

**Tipo** : Guía de Enunciado  
**Capítulo** : ASP NET Core  
**Duración** : 60 minutos

---

## I. OBJETIVO

Implementando autenticación por Token con ASP NET Core.

## II. REQUISITOS

Los siguientes elementos de software son necesarios para la realización del laboratorio:

- Windows 10 (como mínimo Windows 8)
- Visual Studio 2017 como mínimo.

## III. EJECUCIÓN DEL LABORATORIO

1. Abrir la solución “**Cibertec.NetCore.sln**” y ubicarse en el proyecto Cibertec.WebApi.
2. Procede a instalar los siguientes paquetes desde la consola de instalación:
  - Install-Package Microsoft.AspNetCore.Authentication.JwtBearer
  - Install-Package Microsoft.AspNetCore.Identity
  - Install-Package System.IdentityModel.Tokens.Jwt
  - Install-Package System.Security.Cryptography.Csp
3. Una vez instalado procede a crear la carpeta “**Authentication**” y en ella crea la interfaz “ITokenProvider” con el siguiente código:

```
using Cibertec.Models;  
using Microsoft.IdentityModel.Tokens;  
using System;  
  
namespace Cibertec.WebApi.Authentication  
{  
    public interface ITokenProvider  
    {  
        string CreateToken(User user, DateTime expiry);  
  
        TokenValidationParameters GetValidationParameters();  
    }  
}
```

4. En la misma carpeta crea la siguiente clase “JsonWebToken” con el siguiente código:

```
namespace Cibertec.WebApi.Authentication  
{  
  
    public class JsonWebToken  
    {  
        public string Access_Token { get; set; }  
  
        public string Token_Type { get; set; } = "bearer";  
  
        public int Expires_In { get; set; }  
    }  
}
```

```

        public string Refresh_Token { get; set; }
    }
}

```

5. Una vez finalizado procedemos a crear la implementación de ITokenProvider llamado **"RsaJwtTokenProvider"** con el siguiente código:

```

using Cibertec.Models;
using Microsoft.IdentityModel.Tokens;
using System;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Security.Cryptography;
using System.Security.Principal;

namespace Cibertec.WebApi.Authentication
{
    public class RsaJwtTokenProvider : ITokenProvider
    {
        private RsaSecurityKey _key;
        private string _algorithm;
        private string _issuer;
        private string _audience;

        public RsaJwtTokenProvider(string issuer, string audience, string
keyName)
        {
            var parameters = new CspParameters { KeyContainerName = keyName
};
            var provider = new RSACryptoServiceProvider(2048, parameters);

            _key = new RsaSecurityKey(provider);

            _algorithm = SecurityAlgorithms.RsaSha256Signature;
            _issuer = issuer;
            _audience = audience;
        }

        public string CreateToken(User user, DateTime expiry)
        {
            JwtSecurityTokenHandler tokenHandler = new
JwtSecurityTokenHandler();

            ClaimsIdentity identity = new ClaimsIdentity(new
GenericIdentity(user.Email, "jwt"));

            SecurityToken token = tokenHandler.CreateJwtSecurityToken(new
SecurityTokenDescriptor
            {
                Audience = _audience,
                Issuer = _issuer,
                SigningCredentials = new SigningCredentials(_key,
_algorithm),
                Expires = expiry.ToUniversalTime(),
                Subject = identity
            });

            return tokenHandler.WriteToken(token);
        }

        public TokenValidationParameters GetValidationParameters()
        {

```

```

        return new TokenValidationParameters
        {
            IssuerSigningKey = _key,
            ValidAudience = _audience,
            ValidIssuer = _issuer,
            ValidateLifetime = true,
            ClockSkew = TimeSpan.FromSeconds(0)
        };
    }
}

```

6. Ahora que ya tenemos los archivos necesarios para utilizar la autenticación basada en Tokens, procedemos a instanciarla en nuestro archivo Startup de la siguiente manera:

- En el método ConfigureServices, agregamos la siguiente línea de código:
 

```

var tokenProvider = new RsaJwtTokenProvider("issuer", "audience",
"token_cibertec_2017");
services.AddSingleton<ITokenProvider>(tokenProvider);

services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)
    .AddJwtBearer(options =>
    {
        options.RequireHttpsMetadata = false;
        options.TokenValidationParameters =
tokenProvider.GetValidationParameters();
    });

services.AddAuthorization(auth =>
{
    auth.DefaultPolicy = new AuthorizationPolicyBuilder()

.AddAuthenticationSchemes(JwtBearerDefaults.AuthenticationScheme)
    .RequireAuthenticatedUser()
    .Build();
});

```

- Y en el método Configure agregamos la siguiente línea:

```
app.UseAuthentication();
```

Esta línea debe estar antes de la instrucción **app.UseMvc();**

**Guiarse por la imagen:**

```

public void ConfigureServices(IServiceCollection services)
{
    services.AddSingleton<IUnitOfWork>(option => new NorthwindUnitOfWork(Configuration.GetConnectionString("Northwind")));
    var tokenProvider = new RsaJwtTokenProvider("issuer", "audience", "token_cibertec_2017");
    services.AddSingleton<ITokenProvider>(tokenProvider);

    services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)
        .AddJwtBearer(options =>
        {
            options.RequireHttpsMetadata = false;
            options.TokenValidationParameters = tokenProvider.GetValidationParameters();
        });

    services.AddAuthorization(auth =>
    {
        auth.DefaultPolicy = new AuthorizationPolicyBuilder()
            .AddAuthenticationSchemes(JwtBearerDefaults.AuthenticationScheme)
            .RequireAuthenticatedUser()
            .Build();
    });

    services.AddMvc();
}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
0 references | Cesar Velarde, 16 hours ago | 1 author, 1 change | 0 exceptions
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    app.UseAuthentication();
    app.UseMvc();
}

```

7. Una vez configurada procedemos a crear un nuevo controlador llamado **"TokenController"** con el siguiente código:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Cibertec.WebApi.Authentication;
using Cibertec.UnitOfWork;
using Cibertec.Models;

namespace Cibertec.WebApi.Controllers
{
    [Produces("application/json")]
    [Route("api/token")]
    public class TokenController : Controller
    {
        private ITokenProvider _tokenProvider;
        private IUnitOfWork _unit;

        public TokenController(ITokenProvider tokenProvider, IUnitOfWork
unit)
        {
            _tokenProvider = tokenProvider;
            _unit = unit;
        }

        [HttpPost]
        public JsonWebToken Post([FromBody] User userLogin)
        {
            var user = GetUserByCredentials(userLogin.Email,
userLogin.Password);

            if (user == null) throw new UnauthorizedAccessException("No!");

            var lifeInHours = 8;

```

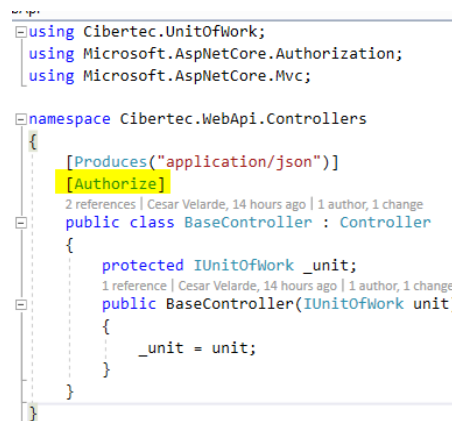
```

        var token = new JsonWebToken
        {
            Access_Token = _tokenProvider.CreateToken(user,
DateTime.UtcNow.AddHours(lifeInHours)),
            Expires_In = lifeInHours * 60
        };
        return token;
    }

    private User GetUserByCredentials(string email, string password)
    {
        return _unit.Users.ValidateUser(email, password);
    }
}

```

8. En la clase base procedemos a agregar la notación “[Authorize]” en nuestro controlador base:



```

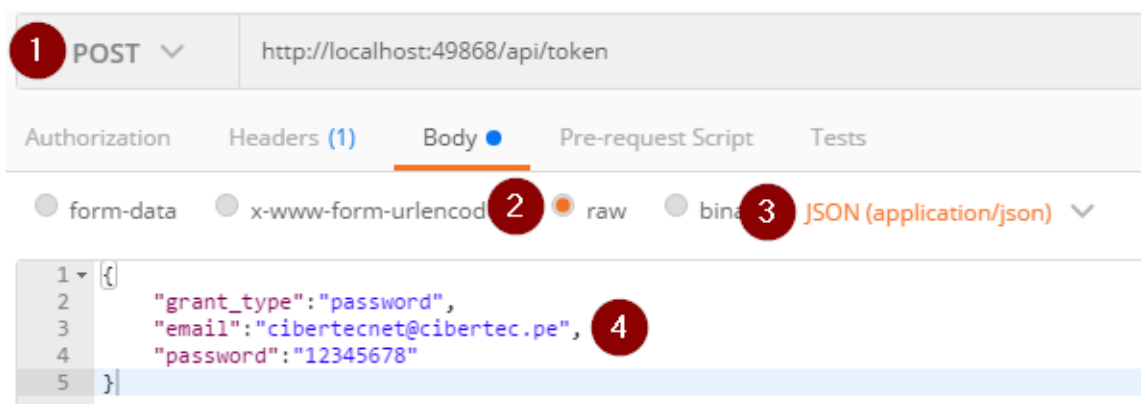
using Cibertec.UnitOfWork;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;

namespace Cibertec.WebApi.Controllers
{
    [Produces("application/json")]
    [Authorize]
    public class BaseController : Controller
    {
        protected IUnitOfWork _unit;

        public BaseController(IUnitOfWork unit)
        {
            _unit = unit;
        }
    }
}

```

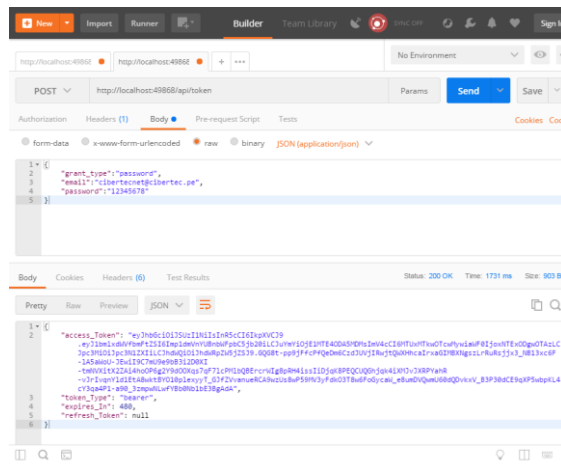
9. Para poder probar esta implementación procedemos a realizarla desde postman con la siguiente configuración:



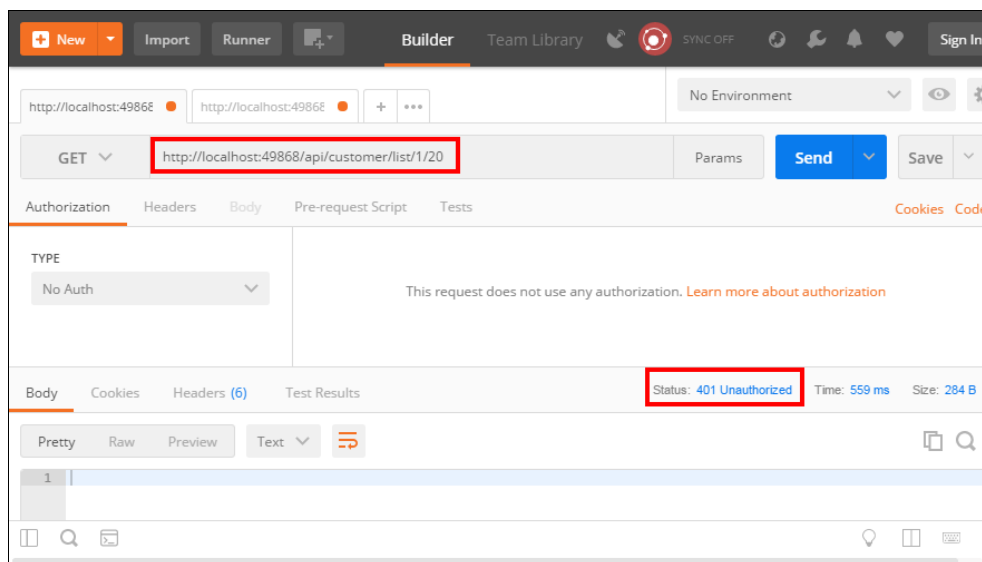
- (1) El método es **POST**
- (2) Seleccionamos la opción **raw**
- (3) En el dropdown, elegir la opción **JSON**
- (4) Como contenido del request

```
{
  "grant_type": "password",
  "email": "{tu_usuario}",
  "password": "{tu_password}"
}
```

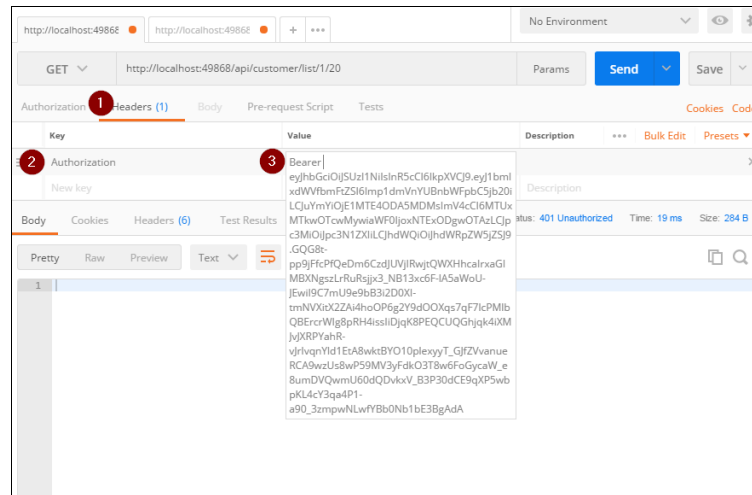
10. El resultado luego de hacer clic en **Send** es:



11. Validamos que el Authorize este validando los request contra la siguiente url **"http://localhost:{tu\_puerto}/api/customer/list/1/20"** y que nos muestre el response **401 – Unauthorized**:

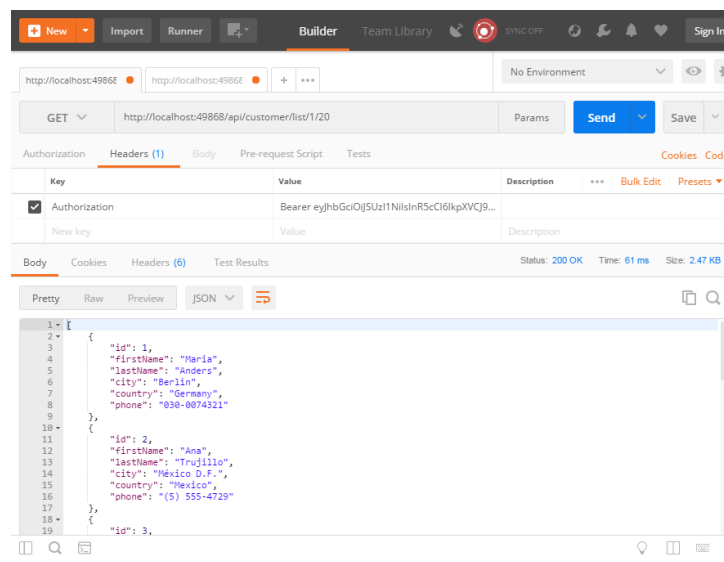


12. Procedemos a agregar el **Token** obtenido en el paso 10 y validar el resultado esperado desde Postman:



- (1) Clic en **Headers**
- (2) En la sección key digitar: “**Authorization**”
- (3) Como valor del key insertar “Bearer {tu\_token}”

13. Hacer clic en Send y validar que obtienes la lista de Customers:



#### IV. EVALUACIÓN

1. ¿Qué diferencia a .NET Core del .NET Framework?
2. ¿Por qué usar Asp Net Core?
3. ¿Cuáles son los beneficios de una Web Api con ASP NET Core?
4. ¿Dónde está el web.config?