Tipo : Guía de Enunciado
Capítulo : ASP NET Web Api
Duración : 60 minutos

I. OBJETIVO

Implementando Autenticación por Token.

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 Visual Studio 2015)

III. EJECUCIÓN DEL LABORATORIO

1. Abrir la solución del módulo anterior.

2. Ubicarse en el proyecto "**Cibertec.WebApi**"

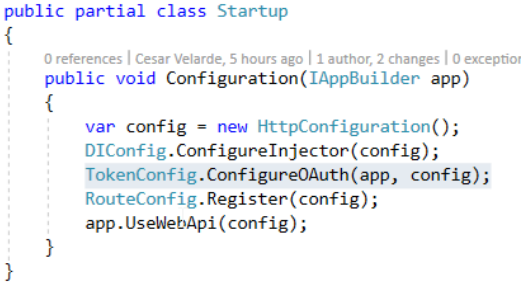
3. Instalar en el proyecto de WebApi la siguiente librería:
   - Install-Package Microsoft.Owin.Security.OAuth

4. En la carpeta "**App_Start**" crea la clase "**TokenConfig**" con el siguiente código:

```
using System;
using Microsoft.Owin;
using Microsoft.Owin.Security.OAuth;
using Owin;
using Cibertec.UnitOfWork;
using System.Threading.Tasks;
using System.Security.Claims;
using System.Web.Http;

namespace Cibertec.WebApi
{
    public static class TokenConfig
    {
        public static void ConfigureOAuth(IAppBuilder app,
HttpConfiguration config)
        {
            var unitOfWork =
(IUnitOfWork)config.DependencyResolver.GetService(typeof(IUnitOfWork));
            OAuthAuthorizationServerOptions OAuthServerOptions = new
OAuthAuthorizationServerOptions()
            {
                AllowInsecureHttp = true,
                TokenEndpointPath = new PathString("/token"),
                AccessTokenExpireTimeSpan = TimeSpan.FromDays(1),
                Provider = new
SimpleAuthorizationServerProvider(unitOfWork)
            };

            app.UseOAuthAuthorizationServer(OAuthServerOptions);
            app.UseOAuthBearerAuthentication(new
OAuthBearerAuthenticationOptions());
```

```
        }
    }

    public class SimpleAuthorizationServerProvider :
OAuthAuthorizationServerProvider
    {
        private readonly IUnitOfWork _unit;
        public SimpleAuthorizationServerProvider(IUnitOfWork unit)
        {
            _unit = unit;
        }
        public override async Task
ValidateClientAuthentication(OAuthValidateClientAuthenticationContext
context)
        {
            await Task.Factory.StartNew(() => { context.Validated(); });
        }

        public override async Task
GrantResourceOwnerCredentials(OAuthGrantResourceOwnerCredentialsContext
context)
        {
            await Task.Factory.StartNew(() =>
            {
                var user = _unit.Users.ValidaterUser(context.UserName,
context.Password);
                if (user == null)
                {
                    context.SetError("invalid_grant", "Wrong user or
password .");
                    return;
                }

                var identity = new
ClaimsIdentity(context.Options.AuthenticationType);
                identity.AddClaim(new Claim("sub", context.UserName));
                identity.AddClaim(new Claim("role", "user"));

                context.Validated(identity);
            });
        }
    }
}
```

5. En nuestro archivo “Startup.cs” agregamos la siguiente llamada: “**TokenConfig.ConfigureOAuth(app, config);**” como se muestra en la imagen:

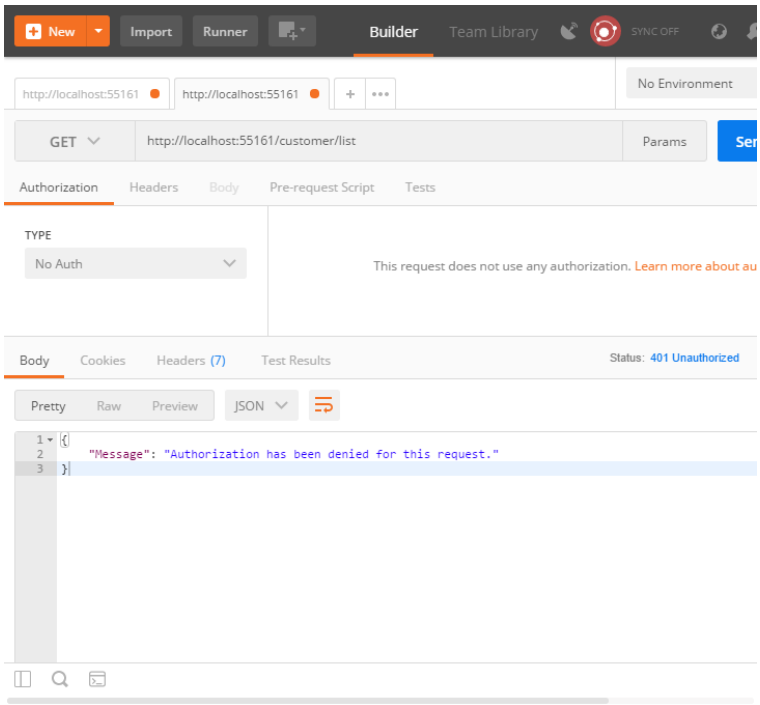


Importante respetar el orden de las llamadas.

6. Procedemos modificar nuestra clase base “**BaseController**”, en la cual agregaremos el atributo “**[Authorize]**” como se muestra en la imagen.

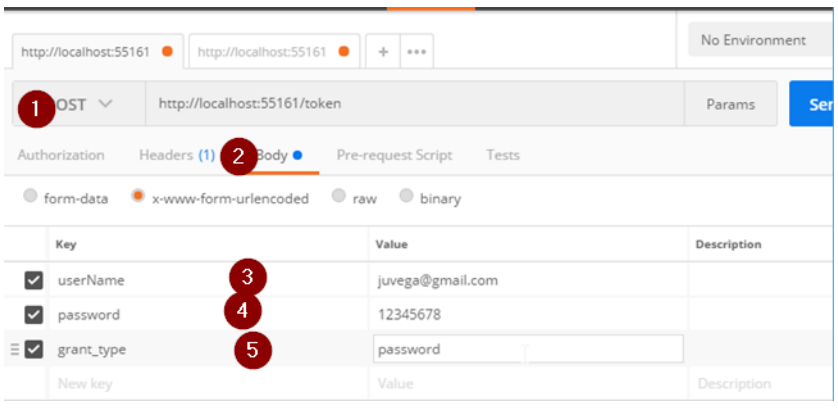
```
namespace Cibertec.WebApi.Controllers
{
    [Authorize]
    3 references | Cesar Velarde, 5 hours ago | 1 author, 1 change
    public class BaseController : ApiController
    {
        protected readonly IUnitOfWork _unit;
        2 references | Cesar Velarde, 5 hours ago | 1 author, 1 change | 0 e
        public BaseController(IUnitOfWork unit)
        {
            _unit = unit;
        }
    }
}
```

7. Una vez completado, procedemos a validar que la autenticación para nuestra WebApi se encuentre funcional, para lo cual haremos uso de Postman.
Iniciamos intentando acceder a la ruta: “**/customer/list**” (Ver Imagen)



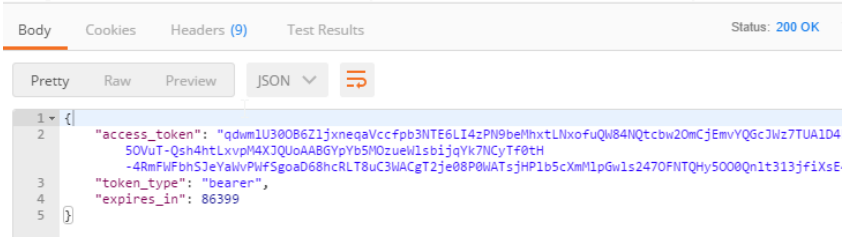
Resultado esperado 401 Unauthorized.

8. Validamos la autenticación de nuestro usuario para obtener el token.



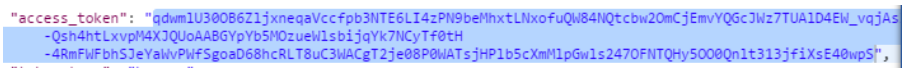
(1) Seleccionar **Post**
(2) Seleccionar **Body**
(3) Digitar como Key "**userName"** y el valor {**tu_usuario**}
(4) Digitar como Key "**password"** y el valor {**tu_password**}
(5) Digitar como Key "**grant_type"** y el valor "**password"**

Hacer clic en Send, el resultado será similar a este:

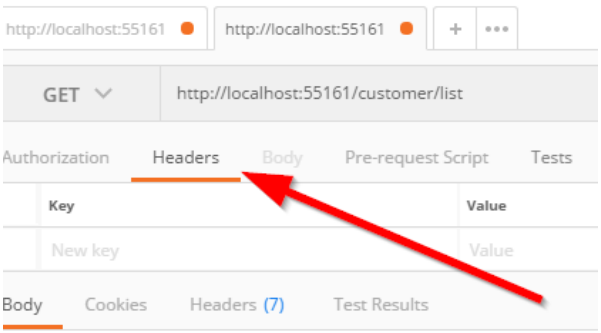


9. Ahora procedemos a probar que el token permita acceso a la lista de Customer previamente testeada. Para eso debemos de realizar los siguientes pasos:

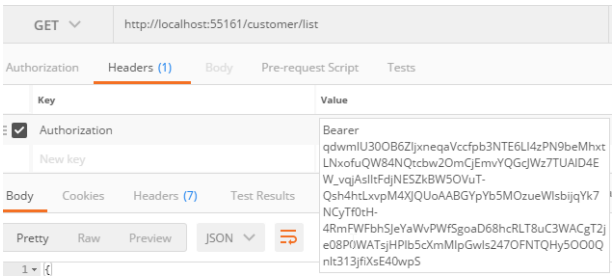
(1) Copiar el token.



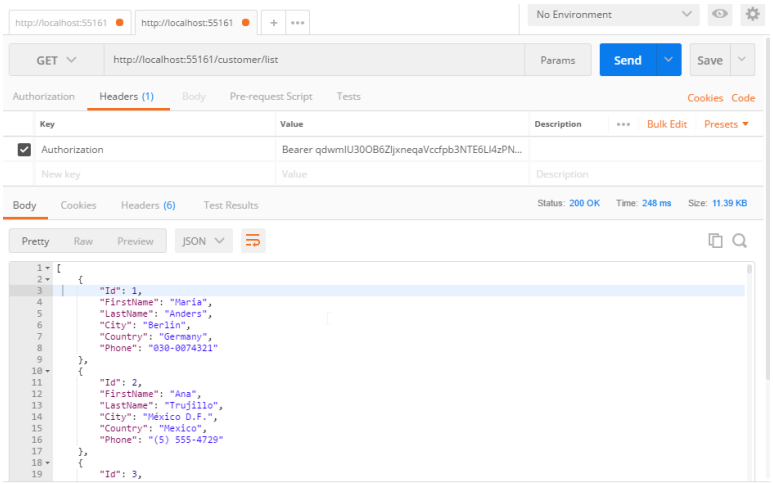
(2) En el tab con la url http://localhost:{puerto}/customer/listm hacer clic en Headers:



(3) En la sección Key, seleccionar "**Authorization**" y como valor poner "**Bearer {tu_token}**"



(4) Hacer clic en Send y validar que ahora si tienes el listado de Customer.



IV. EVALUACIÓN

1. ¿Por qué para vistas parciales se deben de usar “_” el guion bajo?

   Bajo el lenguaje Razor, el guion bajo permite evitar mostrar Web Pages cuando son solicitadas directamente como parte de un request.