

▼ Desafio 4 JWT

Desafio 4 JWT

Proposta:

- ✓ **Iremos utilizar VaotacaoApi como base de estudo.**
- ✓ **Nela iremos adicionar uma autenticação Jwt.**




Desafio 4 JWT

Passo a passo:

- ✓ **Adicionar pacotes nuget.**
- ✓ **Configurar nossa chave.**
- ✓ **Criar nossa classe que irá gerar os tokens.**
- ✓ **Configurar a autenticação.**
- ✓ **Configurar o swagger.**
- ✓ **Adicionar anotações nas controllers.**
- ✓ **Configurar o retorno do endpoint de login.**

Desafio 4 JWT

Adicionar pacotes nuget:

	Microsoft.AspNetCore.Authentication.JwtBearer por Microsoft	v3.1.11
	ASP.NET Core middleware that enables an application to receive an OpenID Connect bearer token.	v5.0.2
	Swashbuckle.AspNetCore por Swashbuckle.AspNetCore	v5.6.3
	Swagger tools for documenting APIs built on ASP.NET Core	v6.0.0
	System.IdentityModel.Tokens.Jwt por Microsoft	v6.8.0 ✖
	Includes types that provide support for creating, serializing and validating JSON Web Tokens.	

Desafio 4 JWT

Atualizar nosso appSettings.json:

```
ALLOWED_HOSTS = [...],  
"SettingsAPI": {  
    "ChaveJWT": "(B3]U5N{ho+KdXxB%>Q,Fblh9E;0:NE*08!ke?b@eM,7kk8Ph{DnRp+}u!Hs?LbE+CpP[*}X^&^fR4w0u0E$H{621%Mr[nw#qC}"  
},
```


Desafio 4 JWT

Criar nossa classe para geração de tokens:

```
5 referências
public class TokenJWTService
{
    private readonly string _ChaveJWT;

    0 referências
    public TokenJWTService(IConfiguration configuration)
    {
        _ChaveJWT = configuration.GetSection("SettingsAPI:ChaveJWT").Get<string>();
    }

    1 referência
    public string GenerateToken(UsuarioQueryResult usuarioQR)
    {
        var tokenHandler = new JwtSecurityTokenHandler();
        var key = Encoding.ASCII.GetBytes(_ChaveJWT);

        var tokenDescriptor = new SecurityTokenDescriptor
        {
            Subject = new ClaimsIdentity(new Claim[]
            {
                new Claim(ClaimTypes.Name, usuarioQR.Login),
                new Claim(ClaimTypes.Role, "User")
            }),
            Expires = DateTime.UtcNow.AddHours(2),
            SigningCredentials = new SigningCredentials(new SymmetricSecurityKey(key), SecurityAlgorithms.HmacSha256Signature)
        };

        var token = tokenHandler.CreateToken(tokenDescriptor);
        return tokenHandler.WriteToken(token);
    }
}
```

Desafio 4 JWT

Configurar a autenticação no arquivo Startup.cs:

```
#region [+] Autenticação JWT

var keyString = Configuration.GetSection("SettingsAPI:ChaveJWT").Get<string>();
var key = Encoding.ASCII.GetBytes(keyString);

services.AddAuthentication(x =>
{
    x.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
    x.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
}).AddJwtBearer(x =>
{
    x.RequireHttpsMetadata = true;
    x.SaveToken = true;
    x.TokenValidationParameters = new TokenValidationParameters
    {
        ValidateIssuerSigningKey = true,
        IssuerSigningKey = new SymmetricSecurityKey(key),
        ValidateIssuer = false,
        ValidateAudience = false
    };
});

#endregion
```

Desafio 4 JWT

Atualizar a configuração do swagger no Startup.cs:

```
c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme
{
    Description = "Para autenticar use a palavra 'Bearer' + (um espaço entre a palavra Bearer e o Token) + 'Token'",
    Name = "Authorization",
    In = ParameterLocation.Header,
    Type = SecuritySchemeType.ApiKey
});

var scheme = new OpenApiSecurityScheme { Reference = new OpenApiReference { Type = ReferenceType.SecurityScheme, Id = "bearer" } };
c.AddSecurityRequirement(new OpenApiSecurityRequirement
{
    [scheme] = new List<string>()
});
```


Desafio 4 JWT

Adicionar anotações nas controllers:

```
[Consumes("application/json")]
[Produces("application/json")]
[ApiController]
[Authorize]
1 referência
public class FilmeController : ControllerBase
{
```

```
[Consumes("application/json")]
[Produces("application/json")]
[ApiController]
[Authorize]
1 referência
public class UsuarioController : ControllerBase
{
```

```
[Consumes("application/json")]
[Produces("application/json")]
[ApiController]
[Authorize]
1 referência
public class VotoController : ControllerBase
{
```

```
// [HttpPost]
[AllowAnonymous]
[HttpPost]
[Route("v1/usuarios/login")]
0 referências
public ICommandResult UsuarioLogin([FromBody] LogarUsuarioCommand command)...
```

Desafio 4 JWT

Configurar o retorno do endpoint de login:

```
[AllowAnonymous]
[HttpPost]
[Route("v1/usuarios/login")]
0 referências
public ICommandResult UsuarioLogin([FromBody] LogarUsuarioCommand command)
{
    try
    {
        var result = _handler.Handle(command);

        var tokenJWT = _tokenJWTService.GenerateToken((UsuarioQueryResult)result.Data);

        result.Data = tokenJWT;

        return result;
    }
    catch (Exception ex)
    {
        throw ex;
    }
}
```




Dúvidas?