Método para fortalecer la seguridad en la autenticación de usuarios de las aplicaciones web, basado en el API de Microsoft Graph

Manuel Balderas Victoria, José Manuel Gómez Zea, José Ney Garrido Vázquez, Alejandro Hernández Cadena, José Ángel Jesús Magaña

Tecnológico Nacional de México Campus Villahermosa. División de Estudios de Posgrado e Investigación. Carretera Villahermosa – Frontera Km. 3.5 Ciudad Industrial Villahermosa, Tabasco, México C.P. 86010.

Resumen

Una de las más grandes preocupaciones al realizar aplicaciones web es la capa de seguridad, durante esta pandemia por COVID-19 se reporta que los ataques cibernéticos han aumentado un 450%, métodos como la autenticación por usuario y contraseña resultan insuficientes, e incluso peligrosos, por lo que se deben fortalecer las medidas de seguridad. Una de las alternativas más efectivas para resolver este problema es la delegación del proceso de autenticación y autorización a plataformas de identificación centralizadas en la nube. Las ventajas que esto conlleva van desde acelerar el proceso de desarrollo, hasta facilitar a los usuarios la administración de sus credenciales. En este artículo se describe el método de implementación de este tipo de funcionalidad, programado en una aplicación web a través del API de Microsoft Graph en el marco de trabajo Laravel.

Abstract

One of the biggest concerns when making web applications is the security layer, during this COVID-19 pandemic it is reported that cyber-attacks have increased by 450%, methods such as user and password authentication are insufficient and even dangerous, therefore, security measures must be strengthened. One of the most effective alternatives to solve this problem is the delegation of the authentication and authorization process to centralized identification platforms in the cloud. The benefits of this range from speeding up the development process to making it easier for users to manage their credentials. This article describes the method of implementing this type of functionality, programmed in a web application through the Microsoft Graph API in the Laravel framework.

Palabras clave: Ingeniería de software, Servicios web, Seguridad Informática
Keywords: Software Engineering, Web Services, Computer Security

1. INTRODUCCIÓN

Desde los comienzos de la programación, los desarrolladores han buscado facilitar el proceso de creación del software, dos de las ideas más atractivas consisten en conseguir que los programas se puedan escribir una sola vez y reutilizarlos de varias formas para la construcción de diferentes aplicaciones; y la otra, es que los programas puedan compartir información con otros programas, independientemente del lenguaje de programación utilizado o la plataforma desde la cual se ejecutan, a estos dos conceptos se les llama reusabilidad e interoperabilidad respectivamente.

En la actualidad el "ecosistema" del software ha evolucionado y se ha vuelto cada vez más complejo, pero al mismo tiempo se ha hecho cada vez más fácil de utilizar, gracias a tecnologías como el Internet, el sueño de la reusabilidad, y de la interoperabilidad se está haciendo realidad. La arquitectura orientada a servicios ha tenido un papel central en esta evolución, ya que ha permitido a las aplicaciones el conectarse de formas diferentes y

compartir información independientemente de la plataforma en que se encuentran, incluso sin importar el dispositivo desde donde se genera y se muestra la información.

Sin embargo no todo lo que han traído las grandes redes son cosas buenas, la seguridad del software ha tomado gran importancia debido al aumento en ataques cibernéticos a través del Internet especialmente a partir de la pandemia por COVID-19 que ha provocado el aumento en un 450% de este tipo de ataques, y si bien, antes las contraseñas eran una medida "segura" para proteger la información de los usuarios, hoy en día no es suficiente ya que hay vulnerabilidades inherentes a este método de autenticado, es necesario entonces contar con sistemas de autenticación de dos pasos, usando otros medios como mensajes, o aplicaciones de autenticación en conjunto con la contraseña, lo que complica el desarrollo de los módulos de identificación de usuario.

Afortunadamente, se pueden aprovechar los beneficios del ecosistema de software, haciendo uso de servicios de autenticación de grandes empresas, como puede ser Facebook, Google o Microsoft, de las cuales la mayoría de los usuarios ya tienen una cuenta, para desarrollar aplicaciones más seguras de una forma más fácil, en este artículo se presenta una manera de acceder a la autenticación de usuarios con cuenta de Microsoft en una aplicación web desarrollada con el lenguaje de programación PHP utilizando la API de Microsoft GRAPH para el acceso a sus servicios en la nube.

2. API DE MICROSOFT GRAPH

"Microsoft Graph API es una API web de RESTful que permite tener acceso a los recursos de servicio de Microsoft Cloud" (Murray, 2020), anteriormente existieron múltiples SDKs para los diferentes servicios de Microsoft, cada uno con sus propios requerimientos de seguridad y formato de datos, lo cual representa una curva de aprendizaje bastante inclinada para quien empieza a desarrollar aplicaciones que hagan uso de estos servicios.

Para corregir esta situación surge Microsoft GRAPH que proporciona un modelo de programación unificado y una estructura centralizada y estandarizada que permite a los desarrolladores hacer uso de los servicios y datos de Microsoft de una manera más fácil y rápida, se convierte en el punto único de acceso a los servicios de la nube de Microsoft, como se aprecia en la figura 1, de tal manera que esta API sea utilizada para crear aplicaciones para las organizaciones y los millones de usuarios que usan los productos de dicha empresa.

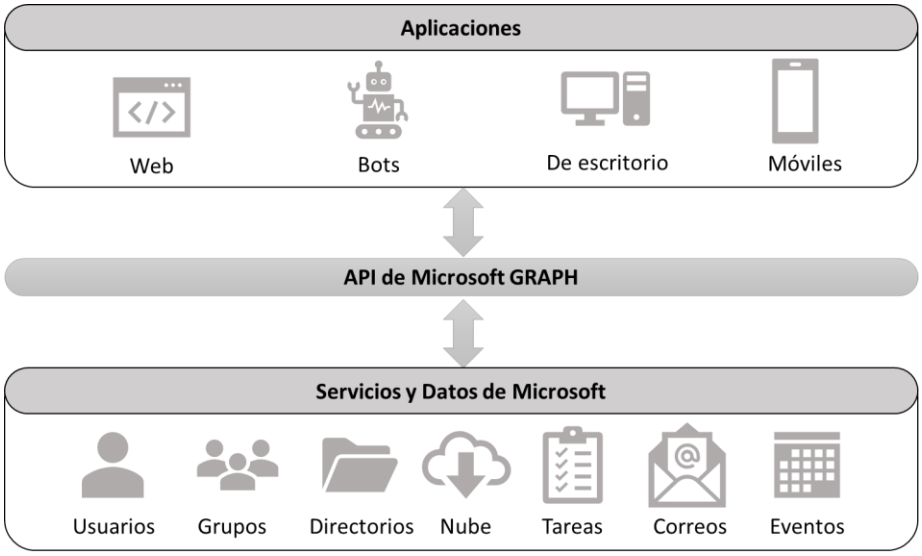

Figura 1. Diagrama de funcionamiento de la API Microsoft GRAPH

El primer paso para utilizar los beneficios de la API es poder identificar a los usuarios y controlar los permisos, es necesario utilizar entonces protocolos de autenticación y autorización, "El punto de conexión de la Plataforma de identidad de Microsoft para la identidad como servicio implementa la autenticación y la autorización con los protocolos estándar del sector, OpenID Connect [OIDC] y OAuth 2.0, respectivamente." (Singhal, 2020), cualquier implementación, aunque varíe un poco va a requerir un cliente con dichos protocolos.

3. MÉTODO DE IMPLEMENTACIÓN

La autenticación se refiere a la comprobación de la identidad del usuario, es decir, poder brindar con cierto grado de certeza la seguridad de que la persona detrás de la pantalla es quien dice ser. La autorización viene después de la autenticación, y se refiere a conceder a un usuario autenticado el permiso para utilizar alguna función dentro del sistema especificando a que datos puede acceder y lo que puede hacer con ellos.

Gestionar su propia información de nombre de usuario y contraseña conlleva una gran carga para las aplicaciones, debiendo tomar en cuenta cuestiones de seguridad y administrativas, de igual forma para los usuarios, teniendo que llevar registro de su nombre de usuario y contraseña con cada sistema diferente que utilizan, por lo anterior es mucho más conveniente el delegar la responsabilidad en un proveedor de identidades centralizado, especialmente en ambientes web, y es por esto que existen cada vez más portales y aplicaciones móviles que solicitan credenciales de cuentas en Facebook, Google o Microsoft.

Presentado de la forma anterior tal vez el utilizar el esquema mencionado parezca ser ideal, sin embargo, plantea un nuevo problema, obliga al usuario a compartir las credenciales de sus cuentas con aplicaciones de terceros, comprometiendo sus datos y recursos, es decir, cualquiera con acceso a estos sistemas podría tener acceso al email, calendario o redes sociales de las personas. Para lidiar con esto se ha creado una capa de abstracción que separa al cliente del propietario, de tal forma que una aplicación de terceros que quiera acceder a una cuenta de usuario de una persona, reciba otro conjunto de credenciales para obtener acceso a los recursos de la cuenta autorizados por la persona, es decir, en vez de que la aplicación tenga las credenciales

del usuario, recibirá un token de acceso el cual tendrá una duración limitada y únicamente tendrá acceso a los datos que el usuario haya autorizado, este proceso de detalla en la figura 2.

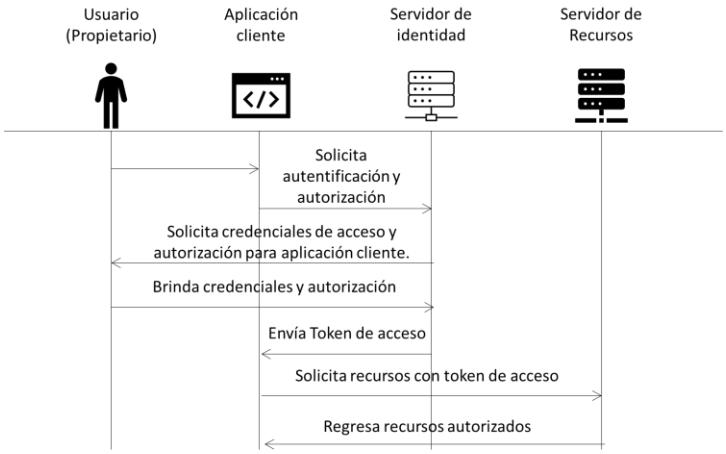

Figura 2. Diagrama de autorización y autenticación

4. CREACIÓN DEL PROYECTO

Para la creación del proyecto es necesario contar con PHP, Laravel y Composer instalados en el equipo, una opción es descargar una distribución de PHP como XAMPP o WAMP y similares, posteriormente instalar Composer desde su página oficial, y después utilizar una instrucción de Composer desde la línea de comandos para realizar la instalación de Laravel.

Es importante comprobar que la ruta de la carpeta de ejecutables de Composer se encuentre en la variable de entorno "PATH" para que el sistema pueda encontrar el comando de Laravel, en Windows se encuentra en el directorio de datos de aplicación de usuario.

```
composer global require laravel/installer
laravel new %proyecto%
composer require league/oauth2-client
composer require microsoft/microsoft-graph
php artisan serve
```

Figura 3. Comandos de Composer y Laravel para creación del proyecto

Se utiliza el comando "new" de Laravel para poder crear un proyecto base con dicho Framework, con este se generará de manera automática la estructura de carpetas necesarias. Una vez creado el proyecto se puede utilizar Composer para descargar los archivos necesarios de Microsoft Graph, que incluye OpenID Connect, y la librería de Oauth2.0 para crear una aplicación cliente con dicho protocolo. Todas las instrucciones utilizadas aparecen en la figura 3.

5. REGISTRO DE APLICACIÓN EN EL PORTAL DE AZURE AD

Para que la plataforma de identidad de Microsoft pueda proporcionar a la aplicación el servicio de autenticación y autorización, primero esta se registra en su portal. Azure Active Directory es un proveedor de identidades centralizado en la nube, para acceder se requieren las credenciales de acceso de una cuenta Microsoft personal, educativa o de trabajo. En la figura 4 se observa el portal como se muestra una vez iniciada la sesión.

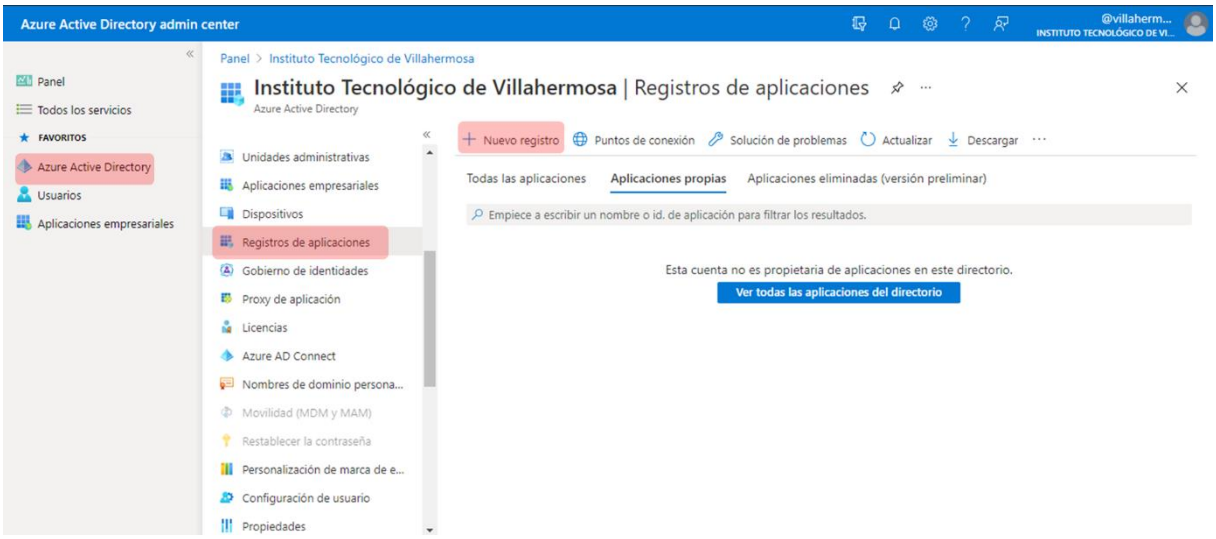

Figura4. Portal de Azure Active Directory

Una vez dentro del portal se localiza la opción para registro de aplicaciones, para crear un nuevo registro, esta abre un formulario, mostrado en la figura 4, que solicita la información general sobre la aplicación como el nombre, URL y tipo de cuenta que tendrá acceso.

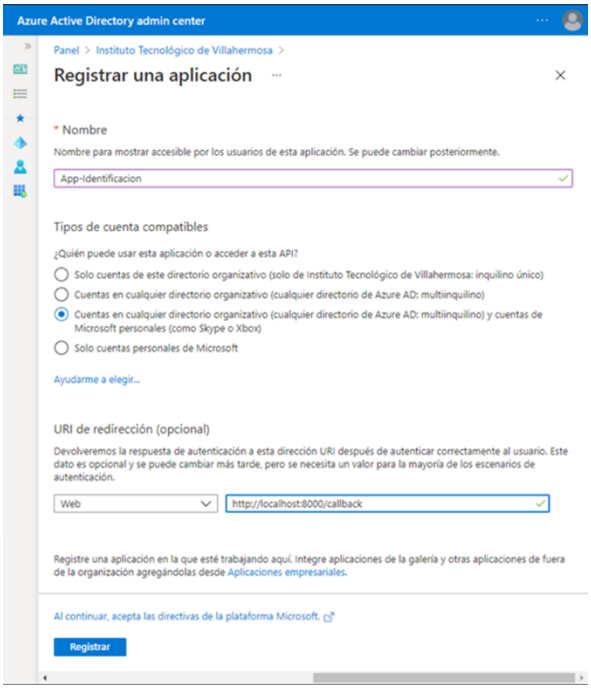

Figura 5. Formulario de registro de nueva aplicación

Una vez registrada la aplicación, el portal provee un identificador bajo el nombre de "Application (client) ID", este se trata de un identificador global de la aplicación dentro del servidor de Azure el cual se utiliza posteriormente en la definición del entorno de Laravel. Su localización se aprecia en la figura 6.

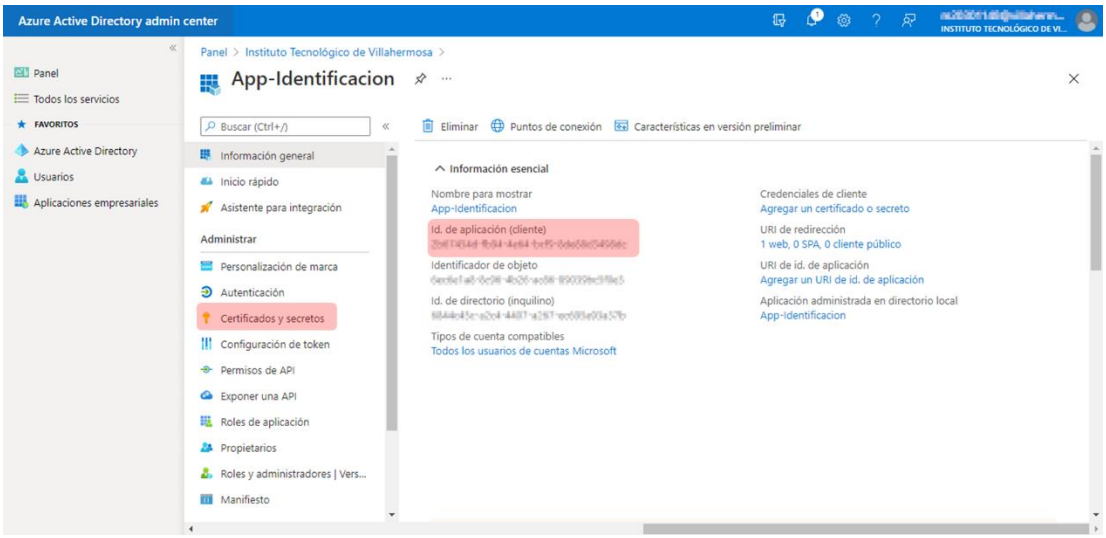

Figura 6. Página de información de aplicación

Lo siguiente es obtener un secreto de cliente para la aplicación, este consiste en una cadena que la aplicación utiliza como prueba de su identidad al momento de solicitar un Token de acceso, también se puede llamar contraseña de aplicación. Para obtenerlo se selecciona la opción de "Certificados y secretos", y luego el botón de "Nuevo secreto de cliente", después se llenan los datos del formulario, que solicita una descripción, así como una fecha de expiración, obsérvese la figura 7.

ISSN: 2007-4786

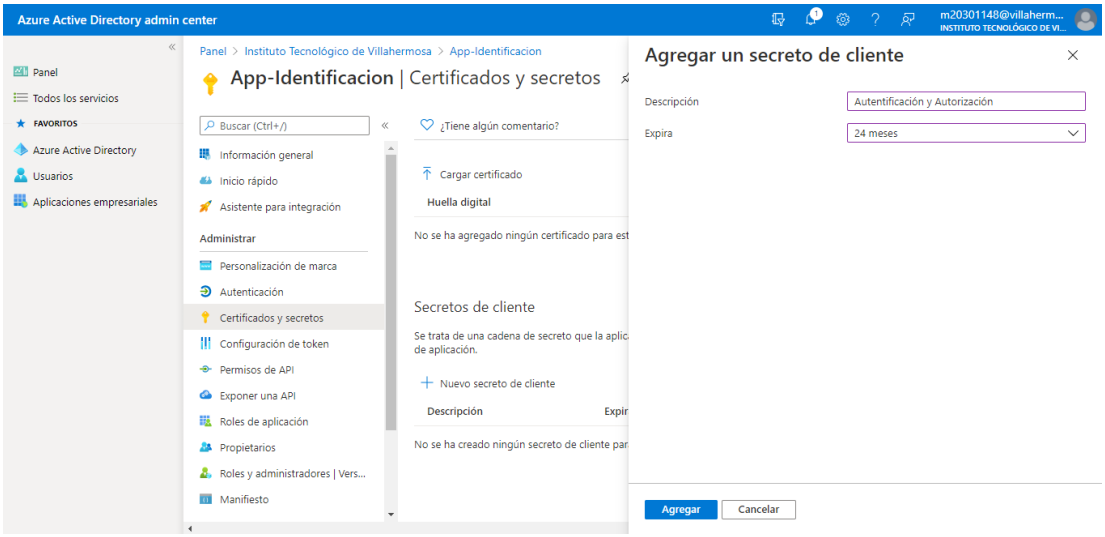

Figura 7. Formulario de secreto de cliente.

Después de rellenar el formulario, el portal muestra una tabla con los datos de descripción y expiración brindados, además de los datos de valor e identificador del secreto como aparece en la figura 8, se anota y resguarda el dato "valor" para utilizarlo después en la definición de entorno de Laravel.

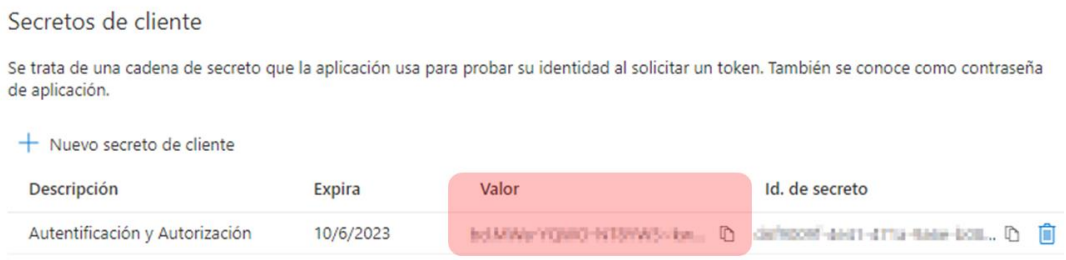

Figura 8. Tabla de secretos de cliente en el portal de Azure AD

6. CONFIGURACIÓN DE LARAVEL

Después de que la aplicación base es creada, el archivo de entorno de Laravel (.env) tiene que ser cambiado para coincidir con el entorno de desarrollo, de lo contrario Laravel será incapaz de conectarse con los servicios requeridos. Se podría modificar directamente los archivos de configuración de Laravel en la carpeta "config", pero es buena práctica que estos archivos toman los valores del archivo ".env". Lo anterior obedece a razones de seguridad puesto que aquí se escriben por lo general datos sensibles en texto, en este caso la contraseña de aplicación, y si se trabaja con repositorios de control de versiones, todo el código, excepción del archivo ".env", es publicado en línea.

Se agregan las siguientes líneas en el archivo ".env" para conectarse al servicio de identificación de Microsoft reemplazando las variables envueltas en símbolos "%" por los datos de ID de aplicación, secreto de aplicación y la URL, dada de alta en Azure AD anteriormente.

```
OAUTH_APP_ID=90de42b0-55ed-4044-bb6b-ce6abda2aba5
OAUTH_APP_SECRET=0V90eIVUF~zk-QlwOth2WQG.G~1A0jI..8
OAUTH_REDIRECT_URI=http://localhost:8000/callback
OAUTH_SCOPES='openid profile offline_access user.read mailboxsettings.read calendars.readwrite'
OAUTH_AUTHORITY=https://login.microsoftonline.com/common
OAUTH_AUTHORIZE_ENDPOINT=/oauth2/v2.0/authorize
OAUTH_TOKEN_ENDPOINT=/oauth2/v2.0/token
```

Figura 1. Código a añadir en archivo de variables de entorno del Proyecto base de Laravel.

La variable "OAUTH_SCOPES" hace referencia a los permisos que se otorgan a la aplicación en el momento que un usuario con credenciales de Microsoft da su autorización, y las otras variables son la URL de autorización y los puntos de autorización y obtención de token. Las líneas de configuración se observan en la figura 9.
Un archivo de configuración obtiene las variables del archivo ".env", la mejor opción es crear un documento nuevo en el directorio "/config" y llamarlo "azure.php" al cual se añade el código necesario para extraer los valores. El código resultante se aprecia en la figura 10.

```
return [
    'appId'               => env('OAUTH_APP_ID', ''),
    'appSecret'           => env('OAUTH_APP_SECRET', ''),
    'redirectUri'         => env('OAUTH_REDIRECT_URI', ''),
    'scopes'              => env('OAUTH_SCOPES', ''),
    'authority'           => env('OAUTH_AUTHORITY', 'https://login.microsoftonline.com/common'),
    'authorizeEndpoint'   => env('OAUTH_AUTHORIZE_ENDPOINT', '/oauth2/v2.0/authorize'),
    'tokenEndpoint'       => env('OAUTH_TOKEN_ENDPOINT', '/oauth2/v2.0/token'),
];
```

Figura 2. Código de configuración a añadir en directorio /config de Proyecto Base.

7. IMPLEMENTACIÓN DE FUNCIÓN DE AUTENTICACIÓN

En la estructura de Laravel se añade un controlador llamado AuthController para la autenticación del usuario, y autorización para la aplicación. En el controlador se definen dos acciones principales, "signIn()" y "callBack()", estas funciones se encargan de cumplir con el flujo requerido por el protocolo OAUTH2.0 como se observa en la figura 11, la primera se encarga de redirigir el navegador a la página de inicio de sesión de Microsoft, mientras que la segunda es a la que, una vez completado el proceso de autenticación y autorización en la plataforma de identidad de Microsoft, será redirigido el navegador y usa el código enviado por Azure AD para solicitar un token de acceso, entonces la aplicación podrá hacer uso del token para acceder a los servicios y datos autorizados por el usuario en la solicitud realizada, es decir, los incluidos en la variable "OAUTH_SCOPES" en el archivo de entorno. También se definen la función "singOut()" para finalizar la sesión y "redirect()", que se encarga del proceso de redirección del navegador.

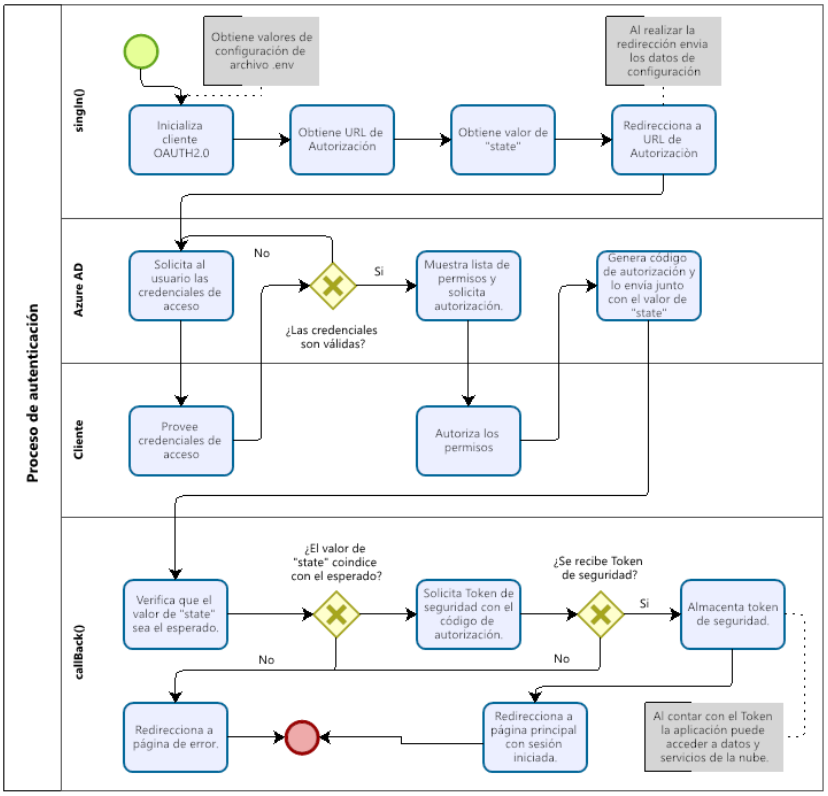

Figura 3. Proceso de autenticación.

En la figura 12 se detalla la implementación de la función signIn() que inicializa el cliente OAUTH2.0 y posteriormente genera la URL de inicio de sesión de Azure AD, almacena el valor de "state", por razones de seguridad, y redirecciona el navegador a la página de inicio de sesión de Azure AD.

```php
public function signin()
{
    // Inicializa el cliente OAUTH 2.0
    $oauthClient = new \League\OAuth2\Client\Provider\GenericProvider([
        'clientId'                => config('azure.appId'),
        'clientSecret'            => config('azure.appSecret'),
        'redirectUri'             => config('azure.redirectUri'),
        'urlAuthorize'            => config('azure.authority') . config('azure.authorizeEndpoint'),
        'urlAccessToken'          => config('azure.authority') . config('azure.tokenEndpoint'),
        'urlResourceOwnerDetails' => '',
        'scopes'                  => config('azure.scopes')
    ]);

    $authUrl = $oauthClient->getAuthorizationUrl();

    // Almacena el valor de "state" para poder validar después
    session(['oauthState' => $oauthClient->getState()]);

    // Redirecciona a la página de inicio de sesión de Azure AD
    return redirect()->away($authUrl);
}
```

Figura 4. Código de función singIn().

Una vez redirigido, el usuario se encarga de ingresar sus credenciales de acceso y autorizar los permisos solicitados en la aplicación, entonces Azure AD genera un código de autorización y lo envía empaquetado junto

ISSN: 2007-4786

con el valor de "state" de regreso a la aplicación cliente, a la función callBack(), pues está en la URL de redirección que se le proporciona en la función, y además es la ruta registrada en la página de Microsoft Azure. En la figura 13 se detalla la implementación de la función callBack(), que actúa como validadora, comienza por verificar el valor de "state" regresado por Azure AD, al comprobar que el valor existe y es el mismo generado en la función "signIn()" se continua con el proceso, de lo contrario solamente realiza una redirección a la página principal informando del error encontrado.

Suponiendo que todo esté en orden, el siguiente paso que realiza la función es solicitar a Azure AD un token de acceso utilizando el código de autorización obtenido, en caso de ocurrir algún error la función redirige de igual forma a la página principal informando sobre el error, en caso inverso la aplicación recibirá el token de acceso, finalizando el proceso de autenticación y autorización.

Posteriormente, la aplicación podrá realizar solicitudes a los servicios de Microsoft autorizados por el usuario, y tendrá que gestionar de manera adecuada el token de acceso almacenándolo en una ubicación segura, como pudiera ser una base de datos, y encargarse de actualizarlo puesto que los tokens de acceso tienen una validez de 24 horas, no es necesario repetir el proceso de autenticación mientras la sesión en Azure AD no sea cerrada por la aplicación. Así mismo, se tendrán que adecuar modelos, vistas y controladores para recibir la información de la nube de Microsoft y presentarla al usuario, sin embargo, estas tareas quedan fuera del alcance de este documento.

```php
public function callback(Request $request)
{
    // Valida valor de "state" esperado.
    $expectedState = session('oauthState');
    $request->session()->forget('oauthState');
    $providedState = $request->query('state');

    if (!isset($expectedState)) {
        // Si no encuentra el valor de "state" esperado, redirecciona a inicio.
        return redirect('/');
    }

    if (!isset($providedState) || $expectedState != $providedState) {
        return redirect('/')
            ->with('error', 'Invalid auth state')
            ->with('errorDetail', 'The provided auth state did not match the expected value');
    }

    // El código de autorización deberia encontrarse en el parámetro code.
    $authCode = $request->query('code');
    if (isset($authCode)) {
        // Inicializa el cliente OAUTH2.0
        $oauthClient = new \League\OAuth2\Client\Provider\GenericProvider([
            'clientId'                => config('azure.appId'),
            'clientSecret'            => config('azure.appSecret'),
            'redirectUri'             => config('azure.redirectUri'),
            'urlAuthorize'            => config('azure.authority') . config('azure.authorizeEndpoint'),
            'urlAccessToken'          => config('azure.authority') . config('azure.tokenEndpoint'),
            'urlResourceOwnerDetails' => '',
            'scopes'                  => config('azure.scopes')
        ]);

    try {
        //Solicita el token de acceso con el código de Azure AD
        $accessToken = $oauthClient->getAccessToken('authorization_code', [
            'code' => $authCode
        ]);

        $graph = new Graph();
        $graph->setAccessToken($accessToken->getToken());

        $user = $graph->createRequest('GET', '/me?$select=displayName,mail,mailboxSettings,userPrincipalName')
            ->setReturnType(Model\User::class)
            ->execute();

        $tokenCache = new TokenCache();
        $tokenCache->storeTokens($accessToken, $user);

        return redirect('/');
    } catch (\League\OAuth2\Client\Provider\Exception\IdentityProviderException $e) {
        return redirect('/')
            ->with('error', 'Error requesting access token')
            ->with('errorDetail', $e->getMessage());
    }
}
```

Figura 5. Código de función callBack().

8. RESULTADO

Al ejecutar la función signIn() la aplicación web muestra al dialogo de inicio de sesión de Microsoft Azure AD como se muestra en la figura 14, su función será el comprobar la identidad del usuario, para realizar la identificación el formulario cuenta con funcionalidad de autenticación multifactor, permite diferentes maneras ya sea utilizando un correo electrónico alterno, un número celular o la aplicación Autenticator de Microsoft.

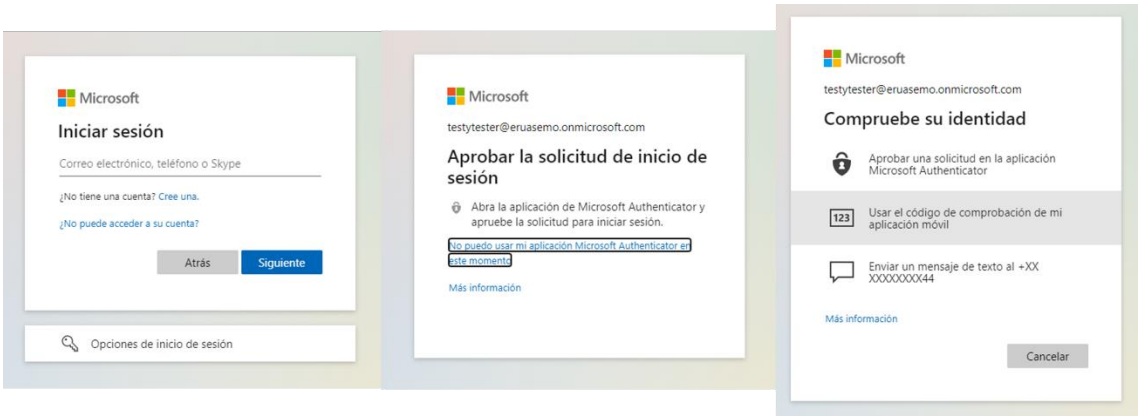

Figura 6. Formulario de inicio de sesión en Microsoft

Una vez completada la autenticación, el portal solicita al usuario su autorización para compartir la información requerida por la aplicación web, informando al usuario varios detalles sobre la aplicación como su nombre, los datos y servicios a los cuales solicita acceso, y si la aplicación se encuentra o no firmada digitalmente por un dominio de confianza. Además, muestra al usuario una explicación resumida de la autorización, así como ligas a páginas donde puede encontrar una explicación más extensa sobre la acción que se encuentra realizando. En la figura 15 se observa como se muestra en el portal, en el caso del prototipo realizado, solicita los permisos de acceder a los datos del perfil de usuario, así como al calendario.

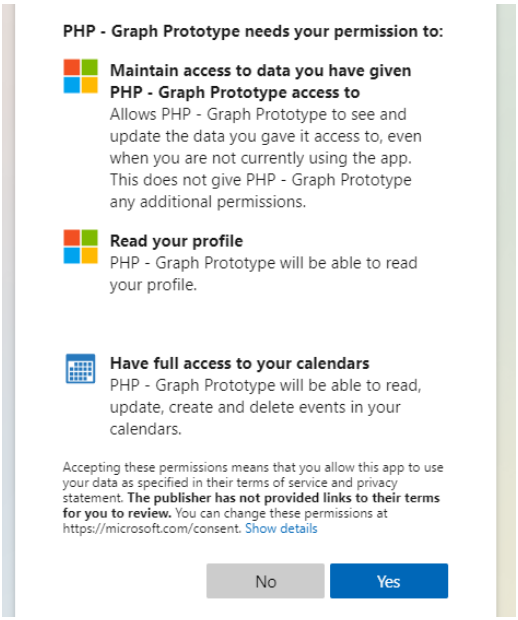

Figura 7. Solicitud de Autorización.

Después de otorgar la autorización solicitada, la aplicación web muestra los datos del usuario, tomados desde el servicio en la nube, y podrá consumir los datos y servicios de la nube para los que cuente con una autorización, hasta que la sesión sea cerrada por el usuario. En la figura 16 se observa el prototipo mostrando el correo y nombre de usuario tomados de la cuenta Microsoft con la que se ha autorizado el acceso.

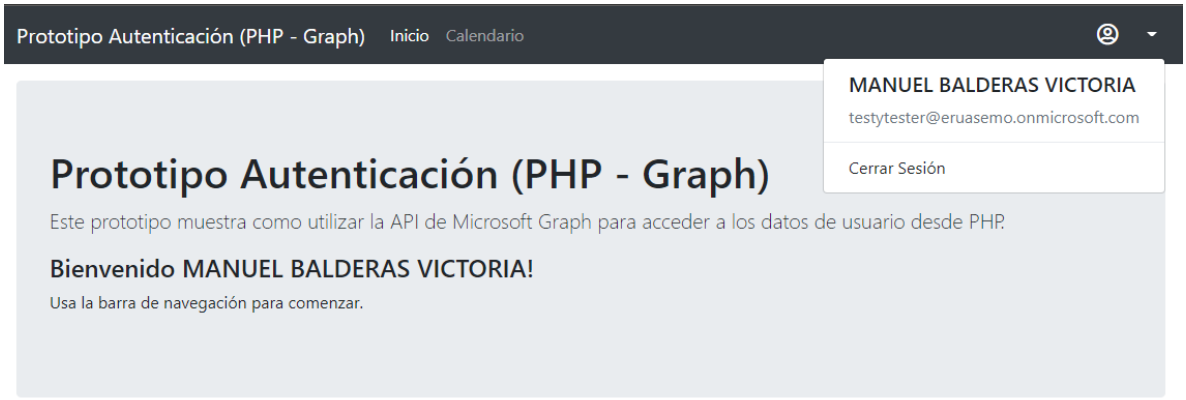

Figura 8. Página de inicio del prototipo con sesión autorizada.

9. CONCLUSIÓN

Gracias a las herramientas y servicios disponibles en el ecosistema web hoy en día, el desarrollo de software se ve optimizado de gran forma, impactando principalmente en el tiempo que toma desarrollar componentes complejos, y en la seguridad que brindan los sistemas desarrollados. La existencia de servicios web, que se encuentran totalmente desacoplados de los sistemas permite acceder a ellos de manera sencilla y utilizarlos de distintas formas en las aplicaciones, los frameworks y las herramientas como Composer permiten tener acceso automatizado al código más reciente de las librerías y componentes más utilizados por la comunidad de desarrollo mundial, hace 10 años era muy difícil pensar en la facilidad con la que se podrían construir sistemas web complejos como se hace hoy en día.

Se pueden contrastar dos formas de trabajar para ejemplificar esto, un desarrollador o un equipo de desarrollo puede trabajar muchos años con un lenguaje como PHP, en múltiples proyectos web, al cabo de unos años habrán acumulado una colección de funciones y clases generales que aplican en sus aplicaciones, estas utilidades "hechas en casa" ayudan a realizar tareas comunes como el inicio de sesión entre otras. También habrán definido una estructura de carpetas para organizar estas funciones y el resto del código de sus sistemas. Sin embargo, todo este trabajo existirá en aislamiento, y ellos serán los únicos responsables del mantenimiento y actualización, así como de añadir nuevas funciones y documentación. En cambio, un equipo de desarrollo que tiene acceso a un framework web services y herramientas estilo Composer, recibe todos los beneficios de años de experiencia en desarrollo, y puede concentrarse en satisfacer las necesidades del negocio, sin preocuparse por lo demás.

Por último, la existencia de protocolos como OpenID y OAuth2.0 permiten al desarrollador común implementar la autenticación de usuarios y autorización de permisos sobre datos y servicios, sin invertir tiempo en diseñar esta interacción, apegándose al estándar siguiendo lo establecido por el flujo del protocolo, con la mayor seguridad con que se puede contar, pues raramente es necesario modificar el código de la implementación, para aprovechar los beneficios de sus actualizaciones.

REFERENCIAS

[1] Armel, J. 2014. "Web application development with Laravel PHP Framework Version 4." Tesis, Media Engineering, Helsinki Metropolia University of Applied Sciences.

[2] Bean, M. (2015). *Laravel 5 Essentials*, 1st edition. Packt Publishing.

[3] Casajus Ramo, A. (2012). Journal of Physics: Conference Series 396 https://doi.org/10.1088/1742-6596/396/5/052019

[4] *Digicert: listo para evitar el robo de información cibernética*. https://forbescentroamerica.com/. Retrieved Jun 18, 2021, from https://forbescentroamerica.com/2020/10/02/digicert-listo-para-evitar-el-robo-de-informacion-cibernetica/

[5] Fett, D. (2021).In H. SCH&MS. Roßnagel (Ed). Open Identity Summit 2021, Gesellschaft für Informatik e.V., Bonn (pp. 71-82). https://dl.gi.de/handle/20.500.12116/36503

[6] Fries, C. 2020. "Security Analysis of Real-Life OpenID Connect." Master's Thesis, Chair for Network and Data Security., Ruhr-Universität Bochum.

[7] Hardt, D. (2012, Oct). *The OAuth 2.0 Authorization Framework*. hjp.at. Retrieved Jun 09, 2021, from https://www.hjp.at/doc/rfc/rfc6749.html

[8] *History of Laravel*. w3schools.in. Retrieved Jun 9, 2021, from https://www.w3schools.in/laravel-tutorial/history/

[9] Johnston, J. (2021). *Build PHP apps with Microsoft Graph*. Microsoft.com. Retrieved Jun 10, 2021, from https://docs.microsoft.com/en-us/graph/tutorials/php

[10] Murray, D. (2020, Febrero 13). *Microsoft Graph API*. Microsoft.com. Retrieved Junio 6, 2021, from https://docs.microsoft.com/es-mx/azure/active-directory/develop/microsoft-graph-intro

[11] Nguyen, Q. H. 2015. "Building a web application with Laravel 5." Bachelor thesis, Business Information Technology, Oulu University of Applied Science.

[12] Singhal, H. (2020, Julio 21). *Protocolos OAuth 2.0 y OpenID Connect en la Plataforma de identidad de Microsoft*. Microsoft.com. Retrieved Junio 8, 2021, from https://docs.microsoft.com/es-es/azure/active-directory/develop/active-directory-v2-protocols

[13] *Usage statistics of server-side programming languages for websites*. w3techs.com. Retrieved Jun 10, 2021, from https://w3techs.com/technologies/overview/programming_language

[14] Wike, R. (2020, Mayo 22). *Autentificación frente a autorización - Microsoft identity platform*. Microsoft.com. Retrieved Junio 6, 2021, from https://docs.microsoft.com/es-es/azure/active-directory/develop/authentication-vs-authorization

[15] Xianjun, C., Zhoupeng, J., Yu, F., & Yongsong, Z. (2017). IOP Conf. Series: Journal of Physics: Conf. Series 910 https://doi.org/10.1088/1742-6596/910/1/012016

Correo de autor: ***m20301148@villahermosa.tecnm.mx***