Contents

[ Yeni ASP.NET API Proje olusturma 3](#_Toc14115145)

[ Environment i Development tan Production a alma. 4](#_Toc14115146)

[ Web de donen hatayi goruntuleyebilmek icin saklama 4](#_Toc14115147)

[ DBContext olusturma 4](#_Toc14115148)

[ SQLite ekleme (Nuget) : 4](#_Toc14115149)

[ Connection String ekleme (Sqlite icin): 4](#_Toc14115150)

[ Migration ekleme: 4](#_Toc14115151)

[ DBContext I Controller a Inject etme: 4](#_Toc14115152)

[ DBContext den (Databaseden) veri listeleme (API Controller API Actionlarinda) 4](#_Toc14115153)

[ Postman de GET metodu ile API dan veri alma 5](#_Toc14115154)

[ Synchronous action i Async e cevirme 5](#_Toc14115155)

[ Angular istenen version i npm ile ekleme 5](#_Toc14115156)

[ VS Code a eklenecek extensionlar 5](#_Toc14115157)

[ Angular yeni -component- yaratma 5](#_Toc14115158)

[ “@angular/http” ekleme (HttpClient icin gerekli) 5](#_Toc14115159)

[ Angular Component icinde property, method yaratma + Dep. Inj. 6](#_Toc14115160)

[ Angular Component i gosterme 7](#_Toc14115161)

[ Cross-Origin Resource Sharing (CORS) izin verme (iki farkli domain arasi) 7](#_Toc14115162)

[ Component taki API dan donen verileri gosterme (\*ngfor) 7](#_Toc14115163)

[ Bootstrap 4 ve Fontawesome ekleme (npm ile) 7](#_Toc14115164)

[ User class ekleme 7](#_Toc14115165)

[ Git ekleme 7](#_Toc14115166)

[ Repository pattern ekleme 7](#_Toc14115167)

[ User islemleri icin Repository pattern 8](#_Toc14115168)

[ User islemlerini servis olarak ekleme 9](#_Toc14115169)

[ Authentication (Auth) Controller ekleme ve Methodlari yaratma (Register) 9](#_Toc14115170)

[ DTO islemleri 10](#_Toc14115171)

[ VS Code ile debugging 11](#_Toc14115172)

[ ASP.NET.CORE API validation 11](#_Toc14115173)

[ Login API method yaratma 11](#_Toc14115174)

[ Authentication servis ekleme 12](#_Toc14115175)

[ Postman de Login olarak API dan veri isteme 13](#_Toc14115176)

[ Angular Frontend baslangic, navbar olusturma 13](#_Toc14115177)

[ Angular form olusturma (login username, password) 13](#_Toc14115178)

[ Angular services yaratma, Login method 14](#_Toc14115179)

[ Angular service i component ta kullanma 14](#_Toc14115180)

[ Angular login durumuna gore ngIf kullanimi 15](#_Toc14115181)

[ Angular home ve onun child i register component yaratmak 15](#_Toc14115182)

[ Angular home dan child (register) a veri gonderme (Input property) 16](#_Toc14115183)

[ Angular child (register) dan home a veri gonderme (Output property) 17](#_Toc14115184)

[ Angular auth service e register method ekleme 17](#_Toc14115185)

[ ASP.NET.CORE API’da error handling (try/catch) 18](#_Toc14115186)

[ ASP.NET.CORE API’da Global error handling (Extension method ile) 18](#_Toc14115187)

[ Angular’da Global error handling 19](#_Toc14115188)

[ Angular’da 3rd party libraryler ekleme (Alertify.js) 20](#_Toc14115189)

[ JWT ile token handling 21](#_Toc14115190)

[ JWT ile decode token (token dan username alma) 22](#_Toc14115191)

[ ngx Bootstrap (Jquery siz Bootstrap fonksiyonlari) 23](#_Toc14115192)

[ Bootswatch (free) design kullanimi 24](#_Toc14115193)

[ Angular routing 24](#_Toc14115194)

[ Angular linklere routing atama 24](#_Toc14115195)

[ Angular event ten sonra routing (login sonrasi members a git gibi) 25](#_Toc14115196)

[ Angular login olmadan bazi routelari gormeyi engelleme. 26](#_Toc14115197)

[ Angular login olmadan bazi routelari gormeyi engelleme (multiple route) 27](#_Toc14115198)

[ ASP API’da User class propertyleri arttirma 27](#_Toc14115199)

[ ASP API’da EF migration islemleri 28](#_Toc14115200)

[ ASP API’da EF relations 28](#_Toc14115201)

[ ASP API’da Seed data ile ornek veri yaratma 28](#_Toc14115202)

[ ASP API’da Seed data ile ornek veri yukleme 29](#_Toc14115203)

[ ASP API’da Dating Repository yaratma (User+Photo CRUD islemleri) 30](#_Toc14115204)

[ ASP API’da Users Controller 31](#_Toc14115205)

[ ASP API’da Users DTO 31](#_Toc14115206)

[ ASP API AutoMapper, User i DTO lara mapping 32](#_Toc14115207)

[ ASP API AutoMapper fazla gelenleri (navigation)cikartma, eksikleri ekleme 33](#_Toc14115208)

[ Angular Typescript Type lari kullanma 34](#_Toc14115209)

[ Angular User icin servis yaratma (environment.ts te API url set ederek) 35](#_Toc14115210)

[ Angular User servisi Componentlerde kullanma (token hatali) 36](#_Toc14115211)

[ Angular User lari card olarak listeleme (Member ust folder yaratma) 37](#_Toc14115212)

[ Angular User card lara CSS verme 37](#_Toc14115213)

[ Angular User card lara animated button ekleme 38](#_Toc14115214)

[ Angular User servisi Componentlerde kullanma (token hata duzeltme) 38](#_Toc14115215)

[ Angular member-detail e link 39](#_Toc14115216)

[ Angular member-detail sayfa tasarimi (sol bolum) 40](#_Toc14115217)

[ Angular member-detail sayfa tasarimi (sag bolum) 41](#_Toc14115218)

[ Angular Root resolver ile component e veri alma (Tamamen opsiyonel user? da olur) 43](#_Toc14115219)

[ Angular User photo gallery ekleme 45](#_Toc14115220)

[ Angular Member Edit Component routing 46](#_Toc14115221)

[ Angular Member Edit Component HTML Part1 47](#_Toc14115222)

[ Angular Member Edit Component HTML Part2 (Degisiklik yapinca alert) 48](#_Toc14115223)

[ Angular Member Edit ten baska route a giderken form dirty ise uyari verme 49](#_Toc14115224)

[ ASP API Member Edit degisiklikleri API a yazma 50](#_Toc14115225)

[ Angular Member Edit degisiklikleri API a gonderme 51](#_Toc14115226)

[ ASP API Photo update Cloudinary settings 51](#_Toc14115227)

[ ASP API Photos Controller 1 52](#_Toc14115228)

[ ASP API Photos Controller 2 54](#_Toc14115229)

[ ASP API Photos Controller Duzeltme 54](#_Toc14115230)

[ Angular Photo Gallery Tasarim 55](#_Toc14115231)

[ Angular Photo ekleme 55](#_Toc14115232)

[ ASP API Photo yu Main Photo yapma 55](#_Toc14115233)

[ Angular Photo yu Main Photo yapma 56](#_Toc14115234)

[ Angular Photo yu Main Photo yapinca UI de (parent comp.) deigisiklik yapma 57](#_Toc14115235)

[ Main photo yu navbar a ekleme (ASP + Angular) 58](#_Toc14115236)

[ Angular Main photo yu navbar da degistirme (Componentler arasi veri iletisimi) 59](#_Toc14115237)

[ ASP API Photo silme 60](#_Toc14115238)

[ Angular Photo silme 61](#_Toc14115239)

[ Angular Reactive register form 62](#_Toc14115240)

[ Angular register form Validators 62](#_Toc14115241)

[ Angular register form custom validation 63](#_Toc14115242)

[ Angular reactive form FormBuilder service 64](#_Toc14115243)

[ Angular registration field ekleme 64](#_Toc14115244)

[ Angular datepicker input 65](#_Toc14115245)

[ ASP API ilave register field lari ekleme 66](#_Toc14115246)

[ Angular register 67](#_Toc14115247)

[ Angular yeni register kullaniciya default photo getirme + ilk photo islemleri 67](#_Toc14115248)

[ Angular tarihleri duzgun formatta goruntuleme (Angular pipes |) 68](#_Toc14115249)

[ ASP API last active maintain etme (Action Filter). 68](#_Toc14115250)

[ ASP API Paging 69](#_Toc14115251)

[ ASP API Paging Headers 70](#_Toc14115252)

[ ASP API Paging Headers API dan dondurme 71](#_Toc14115253)

[ Angular Paging 71](#_Toc14115254)

[ Angular Paging Plugin 72](#_Toc14115255)

[ ASP API Filtering (Defaullt Users Gender + not current user) 73](#_Toc14115256)

[ ASP API Filtering (Age + Gender) 74](#_Toc14115257)

[ Angular Filtering (Age + Gender) 74](#_Toc14115258)

[ ASP API Sorting (Ordering) 76](#_Toc14115259)

[ Angular Sorting (Ordering) 77](#_Toc14115260)

[ ASP API Many to Many relation yaratma (Likes icin) 78](#_Toc14115261)

[ ASP API Like gonderme 79](#_Toc14115262)

[ ASP API Like edilen ve eden kullanicilari listeleme 79](#_Toc14115263)

[ Angular kullanici like etme 81](#_Toc14115264)

[ Angular Like edilen ve eden kullanicilari listeleme 81](#_Toc14115265)

[ ASP API Messages Entity yaratma 83](#_Toc14115266)

### Yeni ASP.NET API Proje olusturma

Dotnet new webapi -o DatingApp.API -n DatingApp.API

-o: Location to place the output, -n : name of output

### Environment i Development tan Production a alma.

launchSettings.json

DatingApp.API : (bu ayarda yapilan degisiklikler var sadece)

"launchBrowser": false,

"applicationUrl": "http://localhost:5000",

"environmentVariables": {

"ASPNETCORE\_ENVIRONMENT": "Development"

environmentVariables buradan Production yapilacak

### Web de donen hatayi goruntuleyebilmek icin saklama

Chrome developer Network’te “Preserve Log” checkbox secilecek

### DBContext olusturma

* + Data folderinin altinda DBContext class yaratilir (DataContext.cs)
  + Ctor ve Dbset<(Tekil Class Adi)> (Cogul Class Adi) ler olusturulur
  + Startup.cs ConfigureServices’ta services.AddMVC’ den once

services.AddDbContext<DataContext>(x => x. UseSqlite("ConnectionString"));

### SQLite ekleme (Nuget) :

Nuget Package dan eklenir. Sonra UseSqlite cozumlenecek

### Connection String ekleme (Sqlite icin):

appsettings.json

"ConnectionStrings": {

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

},

Startup.cs

public void ConfigureServices(IServiceCollection services)

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

### Migration ekleme:

Terminalde :

* + dotnet ef migrations add InitialCreate
  + dotnet ef database update

### DBContext I Controller a Inject etme:

ValuesController.cs

private readonly DataContext \_context;

public ValuesController(DataContext context)

{

\_context = context;

}

### DBContext den (Databaseden) veri listeleme (API Controller API Actionlarinda)

ValuesController.cs

Hepsini listele

// GET api/values

[HttpGet]

public IActionResult GetValues()

{

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

return Ok(values);

}

Tek bir kayit listele

// GET api/values/5

[HttpGet("{id}")]

public IActionResult GetValue(int id)

{

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

return Ok(value);

}

### Postman de GET metodu ile API dan veri alma

GET secili iken:

http://localhost:5000/api/values (Send)

### Synchronous action i Async e cevirme

// GET api/values

[HttpGet]

public async Task<IActionResult> GetValues()

{

var values = await \_context.Values.ToListAsync();

return Ok(values);

}

### Angular istenen version i npm ile ekleme

npm install -g @angular/cli@6.0.8

Sonraki asamada projenin oldugu foldera gelinir, yeni proje yaratilir :

ng new DatingApp-SPA

### VS Code a eklenecek extensionlar

* + C#
  + C# extensions
  + Nuget Package Manager
  + Angular Snippets (John Papa)
  + Angular Files (Alexander Ivanicev)
  + Angular Language Service
  + Angular-2 switcher
  + Auto Rename Tag
  + Bracket Pair Colorizer
  + Debugger for chrome
  + material icon theme
  + path intellisense
  + Prettier - Code formatter
  + tslint

### Angular yeni -component- yaratma

* + src/app folderi sag tiklanir. Extension fonksiyonu “Generate Component” secilir.
  + Bu islem bu component i “app.module.ts” in altina da otomatik olarak ekler.

### “@angular/http” ekleme (HttpClient icin gerekli)

Angular 7 surumunde “@angular/http” depreciate olup common icine alinacak ??

app.module.ts

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

imports: [

BrowserModule,

HttpClientModule

],

### Angular Component icinde property, method yaratma + Dep. Inj.

Component in “ts” dosyasi secilir

value.component.ts

export class ValueComponent implements OnInit {

values: any;

constructor(private http: HttpClient) { }

ngOnInit() { }

getValues() {

this.http.get('http://localhost:5000/api/values').subscribe(response => {

this.values = response;

}, error => {

console.log(error);

});

}

}

PROPERTY yaratma : “values” classtaki property nin adi, type i “any” (gercek type I belirtilebilir)

values: any;

Dependency Injection : constructora “http” adinda HttpClient inject edilir.

constructor(private http: HttpClient) { }

Method yaratma

“getvalues” methodunda “this.http” constructordan gelen http (HttpClient). Bunun get methodu cagrildiginda overridlarda method icin gerekenler gosterilir. Burada ilk parametre URL mutlaka girilmesi lazim. Get methodunun donus tipi “Obsevable”, bu donen observable kullanilabilir bir data degil, bunun yerine bu observable a subscribe etmeliyiz. Subscribe in ilk parametresi callback function, ikincisi error. Ilk callbackte subscribedan donen response I values a atiyoruz.

getValues() {

this.http.get('http://localhost:5000/api/values').subscribe(response => {

this.values = response;

}, error => {

console.log(error);

});

}

Getvalues methodunu component yuklendiginde ngOnInit ten cagiriyoruz. NgOnInit constructordan hemen sonra cagrilir (component initialize olduktan sonra), Constructor icerisinden API den veri alma methodu cagrilamaz (erken olur).

ngOnInit() {

this.getValues();

}

### Angular Component i gosterme

app.component.html

icerisine componentin selector: 'app-value' degeri html tag olarak yazilir.

<app-value></app-value>

### Cross-Origin Resource Sharing (CORS) izin verme (iki farkli domain arasi)

startup.cs

public void ConfigureServices(IServiceCollection services)

{

services.AddCors();

}

public void Configure(IApplicationBuilder app, IHostingEnvironment env)

{

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

app.UseMvc();

### Component taki API dan donen verileri gosterme (\*ngfor)

value.component.html

\*ngfor ile valuelar listelenir. Ngfor “structural directive” dir, basinda \* vardir. Structural directive ler DOM u degistirir.

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

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

</p>

Values Class taki property nin adi

### Bootstrap 4 ve Fontawesome ekleme (npm ile)

D:\Projects\DatingApp\DatingApp-SPA> npm install bootstrap font-awesome

styles.css

@import '../node\_modules/bootstrap/dist/css/bootstrap.min.css';

@import '../node\_modules/font-awesome/css/font-awesome.min.css';

### User class ekleme

public class User

{

public int Id { get; set; }

public string UserName { get; set; }

public byte[] PasswordHash { get; set; }

public byte[] PasswordSalt { get; set; }

}

DataContext.cs

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

Migration eklenir

dotnet ef migrations add AddedUserEntity

dotnet ef database update

### Git ekleme

Git kurulur, (<https://git-scm.com/>) changeler stage edilir, gitignore ASP.NET API a eklenir, github repo register edilir, degisiklikler github repo ya push edilir.

### Repository pattern ekleme

Repository pattern databaseden veri alisverisini Controllerdan ayirmak icin kullanilir. Butun database islemleri burada Interface olarak yer alacak, Controller veritabani islemleri icin sadece bu interfaceleri kullanacak (kontratlari). Interface olmasinin sebebi, bu Interface I implement edecek concrete class bu projede EF icin olacak, ileride EF yi degistirmek istersek interface baska concrete class tarafindan implement edilecek.

### User islemleri icin Repository pattern

User islemleri icin repository intertface: (Data Folderi altinda yaratiliyor)

public interface IAuthRepository

{

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

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

Task<bool> UserExists(string username);

}

Yine Data Altinda Interface i implement edecek Class yaratiliyor:

Once datacontext inject edilir:

public class AuthRepository : IAuthRepository

{

private readonly DataContext \_context;

public AuthRepository(DataContext context)

{

\_context = context;

}

Sonra methodlarin uygulamalarina gecilir:

Register methodu uygulanirken password hash yaratmak gerekiyor. Buradada CreatePasswordHash methodunda out kullanmamizin sebebi degiskenlere referans vermek, bu degiskenler methodda degisince taniminda da degerleri degisecek. Microsoft : “Declaring a method with out arguments is a classic workaround to return multiple values. Beginning with C# 7.0, consider [tuples](https://docs.microsoft.com/en-us/dotnet/csharp/tuples) for similar scenarios.” The Out parameter is mostly used in the Try-pattern. In this pattern, a method returns a bool indicating success or failure and an Out variable that provides the result if the method succeeds. bool isValid = **int**.TryParse(ReadLine(), out **int** age);

CreatePasswordHash methodundaki hash yaratma sadece ornek bir yontem, farkli sekillerde hash yaratilabilir.

Register method

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));

}

}

Login method :

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;

}

UserExists method

public async Task<bool> UserExists(string username)

{

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

return true;

return false;

}

### User islemlerini servis olarak ekleme

Startup.cs

services.AddScoped<IAuthRepository, AuthRepository>();

### Authentication (Auth) Controller ekleme ve Methodlari yaratma (Register)

Controllers altinda AuthController yaratilir.

Buradaki methodlardaki Database islemleri (User ekleme, savechanges, vs.) repository deki methodlar tarafindan yapiliyor, bu yuzden Context buraya inject edilmiyor, context repoya inject ediliyor. Burada repo nun Interface olarak inject edilme sebebi de bu, .Net.Core DI bunu servislerde cozumluyor. En sondaki ReturnStatusCode gecici cozum, sonradan CreatedAtRoute yapilacak.

Route olarak methodlarin basina HttpPosttan sonra route yaziliyor (attribute routing).

public class AuthController : ControllerBase

{

private readonly IAuthRepository \_repo;

public AuthController(IAuthRepository repo)

{

\_repo = repo;

}

[HttpPost("register")]

public async Task<IActionResult> Register(string username, string password)

{

//validate request

username = username.ToLower();

if (await \_repo.UserExists(username))

return BadRequest("Username already exists");

var userToCreate = new User

{

UserName = username

};

var createdUser = \_repo.Register(userToCreate, password);

return StatusCode(201);

}

}

### DTO islemleri

Register method signature da username ve string gelecekmis gibi parmetreler var, oysa ki user JSON object gondericek, parametre degil. Bu kullanicidan donecek olan JSON objecti bize lazim olan parametrelere cevirmek icin DTO lar kullanilir.

Register(string username, string password)

Data altinda DTOs folderi yaratilir. UserForRegisterDTO.cs yaratilir

public class UserForRegisterDto

{

public string Username { get; set; }

public string Password { get; set; }

}

Register method da parametreler DTO olarak degistirilir. ASP.NET.Core parametredeki objeyi kendi bulur, yani parametreye UserForRegisterDTO object yailir, kullanici username ve password iceren JSON gonderdiginde ASP.NET bunu otomatik olarak UserForRegisterDTO objesinin parametreleri olarak algilar (exstradan [FromBody] koymaya gerek olmadan.

Dto lu register method asagidaki gibi olacak.

[HttpPost("register")]

public async Task<IActionResult> Register(UserForRegisterDto userForRegisterDto)

{

//validate request

userForRegisterDto.Username = userForRegisterDto.Username.ToLower();

if (await \_repo.UserExists(userForRegisterDto.Username))

return BadRequest("Username already exists");

var userToCreate = new User

{

UserName = userForRegisterDto.Username

};

var createdUser = \_repo.Register(userToCreate, userForRegisterDto.Password);

return StatusCode(201);

}

### VS Code ile debugging

Soldaki Debug icon secilir. No Configuration a basilarak “Add Configuration” secilir.Buradan “.NET.Core” secilir. Psotman de Method Post secilir, Body raw JSON secilir ve username, password girilir.

### ASP.NET.CORE API validation

API dan gelen verilerin dogrulugunu kontrol etmek icin iki yol var. Ya direk Entity class modelde kontrol edebiliriz veya DTO da kontrol edebiliriz. Bunu DTO da yapmak daha efektif. Data Annotation ile yapacagiz [Required], vs.

public class UserForRegisterDto

{

[Required]

public string Username { get; set; }

[Required]

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

public string Password { get; set; }

}

Burada donen veriyi otomatik olarak validate eden [ApiController]. Bu oldugu icin ModelState.IsValid kontrolu yapmamiza gerek kalmiyor.

[ApiController] i kaldirisak hatalar ortaya cikacak. Hatalari bulmak icin cesitli yollar var :

Postman de donen hatayi (500 server) Body/Previewdan detayli olarak gormek mumkun.

Bu hata Angular tarafinda olsaydi bunu goremeyebilirdik. Burada yapilmasi gereken terminalde 500 hatasini bulup bununla ilgili hata degerlerini bulmak olacak (Object reference sert to null, vb.) Bu tip durumlarda if(modelstate.isvalid) ile de gelen degerlerin kontrolu yapilabilir.

### Login API method yaratma

UserForLoginDto yaratilir

public class UserForLoginDto

{

public string Username { get; set; }

public string Password { get; set; }

}

Configuration Inject edilir (Token i appsettingste configuration olarak kullanmak icin)

private readonly IAuthRepository \_repo;

private readonly IConfiguration \_config;

public AuthController(IAuthRepository repo, IConfiguration config)

{

\_config = config;

\_repo = repo;

}

Appsettings.json (Normal uyugulamada Token cok uzun olmasi gerekir)

"AppSettings:": {

"Token": "super secret key"

},

Login method olusturulur

[HttpPost("login")]

public async Task<IActionResult> Login(UserForLoginDto userForLoginDto)

{

var userFromRepo = await \_repo.Login(userForLoginDto.Username.ToLower(), userForLoginDto.Password);

if (userFromRepo == null)

return Unauthorized();

var claims = new[]

{

new Claim(ClaimTypes.NameIdentifier, userFromRepo.Id.ToString()),

new Claim(ClaimTypes.Name, userFromRepo.UserName)

};

var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(\_config.GetSection("AppSettings:Token").Value));

var creds = new SigningCredentials(key, SecurityAlgorithms.HmacSha512Signature);

var tokenDescriptor = new SecurityTokenDescriptor

{

Subject = new ClaimsIdentity(claims),

Expires = DateTime.Now.AddDays(1),

SigningCredentials = creds

};

var tokenHandler = new JwtSecurityTokenHandler();

var token = tokenHandler.CreateToken(tokenDescriptor);

return Ok(new

{

token = tokenHandler.WriteToken(token)

});

}

### Authentication servis ekleme

Startup.cs

public void ConfigureServices(IServiceCollection services)

{

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

};

});

public void Configure(IApplicationBuilder app, IHostingEnvironment env)

{

app.UseAuthentication();

app.UseMvc();

### Postman de Login olarak API dan veri isteme

Postman de auth/login e Post istegi atilir (body de Json login bilgileri ile). Donen token copy edilir. Authorization gereken API ye Headers a gidilir, Key olarak “Authorization”, value yerine de Bearer (bosluk) token yazilir. 2. Yontem de Authorization tab da Bearer Token secilir, token buraya girilir.

### Angular Frontend baslangic, navbar olusturma

Bootstrap4 ten Examples/Jumbotran a gidilir. Buradan Navbar elementi Copy element ile alinir. Angular-SPA projesinde src/app sag tiklanir, Generate component denilir ve “nav” eklenir. Nav.component.html e bootstrap ten kopyalanan nav paste edilir. Sonrasinda “app.component.html” icerisine nav-component in selectoru <app-nav></app-nav> eklenir.

### Angular form olusturma (login username, password)

Formun ilk HTML hali (bootstrap den alinan)

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

<input class="form-control mr-sm-2" type="text" placeholder="Username">

<input class="form-control mr-sm-2" type="text" placeholder="Password">

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

</form>

Once formun Angular form olmasi saglanir. Angular da 2 tip form var, birisi template tip, ikincisi reactive tip, burada template tip kullanilacak. loginForm Tempalte reference variable kullanilir (# ile baslar), bu variable a ngForm directive degeri verilir.

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

Bu asamada Angular form kullanacagimizi kayit etmeliyiz.

app.module.ts

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

imports: [

BrowserModule,

HttpClientModule,

FormsModule ],

Angular form lar 2-way-binding desteklemektedir. Bunu kullanabilmek icin fieldlara ngModel binding yapmak gerekli. Bunun kisa yolu : input elementin en sonunda “a-ngmo..” yazilinca snipetten olusturmak. Username i model objesinin username ine bind ediyoruz

[(ngModel)]=”model.username”

Passwordu de model objesinin passwordune bind ediyoruz. Burada ASP de oldugu gibi binding yapmak icin field adlarini name attributelari olarak belirtmek gerekiyor.

Form submit eventi en ust satirda login methoduna baglaniyor : Ustteki form tagin en sonunda (ngSubmit)=”login()”

Ayrica HTML5 teki required attributelar kullaniliyor, Angular bunlari validation icin kullanacak. loginform.valid dogrulamasini buton disabled a koyabiliyoruz

<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="text" name="password" placeholder="Password" required [(ngModel)]="model.password">

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

### Angular services yaratma, Login method

App folder I altinda \_services folderi olusturulur (\_ en ustte kalmasi icin). Bu folder sag tiklanir ve “Generate Service” secilir (VSCode extension dan gelen method). Burada yaratilan methodda “@Injectable” decorator i yer alir, bunun sebebi service icerisine Inject yapilabilmesi icin. Component lerde Inject yok cunku onlara otomatik Inject yapilabiliyor. providedIn: 'root' bunun hangi modul tarafindan saglandigini bildiriyor, root yani app.module. Bu sebepten servisi app.module a da register etmemiz gerekiyor.

app.module.ts

providers array icerisine service yaratilir.

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

providers: [

AuthService

],

Auth service icerisine oncelikle HttpClient inject edilir, daha sonra API dan donen object (“token”: degeri JSON olarak) uzerinde islem yapabilmek icin pipe chain olarak kullanilir.

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);

}

})

);

}

}

### Angular service i component ta kullanma

Nav component te auth service I kullanmak icin : Ilk yapilacak sey constructor a servisi inject etmek.

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

constructor(private authService: AuthService) { }

sonra method implement edilir

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

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

@Component({

selector: 'app-nav',

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

styleUrls: ['./nav.component.css']

})

export class NavComponent implements OnInit {

model: any = {};

constructor(private authService: AuthService) { }

ngOnInit() {

}

login() {

this.authService.login(this.model).subscribe(next => {

console.log('Logged in succesfully');

}, error => {

console.log('Failed to login');

});

}

}

### Angular login durumuna gore ngIf kullanimi

Bootstrap dan dropdown kopyalanir.

Token dolu ise true degil ise false doner

return !!token;

loggedIn() {

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

return !!token;

}

logout() {

localStorage.removeItem('token');

console.log('Logged out');

}

ngif direktifi div e eklenir, ngif true ise tum div gosterilir false ise div gosterilmez. Ngif snippet icin “a-ngif”. Ngif structural directive oldugu icin basinda \* var.

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

<form \*ngIf="!loggedIn()" #loginForm="ngForm" class="form-inline my-2 my-lg-0" (ngSubmit)="login()">

Logout ornek olarak

<li class="nav-item">

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

</li>

### Angular home ve onun child i register component yaratmak

Register component home component in child componenti olacak. Home ve register componentlar yaratilir (HTML olarak).

register.component.html (gerisi onceki form ile ayni)

<form #registerForm="ngForm" (ngSubnit)="register()">

Home html de ayni sayfada divlerin dugmeye basinca goruinurlugun toggle etmek icin default degeri false olan registerMode yaratilir. Dugmeye basinca bu true/false toggle olur, ngIf ile gorunurluk degisir.

home.component.html

<div class="container mt-5">

<div \*ngIf="!registerMode" style="text-align: center">

<h1>Find your match</h1>

<p class="lead">Come on in to view your matches... All you need to do is sign up!</p>

<div class="text-center">

<button class="btn btn-primary btn-lg mr-2" (click)="registerToggle()" >Register</button>

<button class="btn btn-info btn-lg">Learn more</button>

</div>

</div>

<div \*ngIf="registerMode" class="container">

<div class="row justify-content-center">

<div class="col-4">

<app-register></app-register>

</div>

</div>

</div>

</div>

home.component.ts

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

@Component({

selector: 'app-home',

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

styleUrls: ['./home.component.css']

})

export class HomeComponent implements OnInit {

registerMode = false;

constructor() { }

ngOnInit() {

}

registerToggle() {

this.registerMode = !this.registerMode;

}

}

### Angular home dan child (register) a veri gonderme (Input property)

Home component de registeri cagirdigimiz tag icine input property [] icinde yaizlir. Child property de bu degerleri almak icin @Input kullanilir ve Ana component deki degisken burada da tanimlanir.

home.component.html

<app-register [valuesFromHome]="values"></app-register>

register.component.ts

export class RegisterComponent implements OnInit {

@Input() valuesFromHome: any;

valuesFromHome icindeki degerleri register html de kullanimi icin ornek dropdown :

register.component.html

<div class="form-group">

<label for="favouriteValue">What is your favourite value?

<select class="form-control" id="favouriteValue">

<option \*ngFor="let value of valuesFromHome" [value]="value">{{value.name}}</option>

</select>

</label>

</div>

### Angular child (register) dan home a veri gonderme (Output property)

Output properties emit events. Output property = new EventEmitter derken EventEmitterlara dikkat etmek lazim 3 tane cikacak @angular/core dogru olani.

register.component.ts

@Output() cancelRegister = new EventEmitter();

this.cancelRegister.emit(false). Burada sadece boolean false degereini emit ediyoruz, hersey emit edilebilir, object dahil.

cancel() {

this.cancelRegister.emit(false);

console.log('Cancelled!');

}

Output property kullanimi inputinkine benzer. Input ta [] kullaniliyor, Output ta () kullaniliyor. Yanliz burada exstradan cagiran ana componentte bir tane daha method yaratmak gerekiyor, Output property degeri cagiran componentte method olmali. Buradaki methoddaki registermode Output property den gelecek, this.registerMode component in kendi propertysi.

home.component.html

<app-register [valuesFromHome]="values" (cancelRegister)="cancelRegisterMode($event)"></app-register>

home.component.ts

cancelRegisterMode(registerMode: boolean) {

this.registerMode = registerMode;

}

### Angular auth service e register method ekleme

auth.service.ts

register(model: any) {

return this.http.post(this.baseUrl + 'register', model);

}

Bu methodun sonucu da Observable donecegi icin, bunu kullanacak componentte buna da subscribe edilmesi gerekiyor.

register.component.html

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

register.component.ts

constructor(private authService: AuthService) { }

register() {

this.authService.register(this.model).subscribe(() => {

console.log('registration successful');

}, error => {

console.log(error);

});

}

Burada () => success callback function i gosteriyor, login de API den donen token i parametre almisti, burada parametre almiyor (.pipe kullanilmiyor).

### ASP.NET.CORE API’da error handling (try/catch)

Development environment ta ASP.NET otomatik olarak developerexception page dondurur. Bunun yerine hata olabilecek kodu tr Try bloku icine alabiliriz, boyle yaparsak hemen sonrasinda catch blok u icerisinde try blokunda hata olmasi durumunda yapilacak islemleri yazariz.

### ASP.NET.CORE API’da Global error handling (Extension method ile)

Startup.cs

Global olarak development haricindeki environment icin errfor handler yailir. Burada exceptionhandler parmaetrelerinden builder ve context e gidilir. Asagidaki code context e error mesaji yazmak icin.

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

else

{

app.UseExceptionHandler(builder => {

builder.Run(async context => {

context.Response.StatusCode = (int)HttpStatusCode.InternalServerError;

var error = context.Features.Get<IExceptionHandlerFeature>();

if (error != null) {

await context.Response.WriteAsync(error.Error.Message);

}

});

});

Bunu yaptiktan sonra dahi error message gelmez, neden cunku CORS (Cross origin) izin vermez. Bunun icin kendi extension methodumuzu yazmamiz lazim ve responsa yeni method ekleyebilmemiz lazim. (Application spesific error yazabilmek icin).

Bunun icin oncelikle API root altinda “Helpers” Folder yaratilir. Bunun altinda “Extensions” static class yaratilir.

Extensions.cs

public static class Extensions

{

public static void AddApplicationERror(this HttpResponse response, string message)

{

response.Headers.Add("Application-Error", message);

response.Headers.Add("Access-Control-Expose-Headers", "Application-Error");

response.Headers.Add("Access-Control-Allow-Origin", "\*");

}

}

### Angular’da Global error handling

Angular daki Interceptor’lar kullanilir. HTTP request leri Intercept edecegiz errorler icin. Service folderinin altinda yeni file eklenir “error.interceptor”

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

import { HttpInterceptor, HttpRequest, HttpHandler, HttpErrorResponse, HttpEvent, 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.errors;

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

};

Daha sonra app.module.ts providers a eklenir.

providers: [

AuthService,

ErrorInterceptorProvider

],

### Angular’da 3rd party libraryler ekleme (Alertify.js)

Sayfasindaki komut ile npm den install edilir (Angular-SPA projesinde iken)

npm install alertifyjs --save

Third party library i eklemek icin angular.json da scripts e ekleme yapilir (DIKKAT!!! Scripts 2 yerde var test olana degil build olana eklenecek) :

angular.json

"scripts": [

"node\_modules/alertifyjs/build/alertify.min.js"

],

style.css

@import '../node\_modules/alertifyjs/build/css/alertify.min.css';

@import '../node\_modules/alertifyjs/build/css/themes/bootstrap.min.css';

Bunu js library olarak kullanabiliriz ama demo sebebi ile service olarak kullanacagiz. Service folderinin icerisinde alertify yaratilir. Tslint error vermesin diye service icinde alertify declare edilir.

alertify.service.ts

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: any) {

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);

}

}

app.module de servis provideers a eklenir.

app.module.ts

providers: [

AuthService,

ErrorInterceptorProvider,

AlertifyService

],

Denemek icin servis nav.component e inject edilir.

nav.component.ts

constructor(private authService: AuthService, private alertify: AlertifyService) { }

login() {

this.authService.login(this.model).subscribe(next => {

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

}, error => {

this.alertify.error(error);

});

}

logout() {

localStorage.removeItem('token');

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

}

### JWT ile token handling

<https://github.com/auth0/angular2-jwt> (Helper library for handling JWTs in Angular 2+ apps)

DIKKAT!! AnglarJS icin cok benzer link var.

Github daki installation

npm install @auth0/angular-jwt

Su anki loggedin method (userin logged in olup olmadigini check icin) nav in icinde bunu servis olarak kullanmamiz lazim, bunu auth service icine alacagiz.

auth.service.ts

export class AuthService {

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

jwtHelper = new JwtHelperService();

Burada JwtHelperService QuickFix ile Import gelmiyor, bunu manual olarak ekleyecegiz.

import { JwtHelperService } from '@auth0/angular-jwt';

loggedIn() {

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

return !this.jwtHelper.isTokenExpired(token);

}

Nav.componentteki loggedIn degistirilir (AuthService zaten inject edilmis durumda)

nav.component.ts

loggedIn() {

return this.authService.loggedIn();

}

### JWT ile decode token (token dan username alma)

Usernam e i Database yerine LocalStorage daki token dan almak cok daha efektif. Bunun icin JWT serviceler I kullanilir.

auth.service.ts

export class AuthService {

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

jwtHelper = new JwtHelperService();

decodedToken: any;

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);

this.decodedToken = this.jwtHelper.decodeToken(user.token);

console.log(this.decodedToken);

}

})

);

}

Login olunca konsola donen :

1. exp: 1561464311
2. iat: 1561377911
3. nameid: "5"
4. nbf: 1561377911
5. unique\_name: "ted4"

nav.component.html de Welcome user yazan yerde user yerne

nav.component.html

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

Welcome {{authService.decodedToken.unique\_name | titlecase}}

</a>

Burada authService kirmizi underline li kalir cunku property private, bunu publice cevirince hata duzelir.

nav.component.ts

constructor(public authService: AuthService, private alertify: AlertifyService) { }

fakat burada da hata var cunku decoded token degerini sadece auth.service login methodunda belirliyoruz. Browser refresh olunca bu deger siliniyor. Oysaki token LocalStorega da degeri refresh te silinmiyor (ExpireDat e kadar)

auth.service.ts

this.decodedToken = this.jwtHelper.decodeToken(user.token);

Bu sebepten bunu auth service ten alip aplikasyonun rootuna alacagiz, boylece butun aplikasyonda bu tokena ulasabilecegiz. Bunu da en ust component olan app.component te yapacagiz. Aplikasyon load ettiginde authService de token varsa decode olacak.

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

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

import { JwtHelperService } from '@auth0/angular-jwt';

@Component({

selector: 'app-root',

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

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

})

export class AppComponent implements OnInit {

jwtHelper = new JwtHelperService();

constructor(private authService: AuthService) { }

ngOnInit() {

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

if (token) {

this.authService.decodedToken = this.jwtHelper.decodeToken(token);

}

}

}

### ngx Bootstrap (Jquery siz Bootstrap fonksiyonlari)

ngx boostrap Valor software <https://valor-software.com/ngx-bootstrap/#/>

npm install ngx-bootstrap --save

app.module e sitedeki usage kismindan dropdownlar eklenir (her ozellik tek tek eklenecek)

import { BsDropdownModule } from 'ngx-bootstrap';

@NgModule({

imports: [BsDropdownModule.forRoot(),...]

})

export class AppModule(){}

app.module.ts

import { BsDropdownModule } from 'ngx-bootstrap';

imports: [

BrowserModule,

HttpClientModule,

FormsModule,

BsDropdownModule.forRoot()

],

Navbar a example dan ekleme yapilir :

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

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

<div class="dropdown-menu" \*dropdownMenu>

Biraz css degisikligi yapmak icin, dropdown menu yu biraz asagiya itmek icin (margin-top-3):

<div class="dropdown-menu mt-3" \*dropdownMenu>

Dropdown (ve altindaki menulerde) uzerine gelince pointer i degistirmek icin

nav.component.css

.dropdown-toggle, .dropdown-item {

cursor: pointer;

}

Dropdown daki logout u aktif etme

<a class="dropdown-item" (click)="logout()"><i class="fa fa-sign-out"></i> Logout</a>

### Bootswatch (free) design kullanimi

Npm install [bootswatch@4.1.1](mailto:bootswatch@4.1.1) (butun theme leri yukler). Ana bootstrap ten sonra istedigimiz theme in css ini styles.css icerisine ekliyoruz.

styles.css

@import '../node\_modules/bootstrap/dist/css/bootstrap.min.css';

@import '../node\_modules/bootswatch/dist/united/bootstrap.min.css';

nav.component.html (bg-primary orange rengi getirir)

<nav class="navbar navbar-expand-md navbar-dark bg-primary">

### Angular routing

Kullanici islemleri icin 3 tane component yaratilir. lists, member-list, messages. Route yaratmak icin app folderi altinda routes.ts adli dosya yaratilir. Routelar array olarak tanimlanir. Her route bir object. Component routing icin path ve component kullanilir.

Routing de ordering onemli match etmeye en ustten baslar asagiya dogru gider.

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'}];

Routing I kullanabilmek icin appmodule daki import kismina eklenir.

imports: [

BrowserModule,

HttpClientModule,

FormsModule,

BsDropdownModule.forRoot(),

RouterModule.forRoot(appRoutes)

],

### Angular linklere routing atama

linklerde href yerine a-routerlink snippeti kullanilir. <li class="nav-item active"> buradaki active yerine routerLinkActive="router-link-active" gelir. (routerLinkActive :  lets you add a CSS class to an element when the link's route becomes active)

<a class="navbar-brand" [routerLink]="['/home']">Dating App</a>

<ul \*ngIf="loggedIn()" class="navbar-nav mr-auto">

<li class="nav-item" routerLinkActive="active">

<a class="nav-link" [routerLink]="['/members']">Matches</a>

</li>

<li class="nav-item" routerLinkActive="active">

<a class="nav-link" [routerLink]="['/lists']">Lists</a>

</li>

<li class="nav-item" routerLinkActive="active">

<a class="nav-link" [routerLink]="['/messages']" >Messages</a>

</li>

</ul>

Linklerin calismasi icin app.component.html de statik olarak home component gelmekte (app-home) bunun yerine router-outlet getirilir (Acts as a placeholder that Angular dynamically fills based on the current router state.).

app.component.html

<app-nav></app-nav>

<router-outlet></router-outlet>

### Angular event ten sonra routing (login sonrasi members a git gibi)

Nav.component.ts dosyasina angular router inject edilir :

nav.component.ts

export class NavComponent implements OnInit {

model: any = {};

constructor(public authService: AuthService, private alertify: AlertifyService, private router: Router) { }

redirection I subscribe in next methounun altinda bir satir acarak yapabiliriz. Burada ornek olmasi acisindan complete icerisinde yapacagiz.

nav.component.ts

login() {

this.authService.login(this.model).subscribe(next => {

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

}, error => {

this.alertify.error(error);

}, () => {

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

}

);

}

logout() {

localStorage.removeItem('token');

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

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

}

### Angular login olmadan bazi routelari gormeyi engelleme.

Bunun icin “route guard” kullanilir.src/app altinda \_guards folderi yaratilir.Bunu yaratmak icin Angular CLI I kullanacagiz.Angular\_Spa projesinde iken “cd src/app/\_guards” denilir. “ng g guard auth --spec=false” g = generate, --spec=false test file yaratmayacak.

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 alertify: AlertifyService) {}

canActivate(): boolean {

if (this.authService.loggedIn()) {

return true;

}

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

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

return false;

}

}

Authguard I app.module.ts de providers alrina eklemek gerekiyor :

app.module.ts

providers: [

AuthService,

ErrorInterceptorProvider,

AlertifyService,

AuthGuard

],

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';

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

export const appRoutes: Routes = [

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

{ path: 'members', component: MemberListComponent, canActivate: [AuthGuard]},

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

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

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

];

### Angular login olmadan bazi routelari gormeyi engelleme (multiple route)

route.ts te dummy route yaratilir;

route.ts

export const appRoutes: Routes = [

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

{

path: '',

runGuardsAndResolvers: 'always',

canActivate: [AuthGuard],

children: [

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

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

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

]

},

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

];

Bu asamada bir problem var nothing icin bor route umuz yok.Outlet is not activated consol hatasi veriyor (sifirdan siteye ayri browserda girmak istersek). Burada route lardaki “home” u silip ‘’ yapmamiz gerek. Boyle yapica dummy path: ‘’ ile ayni gibi olacak ama degil, cunku dummynin sonuna child routelar ekleniyor ‘’members gibi. Routelardan home u silecegiz ‘’ kalacak.

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'},

];

### ASP API’da User class propertyleri arttirma

User class property ler artttirilir. Photo class yaratilir.

user.cs

public class User

{

public int Id { get; set; }

public string UserName { get; set; }

public byte[] PasswordHash { get; set; }

public byte[] PasswordSalt { get; set; }

public string Gender { get; set; }

public DateTime DateOfBirth { get; set; }

public string KnownAs { get; set; }

public DateTime Created { get; set; }

public DateTime LastActive { get; set; }

public string Introduction { get; set; }

public string LookingFor { get; set; }

public string Interests { get; set; }

public string City { get; set; }

public string Country { get; set; }

public ICollection<Photo> Photos { get; set; }

}

photo.cs

public class Photo

{

public int Id { get; set; }

public string Url { get; set; }

public string Description { get; set; }

public DateTime DateAdded { get; set; }

public bool IsMain { get; set; }

}

Datacontext e photos eklenir :

DataContext.cs

public DbSet<Photo> Photos { get; set; }

migration eklenir.

dotnet ef migrations add ExtendedUserClass

Su an yaratilan migration da user ile photo arasinda cascade delete yok (EF default). Bunu cascade delete e cevirecegiz. User silinince photolar da silinecek.

### ASP API’da EF migration islemleri

Son yaptigimiz migration da cascade delete olmadigi icin bu migration i silecegiz.

dotnet ef migrations remove (database e aktarilmamis son migration i siler)

Eger migration i update database yaptiktan sonra silmek istersek, bir onceki migration a donmemiz gerekir (o migrationdan sonraki migrationlar daki down methodlari calistirabilmek icin). Bunun icin bir onceki migration (AddedUserEntity) ye don komutu kullanirsak, SQLite DropColumn u desteklemedigi icin bu basarili olmaz. Development ta iken Database I silip, son migration I silip update database diyebiliriz. Boylece DB deki veriler silinecek fakat DB tekrar normal olusmus olacak.

dotnet ef database drop

### ASP API’da EF relations

Related classlara Navigation Property ve Foreign Key eklenilmesi. Bu sekilde cascade delete yapacak.

photo.cs

public User User { get; set; }

public int UserId { get; set; }

### ASP API’da Seed data ile ornek veri yaratma

<https://www.json-generator.com/> Copy JSON to clipboard.

Data folder inin altinda UserSeedData.json alt folderi yaratilir. Ornek veriler kaydedilir.

### ASP API’da Seed data ile ornek veri yukleme

Data folder inin altinda Seed.cs yaratilir. Burada UserSeedData.json okunmak icin acilir. Bu Json datasini dongude kullanmak icin C# object haline getirmemiz lazim (JsonConvert.DeserializeObject). Burada Passwordleri hash ve salt da yaratmak icin AuthRepository.cs teki CreatePasswordHash methodunu copy paste ile aliyoruz. Bunu static yapmiyoruz, cunku seed sadece development ta kullanilacak AuthRepository i etkilemesin istiyoruz.

public class Seed

{

private readonly DataContext \_context;

public Seed(DataContext context)

{

\_context = context;

}

public void SeedUsers()

{

var userData = System.IO.File.ReadAllText("Data/UserSeedData.json");

var users = JsonConvert.DeserializeObject<List<User>>(userData);

foreach (var user in users)

{

byte[] passwordHash, passwordSalt;

CreatePasswordHash("password", out passwordHash, out passwordSalt);

user.PasswordHash = passwordHash;

user.PasswordSalt = passwordSalt;

user.UserName = user.UserName.ToLower();

\_context.Users.Add(user);

}

\_context.SaveChanges();

}

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));

}

} }

Sonrasinda seed class Configuration a servis olarak eklenir.

public void ConfigureServices(IServiceCollection services)

services.AddCors();

services.AddTransient<Seed>();

public void Configure(IApplicationBuilder app, IHostingEnvironment env, Seed seeder)

seeder.SeedUsers();

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

bunu calistirmadan once database indexlerini 0 dan baslatmak icin once database I drop yapip sonra tekrar database update yapacagiz.

dotnet ef database drop

dotnet ef database update

dotnet run diyince seed calisacak.

Seed bir daha calismasin diye commentliyoruz.

// seeder.SeedUsers();

### ASP API’da Dating Repository yaratma (User+Photo CRUD islemleri)

Data Folderi altinda Interface ve Actual implementation Class ayri yaratilacak. Interface Generic method kullanacak.

IDatingRepository.cs

public interface IDatingRepository

{

void Add<T>(T entity) where T: class;

void Delete<T>(T entity) where T:class;

Task<bool> SaveAll();

Task <IEnumerable<User>> GetUsers();

Task<User> GetUser(int id);

}

DatingRepository.cs

public class DatingRepository : IDatingRepository

{

private readonly DataContext \_context;

public DatingRepository(DataContext context)

{

\_context = context;

}

public void Add<T>(T entity) where T : class

{

\_context.Add(entity);

}

public void Delete<T>(T entity) where T : class

{

\_context.Remove(entity);

}

public async Task<User> GetUser(int id)

{

var user = await \_context.Users.Include(p=>p.Photos).FirstOrDefaultAsync(x=>x.Id==id);

return user;

}

public async Task<IEnumerable<User>> GetUsers()

{

var users = await \_context.Users.Include(p=>p.Photos).ToListAsync();

return users;

}

public async Task<bool> SaveAll()

{

return await \_context.SaveChangesAsync() > 0;

}

}

Aplikasyona servis olarak ekleme :

Startup.cs

services.AddScoped<IDatingRepository, DatingRepository>();

### ASP API’da Users Controller

UsersController.cs

[Authorize]

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

[ApiController]

public class UsersController : ControllerBase

{

private readonly IDatingRepository \_repo;

public UsersController(IDatingRepository repo)

{

\_repo = repo;

}

[HttpGet]

public async Task<IActionResult> GetUsers(){

var users = await \_repo.GetUsers();

return Ok(users);

}

[HttpGet("{id}")]

public async Task<IActionResult> GetUser(int id){

var user = await \_repo.GetUser(id);

return Ok(user); } }

Burada Photos u Include ettigimiz icin Api dogru sonuc vermior, bunu onlemek icin :

Startup.cs

services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version\_2\_2)

.AddJsonOptions(opt => {

opt.SerializerSettings.ReferenceLoopHandling = Newtonsoft.Json.ReferenceLoopHandling.Ignore; });

### ASP API’da Users DTO

API Controller dan donen users data da gereksiz veriler var, bunlari engellemek icin DTO kullanilacak. Id yi sakliyoruz. PasswordHash ve PasswordSalt u siliyoruz. DateofBirth u age yapiyoruz. Introduction. LookingFor, Interests siliyoruz. String PhotoURL ekliyoruz.

UserForListDto.cs

public class UserForListDto

{

public int Id { get; set; }

public string UserName { get; set; }

public string Gender { get; set; }

public int Age { get; set; }

public string KnownAs { get; set; }

public DateTime Created { get; set; }

public DateTime LastActive { get; set; }

public string City { get; set; }

public string Country { get; set; }

public string PhotoUrl { get; set; }

}

Bir DTO daha yapiyoruz, o da UserForDetailedDto

public class UserForDetailedDto

{

public int Id { get; set; }

public string UserName { get; set; }

public string Gender { get; set; }

public int Age { get; set; }

public string KnownAs { get; set; }

public DateTime Created { get; set; }

public DateTime LastActive { get; set; }

public string Introduction { get; set; }

public string LookingFor { get; set; }

public string Interests { get; set; }

public string City { get; set; }

public string Country { get; set; }

public string PhotoUrl { get; set; }

public ICollection<Photo> Photos { get; set; }

}

### ASP API AutoMapper, User i DTO lara mapping

Ctrl+Shift+P => nuget package manager add package => automapper.extensions.microsoft.depedency.injection . Bu paketi de startup class a inject ediyoruz.

services.AddCors();

services.AddAutoMapper();

Bundan sonra AutoMapper I Controller a Inject edebiliriz.

UsersController.cs

[Authorize]

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

[ApiController]

public class UsersController : ControllerBase

{

private readonly IDatingRepository \_repo;

private readonly IMapper \_mapper;

public UsersController(IDatingRepository repo, IMapper mapper)

{

\_mapper = mapper;

\_repo = repo;

}

[HttpGet]

public async Task<IActionResult> GetUsers()

{

var users = await \_repo.GetUsers();

var usersToReturn = Mapper.Map<IEnumerable<UserForDetailedDto>>(users);

return Ok(usersToReturn);

}

[HttpGet("{id}")]

public async Task<IActionResult> GetUser(int id)

{

var user = await \_repo.GetUser(id);

var userToReturn = Mapper.Map<UserForDetailedDto>(user);

return Ok(userToReturn);

}

}

Bundan baska Automapper a mappingleri kaydetmemiz lazim. Bunun icin Herlpers folder i altinda AutoMapperProfiles.cs yaratiyoruz.

AutoMapperProfiles.cs

public class AutoMapperProfiles : Profile

{

public AutoMapperProfiles()

{

CreateMap<User, UserForListDto>();

CreateMap<User, UserForDetailedDto>();

}

}

Burada 2 sorun var :

1. Su an UserForListDto.cs de Age property var, User da ise Date Of Birth var, su an bu ikisi birbirine match etmedigi icin age:0 geliyor.
2. User listesi gelirken Photo havigation property oldugu icin bunun icerisinde ait oldugu user in bilgisi de Pohoto nun propertrysi olarak geliyor, burada user password has ve salt da var buna engel olmamiz lazim.

### ASP API AutoMapper fazla gelenleri (navigation)cikartma, eksikleri ekleme

Photos class ta User navigation property olarak var oldugu icin User bilgileri de geliyor. Bunu engellemek icin Dtos folder i altinda PhotosForDetailedDto.cs yaratiyoruz.

PhotosForDetailDto.cs

public class PhotosForDetailDto

{

public int Id { get; set; }

public string Url { get; set; }

public string Description { get; set; }

public DateTime DateAdded { get; set; }

public bool IsMain { get; set; }

}

UserForDetailedDto daki Icollection<Photo> yu degistiriyoruz.

UserForDetailedDto.cs

public ICollection<PhotosForDetailDto> Photos { get; set; }

Bu degisikligi AutoMapper a da bildirmemiz gerekiyor:

CreateMap<Photo, PhotosForDetailDto>();

User daki DateOfBirth ile UserDto lardaki age I de birbnilerine map etmemiz lazim.

AutoMapperProfiles.cs

public AutoMapperProfiles()

{

CreateMap<User, UserForListDto>()

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

{

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

});

CreateMap<User, UserForDetailedDto>()

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

{

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

});

CreateMap<Photo, PhotosForDetailDto>();

}

Date of Birth i age e cevirmek icin DateTime class a extension method yazmamiz gerekli cunku DateTime methodlari icerisinde age I veren yok. Bunun icin Helpers/Extensions folder ina gidiyoruz :

Extensions.cs

public static int CalculateAge(this DateTime theDateTime)

{

var today = DateTime.Today;

// Calculate the age.

var age = today.Year - theDateTime.Year;

// Go back to the year the person was born in case of a leap year

if (theDateTime.Date > today.AddYears(-age)) age--;

return age;

}

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, PhotosForDetailDto>();

}

### Angular Typescript Type lari kullanma

Hem compile time errorleri bulmada yardimci olr, hem de intellisense kullanma imkani verir. Tamamen optional. Projedeki type leri src/app altinda \_models folderi yaratip bunun altinda yaratacagiz. Bu foldes I sag tiklayip GenerateInterface diyoruz.

(Bunun icin **Swagger** ve **NswagStudio** tool kullanilabilir **veya** With [Web Essentials](http://vswebessentials.com/) installed in VS 2015, you can right-click the class file, then > Web Essentials > Create Typescript Intellisense File from the context menu. **Veya** If you use vscode you can use my extension **csharp2ts** which does exactly that. You just select the pasted C# code and run the Convert C# to TypeScript command from the command palette)

user.ts

export interface User {

id: number;

username: string;

knownAs: string;

age: number;

gender: string;

created: Date;

lastActive: Date;

photoUrl: string;

city: string;

country: string;

interests?: string;

introduction?: string;

lookingFor?: string;

photos?: Photo[];

}

Burada Photos un kendisi de Type oldugu icin (Photo), Photo interface i de yaratiyoruz.

export interface Photo {

id: number;

url: string;

description: string;

dateAdded: Date;

isMain: boolean;

}

sonrasinda

photo.ts

import { Photo } from './photo';

### Angular User icin servis yaratma (environment.ts te API url set ederek)

\_services sag tusla Generate Service denilir user yaratilir. Bu serviste API I query edecegiz, bu yuzden Api in base url ini property olarak atayacagiz. Burada Angular da global variable (configuration gibi) yaratmak icin environments folderindaki environment.ts yi kullanacagiz. Bunu boyle yapiyoruz cunku producton da api adresi degisecek, bunu tek tek girip servislerde degistirmemis olacagiz.

environment.ts

export const environment = {

production: false,

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

};

auth.service.ts te de bu degisikligi yapiyoruz.

export class AuthService { baseUrl = environment.apiUrl + 'auth/';

user.service.ts

export class UserService {

baseUrl = environment.apiUrl;

constructor(private http: HttpClient) { }

getUsers(): Observable<User[]> {

return this.http.get<User[]>(this.baseUrl + 'users');

}

getUser(id): Observable<User> {

return this.http.get<User>(this.baseUrl + 'users/' + id);

}}

Burada get<User[]> yapiyoruz cunku http.get object bekliyor, tipini belirtiyoruz.

Ayrica burada su an authorization yok. API a heder da berarer token da gondermemiz gerekiyor. Get method un optionlarinda header var.

const httpOptions = {

headers: new HttpHeaders({

'Authorization': 'Bearer ' + localStorage.getItem('token')

})

};

getUsers(): Observable<User[]> {

return this.http.get<User[]>(this.baseUrl + 'users', httpOptions);

}

getUser(id): Observable<User> {

return this.http.get<User>(this.baseUrl + 'users/' + id, httpOptions);

}

Bu sekilde userservice tamamlanmis oluoyr, bunu app.module ts e servis olarak register ediyoruz.

app.module.ts

providers: [

AuthService,

ErrorInterceptorProvider,

AlertifyService,

AuthGuard,

UserService

],

### Angular User servisi Componentlerde kullanma (token hatali)

MemberListComponent da User service kullanimi:

MemberListComponent.ts

export class MemberListComponent implements OnInit {

users: User[];

constructor(private userService: UserService, private alertify: AlertifyService) { }

ngOnInit() {

this.loadUsers();

}

loadUsers(){

this.userService.getUsers().subscribe((users: User[]) => {

this.users = users;

}, error => {

this.alertify.error(error);

});

}}

MemberListComponent.html

<div class="container">

<div class="row">

<div class="col-lg-2 col-md-3 col-sm-6">

<p \*ngFor="let user of users">{{user.knownAs}}</p>

</div>

</div>

</div>

Su an ilk giriste token bos oldugu icin hata verecek, bunu sonra duzeltecegiz.

### Angular User lari card olarak listeleme (Member ust folder yaratma)

Member ile ilgili ilave componentler yaratilacak. Bunun icin members folderi yaratilir, diger ilgili componentler bu folderin icinde yaratilacak. Mevcut olan member-list componenti bu foldera suruklenir. Burada otomatik olarak referanslar guncellenmezse, bunlari manuel yapmak gerekir.

Members folderi altinda yeni component yaratilir “member-card”. Direk app icinde component yaratmadigimiz icin bu otomatik app.module e eklenmez, bunu manuel ekleyecegiz.

app.module.ts

declarations: [

MemberCardComponent ],

Member list component tan user i member-card component a aktrarmak istiyoruz.

membercardcomponent.js

export class MemberCardComponent implements OnInit {

@Input() user: User;

membercardcomponent.html

<div class="card mb-4">

<div class="card-img-wrapper">

<img class="card-img-top" src="{{user.photoUrl}}" alt="{{user.knownAs}}">

</div>

<div class="card-body p1">

<h6 class="card-title text-center mb-1"><i class="fa fa-user"></i>

{{user.knownAs}}, {{user.age}}</h6>

<p class="card-text text-muted text-center">{{user.city}}</p>

</div>

</div>

member-list.component.html

<div class="container mt-5">

<div class="row">

<div \*ngFor="let user of users" class="col-lg-2 col-md-3 col-sm-6">

<app-member-card [user]="user"></app-member-card>

</div>

</div>

</div>

member-list component asil goruntuleme yaptigimiz component. Burada aslinda userlari listeliyoruz, ngFor dongusundeki userlari member-card component e gonderiyoruz o da user lari card olarak gosteriyor.

Yani listelemeyi yapan member-list, listelemenin nasil gorunecegi member-card, member-card bir nevi partial-view gorevi goruyor.

### Angular User card lara CSS verme

Card da ki image hover edince image zoom in. bunu member-card.component.css da yapacagiz.

member-card.component.css

.card:hover img {

transform: scale(1.2, 1.2);

transition-duration: 500ms;

transition-timing-function: ease-out;

opacity: 0.7;

}

.card img {

transform: scale(1.0, 1.0);

transition-duration: 500ms;

transition-timing-function: ease-out;

}

.card-img-wrapper {

overflow: hidden;}

### Angular User card lara animated button ekleme

Img tag dan sonra ul button list eklenir. Su an butonlar imag nin altinda, css ile img in ustune cikartacagiz ve img zoom oldugunda dissapear olacaklar. Bunu yine css te yapiyoruz.

member-card.component.css

.card-img-wrapper {

overflow: hidden;

position: relative;

}

.member-icons {

position: absolute;

bottom: -30%;

left: 0;

right: 0;

margin-right: auto;

margin-left: auto;

opacity: 0;

}

.card-img-wrapper:hover .member-icons {

bottom: 0;

opacity: 1;

}

.animate {

transition: all 0.3s ease-in-out;

}

### Angular User servisi Componentlerde kullanma (token hata duzeltme)

app.module.ts de bir takim ilaveler yapmamiz gerekiyor :

app.module.ts

export function tokenGetter() {

return localStorage.getItem('token');

}

@NgModule({

imports: [

………………………

RouterModule.forRoot(appRoutes),

JwtModule.forRoot({

config: {

tokenGetter: tokenGetter,

whitelistedDomains: ['localhost:5000'],

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

}

})

],

Daha sonra user.service.ts den httpOptions kaldirilir.

user.service.ts

@Injectable({

providedIn: 'root'

})

export class UserService {

baseUrl = environment.apiUrl;

constructor(private http: HttpClient) { }

getUsers(): Observable<User[]> {

return this.http.get<User[]>(this.baseUrl + 'users', );

}

getUser(id): Observable<User> {

return this.http.get<User>(this.baseUrl + 'users/' + id);

}}

### Angular member-detail e link

App/members folderinin altinda member-detail componenti yaratilir. app.module.ts e manule olarak ekliyoruz.

app.module.ts

declarations: [

…………………………

MemberCardComponent,

MemberDetailComponent

],

member-detail.component.ts

export class MemberDetailComponent implements OnInit {

user: User;

constructor(private userService: UserService, private alertify: AlertifyService,

private route: ActivatedRoute) {}

ngOnInit() {

this.loadUser();

}

// members/4

loadUser() {

this.userService.getUser(+this.route.snapshot.params['id']).subscribe((user: User) => {

this.user = user;

}, error => {

this.alertify.error(error);

});

}}

member-detail.component.html

<p>

{{user.knownAs}}

</p>

route.ts

children: [

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

{ path: 'members/:id', component: MemberDetailComponent},

member-card.component.html

<ul class="list-inline member-icons animate text-center">

<li class="list-inline-item"><button class="btn btn-primary" [routerLink]="['/members/', user.id]"><i class="fa fa-user"></i></button></li>

Su an bu method calisiyor olsa da console da knownas propertyi bilmiyorum error geliyor. Buna engel olmak icin safenavigation kullanacagiz.

member-detail.component.html

<p>

{{user?.knownAs}}

</p>

### Angular member-detail sayfa tasarimi (sol bolum)

member-detail.component.html

<div class="container mt-4">

<div class="row">

<h1>{{user.knownAs}}'s Profile</h1>

</div>

<div class="row">

<div class="col-sm-4">

<div class="card">

<img class="card-img-top img-thumbnail" src="{{user?.photoUrl}}" alt="{{user?.knownAs}}">

<div class="card-body">

<div>

<strong>Location:</strong>

<p>{{user?.city}}, {{user?.country}}</p>

</div>

<div>

<strong>Age:</strong>

<p>{{user?.age}}</p>

</div>

<div>

<strong>Last Active:</strong>

<p>{{user?.lastActive}}</p>

</div>

<div>

<strong>Member since:</strong>

<p>{{user?.created}}</p>

</div>

</div>

<div class="card-footer">

<div class="btn-group d-flex">

<button class="btn btn-primary w-100">Like</button>

<button class="btn btn-success w-100">Message</button>

</div>

</div>

</div>

</div>

<div class="col-sm-8">

</div>

</div>

</div>

member-detail.component.css

.img-thumbnail {

margin: 25px;

width: 85%;

height: 85%;

}

.card-body {

padding: 0 25px;

}

.card-footer {

padding: 10px 15px;

background-color: #fff;

border-top: none;

}

### Angular member-detail sayfa tasarimi (sag bolum)

Burada ngx-bootstrap deki tab’ ler kullanilacak.

app.module.ts

imports: [

BsDropdownModule.forRoot(),

TabsModule.forRoot(),

member-detail.component.html (Ikinci bolum – sag taraf)

…………………………..

<div class="col-sm-8">

<div class="tab-panel">

<tabset class="member-tabset">>

<tab heading="About {{user?.knownAs}}">

<h4>Description</h4>

<p>{{user?.introduction}}</p>

<h4>Looking for</h4>

<p>{{user?.lookingFor}}</p>

</tab>

<tab heading="Interests">

<h4>Interests</h4>

<p>{{user?.interests}}</p>

</tab>

<tab heading="Photos">

<p>Photos will go here</p>

</tab>

<tab heading="Messages">

<p>Messages will go here</p>

</tab>

</tabset>

</div>

</div>

</div>

</div>

Css icin tabset e class veriyoruz :

<tabset class="member-tabset">

Buna css i global css import edip oradan verecegiz.

styles.css

.tab-panel {

border: 1px solid #ddd;

padding: 10px;

border-radius: 4px;

}

.nav-tabs > li.open, .member-tabset > .nav-tabs > li:hover {

border-bottom: 4px solid #fbcdcf;

}

.member-tabset > .nav-tabs > li.open > a, .member-tabset > .nav-tabs > li:hover > a {

border: 0;

background: none !important;

color: #333333;

}

.member-tabset > .nav-tabs > li.open > a > i, .member-tabset > .nav-tabs > li:hover > a > i {

color: #a6a6a6;

}

.member-tabset > .nav-tabs > li.open .dropdown-menu, .member-tabset > .nav-tabs > li:hover .dropdown-menu {

margin-top: 0px;

}

.member-tabset > .nav-tabs > li.active {

border-bottom: 4px solid #E95420;

position: relative;

}

.member-tabset > .nav-tabs > li.active > a {

border: 0 !important;

color: #333333;

}

.member-tabset > .nav-tabs > li.active > a > i {

color: #404040;

}

.member-tabset > .tab-content {

margin-top: -3px;

background-color: #fff;

border: 0;

border-top: 1px solid #eee;

padding: 15px 0;

}

### Angular Root resolver ile component e veri alma (Tamamen opsiyonel user? da olur)

Resolver in amaci component yuklenmeden veriyi getirmek.

Su an member-list te api dan datayi ngonit loadusers ile aliyoruz.

Resolvers icin app folderi altinda \_resolvers adli yeni folder yaratiyoruz. Generate Resolver olmadigi icin sifirdan kendimiz yaratacagiz. Generate file diyip adini member-detail.resolver.ts veriyoruz

member-detail.resolver.ts

@Injectable()

export class MemberDetailResolver implements Resolve<User> {

constructor(private userService: UserService, private router: Router,

private alertify: AlertifyService) {}

resolve(route: ActivatedRouteSnapshot): Observable<User> {

return this.userService.getUser(route.params['id']).pipe(

catchError(error => {

this.alertify.error('Problem retrieveing data');

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

return of(null);

})

);

}

}

app.module.ts providers a eklenir.

app.module.ts

providers: [

…………………

UserService,

MemberDetailResolver

],

Bunu ayrica route.ts e de eklememiz lazim.

route.ts

export const appRoutes: Routes = [

{ path: '', component: HomeComponent},

{

path: '',

runGuardsAndResolvers: 'always',

canActivate: [AuthGuard],

children: [

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

{ path: 'members/:id', component: MemberDetailComponent, resolve: {user: MemberDetailResolver}},

member-detail.component.ts de ngoninit this.loadUser(); I siliyoruz.

member-detail.component.ts

ngOnInit() {

this.route.data.subscribe(data => {

this.user = data['user'];

});

}

loaduser methodu siliyoruz. member-datail componentteki user? Daki ? leri artik silebiliriz.

Bunu ayrica member-list page icin de yapiyoruz. member-detail.resolver.ts file i ayni folder icerisinde copy/paste yapiyoruz. Adini member-list-resolver yapiyoruz.

member-list-resolver

@Injectable()

export class MemberListResolver implements Resolve<User[]> {

constructor(private userService: UserService, private router: Router,

private alertify: AlertifyService) {}

resolve(route: ActivatedRouteSnapshot): Observable<User[]> {

return this.userService.getUsers().pipe(

catchError(error => {

this.alertify.error('Problem retrieveing data');

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

return of(null);

})

);

}}

routes.ts

children: [

{ path: 'members', component: MemberListComponent, resolve: {users: MemberListResolver}},

{ path: 'members/:id', component: MemberDetailComponent, resolve: {user: MemberDetailResolver}},

app.module.ts

providers: [

AuthService,

ErrorInterceptorProvider,

AlertifyService,

AuthGuard,

UserService,

MemberDetailResolver,

MemberListResolver

],

member-list.component.ts

export class MemberListComponent implements OnInit {

users: User[];

constructor(private userService: UserService, private alertify: AlertifyService,

private route: ActivatedRoute) { }

ngOnInit() {

this.route.data.subscribe(data => {

this.users = data['users'];

});

}

// loadUsers(){

// this.userService.getUsers().subscribe((users: User[]) => {

// this.users = users;

// }, error => {

// this.alertify.error(error);

// });

// }

### Angular User photo gallery ekleme

Bunun icin bir angular component tan istifade edecegiz ngx gallery. <https://www.npmjs.com/package/ngx-gallery>

npm install ngx-gallery –save

app.module.ts

Usage

*// app.module.ts*

**import** { NgxGalleryModule } **from** 'ngx-gallery';

**...**

@NgModule({

    imports: [

**...**

        NgxGalleryModule

**...**

    ],

**...**

})

**export** class AppModule { }

member-detail.component.ts

export class MemberDetailComponent implements OnInit {

user: User;

galleryOptions: NgxGalleryOptions[];

galleryImages: NgxGalleryImage[];

constructor(private userService: UserService, private alertify: AlertifyService,

private route: ActivatedRoute) { }

ngOnInit() {

this.route.data.subscribe(data => {

this.user = data['user'];

});

this.galleryOptions = [

{

width: '500px',

height: '500px',

imagePercent: 100,

thumbnailsColumns: 4,

imageAnimation: NgxGalleryAnimation.Slide;

preview: false

}

];

this.galleryImages = this.getImages();

}

getImages(){

const imageUrls = [];

for (let i = 0; i < this.user.photos.length; i++) {

imageUrls.push({

small: this.user.photos[i].url,

medium: this.user.photos[i].url,

big: this.user.photos[i].url,

description: this.user.photos[i].description

});

}

return imageUrls;

}

}

member-detail.component.html

<tab heading="Photos">

<ngx-gallery [options]="galleryOptions" [images]="galleryImages"></ngx-gallery>

</tab>

### Angular Member Edit Component routing

Members folderi sag tiklanir Generate Component member-edit. Sonra app.module a eklenir.

app.module.ts

@NgModule({

declarations: [

…..

MemberEditComponent

],

path e id yi yazmiyoruz bunu decoded token dan alacagiz.

routes.ts

children: [

{ path: 'member/edit', component: MemberEditComponent},

nav.component.html

Edit Profile linki degistirilir

<div class="dropdown-menu mt-3" \*dropdownMenu>

<a class="dropdown-item" [routerLink]="['/member/edit']"><i class="fa fa-user"></i>Edit Profile</a>

Edit member a veri getirmek icin yine resolver i kullanacagiz. member-detail.resolver.ts i copy paste yapiyoruz. member-edit.resolver.ts olarak rename ediyoruz private auth: AuthService ekliyoruz

@Injectable()

export class MemberEditResolver implements Resolve<User> {

constructor(private userService: UserService, private router: Router, private authService: AuthService,

private alertify: AlertifyService) {}

resolve(route: ActivatedRouteSnapshot): Observable<User> {

return this.userService.getUser(this.authService.decodedToken.nameid).pipe(

catchError(error => {

this.alertify.error('Problem retrieveing your data');

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

return of(null);

})

);

}

}

app.module component a ilave ediyoruz.

app.module.ts

providers: [

….

MemberEditResolver

],

routes.ts

{ path: 'member/edit', component: MemberEditComponent, resolve: {user: MemberEditResolver} },

member-edit.component.ts

export class MemberEditComponent implements OnInit {

user: User;

constructor(private route: ActivatedRoute) { }

ngOnInit() {

this.route.data.subscribe(data => {

this.user = data['user'];

});

}

}

### Angular Member Edit Component HTML Part1

member.detail copy paste edilir.

<div class="container mt-4">

<div class="row">

<div class="col-sm-4">

<h1>Your Profile</h1>

</div>

<div class="col-sm-8">

<div class="alert alert-info">

<strong>Information:</strong> You have made changes. Any unsaved changes will be lost!

</div>

</div>

</div>

<div class="row">

<div class="col-sm-4">

<div class="card">

<img class="card-img-top img-thumbnail" src="{{user.photoUrl}}" alt="{{user.knownAs}}">

<div class="card-body">

<div>

<strong>Location:</strong>

<p>{{user.city}}, {{user.country}}</p>

</div>

<div>

<strong>Age:</strong>

<p>{{user.age}}</p>

</div>

<div>

<strong>Last Active:</strong>

<p>{{user.lastActive}}</p>

</div>

<div>

<strong>Member since:</strong>

<p>{{user.created}}</p>

</div>

</div>

<div class="card-footer">

<button class="btn btn-success btn-block">Save Changes</button>

</div>

</div>

</div>

<div class="col-sm-8">

<div class="tab-panel">

<tabset class="member-tabset">

<tab heading="Edit Profile">

<form>

<h4>Description</h4>

<textarea name="introduction" rows="6" class="form-control"

[(ngModel)]="user.introduction"></textarea>

<h4>Looking for</h4>

<textarea name="lookingFor" rows="6" class="form-control"

[(ngModel)]="user.lookingFor"></textarea>

<h4>Interests</h4>

<textarea name="interests" rows="6" class="form-control"

[(ngModel)]="user.interests"></textarea>

<h4>Location Details:</h4>

<div class="form-inline">

<label for="city">City</label>

<input class="form-control" type="text" name="city" [(ngModel)]="user.city">

<label for="country">Country</label>

<input class="form-control" type="text" name="country" [(ngModel)]="user.country">

</div>

</form>

</tab>

<tab heading="Edit Photos">

<p>Photo edit will go here</p>

</tab>

</tabset>

</div>

</div>

</div>

</div>

### Angular Member Edit Component HTML Part2 (Degisiklik yapinca alert)

Degisiklik yapinca alert cikartmak icin form state i kullanacagiz.

<div \*ngIf="editForm.dirty" class="alert alert-info">

<form #editForm="ngForm" (ngSubmit)="updateUser()">

Save changes butonu icin de benzer sekilde yapilacak.

<div class="card-footer">

<button [disabled]="!editForm.dirty" class="btn btn-success btn-block">Save Changes</button> </div>

Alert teki updateUser methodu yaratilir.

updateUser() {

console.log(this.user);

this.alertify.success('Profile updated successfully');

}

Bu sekilde yapinca bu dugme formun disinda oldugu icin calismayacak.

<form #editForm="ngForm" (ngSubmit)="updateUser()">

Bunu duzeltmek icin form a id verip butonun form attributuna formun adini veriyoruz.

<form #editForm="ngForm" id="editForm" (ngSubmit)="updateUser()">

<button [disabled]="!editForm.dirty" form="editForm" class="btn btn-success btn-block">Save Changes</button>

SaveChanges methodu tamamlandiktan sonra form.dity nin reset olmasi gerekiyor. Bunu icin HTML deki formu resetleyecegiz. Bu forma ulasabilmek icin component te @ViewChild decorator u kullaniyoruz.

@ViewChild('editForm') editForm: NgForm;

updateUser() {

this.editForm.reset(); }

Bu sekilde yaparsak calisir fakat formun icindeki bilgiler de silinir. Buna engel olmak icin reset methodunun propertylerini kullaniriz.

this.editForm.reset(this.user);

### Angular Member Edit ten baska route a giderken form dirty ise uyari verme

Bunun icin route guard kullanilir. Guard folderi altinda new file denilir.

prevent-unsaved-changes-guard.ts

@Injectable()

export class PreventUnsavedChanges implements CanDeactivate<MemberEditComponent> {

canDeactivate(component: MemberEditComponent) {

if (component.editForm.dirty) {

return confirm('Are you sure you want to continue? Any unsaved changes will be lost!');

}

return true;

}

}

Bunu app.module e ekliyoruz.

app.module.ts

providers: [

PreventUnsavedChanges

],

route.ts

{ path: 'member/edit', component: MemberEditComponent,

resolve: {user: MemberEditResolver}, canDeactivate: [PreventUnsavedChanges]},

Kullanici baska bir linke tiklarsa uyari mesaji gelir. Fakat browser i kapar ise buna engel olamaz. Bunun icin ayri bir yontem uygulanir (HostListener).

member-edit.component.ts

export class MemberEditComponent implements OnInit {

@ViewChild('editForm') editForm: NgForm;

user: User;

@HostListener('window:beforeunload', ['$event'])

unloadNotification($event: any) {

if (this.editForm.dirty) {

$event.returnValue = true;

}

}

### ASP API Member Edit degisiklikleri API a yazma

Bunun icin degisiklikleri API a gonderecek Dto ASP de yaratilir.

UserForUpdateDto.cs

public class UserForUpdateDto

{

public string Introduction { get; set; }

public string LookingFor { get; set; }

public string Interests { get; set; }

public string City { get; set; }

public string Country { get; set; }

}

Helpers/AutoMapperProfiles.cs

autoMapperProfiles.cs

public AutoMapperProfiles()

{CreateMap<UserForUpdateDto, User>();

UsersController.cs

Burada update edilecek profilini degistirmek isteyen user id si ile (path taki id) token daki user id si birbirini tutuyor mu kontrol etmemiz gerekli.

if (id != int.Parse(User.FindFirst(ClaimTypes.NameIdentifier).Value))

return Unauthorized();

[HttpPut("{id}")]

public async Task<IActionResult> UpdateUser(int id, UserForUpdateDto userForUpdateDto)

{

if (id != int.Parse(User.FindFirst(ClaimTypes.NameIdentifier).Value))

return Unauthorized();

var userFromRepo = await \_repo.GetUser(id);

\_mapper.Map(userForUpdateDto, userFromRepo);

if (await \_repo.SaveAll())

return NoContent();

throw new Exception($"Updating user {id} failed on save");

}

API i postman de denerken method u Put seciyoruz ve Body de row/JSON olarak asagidaki veriyi gonderiyoruz :

PUT : <http://localhost:5000/api/users/1>

{

"introduction": "updated intro",

"lookingFor": "updated lookingFor",

"interests": "updated interests",

"city": "updated city",

"country": "updated country"

}

### Angular Member Edit degisiklikleri API a gonderme

app/\_services/user.service

user.service.ts

updateUser(id: number, user: User) {

return this.http.put(this.baseUrl + 'users/' + id, user);

}

member-edit.component.ts

constructor(private route: ActivatedRoute, private alertify: AlertifyService,

private userService: UserService, private authService: AuthService) { }

updateUser() {

this.userService.updateUser(this.authService.decodedToken.nameid, this.user).subscribe(next => {

this.alertify.success('Profile updated successfully');

this.editForm.reset(this.user);

}, error => {

this.alertify.error(error);

});

}

### ASP API Photo update Cloudinary settings

Photo storage olarak Cloudinary kullanilacak. Sign Up For free diyerek account yaratiyoruz.

appsettings.json

"AllowedHosts": "\*",

"CloudinarySettings": {

"CloudName": "dl5tj9d2b",

"ApiKey": "959836862532493",

"ApiSecret": "Xg-PtUn1\_d2oSae5oiVdYnBYfFk"

}

Helpers Folder i altinda CloudinarySettings.cs yaratilir.

CloudinarySettings.cs

public class CloudinarySettings

{

public string CloudName { get; set; }

public string ApiKey { get; set; }

public string ApiSecret { get; set; }

}

Startup.cs

services.AddCors(); services.Configure<CloudinarySettings>(Configuration.GetSection("CloudinarySettings"));

Photo.cs

public string PublicId { get; set; }

Photo.cs model degistigi icin yeni migration ekleyecegiz.

dotnet ef migrations add AddedPublicIdToPhoto

dotnet ef database update

Nuget packageManager-> add package -> CloudinaryDotNet

### ASP API Photos Controller 1

(<https://www.udemy.com/build-an-app-with-aspnet-core-and-angular-from-scratch/learn/lecture/8714896#questions/4836538>) For DB storage Photos

Controllers Folderi altinda PhotosController yaratilir. [HttpPost] a ayrica route vermiyoruz Controller route [Route("api/users/{userId}/photos")] icerisinde HttpPost bir tek bu metod var. PhotoForCreationDto yaratilir.

PhotoForCreationDto.cs

public class PhotoForCreationDto

{

public PhotoForCreationDto()

{

DateAdded = DateTime.Now;

}

public string Url { get; set; }

public IFormFile File { get; set; }

public string Description { get; set; }

public DateTime DateAdded { get; set; }

public string PublicID { get; set; }

}

PhotosController.cs

namespace DatingApp.API.Controllers

{

[Authorize]

[Route("api/users/{userId}/photos")]

[ApiController]

public class PhotosController : ControllerBase

{

private readonly IDatingRepository \_repo;

private readonly IMapper \_mapper;

private readonly IOptions<CloudinarySettings> \_cloudinaryConfig;

private Cloudinary \_cloudinary;

public PhotosController(IDatingRepository repo, IMapper mapper,

IOptions<CloudinarySettings> cloudinaryConfig)

{

\_repo = repo;

\_mapper = mapper;

\_cloudinaryConfig = cloudinaryConfig;

Account acc = new Account(

\_cloudinaryConfig.Value.CloudName,

\_cloudinaryConfig.Value.ApiKey,

\_cloudinaryConfig.Value.ApiSecret

);

\_cloudinary = new Cloudinary(acc);

}

[HttpPost]

public async Task<IActionResult> AddPhotoForUser(int userId,

PhotoForCreationDto photoForCreationDto)

{

if (userId != int.Parse(User.FindFirst(ClaimTypes.NameIdentifier).Value))

return Unauthorized();

var userFromRepo = await \_repo.GetUser(userId);

var file = photoForCreationDto.File;

var uploadResult = new ImageUploadResult();

if (file.Length > 0)

{

using (var stream = file.OpenReadStream())

{

var uploadParams = new ImageUploadParams()

{

File = new FileDescription(file.Name, stream),

Transformation = new Transformation()

.Width(500).Height(500).Crop("fill").Gravity("face")

};

uploadResult = \_cloudinary.Upload(uploadParams);

}

}

photoForCreationDto.Url = uploadResult.Uri.ToString();

photoForCreationDto.PublicID = uploadResult.PublicId;

var photo = \_mapper.Map<Photo>(photoForCreationDto);

if (!userFromRepo.Photos.Any(u => u.IsMain))

photo.IsMain = true;

userFromRepo.Photos.Add(photo);

if (await \_repo.SaveAll())

{

return Ok();

}

return BadRequest("Could not a dd the photo");

}

}

}

### ASP API Photos Controller 2

Asagida return Ok dondurmemiz dogru degil bunun degistirilmesi gerekiyor.

if (await \_repo.SaveAll())

{

return Ok();

}

IdatingRepository e GetPhoto eklenir:

Task<Photo> GetPhoto(int id);

DatingRepository.cs

public async Task<Photo> GetPhoto(int id)

{

var photo = await \_context.Photos.FirstOrDefaultAsync(p => p.Id == id);

return photo;

}

PhotoForReturnDto.cs

public class PhotoForReturnDto

{

public int Id { get; set; }

public string Url { get; set; }

public string Description { get; set; }

public DateTime DateAdded { get; set; }

public bool IsMain { get; set; }

public string PublicId { get; set; }

}

AutoMapperProfiles.cs

CreateMap<Photo, PhotoForReturnDto>();

CreateMap<PhotoForCreationDto, Photo>();

PhotosController.cs

[HttpGet("{id}", Name = "GetPhoto")]

public async Task<IActionResult> GetPhoto(int id)

{

var photoFromRepo = await \_repo.GetPhoto(id);

var photo = \_mapper.Map<PhotoForReturnDto>(photoFromRepo);

return Ok(photo);

}

if (await \_repo.SaveAll())

{

var photoToReturn = \_mapper.Map<PhotoForReturnDto>(photo);

return CreatedAtRoute("GetPhoto", new { id = photo.Id }, photoToReturn);

}

### ASP API Photos Controller Duzeltme

Controller methodun icerisine [FromForm] eklememiz gerekiyor.

public async Task<IActionResult> AddPhotoForUser(int userId,

[FromForm]PhotoForCreationDto photoForCreationDto)

### Angular Photo Gallery Tasarim

Members folderi altinda photo-editor generate component denilir. app.module.ts e eklenir. PhotoEditorComponent e photolar ust component ten getirilecek (@Input)

photoEditorComponent.cs

export class PhotoEditorComponent implements OnInit {

@Input() photos: Photo[]

constructor() { }

ngOnInit() { }}

photo-editor.component.html

<div class="row">

<div class="col-sm-2" \*ngFor="let photo of photos">

<img src="{{photo.url}}" class="img-thumbnail p-1" alt="">

<div class="text-center"></div>

<button type="button" class="btn btn-sm">Main</button>

<button type="button" class="btn btn-sm btn-danger"><i class="fa fa-trash-o"></i></button>

</div>

</div>

member-edit.component.html degisiklik

<tab heading="Edit Photos">

<p>Photo edit will go here</p>

</tab>

<tab heading="Edit Photos">

<app-photo-editor [photos]="user.photos"></app-photo-editor>

</tab>

photo-editor.component.css

img.img-thumbnail {

height: 100px;

min-width: 100px !important;

margin-bottom: 2px;

}

### Angular Photo ekleme

ng-2 file upload aratilir. Valor software yailimi bulunur.

npm install ng2-file-upload --save

app.module imports array e eklenir.

imports: [

FileUploadModule,

photo-editor.component.ts

photo-editor.component.html

photo-editor.component.css

### ASP API Photo yu Main Photo yapma

ASP API Photos Controller da Restful prensibine uymasa da sadece bir propertyi degistirecegimiz icin yeni metod yapilir (HttpPost)

IDatingRepository.cs

Task<Photo> GetMainPhotoForUser(int userId);

DatingRepository.cs

public async Task<Photo> GetMainPhotoForUser(int userId)

{

return await \_context.Photos.Where(u => u.UserId == userId).FirstOrDefaultAsync(p => p.IsMain);

}

PhotosController.cs

[HttpPost("{id}/setMain")]

public async Task<IActionResult> SetMainPhoto(int userId, int id)

{

if (userId != int.Parse(User.FindFirst(ClaimTypes.NameIdentifier).Value))

return Unauthorized();

var user = await \_repo.GetUser(userId);

if (!user.Photos.Any(p => p.Id == id))

return Unauthorized();

var photoFromRepo = await \_repo.GetPhoto(id);

if (photoFromRepo.IsMain)

return BadRequest("This is alredy the main photo");

var currentMainPhoto = await \_repo.GetMainPhotoForUser(userId);

currentMainPhoto.IsMain = false;

photoFromRepo.IsMain = true;

if (await \_repo.SaveAll()) {

return NoContent();

}

return BadRequest("Could not set photo to main");

}

postman

http://localhost:5000/api/users/1/photos/11/setMain

### Angular Photo yu Main Photo yapma

User service de post request yaptigimiz icin object gondermemiz gerekiyor, empty object gonderiyoruz {}.

user.service.ts

setMainPhoto(userId: number, id:number){

return this.http.post(this.baseUrl + 'users/' + userId + '/photos/' + id + '/setMain', {});

}

photo-editor.component.ts

constructor(private authService: AuthService, private userService: UserService,

private alertify: AlertifyService) { }

setMainPhoto(photo: Photo) {

this.userService.setMainPhoto(this.authService.decodedToken.nameid, photo.id).subscribe(() => {

console.log('Successfully set to main');

}, error => {

this.alertify.error(error);

});

}

photo-editor.component.html

<button type="button" class="btn btn-sm mr-1" (click)="setMainPhoto(photo)"

[ngClass]="photo.isMain ? 'btn-success active' : 'btn-secondary'"

[disabled]="photo.isMain">Main</button>

Main butona basinca o butonu main yapmak icin javascript kullaniyoruz (filter). Dugme de guncellendi su an sadece profil main photo degismiyor, cunku bu ust component in elementi. Bu bilgiyi (main photo degisiklik) output ile ust componente gondermemiz lazim.

photo-editor.component.ts

currentMain: Photo;

setMainPhoto(photo: Photo) {

this.userService.setMainPhoto(this.authService.decodedToken.nameid, photo.id).subscribe(() => {

this.currentMain = this.photos.filter(p => p.isMain === true)[0];

this.currentMain.isMain = false;

photo.isMain = true;

}, error => {

this.alertify.error(error);

});

}

### Angular Photo yu Main Photo yapinca UI de (parent comp.) deigisiklik yapma

photo-editor.component.ts

@Output() getMemberPhotoChange = new EventEmitter<string>();

setMainPhoto(photo: Photo) {

this.userService.setMainPhoto(this.authService.decodedToken.nameid, photo.id).subscribe(() => {

this.currentMain = this.photos.filter(p => p.isMain === true)[0];

this.currentMain.isMain = false;

photo.isMain = true;

this.getMemberPhotoChange.emit(photo.url);

}, error => {

this.alertify.error(error);

});

}

member-edit.component.html

$event icersinde photo url

<app-photo-editor [photos]="user.photos" (getMemberPhotoChange)="updateMainPhoto($event)"></app-photo-editor>

member-edit.component.ts

updateMainPhoto(photoUrl) {

this.user.photoUrl = photoUrl;

}

### Main photo yu navbar a ekleme (ASP + Angular)

Burada direk navigasyton bar da network call yaparak resmi getirebiliriz fakat ilave bir network call yapmak istemiyoruz. Ikinci yontem olarak token a ekleyip claim olarak getirebiliriz fakat her API call da token i kontrol ettigimiz icin token i daha buyutmek de istemiyoruz + token i surekli update etmek zor olur.

Bunlarin yerine token ile birlikte user logon olduktan sonra main photo bilgisini local storage da tutacagiz. Bunun icin de ayrica bir DTO yaratabiliriz ama bunu userforlistdto nun icerisine ekleyecegiz.

AuthController.cs

private readonly IMapper \_mapper;

public AuthController(IAuthRepository repo, IConfiguration config, IMapper mapper)

{

\_mapper = mapper;

\_config = config;

\_repo = repo;

}

var token = tokenHandler.CreateToken(tokenDescriptor);

var user = \_mapper.Map<UserForListDto>(userFromRepo);

return Ok(new

{

token = tokenHandler.WriteToken(token),

user

});

Boyle yapinca user object olarak gelir fakat Login methodunda users a Photos u inlucde etmedigimiz icin photourl bilgisi gelmez. Bunun icin :

authRepository.cs

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

{

var user = await \_context.Users.Include(p => p.Photos).FirstOrDefaultAsync(x=>x.UserName==username);

Angular da bunu ilave etmek icin:

auth.service.ts

export class AuthService {

currentUser: User;

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);

localStorage.setItem('user', JSON.stringify(user.user));

this.decodedToken = this.jwtHelper.decodeToken(user.token);

this.currentUser = user.user;

console.log(this.decodedToken);

} }) ); }

token da oldugu gibi bunu da aplikasyonun tamaminda kullanabilmek icin app.component a getiriyoruz.

app.component.ts

ngOnInit() {

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

const user: User = JSON.parse(localStorage.getItem('user'));

if (token) {

this.authService.decodedToken = this.jwtHelper.decodeToken(token);

}

if (user) {

this.authService.currentUser = user;

}

}

nav.component.ts

logout() {

localStorage.removeItem('token');

localStorage.removeItem('user');

this.authService.decodedToken = null;

this.authService.currentUser = null;

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

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

}

nav.component.html

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

<span class="mr-2">

<img src="{{authService.currentUser.photoUrl}}" alt="">

</span>

nav.component.css

img {

max-height: 50px;

border: 2px solid white;

display: inline;

}

### Angular Main photo yu navbar da degistirme (Componentler arasi veri iletisimi)

Main photoyu photo-editor component de degistiriyoruz, bu member-edit in alt componenti. Navbar component i ile bu ikisinin de bagi yok, o yuzden Main photoyu degitirince navbar dakini de degitirmemiz lazim. Servisler tum aplikasyon genelinde kullanilir, diger componentler servislerin property ve methodlarini kulanabilir, burada servis kullanacagiz. Burada BehaviorSubject kullanilacak. Bunu AuthService de yapacagiz:

auth.service.ts (user.png yi biz file olarak ekliyoruz)

import { BehaviorSubject } from 'rxjs';

photoUrl = new BehaviorSubject<string>('../../assets/user.png');

currentPhotoUrl = this.photoUrl.asObservable();

changeMemberPhoto(photoUrl: string) {

this.photoUrl.next(photoUrl);

}

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);

localStorage.setItem('user', JSON.stringify(user.user));

this.decodedToken = this.jwtHelper.decodeToken(user.token);

this.currentUser = user.user;

this.changeMemberPhoto(this.currentUser.photoUrl);

} }) ); }

nav.component.ts

export class NavComponent implements OnInit {

model: any = {};

ngOnInit() {

this.authService.currentPhotoUrl.subscribe(photoUrl => this.photoUrl = photoUrl);

}

nav.component.html

<span class="mr-1">

<img src="{{photoUrl}}" alt="">

</span>

app.component.ts

if (user) {

this.authService.currentUser = user;

this.authService.changeMemberPhoto(user.photoUrl);

}

member-edit.component.ts

export class MemberEditComponent implements OnInit {

photoUrl: string;

ngOnInit() { this.authService.currentPhotoUrl.subscribe(photoUrl => this.photoUrl = photoUrl);

member-edit.component.html

<img class="card-img-top img-thumbnail" src="{{photoUrl}}" alt="{{user.knownAs}}">

photo-editor.component.ts

setMainPhoto(photo: Photo) {

this.authService.changeMemberPhoto(photo.url);

this.authService.currentUser.photoUrl = photo.url;

localStorage.setItem('user', JSON.stringify(this.authService.currentUser));

### ASP API Photo silme

PhotosController.cs

[HttpDelete("{id}")]

public async Task<IActionResult> DeletePhoto(int userId, int id)

{

if (userId != int.Parse(User.FindFirst(ClaimTypes.NameIdentifier).Value))

return Unauthorized();

var user = await \_repo.GetUser(userId);

if (!user.Photos.Any(p => p.Id == id))

return Unauthorized();

var photoFromRepo = await \_repo.GetPhoto(id);

if (photoFromRepo.IsMain)

return BadRequest("You cannot delete your main photo");

if (photoFromRepo.PublicId != null)

{

var deleteParams = new DeletionParams(photoFromRepo.PublicId);

var result = \_cloudinary.Destroy(deleteParams);

if (result.Result == "ok")

{

\_repo.Delete(photoFromRepo);

}

}

if (photoFromRepo.PublicId == null)

{

\_repo.Delete(photoFromRepo);

}

if (await \_repo.SaveAll())

return Ok();

return BadRequest("Failed to delete the photo");

}

### Angular Photo silme

user.service.ts

deletePhoto(userId: number, id: number) {

return this.http.delete(this.baseUrl + 'users/' + userId + '/photos/' + id );}

photo-editor.component.ts

deletePhoto(id: number) {

this.alertify.confirm('Are you sure you want to delete this photo?', () => {

this.userService.deletePhoto(this.authService.decodedToken.nameid, id).subscribe(() => {

this.photos.splice(this.photos.findIndex(p => p.id === id), 1);

this.alertify.success('Photo has been deleted');

}, error => {

this.alertify.error('Failed to delete the photo');

}); }); }

photo-editor.component.html

<button type="button" class="btn btn-sm btn-danger"

(click)="deletePhoto(photo.id)"

[disabled]="photo.isMain" >

### Angular Reactive register form

register.component.ts

registerForm: FormGroup;

ngOnInit() {

this.registerForm = new FormGroup({

username: new FormControl(),

password: new FormControl(),

confirmPassword: new FormControl()

});

}

register() {

// this.authService.register(this.model).subscribe(() => {

// this.alertify.success('registration successful');

// }, error => {

// this.alertify.error(error);

// });

console.log(this.registerForm.value);

}

app.module.ts

imports: [

ReactiveFormsModule,

register.component.html

<form [formGroup]="registerForm" (ngSubnit)="register()">

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

<hr>

<div class="form-group">

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

</div>

<div class="form-group">

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

</div>

<div class="form-group">

<input type="password" class="form-control" formControlName="confirmPassword"

placeholder="Confirm Password">

</div>

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

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

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

</div>

</form>

### Angular register form Validators

register.component.ts

username: new FormControl('Hello', Validators.required ),

password: new FormControl('', [Validators.required, Validators.minLength(4), Validators.maxLength(8)]),

confirmPassword: new FormControl('', Validators.required)

### Angular register form custom validation

register.component.ts

ngOnInit() {

this.registerForm = new FormGroup({

username: new FormControl('Hello', Validators.required ),

password: new FormControl('', [Validators.required, Validators.minLength(4), Validators.maxLength(8)]),

confirmPassword: new FormControl('', Validators.required)

}, this.passwordMatchValidator);

}

passwordMatchValidator(g: FormGroup) {

return g.get('password').value === g.get('confirmPassword').value ? null : {'mismatch': true};

}

register.component.html

<form [formGroup]="registerForm" (ngSubnit)="register()">

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

<hr>

<div class="form-group">

<input type="text"

[ngClass]="{'is-invalid': registerForm.get('username').errors

&& registerForm.get('username').touched }"

class="form-control"

formControlName="username" placeholder="Username">

<div class="invalid-feedback">Please choose a username</div>

</div>

<div class="form-group">

<input type="password"

[ngClass]="{'is-invalid': registerForm.get('password').errors

&& registerForm.get('password').touched }"

class="form-control" formControlName="password"

placeholder="Password">

<div class="invalid-feedback" \*ngIf="registerForm.get('password').hasError('required')

&& registerForm.get('password').touched">Password is required</div>

<div class="invalid-feedback" \*ngIf="registerForm.get('password').hasError('minlength')

&& registerForm.get('password').touched">Min length is 4 chars</div>

<div class="invalid-feedback" \*ngIf="registerForm.get('password').hasError('maxlength')

&& registerForm.get('password').touched">Max length is 8 chars</div>

</div>

<div class="form-group">

<input type="password"

[ngClass]="{'is-invalid': registerForm.get('confirmPassword').errors

&& registerForm.get('confirmPassword').touched

|| registerForm.get('confirmPassword').touched

&& registerForm.hasError('mismatch') }"

class="form-control" formControlName="confirmPassword"

placeholder="Confirm Password">

<div class="invalid-feedback" \*ngIf="registerForm.get('confirmPassword').hasError('required')

&& registerForm.get('confirmPassword').touched">Password is required</div>

<div class="invalid-feedback" \*ngIf="registerForm.hasError('mismatch')

&& registerForm.get('confirmPassword').touched">Passwords must match</div>

</div>

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

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

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

</div>

</form>

<p>Form Value: {{registerForm.value | json}}</p>

<p>Form Status: {{registerForm.status | json}}</p>

### Angular reactive form FormBuilder service

register.component.ts

constructor(private authService: AuthService, private alertify: AlertifyService,

private fb: FormBuilder) { }

ngOnInit() {

this.createRegisterForm();

}

createRegisterForm() {

this.registerForm = this.fb.group({

username: ['', Validators.required],

password: ['', [Validators.required, Validators.minLength(4), Validators.maxLength(8)]],

confirmPassword: ['', Validators.required]

}, {validator: this.passwordMatchValidator});

}

### Angular registration field ekleme

register.component.html

<div class="form-group">

<label class="control-label" style="margin-right:10px">I am a: </label>

<label class="radio-inline">

<input class="mr-3" type="radio" value="male" formControlName="gender">Male

</label>

<label class="radio-inline ml-3">

<input class="mr-3" type="radio" value="female" formControlName="gender">Female

</label>

</div>

<div class="form-group">

<input type="text"

[ngClass]="{'is-invalid': registerForm.get('username').errors

&& registerForm.get('username').touched }"

class="form-control"

formControlName="username" placeholder="Username">

<div class="invalid-feedback">Please choose a username</div>

</div>

<div class="form-group">

<input [ngClass]="{'is-invalid': registerForm.get('knownAs').errors

&& registerForm.get('knownAs').touched}" class="form-control"

placeholder="Known as" formControlName="knownAs">

<div class="invalid-feedback"

\*ngIf="registerForm.get('knownAs').touched && registerForm.get('knownAs').hasError('required')">

Known as is required</div>

</div>

<div class="form-group">

<input [ngClass]="{'is-invalid': registerForm.get('dateOfBirth').errors

&& registerForm.get('dateOfBirth').touched}" class="form-control"

placeholder="Date of Birth" formControlName="dateOfBirth" type="date" >

<div class="invalid-feedback" \*ngIf="registerForm.get('dateOfBirth').touched

&& registerForm.get('dateOfBirth').hasError('required')">Date of Birth is required</div>

</div>

<div class="form-group">

<input [ngClass]="{'is-invalid': registerForm.get('city').errors && registerForm.get('city').touched}" class="form-control"

placeholder="City" formControlName="city">

<div class="invalid-feedback" \*ngIf="registerForm.get('city').touched && registerForm.get('city').hasError('required')">City is required</div>

</div>

<div class="form-group">

<input [ngClass]="{'is-invalid': registerForm.get('country').errors && registerForm.get('country').touched}" class="form-control"

placeholder="Country" formControlName="country">

<div class="invalid-feedback" \*ngIf="registerForm.get('country').touched && registerForm.get('country').hasError('required')">Country is required</div>

</div>

### Angular datepicker input

Ngx-boostrap den alinarak kullanilacak.

app.module.ts

imports: [

BsDatepickerModule.forRoot(),

styles.css

@import '../node\_modules/ngx-bootstrap/datepicker/bs-datepicker.css';

register.component.html

placeholder="Date of Birth" formControlName="dateOfBirth" type="text" bsDatepicker>

color red olarak degistirmek icin Partial class (Make all properties optional) olarak ekliyoruz, icinde zorunlu config itemleri var, hepsini implement etmek istemiyoruz.

register.component.ts

bsConfig: Partial<BsDatepickerConfig>;

ngOnInit() {

this.bsConfig = {

containerClass: 'theme-red'

},

this.createRegisterForm();

}

register.component.html

placeholder="Date of Birth" formControlName="dateOfBirth" type="text" bsDatepicker [bsConfig]="bsConfig">

### ASP API ilave register field lari ekleme

UserForRegisterDto.cs

public class UserForRegisterDto

{

public UserForRegisterDto()

{

Created = DateTime.Now;

LastActive = DateTime.Now;

}

[Required]

public string Username { get; set; }

[Required]

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

public string Password { get; set; }

[Required]

public string Gender { get; set; }

[Required]

public string KnownAs { get; set; }

[Required]

public DateTime DateOfBirth { get; set; }

[Required]

public string City { get; set; }

[Required]

public string Country { get; set; }

public DateTime Created { get; set; }

public DateTime LastActive { get; set; }

}

UserForRegisterDto.cs

CreateMap<UserForRegisterDto, User>();

UsersController.cs

[HttpGet("{id}", Name = "GetUser")]

AuthController.cs

[HttpPost("register")]

public async Task<IActionResult> Register(UserForRegisterDto userForRegisterDto)

{

//validate request

userForRegisterDto.Username = userForRegisterDto.Username.ToLower();

if (await \_repo.UserExists(userForRegisterDto.Username))

return BadRequest("Username already exists");

var userToCreate = \_mapper.Map<User>(userForRegisterDto);

var createdUser = await \_repo.Register(userToCreate, userForRegisterDto.Password);

var userToReturn = \_mapper.Map<UserForDetailedDto>(createdUser);

return CreatedAtRoute("GetUser", new {controller = "Users", id = createdUser.Id}, userToReturn);

}

### Angular register

register.component.ts

export class RegisterComponent implements OnInit {

@Output() cancelRegister = new EventEmitter();

// model: any = {};

user: User;

constructor(private authService: AuthService, private alertify: AlertifyService,

private router: Router, private fb: FormBuilder) { }

register() {

if (this.registerForm.valid) {

this.user = Object.assign({}, this.registerForm.value);

this.authService.register(this.user).subscribe(() => {

this.alertify.success('Registration succesful');

}, error => {

this.alertify.error(error);

}, () => {

this.authService.login(this.user).subscribe(() => {

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

});

});

}

}

auth.service.ts

register(user: User) {

return this.http.post(this.baseUrl + 'register', user);

}

### Angular yeni register kullaniciya default photo getirme + ilk photo islemleri

member-card.component.html

<img class="card-img-top" src="{{user.photoUrl || '../../../../../assets/user.png' }}" alt="{{user.knownAs}}">

nav.component.html

<img src="{{photoUrl || '../../../../assets/user.png'}}" alt="">

member-edit.component.html

<img class="card-img-top img-thumbnail" src="{{photoUrl || '../../../../../assets/user.png'}}" alt="{{user.knownAs}}">

member-detail.component.html

<img class="card-img-top img-thumbnail" src="{{user.photoUrl || '../../../../../assets/user.png'}}" alt="{{user.knownAs}}">

Upload photo dedigimiz zaman bu default photo gostermeye devam ediyor, bunu duzeltmek icin:

photo-editor.component.ts

this.photos.push(photo);

if (photo.isMain) {

this.authService.changeMemberPhoto(photo.url);

this.authService.currentUser.photoUrl = photo.url;

localStorage.setItem('user', JSON.stringify(this.authService.currentUser));

}

### Angular tarihleri duzgun formatta goruntuleme (Angular pipes |)

member-detail.component.html

member-edit.component.html

<p>{{user.created | date: 'mediumDate'}}</p>

Time ago icin ise angular timeago pipe diye aratiyoruz, hazir cozum kullanacagiz. (<https://www.npmjs.com/package/time-ago-pipe>)

npm install time-ago-pipe –save

app.module.ts

import {TimeAgoPipe} from 'time-ago-pipe';

declarations: [

TimeAgoPipe

member-detail.component.html

member-edit.component.html

<p>{{user.lastActive | timeAgo}}</p>

### ASP API last active maintain etme (Action Filter).

ASP API daki UsersController da herhangi bir action execute edildiginde user last active update olacak. Bunun icin action filters kullaniyoruz. Helpers folderi altinda class yaratilir.

LogUserActivity.cs

using System;

using System.Security.Claims;

using System.Threading.Tasks;

using DatingApp.API.Data;

using Microsoft.AspNetCore.Mvc.Filters;

using Microsoft.Extensions.DependencyInjection;

namespace DatingApp.API.Helpers

{

public class LogUserActivity : IAsyncActionFilter

{

public async Task OnActionExecutionAsync(ActionExecutingContext context, ActionExecutionDelegate next)

{

var resultContext = await next();

var userId = int.Parse(resultContext.HttpContext.User

.FindFirst(ClaimTypes.NameIdentifier).Value);

var repo = resultContext.HttpContext.RequestServices.GetService<IDatingRepository>();

var user = await repo.GetUser(userId);

user.LastActive = DateTime.Now;

await repo.SaveAll(); } } }

Startup.cs

services.AddScoped<LogUserActivity>();

UsersController.cs

namespace DatingApp.API.Controllers

{

[ServiceFilter(typeof(LogUserActivity))]

[Authorize]

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

[ApiController]

public class UsersController : ControllerBase

### ASP API Paging

Helpers folder altinda herhangi bir type i ex:users (GetUsers ile gelen) pagination information ile dondurecek class yaratilir.

PagedList.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.EntityFrameworkCore;

namespace DatingApp.API.Helpers

{

public class PagedList<T> : List<T>

{

public PagedList(List<T> items, int count, int pageNumber, int pageSize)

{

TotalCount = count;

PageSize = pageSize;

CurrentPage = pageNumber;

TotalPages = (int)Math.Ceiling(count / (double)pageSize);

this.AddRange(items);

}

public int CurrentPage { get; set; }

public int TotalPages { get; set; }

public int PageSize { get; set; }

public int TotalCount { get; set; }

public static async Task<PagedList<T>> CreatedAsync(IQueryable<T> source,

int pageNumber, int pageSize)

{

var count = await source.CountAsync();

var items = await source.Skip((pageNumber - 1) \* pageSize).Take(pageSize).ToListAsync();

return new PagedList<T>(items, count, pageNumber, pageSize);

} } }

### ASP API Paging Headers

Helpers folderi altinda class yaratilir PaginationHeader.cs.

PaginationHeader.cs

public class PaginationHeader

{

public PaginationHeader(int currentPage, int itemsPerPage, int totalItems, int totalPages)

{

this.CurrentPage = currentPage;

this.ItemsPerPage = itemsPerPage;

this.TotalItems = totalItems;

this.TotalPages = totalPages;

}

public int CurrentPage { get; set; }

public int ItemsPerPage { get; set; }

public int TotalItems { get; set; }

public int TotalPages { get; set; }

}

Bunu response header a yazmak icin Helper folders in altinda extension method yazilir:

Extensions.cs

public static void AddPagination(this HttpResponse response, int currentPage,

int itemsPerPage, int totalItems, int totalPages)

{

var paginationHeader = new PaginationHeader(currentPage, itemsPerPage, totalItems, totalPages);

var camelCaseFormatter = new JsonSerializerSettings();

camelCaseFormatter.ContractResolver = new CamelCasePropertyNamesContractResolver();

response.Headers.Add("Pagination", JsonConvert.SerializeObject(paginationHeader, camelCaseFormatter));

response.Headers.Add("Access-Control-Expose-Headers", "Pagination");

}

UserController da GetUsers Action a parametre gondermek icin Helpers folder altinda yeni class yaratiyoruz.

UserParams.cs

public class UserParams

{

private const int MaxPageSize = 50;

public int PageNumber { get; set; } = 1;

private int pageSize = 10;

public int PageSize

{

get { return pageSize; }

set { pageSize = (value > MaxPageSize) ? MaxPageSize : value ; }

}

}

### ASP API Paging Headers API dan dondurme

IDatingRepository.cs

Task <PagedList<User>> GetUsers(UserParams userParams);

DatingRepository.cs

public async Task<PagedList<User>> [FromQuery]GetUsers(UserParams userParams)

{

var users = \_context.Users.Include(p=>p.Photos);

return await PagedList<User>.CreatedAsync(users, userParams.PageNumber, userParams.PageSize);

}

UsersController.cs

public async Task<IActionResult> GetUsers(UserParams userParams)

{

var users = await \_repo.GetUsers(userParams);

var usersToReturn = Mapper.Map<IEnumerable<UserForListDto>>(users);

Response.AddPagination(users.CurrentPage, users.PageSize, users.TotalCount, users.TotalPages);

return Ok(usersToReturn);

}

### Angular Paging

\_models folderinin altida Interface yaratilir:

Pagination.ts

export interface Pagination {

currentPage: number;

itemsPerPage: number;

totalItems: number;

totalPages: number;

}

export class PaginatedResult<T> {

result: T;

pagination: Pagination;

}

user.service.ts

getUsers(page?, itemsPerPage?): Observable<PaginatedResult<User[]>> {

const paginatedResult: PaginatedResult<User[]> = new PaginatedResult<User[]>();

let params = new HttpParams();

if (page != null && itemsPerPage != null) {

params = params.append('pageNumber', page);

params = params.append('pageSize', itemsPerPage);

}

return this.http.get<User[]>(this.baseUrl + 'users', { observe: 'response', params})

.pipe(

map(response => {

paginatedResult.result = response.body;

if (response.headers.get('Pagination') != null) {

paginatedResult.pagination = JSON.parse(response.headers.get('Pagination'))

}

return paginatedResult;

})

);

}

member-list.resolver.ts

export class MemberListResolver implements Resolve<User[]> {

pageNumber = 1;

pageSize = 5;

constructor(private userService: UserService, private router: Router,

private alertify: AlertifyService) {}

resolve(route: ActivatedRouteSnapshot): Observable<User[]> {

return this.userService.getUsers(this.pageNumber, this.pageSize).pipe(

catchError(error => {

this.alertify.error('Problem retrieveing data');

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

return of(null);

})

);

}

}

member-list.component.ts

ngOnInit() {

this.route.data.subscribe(data => {

this.users = data['users'].result;

});

}

### Angular Paging Plugin

Ngx bootstrap / Valor / Pagination / Custom Links Content

app.module.ts

imports: [

PaginationModule.forRoot(),

member-list.component.ts

export class MemberListComponent implements OnInit {

users: User[];

pagination: Pagination;

constructor(private userService: UserService, private alertify: AlertifyService,

private route: ActivatedRoute) { }

ngOnInit() {

this.route.data.subscribe(data => {

this.users = data['users'].result;

this.pagination = data['users'].pagination;

});

}

pageChanged(event: any): void {

this.pagination.currentPage = event.page;

this.loadUsers();

}

loadUsers(){

this.userService.getUsers(this.pagination.currentPage, this.pagination.itemsPerPage)

.subscribe((res: PaginatedResult<User[]>) => {

this.users = res.result;

this.pagination = res.pagination;

}, error => {

this.alertify.error(error);

});

}

}

member-list.component.html

<div class="container mt-5">

<div class="row">

<div \*ngFor="let user of users" class="col-lg-2 col-md-3 col-sm-6">

<app-member-card [user]="user"></app-member-card>

</div>

</div>

</div>

<div class="d-flex justify-content-center">

<pagination [boundaryLinks]="true" [totalItems]="pagination.totalItems"

[itemsPerPage]="pagination.itemsPerPage"

[(ngModel)]="pagination.currentPage"

(pageChanged) = "pageChanged($event)"

previousText="&lsaquo;"

nextText="&rsaquo;" firstText="&laquo;" lastText="&raquo;">

</pagination>

</div>

### ASP API Filtering (Defaullt Users Gender + not current user)

UserParams.cs

public int UserId { get; set; }

public string Gender { get; set; }

UsersController.cs

public async Task<IActionResult> GetUsers([FromQuery]UserParams userParams)

{

var currentUserId = int.Parse(User.FindFirst(ClaimTypes.NameIdentifier).Value);

var userFromRepo = await \_repo.GetUser(currentUserId);

userParams.UserId = currentUserId;

if (string.IsNullOrEmpty(userParams.Gender))

{

userParams.Gender = userFromRepo.Gender == "male" ? "female" : "male";

}

var users = await \_repo.GetUsers(userParams);

var usersToReturn = Mapper.Map<IEnumerable<UserForListDto>>(users);

Response.AddPagination(users.CurrentPage, users.PageSize, users.TotalCount, users.TotalPages);

return Ok(usersToReturn);

}

DatingRepository.cs

public async Task<PagedList<User>> GetUsers(UserParams userParams)

{

var users = \_context.Users.Include(p=>p.Photos).AsQueryable();

users = users.Where(u => u.Id != userParams.UserId);

users = users.Where(u => u.Gender == userParams.Gender);

return await PagedList<User>.CreatedAsync(users, userParams.PageNumber, userParams.PageSize);

}

### ASP API Filtering (Age + Gender)

UserParams.cs

public int MinAge { get; set; } = 18;

public int MaxAge { get; set; } = 99;

DatingRepository.cs

public async Task<PagedList<User>> GetUsers(UserParams userParams)

{

var users = \_context.Users.Include(p=>p.Photos).AsQueryable();

users = users.Where(u => u.Id != userParams.UserId);

users = users.Where(u => u.Gender == userParams.Gender);

if (userParams.MinAge != 18 || userParams.MaxAge != 99 )

{

var minDob = DateTime.Today.AddYears(-userParams.MaxAge - 1);

var maxDob = DateTime.Today.AddYears(-userParams.MinAge);

users = users.Where(u => u.DateOfBirth >= minDob && u.DateOfBirth <= maxDob);

}

return await PagedList<User>.CreatedAsync(users, userParams.PageNumber, userParams.PageSize);

}

### Angular Filtering (Age + Gender)

user.service.ts

getUsers(page?, itemsPerPage?, userParams?): Observable<PaginatedResult<User[]>> {

const paginatedResult: PaginatedResult<User[]> = new PaginatedResult<User[]>();

let params = new HttpParams();

if (page != null && itemsPerPage != null) {

params = params.append('pageNumber', page);

params = params.append('pageSize', itemsPerPage);

}

if (userParams != null) {

params = params.append('minAge', userParams.minAge);

params = params.append('maxAge', userParams.maxAge);

params = params.append('gender', userParams.gender);

}

return this.http.get<User[]>(this.baseUrl + 'users', { observe: 'response', params})

.pipe(

map(response => {

paginatedResult.result = response.body;

if (response.headers.get('Pagination') != null) {

paginatedResult.pagination = JSON.parse(response.headers.get('Pagination'))

}

return paginatedResult;

})

);

}

member-list.component.ts

export class MemberListComponent implements OnInit {

users: User[];

user: User = JSON.parse(localStorage.getItem('user'));

genderList = [{value: 'male', display: 'Males'}, {value: 'female', display: 'Females'}];

userParams: any = {};

pagination: Pagination;

constructor(private userService: UserService, private alertify: AlertifyService,

private route: ActivatedRoute) { }

ngOnInit() {

this.route.data.subscribe(data => {

this.users = data['users'].result;

this.pagination = data['users'].pagination;

});

this.userParams.gender = this.user.gender === 'female' ? 'male' : 'female';

this.userParams.minAge = 18;

this.userParams.maxAge = 99;

}

pageChanged(event: any): void {

this.pagination.currentPage = event.page;

this.loadUsers();

}

resetFilters() {

this.userParams.gender = this.user.gender === 'female' ? 'male' : 'female';

this.userParams.minAge = 18;

this.userParams.maxAge = 99;

this.loadUsers();

}

loadUsers(){

this.userService.getUsers(this.pagination.currentPage,

this.pagination.itemsPerPage, this.userParams)

.subscribe((res: PaginatedResult<User[]>) => {

this.users = res.result;

this.pagination = res.pagination;

}, error => {

this.alertify.error(error);

});

}

member-list.component.html

<div class="text-center mt-3">

<h2>Your matches - {{pagination.totalItems}} found</h2>

</div>

<div class="container mt-3">

<form class="form-inline" #form="ngForm" (ngSubmit)="loadUsers()" novalidate>

<div class="form-group">

<label for="minAge">Age From</label>

<input type="number" class="form-control ml-1" style="width: 70px" id="minAge"

[(ngModel)]="userParams.minAge" name="minAge">

</div>

<div class="form-group px-2">

<label for="maxAge">Age To</label>

<input type="number" class="form-control ml-1" style="width: 70px" id="maxAge"

[(ngModel)]="userParams.maxAge" name="maxAge">

</div>

<div class="form-group px-2">

<label for="gender">Show: </label>

<select class="form-control ml-1" style="width: 130px" id="gender"

[(ngModel)]="userParams.gender" name="gender">

<option \*ngFor="let gender of genderList" [value]="gender.value">

{{gender.display}}

</option>

</select>

</div>

<button type="submit" class="btn btn-primary" style="margin-left:10px">Apply Filters</button>

<button type="button" class="btn btn-info" (click)="resetFilters()" style="margin-left:10px">Reset Filter</button>

</form>

<br>

<div class="row">

<div \*ngFor="let user of users" class="col-lg-2 col-md-3 col-sm-6">

<app-member-card [user]="user"></app-member-card>

</div>

</div>

</div>

<div class="d-flex justify-content-center">

<pagination [boundaryLinks]="true" [totalItems]="pagination.totalItems"

[itemsPerPage]="pagination.itemsPerPage"

[(ngModel)]="pagination.currentPage"

(pageChanged) = "pageChanged($event)"

previousText="&lsaquo;"

nextText="&rsaquo;" firstText="&laquo;" lastText="&raquo;">

</pagination>

</div>

### ASP API Sorting (Ordering)

UserParams.cs

public string OrderBy { get; set; }

DatingRepository.cs

public async Task<PagedList<User>> GetUsers(UserParams userParams)

{

var users = \_context.Users.Include(p => p.Photos).OrderByDescending(u => u.LastActive)

.AsQueryable();

users = users.Where(u => u.Id != userParams.UserId);

users = users.Where(u => u.Gender == userParams.Gender);

if (userParams.MinAge != 18 || userParams.MaxAge != 99)

{

var minDob = DateTime.Today.AddYears(-userParams.MaxAge - 1);

var maxDob = DateTime.Today.AddYears(-userParams.MinAge);

users = users.Where(u => u.DateOfBirth >= minDob && u.DateOfBirth <= maxDob);

}

if (!string.IsNullOrEmpty(userParams.OrderBy))

{

switch (userParams.OrderBy)

{

case "created":

users = users.OrderByDescending(u => u.Created);

break;

default:

users = users.OrderByDescending(u => u.LastActive);

break;

}

}

return await PagedList<User>.CreatedAsync(users, userParams.PageNumber, userParams.PageSize);

}

### Angular Sorting (Ordering)

ngx-bootstrap den buttons/radio alinir

app.module.ts

imports: [

ButtonsModule.forRoot(),

member-list.component.ts

ngOnInit() {

this.route.data.subscribe(data => {

this.users = data['users'].result;

this.pagination = data['users'].pagination;

});

this.userParams.gender = this.user.gender === 'female' ? 'male' : 'female';

this.userParams.minAge = 18;

this.userParams.maxAge = 99;

this.userParams.orderBy = 'lastActive';

member-list.component.html

<button type="submit" class="btn btn-primary" style="margin-left:10px">Apply Filters</button>

<button type="button" class="btn btn-info" (click)="resetFilters()" style="margin-left:10px">Reset Filter</button>

<div class="col">

<div class="btn-group float-right">

<button type="button" name="orderBy" class="btn btn-primary"

[(ngModel)]="userParams.orderBy" (click)="loadUsers()" btnRadio = "lastActive">Last Active</button>

<button type="button" name="orderBy" class="btn btn-primary"

[(ngModel)]="userParams.orderBy" (click)="loadUsers()" btnRadio = "created">Newest Members</button>

</div>

</div>

</form>

user.service.ts

if (userParams != null) {

params = params.append('minAge', userParams.minAge);

params = params.append('maxAge', userParams.maxAge);

params = params.append('gender', userParams.gender);

params = params.append('orderBy', userParams.orderBy);

}

### ASP API Many to Many relation yaratma (Likes icin)

Models folderi altinda Like.cs yaratilir.

Like.cs

public class Like

{

public int LikerId { get; set; }

public int LikeeId { get; set; }

public User Liker { get; set; }

public User Likee { get; set; }

}

User.cs

public ICollection<Like> Likers { get; set; }

public ICollection<Like> Likees { get; set; }

DataContext.cs

public DbSet<Like> Likes { get; set; }

Default relationlari modify etmek icin DataContext’de OnModelCreate i override ediyoruz. LikerId+LikeeId primary key olacak (bir user diger bir useri birden fazla like edemez).

DataContext.cs

protected override void OnModelCreating(ModelBuilder builder)

{

builder.Entity<Like>()

.HasKey(k => new {k.LikerId, k.LikeeId});

builder.Entity<Like>()

.HasOne(u => u.Likee)

.WithMany(u => u.Likers)

.HasForeignKey(u => u.LikeeId)

.OnDelete(DeleteBehavior.Restrict);

builder.Entity<Like>()

.HasOne(u => u.Liker)

.WithMany(u => u.Likees)

.HasForeignKey(u => u.LikerId)

.OnDelete(DeleteBehavior.Restrict);

}

dotnet ef migrations add AddedLikeEntity

dotnet ef database update

### ASP API Like gonderme

User diger user i 1 kere like edebilir bunu kontrol icin IDatingRepository.cs ye method ekleriz.

IDatingRepository.cs

Task<Like> GetLike(int userId, int recipientId);

DatingRepository.cs

public async Task<Like> GetLike(int userId, int recipientId)

{

return await \_context.Likes.FirstOrDefaultAsync(

u => u.LikerId == userId && u.LikeeId == recipientId);

}

UsersController.cs

[HttpPost("{id}/like/{recipientId}")]

public async Task<IActionResult> LikeUser(int id, int recipientId)

{

if (id != int.Parse(User.FindFirst(ClaimTypes.NameIdentifier).Value))

return Unauthorized();

var like = await \_repo.GetLike(id, recipientId);

if (like != null)

return BadRequest("You already liked the user");

if (await \_repo.GetUser(recipientId) == null)

return NotFound();

like = new Like {

LikerId = id,

LikeeId = recipientId

};

\_repo.Add<Like>(like);

if (await \_repo.SaveAll())

return Ok();

return BadRequest("Failed to like user");

}

### ASP API Like edilen ve eden kullanicilari listeleme

UserParams.cs

public bool Likes { get; set; } = false;

public bool Likers { get; set; } = false;

DatingRepository.cs

private async Task<IEnumerable<int>> GetUserLikes(int id, bool likers)

{

var user = await \_context.Users.Include(x => x.Likers).Include(x => x.Likees)

.FirstOrDefaultAsync(u => u.Id == id);

if (likers)

{

return user.Likers.Where(u => u.LikeeId == id).Select(i => i.LikerId);

}

else

{

return user.Likees.Where(u => u.LikerId == id).Select(i => i.LikeeId);

}

}

public async Task<PagedList<User>> GetUsers(UserParams userParams)

{

var users = \_context.Users.Include(p => p.Photos).OrderByDescending(u => u.LastActive)

.AsQueryable();

users = users.Where(u => u.Id != userParams.UserId);

users = users.Where(u => u.Gender == userParams.Gender);

if (userParams.Likers)

{

var userLikers = await GetUserLikes(userParams.UserId, userParams.Likers);

users = users.Where(u => userLikers.Contains(u.Id));

}

if (userParams.Likees)

{

var userLikees = await GetUserLikes(userParams.UserId, userParams.Likers);

users = users.Where(u => userLikees.Contains(u.Id));

}

if (userParams.MinAge != 18 || userParams.MaxAge != 99)

{

var minDob = DateTime.Today.AddYears(-userParams.MaxAge - 1);

var maxDob = DateTime.Today.AddYears(-userParams.MinAge);

users = users.Where(u => u.DateOfBirth >= minDob && u.DateOfBirth <= maxDob);

}

if (!string.IsNullOrEmpty(userParams.OrderBy))

{

switch (userParams.OrderBy)

{

case "created":

users = users.OrderByDescending(u => u.Created);

break;

default:

users = users.OrderByDescending(u => u.LastActive);

break;

}

}

return await PagedList<User>.CreatedAsync(users, userParams.PageNumber, userParams.PageSize);

}

### Angular kullanici like etme

user.service.ts

sendLike(id: number, recipientId: number) {

return this.http.post(this.baseUrl + 'users/' + id + '/like/' + recipientId, {});

}

member-card.component.ts

export class MemberCardComponent implements OnInit {

@Input() user: User;

constructor(private authService: AuthService, private userService: UserService,

private alertify: AlertifyService) { }

ngOnInit() {

}

sendLike(id: number) {

this.userService.sendLike(this.authService.decodedToken.nameid, id).subscribe(data => {

this.alertify.success('You have liked:' + this.user.knownAs);

}, error => {

this.alertify.error(error);

});

} }

member-card.component.html

<li class="list-inline-item"><button class="btn btn-primary" (click)="sendLike(user.id)"><i class="fa fa-heart"></i></button></li>

### Angular Like edilen ve eden kullanicilari listeleme

user.service.ts

getUsers(page?, itemsPerPage?, userParams?, likesParam?): Observable<PaginatedResult<User[]>> {

const paginatedResult: PaginatedResult<User[]> = new PaginatedResult<User[]>();

if (likesParam === 'Likers') {

params = params.append('likers', 'true');

}

if (likesParam === 'Likees') {

params = params.append('likees', 'true');

}

\_resolvers folderi altinda list.resolver yararilir (new file ile)

.resolver.ts

@Injectable()

export class ListsResolver implements Resolve<User[]> {

pageNumber = 1;

pageSize = 5;

likesParam = 'Likers';

constructor(private userService: UserService, private router: Router,

private alertify: AlertifyService) {}

resolve(route: ActivatedRouteSnapshot): Observable<User[]> {

return this.userService.getUsers(this.pageNumber, this.pageSize,

null,this.likesParam).pipe(

catchError(error => {

this.alertify.error('Problem retrieveing data');

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

return of(null);

})

);

}

}

routes.ts

{ path: 'lists', component: ListsComponent, resolve: {users: ListsResolver}},

app.module.ts

providers: [

ListsResolver

lists.component.ts

export class ListsComponent implements OnInit {

users: User[];

pagination: Pagination;

likesParam: string;

constructor(private authServece: AuthService, private userService: UserService,

private route: ActivatedRoute, private alertify: AlertifyService) { }

ngOnInit() {

this.route.data.subscribe(data => {

this.users = data['users'].result;

this.pagination = data['users'].pagination;

});

this.likesParam = 'Likers';

}

loadUsers() {

this.userService.getUsers(this.pagination.currentPage,

this.pagination.itemsPerPage, null, this.likesParam)

.subscribe((res: PaginatedResult<User[]>) => {

this.users = res.result;

this.pagination = res.pagination;

}, error => {

this.alertify.error(error);

});

}

pageChanged(event: any): void {

this.pagination.currentPage = event.page;

this.loadUsers();

}

}

lists.component.html

<div class="text-center mt-3">

<h2>{{likesParam === 'Likers' ? 'Members who like me' : 'Members who I\'ve Liked'}} : {{pagination.totalItems}}</h2>

</div>

<div class="container mt-3">

<div class="row">

<div class="btn-group">

<button class="btn btn-primary" [(ngModel)]="likesParam" btnRadio="Likers" (click)="loadUsers()">Members who like me</button>

<button class="btn btn-primary" [(ngModel)]="likesParam" btnRadio="Likees" (click)="loadUsers()">Members who I like</button>

</div>

</div>

<br>

<div class="row">

<div \*ngFor="let user of users" class="col-sm-6 col-md-4 col-lg-4 col-xl-2">

<app-member-card [user]="user"></app-member-card>

</div>

</div>

</div>

<div class="d-flex justify-content-center">

<pagination [boundaryLinks]="true" [totalItems]="pagination.totalItems" [itemsPerPage]="pagination.itemsPerPage" [(ngModel)]="pagination.currentPage"

(pageChanged)="pageChanged($event)" previousText="&lsaquo;" nextText="&rsaquo;" firstText="&laquo;" lastText="&raquo;"

>

</pagination>

</div>

### ASP API Messages Entity yaratma

Models folderi altinda Message class yartilir: