Residência TCE/RN: Desenvolvimento de soluções Web Aula 05

Professor: José Alex

Pesquisador e Gerente de Projetos
do Smart Metropolis

UFRN UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

IMD INSTITUTO METRÓPOLE DIGITAL

Agenda

- 05/07/2019
  - HTML 5, CSS e JavaScript
- 12/07/2019
  - Ajax, Jquery e Angular
- 19/07/2019
  - Arquitetura de serviços (SOA) e Web Services SOAP, XML e XSLT
- 26/07/2019
  - Serviços REST, C#, ASP.NET Core (API Web) e Swagger
- 02/08/2019
  - Dúvidas aula 04 e Segurança

Dúvidas aula 04

Dúvidas aula 04

- Convenção de nomenclatura no C#:
  - PascalCase => Classes, Interfaces, Enums, Métodos, Propriedades, Campos públicos, Eventos;
  - camelCase => Variáveis locais, Parâmetros de métodos, Campos privados e protegidos.

- ASP.NET Core e SQL Server:
  - Pacote **Microsoft.EntityFrameworkCore.SqlServer;**
  - String de conexão no *appsettings.json*;
  - Modificar classe *Startup.cs;*

```
{
  "ConnectionStrings": {
    "BaseExemplo": "Data Source=.\\MSSQLSERVER2016;Initial Catalog=Exemplo;User ID=SA;Password=teste;"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Warning"
    }
  },
  "AllowedHosts": "*"
}
```

Dúvidas aula 04

```
public void ConfigureServices(IServiceCollection services)
{
    /* services.AddDbContext<ExemploWebApiContext>(opt =>
        opt.UseInMemoryDatabase("ExemploWebApiList"));*/

    services.AddEntityFrameworkSqlServer()
        .AddDbContext<ExemploWebApiContext>(
            opt => opt.UseSqlServer(
                Configuration.GetConnectionString("BaseExemplo")));

    services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_2);

    // Register the Swagger generator, defining 1 or more Swagger documents
    services.AddSwaggerGen(c =>
    {
        c.SwaggerDoc("v1", new OpenApiInfo { Title = "Minha API", Version = "v1" });
    });
}
```

- ASP.NET Core e PostgreSQL:
  - Pacote **Npgsql.EntityFrameworkCore.PostgreSQL;**
  - String de conexão no *appsettings.json*;
  - Modificar classe *Startup.cs;*

Dúvidas aula 04

```
{

  "ConnectionStrings": {

    "BaseExemplo": "Server=localhost;Database=Exemplo;Port=5432;User Id=postgres;Password=teste;"

  },

  "Logging": {
    "LogLevel": {
      "Default": "Warning"
    }
  },
  "AllowedHosts": "*"
}
```

```
using ExemploWebApi.Models;
using Microsoft.OpenApi.Models;
using Npgsql.EntityFrameworkCore.PostgreSQL;
```

```
public void ConfigureServices(IServiceCollection services)
{
    /* services.AddDbContext<ExemploWebApiContext>(opt =>
        opt.UseInMemoryDatabase("ExemploWebApiList"));*/

    /* services.AddEntityFrameworkSqlServer()
        .AddDbContext<ExemploWebApiContext>(
            opt => opt.UseSqlServer(
                Configuration.GetConnectionString("BaseExemplo")));*/

    services.AddEntityFrameworkNpgsql()
        .AddDbContext<ExemploWebApiContext>(
            opt => opt.UseNpgsql(
                Configuration.GetConnectionString("BaseExemplo")));
```

Dúvidas aula 04

- ASP.NET Core e MySql:
  - Pacote **Pomelo.EntityFrameworkCore.MySql;**
  - String de conexão no *appsettings.json*;
  - Modificar classe *Startup.cs;*

```
{

  "ConnectionStrings": {

    "BaseExemplo": "server=localhost;userid=teste;password=teste;database=Exemplo;"

  },

  "Logging": {
    "LogLevel": {
      "Default": "Warning"
    }
  },
  "AllowedHosts": "*"
}
```

Dúvidas aula 04

```
using ExemploWebApi.Models;
using Microsoft.OpenApi.Models;
//using Npgsql.EntityFrameworkCore.PostgreSQL;
using Pomelo.EntityFrameworkCore.MySql;
```

```
public void ConfigureServices(IServiceCollection services)
{
    /* services.AddDbContext<ExemploWebApiContext>(opt =>
        opt.UseInMemoryDatabase("ExemploWebApiList"));*/

    /* services.AddEntityFrameworkSqlServer()
        .AddDbContext<ExemploWebApiContext>(
            opt => opt.UseSqlServer(
                Configuration.GetConnectionString("BaseExemplo")));*/

    /* services.AddEntityFrameworkNpgsql()
        .AddDbContext<ExemploWebApiContext>(
            opt => opt.UseNpgsql(
                Configuration.GetConnectionString("BaseExemplo")));

    */

    services.AddEntityFrameworkMySql()
        .AddDbContext<ExemploWebApiContext>(
           opt => opt.UseMySql(
               Configuration.GetConnectionString("BaseExemplo")));
```

Dúvidas aula 04

- Migration:
  - Alterar a classe de contexto;

```
using Microsoft.EntityFrameworkCore;

namespace ExemploWebApi.Models
{
    4 references
    public class ExemploWebApiContext : DbContext
    {
        0 references
        public ExemploWebApiContext(DbContextOptions<ExemploWebApiContext> options)
            : base(options)
        {
        }

        7 references
        public DbSet<ExemploWebApiItem> ExemploWebApiItems { get; set; }

        1 reference
        protected override void OnModelCreating(ModelBuilder builder)
        {
            base.OnModelCreating(builder);
        }
    }
}
```

  - Na linha de comando:
    - dotnet ef migrations add nomeMigration --context ExemploWebApiContext
    - dotnet ef database update --context ExemploWebApiContext

Dúvidas? Sugestões?
Comentários...

Segurança

Segurança

- Autenticação: Verifica a identidade de um usuário;
- Autorização: Gerencia o que um usuário pode ou não pode acessar, ou seja, suas permissões;
- Encriptação: Mecanismo que auxilia na proteção dos dados e passa-los entre o cliente e o servidor;
- Solução baseada em tokens (OpenID Connect, OAuth 2.0 e JWT);
- OpenID Connect: habilita uma terceira parte para identificar um usuário;
- OAuth 2.0: habilita uma terceira parte para obter acesso limitado para um serviço HTTP;

Segurança

- JSON Web Tokens (JWT):
  - Formato JSON para a utilização de tokens;
  - Pode ser usado de maneira independente;
  - OpenID Connect recomenda o uso de JWT;
  - São protegidos (criptografados);
  - Contém informações sobre a criptografia, o dado em si e a identidade do usuário;
  - **Bearer Authentication**.

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWV9.TJVA9

```
Header: {"alg": "HS256", "type": "JWT"}

Payload: {"sub": "1234567890","name": "John Doe","admin": true}

Signature:HMACSHA256(base64UrlEncode(header) + "." +     base64UrlEncode(payload),secret)
```

Fonte: Aroraa, G.K; Kale, L.; Manish, K. Building Microservices with .NET Core

Segurança

- Passos da autenticação:
  1. Requisição HTTP POST com usuário e senha;
  2. Token JWT gerado;
  3. Token deve ser guardado pelo solicitante. **OBS: O Token possui validade e todo o processo é stateless;**
  4. Requisições a recursos de APIs REST devem conter o token obtido anteriormente;
  5. Caso o token seja válido, o acesso ao recurso é liberado.

Segurança no ASP.NET Core

- Criar classe de modelo para o usuário:

```
namespace ExemploWebApi.Models
{
    6 references
    public class Usuario
    {
        9 references
        public string ID { get; set; }
        4 references
        public string ChaveAcesso { get; set; }
    }
}
```

Segurança no ASP.NET Core

- No classe de contexto:

```
using Microsoft.EntityFrameworkCore;

namespace ExemploWebApi.Models
{
    6 references
    public class ExemploWebApiContext : DbContext
    {
        0 references
        public ExemploWebApiContext(DbContextOptions<ExemploWebApiContext> options)
            : base(options)
        {
        }
        7 references
        public DbSet<ExemploWebApiItem> ExemploWebApiItems { get; set; }
        2 references
        public DbSet<Usuario> Usuarios { get; set; }
        1 reference
        protected override void OnModelCreating(ModelBuilder builder)
        {
            base.OnModelCreating(builder);
        }
    }
}
```

Segurança no ASP.NET Core

- Criar classe para manipular usuário:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using ExemploWebApi.Controllers;
using ExemploWebApi.Models;

namespace ExemploWebApi.Services
{
    3 references
    public class UsuarioService
    {
        4 references
        private ExemploWebApiContext _context;
        0 references
        public UsuarioService(ExemploWebApiContext context)
        {
            _context = context;
        }
        1 reference
        public Usuario Obter(string id)
        {
            return _context.Usuarios.Where(
                u => u.ID == id).FirstOrDefault();
        }
        2 references
        public void Incluir(Usuario dadosUsuario)
        {
            _context.Usuarios.Add(dadosUsuario);
            _context.SaveChanges();
        }
    }
}
```

Segurança no ASP.NET Core

Criar classe de modelo para o Token e alterar o *appsettings.json*:

```
namespace ExemploWebApi.Models
{
    3 references
    public class TokenConfiguration
    {
        2 references
        public string Audience { get; set; }
        2 references
        public string Issuer { get; set; }
        1 reference
        public int Seconds { get; set; }
    }
}
```

```
"TokenConfigurations": {
  "Audience": "ExemploAudience",
  "Issuer": "ExemploIssuer",
  "Seconds": 120
},
```

Segurança no ASP.NET Core

- Criar classe de modelo para a assinatura:

```
using System.Security.Cryptography;
using Microsoft.IdentityModel.Tokens;

namespace ExemploWebApi.Models
{
    2 references
    public class SigningConfigurations
    {
        3 references
        public SecurityKey Key { get; }
        2 references
        public SigningCredentials SigningCredentials { get; }
        1 reference
        public SigningConfigurations()
        {
            using (var provider = new RSACryptoServiceProvider(2048))
            {
                Key = new RsaSecurityKey(provider.ExportParameters(true));
            }
            SigningCredentials = new SigningCredentials(
                Key, SecurityAlgorithms.RsaSha256Signature);
        }
    }
}
```

Segurança no ASP.NET Core

- No *startup.cs*:

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddDbContext<ExemploWebApiContext>(opt =>
        opt.UseInMemoryDatabase("ExemploWebApiList"));
    services.AddScoped<UsuarioService>();

    var signingConfigurations = new SigningConfigurations();
    services.AddSingleton(signingConfigurations);

    var tokenConfigurations = new TokenConfiguration();
    new ConfigureFromConfigurationOptions<TokenConfiguration>(
        Configuration.GetSection("TokenConfigurations"))
            .Configure(tokenConfigurations);
    services.AddSingleton(tokenConfigurations);
```

Segurança no ASP.NET Core

- No *startup.cs*:

```
services.AddAuthentication(authOptions =>
{
    authOptions.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
    authOptions.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
}).AddJwtBearer(bearerOptions =>
{
    var paramsValidation = bearerOptions.TokenValidationParameters;
    paramsValidation.IssuerSigningKey = signingConfigurations.Key;
    paramsValidation.ValidAudience = tokenConfigurations.Audience;
    paramsValidation.ValidIssuer = tokenConfigurations.Issuer;

    // Valida a assinatura de um token recebido
    paramsValidation.ValidateIssuerSigningKey = true;

    // Verifica se um token recebido ainda é válido
    paramsValidation.ValidateLifetime = true;

    // Tempo de tolerância para a expiração de um token (utilizado
    // caso haja problemas de sincronismo de horário entre diferentes
    // computadores envolvidos no processo de comunicação)
    paramsValidation.ClockSkew = TimeSpan.Zero;
});
```

Segurança no ASP.NET Core

- No *startup.cs*:

```
// Ativa o uso do token como forma de autorizar o acesso
// a recursos deste projeto

services.AddAuthorization(auth =>
{
    auth.AddPolicy("Bearer", new AuthorizationPolicyBuilder()
        .AddAuthenticationSchemes(JwtBearerDefaults.AuthenticationScheme)
        .RequireAuthenticatedUser().Build());
});
```

Segurança no ASP.NET Core

- Criar um controller para o Login:

```
using System;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Authorization;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Security.Principal;
using Microsoft.IdentityModel.Tokens;
using ExemploWebApi.Models;
using ExemploWebApi.Services;

namespace ExemploWebApi.Controllers
{
    [Route("api/[controller]")]
    0 references
    public class LoginController : Controller
    {
        [AllowAnonymous]
        [HttpPost]
        0 references
        public object Post(
            [FromBody]Usuario usuario,
            [FromServices]UsuarioService usuarioService,
            [FromServices]SigningConfigurations signingConfigurations,
            [FromServices]TokenConfiguration tokenConfigurations)
            {
            bool credenciaisValidas = false;
            if (usuario != null && !String.IsNullOrWhiteSpace(usuario.ID))
            {
                var usuarioBase = usuarioService.Obter(usuario.ID);
                credenciaisValidas = (usuarioBase != null &&
                    usuario.ID == usuarioBase.ID &&
                    usuario.ChaveAcesso == usuarioBase.ChaveAcesso);
            }
```

Segurança no ASP.NET Core

- Ainda no controller para o Login:

```
if (credenciaisValidas)
{
    ClaimsIdentity identity = new ClaimsIdentity(
        new GenericIdentity(usuario.ID, "Login"),
        new[] {
            new Claim(JwtRegisteredClaimNames.Jti, Guid.NewGuid().ToString("N")),
            new Claim(JwtRegisteredClaimNames.UniqueName, usuario.ID)
        }
    );

    DateTime dataCriacao = DateTime.Now;
    DateTime dataExpiracao = dataCriacao +
        TimeSpan.FromSeconds(tokenConfigurations.Seconds);

    var handler = new JwtSecurityTokenHandler();
    var securityToken = handler.CreateToken(new SecurityTokenDescriptor
    {
        Issuer = tokenConfigurations.Issuer,
        Audience = tokenConfigurations.Audience,
        SigningCredentials = signingConfigurations.SigningCredentials,
        Subject = identity,
        NotBefore = dataCriacao,
        Expires = dataExpiracao
    });
    var token = handler.WriteToken(securityToken);
```

Segurança no ASP.NET Core

- Ainda no controller para o Login:

```
    return new
    {
        authenticated = true,
        created = dataCriacao.ToString("yyyy-MM-dd HH:mm:ss"),
        expiration = dataExpiracao.ToString("yyyy-MM-dd HH:mm:ss"),
        accessToken = token,
        message = "OK"
    };
}
else
{
    return new
    {
        authenticated = false,
        message = "Falha ao autenticar"
    };
}
```

Segurança no ASP.NET Core

- Exigir a autorização para o recurso no controller da API:

```
namespace ExemploWebApi.Controllers
{
    [Authorize("Bearer")]
    [Route("api/[controller]")]
    [ApiController]
    0 references
    public class ExemploWebApiController : ControllerBase
    {
        13 references
        private readonly ExemploWebApiContext _context;

        0 references
        public ExemploWebApiController(ExemploWebApiContext context)
        {
            _context = context;

            if (_context.ExemploWebApiItems.Count() == 0)
            {
                _context.ExemploWebApiItems.Add(new ExemploWebApiItem { Name = "Item1" });
                _context.SaveChanges();
            }
        }

        // GET: api/ExemploWebApi
        [HttpGet]
        0 references
        public async Task<ActionResult<IEnumerable<ExemploWebApiItem>>> GetItems()
        {
```

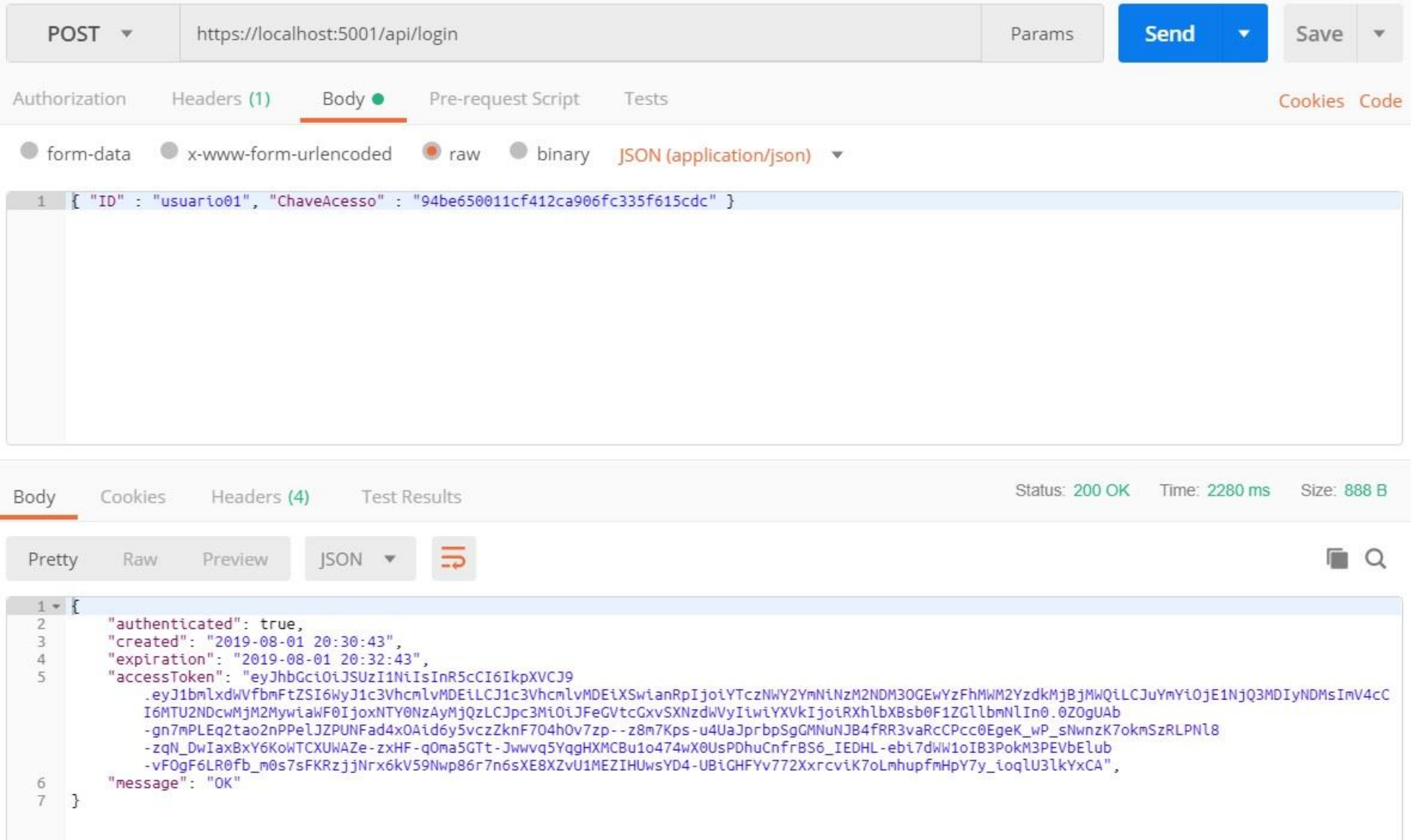
Segurança no ASP.NET Core

- Teste

```
public void Configure(IApplicationBuilder app, IHostingEnvironment env, UsuarioService usrService)
{

    usrService.Incluir(
        new Usuario() { ID = "usuario01", ChaveAcesso = "94be650011cf412ca906fc335f615cdc" });
    usrService.Incluir(
        new Usuario() { ID = "usuario02", ChaveAcesso = "531fd5b19d58438da0fd9afface43b3c" });

    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    else
    {
        // The default HSTS value is 30 days. You may want to change this for production scenarios,
        app.UseHsts();
    }

    app.UseHttpsRedirection();

    // Enable middleware to serve generated Swagger as a JSON endpoint.
    app.UseSwagger();

    // Enable middleware to serve swagger-ui (HTML, JS, CSS, etc.),
    // specifying the Swagger JSON endpoint.
    app.UseSwaggerUI(c =>
    {
        c.SwaggerEndpoint("/swagger/v1/swagger.json", "Minha API V1");
    });

    app.UseMvc();
}
```

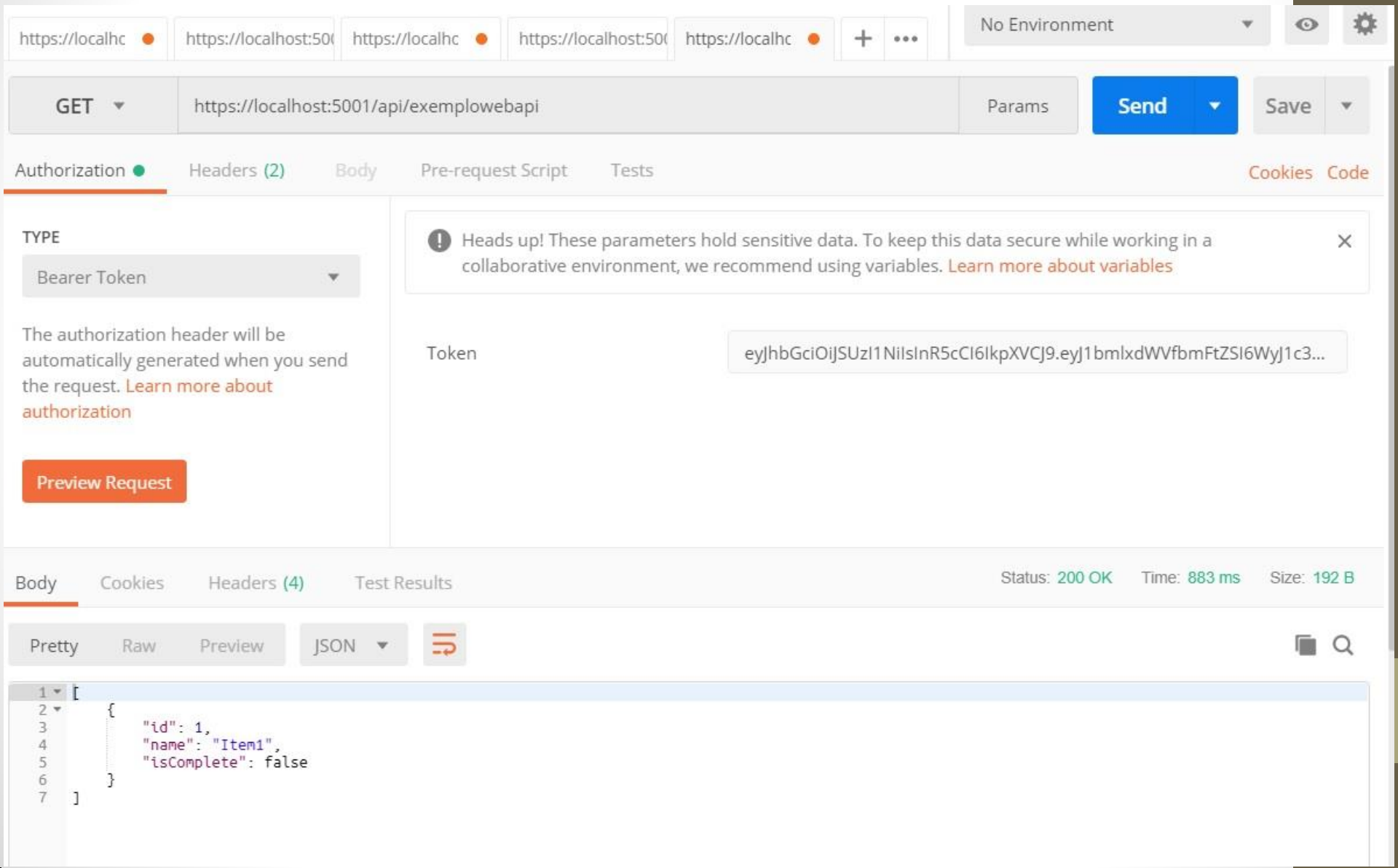
Segurança no ASP.NET Core

- Teste



Segurança no ASP.NET Core

- Teste



Dúvidas? Sugestões?
Comentários...

Vamos praticar!!!