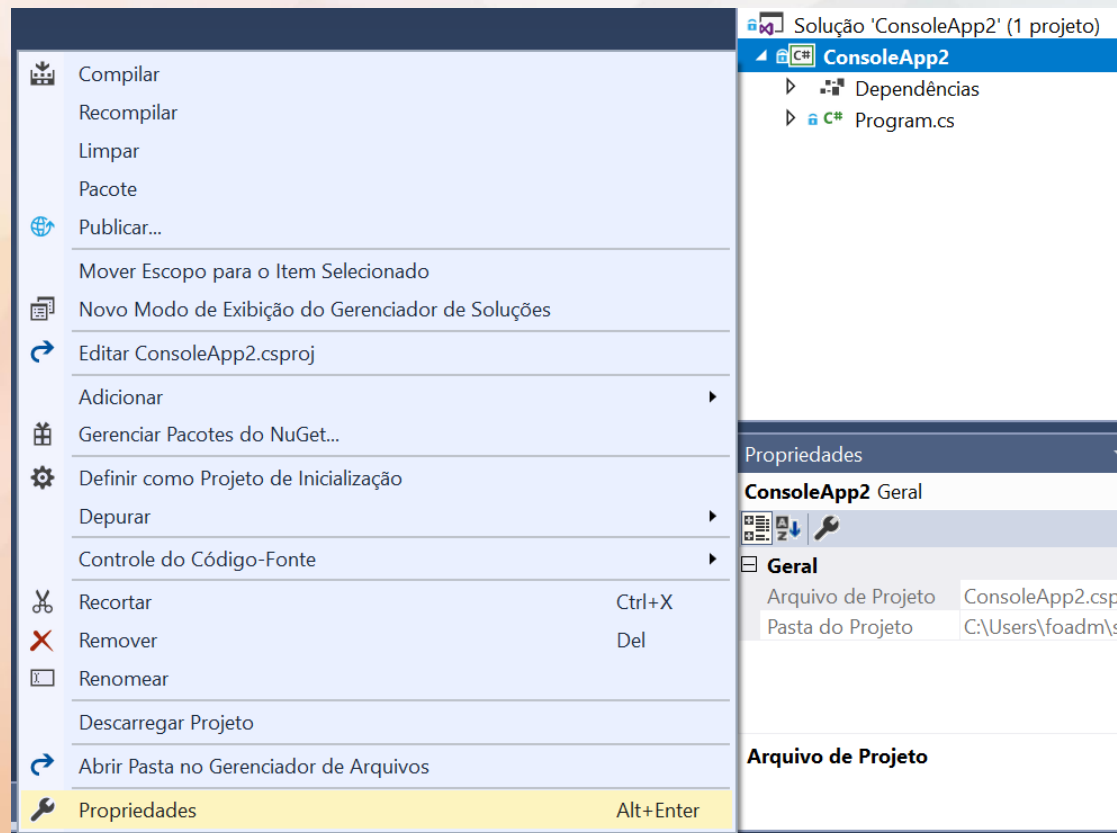


# Projeto de Banco de Dados e OO .NET

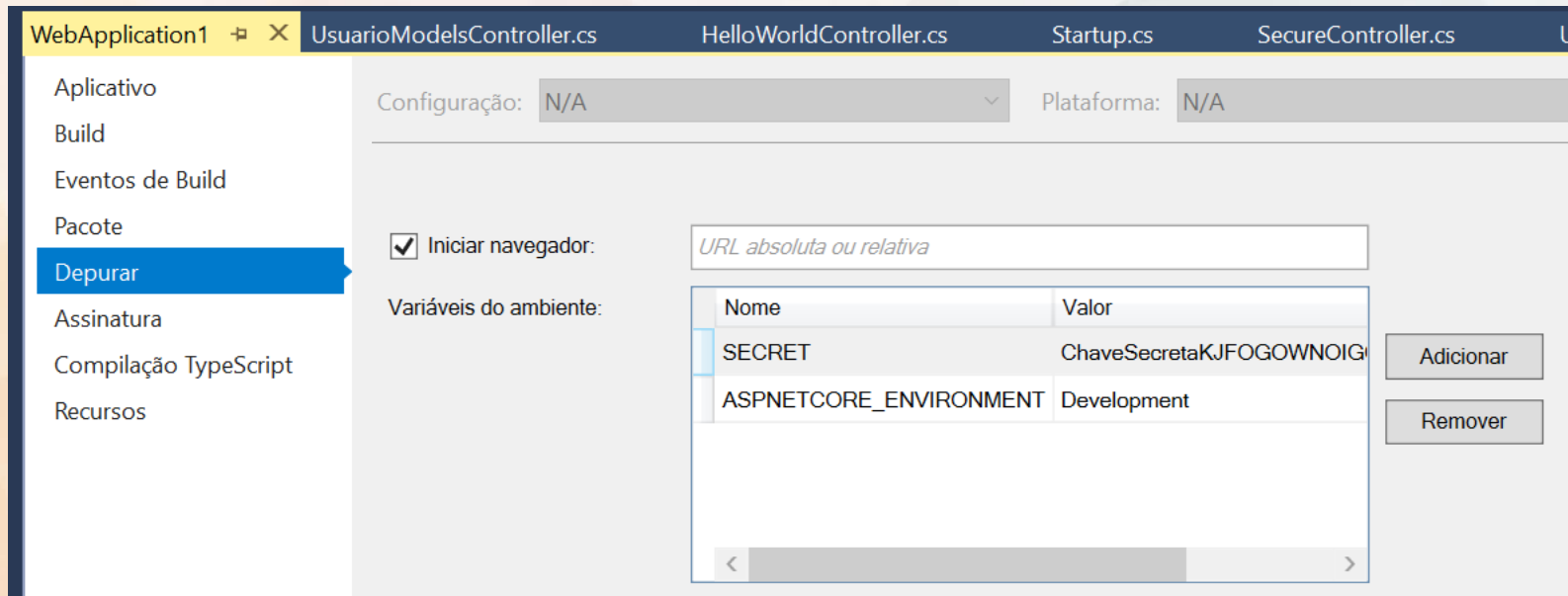
Web API com Autenticação por Token JWT em ASP.NET

# Melhorando nossa API em C#

- Abra a o MVC CRUD Api que foi feito na aula 4
- Entre nas propriedades



# Defina uma variável de ambiente chamada SECRET



```
Environment.GetEnvironmentVariable("SECRET")
```

# No arquivo Startup.cs

Adicione o serviço de validação do Token no método ConfigureServices

0 referências | foadm, 1 dia atrás | 1 autor, 1 alteração

```
public void ConfigureServices(IServiceCollection services)
```

```
{
    services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)
        .AddJwtBearer(options =>
        {
            options.TokenValidationParameters = new TokenValidationParameters
            {
                ValidateIssuer = true,
                ValidateAudience = true,
                ValidateLifetime = true,
                ValidateIssuerSigningKey = true,
                ValidIssuer = "yourdomain.com",
                ValidAudience = "yourdomain.com",
                IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(Environment.GetEnvironmentVariable("SECRET")))
            };
        });
};
```

No Configure adicione o comando app.UseAuthentication()

0 referências | foadm, 1 dia atrás | 1 autor, 1 alteração

```
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
```

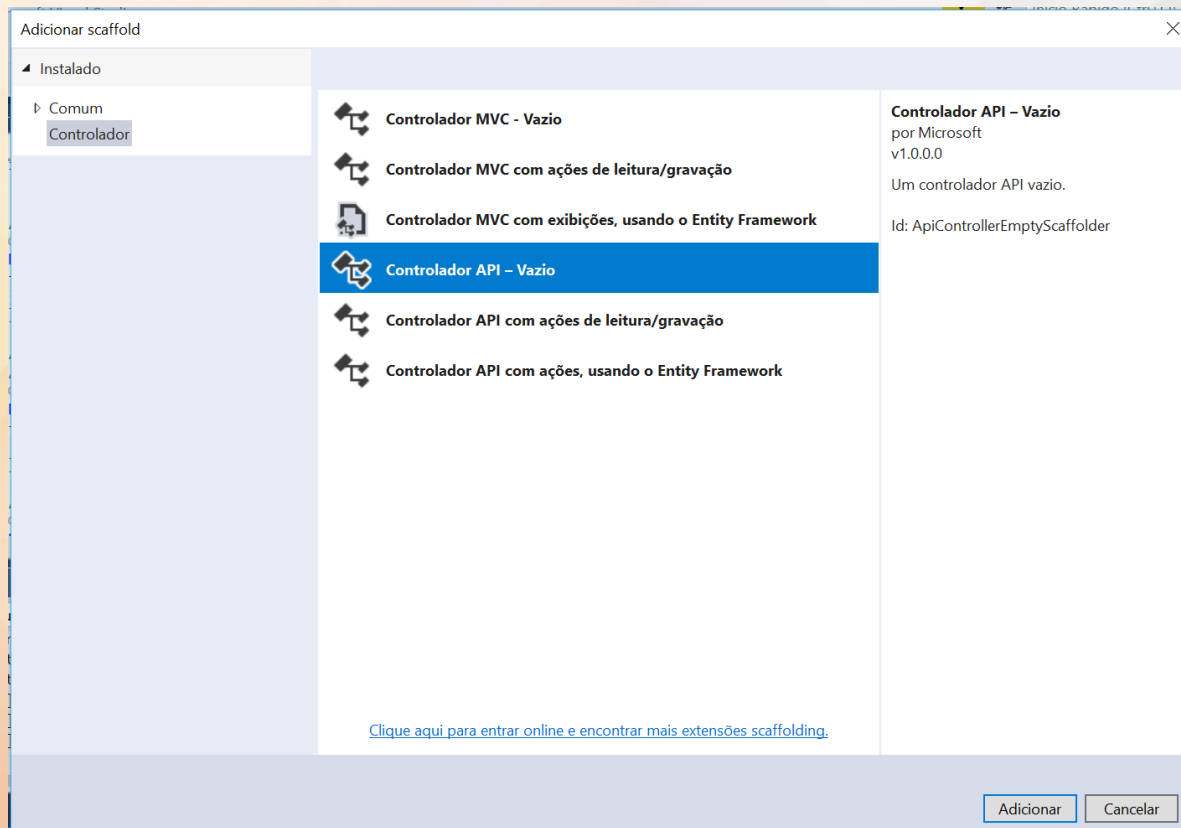
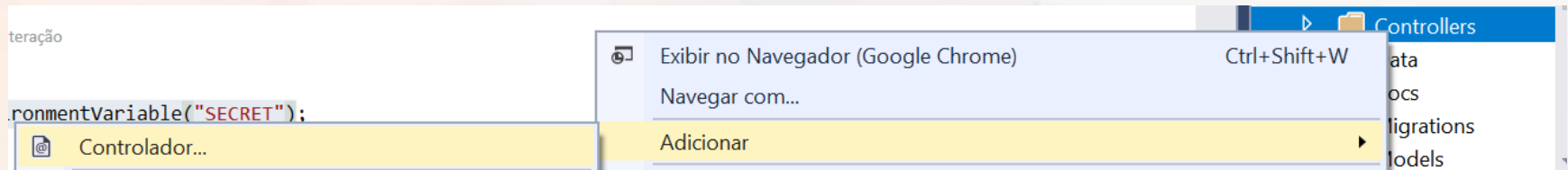
```
{
    app.UseAuthentication();
};
```

# Adicione um classe simples na pasta Controllers

```
namespace WebApplication1.Controllers
{
    1 referência | 0 alterações | 0 autores, 0 alterações
    public class TokenRequest
    {
        1 referência | 0 alterações | 0 autores, 0 alterações
        public string Username { get; set; }
        1 referência | 0 alterações | 0 autores, 0 alterações
        public string Password { get; set; }
    }
}
```

Essa classe será usada como base para os parâmetros que serão passados no login

# Adicione um Controlador de API Vazio



Crie o controlador com nome  
SecureController

# Controlador Seguro

Adicione a diretiz [Authorize]

```
[Authorize]  
[Produces("application/json")]  
[Route("api/Secure")]
```

0 referências | 0 alterações | 0 autores, 0 alterações

```
public class SecureController : Controller
```

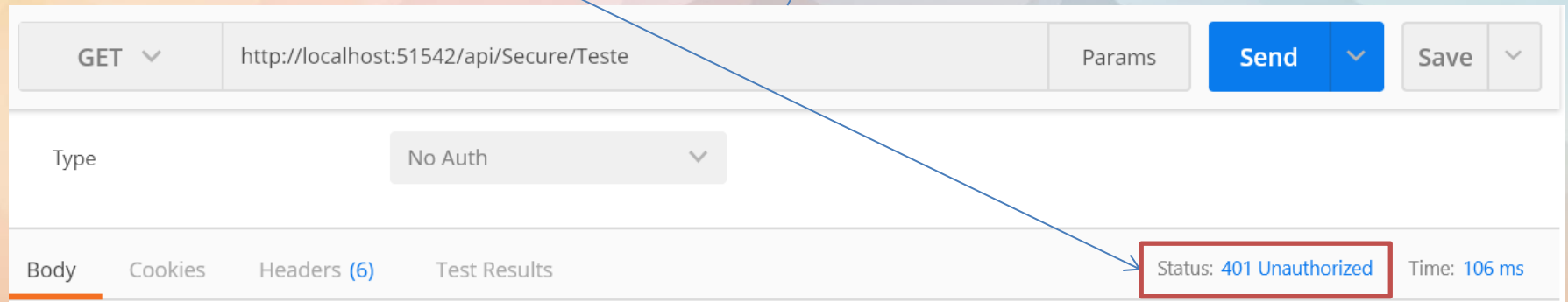
# Controlador Seguro

```
[HttpGet("Teste")]  
0 referências | 0 alterações | 0 autores, 0 alterações  
public IActionResult Teste()  
{  
    return Ok($"Conteúdo super secreto");  
}
```

Crie uma Action Teste

Teste o novo controlador

Esse software utilizado aqui  
é o postman <https://www.getpostman.com>





# Controlador Seguro

```
private readonly MeuContext _context;
```

0 referências | 0 alterações | 0 autores, 0 alterações

```
public SecureController(MeuContext context)
{
    _context = context;
}
```

Adicionar o context no controlador

# Controlador Seguro

→ Habilita o acesso sem token

```
[AllowAnonymous]
[HttpPost("Login")]
0 referências | 0 alterações | 0 autores, 0 alterações
public async Task<IActionResult> Login([FromBody] TokenRequest request)
{
```

```
    var usuarioModel = await _context.UsuarioModel
        .SingleOrDefaultAsync(m => m.UserName == request.Username);
    if (usuarioModel == null)
    {
        return BadRequest("Usuario nao encontrado.");
    }
```

```
    string senha = HashCalculator.GenerateMD5(request.Password);
```

```
    if (senha == usuarioModel.Senha)
    {
```

```
        var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(Environment.GetEnvironmentVariable("SECRET")));
        var creds = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);
```

```
        var token = new JwtSecurityToken(
            issuer: "yourdomain.com",
            audience: "yourdomain.com",
            expires: DateTime.Now.AddMinutes(30),
            signingCredentials: creds);
```

```
        return Ok(new
        {
            token = new JwtSecurityTokenHandler().WriteToken(token)
        });
    }
```

```
    return BadRequest("Usuario e senha incorretos.");
}
```

Recupera os dados do usuário

Verifica se a senha está correta

→ Gera e retorna um token JWT

# Fazendo um Teste

POST ▾

http://localhost:51542/api/Secure/Login

Params

Send ▾

Save ▾

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

JSON (application/json) ▾

1 ▾ {

2   "Username": "foadm",

3   "Password": "12345"

4 }|

Body

Cookies

Headers (6)

Test Results

Status: 200 OK

Time: 2619 ms

Pretty

Raw

Preview

JSON ▾



Save Response

1 ▾ {

2   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.  
          .eyJleHAiOiE1MjM0MTkwNTgsImZscyI6InlvdXJkb21haw4uY29tIiwiaXVkiOiJoiew91cmRvbWVpbi5jb20ifQ.  
          .sr\_JgDBTuG6xZ1IL0KQvUpmAiY7QTWBLPznXpBsfmt"

3 }|

# Token Gerado

## Encoded

PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE1MjM0MTkwNTgsImZcyI6InlvdXJkb21haW4uY29tIiwiaXVkiOiieW91cmRvbWFPbi5jb20ifQ.sr_JgDBTuG6xZ1IL0KQvUpmAiIY7QTWBLPznXpBsfmt
```

## Decoded

EDIT THE PAYLOAD AND SECRET

### HEADER: ALGORITHM & TOKEN TYPE

```
{  "alg": "HS256",  "typ": "JWT"}
```

### PAYLOAD: DATA

```
{  "exp": 1523419058,  "iss": "yourdomain.com",  "aud": "yourdomain.com"}
```

# Acessando Action Protegida




► http://localhost:51542/api/Secure/Test Examples (0) ▼

GET ▼ http://localhost:51542/api/Secure/Teste Params **Send** ▼ Save ▼

Authorization Headers (1) Body Pre-request Script Tests Code

	Key	Value	Description	...	Bulk Edit	Presets ▼
<input checked="" type="checkbox"/>	Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOi...				
	New key	Value	Description			

Body Cookies Headers (6) Test Results Status: 200 OK Time: 135 ms

Pretty Raw Preview JSON ▼    Save Response

1 "Conteudo super secreto"

# Fontes

- <https://jwt.io/>
- <https://github.com/jonhilt/NetCoreAuth>
- <https://jonhilton.net/2017/10/11/secure-your-asp.net-core-2.0-api-part-1---issuing-a-jwt/>