Protegiendo API con JWT NET CORE

Capa de Entity

```
using System.ComponentModel.DataAnnotations;

namespace Entity
{
    public class User
    {
        [Key]
        public string UserName { get; set; }
        public string Password { get; set; }
        public string Estado { get; set; }
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public string Email { get; set; }
        public string MobilePhone { get; set; }
    }
}
```

Capa Datos: Agregar DbSet User a Contexto

```
using Entity;
using Microsoft.EntityFrameworkCore;

namespace Datos
{
    public class PulsacionesContext : DbContext
    {
        public PulsacionesContext(DbContextOptions options) : base(options)
        {
        }
        public DbSet<Persona> Personas { get; set; }

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

Capa de Lógica: Agregar clase UserService

```
using Datos;
using Entity;
using System.Linq;

namespace Logica
{
    public class UserService
    {
        private readonly PulsacionesContext _context;

        public UserService(PulsacionesContext context)=> _context = context;

        public User Validate(string userName, string password)
        {
            return _context.Users.FirstOrDefault(t => t.UserName == userName && t.Password == password && t.Estado == "AC");
        }
    }
}
```

Capa de Web

- Agregar Nuget Microsoft.AspNetCore.Authentication.JwtBearer y System.IdentityModel.Tokens.Jw
- Modificación del AppSetting.json
- Agregar clase Config/AppSetting
- StartUp. Configurar Inyección de dependencia para leer el AppSetting.Secret
- StartUp.Configure, se habilita la Autenticación, Autorización y CORS
- Implementar Service/JwtService.cs para crear el TOKEN (JWT)
- Se crea el LoginController

AppSetting: Agregar AppSetting.Secret

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "DefaultConnection": "Server=.;Database=PulsacionesEf;Trusted_Connection = True; MultipleActiveResultSets = true"
  },
  "AppSetting": {
    "Secret": "THIS IS USED TO SIGN AND VERIFY JWT TOKENS, REPLACE IT WITH YOUR OWN SECRET, IT CAN BE ANY STRING"
  }
}
```

Agregar Nuget

```
<ItemGroup>
    <PackageReference Include="Microsoft.AspNetCore.Authentication.JwtBearer" Version="3.0.0" />

   <PackageReference Include="Microsoft.EntityFrameworkCore.Tools" Version="3.1.3">

  <PrivateAssets>all</PrivateAssets>
  <IncludeAssets>runtime; build; native; contentfiles; analyzers</IncludeAssets>
   </PackageReference>
    <PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer" Version="3.0.0" />
    <PackageReference Include="Microsoft.AspNetCore.SpaServices.Extensions" Version="3.0.0" />
    <PackageReference Include="Microsoft.Extensions.Logging.Debug" Version="3.0.0" />
    <PackageReference Include="Microsoft.VisualStudio.Web.CodeGeneration.Design" Version="3.0.0" />
    <PackageReference Include="Swashbuckle.AspNetCore" Version="5.0.0" />
    <PackageReference Include="System.IdentityModel.Tokens.Jwt" Version="6.6.0" />

  </ItemGroup>
```

Agregar clase Config/AppSetting para Obtener valores del AppSetting.json

Cree una Carpeta Config y cree una clase AppSetting

```
namespace WebPulsaciones.Config
{
    public class AppSetting
    {
        public string Secret { get; set; }
    }
}
```

StartUp. Configurar Inyección de dependencia para leer el AppSetting.Secret

```
public void ConfigureServices(IServiceCollection services)
{
        ///antecede configuración de DbContext
        services.AddControllersWithViews();

        #region    configure strongly typed settings objects
        var appSettingsSection = Configuration.GetSection("AppSetting");
        services.Configure<AppSetting>(appSettingsSection);
        #endregion

        #region Configure jwt authentication inteprete el token
        var appSettings = appSettingsSection.Get<AppSetting>();
        var key = Encoding.ASCII.GetBytes(appSettings.Secret);
        services.AddAuthentication(x =>
        {
            x.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
            x.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
        })
        .AddJwtBearer(x =>
        {
            x.RequireHttpsMetadata = false;
            x.SaveToken = true;
            x.TokenValidationParameters = new TokenValidationParameters
            {
                ValidateIssuerSigningKey = true,
                IssuerSigningKey = new SymmetricSecurityKey(key),
                ValidateIssuer = false,
                ValidateAudience = false
            };
        });
        #endregion
        ///sigue código de configuración de swagger
```

StartUp.Configure, se habilita la Autenticación, Autorización y CORS

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
            //Codigo suprimido por facilitar la visualización del nuevo código

            app.UseRouting();

            #region global cors policy activate Authentication/Authorization
            app.UseCors(x => x
                .AllowAnyOrigin()
                .AllowAnyMethod()
                .AllowAnyHeader());

            app.UseAuthentication();
            app.UseAuthorization();
            #endregion

            //Codigo suprimido por facilitar la visualización del nuevo código
```

Implementar Service/JwtService.cs para crear el TOKEN (JWT)

```
using Entity;
using Microsoft.Extensions.Options;
using Microsoft.IdentityModel.Tokens;
using System;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Text;
using WebPulsaciones.Config;
using WebPulsaciones.Models;

namespace WebPulsaciones
{
    public class JwtService
    {
        private readonly AppSetting _appSettings;
        public JwtService(IOptions<AppSetting> appSettings)=> _appSettings = appSettings.Value;
        public LoginViewModel GenerateToken(User userLogIn)
        {
            // return null if user not found
            if (userLogIn == null) return null;
            var userResponse = new LoginViewModel() { FirstName = userLogIn.FirstName, LastName = userLogIn.LastName, Username = userLogIn.UserName };
            // authentication successful so generate jwt token
            var tokenHandler = new JwtSecurityTokenHandler();
            var key = Encoding.ASCII.GetBytes(_appSettings.Secret);
            var tokenDescriptor = new SecurityTokenDescriptor
            {
                Subject = new ClaimsIdentity(new Claim[]
                {
                    new Claim(ClaimTypes.Name, userLogIn.UserName.ToString()),
                    new Claim(ClaimTypes.Email, userLogIn.Email.ToString()),
                    new Claim(ClaimTypes.MobilePhone, userLogIn.MobilePhone.ToString()),
                    new Claim(ClaimTypes.Role, "Rol1"),
                    new Claim(ClaimTypes.Role, "Rol2"),
                }),
                Expires = DateTime.UtcNow.AddDays(7),
                SigningCredentials = new SigningCredentials(new SymmetricSecurityKey(key), SecurityAlgorithms.HmacSha256Signature)
            };
            var token = tokenHandler.CreateToken(tokenDescriptor);
            userResponse.Token = tokenHandler.WriteToken(token);
            return userResponse;
        }
    }
}
```

LoginController

```
1   using Datos;
2   using Logica;
3   using Microsoft.AspNetCore.Authorization;
4   using Microsoft.AspNetCore.Mvc;
5   using Microsoft.Extensions.Options;
6   using WebPulsaciones.Config;
7   using WebPulsaciones.Models;
8
9   namespace WebPulsaciones.Controllers
10  {
11      [Authorize]
12      [ApiController]
13      [Route("api/[controller]")]
            1 referencia
14      public class LoginController : ControllerBase
15      {
16          PulsacionesContext _context;
17          UserService _userService;
18          JwtService _jwtService;
                0 referencias
19          public LoginController(PulsacionesContext context, IOptions<AppSetting> appSettings)...
31
32          [AllowAnonymous]
33          [HttpPost("login")]
                0 referencias
34          public IActionResult Login([FromBody]LoginInputModel model)...
41      }
42  }
43
```

LoginController Parte1

```
public LoginController(PulsacionesContext context, IOptions<AppSetting> appSettings)
{
      _context = context;
       var admin = _context.Users.Find("admin");
      if (admin == null)
      {
            _context.Users.Add(new User()
            {
                     UserName="admin",
                      Password="admin",
                      Email="admin@gmail.com",
                     Estado="AC",
                     FirstName="Adminitrador",
                     LastName="",
                     MobilePhone="31800000000"}
            );
            var registrosGuardados=_context.SaveChanges();
      }
      _userService = new UserService(context);
      _jwtService = new JwtService(appSettings);
}
```

LoginController Parte2

```
[AllowAnonymous]
[HttpPost]
public IActionResult Login([FromBody]LoginInputModel model)
{
    var user = _userService.Validate(model.Username, model.Password);
    if (user == null) return BadRequest("Username or password is incorrect");
    var response= _jwtService.GenerateToken(user);
    return Ok(response);
}
```