**CLI**

**Khoi tao:**

Dotnet new webapi -n tenduan

**Chạy:**

Dotnet run

**Đồng bộ hóa:**

Thêm async Task<IActionResult>

Thêm await: var values = await \_context

**CORS:**

Startup.cs

* ConfigureServices()

Thêm: services.AddCors();

* Configure()

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

**Boottrap**

Npm install boostrap font –awesome

Vào style.css @import `../nodemodules/bootstrap/dist/css/bootstrap.min.css`

@import `../nodemodules/font-awesome/css/ font-awesome.min.css`

**Gitignore aspnet**

.vscode

Bin

Obj

\*.db

**Kết nối cơ sở dữ liệu**

Appsettings.json

"ConnectionStrings": {  
 "DefaultConnection": "server=localhost;port=3306;user=root;password=123456;database=test02"  
}

Startup.cs

services.AddDbContextPool<DataContext>(  
 options => options.UseMySql(Configuration.GetConnectionString("DefaultConnection")  
 ));

Chạy lệnh dotnet tool install --global dotnet-ef cài migrations

Chay de hien thu muc migrations dotnet ef migrations add InitialCreate

*Pomelo.EntityFrameworkCore.MySql có sẵn kết nối mysql*

1. *Microsoft.EntityFrameworkCore*
2. *Pomelo.EntityFrameworkCore.MySql*
3. *Microsoft.EntityFrameworkCore.Tools*

dotnet ef migrations add MyFirstMigration

dotnet ef database update

dotnet --project "C:\Users\admin\Desktop\test\test02\WebApplication\WebApplication" ef migrations add InitialMigration

**service angular**

khai bao trong provinve app.modul.ts

shouldRefresh = new Subject<any>();

constructor(private httpClient: HttpClient) { }

**componen: khoi tao khi khai bao**

constructor(private activatedRoute: ActivatedRoute,

private songsService: SongsService, private playlistService: PlaylistService

) { }

Loi self referencing loop…. Khi co qua nhieu du lieu dc load

services.AddControllers().AddNewtonsoftJson(options =>  
 options.SerializerSettings.ReferenceLoopHandling = Newtonsoft.Json.ReferenceLoopHandling.*Ignore*);

### public IActionResult FromHeader( [FromBody

] string Host) { **return** Content(Host); }

* **Security**

B1: Viết IauthRepository

Task<User> Register(User user, string password);  
Task<User> Login(string username, string password);  
Task<bool> UserExists(string username);

B2: Viết AuthRepository

private readonly DataContext \_context;  
  
public AuthRepository(DataContext context)  
{  
 \_context = context;  
}  
  
public async Task<User> Register(User user, string password)  
{  
 byte[] passwordHash, passwordSalt;  
 CreatePasswordHash(password, out passwordHash, out passwordSalt);  
 user.PasswordHash = passwordHash;  
 user.PasswordSalt = passwordSalt;  
 await \_context.Users.AddAsync(user);  
 await \_context.SaveChangesAsync();  
 return user;  
}  
  
private void CreatePasswordHash(string password, out byte[] passwordHash, out byte[] passwordSalt)  
{  
 using (var hmac = new System.Security.Cryptography.HMACSHA512())  
 {  
 passwordSalt = hmac.Key;  
 passwordHash = hmac.ComputeHash(System.Text.Encoding.UTF8.GetBytes(password));  
 }  
}  
  
public async Task<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 computeHash = hmac.ComputeHash(System.Text.Encoding.UTF8.GetBytes(password));  
 for (int i = 0; i < computeHash.Length; i++)  
 {  
 if (computeHash[i] != passwordHash[i]) return false;  
 }  
 }  
  
 return true;  
}  
  
public async Task<bool> UserExists(string username)  
{  
 if (await \_context.Users.AnyAsync(x => x.Username == username))  
 return true;  
 return false;  
  
}

b3: Khai báo trong startup

services.AddScoped<IAuthRepository, AuthRepository>();

CreatedAtRoute()

Tra ve theo duong link nao do.

* Validate

Trong dto chen them [Required]  
[StringLength(8,MinimumLength = 4,ErrorMessage = "You must specify password between 4-8 chracters")]

Hoac email, phone…

* Jwt
* Them thu vien Microsoft.IdentityModel.Tokens
* System.IdentityModel.Tokens.Jwt
* Microsoft.AspNetCore.Authentication.JwtBearer

Trong appsetting.json

"AppSettings": {  
 "Token": "super secret key"  
},

Startup.cs (doc nguoc jwt thanh ma xac nhan)

services.AddAuthentication(auth =>  
 {  
 auth.DefaultAuthenticateScheme = JwtBearerDefaults.*AuthenticationScheme*;  
*//lien ket voi he thong [Authorize] theo mac dinh*auth.DefaultChallengeScheme = JwtBearerDefaults.*AuthenticationScheme*;  
*//dieu huong den trang dang nhap khi khong co quyen*

})  
 .AddJwtBearer(options =>  
 {  
 options.TokenValidationParameters = new TokenValidationParameters  
 {  
 ValidateIssuerSigningKey = true,  
 IssuerSigningKey = new SymmetricSecurityKey(Encoding.ASCII  
 .GetBytes(Configuration.GetSection("AppSettings:Token").Value)),  
 ValidateIssuer = false,  
 *//Usually, this is your application base URL* ValidateAudience = false,  
 *//Here, we are creating and using JWT within the same application.  
 //In this case, base URL is fine.  
 //If the JWT is created using a web service, then this would be the consumer URL.* RequireExpirationTime = true,  
 ValidateLifetime = true,  
 ClockSkew = TimeSpan.*Zero* };  
 });

app.UseAuthentication();  
  
app.UseAuthorization();

Them vao trong controller de biet can qua jwt

[Authorize]

[AllowAnonymous] : Tat ca deu co quyen truy cap

Con khong de la phai co quyen moi dc truy cap

* Dtos (data transfer objects) de thay doi hien thi thong tin oject

@Output() cancelRegister = new EventEmitter();

this.cancelRegister.emit(false);

(cancelRegister)="cancelRegisterModel($event)

Event o day da nhan gia gi false

* Excep

Trong statup

else

{

app.UseExceptionHandler(builder =>

{

builder.Run(async context =>

{

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

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

if (error != null)

{

context.Response.AddApplicationError(error.Error.Message);

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

}

});

});

}

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","\*");  
 }  
}

* Intercepter error
* @Injectable()  
  export class ErrorIntercepter implements HttpInterceptor {  
   intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {  
   return next.handle(req).pipe(  
   *catchError*(err => {  
   if (err.status === 401) {  
   return *throwError*(err.statusText);  
   }  
   if (err instanceof HttpErrorResponse) {  
   const applicationError = err.headers.get('Application-Error');  
   if (applicationError) {  
   return *throwError*(applicationError);  
   }  
   const severError = err.error;  
   let modalStateErrors = '';  
   if (severError.errors && typeof severError.errors === 'object') {  
   for (const key in severError.errors) {  
   if (severError.errors[key]) {  
   modalStateErrors += severError.errors[key] + '\n';  
   }  
   }  
   }  
   return *throwError*(modalStateErrors || severError || 'Server Error');  
   }  
   })  
   );  
   }  
  }  
    
  export const ErrorInterceptorProvider = {  
   provide: HTTP\_INTERCEPTORS,  
   useClass: ErrorIntercepter,  
   multi: true  
  };

[**https://alertifyjs.com/**](https://alertifyjs.com/)

**cai dat thong bao**

**npm i @auth0/angular-jwt**

npm install ngx-bootstrap --save

[**https://valor-software.com/ngx-bootstrap/#/documentation#getting-started**](https://valor-software.com/ngx-bootstrap/#/documentation)

npm install bootswatch

guard router

canActivate: [AuthGuard] trong router

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']);  
 }  
  
}

* **Automapper.Extensions.Microsoft.DependencyInjectstion**

**Mapper: Thay doi thuoc tinh (dung kem DTO)**

**Taoj dto thay doi**

**Tao phan chuyen doi trong helpper**

public class AutoMapperProfiles: Profile  
{  
 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.MapFrom(src=>src.DateOfBirth.CaculateAge()));  
   
 CreateMap<User, UserForDetailedDto>()  
 .ForMember(dest=>dest.PhotoUrl,opt  
 =>opt.MapFrom(src=>src.Photos.FirstOrDefault(p=>p.IsMain).Url))  
 .ForMember(dest=>dest.Age,opt  
 =>opt.MapFrom(src=>src.DateOfBirth.CaculateAge()));  
   
 CreateMap<Photo, PhotosForDetailedDto>();  
}

}

**Phan tao tuoi viet trong file extencent**

* **Intercepter angular**
* JwtModule.*forRoot*({  
   config: {  
   tokenGetter: () => {  
   return localStorage.getItem('token');  
   },  
   allowedDomains: ['localhost:5000'],  
   disallowedRoutes: ['localhost:5000/api/auth']  
   }  
   })  
  ],

**Tab nhay trang con ngx boostrap**

**Resovel:**

**Tao 1 resover**

@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*(err => {  
 this.alertify.error('Problem retrieving data');  
 this.router.navigate(['/members']);  
 return *of*(null);  
 })  
 );  
 }  
}

* Trong oninit :

ngOnInit(): void {  
 this.route.data.subscribe(data => {  
 this.user = data.user;  
 });  
}

private route: ActivatedRoute

**Khai bao trong app.modul** MemberDetailResolver,

**Khai bao trong router**

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

**Ngx galery de hien thi anh**

**Cai dat:**

**npm install** @kolkov/**ngx**-**gallery** –save

import NgxGalleryModule,

* **Sua form xong cho moi thu ve nhu luc dau(cau lenh thong bao , nut edit khong hien thi)**

@ViewChild('editForm,{static:true}')editForm: NgForm;

this.editForm.reset(this.user);

* **Thong bao xac nhan xem co thay doi hay khong?**
* **Tao guard kieu canDeactive**
* @Injectable({  
   providedIn: 'root'  
  })  
  export class PreventUnsaveChangesGuard implements CanDeactivate<MemberEditComponent> {  
    
   *// tslint:disable-next-line:typedef* canDeactivate(component: MemberEditComponent) {  
   if (component.editForm.dirty) {  
   return *confirm*('Are you sure you want to continue? Any unsaved changes will be lost');  
   }  
   return true;  
   }  
  }
* Khai bao trong appmodule: provide

Khai bao trong router: canDeactivate: [PreventUnsaveChangesGuard]

* **Tat web khi chua luu thong tin**
* @HostListener('window:beforeunload', ['$event'])  
  *// tslint:disable-next-line:typedef*unloadNotification($event: any) {  
   if (this.editForm.dirty) {  
   $event.returnValue = true;  
   }  
  }
* **Api edit**

[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");  
}

* **Reactive form module**

**App-modul: import reactiveForm Modul**

ngOnInit(): void {  
 this.registerForm = new FormGroup({  
 username: new FormControl('', Validators.*required*),  
 password: new FormControl('', [Validators.*required*, Validators.*minLength*(4), Validators.*maxLength*(8)]),  
 confirmPassword: new FormControl('', [Validators.*required*, Validators.*minLength*(4), Validators.*maxLength*(8)])  
 }, this.passwordMatchValidator);  
}  
  
*// tslint:disable-next-line:typedef*passwordMatchValidator(g: FormGroup) {  
 if (g.get('password').value === g.get('confirmPassword').value) {  
 return null;  
 }  
 return {mismatch: true};  
}

[formGroup]="registerForm"

formControlName="username"

<input type="password"  
 [ngClass]="{'is-invalid': registerForm.get('password').errors && registerForm.get('password').touched}"  
 class="form-control" placeholder="Password" formControlName="password">  
<div class="invalid-feedback"  
 *\*ngIf*="registerForm.get('password').hasError('required')&&registerForm.get('password').touched">Password is  
 required

* **Daypicker**

**Import:** BrowserAnimationsModule,

BsDatepickerModule.*forRoot*(),

**Import** node\_modules/ngx-bootstrap/datepicker/bs-datepicker.css trong style.css

bsDatepicker trong html

* return CreatedAtRoute("GetUser",new{controller="User",id=createdUser.Id},userToReturn);
* **tro ve trang nao do**
* register() {  
   if (true) {  
   this.user = Object.assign({}, this.registerForm.value);  
   this.authService.register(this.user).subscribe(() => {  
   this.alertify.success('Registration successful');  
   }, error => {  
   this.alertify.error(error);  
   }, () => {  
   this.authService.login(this.user).subscribe(() => {  
   this.router.navigate(['/members']);  
   });  
   });  
   }  
  }
* **Set anh default khi khong co**
* <img class="card-img-top" src="{{user.photoUrl|| '../../../assets/user.jpg'}}" alt="{{user.knownAs}}">
* Loc hien thi thoi gian (filter)
* {{user.created | date: 'mediumDate'}}

Cai timeago de hien thi thoi gian dem nguoc

Npm I ngx-timeago

TimeagoModule.*forRoot*(),

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

* Last activiti

Tao file loguseractivity

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();  
 }  
}

khai bao service

services.AddScoped<LogUserActivity>();

controller nao dung thi them truong

[ServiceFilter(typeof(LogUserActivity))]

* Paging (Phan trang) backend

Trong helper tao trang phan trang

* Tao pagelist

public class PagedList<T> : List<T>  
{  
 public int CurrentPage { get; set; }  
 public int TotalPages { get; set; }  
 public int PageSize { get; set; }  
 public int TotalCount { get; set; }  
  
 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 static async Task<PagedList<T>> CreateAsync(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);  
 }  
}

* Tao PaginationHeader
* public class PaginationHeader  
  {  
   public int CurrentPage { get; set; }  
   public int ItemsPerPage { get; set; }  
   public int TotalItems { get; set; }  
   public int TotalPages { get; set; }  
    
   public PaginationHeader(int currentPage, int itemsPerPage, int totalItems, int totalPages)  
   {  
   CurrentPage = currentPage;  
   ItemsPerPage = itemsPerPage;  
   TotalItems = totalItems;  
   TotalPages = totalPages;  
   }  
     
     
     
  }
* dinh vao request header tai extensions

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");  
   
}

* tao userParam

public class UserParams  
{  
 private const int *MaxPageSize* = 50;  
 public int PageNumber { get; set; } = 1;  
 public int pageSize = 10;  
  
 public int PageSize1  
 {  
 get => pageSize;  
 set => pageSize = (value > *MaxPageSize*) ? *MaxPageSize* : value;  
 }  
}

* Chinh lai repository

public async Task<PagedList<User>> GetAllValue(UserParams userParams)  
{  
 var users = \_context.Users.Include(p => p.Photos);  
 return await PagedList<User>.CreateAsync(users,userParams.PageNumber,userParams.pageSize);  
}

* Controller

public async Task<IActionResult> GetUsers([FromQuery]UserParams userParams)  
{  
 var users = await \_repo.GetAllValue(userParams);  
 var usersToReturn = \_mapper.Map<IEnumerable<UserForListDto>>(users);  
 Response.AddPagination(users.CurrentPage, users.PageSize, users.TotalCount, users.TotalPages);  
 return Ok(usersToReturn);  
}

* **Phan trang frontend**
* Tao interface paginate

export interface Pagination {  
 currentPage: number;  
 itemsPerPage: number;  
 totalItems: number;  
 totalPages: number;  
}  
  
export class PaginatedResult<T>{  
 resullt: T;  
 pagination: Pagination;  
}

* Sua lai trong service

getAllUser(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 + 'user', {observe: 'response', params})  
 .pipe(  
 *map*(response => {  
 paginatedResult.resullt = response.body;  
 if (response.headers.get('Pagination') != null) {  
 paginatedResult.pagination = JSON.parse(response.headers.get('Pagination'));  
 }  
 return paginatedResult;  
 })  
 );  
}

* Chinh lai trong resolver(hoc load thang trong oninit)

pageNumber = 1;  
pageSize = 5;

truyen lai vao trong ham

ngOnInit(): void {  
 this.loadUsers(this.pageNumber, this.pageSize);  
 *// this.route.data.subscribe(data => {  
 // this.users = data.users.result;  
 // console.log(this.users);  
 // });*}  
  
*// tslint:disable-next-line:typedef*loadUsers(x, y) {  
 this.userService.getAllUser(x, y).subscribe(value => {  
 this.users = value.resullt, this.pagination = value.pagination;  
 },  
 error => {  
 this.alertify.error(error);  
 });  
}  
  
pageChanged(event: any): void {  
 this.pagination.currentPage = event.page;  
 this.loadUsers(this.pagination.currentPage, this.pagination.itemsPerPage);  
}

dung ngx boostrap de hien thi

* Finter (tim kiem)
* Sua trong controller (auto hien ra nguoi khac gioi tinh khi dang nhap)

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";  
 }

* Sua repo

public async Task<PagedList<User>> GetAllValue(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>.CreateAsync(users, userParams.PageNumber, userParams.pageSize);  
}

* Sua trong angular
* Chinh sua trong service

getAllUser(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 + 'user', {observe: 'response', params})  
 .pipe(  
 *map*(response => {  
 paginatedResult.resullt = response.body;  
 if (response.headers.get('Pagination') != null) {  
 paginatedResult.pagination = JSON.parse(response.headers.get('Pagination'));  
 }  
 return paginatedResult;  
 })  
 );  
}

* Chinh sua trong component
* users: User[];  
  user: User = JSON.parse(localStorage.getItem('user'));  
  genderList = [{value: 'male', display: 'Males'}, {value: 'female', display: 'Females'}];  
  **userParams: any = {};**  
  pageNumber = 1;  
  pageSize = 3;  
  pagination: Pagination;  
    
  constructor(private userService: UserService, private alertify: AlertifyService, private route: ActivatedRoute) {  
  }  
    
  ngOnInit(): void {  
   **this.userParams.gender = this.user.gender === 'female' ? 'male' : 'female';  
   this.userParams.minAge = 1;  
   this.userParams.maxAge = 99;  
   this.loadUsers(this.pageNumber, this.pageSize);**  
    
  }  
    
  *// tslint:disable-next-line:typedef*loadUsers(x, y) {  
   this.userService.getAllUser(x, y, **this.userParams**).subscribe(value => {  
   this.users = value.resullt, this.pagination = value.pagination;  
   },  
   error => {  
   this.alertify.error(error);  
   });  
  }  
    
  pageChanged(event: any): void {  
   this.pagination.currentPage = event.page;  
   this.loadUsers(this.pagination.currentPage, this.pagination.itemsPerPage);  
  }  
    
  *// tslint:disable-next-line:typedef***resetFilters() {  
   this.userParams.gender = this.user.gender === 'female' ? 'male' : 'female';  
   this.userParams.minAge = 1;  
   this.userParams.maxAge = 99;  
   this.loadUsers(this.pageNumber, this.pageSize);**  
  }
* role

user: IdentityUser<int>

public virtual ICollection<UserRole> UserRoles { get; set; }

UserRole:

public class UserRole: IdentityUserRole<int>

{  
 public virtual User User { get; set; }  
 public virtual Role Role { get; set; }  
}

Role:

public class Role:IdentityRole<int>  
{  
 public ICollection<UserRole> UserRoles { get; set; }  
}

nuget: Microsoft.aspNEtCore.Identity.EntityFrameworkCore

public class DataContext : IdentityDbContext<User, Role, int, IdentityUserClaim<int>, UserRole,  
 IdentityUserLogin<int>, IdentityRoleClaim<int>, IdentityUserToken<int>>  
{

trong startup:

ConfigureServices

IdentityBuilder builder = services.AddIdentityCore<User>(opt =>  
{  
 opt.Password.RequireDigit = false;  
 opt.Password.RequiredLength = 4;  
 opt.Password.RequireNonAlphanumeric = false;  
 opt.Password.RequireUppercase = false;  
});  
  
builder = new IdentityBuilder(builder.UserType, typeof(Role),builder.Services);  
builder.AddEntityFrameworkStores<DataContext>();  
builder.AddRoleValidator<RoleValidator<Role>>();  
builder.AddRoleManager<RoleManager<Role>>();  
builder.AddSignInManager<SignInManager<User>>();

services.AddControllers(option =>  
{  
 var policy = new AuthorizationPolicyBuilder()  
 .RequireAuthenticatedUser().Build();  
 option.Filters.Add(new AuthorizeFilter(policy));  
}).AddNewtonsoftJson(options =>  
 options.SerializerSettings.ReferenceLoopHandling = Newtonsoft.Json.ReferenceLoopHandling.*Ignore*);

policy: doc xem co quyen gi thi duoc thuc hien lenh gi trong controller

services.AddAuthorization(options =>  
{  
 options.AddPolicy("RequireAdminRole", policy => policy.RequireRole("Admin"));  
 options.AddPolicy("ModeratePhotoRole", policy => policy.RequireRole("Admin","Moderator"));  
 options.AddPolicy("VipOnly", policy => policy.RequireRole("VIP"));  
});

controller

[Authorize(Policy = "RequireAdminRole")]  
[HttpGet("usersWithRoles")]  
  
public IActionResult GetUsersWithRole()  
{  
 return Ok("Only admins can see this");  
}  
  
[Authorize(Policy = "ModeratePhotoRole")]  
[HttpGet("photosForModeration")]  
public IActionResult GetPhotosForModeration()  
{  
 return Ok("Admins or moderators can see this");  
}