TUGAS .NET – SIBKM BATCH 4

NAMA : Ma’ruf Mubaroq

Universitas : Universitas Muhammadiyah Surakarta

1. Register

```
using API.Context;
using API.Handlers;
using API.Models;
using API.Repositories.Interface;
using API.ViewModels;

namespace API.Repositories.Data;

public class AccountRepository : GeneralRepository<Account, string, MyContext>, IAccountRepository
{
    public AccountRepository(MyContext context) : base(context) { }

    public int Register(RegisterVM registerVM)
    {
        int result = 0;
        // University Table
        var university = new University
        {
            Name = registerVM.UniversityName
        };
        _context.Set<University>().Add(university);
        result = _context.SaveChanges();

        // Education Table
        var education = new Education
        {
            Major = registerVM.Major,
            Degree = registerVM.Degree,
            GPA = registerVM.GPA,
            UniversityId = university.Id
        };
        _context.Set<Education>().Add(education);
        result += _context.SaveChanges();

        // Employee Table
        var employee = new Employee
        {
            NIK = registerVM.NIK,
            FirstName = registerVM.FirstName,
            LastName = registerVM.LastName,
            Gender = registerVM.Gender,
            BirthDate = registerVM.BirthDate,
            Email = registerVM.Email,
            HiringDate = DateTime.Now,
            PhoneNumber = registerVM.PhoneNumber
        };
        _context.Set<Employee>().Add(employee);
        result += _context.SaveChanges();

        // Account Table
        var account = new Account
        {
            EmployeeNIK = registerVM.NIK,
            Password = Hashing.HashPassword(registerVM.Password)
        };
        _context.Set<Account>().Add(account);
        result += _context.SaveChanges();

        // Profiling Table
        var profiling = new Profiling
        {
            EmployeeNIK = registerVM.NIK,
            EducationId = education.Id
        };
        _context.Set<Profiling>().Add(profiling);
        result += _context.SaveChanges();

        // AccountRole Table
        var accountRole = new AccountRole
        {
            AccountNIK = registerVM.NIK,
            RoleId = 1 // user
        };
        _context.Set<AccountRole>().Add(accountRole);
        result += _context.SaveChanges();

        return result;
    }
```

Methode register adalah sebeuah metode yang dibuat untuk melakukan penambahan data ke database sesuai dengan apa yang di input oleh user dan di masukan ke masing-masing tabel

2. Login

```
public bool Login(LoginVM loginVM)
{
    // ambil data dari database berdasarkan email di tabel employee
    // gabungkan dari data tabel employee dengan tabel account berdasarkan NIK
    // cocokan data tersebut dengan password yang diinputkan
    // cek apakah data falid atau tidak

    var getEmployeeAccount = _context.Employees.Join(_context.Accounts,
        e => e.NIK,
        a => a.EmployeeNIK,
        (e, a) => new
        {
            Email = e.Email,
            Password = a.Password
        }).FirstOrDefault(e => e.Email == loginVM.Email);
    if (getEmployeeAccount == null)
    {
        return false;
    }

    return Hashing.ValidatePassword(loginVM.Password, getEmployeeAccount.Password);
}
}
```

Methode yang dibuat untuk memanggil data yang sudah atau pernah di inputkan oleh user, lalu melakukan validasi apakah data yang di input sesuai dengan datbase? Kalau sama akan dapat masuk kalau tidak akan eror.

3. Hashing

```
namespace API.Handlers;

public class Hashing
{
    private static string GetRandomSalt()
    {
        return BCrypt.Net.BCrypt.GenerateSalt(12);
    }
    public static string HashPassword(string password)
    {
        return BCrypt.Net.BCrypt.HashPassword(password, GetRandomSalt());
    }
    public static bool ValidatePassword(string password, string correctHash)
    {
        return BCrypt.Net.BCr
    }
}
```

bool Hashing.ValidatePassword(string password, string correctHash)

Method hashing adalah berfungsi untuk mengencrypt password.
GetRandomSalt() berfungsi menambahkan 12 digit random
Validatepassword berfungsi untuk mengecek apakah password yang di input sama dengan yang di database
Hashpassword() berfungsi untuk menghash passwoRd dan ditambah 12 digit.

4. AccountRoleRepository

```
using API.Context;
using API.Models;
using API.Repositories.Interface;

namespace API.Repositories.Data;

public class AccountRoleRepository : GeneralRepository<AccountRole, int, MyContext>, IAccountRoleRepository
{
    public AccountRoleRepository(MyContext context) : base(context)
    {
    }
    public IEnumerable<string> GetRolesByEmail(string email)
    {
        var EmployeeNIK = _context.Employees.FirstOrDefault(e => e.Email == email).NIK;
        var AccountRoles = _context.AccountRoles
            .Where(ar => ar.AccountNIK == EmployeeNIK)
            .Join(_context.Roles, ar => ar.RoleId, r => r.Id, (ar, r) => new { ar, r })
            .Select(role => role.r.Name);

        return AccountRoles;
    }
}
```

Di gunakan untuk memanggil nama role berdasarkan email

5. EmployeeRepository

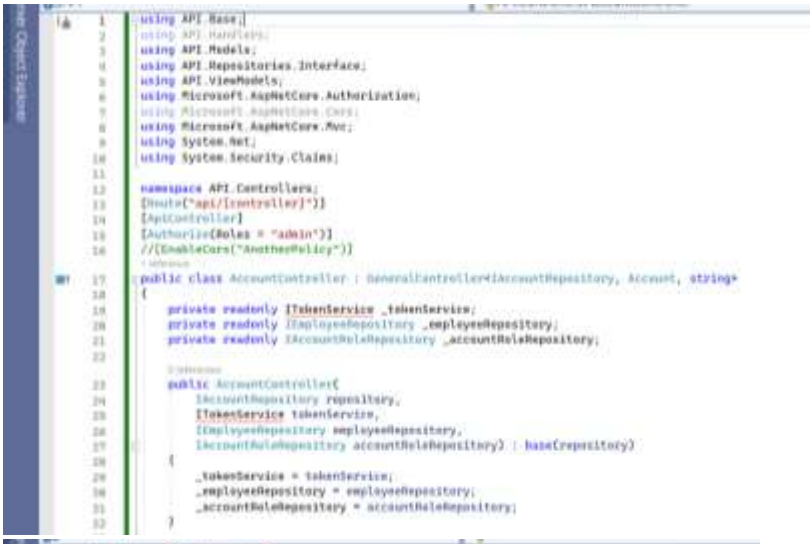
```
using API.Context;
using API.Models;
using API.Repositories.Interface;

namespace API.Repositories.Data;

public class EmployeeRepository : GeneralRepository<Employee, string, MyContext>, IEmployeeRepository
{
    public EmployeeRepository(MyContext context) : base(context) { }
    public string GetFullNameByEmail(string email)
    {
        var employee = _context.Employees.FirstOrDefault(e => e.Email == email);
        return employee.FirstName + " " + employee.LastName;
    }
}
```

Befungsi untuk mengembalikan nama awal dan akhir berdasarkan email.

6. AccountContriller



```
using API.Base;
using API.Handlers;
using API.Models;
using API.Repositories.Interface;
using API.ViewModels;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Cors;
using Microsoft.AspNetCore.Mvc;
using System.Net;
using System.Security.Claims;

namespace API.Controllers;
[Route("api/[controller]")]
[ApiController]
[Authorize(Roles = "admin")]
//[EnableCors("AnotherPolicy")]
public class AccountController : GeneralController<IAccountRepository, Account, string>
{
    private readonly ITokenService _tokenService;
    private readonly IEmployeeRepository _employeeRepository;
    private readonly IAccountRoleRepository _accountRoleRepository;

    public AccountController(
        IAccountRepository repository,
        ITokenService tokenService,
        IEmployeeRepository employeeRepository,
        IAccountRoleRepository accountRoleRepository) : base(repository)
    {
        _tokenService = tokenService;
        _employeeRepository = employeeRepository;
        _accountRoleRepository = accountRoleRepository;
    }
```

```
    [AllowAnonymous]
    [HttpPost("Login")]
    public ActionResult Login(LoginVM loginVM)
    {
        var login = _repository.Login(loginVM);
        if (!login)
        {
            return NotFound(new ResponseErrorVM<string>
            {
                Code = StatusCodes.Status404NotFound,
                Status = HttpStatusCode.NotFound.ToString(),
                Errors = "Login Failed, Account or Password Not Found!"
            });
        }

        var claims = new List<Claim>() {
            new Claim("Email", loginVM.Email),
            new Claim("FullName", _employeeRepository.GetFullNameByEmail(loginVM.Email))
        };

        var getRoles = _accountRoleRepository.GetRolesByEmail(loginVM.Email);
        foreach (var role in getRoles)
        {
            claims.Add(new Claim(ClaimTypes.Role, role));
        }

        var token = _tokenService.GenerateToken(claims);

        return Ok(new ResponseDataVM<string>
        {
            Code = StatusCodes.Status200OK,
            Status = HttpStatusCode.OK.ToString(),
            Message = "Login Success",
            Data = token
        });
    }
```

```
    [AllowAnonymous]
    [HttpPost("Register")]
    public ActionResult Register(RegisterVM registerVM)
    {
        var register = _repository.Register(registerVM);
        if (register > 0)
        {
            return Ok(new ResponseDataVM<string>
            {
                Code = StatusCodes.Status200OK,
                Status = HttpStatusCode.OK.ToString(),
                Message = "Insert Success"
            });
        }

        return BadRequest(new ResponseErrorVM<string>
        {
            Code = StatusCodes.Status500InternalServerError,
            Status = HttpStatusCode.InternalServerError.ToString(),
            Errors = "Insert Failed / Lost Connection"
        });
    }
}
```

Method login dan Register ditambahkan anotasi AllowAnonymous sehingga user dapat melakukan login/register tanpa token

7. AppSetting

Di tambhkan object jwt yang berisi key issuer dan audience

8. Program

```
builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)
    .AddJwtBearer(options =>
    {
        options.RequireHttpsMetadata = false;
        options.SaveToken = true;
        options.TokenValidationParameters = new TokenValidationParameters
        {
            ValidateIssuer = true,
            ValidateAudience = true,
            ValidateLifetime = true,
            ValidateIssuerSigningKey = true,
            ValidIssuer = builder.Configuration["Jwt:Issuer"],
            ValidAudience = builder.Configuration["Jwt:Audience"],
            IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(builder.Configuration["Jwt:Key"])),
            ClockSkew = TimeSpan.Zero
        };
    });

// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseHttpsRedirection();

app.UseCors();

app.UseAuthentication();
```

Untuk kofigurasi JWT dengan API membutuhkan token.