}

get isLoggedIn(): boolean {

// Check if current date is before token

// expiration and user is signed in locally

return Date.now() < this.expiresAt && this.authenticated;

}

* Create a callback component that call handleLoginCallback(); in his ngOnInit() function and add it to your routes,
* Add de accessToken to the header of your http call. And import Headers in your service.

import { Headers } from "@angular/http";

## BackEnd

* Install the dependencies
* npm install express-jwt express-jwt-authz jwks-rsa –-save
* add this code to your routes

const jwt = require('express-jwt');

const jwtAuthz = require('express-jwt-authz');

const jwksRsa = require('jwks-rsa');

// Authentication middleware. When used, the

// Access Token must exist and be verified against

// the Auth0 JSON Web Key Set

const checkJwt = jwt({

// Dynamically provide a signing key

// based on the kid in the header and

// the signing keys provided by the JWKS endpoint.

secret: jwksRsa.expressJwtSecret({

cache: true,

rateLimit: true,

jwksRequestsPerMinute: 5,

jwksUri: `https://sprint-shcool.auth0.com/.well-known/jwks.json`

}),

audience: 'https://sprint-shcool.auth0.com/api/v2/',

issuer: `https://sprint-shcool.auth0.com/`,

algorithms: ['RS256']

});

* Call checkJWT in the routes you need to secure

Ex: app.get('/api/deals/private', authCheck, (req,res)=>{

Source:

* https://auth0.com/blog/angular-2-authentication/