ANTEPROYECTO FIN DE CARRERA

E.T.S.I. Informática

**Un Servidor de Autorización OAuth 2.0**

*An OAuth 2.0 Authorization Server*

# 

1. **Introducción**

OAuth es un estándar abierto para autorización. OAuth provee a aplicaciones cliente de un “acceso delegado seguro” a servidores de recursos en nombre del propietario del recurso. Especifica un proceso para que propietarios de recursos autoricen el acceso de terceras partes a sus servidores de recursos sin compartir sus credenciales. Diseñado específicamente para trabajar con HTTP, OAuth esencialmente permite que un servidor de autorización entregue tokens de acceso a aplicaciones cliente, con la aprobación del dueño de los recursos o usuario final (el token de acceso es una cadena especificando un ámbito específico, un tiempo de vida y otros atributos de acceso). La aplicación cliente hace uso del token de acceso para acceder a los recursos protegidos alojados en el servidor de recursos.

OAuth es comúnmente usado como una manera para que los usuarios accedan a sitios web de terceros usando sus cuentas de Microsoft, Google, Facebook, Twitter, etc. sin preocuparse por si sus credenciales de acceso están siendo comprometidas.

OAuth 2.0 es la siguiente evolución de OAuth y está enfocado a la simplicidad de desarrollo del cliente mientras provee de flujos de autorización específicos para aplicaciones web, aplicaciones de escritorio, teléfonos móviles y otros dispositivos multimedia.

La propuesta de este proyecto es realizar un servidor de autorización OAuth 2.0 que permitirá a usuarios de aplicaciones de terceros conectarse a dichas aplicaciones sin que éstas tengan la necesidad de conocer ni validar las credenciales de los usuarios.

Como se ha dicho antes, el servidor entrega al cliente tokens de acceso con un tiempo de validez limitado. Si el tiempo de validez del token es muy pequeño, hará que el usuario tenga que estar introduciendo sus credenciales en el servidor de autenticación cada poco tiempo, lo que será bastante incómodo para el usuario e irá en detrimento de la aplicación cliente.

Si el tiempo de validez del token es muy elevado, el uso de la aplicación será más cómodo para el usuario ya que se verá obligado a introducir sus credenciales menos veces, pero un hacker que eventualmente pueda interceptar alguna comunicación, tendrá más tiempo para poder suplantar la identidad del usuario, con los riesgos que esto puede conllevar.

Para evitar estos inconvenientes y algunos más, haremos uso de los llamados tokens de refresco. De este modo tendremos tokens de acceso con un tiempo de vida muy pequeño y tokens de refresco con un tiempo de vida más elevado que serán usados para obtener nuevos tokens de acceso cada vez que éstos caduquen.

La implementación del servidor de autorización OAuth 2.0 se realizará usando el middleware OWIN OAuth, desarrollado dentro del proyecto Katana de Microsoft. Por tanto dicho servidor estará desarrollado utilizando el framework de .NET. Además, el servidor expondrá sus servicios mediante Web API 2 y el acceso a base de datos se realizará utilizando Entity Framework. La gestión de los usuarios se hará con ASP.NET Identity. Asimismo se realizará utilizando el patrón de diseño Inyección de Dependencias, para lo que se utilizará el contenedor de control de inversión Autofac. Todo ello utilizando el lenguaje de programación C#.

También se ha pensado en la realización de una SPA (*Single Page Application*, Aplicación de una sola página en español) realizada con AngularJS. Esta aplicación tendrá varias secciones entre las que se incluirán la gestión de los usuarios, la gestión de las aplicaciones cliente que vayan a hacer uso del servidor de autenticación y la gestión de los tokens de refresco que el servidor haya emitido, lo que permitirá invalidarlos por cualquier motivo, forzando al usuario a introducir nuevamente sus credenciales.

Esta aplicación podrá tener usuarios con diferentes niveles de privilegios mediante el uso de roles, cuya implementación será facilitada por la utilización de ASP.NET Identity en el servidor.

La propia aplicación actuará como cliente del servidor de autorización, por lo que sus usuarios harán uso del servidor para su autenticación.

La utilización de diferentes tecnologías de reciente aparición y la complejidad que puede conllevar combinarlas todas en el servidor y la aplicación arriba descritos, con orientación real, justifican la realización del presente proyecto de fin de carrera.

1. **Objetivos**

El principal objetivo del proyecto es diseñar e implementar un servidor de autorización utilizando el estándar OAuth 2.0.

Gracias a la utilización de OWIN, podremos dividir nuestra aplicación de una manera sencilla en los siguientes módulos:

* **Autorización/autenticación**: Este módulo implementará toda la funcionalidad de autorización de OAuth 2.0.
* **Servicios HTTP**: El uso de Web API nos facilitará la tarea de construir nuestros servicios HTTP. Dividiremos este módulo en tres capas:
  + **Capa de presentación:**

Esta capa la identificaremos con nuestros controladores de Web API. Aquí se recibirán las peticiones HTTP de los clientes, se realizarán las validaciones necesarias, se hará uso de la capa de negocio para obtener o modificar los datos y se enviará la respuesta a través de HTTP a los clientes.

* + **Capa de negocio**:

En esta capa se implementará toda la lógica necesaria para manipular los datos.

* + **Capa de acceso a datos**:

Como su nombre indica, aquí implementaremos el acceso a la base de datos. Dicho acceso se implementará utilizando Entity Framework en su vertiente *Code-First*.

El segundo objetivo que nos hemos propuesto es la realización de una SPA con AngularJS. AngularJS plantea un patrón de desarrollo MVC (Modelo-Vista-Controlador), por lo que implementaremos nuestra aplicación siguiendo este patrón:

* **Modelo:**

Es el más bajo nivel del patrón, responsable de mantener los datos.

* **Vista**:

Es responsable de mostrar todos o parte de los datos al usuario.

* **Controlador**:

Es la parte del código que controla la interacción entre los datos y la vista.

1. **Fases**

La realización del proyecto contará con las siguientes fases:

* Estudio de las diferentes tecnologías y herramientas a usar (OAuth 2.0, OWIN OAuth, Web API 2, Entity Framework, ASP.NET Identity, AngularJS, Autofac, Visual Studio 2013, C#, …).
* Diseño funcional.
* Diseño técnico.
* Implementación.
* Evaluación del servidor de autorización y de la aplicación.+

1. **Material y recursos a utilizar**

El presente proyecto requiere del siguiente material para su desarrollo:

El hardware necesario para la fase de desarrollo será un pc con sistema operativo Windows 7 o Windows 8.1; para las pruebas será necesario usar varios ordenadores conectados en red.

En cuanto al software, se usará: Visual Studio 2013, OWIN OAuth, .NET Framework 4.5, ASP.NET Web API 2, Entity Framework, Autofac, lenguaje C#, javascript, JQuery, AngularJS.

1. **Referencias**

[1] OAuth 2.0: <http://oauth.net/2/>

[2] The OAuth 2.0 Authorization Framework: <http://tools.ietf.org/html/rfc6749>

[3] Token Based Authentication using ASP.NET Web API 2, Owin, and Identity: <http://bitoftech.net/2014/06/01/token-based-authentication-asp-net-web-api-2-owin-asp-net-identity/>

[4] OWIN and Katana: <http://www.asp.net/aspnet/overview/owin-and-katana>, <http://www.asp.net/aspnet/overview/owin-and-katana/owin-oauth-20-authorization-server>

[5] ASP.NET Web API 2: <http://www.asp.net/web-api>

[6] Entity Framework: <http://www.asp.net/entity-framework>

[7] Depency Injection pattern: <http://martinfowler.com/articles/injection.html>, <http://oscarsotorrio.com/post/2013/02/18/Contenedores-de-inversion-de-control-y-el-patron-inyeccion-de-dependencias.aspx>

[8] Autofac, <http://autofac.org/>

[9] AngularJS, <https://angularjs.org/>

[10] Oauth2 with Angular: The right way: <http://jeremymarc.github.io/2014/08/14/oauth2-with-angular-the-right-way/>

[11] OAuth for ASP.NET: <http://www.oauthforaspnet.com/>

[12] ASP.NET Identity: <http://www.asp.net/identity>