public class DataContext :DbContext

{

public DataContext(DbContextOptions<DataContext> option):base(option)

{

}

public DbSet<Value> Values { get; set; }

}

**Extension**

C# for Visual Studio Code

C# Extensions

vscode-nuget-package-manager

Angular TypeScript Snippets for VS Code

Angular Language Service

VS Code Angular Files

angular2-switcher

Auto Rename Tag

Bracket Pair Colorizer

VS Code - Debugger for Chrome

Material Icon Theme

Path Intellisense

Prettier formatter for Visual Studio Code

vscode-tslint

Startup.cs

services.AddDbContext<DataContext>(x => x.UseSqlite(Configuration.GetConnectionString("DefaultConnection")));

appsettings

"ConnectionStrings": {

"DefaultConnection": "Data Source=BlogApp.db"

},

dotnet ef migrations add InitialCreate  
dotnet ef database update

ValuesController

private readonly DataContext \_context;

public ValuesController(DataContext context)

{

\_context = context;

}

// GET api/values

[HttpGet]

public ActionResult GetValues()

{

var values = \_context.Values.ToList();

return Ok(values);

}

// GET api/values/5

[HttpGet("{id}")]

public ActionResult GetValue(int id)

{

var value = \_context.Values.FirstOrDefault(p => p.Id == id);

return Ok(value);

}

npm install -g @angular/cli@6.0.8

ng new BlogApp-SPA

ng serve –open

app.module.ts

app.module.ts

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

imports: [

BrowserModule,

HttpClientModule

],

values: any;

value.component.ts

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

constructor(private http: HttpClient) {}

getValues() {

this.http.get('http://localhost:5000/api/values').subscribe(

response => {

this.values = response;

},

error => {

console.log(error);

}

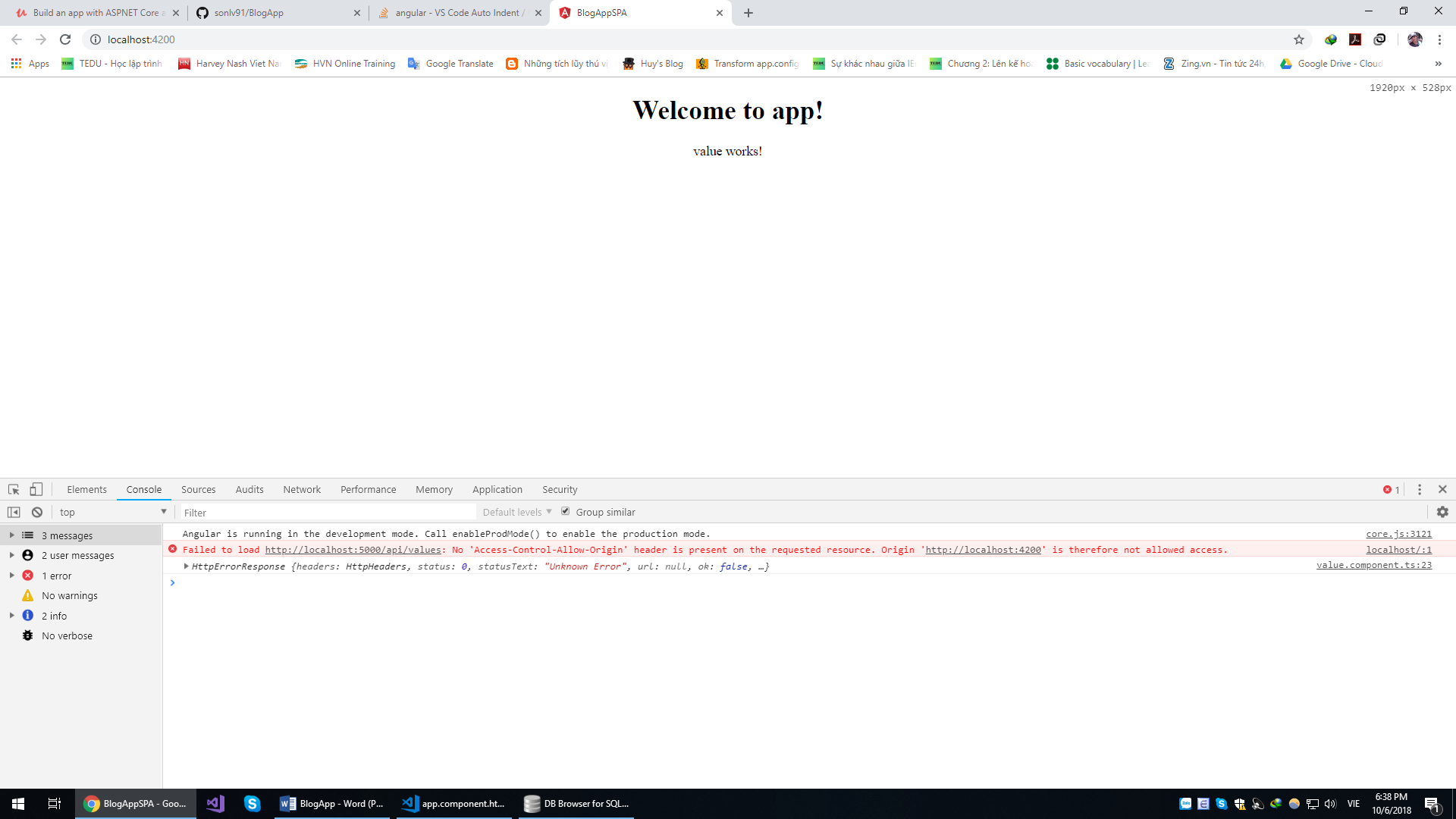
);

}

ngOnInit() {

this.getValues();

}



Startup.cs

services.AddCors();

app.UseCors(x =>x.AllowAnyOrigin().AllowAnyMethod().AllowAnyHeader());

value.component.html

<p \*ngFor="let value of values">

{{value.id}}, {{value.name}}

</p>

npm install bootstrap font-awesome

git status

.gitignore

.vscode

bin

obj

\*.db

git init

git remote add origin <https://github.com/sonlv91/BlogApp.git>

public class User

{

public int Id { get; set; }

public string Username { get; set; }

public byte[] PasswordHash { get; set; }

public byte[] PasswordSalt { get; set; }

}

public class DataContext :DbContext

{

public DataContext(DbContextOptions<DataContext> option):base(option)

{

}

public DbSet<Value> Values { get; set; }

public DbSet<User> Users { get; set; }

}

dotnet ef migrations add AddUserEntity

dotnet ef database update

public interface IAuthRepository

{

Task<User> Register(User user, string password);

Task<User> Login(string username, string password);

Task<bool> UserExists(string username);

}

public class AuthRepository : IAuthRepository

{

private readonly DataContext \_context;

public AuthRepository(DataContext context)

{

\_context = context;

}

public async Task<User> Login(string username, string password)

{

var user = await \_context.Users.FirstOrDefaultAsync(x => x.Username == username);

if (user == null)

return null;

if (!VerifyPasswordHash(password, user.PasswordHash, user.PasswordSalt))

return null;

return user;

}

private bool VerifyPasswordHash(string password, byte[] passwordHash, byte[] passwordSalt)

{

using (var hmac = new System.Security.Cryptography.HMACSHA512(passwordSalt))

{

var computedHash = hmac.ComputeHash(System.Text.Encoding.UTF8.GetBytes(password));

for (int i = 0; i < computedHash.Length; i++)

{

if (computedHash[i] != passwordHash[i])

return false;

}

}

return true;

}

public async Task<User> Register(User user, string password)

{

byte[] passwordHash, passwordSalt;

CreatePasswordHash(password, out passwordHash, out passwordSalt);

user.PasswordHash = passwordHash;

user.PasswordSalt = passwordSalt;

await \_context.Users.AddAsync(user);

await \_context.SaveChangesAsync();

return user;

}

private void CreatePasswordHash(string password, out byte[] passwordHash, out byte[] passwordSalt)

{

using (var hmac = new System.Security.Cryptography.HMACSHA512())

{

passwordSalt = hmac.Key;

passwordHash = hmac.ComputeHash(System.Text.Encoding.UTF8.GetBytes(password));

}

}

public async Task<bool> UserExists(string username)

{

if (await \_context.Users.AnyAsync(x => x.Username == username))

return true;

return false;

}

}

services.AddScoped<IAuthRepository,AuthRepository>();

public class UserForRegisterDto

{

[Required]

public string Username { get; set; }

[Required]

[StringLength(8, MinimumLength = 4, ErrorMessage = "You must specify password between 4 and 8")]

public string Password { get; set; }

}

ValuesController

[Authorize]

[Route("api/[controller]")]

[ApiController]

public class ValuesController : ControllerBase

[AllowAnonymous]

[HttpGet("{id}")]

public async Task<ActionResult> GetValue(int id)

Startup.cs

services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)

.AddJwtBearer(options =>

{

options.TokenValidationParameters = new TokenValidationParameters()

{

ValidateIssuerSigningKey = true,

IssuerSigningKey = new SymmetricSecurityKey(Encoding.ASCII

.GetBytes(Configuration.GetSection("AppSettings:Token").Value)),

ValidateIssuer = false,

ValidateAudience = false

};

});

nav.component.html

<form #loginForm="ngFrom" class="form-inline my-2 my-lg-0">

app.module.ts

import { FormsModule } from '@angular/forms';

imports: [

BrowserModule,

HttpClientModule,

FormsModule

],

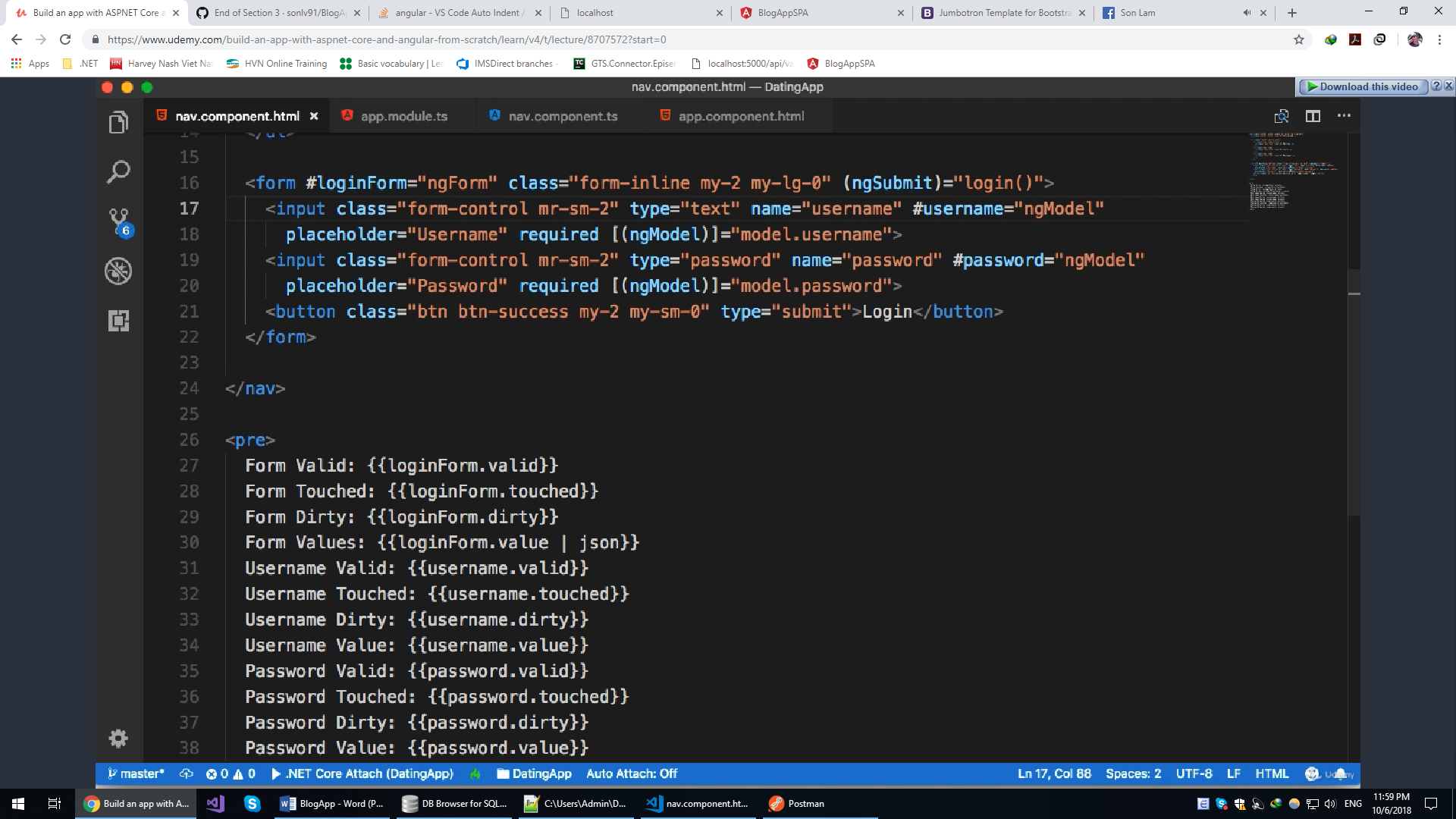
<form #loginForm="ngForm" class="form-inline my-2 my-lg-0" (ngSubmit)="login()">

<input class="form-control mr-sm-2" type="text" name="username" placeholder="Username" required [(ngModel)]="model.username">

<input class="form-control mr-sm-2" type="password" name="password" placeholder="Password" required [(ngModel)]="model.password">

<button class="btn btn-outline-success my-2 my-sm-0" type="submit">Login</button>

</form>



<button [disabled]="!loginForm.valid" class="btn btn-success my-2 my-sm-0" type="submit">Login</button>

app.module.ts

import { AuthService } from './\_services/auth.service';

providers: [

AuthService

],

auth.service.ts

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

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

import { map } from 'rxjs/operators';

@Injectable({

providedIn: 'root'

})

export class AuthService {

baseUrl = 'http://localhost:5000/api/auth';

constructor(private http: HttpClient) {}

login(model: any) {

return this.http.post(this.baseUrl + 'login', model).pipe(

map((response: any) => {

const user = response;

if (user) {

localStorage.setItem('token', user.token);

}

})

);

}

}

nav.component.ts

login() {

this.authService.login(this.model).subscribe(

next => {

console.log('Logged in successfully');

},

error => {

console.log('Failed to login');

}

);

}

loggedIn() {

const token = localStorage.getItem('token');

return !!token;

}

logout() {

localStorage.removeItem('token');

console.log('logged out');

}

nav.component.html

<div \*ngIf="loggedIn()" class="dropdown">

<a class="dropdown-toggle text-light">

Welcome User

</a>

<div class="dropdown-menu" aria-labelledby="dropdown01">

<a class="dropdown-item" href="#"><i class="fa fa-user"></i> Edit Profile</a>

<div class="dropdown-divider" href="#"></div>

<a class="dropdown-item" href="#"><i class="fa fa-signout"></i>Logout</a>

</div>

</div>

<li class="nav-item">

<a class="nav-link" (click)="logout()">Logout</a>

</li>

register.component.html

<form>

<h2 class="text-center text-primary">Sign Up</h2>

<hr>

<div class="form-group">

<input type="text" class="form-control" placeholder="Username">

</div>

<div class="form-group">

<input type="password" class="form-control" placeholder="Password">

</div>

<div class="form-group text-center">

<button class="btn btn-success" type="submit">Register</button>

<button class="btn btn-default" type="button">Cancel</button>

</div>

</form>

register.component.ts

export class RegisterComponent implements OnInit {

@Output() cancelRegister = new EventEmitter();

model: any = {};

constructor(private authService: AuthService) {}

ngOnInit() {}

register() {

this.authService.register(this.model).subscribe(

() => {

console.log('registration successfully');

},

error => {

console.log(error);

}

);

}

cancel() {

this.cancelRegister.emit(false);

console.log('cancelled');

}

}

<div class="form-group text-center">

<button class="btn btn-success mr-2" type="button" (click)="register()">Register</button>

<button class="btn btn-default" type="button" (click)="cancel()">Cancel</button>

</div>

Handle error

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

import {

HttpInterceptor,

HttpRequest,

HttpHandler,

HttpEvent,

HttpErrorResponse,

HTTP\_INTERCEPTORS

} from '@angular/common/http';

import { Observable, throwError } from 'rxjs';

import { catchError } from 'rxjs/operators';

@Injectable()

export class ErrorInterceptor implements HttpInterceptor {

intercept(

req: HttpRequest<any>,

next: HttpHandler

): Observable<HttpEvent<any>> {

return next.handle(req).pipe(

catchError(error => {

if (error instanceof HttpErrorResponse) {

if (error.status === 401) {

return throwError(error.statusText);

}

const applicationError = error.headers.get('Application-Error');

if (applicationError) {

console.error(applicationError);

return throwError(applicationError);

}

const serverError = error.error;

let modalStateErrors = '';

if (serverError && typeof serverError === 'object') {

for (const key in serverError) {

if (serverError[key]) {

modalStateErrors += serverError[key] + '\n';

}

}

}

return throwError(modalStateErrors || serverError || 'server Error');

}

})

);

}

}

export const ErrorInterceptorProvider = {

provide: HTTP\_INTERCEPTORS,

useClass: ErrorInterceptor,

multi: true

};

providers: [

AuthService,

ErrorInterceptorProvider

],

import { ErrorInterceptorProvider } from './\_services/error.interceptor';

npm install alertifyjs –save

angular.json

"scripts": ["node\_modules/alertifyjs/build/alertify.min.js"]

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

declare let alertify: any;

@Injectable({

providedIn: 'root'

})

export class AlertifyService {

constructor() {}

confirm(message: string, okCallback: () => any) {

alertify.confirm(message, function(e) {

if (e) {

okCallback();

} else {

}

});

}

success(message: string) {

alertify.success(message);

}

error(message: string) {

alertify.error(message);

}

warning(message: string) {

alertify.warning(message);

}

message(message: string) {

alertify.message(message);

}

}

register.component.ts

import { AlertifyService } from '../\_services/alertify.service';

constructor(

private authService: AuthService,

private alertify: AlertifyService

) {}

register() {

this.authService.register(this.model).subscribe(

() => {

this.alertify.success('registration successfully');

},

error => {

this.alertify.error(error);

}

);

}

<https://github.com/auth0/angular2-jwt>

npm install @auth0/angular-jwt

npm install ngx-bootstrap --save

npm install [bootswatch@4.1.1](mailto:bootswatch@4.1.1)

routes.ts

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

import { HomeComponent } from './home/home.component';

import { MemberListComponent } from './member-list/member-list.component';

import { MessagesComponent } from './messages/messages.component';

import { ListsComponent } from './lists/lists.component';

export const appRoutes: Routes = [

{ path: 'home', component: HomeComponent },

{ path: 'members', component: MemberListComponent },

{ path: 'messages', component: MessagesComponent },

{ path: 'lists', component: ListsComponent },

{ path: '\*\*', redirectTo: 'home', pathMatch: 'full' }

];

import { appRoutes } from './routes';

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

imports: [

BrowserModule,

HttpClientModule,

FormsModule,

BsDropdownModule.forRoot(),

RouterModule.forRoot(appRoutes)

],

app.component.html

<app-nav></app-nav>

<router-outlet></router-outlet>

nav.component.ts

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

constructor(

public authService: AuthService,

private alertify: AlertifyService,

private router: Router

) {}

login() {

this.authService.login(this.model).subscribe(

next => {

this.alertify.success('Logged in successfully');

},

error => {

this.alertify.error(error);

},

() => {

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

}

);

}

logout() {

localStorage.removeItem('token');

this.alertify.message('logged out');

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

}

ng g guard auth --spec=false

import { AuthGuard } from './\_guards/auth.guard';

export const appRoutes: Routes = [

{ path: '', component: HomeComponent },

{

path: '',

runGuardsAndResolvers: 'always',

canActivate: [AuthGuard],

children: [

{

path: 'members',

component: MemberListComponent

},

{ path: 'messages', component: MessagesComponent },

{ path: 'lists', component: ListsComponent }

]

},

{ path: '\*\*', redirectTo: '', pathMatch: 'full' }

];

auth.guard.ts

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

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

import { AuthService } from '../\_services/auth.service';

import { AlertifyService } from '../\_services/alertify.service';

@Injectable({

providedIn: 'root'

})

export class AuthGuard implements CanActivate {

constructor(

private authService: AuthService,

private router: Router,

private alerttify: AlertifyService

) {}

canActivate(): boolean {

if (this.authService.loggedIn()) {

return true;

}

this.alerttify.error('You shall not pass !!!');

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

return false;

}

}

dotnet ef –h

dotnet ef migrations add ExtendedUserClass

dotnet ef migrations list

dotnet ef database update

dotnet ef database drop

dotnet ef migrations remove

<https://www.json-generator.com> <= create data json

**Seeding Data to the Database**

Shaping the data to return with DTOs

Add files:

UserForDetailedDto.cs

UserForListDto.cs

PhotoForDetailedDto.cs

**Creating a new repository for our API**

BlogRepository.cs

IBlogRepository.cs

**Using AutoMapper**

add nuget

<PackageReference Include="AutoMapper.Extensions.Microsoft.DependencyInjection" Version="4.0.1"/>

Startup.cs

public void ConfigureServices(IServiceCollection services)

services.AddAutoMapper();

AutoMapperProfiles.cs

public AutoMapperProfiles()

{

CreateMap<User, UserForListDto>()

.ForMember(dest => dest.PhotoUrl, opt =>

{

opt.MapFrom(src => src.Photos.FirstOrDefault(p => p.IsMain).Url);

})

.ForMember(dest => dest.Age, opt =>

{

opt.ResolveUsing(d => d.DateOfBirth.CalculateAge());

});

CreateMap<User, UserForDetailedDto>()

.ForMember(dest => dest.PhotoUrl, opt =>

{

opt.MapFrom(src => src.Photos.FirstOrDefault(p => p.IsMain).Url);

})

.ForMember(dest => dest.Age, opt =>

{

opt.ResolveUsing(d => d.DateOfBirth.CalculateAge());

});

CreateMap<Photo, PhotoForDetailedDto>();

}

**Using Auth0 JwtModule to send up jwt tokens automatically**

export const environment = {

production: false,

apiUrl: 'http://localhost:5000/api/'

};

export class UserService {

baseUrl = environment.apiUrl;

JwtModule.forRoot({

config: {

tokenGetter: tokenGetter,

whitelistedDomains: ['localhost:5000'],

blacklistedRoutes: ['localhost:5000/api/auth']

}

})