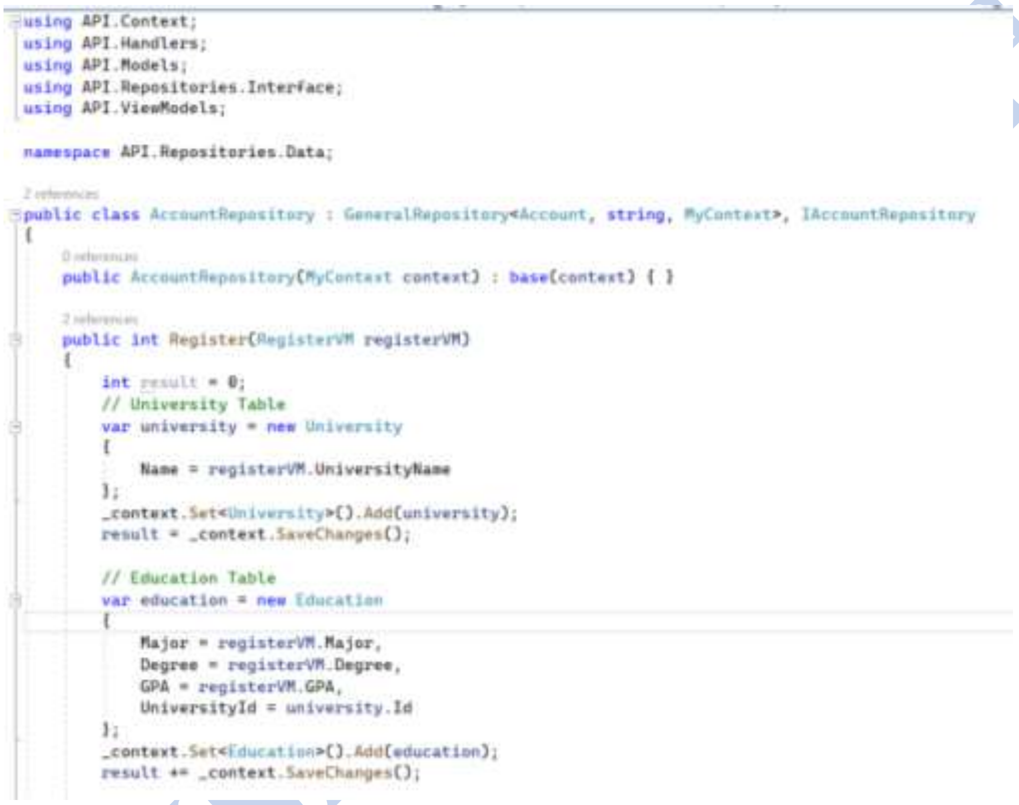
C# .NET – SIBKM BATCH 4
WEEK-13
TUGAS dan MEETS

Nama	: Annisah Nur Aidah
Instansi	: Universitas Muhammadiyah Surakarta

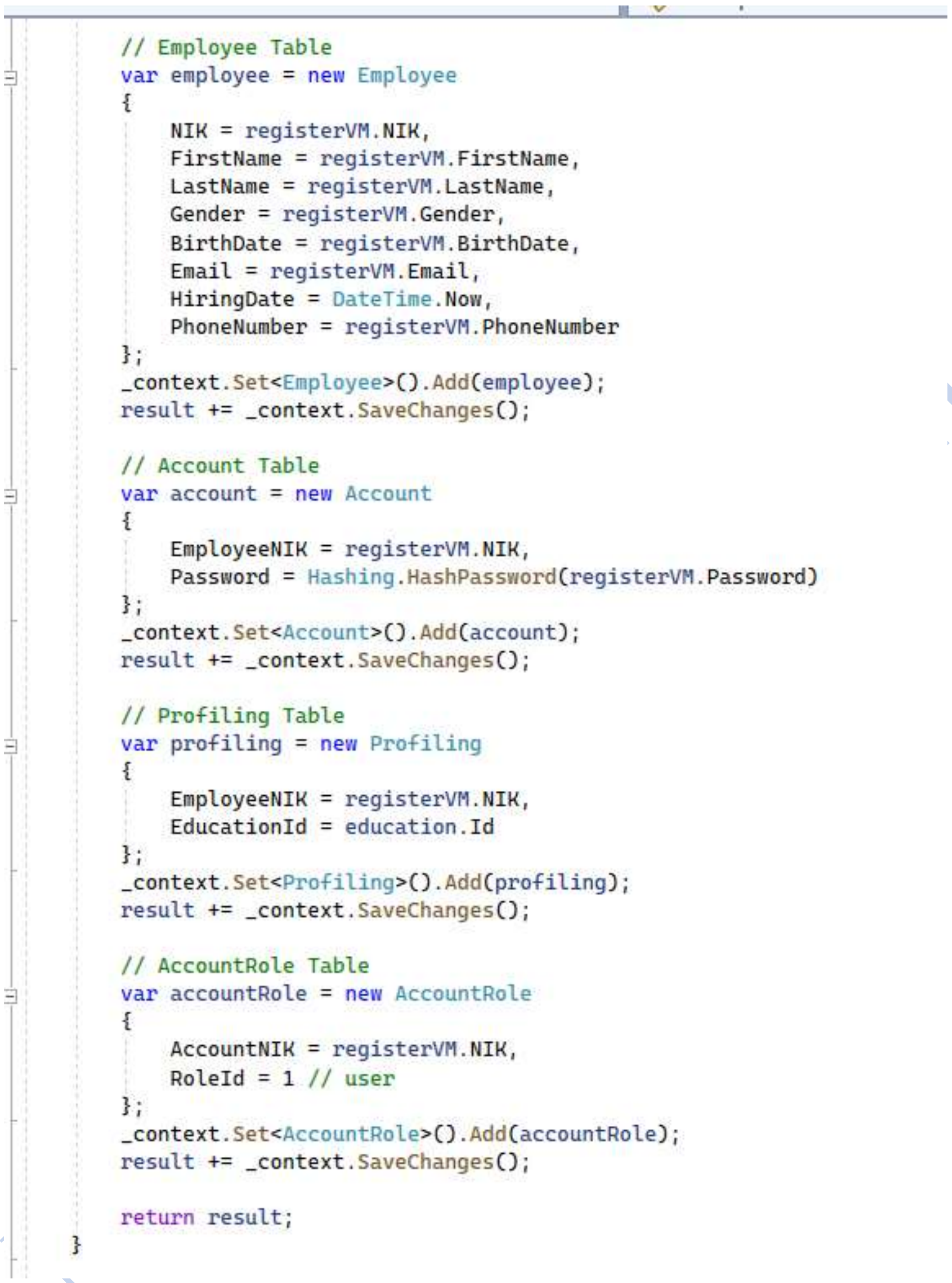
1. **Register**
   Penjelasan :

   Dalam method register ini berfungsi untuk menampilkan suatu konteks dalam integer yang berhasil ditambahkan kedalam database dan diterima dengan pengambilan tipe RegisterVM, yang memasukkan beberapa data yang diperlukan untuk ditambahkan kedalam database.

   Terdapat beberapa tabel yang akan diisi sesuai dengan yang diinputkan oleh user dan dimasukkan kedalam database.



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

namespace API.Repositories.Data;

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

    2 references
    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();
```



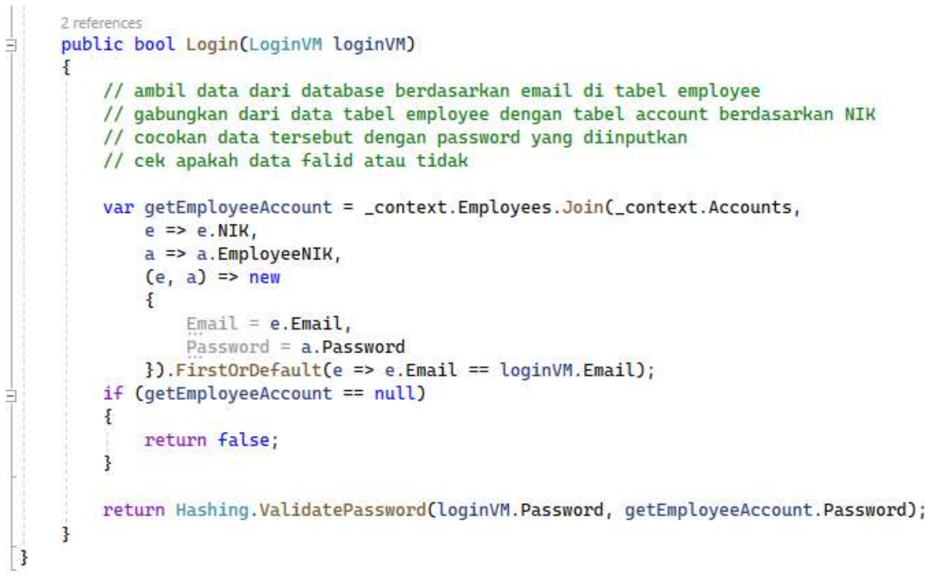
```
        // Employee Table
        var employee = new Employee
        {
            NIK = registerVM.NIK,
            FirstName = registerVM.FirstName,
            LastName = registerVM.LastName,
            Gender = registerVM.Gender,
            BirthDate = registerVM.BirthDate,
            Email = 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;
    }
```

2. Login

Penjelasan :

Dalam method login ini digunakan untuk mengecek apakah user dapat memasukkan email dan password dengan valid atau tidak valid. Dalam penulisan getEmployeeAccount berfungsi untuk menghubungkan tabel employee dengan account sesuai dengan NIK.

Dalam pengecekan password bisa dilakukan dengan cara validate password pada class hashing, dengan ketentuan apabila benar yaitu true dan apabila salah berarti false.

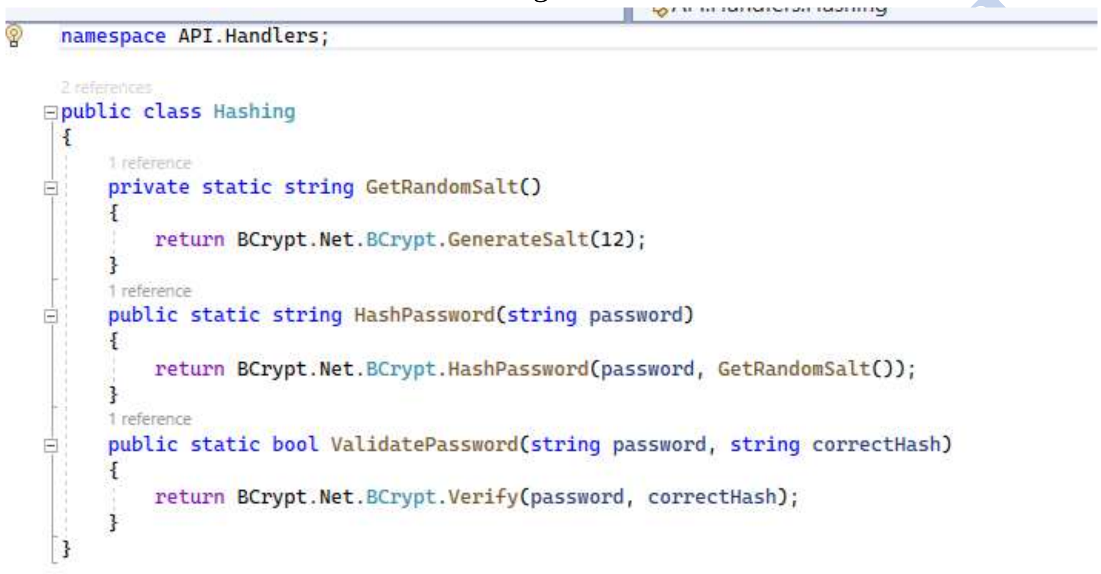


```
2 references
public bool Login(LoginVM loginVM)
{
    // ambil data dari database berdasarkan email di tabel employee
    // gabungkan dari data tabel employee dengan tabel account berdasarkan NIK
    // cocokan data tersebut dengan password yang diinputkan
    // cek apakah data falid atau tidak

    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);
}
}
```

3. Hashing

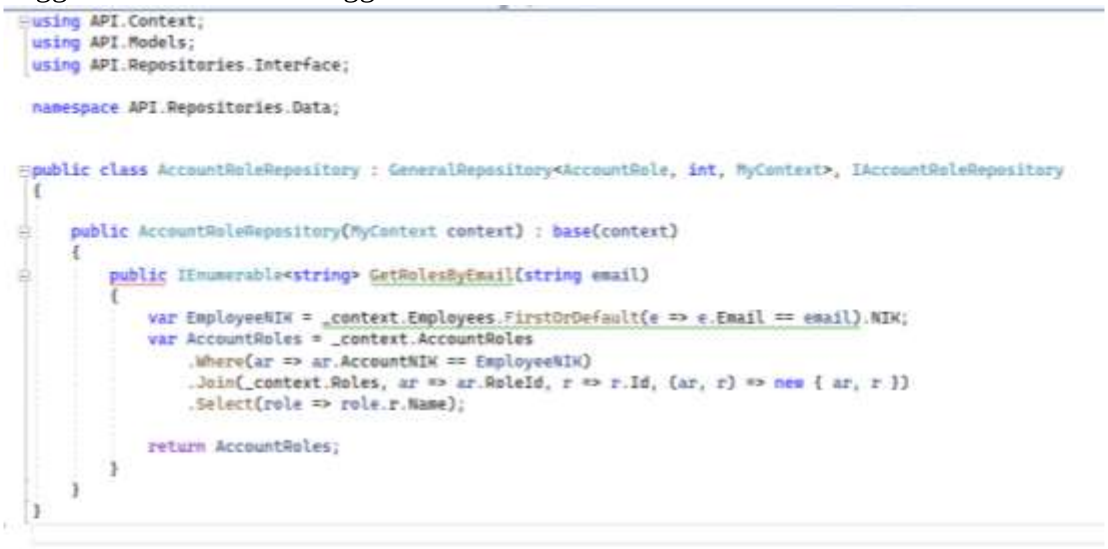
Penjelasan :

Hashing berfungsi untuk mengcrypte password menggunakan Bcrypt. Dalam method GetRandomSalt() berfungsi untuk menambahkan sejumlah 12 digit random salt pada password yang ingin di hash.
Method HashPassword() digunakan untuk mengcrypte password yang telah dimasukkan lalu ditambahkan dengan 12 digit dari GetRandomSalt().
Method ValidatePassword() digunakan untuk mengecek apakah password yang dimasukkan sudah sesuai atau belum dengan database.



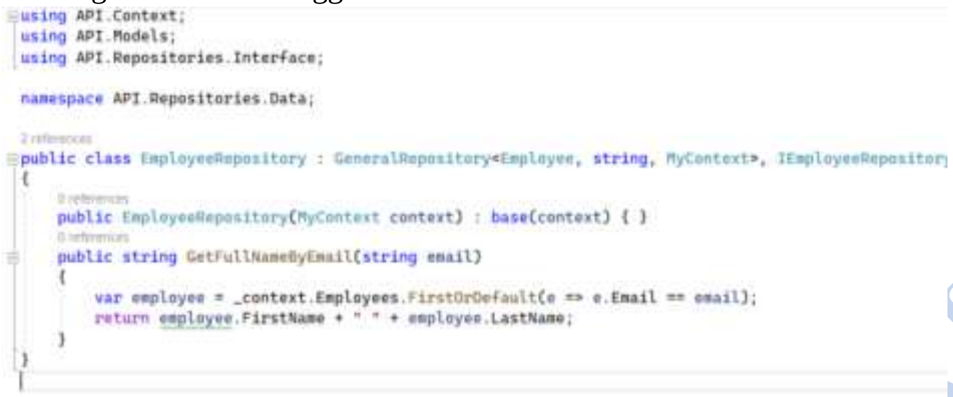
```
namespace API.Handlers;

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

4. **AccountRoleRepository**
   Penjelasan:
   Diggunakan untuk memanggil nama role berdasarkan email.



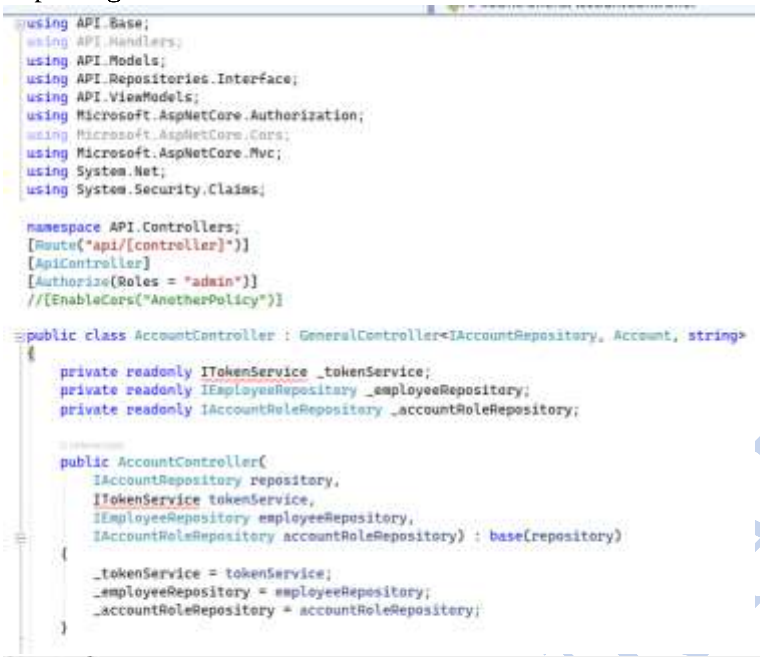
```
using API.Context;
using API.Models;
using API.Repositories.Interface;

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;
        }
    }
}
```

5. **EmployeeRepository**
   Penjelasan:
   Berfungsi untuk memanggil nama awal dan akhir bedasarkan email.

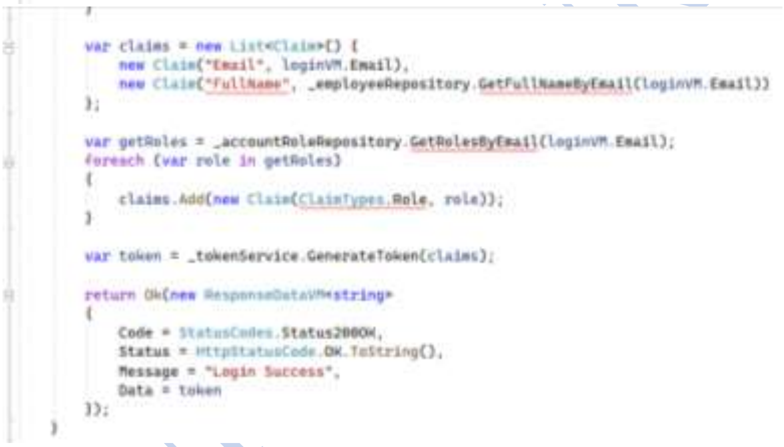


```
using API.Context;
using API.Models;
using API.Repositories.Interface;

namespace API.Repositories.Data;

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

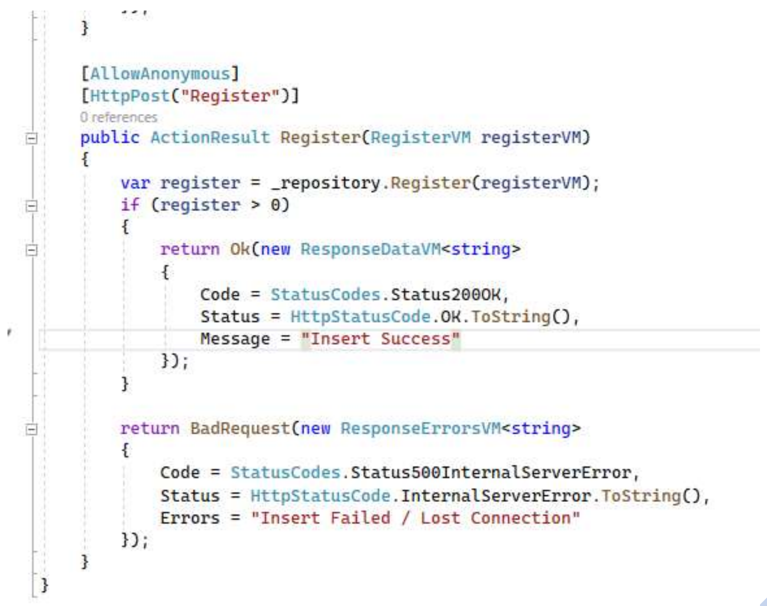
6. **AccountContriller**
Pejelasan:
Method login dan register ditambahkan anotasi AllowAnonymous sehingga user dapat login



```
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;
    }
```

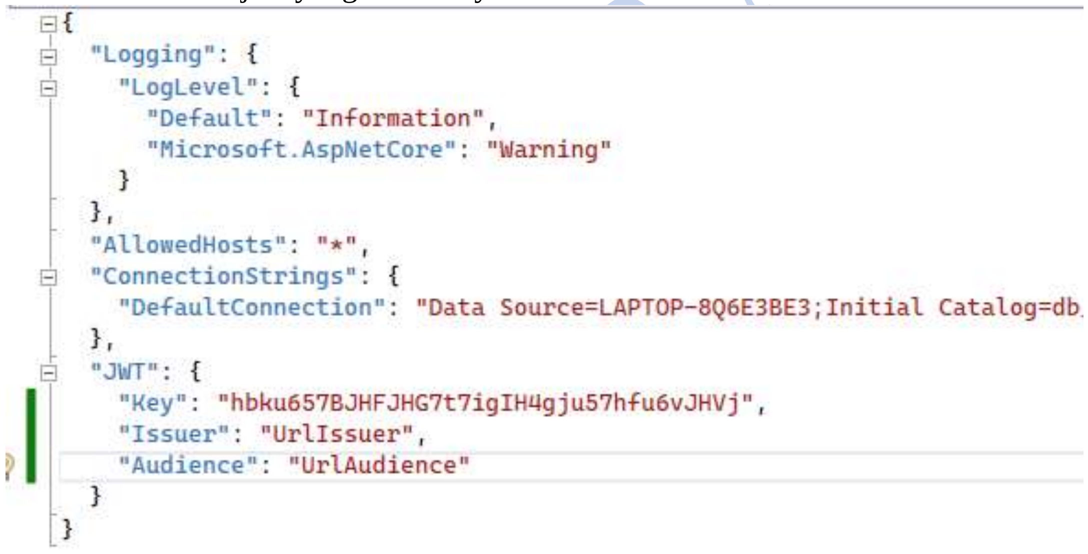


```
        var claims = new List<Claim>() {
            new Claim("Email", loginVM.Email),
            new Claim("FullName", _employeeRepository.GetFullNameByEmail(loginVM.Email))
        };

        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")]
        0 references
        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 ResponseErrorsVM<string>
            {
                Code = StatusCodes.Status500InternalServerError,
                Status = HttpStatusCode.InternalServerError.ToString(),
                Errors = "Insert Failed / Lost Connection"
            });
        }
    }
```

7. **AppSetting**
   Penjelasan:
   Di tambahakan object yang berisi key issuer dan audience

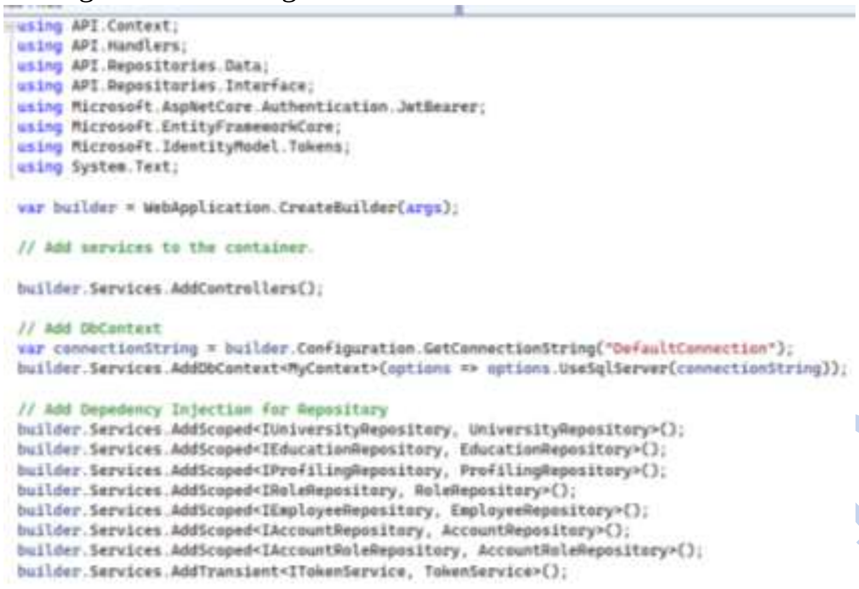


```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "DefaultConnection": "Data Source=LAPTOP-8Q6E3BE3;Initial Catalog=db
  },
  "JWT": {
    "Key": "hbku657BJHFJHG7t7igIH4gju57hfu6vJHVj",
    "Issuer": "UrlIssuer",
    "Audience": "UrlAudience"
  }
}
```

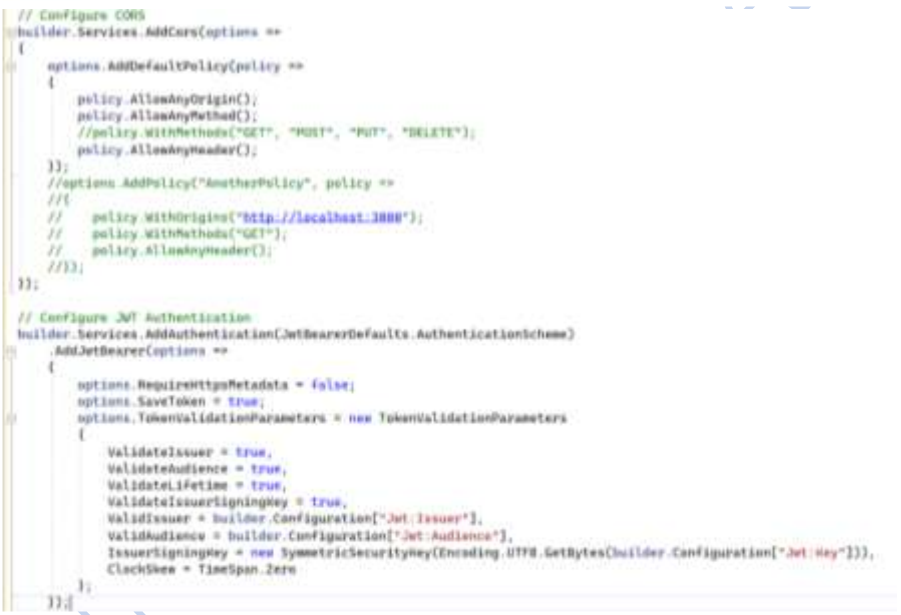
8. Program

Penjelasan:

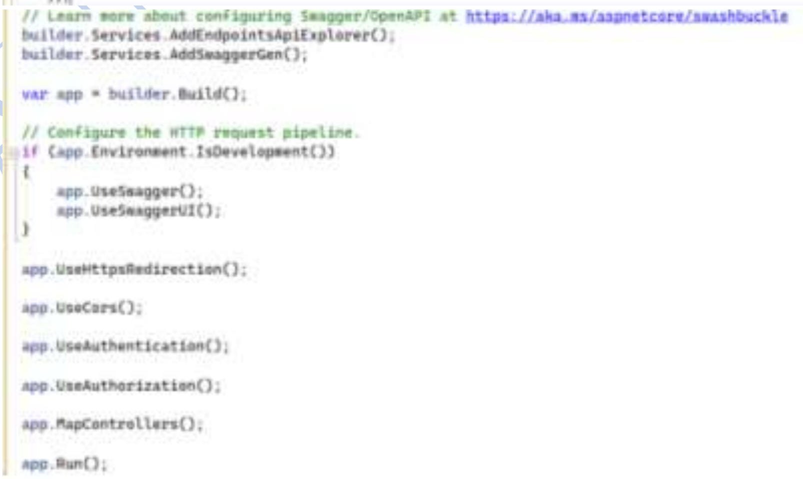
Konfigurasi JWT dengan API



```
using API.Context;
using API.Handlers;
using API.Repositories.Data;
using API.Repositories.Interface;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.EntityFrameworkCore;
using Microsoft.IdentityModel.Tokens;
using System.Text;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();

// Add DbContext
var connectionString = builder.Configuration.GetConnectionString("DefaultConnection");
builder.Services.AddDbContext<MyContext>(options => options.UseSqlServer(connectionString));

// Add Depedency Injection for Repositary
builder.Services.AddScoped<IUniversityRepository, UniversityRepository>();
builder.Services.AddScoped<IEducationRepository, EducationRepository>();
builder.Services.AddScoped<IProfilingRepository, ProfilingRepository>();
builder.Services.AddScoped<IRoleRepository, RoleRepository>();
builder.Services.AddScoped<IEmployeeRepository, EmployeeRepository>();
builder.Services.AddScoped<IAccountRepository, AccountRepository>();
builder.Services.AddScoped<IAccountRoleRepository, AccountRoleRepository>();
builder.Services.AddTransient<ITokenService, TokenService>();
```



```
// Configure CORS
builder.Services.AddCors(options =>
{
    options.AddDefaultPolicy(policy =>
    {
        policy.AllowAnyOrigin();
        policy.AllowAnyMethod();
        //policy.WithMethods("GET", "POST", "PUT", "DELETE");
        policy.AllowAnyHeader();
    });
    //options.AddPolicy("AnotherPolicy", policy =>
    //{
    //    policy.WithOrigins("http://localhost:3000");
    //    policy.WithMethods("GET");
    //    policy.AllowAnyHeader();
    //});
});

// Configure JWT Authentication
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();

app.UseAuthorization();

app.MapControllers();

app.Run();
```