Nº42 diciembre 2019

CompartiMOSS

REVISTA ESPECIALIZADA EN TECNOLOGÍAS MICROSOFT

Entrevista Rodrigo Díaz Concha

La arquitectura de SharePoint Online

Introducción a Azure Synapse

Productivity Tips con Microsoft Teams: Búsqueda de Información

Contenido

03 Editorial

04 La arquitectura de SharePoint Online

06 ChatBots al rescate

10 Usando Bing News API desde SPFx

13 Como consumir API's securizadas con el Azure Active Directory en nuestros desarrollos de SPFx

17 Entrevista: Rodrigo Díaz Concha

19 Uso de Funciones de Azure con PowerAutomate y PowerApps - Parte 2: Usando OpenAPI en PowerAutomate

23 Seguridad en el Cloud

25 Introducción a Azure Synapse

29 Nuevos planes de licenciamiento de Microsoft Flow

32 Optimizando tu Cosmos DB, mejorando tus consultas con Change Feed

37 Productivity Tips con Microsoft Teams: Búsqueda de Información

39 Entrevista: Raona

43 Nosotros

Comparti MOSS

Staff

CompartiMOSS es una publicación independiente de distribución libre en forma electrónica. Las opiniones aquí expresadas son de estricto orden personal, cada autor es completamente responsable de su propio contenido.

DIRECCIÓN GENERAL

- Gustavo Velez
- Juan Carlos Gonzalez
- Fabian Imaz
- Alberto Diaz

DISEÑO Y DIAGRAMACIÓN

- Santiago Porras Rodríguez

Contacte con nosotros

revista@compartimoss.com
gustavo@gavd.net
jcgonzalezmartin1978@hotmail.com
fabian@siderys.com.uy
adiazcan@hotmail.com

BLOGS
http://www.gavd.net
http://geeks.ms/blogs/jcgonzalez
http://blog.siderys.com
http://geeks.ms/blogs/adiazmartin

REDES SOCIALES
Facebook:
http://www.facebook.com/group.php?gid=128911147140492
LinkedIn:
http://www.linkedin.com/groups/CompartiMOSS-3776291
Twitter:
@CompartiMOSScom

03

Editorial

Este mes, es un mes diferente a todos, un mes que marca el final de un nuevo año y este número, #42, es el último que lanzaremos este 2019. Fue un año con muchas noticias, buenas y malas, se lanzó una nueva versión de nuestro servidor favorito, SharePoint 2019, hemos vistos muchas nuevas funcionalidades en Office 365, Azure y el ecosistema de Microsoft que nos mantiene ocupados como profesionales día a día. Nos entristeció la pérdida de un colega y amigo, Ricardo, que participó activamente como colaborador de la revista y el #40 de este año fue en su homenaje, por su amistad y trayectoria.

El 2020 estará lleno de desafíos, trabajaremos en cada número para hacerles llegar todas las novedades tecnológicas más importantes, seguiremos en la búsqueda de nuevos autores que deseen colaborar en cada número, incorporaremos nuevos temas y sobre todo trabajaremos arduamente para que cada número sea mejor que el anterior.

Les queremos desear que tengan una muy feliz navidad y un excelente comienzo de año junto a sus seres queridos.

¡Muchas gracias por leernos estos 10 años, nos vemos en el 2020!

El Equipo Editorial de CompartiMOSS

4

La arquitectura de SharePoint Online

Siempre he sido un enamorado de la arquitectura de SharePoint Server, sobre todo cuando en la versión 2010 nació el concepto de aplicación de servicio.

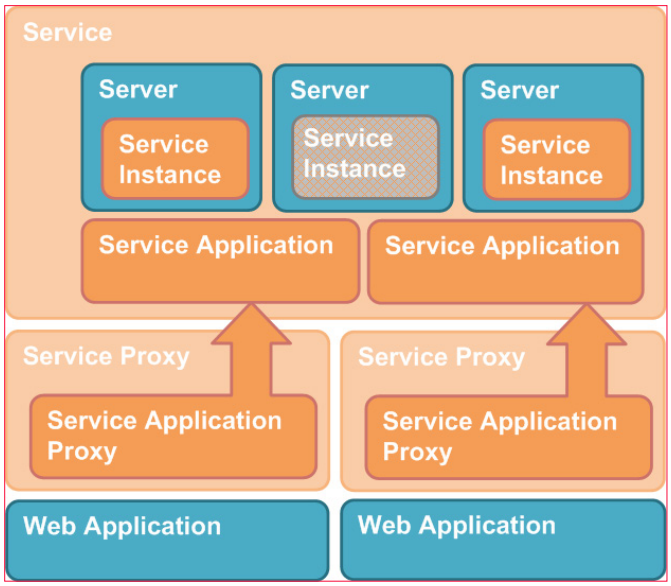

Imagen 1. Topología de múltiples aplicaciones de servicio

Gracias a esta arquitectura, podíamos ejecutar diferentes instancias de un servicio en diferentes servidores, podíamos escalar horizontalmente las aplicaciones de servicio. Para que esto funcionara, cada aplicación web de SharePoint ejecutaba un proxy que se encargaba de enrutar las peticiones al servidor más adecuado que estuviera ejecutando la instancia de la aplicación de servicio. Simplemente, una revolución para nuestro servidor favorito.

En 2011, Microsoft empieza a construir Office 365 y SharePoint Online hereda esta arquitectura desde la versión SharePoint 2013. Todo administrador de SharePoint se puede hacer la idea de lo que significa esto para Microsoft, tener que administrar granjas gigantescas de SharePoint con una topología que no es monolítica pero no está pensada ni diseñada para ejecutarse en el cloud, lo que implica grandes esfuerzos en administración para mantener los sistemas con el nivel de SLA que se le exigía.

Entre 2015 y 2016, el equipo de producto de SharePoint empieza a trabajar en la primera versión de SharePoint diseñada para ejecutarse en cloud y SharePoint 2016 (OnPremises) fue la primera versión que pudimos instalar en nuestros servidores que nacía de una rama del código de SharePoint Online, y no al revés. ¿Esto que implicaciones tenía? Pues a nivel de servicios clásicos de SharePoint no mucho, porque simplemente no han cambiado, más bien, se deprecaron algunos que no iban a llevar ni nuevas funcionalidades ni más soporte por parte de Microsoft. Lo relevante es que ya veíamos servicios en SharePoint Online que no íbamos a tener en SharePoint Server, por ejemplo, Delve.

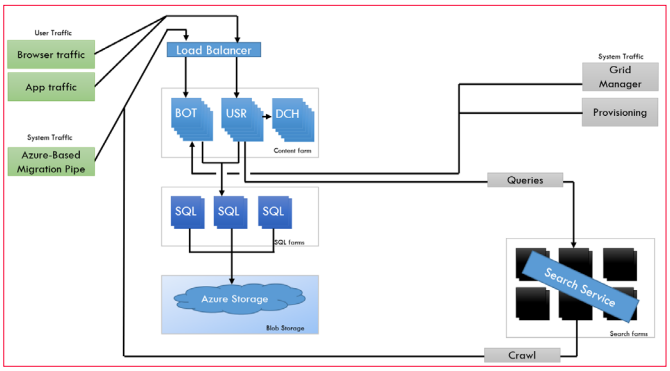

Imagen 2. Arquitectura de SharePoint Online en 2016.

Como podemos ver en la imagen anterior, se empezaron a desarrollar servicios fuera del core de SharePoint, como puede ser el servicio de búsqueda, y la granja se distribuía en tres roles:

- BOT -> donde se administraba todo el tráfico que se produce en background.
- USR -> encargado de todo el tráfico del usuario que usa SharePoint.
- DCH -> un servidor de caché en memoria.

Con esta topología, los SQL Server eran más independientes, los documentos se almacenaban en un Blob Storage y los servicios eran más independientes para escalar y más fiables en ejecución. Lo que permitía incorporar nuevos servicios de forma totalmente transparente a SharePoint.

"Gracias a esta arquitectura, podíamos ejecutar diferentes instancias de un servicio en diferentes servidores, podíamos escalar horizontalmente las aplicaciones de servicio"

La nueva generación de la arquitectura de SharePoint viene a partir de esta entrega de 2016, una nueva generación preparada para ejecutarse en más de 50 datacenters en 20 localizaciones diferentes en todo el mundo. Una evolución de la versión anterior, con componentes mucho más

desacoplados entre ellos y con ejecución en un clúster de Service Fabric como microservicios.

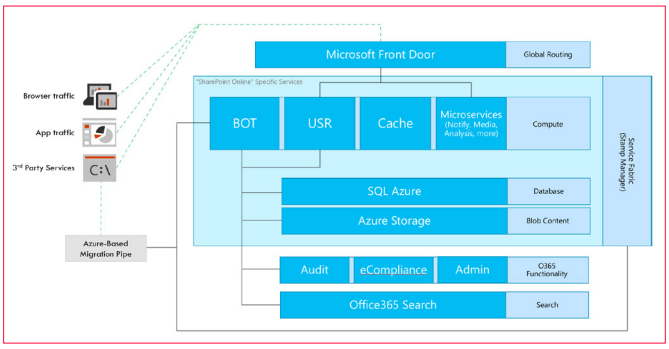

Imagen 3. Arquitectura de SharePoint Online.

Los servicios específicos de SharePoint Online se ejecutan como microservicios con una topología parecida a la versión 2016, con los roles de BOT, USR, Cache. Los nuevos servicios específicos se desarrollan y ejecutan con esta arquitectura en un rol de Microservicio, y todo se almacena en SQL Azure (servicio PaaS) o en Azure Storage, pero esta vez ya de forma independiente uno del otro, son los servicios los que se encargan de ir a los datos relacionales o al almacenamiento de blobs de Azure.

Con esta misma arquitectura, el resto de los servicios que se desarrollan para Office 365 que no son específicos de SharePoint, son implementados en SharePoint usando el rol de BOT o el de USR, lo que permite incluir, de una forma muy simple y estándar, cualquier servicio como Microsoft Search o que se puedan consumir servicios de SharePoint en el resto de los productos, como, por ejemplo, el propio SharePoint.

Por último, pero no menos importante, tenemos Microsoft Front Door, el encargado de dar un punto de entrada seguro a todos los clientes que usan SharePoint Online y enrutador de forma adecuada al servicio correspondiente.

Uno de los puntos fuertes de SharePoint se basa en la capacidad de personalización que hemos tenido, no sólo de la interfaz de usuario, pensad en le cantidad de líneas de código que existen para integrar SharePoint con otras plataformas, para generar aplicaciones de negocio a partir del modelo de objetos de SharePoint. Todo esto siempre ha sido posible gracias al modelo de objetos, gracias a la API REST o a los servicios ASMX que nos permitían hacer cualquier tipo de consulta, independientemente del esquema que el usuario había pensado. Si, lo sé, la tabla AllDocuments sufría de lo lindo con estas consultas y el DBA siempre maldecía a SharePoint porque su SQL Server no podía con esas mega-consultas no optimizadas adecuadamente. La recomendación es que usemos Microsoft Graph porque nos abstrae de la API de SharePoint y sus consultas están optimizadas para que, por ejemplo, no hagamos un GetByTitle si podemos hacer un GetById que es muchísimo más eficiente.

"Uno de los puntos fuertes de SharePoint se basa en la capacidad de personalización que hemos tenido, no sólo de la interfaz de usuario"

SharePoint Online seguirá evolucionando, seguro que ya están trabajando en mejorar esta arquitectura que ha conseguido atender a millones de usuarios que usan la plataforma y las aplicaciones que hemos desarrollado usando su API. Personalmente, me ha parecido una buena aplicación de microservicios, algo que en SharePoint 2010 se venía venir, pero no estaba madura la tecnología.

ALBERTO DIAZ MARTIN
MVP Azure
adiazcan@hotmail.com
@adiazcan
http://blogs.encamina.com/por-una-nube-sostenible/

6

ChatBots al rescate

Casi siempre a la hora de buscar información dentro del ámbito empresarial acabamos con dos resultados:

- Perdiendo muchísimo tiempo sin dar con un buen resultado.
- Frustrados y al rato dejar de buscar y pedirlo a algún compañero.

¿Y si existiese una tercera opción que nos facilite las cosas y que la búsqueda de información no se convierta en un total fracaso?

Esa solución tiene nombre, es Microsoft Bot Framework y Office 365.

En este articulo vamos a detallar más a fondo desde un enfoque funcional, técnico y sin perder de vista el lado de negocio las funcionalidades que Azure y Office 365 en conjunto, nos ofrecen para distribuir información en un formato unificado que casi siempre llamamos “intranet corporativa”.

“Dentro de las configuraciones disponibles tenemos un apartado muy importante, la plantilla con la que crearemos el Bot”

Para comprender mejor este enfoque debemos hacer un inventario de las herramientas tecnológicas a usar y separarlas en dos grandes bloques:

Azure:

- Azure BotFramework V4 (Node.js usando TypeScript): es donde crearemos y alojaremos nuestro Bot.
- Azure Active Directory (registro de aplicaciones): plataforma de identidad desde la cual podremos conceder permisos al ChatBot para acceder a los recursos necesarios.
- Cognitive Services (LUIS y QnAMaker): como fuente de datos y entrenamiento para el Bot.

Office 365:

- Office 365 (SharePoint Online): plataforma base sobre la cual esta alojada nuestra Intranet.
- Extensión de SharePoint Framework en SharePoint Online: canal principal para conversación directa con el Bot.
- Microsoft Teams: como canal añadido de comunicación e interacción con el Bot.

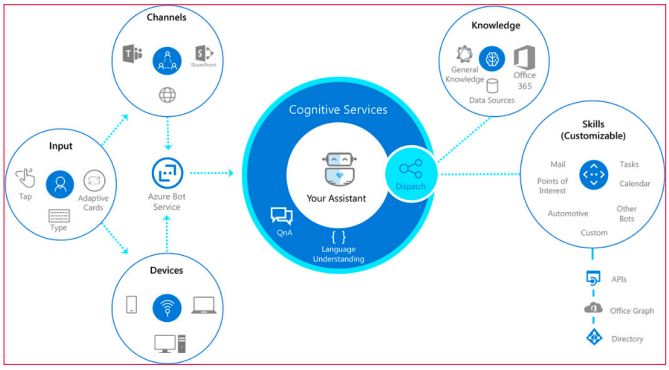

Imagen 1.- Diagrama de la arquitectura de Microsoft Bot Framework.

Bot Services

Dentro del portal de Azure, encontraremos bajo la categoría de “Bot Services” la forma de crear un nuevo “Web App Bot” con la configuración que mejor se ajuste a nuestras necesidades.

Dentro de las configuraciones disponibles tenemos un apartado muy importante, la plantilla con la que crearemos el Bot. Como base tenemos varias opciones en función del lenguaje SDK que queremos utilizar (C# o Node.JS). En ambos casos podemos crear un Bot vacío (Echo Bot) desde cero o un Bot que nos facilitará el enlace con los servicios cognitivos de LUIS y QnAMaker (Basic Bot).

En este caso vamos a utilizar la plantilla basada en Node.JS y Basic Bot. Por defecto este Bot se creará en lenguaje JavaScript, teniendo la posibilidad de descargar el código completo del mismo y empezar a añadirle funcionalidades según se necesiten.

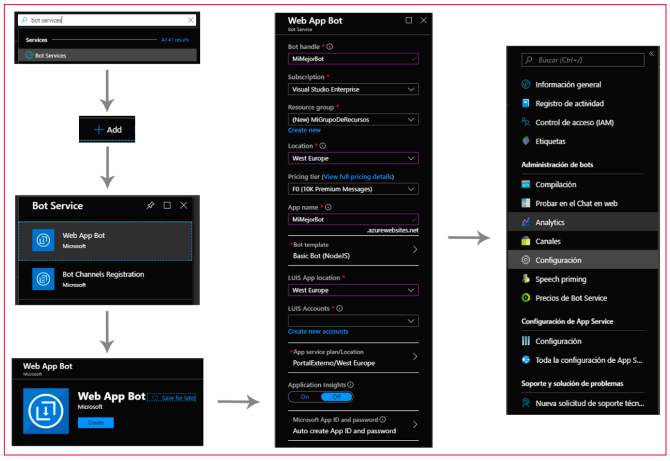

Imagen 2. – Flujo de creación del Web App Bot.

Después de la creación podremos acceder a los apartados de Canales, Configuración del Bot en sí, Configuración del App Service que hay por debajo, etc. En definitiva, nuestro Bot realmente será un App Service con una API Rest a la que haremos peticiones a través de un endpoint del tipo "/api/messages" y este endpoint nos devolverá las respuestas en "formato conversación".

(*) En nuestro caso hemos optado por darle un poco más de "emoción" al asunto y hemos empezado con un proyecto vacío de Bot Framework y poco a poco bajo los patrones de la versión 4 del mismo, hemos ido añadiéndole las partes necesarias, así como endpoint Rest como punto de entrada del Bot, diálogos, clases de ayuda y demás. Como preferencia en cuanto al lenguaje de programación hemos usado TypeScript (que al final se convertirá en JavaScript después del compilado final).

"Después de la creación podremos acceder a los apartados de Canales, Configuración del Bot en sí, , Configuración del App Service que hay por debajo, etc"

Como último paso después de la creación del Web App Bot necesitamos ir al apartado de "Registro de aplicaciones" en Azure Active Directory y otorgar permisos de acceso a los diferentes recursos, así como Graph API y SharePoint Online. Como veíamos antes, nuestro bot es una App Service, por lo tanto, esta App se registra automáticamente en Azure AD al crear el Bot.

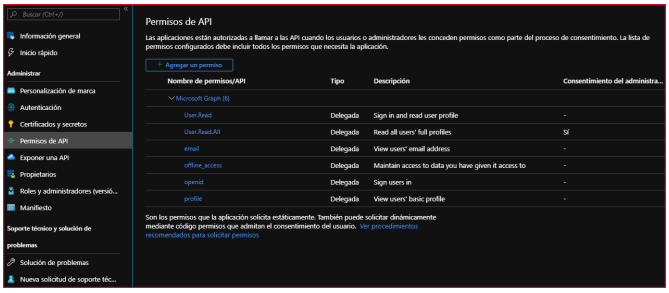

Imagen 3. – Permisos para la App registrada en Azure Active Directory.

Office 365

Nuestra intranet corporativa estará alojada en SharePoint Online, sirviendo este como repositorio de noticias, documentación y punto de enlace con las aplicaciones empresariales ad-hoc. Como ejemplo vamos a utilizar un sitio de comunicación con varias páginas y noticias contribuidas, siendo este la fuente de información principal para el Bot que luego conectaremos con esta fuente de datos mediante el servicio de búsqueda de SharePoint Online y MS Graph.

Uno de los puntos fuertes de SharePoint Online es la segmentación de los datos mediante permisos en bibliotecas, documentos o páginas. De esta manera el Bot nos dará solo información a la que tenemos acceso, es decir, trabajará para nosotros como si fuéramos nosotros mismo buscada información. Esto lo vamos a conseguir "delegando" al Bot nuestros accesos a la información, mediante tokens de forma segura y transparente para el usuario.

SharePoint Framework

Desde una extensión de SharePoint Framework presente en todo momento en cualquier página (en la parte inferior derecha) abriremos un panel lateral dentro del cual vamos a realizar la conexión con el Bot. Gracias a la librería de "botframework-webchat" podemos integrar el componente de ReactWebChat que nos proporcionará el canal de comunicación rápido y directo con nuestro Bot desde el lado de Office365.

```
public render() {
    // Be careful, the user Id is mandatory to be able to use the bot state service (i.e privateConversation
    return (
        <div className={styles.banner}>
            <ActionButton onClick={this._login} checked={true}
                iconProps={{ iconName: "Robot", className: styles.banner__chatButtonIcon }}
                className={styles.banner__chatButton} />
            <Panel
                isOpen={this.state.showPanel}
                type={PanelType.medium}
                isLightDismiss={true}
                onDismiss={() => this.setState({ showPanel: false })}
            >
                {this.state.isBotInitializing ?
                    <div className={styles.overlayList} >
                        <Spinner size={SpinnerSize.large} label={strings.GraphBotInitializationMessage} />
                    </div>
                    :
                    <ReactWebChat
                        directLine={this._botConnection}
                        userID={
                            this.props.context.pageContext.user.email
                        }
                        locale={this.props.context.pageContext.cultureInfo.currentUICultureName}
                        attachmentMiddleware={this.attachmentMiddleware}
                    />
                }
            </Panel>
        </div>
    );
}
```

Imagen 4. – Método de renderizado de la extensión de SPFX.

Podemos observar en la anterior captura, el parámetro "directLine={this._botConnection}", responsable de la creación del canal de comunicación con nuestro Bot. En esta misma variable, vamos a definir el evento mediante el cual le enviaremos al Bot nuestro token de acceso, nada más pulsar el botón del Bot y abrir el panel de conversación.

```
/**
 * Initialize the chat bot by sending the access token of the current user
 * @param token The access token of the current user
 */
private _sendAccessTokenToBot(token: string): void {
    // Using the backchannel to pass the auth token retrieved from OAuth2 implicit grant flow
    this._botConnection.postActivity({
        type: "event",
        value: {
            accessToken: token,
            userDisplayName: this.props.context.pageContext.user.displayName, // For the welcome message
            pageUrl: this.props.context.pageContext.site.absoluteUrl
        },
        from: {
            // IMPORTANT (1 of 2): USE THE SAME USER ID FOR BOT STATE TO BE ABLE TO GET USER SPECIFIC DATA
            id: this.props.context.pageContext.user.email
        },
        name: "LoginTokenUpdate" // Custom name to identify this event in the bot
    }).subscribe(
        id => {
            // Show the panel only if the event has been well received by the bot (RxJs format)
            this.setState({
                isBotInitializing: false
            });
        },
        error => {
            Logger.write(Text.format("[GraphBot_sendAccessTokenToBot]: Error: {0}", error));
        }
    );
}
```

Imagen 5. – Definición de la variable DirectLine del Bot.

Aprovechando las variables de contexto de las extensiones SPFx utilizaremos el AadTokenFactoryProvider para crear un nuevo token de acceso a través del método ".getToken()" pasándole como parámetro nuestra aplicación (el Client ID del Bot) registrada en Azure AD.

```
private async _invokeTokenEndPoint(): Promise<any> {
    var tokenProvider = await this.props.aadTokenProviderFactory.getTokenProvider();

    //get the token for the AadAppRegistrationID
    return await tokenProvider.getToken(this.props.clientID);
}
```

Imagen 6. – Recuperación del token para la aplicación registrada en Azure AD.

```
(method) AadTokenProvider.getToken(resourceEndpoint: string): Promise<string>

Fetches the AAD OAuth2 token for a resource if the user that's currently logged
in has access to that resource.

The OAuth2 token should not be cached by the caller since it is already cached
by the method itself.

@param resourceEndpoint — the resource for which the token should be
obtained An example of a resourceEndpoint would be
https://graph.microsoft.com

@returns — A promise that will be fullfiled with the token or that will reject with
an error message
getToken(this.props.clientID);
```

Imagen 7. – Definición del método getToken().

Si nos fijamos en la definición de este método, nos indica que el usuario debe tener permisos sobre la aplicación registrada (nuestro Bot). Estos permisos los conseguiremos a la hora de desplegar la extensión en el catálogo de aplicaciones de SharePoint Online. Será necesaria la aprobación de estos permisos por parte de un administrador del tenant de Office365.

Con los siguientes parámetros en el fichero "package-solution.json" de nuestra extensión podremos crear una solicitud de permisos al desplegar la extensión:

```
"webApiPermissionRequests": [{
    "resource": "GUID de mi aplicacion registrada en Azure AD (el bot)",
    "scope": "user_impersonation"
  }
],
```

Imagen 8. – Petición de permisos sobre Office 365 desde extensión de SPFx.

Gracias a estos pasos hemos conseguido conectarnos al Bot desde la extensión de SPFx. Ya tenemos el canal DirectLine abierto con el Bot. El siguiente paso será ir al código del Bot y adaptarlo para que sepa hacer uso de toda la información que le estamos enviando.

Bot Framework V4

En el proyecto de Bot Framework, tendremos una estructura de archivos bastante común, con un punto de entrada de las peticiones hacia la API Rest, unos eventos dentro de los cuales procesaremos esas peticiones y la devolución de la respuesta hacia el usuario en diferentes formatos: texto, imagen, AdaptiveCard, carrousel con contenidos, etc. El procesado de las peticiones se realizará dependiendo del tipo de las mismas. Identificaremos el tipo de evento que ha desencadenado la petición y procedemos a darle una respuesta.

En este ejemplo hemos identificado varios tipos de eventos (existen más) y les hemos intentado dar un procesado especifico:

```
async processActivityType(context: TurnContext) {
    const dialogContext = await this._dialogs.createContext(context);
    await dialogContext.continueDialog(); //continue dialog if it's response

    switch (dialogContext.context.activity.type) {
        case ActivityTypes.Message:  // normal message from user -> pass to LUIS and QnAMaker
            await this.processInput(dialogContext);
            break;
        case ActivityTypes.Invoke:  // MS Teams login handler
            if (!dialogContext.context.responded) {
                await dialogContext.beginDialog("Login");
            };
            break;
        case ActivityTypes.ConversationUpdate:  // Welcome message on new conversation
            await this.sendWelcomeMessage(context);
            break;
        case ActivityTypes.Event:  // SharePoint Extension Token interception
            if (dialogContext.context.activity.name == "LoginTokenUpdate") {
                await this.setConversationProperties(context, dialogContext);
            }
            break;
        case ActivityTypes.MessageReaction:  // MS Teams message reaction (Like, Heart, Smile, etc...)
            dialogContext.context.activity.reactionsAdded.map(async (value: MessageReaction, index: number, allreactions: MessageReaction[]) => {
                if (value.type === 'like' || value.type === 'heart') {
                    await dialogContext.context.sendActivity(`Gracias \u083D\uDE0A .`);
                }
                else {
                    await dialogContext.context.sendActivity(`Valor: ${value.type}`);
                }
            });
            break;
        default:  //default, unknown event type, show on screen
            await dialogContext.context.sendActivity(`[${dialogContext.context.activity.type}]-type activity detected.`);
    }
    //save conversation and user state
    await this._conversationState.saveChanges(context);
    await this._userState.saveChanges(context);
}
```

Imagen 9. – Switch para identificar los tipos de evento que desencadenan una acción.

En el anterior código, una de las partes interesantes es el evento personalizado que hemos interceptado y a través del cual recibimos el token del usuario desde la extensión de SharePoint Online. Este token lo iremos pasando a los diferentes diálogos que lo necesiten para realizar las peticiones oportunas. Como ayuda hemos utilizado la librería de "@pnp/js" y asi poder facilitar las llamadas desde el Bot hacia las APIs de Office 365 (búsqueda de SharePoint y MS Graph).

Gracias a los servicios cognitivos podemos entrenar un modelo que reconozca las peticiones del usuario, para que nuestro Bot tenga la capacidad de entender que se le esta pidiendo y recuperar palabras clave de cada una de esas solicitudes.

Como ejemplo podemos pensar en una solicitud del tipo "quiero ver las ultimas noticias". LUIS mediante entrenamiento previo sabrá que el usuario quiere conocer las ultimas noticias y le devolverá al Bot la intención del usuario. El Bot a su vez realizará las llamadas necesarias a SharePoint Online para recuperar las ultimas noticias (siempre respetando el token del usuario y a su vez los permisos del mismo). De esta manera el usuario solo recibirá la información que va dirigida hacia él.

Veamos el resultado final en un entorno integrado:

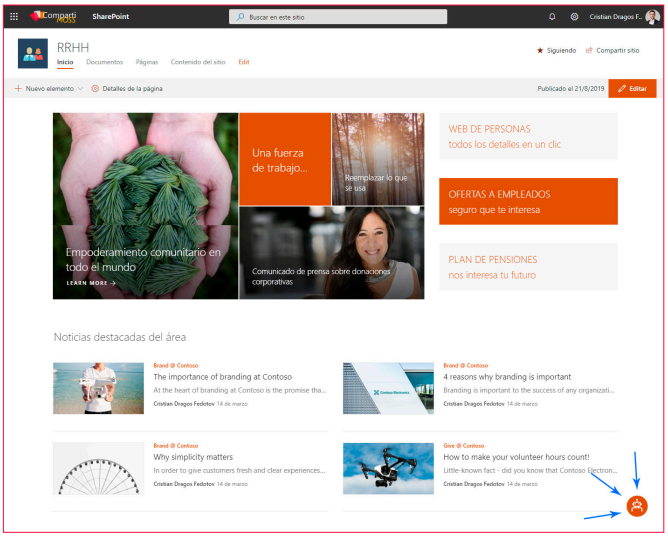

Imagen 10. – Página de SharePoint Online con la extensión de Bot en la parte inferior derecha.

"Gracias a los servicios cognitivos podemos entrenar un modelo que reconozca las peticiones del usuario, para que nuestro Bot tenga la capacidad de entender que se le esta pidiendo"

En este caso vamos a poner varios casos de uso, expuestos en las siguientes capturas de pantalla:

- Petición de información genérica tipo preguntas frecuentes (fuente de datos de QnAMaker).
- Petición de las últimas noticias presentes en la intranet con respuesta en formato carrousel de AdaptiveCards (interpretación por LUIS y recuperación de información desde servicio de búsqueda de SharePoint Online).
- Petición de la/las últimas nóminas del usuario (interpretación por LUIS y petición de datos a servicio ad-hoc dentro del entorno empresarial).
- Petición de vacaciones con respuesta en Adaptative Card (envío de formulario al usuario y registro de datos introducidos en sistema interno de la empresa).

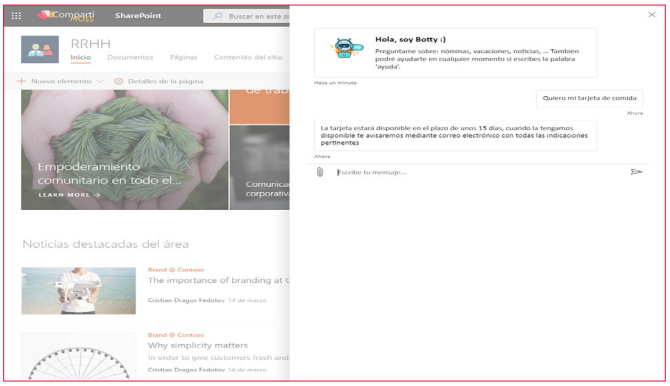

Imagen 11. – Petición de información genérica al Bot con respuesta de QnAMaker.

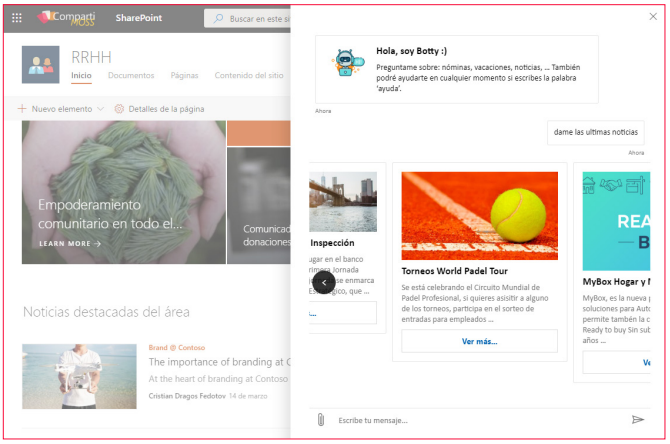

Imagen 12. – Petición de las ultimas noticias y respuesta en carrousel de AdaptiveCards.

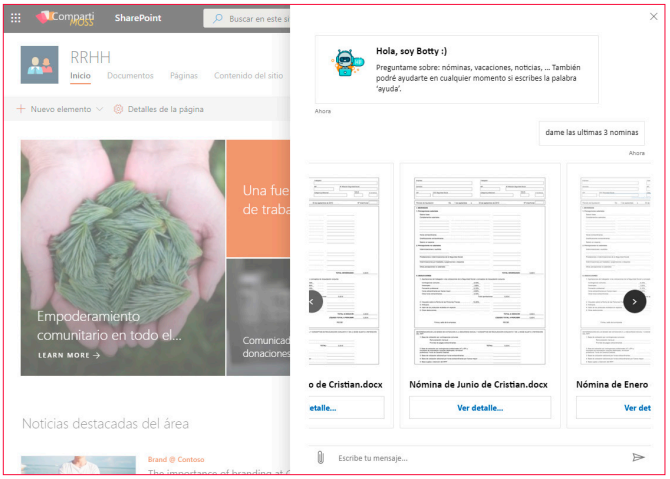

Imagen 13. – Petición de las ultimas 3 nominas y respuesta en formato carrousel.

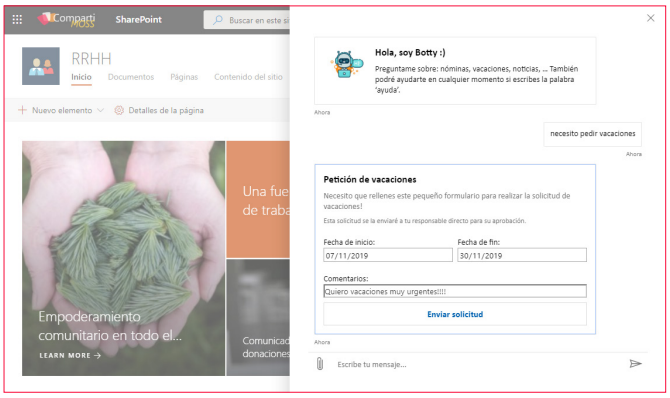

Imagen 14. – Petición de vacaciones en formato AdaptiveCard y envío de solicitud.

Otro canal de comunicación podría ser Microsoft Teams, desplegando el Bot dentro del entorno empresarial a través del App Studio de Teams. El formato seguirá siendo el mismo y las conversaciones se mantendrán de la misma manera, incluso podremos interactuar con los mensajes mediante Likes y otras emociones disponibles en Teams.

En este ejemplo podemos ver la misma conversación mantenida con el Bot y la interacción con uno de los mensajes recibidos dándole al botón de Like y la respuesta del Bot de agradecimiento a esa interacción.

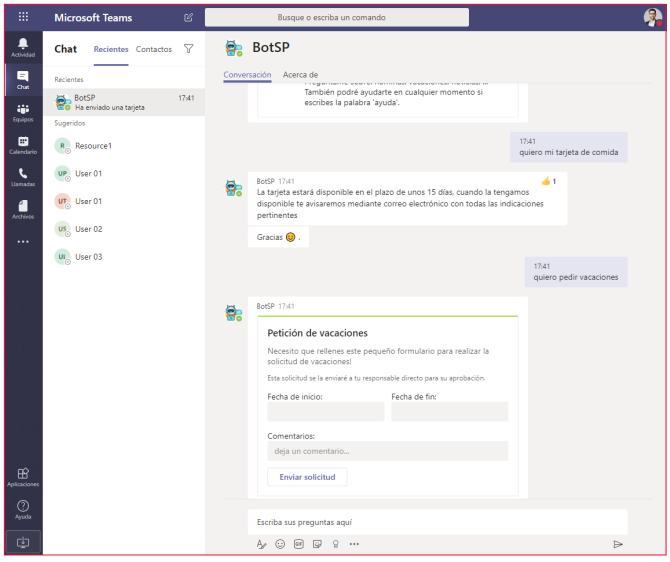

Imagen 15. – Conversación con el Bot desde MS Teams e interacción con los mensajes recibidos.

Conclusiones

Como hemos visto, es posible crear herramientas que complementen el uso de Office 365 y quizás nos facilite el trabajo en el día a día. En este articulo se describen varios casos de uso más comunes fácilmente ampliables y adaptables a las necesidades de nuestros clientes y usuarios, gracias a las herramientas que ofrece Office 365 en conjunto con Microsoft Azure.

CRISTIAN DRAGOS FEDOTOV
Solutions Architect / Office365 & SharePoint at Everis
cristianfedotov@gmail.com
https://www.linkedin.com/in/cristianfedotov/

10

Usando Bing News API desde SPFx

En este artículo vamos a hacer uso de la Bing News API desde una extensión de SPFx. El caso de uso, bastante común en muchas organizaciones, es poder sacar información relacionada, publicada en la web, con el artículo publicado en nuestra Intranet. Imagina un artículo que habla sobre un cliente concreto, o un sector de nuestra organización, y queremos ofrecer noticias publicadas recientemente y relacionadas con ese cliente o sector.

Antes de entrar en materia, aquí podéis ver el resultado final, de lo que vamos a ver en el artículo, en la imagen siguiente:

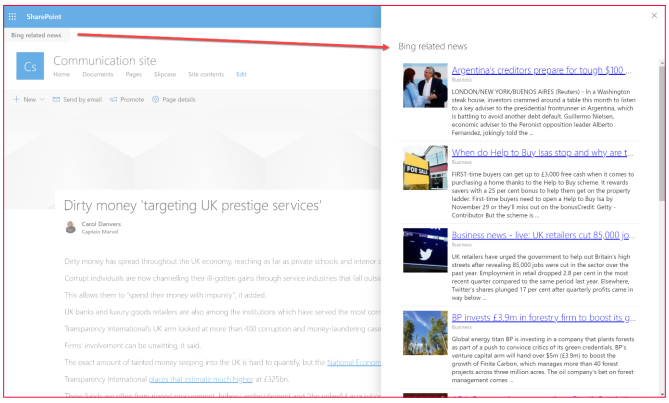

Imagen 1.- WebPart a desarrollar.

Registrando el servicio en el portal de Azure

Lo primero para poder utilizar la Bing News Search API, es registrar el servicio en el portal de Azure:

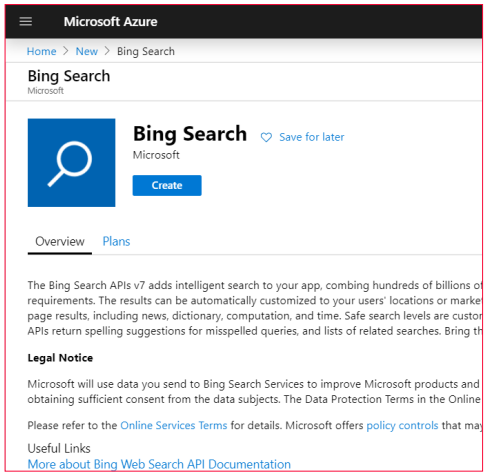

Imagen 2.- Registro de la API en el Portal de Azure.

Una vez creado el servicio, tendremos nuestra típica API Key con la que ya podremos llamar a la API:

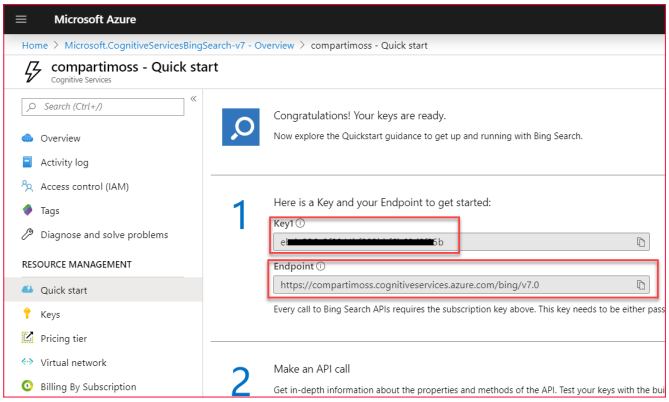

Imagen 3.- Key requerida para usar la API.

Nota: Como ves en la imagen anterior, además de la Key, se crea un Endpoint. Dicho endpoint lo puedes usar para hacer la llamada a la API, pero también funciona con el endpoint genérico: https://api.cognitive.microsoft.com/bing/v7.0 Aconsejo utilizar el endpoint genérico, ya que al moverte entre diferentes entornos, tendrás menos cosas que configurar en tu código.

Consultando Bing News Search API

Existen diferentes queries que podemos lanzar a la Search API.

Obteniendo News "en general"

Esto equivaldría a la búsqueda más común de cualquier portal de búsquedas (Bing, Google…), donde se lanza un query a todo el motor, y éste nos devuelve los resultados. Para hacer una búsqueda de este tipo, haríamos la siguiente request (en este ejemplo, el término de búsqueda sería: sharepoint framework):

```
GET https://api.cognitive.microsoft.com/bing/v7.0/news/search?q=sharepoint framework&mkt=en-us HTTP/1.1
Ocp-Apim-Subscription-Key: 123456789ABCDE
User-Agent: Mozilla/5.0 (compatible; MSIE 10.0; Windows Phone 8.0; Trident/6.0; IEMobile/10.0; ARM; Touch; NOKIA; Lumia 822)
X-Search-ClientIP: 999.999.999.999
X-Search-Location: lat:47.60357;long:-122.3295;re:100
X-MSEdge-ClientID: <blobFromPriorResponseGoesHere>
Host: api.cognitive.microsoft.com
```

Como podemos ver, usamos el endpoint search y pasamos en la querystring el parámetro q, con el término a buscar

"…/search?q=sharepoint framework".

En el término de búsqueda, también podemos especificar un dominio específico sobre el que se buscará. Ejemplo: "…/search?q=sharepoint framework site:compartimoss.com"

Fijaros como en las cabeceras de la petición, debemos especificar una cabecera Ocp-Apim-Subscription-Key, con el valor de la Key que hemos obtenido al registrar el servicio en Azure

En todas las peticiones, también podemos especificar un market sobre el que se acotará la búsqueda, por ejemplo, podemos especificar como market, el local de España, y obtener así las noticias de España. Ejemplo: https://api.cognitive.microsoft.com/bing/v7.0/news?q=&mkt=es-es

Obteniendo noticias por categoría

Podemos acotar la búsqueda a una categoría concreta. Ahora mismo Bing News ofrece diferentes categorías, que van a depender del market. En este enlace tenéis los listados de cada categoría según su market: https://docs.microsoft.com/en-us/rest/api/cognitiveservices-bingsearch/bing-news-api-v7-reference#news-categories-by-market

De momento no todos los markets tienen disponible esta funcionalidad. Así mismo, las categorías varían según el market. Por ejemplo, USA, tiene un listado de categorías muy completo (e incluso jerárquico en 2 niveles), sin embargo, las categorías para el market australiano, son muy reducidas, y otros markets, como el español, no tienen disponible esta búsqueda.

Para ver todas las posibles tipos de Queries, podéis visitar el siguiente artículo: https://docs.microsoft.com/en-us/azure/cognitive-services/bing-news-search/concepts/search-for-news

Paginando resultados

Para realizar paginación, primero tenemos que fijarnos en el campo de la respuesta totalEstimatedMatches


```
{
    "_type": "SearchResponse",
    "queryContext": {
        "originalQuery": "sailing dinghies"
    },
    "webPages": {
        "webSearchUrl": "https://www.bing.com/search?q=sailing+dinghies",
        "totalEstimatedMatches": 3720000,
        "value": [
            {
                "id": "https://api.cognitive.microsoft.com/api/v7/#WebPages
                "name": "What Is Dinghy Sailing?",
                "url": "https://www.wisegeek.com/what-is-dinghy-sailing.htm
```


Al realizar la query, podemos usar el parámetro count, para especificar el número máxima de elementos retornados que queremos, así como el parámetro offset para empezar los resultados a partir de ese valor. Ejemplo:

https://api.cognitive.microsoft.com/bing/v7.0/search?q=sharepoint&count=5&offset=45&mkt=en-us

Invocando la API desde SPFx

Una vez entendido qué tipo de queries podemos ejecutar contra la API, podemos ya ver cómo hacerlo desde una extensión de SPFx.

Para el ejemplo, vamos a usar la búsqueda basada en una categoría y market. El market lo estableceremos como un valor fijo para el mercado de UK, mientras que la categoría, la sacaremos de un campo de la página, que será informado en el momento de crear la página en SharePoint. Este campo es de tipo Taxonomy, y estará vinculado a un TermSet que contiene la lista de posibles categorías que tenemos disponibles en el mercado de UK.

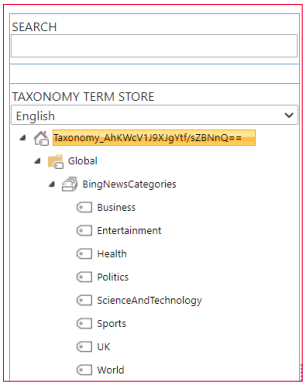

Imagen 4.- Categoría de Noticias en el Term Set de SPO.

A continuación, os muestro las partes de código más importantes.

Primero de todo, necesitamos saber qué Categoría tiene la página actual. Para ello, haremos una consulta a la API REST de SharePoint:

```
const getItemByIdEndpoint: string =
    `${this._baseUrl}/_api/web/lists('${this._listId}')/GetItemB-
yId(${this._listItemId})?$select=id,title,NewsCategory`;

const response: SPHttpClientResponse = await this._spHttp-
Client.get(getItemByIdEndpoint, SPHttpClient.configurations.
v1);

const responseJson: any = await response.json();
```

Una vez obtenida la categoría, ya podemos llamar a la Bing News Search API:

```
const bingEndpoint: string = `https://api.cognitive.microsoft.
com/bing/v7.0/news?mkt=en-GB&category=${category}`;
```

Preparamos las Headers de la petición para que incluyan la API Key

```
const requestHeaders: Headers = new Headers();
requestHeaders.append("Content-type", "application/json");
requestHeaders.append("Cache-Control", "no-cache");
requestHeaders.append("Ocp-Apim-Subscription-Key", this.
props.bingSearchApiKey);
```

Finalmente, utilizaremos el HttpClient del framework para lanzar la petición

```
const response: HttpClientResponse = await this._httpClient.
get(
    bingEndpoint,
    HttpClient.configurations.v1,
    httpOptions);
const responseJson: any = await response.json();
```

El resultado de la petición (JSON), lo mapeamos a una interfaz custom que nos hemos definido con la información del artículo que más nos interesa:

```
const relatedNews: INewsArticle[] = responseJson.value.ma-
p((item: any) => {
    const article: INewsArticle = {
     name: item.name,
     category: item.category,
     description: item.description,
     thumbnailUrl: item.image.thumbnail.contentUrl,
     datePublished: new Date(item.datePublished),
     url: item.url
    };
    return article;
});
```

Tenéis todo el ejemplo completo en el repositorio de GitHub del PnP

https://github.com/SharePoint/sp-dev-fx-extensions/tree/master/samples/react-application-page-related-bing-news

¡Hasta el próximo artículo!

LUIS MAÑEZ
Cloud Architect en ClearPeople LTD
@luismanez
https://medium.com/inherits-cloud

13

Como consumir API's securizadas con el Azure Active Directory en nuestros desarrollos de SPFx

Las posibilidades dentro del desarrollo en SharePoint han cambiado mucho y nos permite un amplio abanico de opciones. En muchos artículos podemos ver como empezar con el desarrollo de SPFx y entender toda la solución que se plantea. Ahora bien, dentro de esta nueva forma de desarrollar, se es cada vez más habitual tener que consumir un servicio de terceros para poder mostrar información. El abanico de servicios de terceros puede ser muy amplio: aplicaciones tipo Facebook, Twitter, LinkedIn, o bien un servicio de la propia compañía en el que utilice una autenticación contra el mismo Azure Active Directory. En este articulo vamos a centrarnos en este último punto tenemos una API que esta securizada mediante el Azure Active Directory y como la podemos consumir desde nuestro desarrollo en Spfx sin que el usuario tenga que volver a autenticarse.

"cómo podemos consumir desde nuestro desarrollo en Spfx un API securizado sin que el usuario tenga que volver a autenticarse"

Creando la API Custom en .NET Core con la autenticación contra el Azure Active Directory

En primer lugar, crearemos una API. Para este ejemplo vamos a utilizar .NET Core (pudiéndose utilizar la tecnología que el lector requiera). Para ello lo haremos desde la línea de comando de .NET Core y con poner la siguiente instrucción tendremos un "template" listo para empezar a desarrollar una API.

```
C:\compartimoss\backend
λ dotnet  new webapi
The template "ASP.NET Core Web API" was created successfully.
```

Imagen 1.- Creación de proyecto de WebApi desde línea de comandos.

Para las personas que no están muy relacionadas con el desarrollo en .NET Core, con el nuevo Framework una de sus principales características es su modularidad y la forma en la que vamos inyectando las necesidades a nuestros proyectos a través de añadir diversos pipelines a nuestro desarrollo. Por ello vamos a añadirle por un lado que nuestra API va a usar autenticación y esta va a ser contra el Azure Active Directory. Para hacerlo de una forma "automática" podemos hacerlo desde el propio Visual Studio=> Seleccionamos el proyecto=> Botón derecha y seleccionamos "Add"-> "Connected Services" tal y como se muestra en la siguiente imagen.

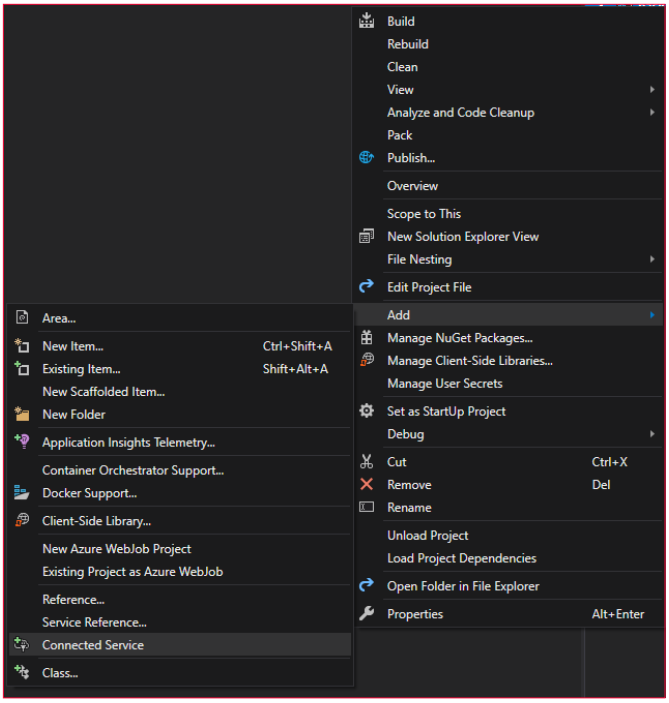

Imagen 2.- Añadiendo autenticación a la API.

Se muestra una pantalla como la siguiente y seleccionamos la opción de "Authentication with Azure Active Directory"

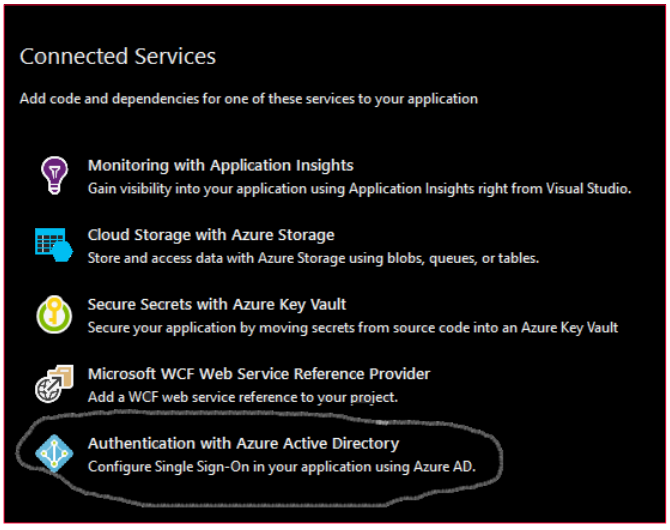

Imagen 3.- Selección del método de autenticación.

Al pulsar sobre esta opción quedará pendiente por un lado

ver qué tipo de autenticación vamos a tener: Autenticación mediante login por parte del usuario o bien mediante un token de autorización. Para el caso de una API es casi un estándar el uso de un token de autorización y por lo tanto seleccionamos esta opción.

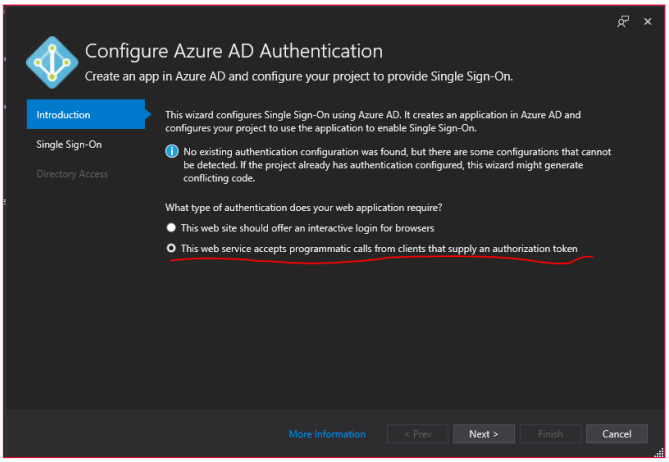

Imagen 4.- Configuración de la autenticación en Azure AD

Una vez seleccionada esta opción el siguiente paso es seleccionar el Tenant sobre el que se va a registrar la Aplicación en el directorio activo.

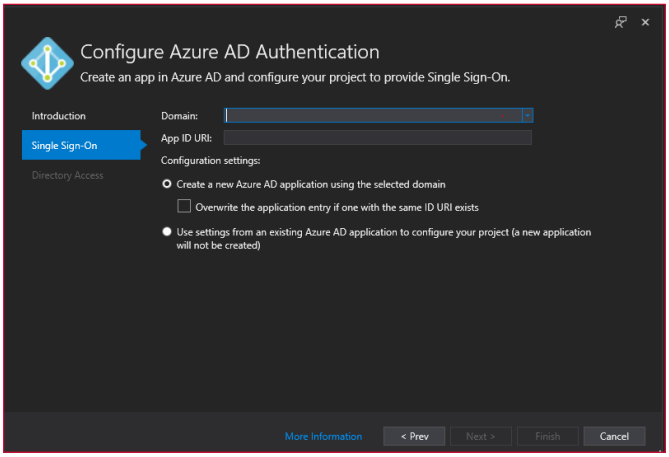

Imagen 5.- Registro de la Aplicación de Azure AD.

Una vez tenemos todos estos datos cumplimentados, al darle a finalizar el proceso nos creará una aplicación en el directorio Activo de Azure. Ademas nos añade una carpeta Extensions donde creará un Metodo AddAzureAddBearer para permitir establecer la autenticación del Azure Active Directory mediante un token Jwt en la petición. Este método lo tiene añadido en la configuración del StartUp para establecer que las llamadas a la API que requieran autenticación lo van a realizar mediante esta configuración.

La creación de la App en el directorio activo de Azure tambien la podemos realizar directamente desde el propio Portal de Azure y obtener el clientId, tentant y secret de nuestra aplicación. El ejemplo mostrado es en modo ilustrativo para ahorrar tiempo al desarrollador.

Antes de continuar con el desarrollo de la API, vamos a configurar el CORS de la aplicación para permitir las llamadas que se realicen desde SharePoint. En este ejemplo y al tratarse de una ejemplo vamos a permitir cualquier llamadas, esto NO se debe de realizar ninguna circunstancia en entornos productivos. Nota el orden en el que se inyectan los pipilines en el fichero de arranque es importante, si no se ponen en el orden adecuado es posible que no funcione correctamente, recomiendo leer la documentación para saber el motivo https://docs.microsoft.com/en-us/aspnet/core/fundamentals/middleware/?view=aspnetcore-3.0. El fichero Startup tendra la siguiente estructura:

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddCors(options =>
    {
        options.AddPolicy("CorsPolicy",
        builder => builder.AllowAnyOrigin()
                          .AllowAnyMethod()
                          .AllowAnyHeader());
    });
    services.AddAuthentication(sharedOptions =>
    {
        sharedOptions.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;
    })
    .AddAzureAdBearer(options => Configuration.Bind("AzureAd", options));

    services.AddMvc();
}
```

```
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    app.UseCors("CorsPolicy");
    app.UseAuthentication();
    app.UseMvc();
}
```

El siguiente paso que realizaremos es crearnos un método en la API que nos devuelva los datos que requieres la aplicación. En este caso vamos a realizar una API que nos devuelve una lista de Avengers. Al tratarse de un ejemplo vamos a leer los datos de un fichero json que se encuentran en la misma API. Para ellos podemos crearnos un controlador como el siguiente:

```
[Route("api/[controller]")]
[Authorize]
[ApiController]
0 references | 0 requests
public class AvengerController : ControllerBase
{
    [HttpGet]
    0 references | 0 requests | 0 exceptions
    public ActionResult<IEnumerable<Avengers>> Get()
    {
        var dataCustomer = JObject.Parse( System.IO.File.ReadAllText(@"./Data/avengers.json"));
        var customerCollection = (JArray)dataCustomer["d"];
        var avengersCollection = customerCollection.ToObject<IList<Avengers>>();
        return Ok(avengersCollection);
    }
}
```

Una vez ya tenemos nuestra API implementada y securizada contra el Azure Active Directory vamos a crearnos nuestro componente SPFx y ver todos los aspectos de configuración que nos hacen falta para establecer el Single Sign On del desarrollo.

Creando el SPFx

Con el generador de plantilla de Yeoman generaremos una nueva solución , tal y como hemos mostrado en anteriores articulos. Para este caso seleccionaremos ReactJs como Frameworks JS para los desarrollos.

```
Let's create a new SharePoint solution.
? What is your solution name? compartimoss
? Which baseline packages do you want to target for your component(s)? SharePoint Online only (latest)
? Where do you want to place the files? Use the current folder
Found npm version 6.4.1
? Do you want to allow the tenant admin the choice of being able to deploy the solution to all sites immediately without running any feature deployment or adding apps in sites? Yes
? Will the components in the solution require permissions to access web APIs that are unique and not shared with other components in the tenant? Yes
? Which type of client-side component to create? WebPart
Add new Web part to solution compartimoss.
? What is your Web part name? API
? What is your Web part description? API description
? Which framework would you like to use?
  No JavaScript framework
> React
  Knockout
```

Una vez la solución se haya creado, en primer lugar tendremos que modifciar el fichero package-solution.json que se encuentra dentro de la carpeta config. En este fichero tendremos que añadir un nuevo nodo dentro de "solution" llamado "WebAPIPermissionRequests" donde tendremos que indicar que permisos vamos a solicitar. Por ejemplo pediremos permisos de lectura a Graph y permisos a nuesta api tendriamos que añadir el siguiente código:

```
"webApiPermissionRequests": [
  {
    "resource": "Microsoft Graph",
    "scope": "User.ReadBasic.All"
  },
  {
    "resource": "Compartimoss.API.Backend",
    "scope": "user_impersonation"
  }
]
```

El nombre del recurso de de nuestra API coincide con el nombre de nuestro proyecto cuando registramos la aplicación en Azure, en caso de que no saber de donde obtenerlo se puede consultar el nombre desde el propio portal de Azure donde se registro la aplicación.

Con todo esto el siguiente paso es ver como vamos a consultar nuestra API. Generalmente en cualquier desarrollo utilizando ADAL o MSAL para la obtención del token lo realizaba siguiendo el flujo de autenticación de OAuth establecido. Ahora bien tras varios intentos el equipo de SPFx ha implementado un "Factoria" para que la autenticación sea transparente para el usuario. Para ello en primer lugar dentro del contexto del propio WebPart existe un objeto addHttpClientFactory que indicandole el clientID de nuestra aplicación del directorio activo nos devuleve un metodo AAHttClient en el que le incoporamos la petición a la API y le añade un token valido para poder consultar la API securizada. El codigo seria el siguiente:

```
public componentWillMount(){
    let avengerCollection:IAvengers[]=[];
    this.props.context.aadHttpClientFactory
    .getClient('94d59495-ba31-47b8-99e8-590942c6509f')
    .then((client:AadHttpClient):void =>{
      client
      .get('https://localhost:44386/api/avenger',AadHttpClient.configurations.v1)
      .then((response:HttpClientResponse):Promise<IAvengers[]> =>{
        return response.json();
      })
      .then ((values:IAvengers[]):void=>{
          console.log(values);
          values.forEach(element => {
            avengerCollection.push({
                                "id":element.id,
                                "name":element.name,
                                "image":element.image,
                                "description":element.description
                            });
                        });
    this.setState({avengerCollection:avengerCollection});
      });
    });
```

Este código lo podriamos poner bien cuando se vaya a cargar nuestro componente, bien vincularlo a una acción del reducer (en caso de que lo usemos). O dependiendo de las necesidades que tenga el cliente. Para simplificar, en este caso vamos a invocarlo dentro del componentWillMount y guardaremos los datos en el propio estado del componente. A continuación en el componente Render mostraremos los datos que nos devuelve la API con el siguiente método:

```
public render(): React.ReactElement<IExampleapicustomProps> {
  let i:number=0;
  return (

      <div id="products" className="row view-group">
          {
            this.state.avengerCollection.map((item:IAvengers)=>{
                i=i+1;
                return (
                    <div className="item col-xs-4 col-lg-4" key={item.id}>
                    <div className="thumbnail card">
                        <div className="img-event">
                            <img className="group list-group-image img-fluid" src=
                        </div>
                        <div className="caption card-body">
                            <h4 className="group card-title inner list-group-item-
                                {item.name}</h4>
                            <p className="group inner list-group-item-text">
                                {item.description}
                            </p>
                        </div>
                    </div>
                </div>                );
            })
          }
```

Una vez ya hemos terminado el desarrollo, vamos a desplegar el paquete de la solución en el catalago de aplicaciones. Al desplegarlo en primer lugar indica que esta aplicación requiere que se aprueben los permisos solicitados anteriormente en el manifiesto.

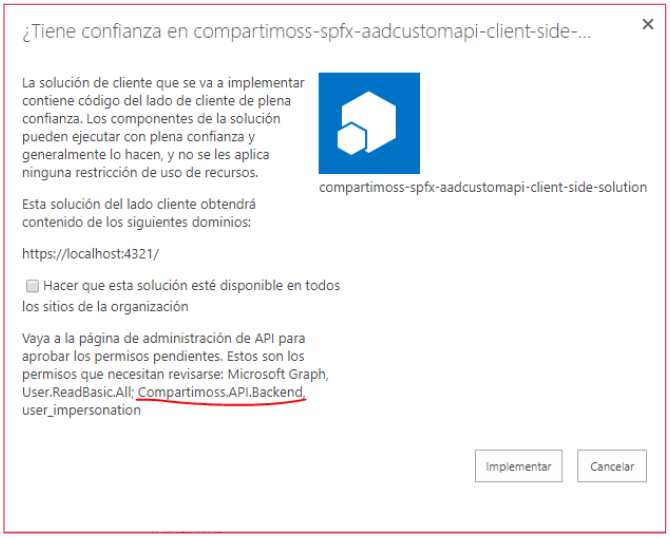

Imagen 6.- Instalación de la App en SPO.

Una vez la solución esta implementada el siguiente paso es ir a la Administración de SharePoint para la aprobación de dichos permisos.

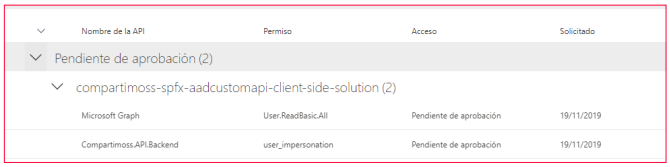

Imagen 7.- Aprobación de permisos para la App en la administración de SPO.

Si ahorra arrancamos el Workbench en SharePoint Online (https://nuestrositio/_layouts/15/workbench.aspx) y añadimos el WebPart podemos visualizar un componente que muestra una lista de Avengeres con la estructura que nosotros hemos definido en el HTML.

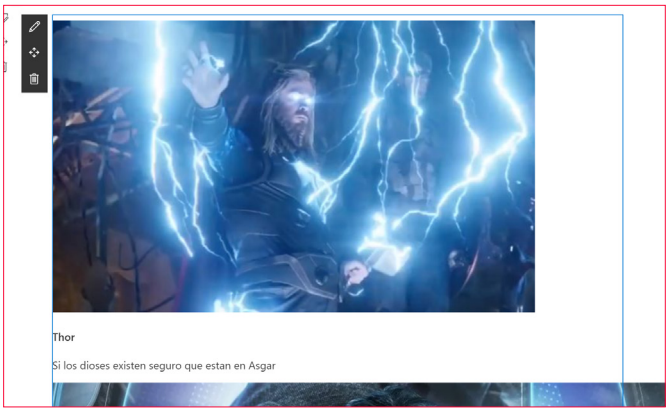

Imagen 8.- Visualización del WebPart en el Workbench.

Es importante indicar que en el Workbench de localhost no funciona la llamada AADHttpclientFactory seria intereran-te en los desarrollos optar por inyección de dependendicas dependiendo del entorno en el que se encuentre obtenga los valores de un sitio o de otro.

Declaración implicita de los permi-sos

Para las personas un poco curiosas les recomendamos ver las aplicaciones del Directorio activo que se crean en Azu-re. A pesar de todo este automatismo el flujo cuando insta-las/desinstalaas/acutalizas la aplicación en el Catalago de aplicaciones va creando apps nuevas y dejan muchas apli-caciones que no se utilizan dentro de nuestro Directorio Activo. El equipo de SPFx ha montando un sistema de per-misos para cumplir los requerimientos del SSO sin embar-go en organizaciones grandes o por lo menos en aquellas en las que la gobernanza de todos los artefactos que tie-ne Azure es algo importante para ellos no les hace mucha "gracia" este comportamiento a la hora de dar permisos. Uno de las persosas más relevantes dentro del desarrollo en SPFx Andre Connell escribio un articulo de como dar permisos directamente y el porque no utilizar esta forma de dar permisos. https://www.andrewconnell.com/blog/con-sider-avoiding-declarative-permissions-with-azure-ad-ser-vices-in-sharepoint-framework-projects/ Resumiendo un poco lo que indica este articulo es que hay una instrucción en el Office 365 CLI en el que podemos dar permisos a Sha-rePoint Online para acceder a nuestra API sin tener que desplegar nada en el Catalogo de soluciones. Esta opción para entornos de desarrollo donde el componente no se ha desarrollado es algo bastante importante y asi facilita mucho la depuración de las aplicaciones. Para ello tendria-mos que poner la siguiente linea

```
o365$ spo serviceprincipal grant add --resource "Compartimoss.API.Backend" --scope "user_impersonation"
```

Conclusión

Uno de los grandes problemas que se plantean en muchos desarrollos dentro de Office y por ello en SharePoint es la autenticación de nuestras aplicaciones dentro del marco de ejecución de Office. En algunas plataformas se necesita una doble autenticación, mostrar al usuario un login y con-traseña para poder empezar a consumir dichos servicios. Esto hace que muchos desarrollos no tengan una expe-riencia de usuario del todo satisfactoria y bien se busquen otras alternativas o incluso otras plataformas. Aunque esto parezca algo raro en Microsoft Teams no se podía hacer de serie. Con el uso de esta clase es un gran alivio para el desarrollador: se encarga de consumir el servicio de Office 365 solamente dando permisos y hacienda uso del objeto habilitado para ello. También de cara al cliente de la aplica-ción que tiene la seguridad integrada y tiene una experien-cia normal cuando está en el desarrollo.

ADRIÁN DIAZ CERVERA
Architect Software Lead at Encamina
MVP Office Development
http://blogs.encamina.com/desarrollandosobresharepoint
adiaz@encamina.com
@AdrianDiaz81

17

Entrevista Rodrigo Díaz Concha

Hola, soy Rodrigo Díaz Concha.

He sido condecorado como Microsoft Regional Director desde hace 3 años.

Por 11 años consecutivos he sido reconocido como Microsoft MVP, actualmente en las categorías de Developer Technologies y Microsoft Azure.

Soy autor de diversos cursos en LinkedIn Learning, y soy el autor de los libros “Xamarin.Forms en acción: Aplicaciones de negocio” y “Aplicaciones de Negocio con Silverlight 5”.

Con más de 20 años de experiencia profesional, he creado soluciones de software ganadoras de premios internacionales.

Pueden escuchar mi podcast semanal “Interfaz Podcast” en Spotify y en Apple Podcasts.

¿Por qué y cómo empezaste en el mundo de la tecnología?

Desde muy pequeño, me fascinaron las computadoras, y, como todo buen niño con interés en el cómputo en la década de los 80’s, mi primer contacto fue con equipos Commodore y PC’s 8086 de familiares y vecinos. A tan corta edad, mi principal fuente de interés eran los videojuegos de aventura basados en texto: aquellos donde tenías que escribir los comandos de las acciones que querías que tu personaje realizara, tales como King’s Quest, Space Quest y Police Quest. Posteriormente, en el año 1990, mi padre compró una computadora 80-286 para nuestra familia, la cual incluía MS-DOS 4.01 y Windows 3.0, ambos recién lanzados al mercado. A partir de ese momento, mi interés por los sistemas operativos y lenguajes de programación fue creciendo. Recuerdo haber leído de inicio a fin los manuales del sistema operativo y de GWBasic, lo cual me abrió la mente y me dio habilidades que conservo hasta el día de hoy. Pocos años más tarde, en 1994, a los 17 años, comencé a trabajar de forma profesional en el área de sistemas y desarrollo de software para pequeñas empresas con giros de todo tipo, desarrollando software con FoxPro, Clipper, dBase, Visual Basic, QuickBasic y algunas otras tecnologías, que el día de hoy podemos considerar extintas. En el año 2001 obtuve mi primera certificación de Microsoft, cosa que, me animó muchísimo a seguir aprendiendo y continuar con mi edificación profesional hasta el día de hoy.

¿Cuáles son tus principales actividades tecnológicas hoy en día?

Como Chief Software Architect en Lumed, empresa de la cual soy socio y donde hacemos soluciones de software para telemedicina, tomo las decisiones técnicas con respecto al roadmap que llevarán nuestras soluciones a corto y mediano plazo. Gran parte de mi tiempo la dedico a la arquitectura y el desarrollo de nuestro software con tecnologías .NET Core, .NET Framework y Azure. Por otro lado, soy Mentor y Arquitecto de Software para algunas empresas locales e internacionales, donde ayudo a definir la arquitectura de diversas soluciones de software, así como también el desarrollo de las mismas, ayudando a implementar las mejores prácticas, incrementando al máximo la calidad del software para que sean un éxito total. Adicionalmente, soy instructor en LinkedIn Learning para los tracks de arquitectura y desarrollo de software con Azure y con .NET. Finalmente, participo de forma profesional con otras empresas de tecnología, para crear material de entrenamiento, talleres y soluciones directamente para Microsoft Corporation.

¿Cuáles son tus principales actividades NO tecnológicas hoy en día?

Desde hace más de 3 años, mantengo Interfaz Podcast, el cual es un podcast semanal de arquitectura de software, desarrollo de software, nuevas y tecnologías y soft skills, y que pueden escuchar todos a través de Spotify, Apple Podcasts o a través de mi blog. Esta iniciativa la hago en mi tiempo libre, y gracias a ella, estoy en contacto constante con personalidades del mundo del software, principalmente del ecosistema de Microsoft. También, soy orador frecuente en conferencias internacionales de software, lo cual me exige viajar de forma constante. Tengo una gran cantidad de consejos, tips y hacks para viajeros, que me gustaría darlos a conocer, sin embargo, necesitaría prácticamente todo un número completo de CompartiMOSS para hacerlo. Tal vez, para la siguiente ocasión.

¿Cuáles son tus hobbies?

Leer se ha convertido, además de un hobbie, en un hábito. Procuro leer diariamente por lo menos una hora, o si se puede más. He descubierto que, si combinas libros en sus diferentes formatos: papel, e-book y audiolibro, puedes incrementar de gran manera tu capacidad y cantidad de lectura. Mi fórmula es la siguiente: e-books durante el día, libros de papel durante la noche y audiolibros durante actividades que no requieran gran concentración, como caminar, mientras me transporto a algún lado, o al hacer actividades del hogar. Además de libros técnicos, me gusta leer novelas, thrillers y libros de desarrollo personal.

¿Cuál es tu visión de futuro en la tecnología de acá a los próximos años?

A corto plazo, el cómputo en la Nube será el común denominador en cualquier vertical y mercado. Por lo tanto, aquellos profesionales que no estén inmersos hoy mismo en este modelo tecnológico están completamente fuera de juego. A mediano y largo plazo: la omnipresencia de dispositivos de tipo IoT será la constante. Eso, sumado al perfeccionamiento de tecnologías de Inteligencia Artificial y del cómputo cuántico, hará que los modelos de negocio y dinámicas sociales y culturales cambien por completo. Es fascinante pensar que el tipo de empleos y actividades dentro de un par de décadas, aún no han sido inventados. El futuro será, por lo menos, brillante.

RODRIGO DÍAZ CONCHA
Microsoft Regional Director
Microsoft MVP
@rdiazconcha
rdiazconcha.com

19

Uso de Funciones de Azure con PowerAutomate y PowerApps - Parte 2: Usando OpenAPI en PowerAutomate

PowerAutomate (conocido originalmente como Microsoft Flow) es la implementación que Microsoft ha creado para integrar el motor de flujos de Azure, Logic Apps, en Office 365. A su vez, PowerApps es un intento para facilitar la creación de aplicaciones sin necesidad de programación, que pueden ser utilizadas por sí mismas, o integradas en SharePoint Online. Ambos sistemas, aunque fáciles de utilizar y bastante poderosos en cuanto a funcionalidad, carecen de la flexibilidad para agregar nuevas posibilidades de cálculo y procesamiento. Esta falta de los dos sistemas se puede solucionar por medio de las Funciones de Azure.

Esta es la segunda parte de una serie de tres artículos:

1.– Como crear Funciones de Azure para que puedan ser utilizadas por PowerAutomate y PowerApps (CompartiMOSS No. 41, http://www.compartimoss.com/revistas/numero-41/uso-de-funciones-de-azure-con-flow-y-powerapps-part-1).

2.– Usando Funciones de Azure con PowerAutomate (CompartiMOSS No. 42).

3.– Usando Funciones de Azure con PowerApps (CompartiMOSS No. 43).

Introducción

Microsoft Office PowerAutomate es el motor de flujos de trabajo creado por Microsoft en base a Azure Logic Apps. Los flujos que se pueden crear están totalmente basados en componentes estándar que ofrecen una lógica interna de trabajo (loops, estamentos, etc.) y conexión a otros sistemas (Exchange, SharePoint y muchos otros conectores, internos y externos a Microsoft). El principal problema de esta forma de trabajo es que no se puede crear ("programar") nueva funcionalidad dentro del sistema propiamente dicho. Lo mismo se puede decir de PowerApps: aunque ofrece conectividad con muchos otros tipos de sistemas, no es posible definir capacidades de cálculo dentro de la aplicación misma.

"PowerAutomate (conocido originalmente como Microsoft Flow) es la implementación que Microsoft ha creado para integrar el motor de flujos de Azure, Logic Apps, en Office 365."

Para solucionar el problema, ambos sistemas permiten la utilización de "OpenAPI", un estándar internacional que fue creado por un consorcio de industrias que se propuso unificar la forma cómo se describen los APIs de REST, creando un formato de descripción neutral y no controlado por cualquier proveedor comercial (https://www.openapis.org). OpenAPI está basado a su vez en "Swagger", una manera conocida desde hace mucho tiempo para describir APIs de REST.

Aunque REST (REpresentational State Transfer) es una forma unificada para crear y utilizar APIs por medio de internet, la forma de usarlo es más un Framework que un estándar. Por tal motivo, los APIs de REST, como varían de uno a otro, se deben describir mediante una definición de OpenAPI para que otros sistemas "entiendan" como usar el API. Esta definición (OpenAPI) contiene información sobre qué operaciones están disponibles en una API y cómo se deben estructurar sus datos de solicitud y respuesta. Haciendo que PowerAutomate y PowerApps puedan utilizar OpenAPI hace que, a su vez, los dos sistemas estén abiertos a usar cualquier clase de funcionalidad proporcionada por cualquier tipo de sistemas externos.

Por su lado, las Funciones de Azure proporcionan toda la infraestructura técnica para poder crear funcionalidad "serverless", es decir, que los desarrolladores se pueden enfocar en crear el código que se necesita, sin necesidad de ocuparse de servidores, redes, rendimiento bajo carga, etc. Las Funciones de Azure se pueden programar en una variedad de idiomas de programación (C#, PowerShell, Python, etc.), utilizando Visual Studio, Visual Studio Code o directamente desde un navegador en el sitio de diseño de Funciones del portal de Azure. El problema, a su vez, con las Funciones de Azure es que la definición Swagger que generan no es OpenAPI ni utilizable directamente por PowerAutomate o PowerApps.

Microsoft ha publicado parches para poder crear Funciones de Azure para que puedan ser utilizadas directamente desde PowerAutomate y PowerApps, pero es requerido que tanto Azure como Office 365 utilicen el mismo Directorio Activo, lo que generalmente no es el caso en aplicaciones Enterprise. La forma para solucionar el problema es utilizar el Azure API Management, un servicio de Azure que permite exponer APIs al mundo externo por medio de OpenAPI.

En este segundo artículo de la serie de tres, se indica como crear la definición OpenAPI de la función de Azure, que es expuesta al mundo externo por medio del servicio de Azure de API Management, y como utilizarla desde Office PowerAutomate. Este ejemplo se base en toda la información dada en el primer artículo de la serie. El ejemplo a continuación es un sistema de reserva de salas de conferencias en una empresa: empleados pueden reservar una sala de conferencias desde una Lista de SharePoint; un flujo acoplado a la Lista pasa los datos de la reserva a la Función de Azure que calcula si la reserva es posible, o si el sitio está ocupado y retorna una indicación al respecto al flujo, el que retransmite la información a SharePoint y al usuario. Todo el sistema de reserva se puede utilizar también desde una aplicación de PowerApps, como se verá en el siguiente artículo. El algoritmo para determinar si una sala está ocupada o no, no está implementado, solamente se ha simulado por medio de un generador random. Pero el ejemplo indica claramente el potencial que ofrece la combinación de los cuatro sistemas.

Exportar la definición de OpenAPI

1.– Abra el portal de Azure (https://portal.azure.com) donde se creó el Grupo de Recursos que contiene la función y la definición del API Management.

2.– Abra el Grupo de Recursos (Resource Groups) donde se crearon los servicios del sistema.

3.– Haga clic sobre el servicio de API Management y luego sobre el vínculo de “APIs” (menú vertical izquierdo). Seleccione el API creado y, haciendo clic sobre el botón de elipse (“…”), seleccione “Export”.

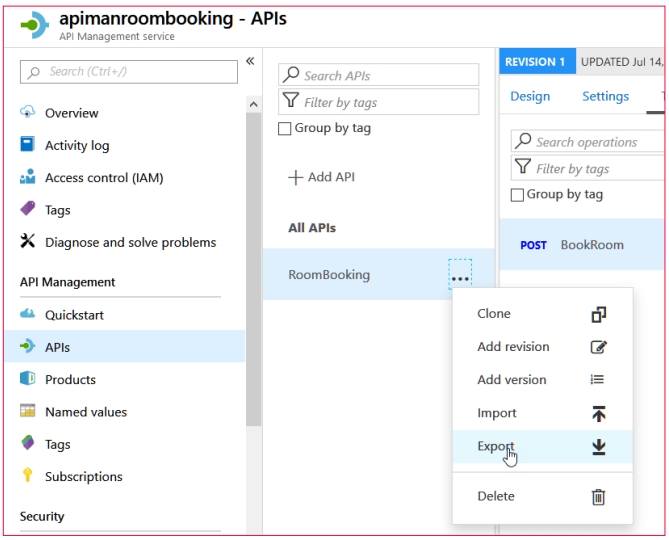

Imagen 1.- Exportar la definición de OpenAPI desde Azure.

Haga clic sobre el botón “OpenAPI v2 (JSON)”. La ventana de descargas por defecto de Windows abre para guardar el archivo de descarga localmente. El archivo es en formato JSON y contiene todos los parámetros para poder hacer una llamada al servicio REST, el que, a su vez, manda la llamada a la Función de Azure. El archivo se puede abrir con cualquier editor ASCII.

Crear una Lista de SharePoint

4.– Para el ejemplo, el flujo se inicia desde una Lista de SharePoint. Desde alguna colección de sitios de SharePoint Online cree una Lista Personalizada con dos campos de texto sencillo, uno llamado “RoomName” y el otro “Persons”.

Conectar el servicio a PowerAutomate

5.– Abra el sitio de PowerAutomate desde el portal de Office (https://portal.office.com), o desde el portal de PowerAutomate mismo (https://powerautomate.microsoft.com) y lóguese con credenciales que tengan permisos suficientes para crear y utilizar flujos.

6.– Expanda el menú de “Data” (menú vertical al lado izquierdo) y haga clic sobre “Custom connectors”. Use el botón “+New custom connector” en la esquina superior derecha para expandir un nuevo menú. Haga clic sobre “Importar an OpenAPI file”

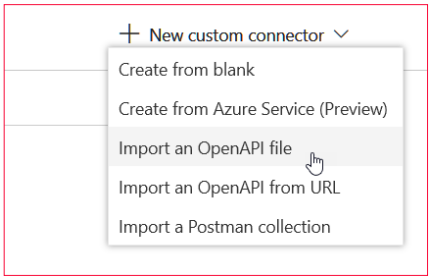

Imagen 2.- Importar la definición de OpenAPI.

Defina un nombre para el conector y seleccione el archivo descargado desde Azure en el punto 3.

Casi toda la información necesaria para configurar el conector es extraída automáticamente del archivo JSON e insertada en los campos del conector. Solamente hay que agregar el URL del host del servicio API. Regrese al portal de Azure, abra el Grupo de Aplicaciones donde están los servicios del proyecto, abra el servicio del API Management, clic sobre “APIs”, seleccione el API indicado y seleccione la pestaña “Settings”. Copie el dominio de la casilla “Base URL”, es decir, si en la casilla aparece un URL de “https://apimanroombooking.azure-api.net/roombooking”, copie solamente la parte de “apimanroombooking.azure-api.net”.

Ahora regrese a la configuración que se tenia abierta en PowerAutomate y copie este dominio en la casilla “Host” de la pestaña “Settings”. Todas las otras configuraciones se pueden dejar por defecto. Utilice el botón de “Create connector” en la esquina superior derecha de la ventana. Haga clic sobre “Custom connectors” de la sección “Data” de nuevo, y el nuevo conector debe aparecer en la lista de conectores disponibles.

7.– Seleccione el vínculo “My flows” y luego cree un nuevo flujo por medio del botón “+New” - “Au-

tomated-from blank". Asígnele un nombre al flujo y seleccione "When an ítem is created SharePoint" en la casilla de "Choose your flow's trigger". Uso el botón de "Create". Seleccione la dirección del sitio en la primera casilla del trigger, y el nombre de la Lista.

8.– Clic sobre el botón "+New step" y seleccione la pestaña de "Custom". La lista de custom connectors aparece. Seleccione el conector creado en el punto 6.

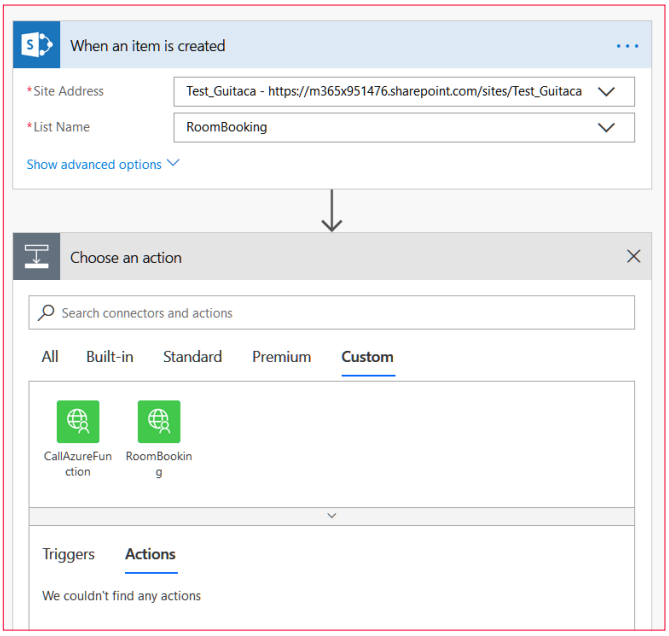

Imagen 3.- Creación del flujo.

9.– Después de seleccionar el conector aparece su ventana de configuración para crear una conexión con el conector:

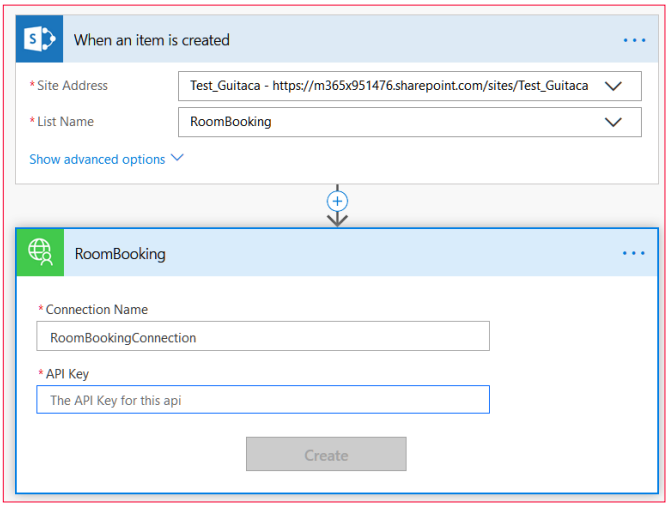

Imagen 4.- Creación de una conexión para el conector

Cuando se creó la función de Azure, se seleccionó que su autorización debería ocurrir por medio de una llave (punto 7 del primer articulo de la serie, "Authorization level" = "Function"). Esta llave es indispensable para que las llamadas a la función sean aceptadas. Si se hubiera seleccionado "Anonymous", la llave no sería necesaria, y no aparecería en la ventana de la imagen 4, pero cualquier llamada al servicio seria aceptada sin validación. El API Management mantiene la llave, y expone otra llave que es necesaria para hacer llamadas a la función.

Para encontrar el valor de la llave, desde el portal de Azure abra el servicio del API Management y use el vínculo de "Subscriptions" en el menú vertical izquierdo. Utilice el botón de elipse ("…") al lado derecho de "Buit-in-all-access" y seleccione "Show/hide keys". Use el botón con un icono de copiar de la "Primary key". Regrese al flujo y pegue el valor copiado en la casilla de "API key" de la imagen 4. Guarde los cambios en el flujo con el botón de "Create".

"En este segundo artículo de la serie de tres, se indica como crear la definición OpenAPI de la función de Azure"

Note que este paso es necesario solamente la primera vez que se utiliza el conector. Si el conector ya ha creado una conexión anteriormente, este procedimiento inicial no es necesario. También es posible crear primero la conexión directamente desde la ventana de "Connections" en el portal de PowerAutomate, y usar el botón de "+New connection" para crear una conexión con el conector manualmente. En cualquiera de los dos casos, el procedimiento para encontrar la llave es igual al descrito.

Si los valores son aceptados (es decir, la llave es válida), la ventana cambia para mostrar los valores de las dos propiedades que la función va a recibir a través del API Management. Haciendo clic sobre "roomname" se abre la ventana de "Dynamic content" con las columnas de la Lista que contienen el valor a enviar. Seleccione "RoomName". Haga la misma operación para el campo de "persons":

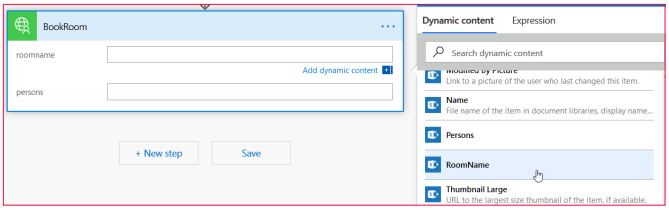

Imagen 5.- Ventana de configuración de la llamada al API Management.

10.– Agregue otro nuevo paso en el flujo. Utilice el botón de "+New step", use el botón de "Office 365 Outlook" y seleccione la acción de "Send an email". Configure las casillas de "To" y "Subject". Clic sobre la casilla de "Body" para abrir el "Dynamic content" y seleccione "Body" del conector. La respuesta de la función de Azure es enviada al API Management, y este la retransmite al flujo en el body de la respuesta de HTTP:

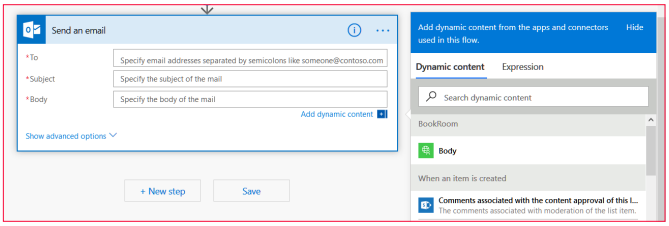

Imagen 6.- Ventana de configuración del email en Outlook.

Use el “Flow Checker” (esquina superior derecha) para comprobar que el flujo no tiene errores, y guarde todo el flujo con el botón de “Save”.

11.– Para testear el funcionamiento de todo el sistema, cree un nuevo elemento en la Lista de SharePoint. Al crear el elemento se dispara el flujo, lo cual se puede seguir en la ventana de detalles del flujo mismo:

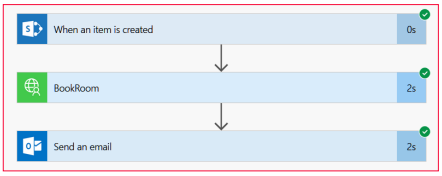

Imagen 7.- Resumen del Flujo después de funcionar.

El flujo hace una llamada REST al Azure API Management, y este a su vez llama a la función. El funcionamiento de la función se puede seguir abriendo la función misma y su ventana de Logs:

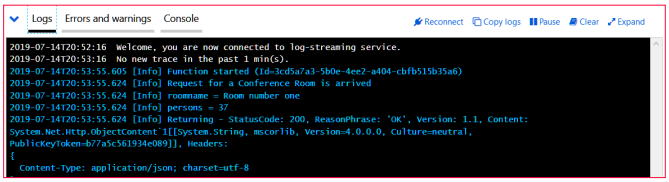

Imagen 8.- La Función de Azure trabajando con los datos del Flujo.

Finalmente, el flujo envía un Email a la cuenta configurada indicando el valor generado por la función:

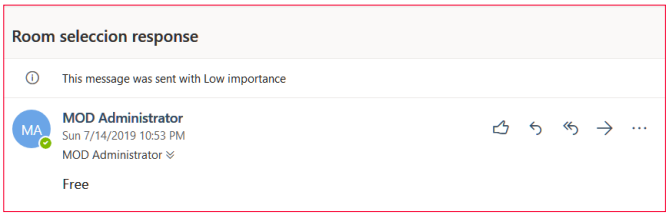

Imagen 9.- El email enviado por el flujo con los resultados de la Función.

Conclusión

Para darle mas flexibilidad y capacidad de interacción con otros sistemas, Microsoft PowerAutomate y PowerApps pueden utilizar conectores a procedimientos externos. Azure Functions es el método ideal para crear esa funcionalidad. En esta serie de tres artículos se indica como crear funciones de Azure y hacerlas funcionar bajo el OpenAPI estándar (primer artículo), como conectar PowerAutomate con la función (segundo articulo) y como puede utilizar PowerApps la misma función (tercer articulo).

GUSTAVO VELEZ
MVP Office Apps & Services
gustavo@gavd.net
http://www.gavd.net

23

Seguridad en el Cloud

A lo largo de estos años de transformación al Cloud, nos encontramos que la ciberseguridad es ámbito al que no se le suele prestar mucha atención. Creemos que como nuestros datos están en el Cloud (sea del proveedor que sea), la seguridad, ya no es problema como tal. Sin embargo, nos equivocamos, ya que estamos observando que a medida que más empresas mueven toda su infraestructura y datos al Cloud, los ataques contra este tipo de sistemas incrementan a la par.

Da igual que estemos puramente Cloud o bien sea una pieza de hardware, nosotros somos los responsables de la seguridad del dato, por lo que este artículo vamos a tratar algunas de las opciones de cómo podemos afrontar la seguridad al movernos al Cloud.

Las recomendaciones de seguridad que vamos a tratar en este artículo están basadas en:

1.– Preparar: Debemos de crear una cultura que no tenga puntos débiles

2.– Proteger: La protección es crítica, invertir dinero en mecanismos de protección reduce el riesgo de sufrir un ataque

3.– Persistir: Hay que ser persistente y vigilantes, ya que hasta la infraestructura que parece más segura puede tener una brecha de seguridad.

Securiza tus dispositivos físicos

Que nuestros datos estén en el Cloud, no quita que no debamos de proteger el acceso a la información, ya sea en un PC, móviles u otros endpoints. Es una obligación disponer de más visibilidad y conocimiento, adelantándonos así a cualquier tipo de comportamiento anómalo que podamos detectar.

Por ejemplo, nos encontramos que los inicios de sesión y descarga de aplicaciones, terminan siendo un indicativo del comienzo de una posible brecha de seguridad. Por lo que vamos a necesitar de herramientas, un equipo experto que sepa identificar la actividad, sin importar la localización de nuestros datos.

¿Cómo puedo proteger mis dispositivos? Podríamos empezar mencionando como algo tan sencillo como Windows Hello, que nos permite utilizar un dispositivo biométrico (ya sea la huella o la cara) para iniciar sesión en el equipo, con lo que ya estamos yendo hacia un mundo passwordless.

Así mismo, debemos gestionar nuestros dispositivos mediante un MDM como Intune, de manera que en caso de robo/perdida, podamos hacer un wipe de la información y así evitar que la información caiga en manos no deseadas. En el caso de que dejemos que nuestros usuarios traigan su propio dispositivo siempre podemos utilizar MAM y tener ese pedazo de información siempre controlado.

Ahora bien, si lo que queremos es proteger el contenido de los dispositivos, no podemos olvidarnos la protección endpoint que nos ofrece Windows Defender ATP, o bien el cifrado de los dispositivos mediante BitLocker

No quiero dejar de lado la nuestra infraestructura que tengamos desplegada en Azure, para ello podemos seguir las buenas prácticas recomendadas en el Cloud Operations Framework. Aquí podremos encontrar recomendaciones para evitar males mayores.

Por lo que, debemos de tener en cuenta, que proteger estos endpoints, debe de ser un componente esencial a la hora de desplegar nuestra estrategia de seguridad.

Protege tu información

Por norma general, y antes de embarcarnos con un proveedor Cloud, miramos que cumpla con nuestros estándares de seguridad, y nos solemos fijar en detalles como:

- ¿Dispongo de un backup de mi información?
- ¿Quién tiene acceso a mis datos?
- ¿Está cifrada en tránsito y en descanso?

Pero ello, provoca una cortina de humo, que hace que dejemos de fijarnos en lo que realmente importa, la información a la que accede el usuario final.

Primero de todo, debemos de controlar la información, sobre todo aquella que es sensible para la compañía, para ello podemos utilizar soluciones como Azure Information Protection, identificando y clasificando toda aquella información, bien resida en el Cloud u OnPremises, y nos permita controlar quien tiene acceso en todo momento y si hiciera falta revocar remotamente los permisos que un usuario tiene a la documentación.

El uso de sistemas de cifrado nativo, permiten que otra persona que no sea el destinatario, no pueda abrir los correos. O bien mediante el uso de políticas de DLP que permiten controlar que cierta información sensible no salga de la organización, resida en Exchange, SharePoint o OneDrive.

Algo tan sencillo como ATP en Exchange Online, nos puede ayudar a evitar algún que otro susto de phishing en el correo, ya sea evitando posibles ataques o bien analizando las URLs adjuntas que nos llegan en los mismos.

Para aquellas empresas que necesitan almacenar credenciales de usuarios en repositorios, certificados u otros datos de carácter importante, siempre pueden disponer de soluciones como Azure Key Vault para almacenar toda esta información sensible de manera segura.

¿Cómo de segura está la identidad de mis usuarios?

Hemos llegado a un punto crítico, hasta ahora hemos estado hablando de dispositivos e información, pero por mucho que securizemos nuestros entornos y dispositivos con herramientas, el punto débil de las organizaciones, seguimos siendo los usuarios.

Ya no es necesario instalar un Malware en nuestros dispositivos, basta con conocer el usuario y password para causar estragos en nuestra organización. Por lo que no debemos de dejar de lado el uso de passwords fuertes, incluso si estamos en un entorno Cloud.

Uso de tecnologías como Azure Identity Protection, que nos permite detectar comportamiento inusual en nuestros tenant, por ejemplo, podemos detectar múltiples inicios de sesión desde diferentes localizaciones en un corto período de tiempo.

Debemos de empezar a plantearnos el uso de herramientas de seguridad como Acceso condicional junto con MFA, con lo que podemos controlar como nuestros usuarios acceden a aplicaciones Cloud, y en el caso que sea necesario que deban de introducir un segundo factor de autenticación (por ejemplo, con App Authenticator).

Cuando pasamos al terreno de la administración, el primer punto a tener en cuenta es que; debemos de disponer cuentas de administración separadas de las cuentas nominales de los usuarios, con ello empezamos a segmentar el entorno y securizarlo un poquito más.

Otra tecnología que podemos usar es PIM, que nos permite la habilidad de asignar a los usuarios un administrador temporal, lo que nos permite tener controlado quien tiene permisos y durante cuánto.

Gobierna tu Cloud

Parece que hemos llegado al final del camino, pero no, como en toda estrategia, debemos de gobernarla una vez establecida. Para ello disponemos de diferentes herramientas que nos ayudarán a conseguirlo

Durante este artículo no hemos prestado mucha atención a Azure, pero disponemos de un centro llamado Azure Security Center, en el cual se nos van a presentar recomendaciones de seguridad para todos aquellos recursos que hemos desplegado sobre la plataforma.

Hablamos de ASC para Azure, pero en O365 tenemos su hermano gemelo, Secure Score, que nos proporcionará recomendaciones para incrementar la seguridad de nuestro tenant, balanceando siempre entre la productividad y la seguridad. Algo que hay que destacar, que nos permitirá conocer nuestra puntuación junto con una puntuación media de otras empresas de nuestro sector, lo cual ayuda a establecer una línea base en torno a la seguridad.

Hay algo que hasta ahora no hemos comentado, y es el temido Shadow IT, que no es más ni menos el hecho de utilizar dispositivos, servicios Cloud y aplicaciones que no están gestionadas por IT, lo cual supone un riesgo de exposición para la compañía. Todo esto lo podemos controlar mediante el uso de herramientas como Microsoft Cloud App Security, que nos ayudará a identificar y remediar todos estos casos.

Por ultimo y no menos importante, forma a tus usuarios, esto promueve una cultura de seguridad en la organización, de forma que los usuarios sean capaces de identificar ataques de phishing. Por ejemplo, en O365 disponemos de herramientas que permiten simular este tipo de ataques y a partir de ahí crear campañas de concienciación. Hay que tener paciencia con este punto, ya lleva tiempo el educar y entender el comportamiento humano en las organizaciones.

Conclusiones

Durante este artículo hemos visto algunas de las opciones que tenemos para protegernos frente a posibles ataques, pero no debemos de olvidarnos que la productividad del usuario final es algo que debemos dejar de lado a la hora de implementar estas herramientas.

Pero lo más importante y debemos grabarlo a fuego, es que, una vez tenemos una brecha de seguridad, no importa donde estuvieran almacenados los datos, solo importa el impacto negativo que pueda tener en el negocio.

Sin embargo, somos responsables de que nuestros estén debidamente securizados, identificando cuando los datos no están siendo compartidos con alguien que no se debiera, bien cuando detectamos que hay algún uso inadecuado, y sobre todo aplicando políticas de cumplimento y gobierno

ALBERTO ANDRÉS RODRÍGUEZ
Cloud Architect
@albandrod

25

Introducción a Azure Synapse

En los últimos días se está hablando mucho de este nuevo lanzamiento de Microsoft: Azure Synapse Analytics.

El objetivo de estas líneas es arrojar un poco de luz sobre qué es Synapse, para qué nos va a servir y conocer si ya está disponible para su utilización. Intentaré en próximos artículos traer más novedades sobre este tema.

¿Qué es Synapse?

Synapse es un nuevo servicio de análisis de Azure, evolución de SQL Data Warehouse, cuyo objetivo es unir las capacidades del datawarehousing empresarial con las capacidades de análisis de Big Data. En otras palabras, hablamos de hacer consultas donde mezclamos los recursos propios y/o aprovisionados con otros que pueden no ser propios y/o no queremos aprovisionar. Obviamente, el plus principal es hacer esto cuando queramos, pagar solo según demanda y todavía poder escalar. Eso es Synapse. La guinda del pastel es que podremos realizar las consultas, sea cual sea el origen, utilizando Transact-SQL además de poder integrar los resultados con Power BI o aplicar el Machine Learning que queramos.

La base de Synapse y alguna ventaja.

En realidad, Synapse es la evolución de lo que en Azure conocíamos con SQL Datawarehouse, por lo que tenemos una ventaja principal: recoge y aumenta características que ya teníamos harto probadas. Por ello, podríamos decir que es una muy buena herramienta para implementar rápidamente Data Warehouse de forma global y segura. Otro punto para tener muy en cuenta es poder escalar la capacidad de cómputo por un lado y la capacidad de almacenamiento por otro, y obviamente: en caliente.

"Synapse es un nuevo servicio de análisis de Azure, evolución de SQL Data Warehouse"

Para conocer las capacidades actuales y lo que se viene basta con darle un vistazo a la tabla de funcionalidades (tener en cuenta que este artículo se ha escrito a mediados Nov 2019 y unos días antes de la publicación de la revista, por lo que podríais ya tener muchas funcionalidades en GA que ahora aparecen en Preview):

Unified experience		
Hybrid data ingestion		✓
Azure Synapse studio		✓
Unmatched security		
Column- and row-level security	✓	
Dynamic data masking	✓	
Private endpoints		✓

Azure Synapse Analytics features

Limitless scale	**GA**	**Vista previa**
Provisioned compute (data warehouse)	✓	
Materialized views	✓	
Workload importance	✓	
Workload isolation		✓
On-demand query		✓
Powerful insights		
Power BI integration		✓
Azure Machine Learning integration		✓
Data lake exploration		✓
Streaming analytics (data warehouse)		✓
Apache Spark integration		✓

Imagen 1.- Características de Azure Synapse.

Los tres pilares principales que lo diferenciarán de SQL DataWarehouse

Básicamente los 3 pilares en los que se ha basado la re-escritura desde SQL Data Warehouse son:

- Acelerar el core del motor de almacenamiento, que ahora es capaz de diferenciar y acomodar cargas de trabajo (workloads) según la infraestructura previamente provista o (por el lado contrario) gestionarlas exclusivamente bajo demanda.
- La integración entre los Data Lakes y Apache Spark en un solo punto.
- Azure Synapse studio, una nueva interfaz, unificada que nos permite gestionar juntos los Data Warehouses y los Data Lakes.

Hay una cosa que es lo que realmente me ha llamado más la atención que no es otra que poder tirar querys sobre un entorno NO SQL en un formato 100% Transact-SQL, o dicho de una forma más bonita: querys contra un Apache Spark

(por ejemplo) simplemente agregándolo a nuestra zona de trabajo de Synapse.

La integración

Obviamente, unir dos mundos cercanos pero dispares como el de los Data Warehouses y los Data Lakes nos han obligado siempre a utilizar herramientas para su explotación que no se casan entre sí o lo hacen difícilmente. Ahora esta barrera se rompe y las primeras herramientas que forman parte de la integración "nativa" con Synapse serán Power Bi y Azure Machine Learning. Pero ahí no termina la cosa, otros servicios de terceros también están en la cola de la integración como ser Databricks, Panoply o Pragmatic Works.

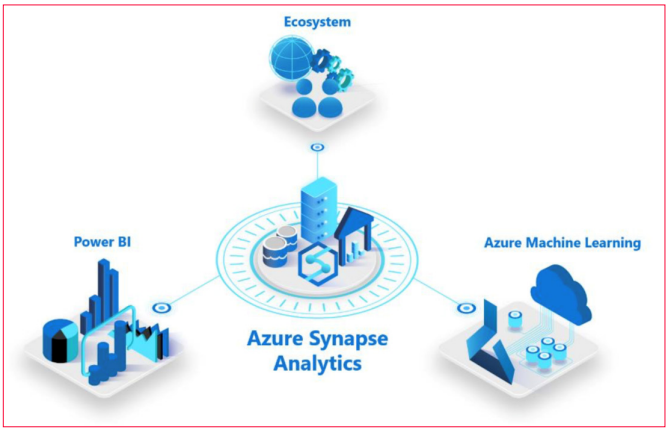

Imagen 2.- Azure Synapse Analytics.

Veamos cómo funciona

No quería terminar este artículo sin mostraros algo de funcionamiento, por lo que ¡vamos a por ello!.

"El tiempo de creación del recurso puede variar según la zona, hasta 17 minutos el algún caso"

Lo primero que necesitarás para jugar un poco con Synapse es obviamente un Tenant de Azure, en el que te recomiendo crearte un grupo de recursos exclusivamente para las cosas que vamos a crear. Daré por supuesto que estás familiarizado con el portal de Azure, por lo que una vez creado tu Grupo de Recursos para este ejercicio agregaremos a ese grupo un nuevo recurso llamado Azure Synapse Analytics.

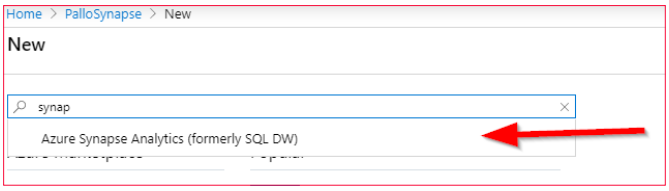

Imagen 3.- Grupo de recursos a crear.

Verás que la creación es muy similar (o idéntica) a crearte un SQL Database, aunque más específicamente un SQL Data Warehouse. Te recomiendo seguir los pasos clásicos del wizard de creación, pero por favor presta atención a este pantallazo ya que hay cosas importantes a tener en cuenta:

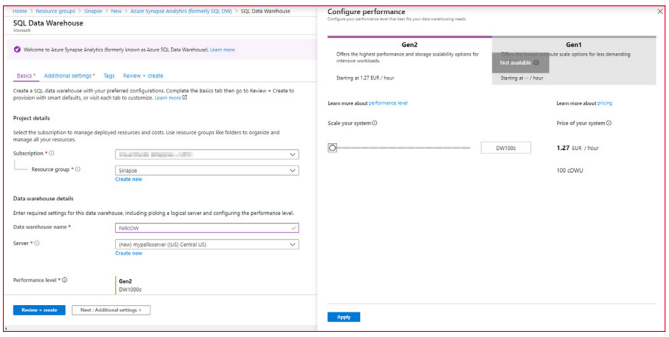

Imagen 4.- Configuración de Azure Synapse Analytics.

- Cuando vayas a crear el servidor donde está alojado te recomiendo que la zona coincida con la zona de tu grupo de recursos.
- Otra cosa muy importante es el nivel de Performance… ten cuidado porque el nivel por defecto está en un escalado medio, y en nuestro caso de testing puedes contar con que el mínimo es suficiente. Obviamente a mayor escalado más caro.
- Otra cosa a tener en cuenta, en la pestaña de configuración adicional, es la pre-carga de algunos modelos. Lo dejo a tu criterio, pero para jugar te recomiendo que hagas la precarga de "Sample" lo que cargará tu modelo con AdventureWorksDW. Y por supuesto, cuidado con la collation:

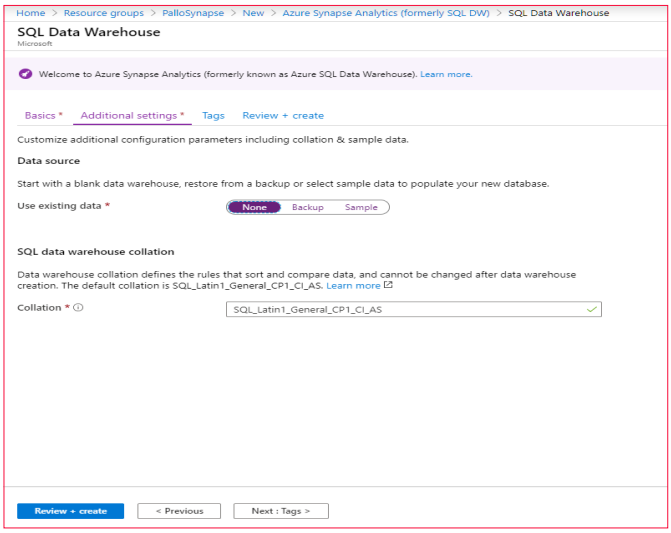

Imagen 5.- Configuración adicional.

El tiempo de creación del recurso puede variar según la zona, hasta 17 minutos el algún caso. Terminado lo anterior te encontrarás en tu grupo de trabajo con los dos elementos importantes: el servidor y la base de datos DW:

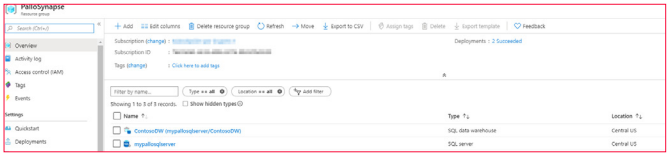

Imagen 6.- Servidor y BD.

Ahora lo importante es la carga de información (data interna) y la integración con información exterior (data externa). Tienes varios ejemplos sobre cómo hacer esa carga,

y prepárate un café y las zapatillas porque en todos ellos podrás llegar a estar unas horas esperando por la finalización de las queries. Para que elijas:

- Si quieres realizar la carga de Data de Taxis de Nueva York sigue este tutorial: https://docs.microsoft.com/en-us/azure/sql-data-warehouse/load-data-from-azure-blob-storage-using-polybase
- Si quieres realizar la carga de Data del modelo de Importaciones WideWorldImporters sigue este tutorial: https://docs.microsoft.com/en-us/azure/sql-data-warehouse/load-data-wideworldimportersdw
- Si quieres realizar la carga del modelo de Contoso Retail sigue este: https://docs.microsoft.com/en-us/azure/sql-data-warehouse/sql-data-warehouse-load-from-azure-blob-storage-with-polybase

En mi caso los he hecho todos, pero si estás impaciente por ver resultados te recomiendo el de Contoso ya que es más simple y rápido, o el de los Taxis si quieres ver más cosillas. Para este artículo he elegido el de Contoso.

Una vez terminado el tutorial hay dos cosas importantes que debes tener en cuenta:

- En la instrucción del pantallazo has agregado un blob externo a tu base de datos, en este caso el Blob público de Data de Contoso, que no es otra cosa que un Hadoop.

```
SQL

CREATE EXTERNAL DATA SOURCE AzureStorage_west_public
WITH
(
    TYPE = Hadoop
,   LOCATION = 'wasbs://contosoretaildw-tables@contosoretaildw.blob.core.windows.net/'
);
```

- Y la segunda cosa a tener en cuenta es que le des un vistazo a lo que tienes en la base de datos que puedes ver a través del Management Studio: tablas externas (enlazadas) y tablas internas (parte de las cuales se han creado a partir de las externas):

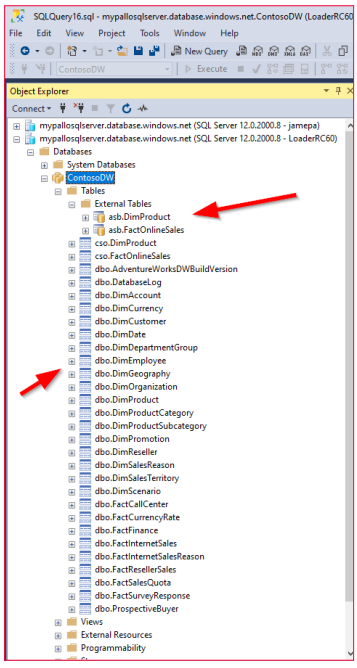

Imagen 7.- Acceso al DW desde SQL Server Management Studio.

"si estás impaciente por ver resultados te recomiendo el de Contoso ya que es más simple y rápido, o el de los Taxis si quieres ver más cosillas. "

Llegamos ahora a la parte más guapa que es verificar como funciona todo esto. Intenta ejecutar este query:

```
SELECT  SUM(f.[SalesAmount]) AS [sales_by_brand_amount] ,
p.[manufacturer]
FROM   [cso].[FactOnlineSales] AS f
JOIN   [asb].[DimProduct]    AS p
 ON f.[ProductKey] = p.[ProductKey]
GROUP BY p.[manufacturer]
```

Y verás que el resultado es:

161 %

Results | Messages

	sales_by_brand_amount	manufacturer
1	182147024,4151	Wide World Importers
2	108163138,9388	Northwind Traders
3	124810502,38	The Phone Company
4	223288734,2184	Proseware, Inc.
5	297378189,4734	Litware, Inc.
6	179451946,306	A. Datum Corporation
7	622430296,2291	Contoso, Ltd
8	370291857,5382	Adventure Works
9	448626054,6932	Fabrikam, Inc.
10	131026015,5558	Southridge Video
11	30588870,0602	Tailspin Toys

Imagen 8.- Resultado del Query.

¿Y qué es lo curioso de ese query?: pues que acabas de tirar un Join de tablas externas (léase Hadoop) con tablas internas de SQL, escrito en Trasact-SQL.

Nada más, hasta el siguiente número!

JAVIER MENENDEZ PALLO
@JavierPallo

Mentoring

Comparti*MOSS*

Un servicio experto alrededor de su SharePoint

CompartiMOSS le puede ayudar a través de su programa de Mentoring!

Contacte con nosotros y le enviaremos los planes de mentoring que tenemos disponibles para SharePoint.

29

Nuevos planes de licenciamiento de Power Automate

Introducción

Antes de la aparición de los nuevos planes, el licenciamiento en Power Automate era bastante claro y sencillo, tal y como se muestra en la siguiente tabla:

	Flow Free	Flow for Office 365 and Flow for Dynamics 365	Flow Plan 1	Flow Plan 2
	Free		USD $5.00 / user / mo	USD $15.00 / user / mo
General	Choose plan		Start free trial	Start free trial
Create unlimited automated workflows	✓	✓	✓	✓
Maximum number of runs per month (per user)	750	2,000 [1]	4,500 [1]	15,000 [1]
Maximum flow frequency	15 minutes	5 minutes	3 minutes	1 minute
Create flows from thousands of templates available in the public gallery	✓	✓	✓	✓
Use Business process flows to model multi-stage processes				✓
Optimum flow performance [2]				✓
SLA	Not available	99.9%	99.9%	99.9%

Imagen 1.- Licenciamiento en Flow (antes del 01/10/2019).

Las características de la mayoría de los planes se pueden resumir en:

1.– La cantidad de flujos que puede crear un usuario es ilimitada.

2.– El número máximo de ejecuciones mensuales (la suma de ejecuciones de todos los flujos) depende del plan, pasando de las 750 (Flow free) a las 15.000 (Flow plan 2).

3.– La frecuencia de ejecución de los flujos depende del plan, pasando de los 15 minutos (Flow free) a 1 minuto (Flow plan 2). Cabe recordar que la ejecución de los flujos se añade a una cola, y éstos se ejecutan en bloques (batches) cada cierto tiempo, el cual depende del plan del usuario.

4.– Existen una serie de conectores Premium, con lo que se necesita un plan P1 o P2 si se quieren usar (por ejemplo, conectores con SalesForce, Jira o MailChimp).

En cualquier caso, cabe destacar que un usuario de Office 365 y/o Dynamics 365 ya tiene una licencia de Power Automate incluida con las características arriba indicadas, independientemente de las tareas que ejecute el mismo, y solo teniendo en cuenta la frecuencia máxima de ejecución y el número máximo de ejecuciones mensuales.

Nuevos planes

El 1 de octubre entraron en vigor los nuevos planes, que quedan resumidos en la siguiente figura:

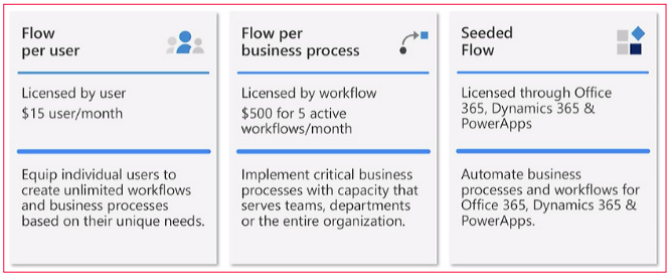

Figura 2 - Nuevos planes de Flow (en vigor desde el 01/10/2019)

Antes de entrar en detalle en cada uno de los planes, cabe mencionar especialmente un aspecto muy importante, que no es más que el cambio en el modelo de "facturación" o "contabilización" de los flujos. Anteriormente, en Power Automate se contabilizaban los flujos por unidades de ejecución, y era una de las diferencias importantes con Azure Logic Apps[1]. Con la entrada en vigor de los nuevos planes, este modelo queda obsoleto, y a partir de este momento, en cualquier tipo de flujo se tendrá en cuenta el número de llamadas a APIs, o lo que es lo mismo, todas aquellas acciones que impliquen una consulta o modificación de datos (ya sean en Office 365, Azure, AWS, Dynamics 365, etc.) o uso de un servicio (SendGrid, Adobe Docusign, etc.) que se llamen al ejecutar un flujo.

Vamos a verlo con un ejemplo: Imaginemos que tenemos un flujo que se ejecuta al crear un ítem en SharePoint y envía un correo, y que tendría un aspecto como el siguiente:

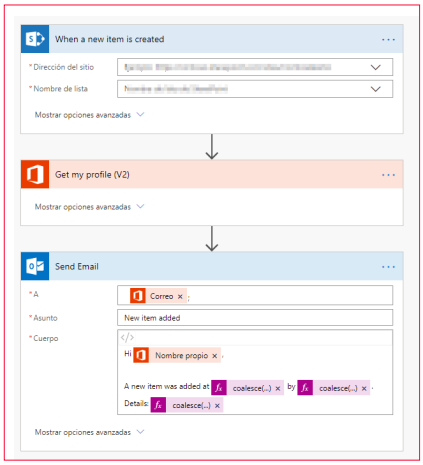

Figura 3 - Flujo de ejemplo

1 http://www.compartimoss.com/revistas/numero-40/microsoft-flow-vs-azure-logic-apps

Con el antiguo modelo de ejecución se contabilizaría 1 ejecución, mientras que con el nuevo modelo se contabilizarían 2 llamadas a APIs (la acción Get my profile, y la acción Send mail). Por lo tanto, si tenemos un flujo relativamente complejo, con distintas condiciones, solo se tendrán en cuenta las llamadas a APIs que se haya realizado en la ejecución.

En los siguientes apartados vamos a detallar cada uno de los planes indicados anteriormente.

Plan Flow por Usuario

Las características de este plan se pueden resumir de la siguiente manera:

PLAN *FLOW POR USUARIO*	
Objetivo	Productividad personal y procesos de negocio
Precio	13,00 € /mes
Unidades mínimas	20 licencias
Número de ejecuciones	Sin límite
Límite de llamadas diarias a APIs	2.000

Tabla 1.- - Características del plan Flow por Usuario.

Cabe destacar que tanto el número de flujos que un usuario puede crear, así como el número de ejecuciones de los flujos es ilimitado, pero deberemos tener en cuenta en este último caso el límite de llamadas diarias a las APIs. Por otra parte, si se quiere elegir este modelo, se deberán adquirir un mínimo de 20 licencias.

Plan Flow por Proceso de Negocio

En este tipo de licenciamiento podríamos decir que el enfoque está centrado más en el flujo que en el propio usuario. Es decir, en este caso se podrán crear hasta 5 flujos corporativos, pero los podrá utilizar cualquier usuario de la organización sin necesitar algún otro plan adicional.

PLAN *FLOW POR PROCESO DE NEGOCIO*	
Objetivo	Procesos de negocio transversales a equipos o departamentos
Precio	500 €
Número máximo de flujos	5
Número de ejecuciones	Sin límite
Límite de llamadas diarias a APIs	75.000 (entre los 5 flujos)

Tabla 2.- Características del plan Flow por Proceso de Negocio.

Así, en caso de que necesitemos flujos corporativos que deben usar todos los usuarios (por ejemplo, flujos de solicitudes de vacaciones o reservas de salas u otro equipamiento), esta opción de licenciamiento seria la adecuada (obviamente, teniendo en cuenta el tamaño de la organización).

Podríamos resumir las diferencias entre ambos planes mediante la siguiente tabla:

FUNCIONALIDAD		USUARIO	PROCESO DE NEGOCIO
Ejecutar flujos	Compra mínima	20 licencias	5 flujos
	Número de ejecuciones mensuales	Ilimitada	Ilimitada
Uso de conectores	Uso de conectores predefinidos	Sí	Sí
	Uso de conectores customizados y on-premise	Sí	Sí
Capacidades y límites	Número máximo diario de llamadas a API	2.000	75.000

Tabla 3.- Comparativa de funcionalidades entre planes de Flow.

Podemos concluir que ambos licenciamientos tienen una orientación claramente diferenciada. El plan de Flow por Usuario puede servir para que un conjunto de empleados pueda crear y ejecutar flujos a nivel de equipo o departamento, mientras que la opción de Flow por Proceso de Negocio está más orientada a flujos que puedan ser usados por todos los miembros de la organización. En este punto cabrá tener en cuenta el tamaño de la misma, y qué opción es económicamente más rentable.

"Cabe destacar que tanto el número de flujos que un usuario puede crear, así como el número de ejecuciones de los flujos es ilimitado"

Plan Seeded Flow

Este plan es el que tenemos incluido con una licencia de Office 365 y/o Dynamics 365, y apenas ha sufrido algún cambio. Sí que es destacable que algunos conectores que estaban como Standard han pasado a ser Premium, con lo que necesitaremos uno de los dos planes presentados anteriormente si los queremos usar. La lista de conectores que han pasado de Standard a Premium es la siguiente:

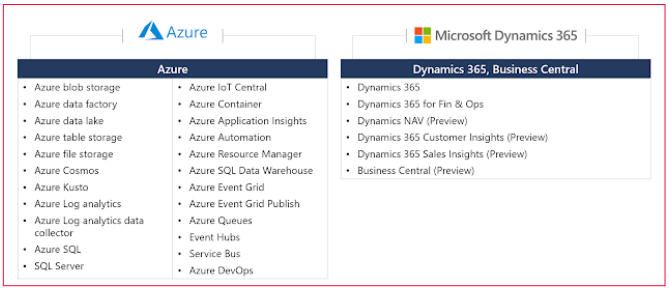

Figura 4.- Conectores que han pasado de Standard a Premium[2].

Por otra parte, se han eliminado las restricciones en el número de ejecuciones y la frecuencia de ejecución, con lo que únicamente se tiene en cuenta el número de llamadas a APIs diaria, y que se puede resumir en la siguiente tabla:

2 https://docs.microsoft.com/en-us/power-platform/admin/powerapps-flow-licensing-faq

User licenses	Number of API requests / 24 hours
Dynamics 365 Enterprise applications[1]	20,000
Dynamics 365 Professional [2]	10,000
Dynamics 365 Team Member	5,000
PowerApps per user plan	5,000
Microsoft Flow per user plan	5,000
Office licenses (that include PowerApps/Microsoft Flow)	2,000

Figura 5 - Número máximo diario de llamadas a APIs según el tipo de licencia de Office 365 y/o Dynamics 365[3]

Finalmente, hay que indicar que en el caso de licencias de Dynamics 365, solo se podrán crear flujos asociados con el servicio. Es decir, se deberán disparar desde orígenes de datos con derecho de uso por parte de aplicaciones licenciadas con Dynamics 365, tal y como se indica en esta nota de Microsoft:

> Microsoft Flow use rights with Dynamics 365 licenses: Dynamics 365 licenses will no longer[4] include general purpose Microsoft Flow capabilities. Microsoft Flows will need to map to licensed Dynamics 365 application context - Microsoft Flows should trigger from OR connect to data sources within use rights of licensed Dynamics 365 application(s). Use of standalone flows will require a Microsoft Flow license.

Transición a los nuevos planes

Microsoft ha establecido un periodo de transición a los nuevos planes, que podemos resumir de la siguiente manera:

5.– Desde el 01/10/2019 solo se pueden contratar los nuevos planes.

6.– Los usuarios con planes Flow P1 y Flow P2 pueden renovarlos hasta el 31/03/2020.

7.– A partir del 31/03/2020 no se podrán renovar los planes Flow P1 y P2 y se deberá elegir alguno de los planes de Flow indicados.

Conclusión

En este artículo hemos visto los nuevos planes de licenciamiento de Power Automate, y lo que parece ser la visión que tiene Microsoft sobre el servicio: Orientación a procesos de negocio de toda la organización, y no tanto a nivel de productividad personal.

Ciertamente aparecen dudas de si Power Automate puede ser el servicio adecuado, o si es mejor optar por Azure Logic Apps y Azure Functions, por ejemplo, aunque en este caso la orientación es más para usuario de TI y no usuario final.

Probablemente en próximos meses surgirán dudas y disyuntivas sobre qué planes utilizar llegado el caso (sobre todo en caso de necesitar conectores Premium), con lo que tendremos que esperar un tiempo para ver si estos cambios han sido para bien.

FERRAN CHOPO GARCIA
IT Consultant & Trainer
ferran@ferranchopo.com
@fchopo
http://www.ferranchopo.com

3 https://docs.microsoft.com/es-es/power-platform/admin/api-request-limits-allocations

4 https://docs.microsoft.com/en-us/power-platform/admin/powerapps-flow-licensing-faq

32

Optimizando tu Cosmos DB, mejorando tus consultas con Change Feed

Para los que no lo conozcan, aunque creo serán cada vez menos, Cosmos DB es un servicio de base de datos en Azure, que se distribuye globalmente en las distintas regiones, nos permite a golpe de clic un escalado elástico y es la máxima representación de un modelo de datos no relacional dentro del mundo Azure.

¿Cómo entender a tu Cosmos DB y modelar de forma correcta?

Cosmos DB puede llevar a mucha confusión la primera vez que lo usas, y yo creo que es porque proporciona diferentes API para comunicarnos con nuestros datos.

No debemos olvidarnos de que Cosmos DB almacena la información en documentos y que además tiene una serie de particularidades en cuanto a cómo almacena la información, es decir por mucho que podamos atacar a nuestro Cosmos DB con un API SQL, y tirar una sentencia SQL, no es un SQL y no se organiza la información en tablas.

Cosmos DB modela los datos como hemos dicho por Documentos, y los almacena en un contenedor.

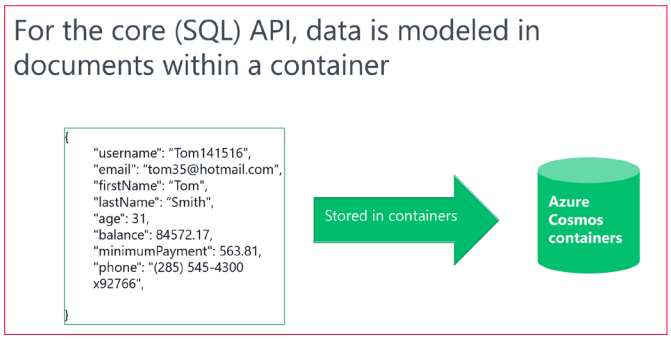

Imagen 1.- Contenedor y documentos.

Hasta ahora nada muy diferente a lo que podríamos hacer en un Mongo DB por ejemplo, que almacena documentos desnormalizados también, por lo que para mí la clave para hacer diferencial a tu Cosmos DB, es entender como "distribuye" esos documentos. Cosmos DB distribuye la información por particiones Lógicas, en función de una clave de partición que debemos escoger en la fase de modelado, y estas a su vez se distribuyen en N particiones físicas.

Por ejemplo, si almacenamos en nuestra base de datos coches, y los distribuimos con una clave de partición "COLOR", vamos a tener en una partición por ejemplo de coches rojos, y en otra coches verdes, coches negros…, con tantas particiones como colores tengamos.

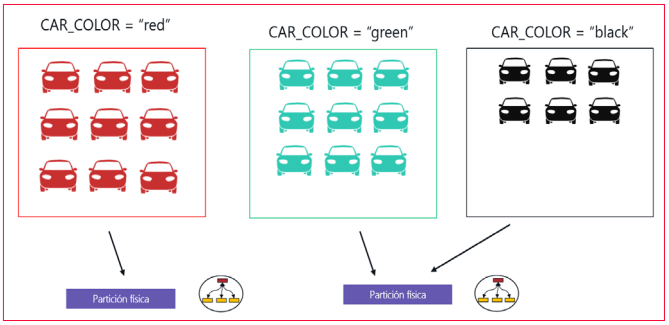

Imagen 2.- Particiones en Cosmos DB.

Todos los documentos de una partición lógica que tienen una misma clave de partición, Cosmos DB asegura que se almacenan en una misma partición física, gracias a una función HASH, que permite localizar rápidamente a los documentos. Es muy importante que escojamos una clave de partición que nos permita tener particiones similares en tamaño, para poder distribuir la carga de forma uniforme, porque de lo contrario crearemos particiones Activas o Calientes que nos penalizarán en rendimiento.

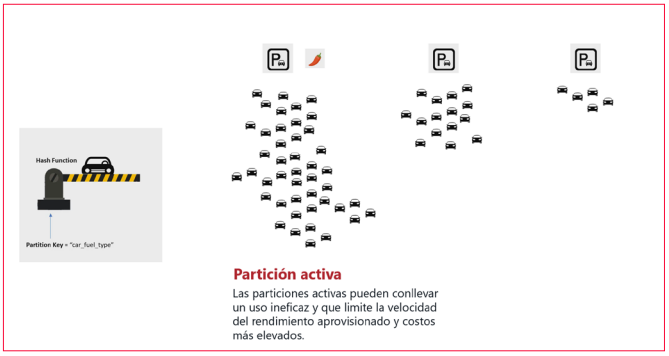

Imagen 3.- Partición activa.

Caso práctico de consulta sobre tu Cosmos DB – Leaderboard

Una vez entendido como se distribuye la información, y la mejor forma para optimizar nuestro Cosmos DB, nos daremos cuenta de que, aunque se exponga una API de SLQ no es un proceso sencillo subir la información de nuestro SQL a Cosmos DB, sino que hay que tener una estrategia de modelado para desnormalizar la información de la forma más correcta y escoger una partición idónea. Aun así, vamos a ver un ejemplo que ni con el mejor modelado

del mundo podemos optimizar una consulta sin recurrir a otras alternativas de procesado.

Vamos a imaginar que, de inicio, tenemos un modelo de datos en el que almacenamos los accesos de usuario de una aplicación, y la clave de partición es el "USERNAME".

```
{
"username": "Tom141516",
"email": "tom35@hotmail.com",
"firstName": "Tom",
"lastName": "Smith",
"age": 31,
"balance": 84572.17,
"minimumPayment": 563.81,
"phone": "(285) 545-4300 x92766",
}
```

Imagen 4.- User json.

Este modelado va a provocar que tengamos un documento por login para un usuario, teniendo una partición lógica por nombre de usuario, por lo que si hacemos una consulta del tipo SELECT * From Logins WHERE username = "Sergio", vamos a tener un rendimiento muy bueno, obteniendo todos los accesos de Sergio en un tiempo muy bueno, con un consumo de RU's (Request Units - medida de Cosmos DB para medir el rendimiento y el consumo de nuestra BBDD) muy bajo.

Operation type	RU Consumption	Operations/second	Total RU's consumed/second
Document write	5 RU's	10 Writes/second	50 RU's
Query: **SELECT * FROM c WHERE userName = "Sergio"**	3 RU's	1,000 Queries/second	3,000 RU's

Total RU's consumed: ~3,050

Imagen 5.- Consumo RU's para obtener un usuario.

Vamos a imaginar que ahora queremos trasformar nuestro modelo, y usar esta base de usuarios para guardar las puntaciones de un videojuego, sería muy sencillo guardar un documento por cada nueva puntuación de partida jugada. Ahora vamos a imaginar que queremos sacar los 10 jugadores con más puntuación en el juego, lo cual nos implicaría comparar los distintos usuarios, o lo que es lo mismo traernos todas las puntuaciones de un usuario, hacer un sumatorio y traernos los 10 elementos de mayor suma.

Esto es posible hacerlo con Cosmos DB, ya que hay juegos que usan tablas de ranking sobre Cosmos para sacar esta información como The Walking Dead.

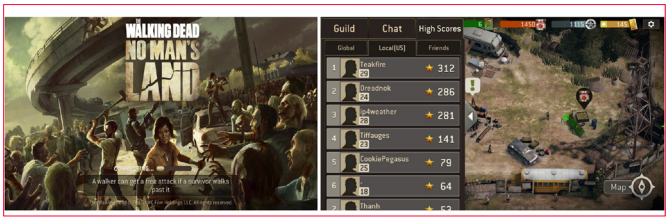

Imagen 6.- Ranking de Walking dead en Cosmos DB.

¿Pero cómo lo hacemos?, si fuéramos a la forma sencilla sería mediante el modelado anterior, sería hacer una consulta para traernos la puntuación de cada jugador, y luego calcular el ranking total en base a la puntuación de los 10 con mayor puntuación. Esto nos implica lectura de muchos usuarios por clave de partición que podemos estimar en 50 RU's y visualización de este leaderboard que se puede disparar a 100 RU's. Además, para esta consulta, a mayor número de puntuaciones, menor rendimiento obtendremos, disparando el consumo de RU's de nuestro Cosmos DB o lo que es lo mismo sobrecargando la base de datos.

Event	RU Consumption	Operations/second	Total RU's consumed/second
New score event (write)	5 RU's	10 score events/second	50 RU's
Display player score	50 RU's	1,000 Reads/second	50,000 RU's
Display leaderboard	100 RU's	1,000 Reads/second	100,000 RU's

Total RU's consumed: ~150,050

Imagen 7.- Rendimiento de obtener el ranking.

Change Feed Processer y Vistas Materializadas

Vamos a pensar en que la mejor forma de reducir el consumo de RU'S para este caso de uso del ranking, puede pasar por tener una tabla o vista resumen, que almacene simplemente los 10 jugadores de mayor ranking, y que se actualice por cada entrada de una nueva puntuación.

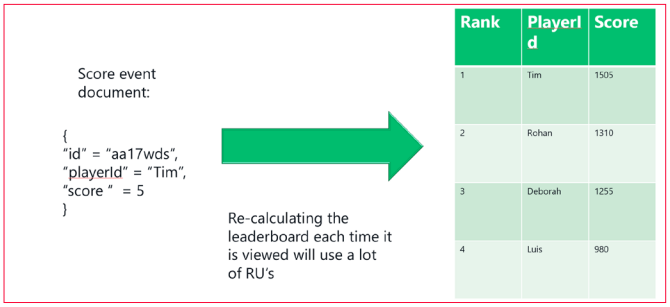

Imagen 8.- Vista resumen

Para conseguir esto necesitamos poder capturar los eventos o cambios que sufre nuestro modelo por cada inserción y actualizar la vista materializada que va a contener nuestra tabla de ranking. Esto lo vamos a conseguir con Change Feed, que no es más que la pila de cambios de todo lo que sucede en nuestro Cosmos DB, al cual podemos conectar nuestros procesos tales como una Azure Function o un código a medida mediante la librería Change Feed Processer.

Mediante Change Feed vamos a poder obtener los cambios de cada documento en tiempo real, lo que nos va a permitir capturar esas nuevas puntuaciones y almacenarlas en nuestro documento de resumen, que vamos a denominar LeaderBoard.

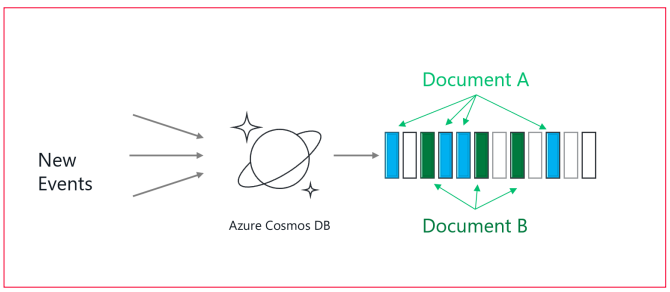

Imagen 9.- Pila de cambios.

Si además nos centramos en Change Feed Processor, podemos definir un código a medida que, mediante esta librería, podamos controlar los cambios de nuestro Cosmos DB y procesarlos bajo de manda.

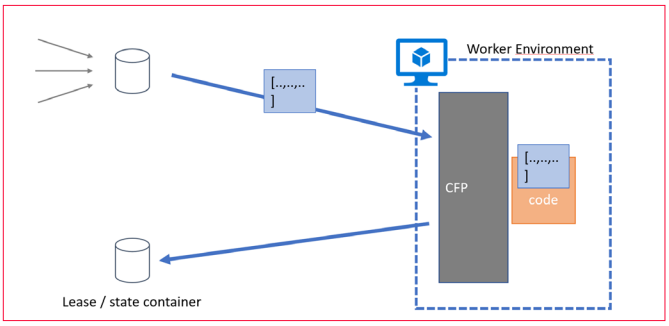

Imagen 10.- Change feed processor.

Change Feed Processor necesita de un segundo contendor que denominamos Lease Container, para poder comparar que cambios aún no han sido procesados. Hemos escogido CFP para procesar los cambios porque para el ejemplo nos permite definir entre otras cosas, la frecuencia de cada cuantos segundos queremos chequear la pila de cambios, lo cual permite controlar un poco más el proceso que con un trigger con Azure Function.

Bajando a código nuestro caso, un proceso de Consola que genere nuestro LeaderBoard

Para poder construir esa vista materializada con el ranking de nuestros jugadores, vamos a construir una aplicación de consola en Visual Studio, y vamos a utilizar el paquete Microsoft.Azure.DocumentDB.ChangeFeedProcessor, que tiene las siguientes dependencias tal y como se ve en la imagen.

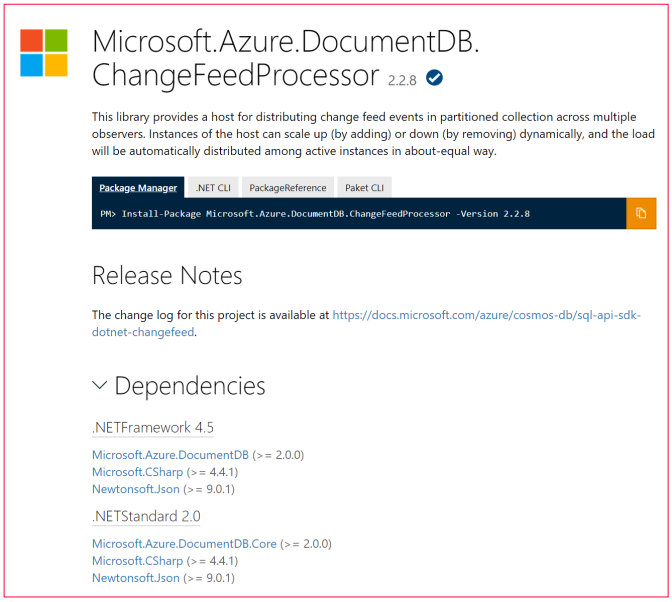

Imagen 11.- Nuget package.

Posiblemente la pega más grande ahora mismo respecto a construir un trigger con Azure Function es que dependemos de Net Framework y no podemos utilizar Net Core.

Program.cs

Si revisamos el código que he dejado en mi Github personal, debemos analizar lo primero el fichero Program de nuestra consola y que es el que va a ejecutarse por cada cambio de nuestro Cosmos DB. Lo primero es definir las colecciones de datos para nuestros dos contenedores, en este caso el contenedor del ranking (origen) y por otro lado el contenedor del Lease, donde almacenamos los cambios procesados.

```
DocumentCollectionInfo feedCollectionInfo = new DocumentCollectionInfo()
{
    DatabaseName = "leaderboard",
    CollectionName ="leaderboard",
    Uri = new Uri("https://sqlcomos2019.documents.azure.com:443/"),
    MasterKey = "0oyBJST0UzbkkU2hIYMmePZAEjBoTPTW7EEqUCL7WPhxNN2DNai8rE6mlN4R93PGe08fGIoK2FtAYfw=="
};

DocumentCollectionInfo leaseCollectionInfo = new DocumentCollectionInfo()
{
    DatabaseName = "leaderboard",
    CollectionName = "leasesScore",
    Uri = new Uri("https://sqlcomos2019.documents.azure.com:443/"),
    MasterKey = "0oyBJST0UzbkkU2hIYMmePZAEjBoTPTW7EEqUCL7WPhxNN2DNai8TwpKfn08fGIoK2FtAYfw=="
};
```

Imagen 12.- Program.cs.

Después de esto solo nos queda definir el proceso de ejecución para nuestro CFP, en base a la clase ChangeFeedObserver.

```
ChangeFeedProcessorOptions feedProcessorOptions = new ChangeFeedProcessorOptions();

feedProcessorOptions.LeaseRenewInterval = TimeSpan.FromSeconds(5);
feedProcessorOptions.StartFromBeginning = false;

var builder = new ChangeFeedProcessorBuilder();
var processor = await builder
    .WithHostName("LeaderboardHost")
    .WithFeedCollection(feedCollectionInfo)
    .WithLeaseCollection(leaseCollectionInfo)
    .WithObserver<ChangeFeedObserver>()
    .WithProcessorOptions(feedProcessorOptions)
    .BuildAsync();

await processor.StartAsync();

Console.WriteLine("Change Feed Processor started. Press <Enter> key to stop...");

Console.ReadLine();

await processor.StopAsync();
```

Imagen 13.- Cofigurando CFP.

Si revisamos el Código, debemos indicarle al proceso cada cuanto tiempo debemos chequear los cambios, y darle un nombre al proceso con HostName, y pasarle las colecciones que hemos definido previamente como FeedCollection (donde obtener los cambios) y LeaseCollection (donde llevar un histórico para poder comparar los cambios pendientes).

ChangeFeedObserver.cs

En esta clase definimos “que hace” nuestro proceso, para cada cambio recibido, esta clase se encarga de gestionar los hilos del proceso y las tareas de cambio.

```
class ChangeFeedObserver : IChangeFeedObserver
{
    public Task CloseAsync(IChangeFeedObserverContext context, ChangeFeedObserverCloseReason reason)
    {
        return Task.CompletedTask;  // Note: requires targeting .Net 4.6+.
    }

    public Task OpenAsync(IChangeFeedObserverContext context)
    {
        return Task.CompletedTask;
    }

    public Task ProcessChangesAsync(IChangeFeedObserverContext context, IReadOnlyList<Document> newScoresList, CancellationToken cancellationToken)
    {
        Console.WriteLine("Processing " + newScoresList.Count + " score updates");
        UpdateMaterializedView updateMaterializedView = new UpdateMaterializedView();
        updateMaterializedView.Update(newScoresList);

        return Task.CompletedTask;
    }
}
```

Imagen 14.- Task CFP.

El resto del código se basa en una clase UpdateMaterializedView, que se encarga de obtener el ranking actual y ver si se debe actualizar o no con la entrada de una nueva puntuación. La forma de hacerlo no puede pasar por leer todos los documentos y sumar puntuaciones, por lo que en este caso nos traemos la última puntuación sumada de un jugador y la actualizamos con la nueva entrada.

```
Document playerDoc = await client.ReadDocumentAsync(playerDocumentUri, new RequestOptions { PartitionKey = new PartitionKey(newScoreEvent.playerId) });
var existingPlayer = JsonConvert.DeserializeObject<Player>(playerDoc.ToString());
Player updatedPlayer = new Player(newScoreEvent.playerId, newScoreEvent.score + existingPlayer.score);
await client.UpsertDocumentAsync(collectionLink, updatedPlayer);
```

Imagen 15.- Get score by user.

Esa puntuación actualizada la vamos a añadir al ranking, bien como una nueva entrada o actualizando una existente si ya estaba entre los 10 primeros jugadores.

```
Uri leaderboardDocumentUri = UriFactory.CreateDocumentUri(databaseName, collectionName, "leaderboard");
Document leaderboardDoc = await client.ReadDocumentAsync(leaderboardDocumentUri, new RequestOptions { PartitionKey = new PartitionKey("leaderboard
var existingLeaderboardDoc = JsonConvert.DeserializeObject<Leaderboard>(leaderboardDoc.ToString());
Leaderboard existingLeaderboard = new Leaderboard(existingLeaderboardDoc.id, existingLeaderboardDoc.playersRanked);

if (existingLeaderboard.containsPlayer(updatedPlayer))
{
    existingLeaderboard.updatePlayer(updatedPlayer);
}

else
{
    existingLeaderboard.playersRanked.Add(updatedPlayer);
}
```

Imagen 16.- Actualizando el ranking.

Este código lo que hace es añadir al jugador actual al raking, lo que puede dejar la lista en 10 o más jugadores. (en este punto gana fuerza el usar CFP ya que tenemos un único hilo que toca el fichero leaderboard, si lo lleváramos a un escalado con Functions deberíamos buscar otra estrategia de añadir al ranking usuarios). Ya solo nos queda ordenar el ranking, y actualizar el fichero LeaderBoard con únicamente 10 miembros.

```
Leaderboard updatedLeaderboard = existingLeaderboard;
updatedLeaderboard.playersRanked = updatedLeaderboard.sortleaderboard(updatedLeaderboard.playersRanked);
await client.UpsertDocumentAsync(collectionLink, updatedLeaderboard);
```

Imagen 17.- Ranking final.

El resto del código se basa en modelos de clases y configuraciones de Cosmos DB que no explico en este artículo pero que podéis encontrar en la documentación oficial de Microsoft.

"lo que conseguimos es optimizar nuestro Cosmos DB y en concreto evitar consultas que nos lleven a retorcer a nuestros modelos"

Analizando que hemos conseguido

En resumen, lo que conseguimos es optimizar nuestro Cosmos DB y en concreto evitar consultas que nos lleven a retorcer a nuestros modelos. Posiblemente esto es un choque cultural y el nombre SQL API lleve a pensar que, optimizando las consultas podemos mejorar nuestro rendimiento, pero cuando eso os pase por la mente volver al punto uno y recordar como almacena los documentos Cosmos y como particiona la información, ya que no toda consulta es optimizable para Cosmos.

Por norma debemos siempre filtrar por la clave de partición y a ser posible evitar hacer cruces sobre distintas claves; y en vez de eso sacar vistas de agregados ya que el rendimiento mejora mucho.

Por concluir vamos a ver que la consulta que estamos utilizando para traer la vista materializada es muy sencilla, tal y como muestra la siguiente imagen.

```
SELECT TOP 10 c.IdPlayer, (SELECT SUM(c.Score) FROM c GROUP BY c.IdPlayer) as total FROM c
SELECT * From c WHERE c.IdPlayer = "leaderboard"
```

Results Query Stats

- 1


```
[
    {
        "IdPlayer": "leaderboard",
        "id": "leaderboard",
        "playersRanked": [
            {
                "Type": "Score",
                "id": "Barbara",
                "score": 60,
                "IdPlayer": "Barbara"
            },
            {
                "Type": "Score",
                "id": "Lola",
                "score": 56,
                "IdPlayer": "Lola"
            },
            {
                "Type": "Score",
                "id": "Olivia",
                "score": 45,
                "IdPlayer": "Olivia"
            },
            {
                "Type": "Score",
                "id": "Alberto",
                "score": 35,
                "IdPlayer": "Alberto"
            },
            {
                "Type": "Score",
                "id": "Leticia",
                "score": 24,
                "IdPlayer": "Leticia"
            },
```


Imagen 18.- Consulta realizada.

Haciendo una vista materializada que resuma nuestro ranking, tenemos una simple consulta por Clave de partición en este caso "LeaderBoard", reduciendo en mucho el consumo de RU's.

Event	RU Consumption	Operations/second	Total RU's consumed/sec
New score event (write)	5 RU's	10 score events/second	50 RU's
Reads from change feed	1 RU	20-30 Reads/second	20-30 RU's
Update player document(write)	10 RU's	10 updates/second	100 RU's
Update leaderboard document (write)	10 RU's	10 updates/second	100 RU's
Display player document (read)	1 RU	1,000 Reads/second	1,000 RU's
Display leaderboard document (read)	1 RU	1,000 Reads/second	1,000 RU's

Total RU's Consumed: ~2,280

Imagen 19.- RU's consumidas.

Con una vista materializada tenemos un consumo 2,280 RU's en media, para obtener el leaderboard del juego y la lectura de una nueva puntuación, en cambio sin vista materializada nos implica 150 RU's de media. Debemos tener en cuenta que cada RU's que podamos ahorrarnos significa que implica un mayor rendimiento de nuestro base de datos, y sobre todo ahorro económico por escalados innecesarios.

Por todo esto necesitamos por un lado un modelado inteligente de inicio y sobre todo escoger la mejor estrategia para extraer nuestra información, bien por consulta o bien por proceso de desnormalización y vistas agregadas.

Conclusión

Es muy cierto que el rendimiento de las consultas de Cosmos DB mejora día a día, que además presenta una gran variedad de APIS para poder atacar a la información, haciendo a Cosmos DB un producto muy atractivo para mu-

chos tipos de desarrolladores y de administradores de base de datos.

Pero también es cierto que aún tiene mucho por mejorar, que puede ser confuso para una persona que viene de SQL el tener que pensar en modelado por particiones, y más aun el no poder hacer cualquier tipo de consulta libremente sin pensar en particiones o claves de partición.

Pero también es muy cierto que esto corre y mucho, por ejemplo, cuando preparé el material para este artículo, la mejor forma para hacer un GROUP BY (que no es otra cosa que lo que hace esta vista materializada), era el pensar en un proceso en backend para crear esta vista de resumen. Pues desde el pasado mes de octubre tenemos posibilidad de hacer consultas con GROUP BY y con un rendimiento muy similar al obtenido por la vista materializada, igual aun un poco por encima pero que puede compensar si en el equipo no tenemos desarrolladores de backend o no tenemos tiempo para afinar tanto el consumo de RU's.

Por ejemplo, para obtener el Leaderboard podríamos plantear una consulta muy parecida a esto:

```
SELECT TOP 10 c.IdPlayer, (SELECT SUM(c.Score) FROM c
GROUP BY c.IdPlayer) as total FROM c
SELECT * From c WHERE c.IdPlayer = "leaderboard"
```

Y el consumo de RU's no sería muy por encima de 3 – 4 RU's para obtener el ranking, lo que sin duda habla de que en pocos meses mejora mucho el sistema de consultas, y si lo unimos a un entendimiento de como modela los datos Cosmos DB, en poco tiempo podemos estar ante una máquina de triturar datos y de fácil inserción (como cualquier modelo no relacional que se precie).

SERGIO HERNÁNDEZ MANCEBO
Azure MVP
@shmancebo
shernandez@encamina.com

37

Productivity Tips con Microsoft Teams: Búsqueda de Información

Este artículo es el primero de una serie de artículos sobre Tips en el uso de Microsoft Teams que faciliten al usuario tareas habituales con la herramienta de Teamwork por excelencia de Microsoft. En este primer artículo vamos a revisar como buscar información en Teams.

Buscando información en Teams: Por donde empiezo

Para buscar información en Teams, lo único que tenemos que hacer es indicar la palabra clave a buscar en la barra de línea de comandos que también se utiliza para ejecutar búsquedas:

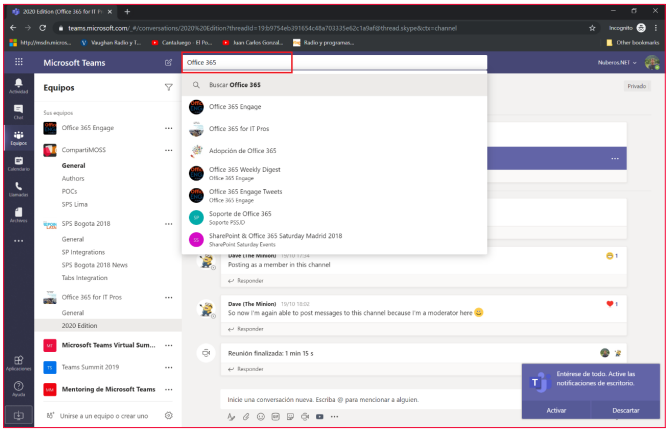

Imagen 1.- Ejemplo de realización de una búsqueda en Teams.

Como se puede apreciar en la Imagen 1, a medida que se indica un criterio de búsqueda se sugieren Teams que estén relacionados con dicho criterio bien en el propio nombre de un Team, en conversaciones que tienen lugar en un Team en concreto o en archivos almacenados en un Team.

"a medida que se indica un criterio de búsqueda se sugieren Teams que estén relacionados con dicho criterio"

Para que se muestren resultados de un criterio de búsqueda, simplemente pulsamos la tecla "Intro" en nuestro equipo o bien hacemos clic en el icono de buscar que se muestra bajo la barra de comandos de Teams. A continuación, en la parte derecha de Teams se muestran los resultados que coinciden con dicho criterio de búsqueda.

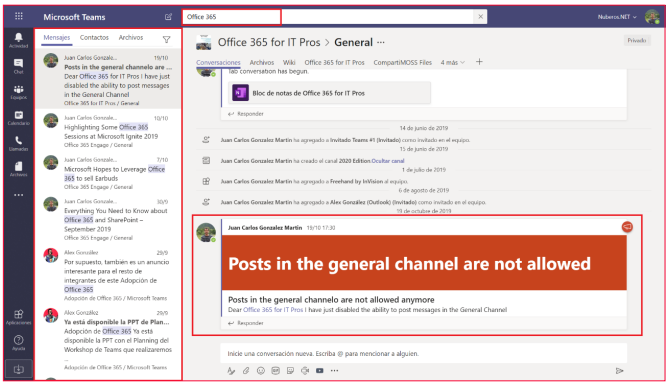

Imagen 2.- Resultados de una búsqueda en Teams.

Analizando los resultados de una búsqueda en Teams

Como se puede apreciar en la Imagen 2, los resultados de una búsqueda en Teams se organizan de acuerdo con tres ámbitos diferentes:

- Mensajes, es decir, se muestran todas las conversaciones que han tenido lugar en distintos Teams y que contienen la palabra clave de búsqueda introducida. Al hacer click en uno de los resultados, se muestra el mensaje concreto que contiene dicha palabra clave.
- Contactos, es decir, aquellos contactos en Teams que contienen la palabra clave indicada.
- Archivos, es decir, se muestran todos los archivos que contiene en su nombre o contenido la palabra clave indicada. Como en el caso del ámbito de Mensajes, al hacer clic en un resultado se visualiza el archivo concreto que contiene dicha palabra clave.

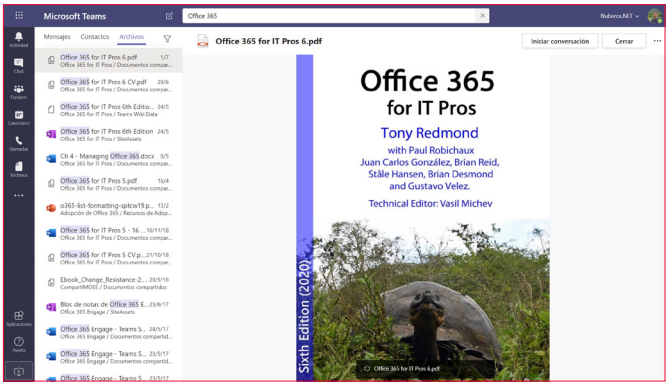

Imagen 3.- Ámbito de resultado "Archivos" y detalle de un resultado concreto.

"Cada ámbito de búsquedas en Teams cuenta con su propio juego de filtros que son propios de este y que permiten acotar las búsquedas"

Aplicando filtros a las búsquedas en Teams

Cada ámbito de búsquedas en Teams cuenta con su propio juego de filtros que son propios de este y que permiten acotar las búsquedas.

- Para el ámbito de "Mensajes", tendremos los filtros que se muestran en la Imagen 4:
  - Asunto, para indicar un asunto concreto en los mensajes que contengan la palabra clave.
  - Intervalo de fecha, para indicar uno de los siguientes valores: Hoy, Ayer, Esta semana, Semana pasada, Este Mes, Mes pasado, Este año, Último año.
  - Equipo, para indicar un Team concreto en el que realizar la búsqueda. No es posible seleccionar más de un Team.
  - Canal, para indicar un canal concreto en el Team. No es posible seleccionar más de un canal.
  - De para indicar el nombre de la persona que ha escrito un mensaje conteniendo dicha palabra clave.
  - Incluir/Excluir los mensajes en los que se mencione al usuario
  - Incluir/Excluir los mensajes que contengan archivos adjuntos.

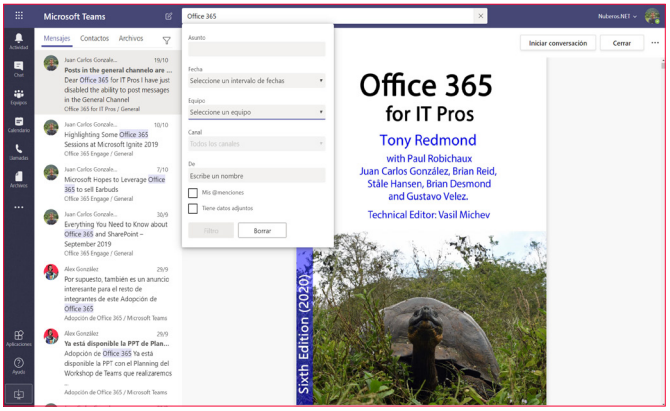

Imagen 4.- Filtros para el ámbito de "Mensajes".

- Para el ámbito "Contactos" no hay filtros definidos y la experiencia de usuario en los resultados de las búsquedas es la que se muestra en la Imagen 5.

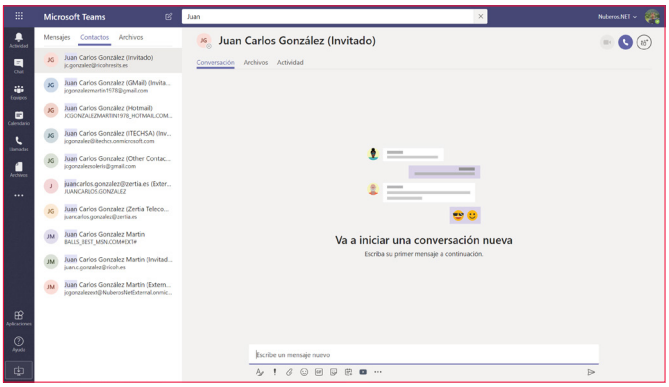

Imagen 5.- Experiencia de usuario en los resultados en el ámbito "Contactos".

- Para el ámbito "Archivos", los filtros disponibles son los siguientes:
  - Equipo, permite indicar el "Team" en el que localizar los archivos que cumplen con el criterio de búsqueda.
  - Tipo de archivo, permite indicar el tipo de archivo a buscar. Los formatos incluidos en este filtro son: Excel, PowerPoint, Word, Imágenes, OneNote, Archivo de texto, PDF.
  - Modificado por, permite indicar el nombre del usuario que modificó el archivo a buscar.

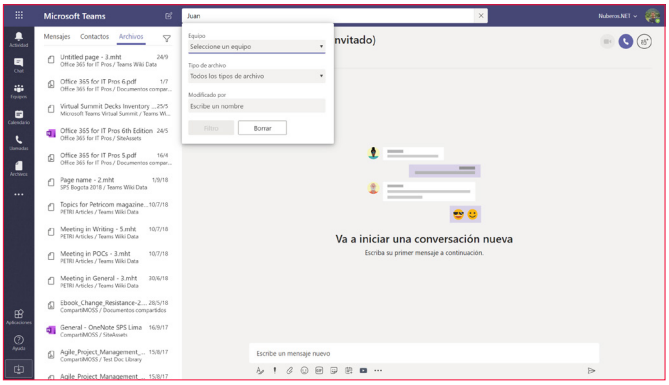

Imagen 6.- Filtros para el ámbito "Archivos".

Conclusiones

En este artículo hemos visto como los usuarios en Teams pueden realizar búsquedas de palabras clave en la plataforma. Los resultados de las búsquedas se pueden agrupar en tres ámbitos diferenciados (Mensajes, Contactos y Archivos) en los que es posible aplicar filtros de búsqueda para acotar más los resultados.

JUAN CARLOS GONZÁLEZ
Office 365 SME en RICOH
Office Apps & Services MVP
@jcgm1978 | https://www.linkedin.com/in/juagon/

31 Entrevista Raona

Somos una consultora internacional en nuevas tecnologías nacida en 2003 especializados en el desarrollo de soluciones Microsoft, del que somos Gold Partners en Collaboration & Content, Application Development y Application Integration. Contamos con oficinas en Barcelona, Madrid, Andorra y Argentina y con un equipo de más de 170 consultores y desarrolladores expertos que colaboran en el día a día con clientes líderes en todos los sectores empresariales.

¿Por qué y cómo empezó en el mundo de la tecnología?

Raona nació de la evolución de anteriores empresas tecnológicas con la ambición y la visión de hacer del ingeniero y la tecnología el centro de la empresa y apostando siempre por el conocimiento del sector y la excelencia en el dominio de las tecnologías IT. Todo con el objetivo de poder ofrecer así un servicio completo y transversal a nuestros clientes.

¿Cuáles son las principales actividades tecnológicas hoy en día?

En Raona desarrollamos proyectos a medida para nuestros clientes, proporcionamos servicios de consultoría para ofrecerles las mejores soluciones y diseñar herramientas innovadoras, colaboramos con los equipos internos de nuestros clientes para aportarles conocimiento tecnológico, proporcionamos servicios gestionados y servicios de mentoring. Entre nuestras soluciones, encontramos la creación de intranets sociales y corporativas, portales web, Business Applications, Cloud Integration, Touch & Mobility y Modern Apps & Custom Development.

¿Cuáles son las principales actividades NO tecnológicas hoy en día?

Año tras año, en Raona celebramos el Enraona, un evento corporativo en el que nos reunimos todos los Raonencs para poner en común nuestras ideas y hacer un repaso de todo el año mientras disfrutamos de actividades lúdicas y de una jornada relajada y divertida. Además, nos gusta reunirnos periódicamente durante el año y organizar actividades deportivas como por ejemplo los torneos de pádel que hacemos anualmente para recaudar fondos para la Maratón de TV3. A todo esto, hay que añadirle que tenemos un equipo de e-gamers llamado Raona Coders que une dos de nuestras pasiones, el mundo digital y el trabajo en equipo. El objetivo de esta actividad es que nos podamos reunir una vez por semana y pasar tiempo juntos jugando a videojuegos para que una vez al mes, podamos hacer un torneo.

¿Cuáles son las actividades que realiza en la comunidad técnica?

Nos gusta participar en eventos de comunidad como los SharePoint Saturdays y los CRM Saturdays donde realizamos ponencias y donde nos encontramos con más expertos para compartir las ideas sobre estos temas y adquirir nuevos conocimientos.

Somos una empresa joven que nos gusta asistir a todos aquellos eventos de talento que nos permiten conocer los nuevos profesionales que están entrando en el sector y así seguir creciendo con un aire fresco en nuestros equipos. Además, para transmitir nuestra visión sobre la industria, nuestros expertos, junto a otros profesionales del sector, se reúnen en una mesa redonda o conferencia para comunicar nuestra visión sobre una temática en concreto.

¿Cuál es la visión de futuro en la tecnología de acá a los próximos años?

A nivel de las empresas, la tecnología está evolucionando para facilitar los procesos a los trabajadores y mejorar su día a día en el trabajo consiguiendo así más facilidades y potenciar el Employee Engagement. Todo ello, lleva a una mayor productividad y creatividad en los procesos y proyectos.

43

Nosotros

Alberto Diaz

Alberto Diaz cuenta con más de 15 años de experiencia en la Industria IT, todos ellos trabajando con tecnologías Microsoft. Actualmente, es Chief Technology Innovation Officer en ENCAMINA, liderando el desarrollo de software con tecnología Microsoft, y miembro del equipo de Dirección.

Desde 2011 ha sido nombrado Microsoft MVP, reconocimiento que ha renovado por séptimo año consecutivo. Se define como un geek, amante de los smartphones y desarrollador. Fundador de TenerifeDev (www.tenerifedev.com), un grupo de usuarios de .NET en Tenerife, y coordinador de SUGES (Grupo de Usuarios de SharePoint de España, www.suges.es)

Email: adiazcan@hotmail.com
Twitter: @adiazcan

Fabián Imaz

Fabián Imaz, MVP de SharePoint Server trabaja en el mundo del desarrollo de software desde hace más de 10 años, teniendo la suerte de trabajar en distintas arquitecturas y tecnologías Microsoft. Pertenece a la firma Siderys, http://www.siderys.com empresa de desarrollo de Software especializada en SharePoint 2007/2010/2013 y en desarrollo de soluciones inteligentes.

Desde los comienzos Fabián ha trabajado en distintitas comunidades donde organiza y promueve eventos locales para la difusión de tecnología dentro de los miembros de las mismas. Es director de la carrera SharePoint 2010 y SharePoint 2013 en Microsoft Virtual Academy, http://www.mslatam.com/latam/technet/mva2/ Home.aspx y cuenta con un sitio en CodePlex con varios desarrollos http://siderys.codeplex.com.

Sitio Web: http://www.siderys.com
Email: fabiani@siderys.com.uy
Blogs: http://blog.siderys.com
Twitter: @fabianimaz

Gustavo Velez

Gustavo Velez es Ingeniero Mecánico y Electrónico; trabaja en la arquitectura, diseño e implementación de sistemas de IT basados en tecnologías de Microsoft, especialmente SharePoint, Office 365 y Azure.

Propietario del sitio especializado en información sobre SharePoint en español http://www. gavd.net, autor de ocho libros sobre SharePoint y sus tecnologías y numerosos articulos y conferencias sobre el tema.

Sitio Web: http://www.gavd.net
Email: gustavo@gavd.net
Blogs: http://geeks.ms/blogs/gvelez/

Juan Carlos González Martín

Ingeniero de Telecomunicaciones por la Universidad de Valladolid y Diplomado en Ciencias Empresariales por la Universidad Oberta de Catalunya (UOC). Cuenta con más de 14 años de experiencia en tecnologías y plataformas de Microsoft diversas (SQL Server, Visual Studio, .NET Framework, etc.), aunque su trabajo diario gira en torno a las plataformas SharePoint & Office 365. Juan Carlos es MVP de Office Apps & Services y co-fundador del Grupo de Usuarios de SharePoint de España (SUGES, www.suges.es), delGrupo de Usuarios de Cloud Computing de España (CLOUDES) y de la Comunidad de Office 365. Hasta la fecha, ha publicado 11 libros sobre SharePoint & Office 365, así como varios artículos encastellano y en inglés sobre ambas plataformas.

Email: jcgonzalezmartin1978@hotmail.com
Blogs: http://geeks.ms/blogs/jcgonzalez &
http://jcgonzalezmartin.wordpress.com/

Santiago Porras

Innovation Team Leader en ENCAMINA, lidera el desarrollo de productos mediante tecnologías Microsoft. Se declara un apasionado de la tecnología, destacando el desarrollo para dispositivos móviles y web, donde ya cuenta con 16 años de experiencia.

Microsoft MVP in Developer Technologies, colabora con las comunidades de desarrolladores desde su blog personal **http://geeks.ms/santypr** y ocasionalmente en **CompartiMOSS.com**. Además, es uno de los coordinadores de TenerifeDev, grupo de usuarios de .NET en Tenerife (**http://www.tenerifedev.com**)

Sitio Web: http://www.santiagoporras.com
Email: santiagoporras@outlook.com
Blogs: http://blog.santiagoporras.com
Twitter: @saintwukong

Coordinadores de sección

GASTÓN CRUZ
Coordinador de PowerBi
gastoncruz@gmail.com

¿Desea colaborar con CompartiMOSS?

La subsistencia del magazine depende de los aportes en contenido de todos. Por ser una revista dedicada a información sobre tecnologías de Microsoft en español, todo el contenido deberá ser directamente relacionado con Microsoft y escrito en castellano. No hay limitaciones sobre el tipo de articulo o contenido, lo mismo que sobre el tipo de tecnología.
Si desea publicar algo, por favor, utilice uno de los siguientes formatos:

- Artículos de fondo: tratan sobre un tema en profundidad. Normalmente entre 2000 y 3000 palabras y alrededor de 4 o 5 figuras. El tema puede ser puramente técnico, tanto de programación como sobre infraestructura, o sobre implementación o utilización.

- Artículos cortos: Artículos cortos: Máximo 1000 palabras y 1 o 2 figuras. Describen rápidamente una aplicación especial de alguna tecnología de Microsoft, o explica algún punto poco conocido o tratado. Experiencias de aplicación en empresas o instituciones puede ser un tipo de artículo ideal en esta categoría.

- Ideas, tips y trucos: Algunos cientos de palabras máximo. Experiencias sobre la utilización de tecnologías de Microsoft, problemas encontrados y como solucionarlos, ideas y trucos de utilización, etc. Los formatos son para darle una idea sobre cómo organizar su información, y son una manera para que los editores le den forma al magazine, pero no son obligatorios. Los artículos deben ser enviados en formato Word (.doc o .docx) con el nombre del autor y del artículo.

Si desea escribir un artículo de fondo o corto, preferiblemente envíe una proposición antes de escribirlo, indicando el tema, aproximada longitud y número de figuras. De esta manera evitaremos temas repetidos y permitirá planear el contenido de una forma efectiva.

Envíe sus proposiciones, artículos, ideas y comentarios a la siguiente dirección:
revista@compartimoss.com adiazcan@hotmail.com
fabiani@siderys.com.uy jcgonzalezmartin1978@hotmail.com
gustavo@gavd.net


Comparti
MOSS