**Importing the NgModule**

import { HttpClientModule } from '@angular/common/http';

import { OAuthModule } from 'angular-oauth2-oidc';

// etc.

@NgModule({

imports: [

// etc.

HttpClientModule,

OAuthModule.forRoot()

],

declarations: [

AppComponent,

HomeComponent,

// etc.

],

bootstrap: [

AppComponent

]

})

export class AppModule {

}

**Configuring for Password Flow**

const apiUrl = 'https://localhost:5001'; // serwer autoryzacji

const appUrl = 'http://localhost:4200'; // adres aplikacji

export function oidcConfig(url: string): NgxMAuthOidcConfig {

return {

origin: appUrl,

issuer: apiUrl + '/',

loginUrl: apiUrl + '/login',

redirectUri: appUrl + '/auth',

responseType: 'code',

clientId: id klienta,

dummyClientSecret: hasło klienta,

scope: 'openid profile email roles offline\_access' + inne uprawnienia do magazynu danych

};

}

Configure the OAuthService with this config object when the application starts up:

import { Component, OnInit } from '@angular/core';

import { Router } from '@angular/router';

import { NgxMAuthOidcService, NgxMAuthOidcConfig } from 'ngx-mauth-oidc';

import { AuthService } from './core/services/auth.service';

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.scss']

})

export class AppComponent implements OnInit {

constructor(

public router: Router,

public oidcService: NgxMAuthOidcService,

public authService: AuthService) { }

public ngOnInit(): void {

this.oidcService.configure(oidcConfig(apiUrl));

this.oidcService.setStorage(sessionStorage);

this.oidcService.loadDocumentAndTryLogin()

.then(d => {

console.log('loadDocumentAndTryLogin', d);

});

this.oidcService.events.subscribe(e => {

console.log('[app events]', e);

switch (e.type) {

case 'token\_received': {

this.oidcService.loadUserProfile()

.then()

.catch(error => console.log(error));

break;

}

case 'token\_expires': {

this.oidcService.scope = 'openid profile email roles offline\_access api-resource-server-A api-resource-server-B';

this.oidcService.refreshToken()

.then(result => console.log(e, result))

.catch(error => console.log(e, error));

break;

}

case 'token\_error':

case 'token\_refresh\_error': {

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

break;

}

case 'logout': {

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

break;

}

}

});

}

isLoggedIn(): boolean {

return this.authService.isLoggedIn();

}

get user() {

return this.authService.getUser();

}

public logout(): void {

this.authService.logout();

}

}

**Implementing a Login Form**

import { Component, OnInit } from '@angular/core';

import { Router, NavigationExtras } from '@angular/router';

import { AuthService } from '../../core/services/auth.service';

@Component({

selector: 'app-login-component',

styleUrls: ['./login.component.scss'],

templateUrl: './login.component.html'

})

export class LoginComponent implements OnInit {

email: string;

password: string;

loginProviders: any[] = [

{ name: 'Google', key: 'Google', icon: 'google' },

{ name: 'Facebook', key: 'Facebook', icon: 'facebook' },

{ name: 'Twitter', key: 'Twitter', icon: 'twitter' },

{ name: 'Microsoft', key: 'Microsoft', icon: 'microsoft' }

];

constructor(

private router: Router,

private authService: AuthService) { }

public ngOnInit() { }

public login(): void {

this.authService.login(this.email, this.password).then(result => {

if (result) {

// Redirect the user

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

} else {

}

}).catch(error => {

console.log('error', error.error);

});

}

public loginSocial(providerKey: string): void {

this.authService.loginSocial(providerKey);

}

}

Template for login page:

<form #loginForm="ngForm" class="ui medium form">

<div class="ui basic">

<div class="field">

<label>Adres e-mail</label>

<div class="ui left icon input">

<i class="user icon"></i>

<input name="email" [(ngModel)]="email" placeholder="Adres e-mail" type="text">

</div>

</div>

<div class="field">

<label>Hasło</label>

<div class="ui left icon input">

<i class="lock icon"></i>

<input name="password" [(ngModel)]="password" placeholder="Hasło" type="password">

</div>

</div>

<div class="ui fluid blue button" (click)="login()">

<i class="sign-in icon"></i>

Zaloguj się

</div>

<div class="ui error message"></div>

</div>

</form>

<div class="four ui buttons">

<ng-template ngFor let-provider let-last="last" [ngForOf]="loginProviders">

<button class="ui button social"

[attr.data-tooltip]="provider.name"

data-variation="basic"

data-position="bottom left"

(click)="loginSocial(provider.key)">

<i class="icon {{ provider.icon }}"></i>

</button>

<span class="size-height-12"></span>

</ng-template>

</div>

import { Inject, Injectable, EventEmitter } from '@angular/core';

import { HttpClient, HttpHeaders, HttpParams } from '@angular/common/http';

import { NgxMAuthOidcService } from 'ngx-mauth-oidc';

import { BaseService } from './base.service';

@Injectable()

export class AuthService {

constructor(

public http: HttpClient,

public oidcService: NgxMAuthOidcService) { }

isLoggedIn(): boolean {

return this.oidcService.hasValidAccessToken();

}

getUser() {

return this.oidcService.getIdentityClaims();

}

login(email: string, password: string): Promise<object> {

return this.oidcService.fetchTokenUsingPasswordFlow(email, password); // Password Flow

}

loginSocial(providerKey) {

const params = { provider: providerKey };

return this.oidcService.initAuthorizationCode(params); // Authorization Code

// return this.oidcService.initImplicitFlow(params); // Implicit Flow

}

logout(): void {

this.oidcService.logout();

}

}

**Request service**

import { Inject, Injectable } from '@angular/core';

import { HttpClient, HttpHeaders, HttpParams } from '@angular/common/http';

import { NgxMAuthOidcService } from 'ngx-mauth-oidc';

import { APP\_CONFIG, IAppConfig } from '../../../app/app.config';

import { BaseService } from './base.service';

@Injectable()

export class ResourceServerService extends BaseService {

constructor(

protected http: HttpClient,

protected oidcService: NgxMAuthOidcService,

@Inject(APP\_CONFIG) protected config: IAppConfig) {

super(http, oidcService, config);

}

getResourcePrivateA(): Promise<object> {

const headers = new HttpHeaders().set(

'Authorization', 'Bearer ' + this.oidcService.getAccessToken()

);

return this.http.get('https://localhost:5011/api/resource/private', { headers }).toPromise();

}

getResourcePrivateB(): Promise<object> {

const headers = new HttpHeaders().set(

'Authorization', 'Bearer ' + this.oidcService.getAccessToken()

);

return this.http.get('https://localhost:5021/api/resource/private', { headers }).toPromise();

}

}