

UNIVERSIDAD DE MÁLAGA
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
INGENIERO TÉCNICO EN INFORMÁTICA DE GESTIÓN

Un Servidor de Autorización OAuth 2.0

Realizado por
Juan Alberto Muñoz Rodríguez

Dirigido por
Antonio Jesús Nebro Urbaneja

Departamento
Lenguajes y Ciencias de la Computación

Málaga, 16 de diciembre de 2015

Contenido

- Objetivos
- Qué es OAuth 2.0
- Roles en OAuth 2.0
- Flujo abstracto del protocolo
- Tipos de tokens
- Flujo de refresco de token de acceso expirado
- Arquitectura OWIN
- Obtención de token de acceso
- Servidor de recursos
- Diagrama de base de datos
- Aplicación Cliente
- Conclusiones
- Trabajo futuro
- Resumen de tecnologías y herramientas utilizadas

Objetivos

1. Servidor de autorización OAuth 2.0
2. Servidor de recursos
3. Aplicación cliente (*SPA*)

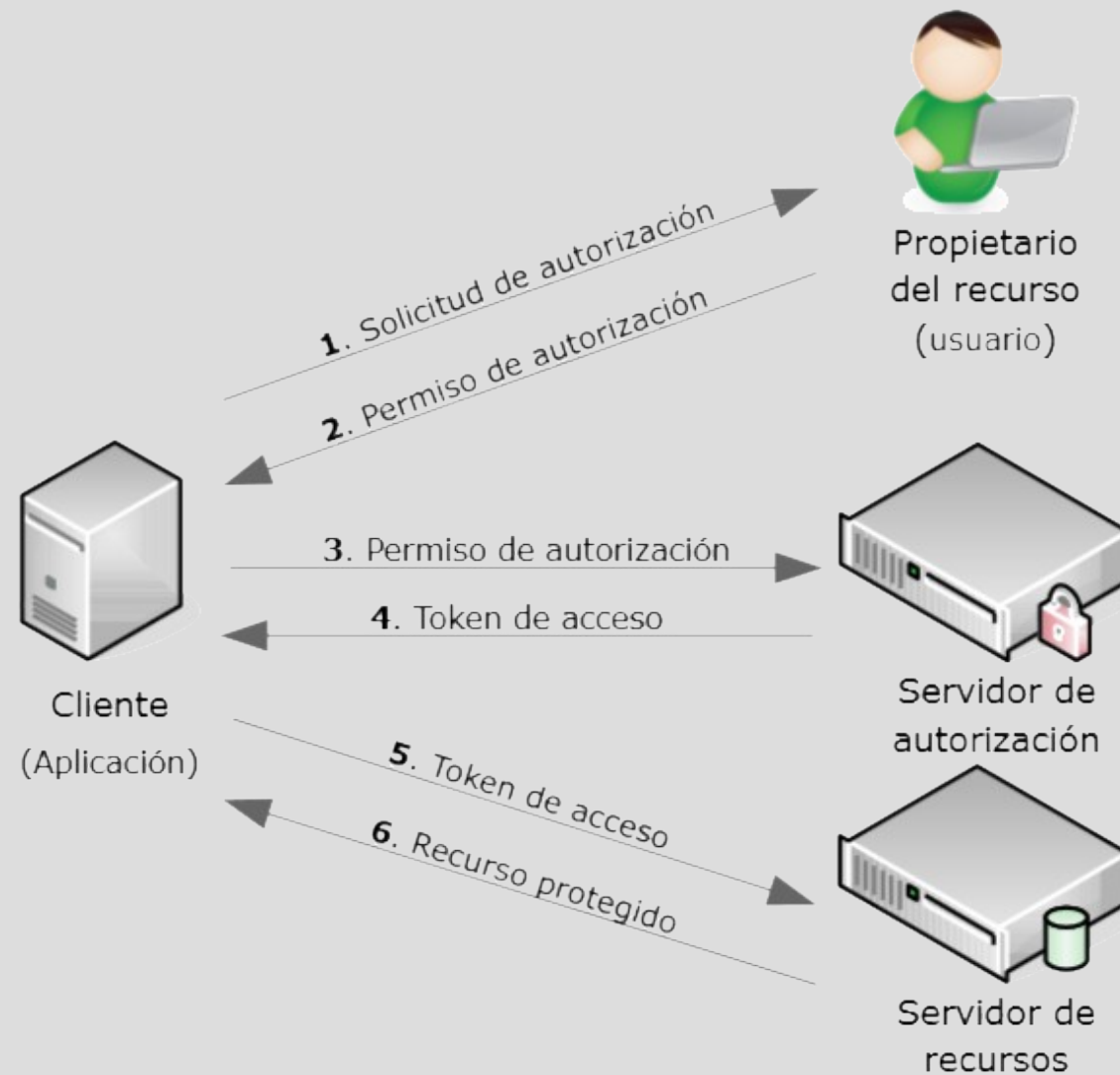
Qué es OAuth 2.0

- Es un estándar abierto para autorización.
- Provee a aplicaciones cliente de un “acceso delegado seguro” a servidores de recursos en nombre del propietario del recurso.
- Diseñado específicamente para trabajar con HTTP.
- Desarrollado por el *Internet Engineering Task Force OAuth Work Group (IETF OAuth WG)*.
- Es comúnmente usado para que usuarios accedan a sitios web de terceros usando sus cuentas de Microsoft, Google, Facebook, Twitter, etc.

Roles en OAuth 2.0

- Propietario de recursos
- Servidor de recursos
- Cliente
- Servidor de autorización

Flujo abstracto del protocolo



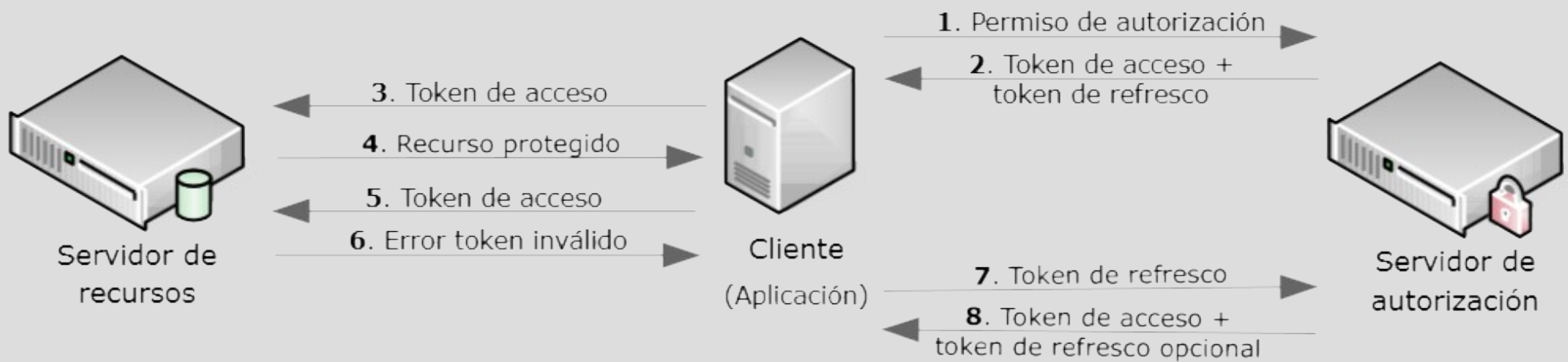
Tipos de tokens

- Token de Acceso
 - Credenciales para acceder a los recursos protegidos.
 - Cadena de texto que representa la autorización concedida al cliente.
 - Normalmente opaca al cliente.
 - Ámbito de acceso específico.
 - Duración de acceso específica, normalmente reducida.

Tipos de tokens

- Token de Refresco
 - Credenciales para obtener tokens de acceso.
 - Es opcional.
 - Se otorga junto con el token de acceso.
 - Cadena de texto que representa la autorización concedida al cliente.
 - Normalmente opaca al cliente.
 - Ámbito de acceso específico.
 - Duración de acceso específica, mayor que la del token de acceso.
 - Sólo se usa con el servidor de autorización.

Flujo de refresco de token de acceso expirado



Arquitectura OWIN



Arquitectura OWIN (II)

Detalle de implementación

```
using Microsoft.Owin;  
using Owin;
```

Clase de arranque OWIN →

```
[assembly: OwinStartup(typeof(OAuthServer.Startup))]
```

```
namespace OAuthServer  
{
```

```
    public partial class Startup  
    {
```

```
        public void Configuration(IAppBuilder app)  
        {
```

Componente Autofac →

```
            //Configure Autofac
```

```
            var dependencyResolver = ConfigureAutofac(app);
```

Componente CORS →

```
            //Configure CORS
```

```
            ConfigureCors(app);
```

Componente OAuth →

```
            //Configure OAuth
```

```
            ConfigureOAuth(app, dependencyResolver);
```

Componente Web API →

```
            //Configure Web API
```

```
            ConfigureWebApi(app, dependencyResolver);
```

```
        }
```

```
    }
```

```
}
```

Arquitectura OWIN (III)

Detalle de implementación

```
namespace OAuthServer
{
    public partial class Startup
    {
        public void ConfigureOAuth(IAppBuilder app, IDependencyResolver dependencyResolver)
        {
            ...

            var OAuthServerOptions = new OAuthAuthorizationServerOptions()
            {
                AllowInsecureHttp = false,
                TokenEndpointPath = new PathString(Paths.TokenPath),
                AccessTokenExpireTimeSpan =
                    TimeSpan.FromMinutes(Convert.ToDouble(ConfigurationManager.AppSettings["DefaultAccessTokenExpireTime"])),
                Provider =
                    (IOAuthAuthorizationServerProvider)dependencyResolver.GetService(typeof(IOAuthAuthorizationServerProvider)),
                RefreshTokenProvider =
                    (IAuthenticationTokenProvider) dependencyResolver.GetService(typeof(IAuthenticationTokenProvider))
            };

            var OAuthBearerOptions = new OAuthBearerAuthenticationOptions();

            // Token generation
            app.UseOAuthAuthorizationServer(OAuthServerOptions);
            // Token validation
            app.UseOAuthBearerAuthentication(OAuthBearerOptions);
        }
    }
}
```

Usar HTTPS → AllowInsecureHttp = false,

Tiempo expiración token acceso → AccessTokenExpireTimeSpan =

Proveedor OAuth → (IOAuthAuthorizationServerProvider)dependencyResolver.GetService(typeof(IOAuthAuthorizationServerProvider)),

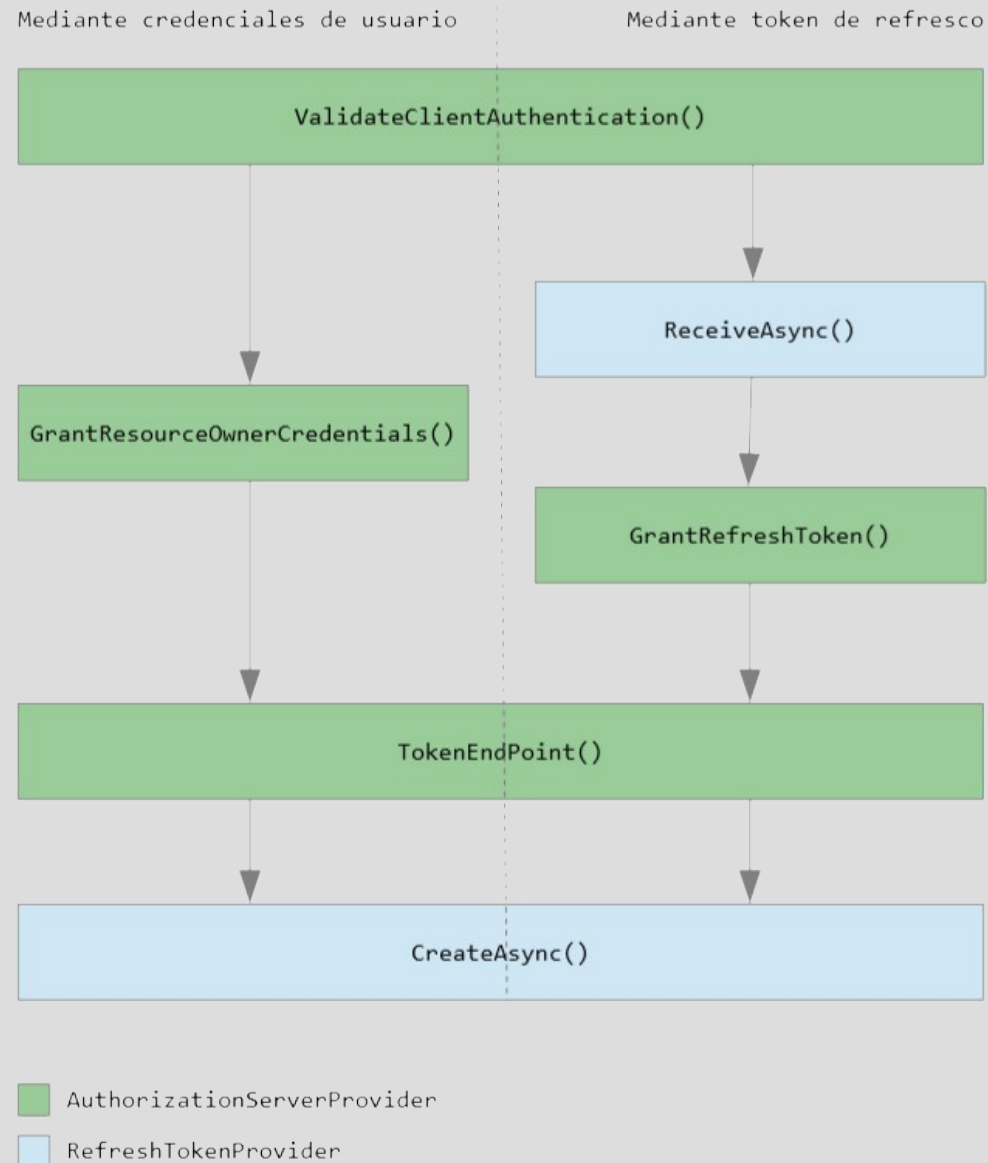
Proveedor gestión token refresco → (IAuthenticationTokenProvider) dependencyResolver.GetService(typeof(IAuthenticationTokenProvider))

→ /Token

→ Componente servidor OAuth

→ Componente bearer token

Obtención de token de acceso



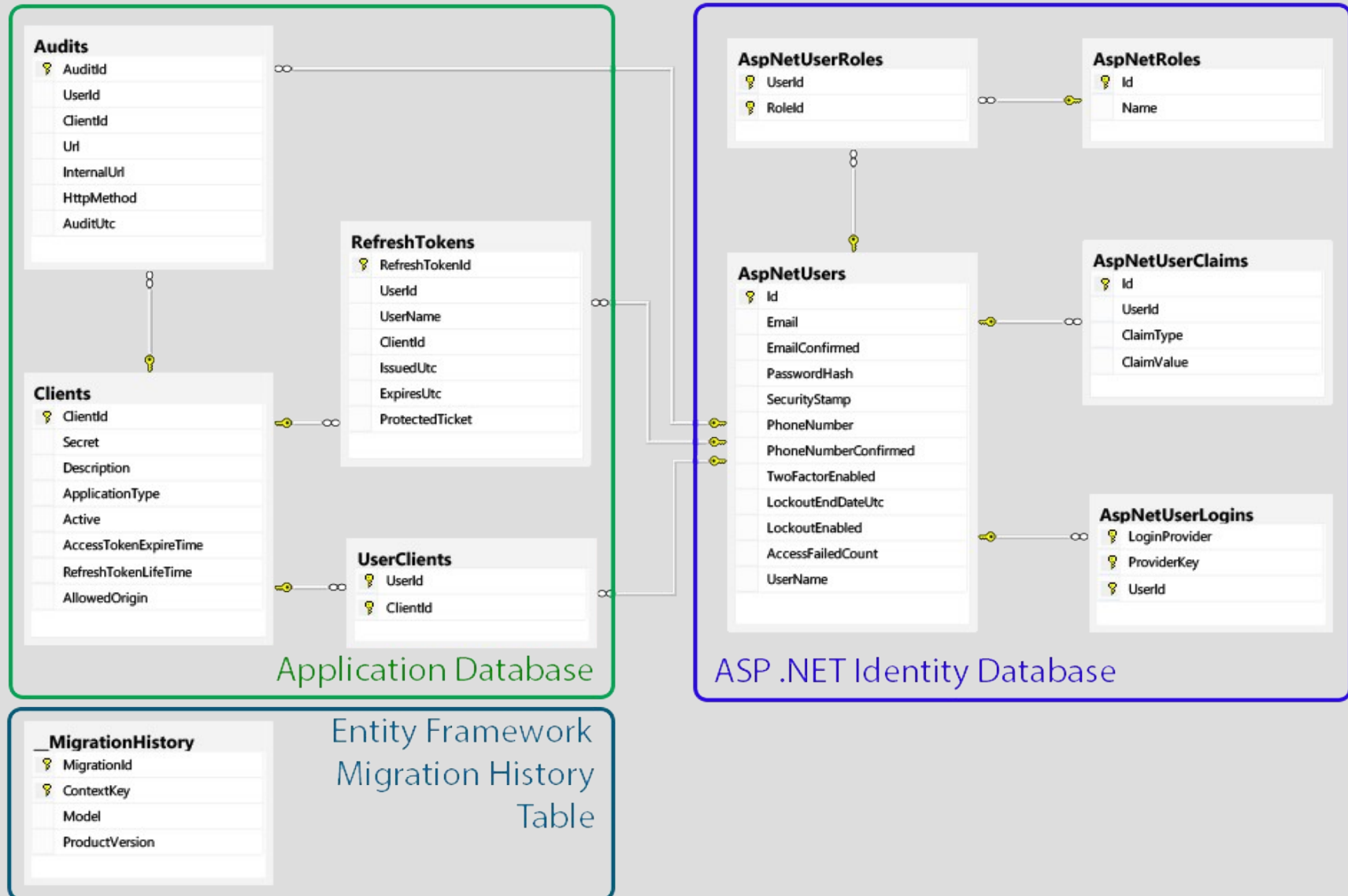
Servidor de recursos

PRESENTACIÓN
Web API, RESTful

NEGOCIO
Servicios

ACCESO A DATOS
Entity Framework *Code First*

Diagrama de base de datos



Aplicación Cliente

- SPA
- AngularJS
- HTML5
- Bootstrap

Demostración...

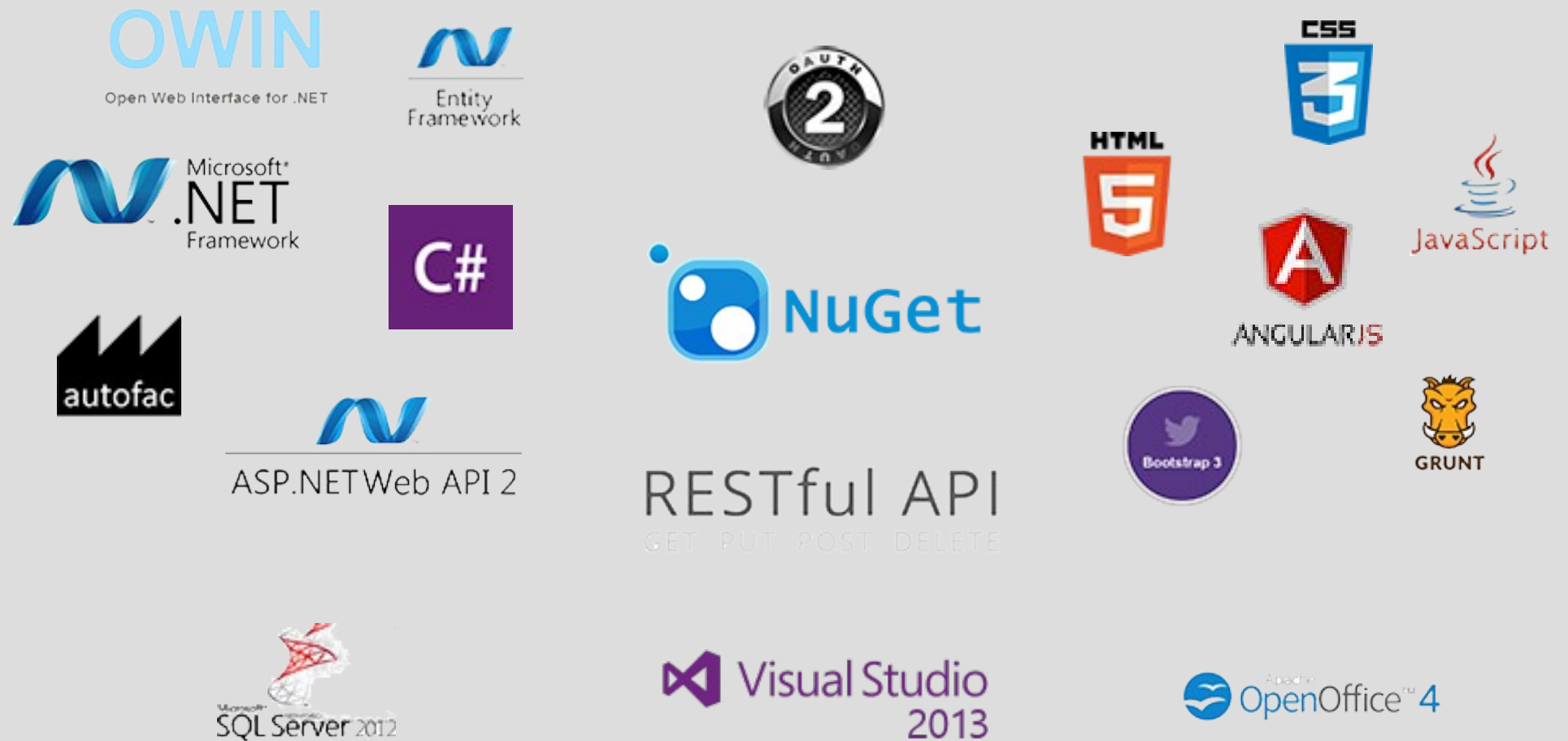
Conclusiones

- Se ha desarrollado un servidor de autorización completamente funcional.
- El servidor también es servidor de recursos.
- Se ha desarrollado la herramienta de administración para la gestión del sistema.
- Se han extraído diversos módulos funcionales a paquetes NuGet.
- Todo junto se podría implantar en un sistema real.

Trabajo futuro

- Añadir funcionalidad para permitir la autenticación con servidores OAuth externos como Google, Facebook, Twitter, StackExchange, etc.
- Mejorar la gestión de los tokens y datos de cliente en la aplicación cliente.
- Añadir página de registro de usuarios.

Resumen de tecnologías y herramientas utilizadas



¡Gracias por su atención!