Nº35 marzo 2018

CompartiMOSS

REVISTA ESPECIALIZADA EN TECNOLOGÍAS MICROSOFT

10 años

Entrevista CompartiMOSS

Introducción a PowerShell para Microsoft Teams

Introducción a Azure Durable Functions

Setup Google Analytics on modern SharePoint using SPFx Extensions

Edición especial 10 años

Contenido

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

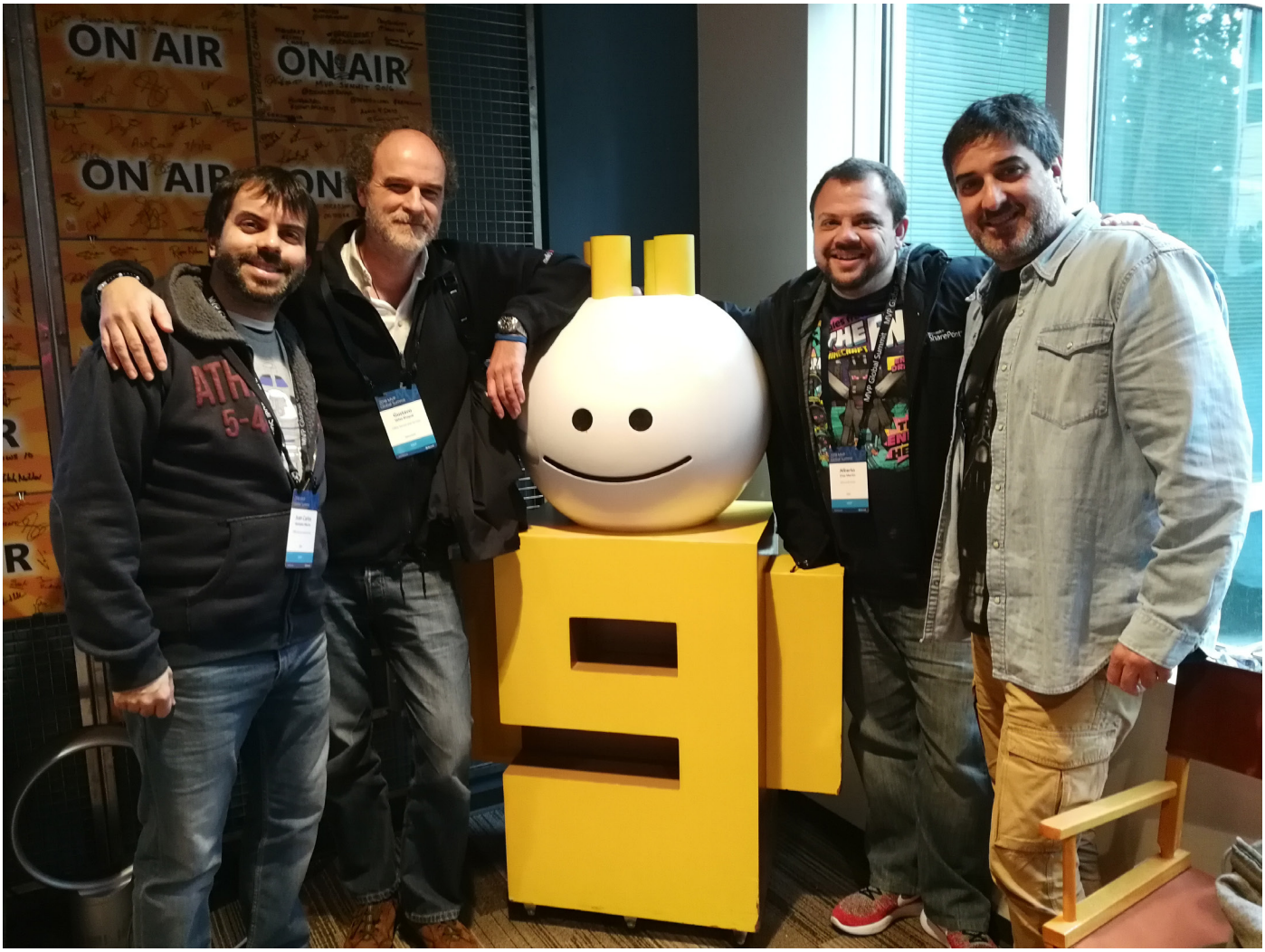
REDES SOCIALES
Facebook:
http://www.facebook.com/group.php?gid=128911147140492
LinkedIn:
http://www.linkedin.com/groups/CompartiMOSS-3776291
Twitter:
@CompartiMOSScom

03

Editorial

El número 35 de CompartiMOSS es un número muy especial tanto para el Equipo Editorial de la revista como para sus autores y lectores ya que la revista cumple 10 años durante los que no hemos dejado de hablar de tecnologías Microsoft desde distintos puntos de vistas. Este cumpleaños tan especial de la revista ha coincidido también con la presencia de todo nuestro equipo en el MVP Summit 2018 en el Campus de Microsoft en Redmond y que hemos celebrado, gracias a Channel 9 y a Cristina González Herrero, realizando un WebCast en el que hemos hablado del presente y futuro de las tecnologías y servicios clave de Microsoft en el momento: Office 365, Azure y SharePoint. También hemos aprovechado tan importante aniversario para que los fundadores de la revista, Gustavo Vélez y Héctor Insua, escribiesen unas palabras de lo que ha sido y sigue siendo la revista de referencia de tecnologías Microsoft en habla hispana. Como siempre, esperamos que disfruten de los artículos preparados por nuestros autores gracias a los cuáles la revista es posible y también a ustedes, nuestros lectores.

El Equipo Editorial de CompartiMOSS



HÉCTOR INSUA

Cuando Fabián me escribió para comentarme de los 10 años de la revista, solo pensé inmediatamente “Que rápido que pasa el tiempo…” 10 años ya, no lo podía creer, y me puse a recordar los orígenes de CompartiMOSS, en esos años, la información de SharePoint en Español estaba centralizada en Gustavo Velez y Luis Du Solier, quienes gentilmente compartían todas sus experiencias con la plataforma, por mi rol en Microsoft en ese momento, ya había comenzado a difundir información para mis Clientes y poco a poco mi blog comenzaba a recibir varios miles de visitas al mes, entré en contacto con Gustavo para proponerle la idea de generar un espacio de difusión de conocimientos en el cual se concentraran las experiencias de quienes estábamos colaborando con Technet, artículos en nuestros blogs y distintos eventos por Latinoamérica, y así surgió la idea de la Revista, al principio no fue fácil, se requerían conocimientos de diseño había que buscar un logotipo, y generar artículos interesantes para una creciente comunidad de usuarios de habla hispana. Y de poco partió, todavía recuerdo el primer número, contactar a otros referentes (Vladimir Medina, Fabian Imaz, Ricardo Muñoz) y se fue generando contenido, Gustavo generó un nuevo espacio en su portal para poder disponibilizar las revistas, y el éxito fue inmediato… varios MVPs colaborábamos con la premisa de ayudar a muchos usuarios, administradores y desarrolladores con SharePoint 2007 y 2010 que recién estaba en proyecto.

Por razones laborales, el tiempo fue escaseando, me cambié de país, llegó mi hija y no pude continuar colaborando como lo había habitualmente, pero siempre estoy vinculado a la plataforma SharePoint, durante todos estos años, ejecutando proyectos, desarrollando soluciones e incorporando nuevas tecnologías al mundo de la Colaboración y la Productividad. Hoy en día me siento orgulloso de que un proyecto que iniciamos sin esperar nada a cambio, con el objetivo de compartir nuestras experiencias buenas y malas, buscando soluciones y ayudando a solucionar problemas de cientos de usuarios que nos escriben, siga en el aire, con muchos colaboradores, y con la mismas buena onda con la que fue concebido.

Solo puedo agradecer a todos quienes tomaron la posta de continuar, evolucionar y hacer crecer Compartimoss ya que el merito de cumplir 10 años es completamente de ellos. En lo personal, intentaré volver a compartir información y nuevamente generar contenido y artículos con la realidad actual de SharePoint, que también 10 años mas maduro, es hoy en día líder en Productividad corporativa.

GUSTAVO VÉLEZ

Desde hace 10 años hasta ahora: como hemos llegado hasta aquí...

Cuando hace 10 años, junto con Héctor Insua, comenzamos con la aventura de CompartiMOSS, ni sabíamos en el lio que nos metíamos, ni teníamos idea hasta donde íbamos a llegar. En un par de palabras, CompartiMOSS ha sido un sitio ideal para conocer un montón de amigos, recorrer un montón más de kilómetros y divertirnos aún más con las tecnologías de Microsoft.

Aunque, desafortunadamente, después de un par de números, Héctor no pudo formar mas parte del grupo de editores de la revista, poco tiempo después de su partida Fabian Imaz se unió a la aventura para continuar el camino conmigo, y muy rápidamente después de eso, Juan Carlos Gonzalez y Alberto Diaz complementaron el grupo que trabaja, organiza, sufre y se ríe cada tres meses con la creación de cada nuevo número. Por supuesto que nosotros cuatro no podríamos hacer este trabajo con CompartiMOSS sin la ayuda de amigos como Santiago Porras, el que le da la presencia visual a la revista y al sitio (llamado injustamente “el salón de belleza” en algún otro lugar en este número), y, por supuesto, todos los autores que han pasado por sus páginas en estos años.

Mirando como comenzamos y como estamos ahora, personalmente puedo decir que la jornada ha sido llena de baches por el camino, pero siempre recompensante y reconfortante. Profesionalmente hablando, ha representado el seguir la marcha, también llena de baches, de Microsoft: pasando del enorme éxito inicial de SharePoint (2003, 2007, 2010), a su decadencia como servidor OnPremises; pasando de una iniciativa en la nube que no era más que pura especulación en el principio, a estar intentado hacer algo interesante (sin mayor éxito) con Office 365; pasando a trabajar cada vez más con Azure y sus servicios llenos de oportunidades; mejor dicho, cambiando continuamente...

Y siempre que se mira hacia atrás, es imposible evitar mirar hacia adelante también. ¿En dónde estará CompartiMOSS cuando cumpla sus 20 años? ¿En dónde estaré yo personalmente, y por donde andará la tecnología en general y la de Microsoft en particular? CompartiMOSS ya ha pasado su infancia, y ahora le llega el tiempo difícil de la pubertad. Es a todos nosotros, editores, colaboradores y autores, hacer que su adolescencia sea tan exitosa como su niñez.

05

Introducción a PowerShell para Microsoft Teams

Una de las características más demandadas en Microsoft Teams no disponible hasta hace relativamente poco tiempo es la posibilidad de poder gestionar y administrar la plataforma por medio de PowerShell. Afortunadamente, para dar respuesta a este requerimiento Microsoft liberó en noviembre de 2017 una primera versión de los comandos PowerShell para Microsoft Teams que, aunque actualizada en diciembre del mismo año, aún tiene mucho margen de mejora como veremos en este artículo.

Pre-requisitos para trabajar con PowerShell para Microsoft Teams

Para poder comenzar a administrar y gestionar Microsoft Teams por medio de PowerShell, lo primero que tendremos que hacer es descargarnos e instalar la versión más reciente del Módulo de PowerShell para Microsoft Teams:

- La versión más reciente del módulo de PowerShell para Microsoft Teams disponible a la fecha de redacción de este artículo es la 0.9.1 y para poder descargarla e instalarla hay que seguir el procedimiento detallado en la PowerShell Gallery: https://www.powershellgallery.com/packages/MicrosoftTeams/0.9.1
- De acuerdo con las instrucciones de instalación, para instalar el módulo de PowerShell para Microsoft Teams simplemente necesitamos abrir la consola de Windows PowerShell en nuestro equipo y ejecutar la siguiente instrucción:

```
Install-Module -Name MicrosoftTeams
```

- Cuando se solicite, confirmamos que deseamos instalar el módulo desde el repositorio de la PowerShell Gallery:

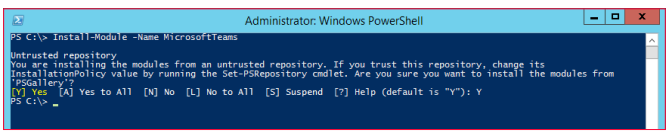


Imagen 1.- Instalación del módulo de PowerShell para Microsoft Teams.

- Una vez instalado, podemos de forma sencilla obtener el listado completo de comandos disponibles. Como se puede apreciar en la Figura 1, la versión 0.9.1 de los comandos PowerShell para Microsoft Teams cuenta con un total de 23 cmdlets.

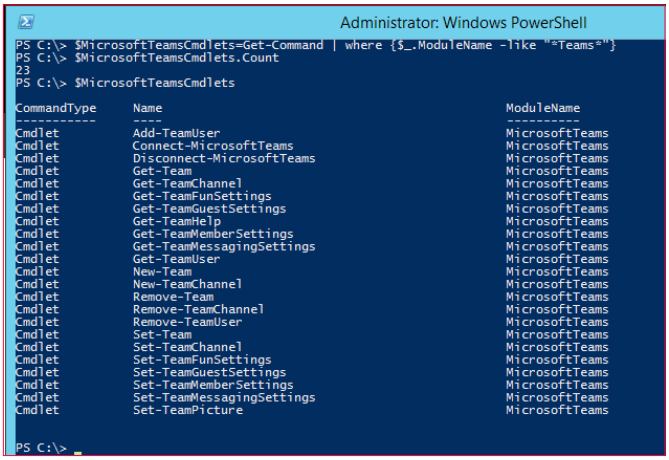


Imagen 2.- Listado de comandos PowerShell disponibles para Microsoft Teams.

Conectándose a Microsoft Teams por medio de PowerShell

Para conectarnos a Microsoft Teams por medio de PowerShell podemos hacer uso del cmdlet Connect-MicrosoftTeams tal y como muestra el siguiente código:

```
$sUserName=”jcgonzalez@nuberosnet.onmicrosoft.com”
$sMessage=”Type your Office 365 credentials”
$TeamsCredentials=Get-Credential -UserName $sUserName
-Message $sMessage
Connect-MicrosoftTeams -Credential $TeamsCredentials
```

Como resultado de la ejecución del código anterior, se muestra por pantalla la cuenta que se está conectando a Microsoft Teams así como los valores de las siguienes propiedades:

- Environment
- Tenant.
- TenantId.
- TenantDomain.

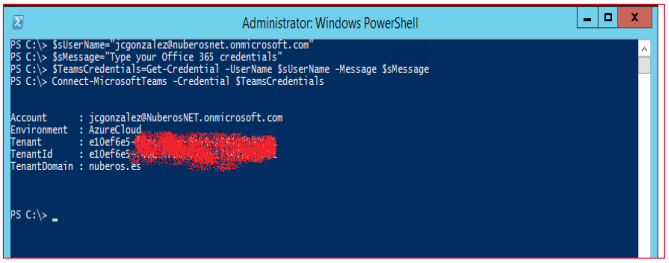


Imagen 3.- Salida por pantalla resultante de ejecutar el cmdlet Connect-MicrosoftTeams.

Operaciones básicas con Teams utilizando PowerShell

El comando Get-Team permite obtener el listado de Teams a los que pertenece el usuario actual y no todos los Teams del tenant como cabría esperar. Para cada Team se devuelven las siguientes propiedades:

- El ID del Grupo de Office 365 vinculado al Team.
- El nombre para mostrar del Team.
- La Descripción del Team.

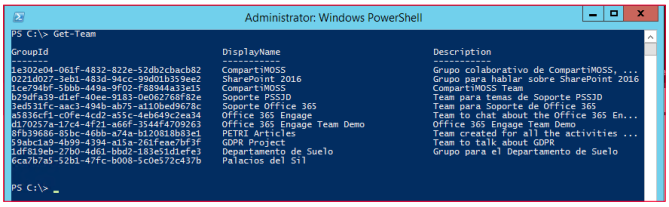


Imagen 4.- Listado de Teams a los que pertenece el usuario.

El comando Get-TeamUser permite obtener los integrantes de un Team (Propietarios, Miembros e Invitados) identificado por medio del identificador de Grupo (GroupId). Por ejemplo, para obtener las propiedades User, Name y Role de los integrantes de un Team basta con ejecutar las siguientes sentencias PowerShell:

```
$GroupID="1e302e04-061f-4832-822e-52db2cbacb82"
Get-TeamUser -GroupID $GroupID | Format-Table User,Name,Role
```

La salida de pantalla que se obtiene al ejecutar el código PowerShell anterior es el siguiente:

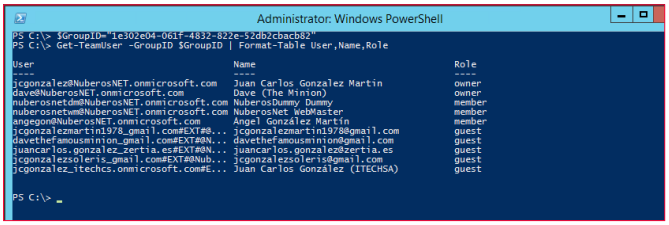


Imagen 5.- Resultado de ejecutar el cmdlet Get-TeamUser.

El cmdlet New-Team permite crear un nuevo Team en el tenant indicando o bien el Grupo de Office 365 a partir del qué se va a crear el Team o bien el nombre para mostrar (DisplayName). Opcionalmente se puede indicar también valores para los parámetros Alias y Description del comando.

```
$sTeamName="SharePoint Saturday Events"
New-Team -DisplayName $sTeamName
```

Una vez ejecutado el comando New-Team, se visualiza por pantalla el identificador del Grupo de Office 365 asociado al Team:

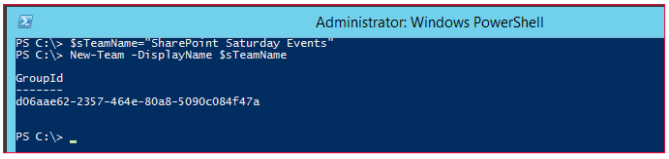


Imagen 6.- Resultado de la ejecución del comando New-Team.

Una secuencia PowerShell más completa para crear un nuevo Team con New-Teams es la siguiente:

```
$sTeamName="SharePoint & Office 365 Madrid 2018"
$sTeamAlias="SPSO365Madrid2018"
$sTeamDescription="Team for the SharePoint & Office 365 Madrid 2018"
$Classification="Internal Only"
New-Team -DisplayName $sTeamName -Alias $sTeamAlias -Description $sTeamDescription -AccessType Private -Classification $Classification -AddCreatorAsMember $False
```

Como vemos, la secuencia anterior permite crear un nuevo Team indicado nombre, alias, tipo de acceso (privado), escala de clasificación ("Internal Only") y también qué el usuario que está creando el Team no será añadido como propietario al Team.

"Para administrar y gestionar Microsoft Teams por medio de PowerShell, tendremos que instalar la versión más reciente del Módulo de PowerShell para Microsoft Teams"

Para añadir un nuevo usuario de tipo integrante a un Team utilizaremos el cmdlet Add-TeamUser de la siguiente forma:

```
$GroupID="1e302e04-061f-4832-822e-52db2cbacb82"
$sUserName="agonzalez@nuberos.es"
Add-TeamUser -GroupId $GroupID -User $sUserName -Role Member
```

De la misma forma, el comando Remove-TeamUser nos permite eliminar un usuario de acuerdo con las siguientes sentencias PowerShell:

```
$GroupID="1e302e04-061f-4832-822e-52db2cbacb82"
$sUserName="agonzalez@nuberos.es"
Remove-TeamUser -GroupId $GroupID -User $sUserName
```

Operaciones básicas con canales de Teams utilizando PowerShell

De la misma forma que podemos realizar operaciones básicas con Teams en nuestro tenan utilizando PowerShell, podremos hacer uso también de PowerShell para realizar operaciones con calanes de un Team:

- El comando Get-TeamChannel nos permite obtener los canales de un Team concreto identificado por medio del parámetro GroupId:

```
$GroupID="1e302e04-061f-4832-822e-52db2cbacb82"
Get-TeamChannel -GroupId $GroupID
```

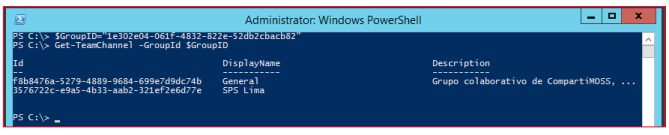


Imagen 7.- Listado de canales en un Team.

- Para crear un nuevo canal en un Team disponemos del cmdlet New-TeamChannel:

```
$GroupID=”d06aae62-2357-464e-80a8-5090c084f47a”
$sChannelName=”SharePoint Saturday Madrid 2018”
$sTeamDescription=”Canal para el evento SharePoint Saturday Madrid 2018”
New-TeamChannel -GroupId $GroupID -DisplayName $sChannelName -Description $sTeamDescription
```

- El cmdlet Set-TeamChannel permite actualizar un canal de un Team de acuerdo con la siguiente sintaxis:

```
$GroupID=”d06aae62-2357-464e-80a8-5090c084f47a”
$sCurrentChannelName=”SharePoint Saturday Madrid 2018”
$sNewChannelName=”SharePoint & Office 365 Saturday Madrid 2018”
Set-TeamChannel -GroupId $GroupID -CurrentDisplayName $sChannelName -NewDisplayName $sNewChannelName
```

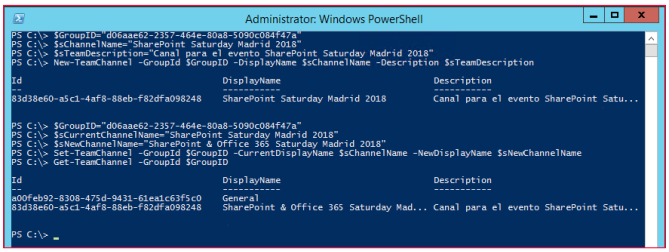


Imagen 8.- Actualización de un canal utilizando Set-TeamChannel.

- Finalmente, el comando Remove-TeamChannel se puede utilizar para borrar un canal en un Team.

Comandos PowerShell para acceder a las configuraciones de Teams

“Desde el pasado mes de noviembre disponemos de una primera versión (Beta) del módulo de comandos PowerShell para Microsoft Teams”

Para finalizar el artículo, a continuación, se destallan algunos de los comandos que permiten interactuar con un Team desde el punto de vista de su configuración:

- Para visualizar las configuraciones de lo que pueda hacer un integrante de un Team disponemos del comando Get-TeamMemberSettings:

```
$GroupID=”d06aae62-2357-464e-80a8-5090c084f47a”
Get-TeamMemberSettings -GroupId $GroupID
```

Como resultado se mostrarán las configuraciones aplicadas a integrantes del Team:

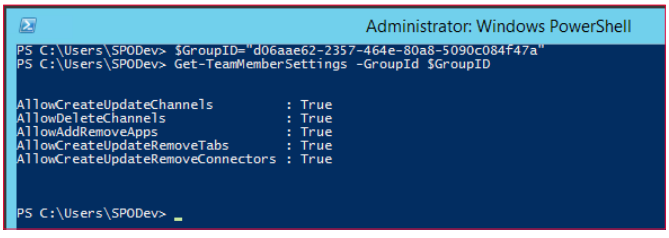


Imagen 9.- Configuraciones relativas a integrantes de un Team.

- Para cambiar las configuraciones de un integrante de un Team se puede utilizar el comando Set-TeamMemberSettings:

```
$GroupID=”d06aae62-2357-464e-80a8-5090c084f47a”
Set-TeamMemberSettings -GroupId $GroupID -AllowDeleteChannels $false
Get-TeamMemberSettings -GroupId $GroupID
```

- El comando Get-TeamMessagingSettings permite visualizar las configuraciones a nivel de mensajes en Teams:

```
$GroupID=”d06aae62-2357-464e-80a8-5090c084f47a”
Get-TeamMessagingSettings -GroupId $GroupID
```

- Las configuraciones de los mensajes en un Team se pueden modificar por medio de Set- TeamMessagingSettings.
- Las configuraciones de usuarios invitados en Teams, así como uso de Stickers & Memes se pueden obtener con los comandos Get-TeamGuestSettings y Get-TeamFunSettings.

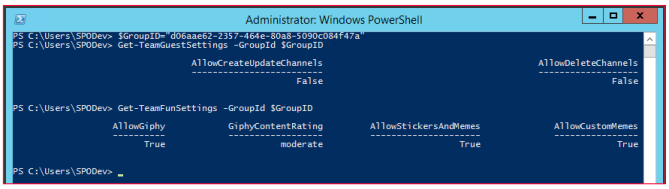


Imagen 10.- Resultados de ejectutar Gest-TeamGuestSettings y Get-TeamFunSettings.

Conclusiones

Desde el pasado mes de noviembre disponemos de una primera versión (Beta) del módulo de comandos PowerShell para Microsoft Teams que fue actualizado el pasado mes de diciembre. Los comandos PowerShell disponibles para Microsoft Teams permiten realizar operaciones básicas con Teams, Canales de Teams y las configuraciones de un Team y presentan ciertas limitaciones como por ejemplo la imposibilidad de listar todos los Teams de un tenant con el comando Get-Team.

Referencias

- A Teams PowerShell Primer: https://www.petri.com/teams-powershell-primer
- Why the PowerShell Module for Teams is Critically Flawed: https://www.petri.com/powershell-module-teams-critically-flawed
- Documentación de los commandos PowerShell para Microsoft Teams: https://docs.microsoft.com/en-us/powershell/module/teams/?view=teams-ps

JUAN CARLOS GONZÁLEZ MARTÍN
Office Servers and Services MVP
Cloud & Productivity Advisor
jcgonzalezmartin1978@hotmail.com
@jcgm1978
https://jcgonzalezmartin.wordpress.com/

08

Haciendo uso de los Cognitive Services desde SPFx – Text Analytics API

En este artículo, vamos a seguir con los MS Cognitive Services desde SPFx. En este caso hablaremos de la Text Analytics API, que como podemos deducir de su nombre, nos va a dar capacidades de análisis de texto. La API ofrece 3 endpoints:

- Autodetección de Idioma: Podemos detectar el idioma en el que está el texto analizado.
- Keywords: Palabras clave del texto analizado.
- Sentimiento: Indica cuánto de sentimiento negativo o positivo hay en el texto analizado.

"El score del análisis es un porcentaje representado por un valor entre 0 y 1"

Como parte del artículo vamos a desarrollar un "spfx field customiser", que mostrará un emoticono resultado de analizar el sentimiento del texto proporcionado en un campo de la lista. Además, utilizaremos la función de autodetección de idioma, para previamente a invocar la función de sentimiento, sepamos en qué idioma está escrito el texto.

La siguiente imagen muestra el resultado final:

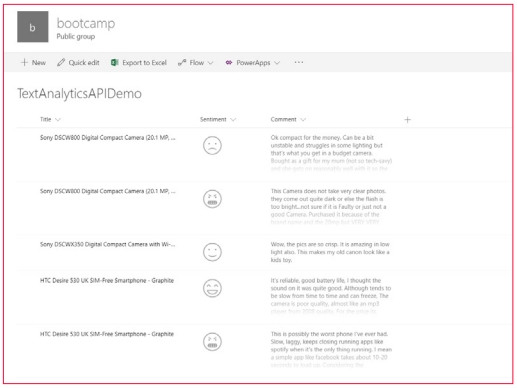


Imagen 1.- Resultado final obtenido.

Registrar servicio Cognitive Services Vision API

Antes de entrar en código, recordemos del artículo anterior que para poder utilizar la Text Analytics API de Cognitive Services, primero tenemos que registrar el servicio desde el portal de Azure, para así obtener la Key para invocar la API.

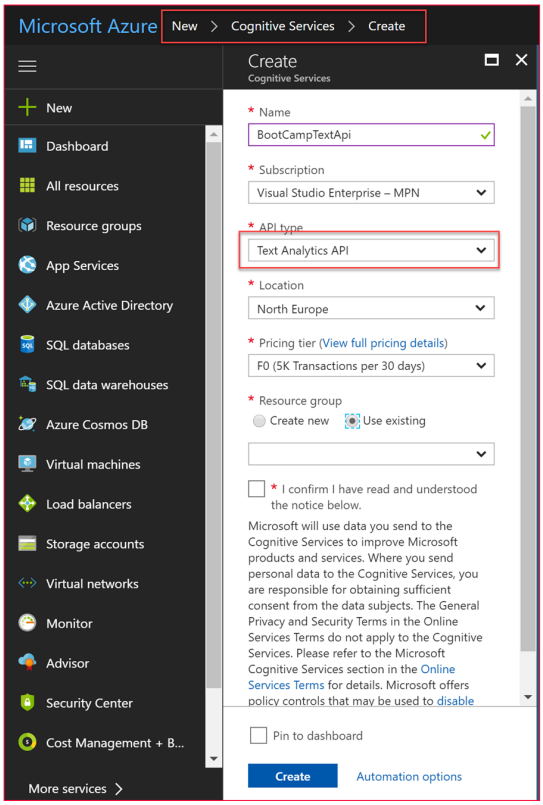


Imagen 2.- Registro de la Text Analytics API.

Nota: Debemos seleccionar expresamente el servicio que queremos utilizar, en este caso, Text Analytics API. Si queremos hacer uso de varios servicios, hay que registrarlos expresamente, y obtener una Key diferente para cada uno.

Una vez registrado, podemos obtener la Key desde la sección Keys del servicio:

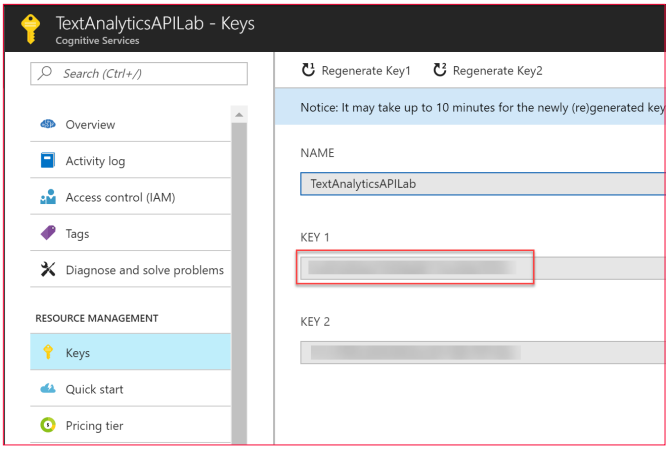


Imagen 3.- Obteniendo las claves de Text Analytics.

Nota sobre la Tenant Properties API GetStorageEntity

En el artículo anterior, vimos como almancear la API Key como una Tenant property, de tal manera que la Key se recuperaba desde código, evitando hard-codearla en TypeScript. Este método sigue siendo válido, pero cabe destacar que actualmente hay un bug reconocido en la API que recupera un Tenant Property y que no funciona cuando el usuario es un usuario externo. El bug está registrado en GitHub y por el momento no ha sido corregido todavía. Tenéis más información en el siguiente link:

https://github.com/SharePoint/sp-dev-docs/issues/1137

"hemos creado un nuevo componente de React que hace la traducción entre el número y el icono"

Una alternativa a utilizar la Tenant properties API para guardar la Key del Text-Analytics API es proporcionar la Key como Property de la SPFx extensión. El siguiente fragmento muestra como configurarlo desde el serve.json para debug en SPO Workbench:

```
"serveConfigurations": {
  "default": {
    "pageUrl": "https://[TENANT].sharepoint.com/sites/ModernTeamSite/Lists/TextAnalyticsDemo/AllItems.aspx",
    "fieldCustomizers": {
      "SPFxSentimentIcon": {
        "id": "49407339-ef31-4775-b341-6479f2d8e25d",
        "properties": {
          "textAnalyticsApiKey": "[TEXT_API_KEY]",
          "fieldToAnalyse": "Title"
        }
      }
    }
  },
```

Posteriormente en nuestro código de la Extensión, podemos acceder al valor con un simple:

```
this.properties.textAnalyticsApiKey
```

Invocando la API para auto detección de idioma

Antes de nada, para invocar cualquier método de la Text Analytics API, tenemos que usar la siguiente URL base, que además, va a depender de la zona en la que hayamos registrado el servicio en el portal de Azure. En este caso estamos haciendo uso de la zona "Westus"

```
private cognitiveServicesTextUrl: string = `https://westus.api.cognitive.microsoft.com/text/analytics/v2.0/`;
```

Desde esa URL base, para detectar el idioma del texto que queremos analizar, podemos hacer una llamada al endpoint de "languages"

```
private async _autodetectLanguage(text: string, id: string): Promise<string> {

  if (this.props.textAnalyticsApiKey !== '') {
    const httpOptions: IHttpClientOptions = this._prepareHttpOptionsForTextApi(text, id);
    const cognitiveResponse: HttpClientResponse = await this.props.httpClient.post(`${this.cognitiveServicesTextUrl}/languages`, HttpClient.configurations.v1, httpOptions);
    const cognitiveResponseJSON: any = await cognitiveResponse.json();

    console.log(cognitiveResponseJSON);

    const language = cognitiveResponseJSON.documents[0].detectedLanguages[0].iso6391Name; //getting first language detected (i.e: "en")

    return language;
  }

  return null;
}
```

<u>Nota:</u> Recordar del artículo anterior, que podemos enviar varios textos a analizar en la misma petición (es lo que la API llama "documents"). Además, el texto puede contener diferentes idiomas, de ahí que en código estemos accediendo al primer idioma detectado:

```
detectedLanguages[0]
```

Y de ahí, a su código ISO (i.e: en para inglés, es para español, etc.)

Invocando a la API para análisis de sentimiento

Una vez que tenemos claro el idioma del texto, ya podemos invocar al endpoint de análisis de sentimiento, la siguiente imagen muestra el código más relevante:

```
const cognitiveResponse: HttpClientResponse = await this.props.httpClient.post(`${this.cognitiveServicesTextUrl}/sentiment`,
  HttpClient.configurations.v1,
  httpOptions);

  const cognitiveResponseJSON: any = await cognitiveResponse.json();

const score = cognitiveResponseJSON.documents[0].score;
```

El "score" es un número decimal entre 0-1, donde 0 sería un sentimiento muy negativo, y 1 un sentimiento muy positivo (un valor entre medias indicaría un comentario neutro, donde no se expresa ningún tipo de sentimiento).

Pintando el emoticono del sentimiento

Para acabar, ya sólo nos queda pintar el "score" como emoticono. Para ello, hemos creado un nuevo componente de React que hace la traducción entre el número y el icono, aplicando nuestras propias reglas de negocio (por ejemplo, es responsabilidad de cada uno decidir si quiere considerar una puntuación de 0.6 como positiva, neutra, muy positiva, etc.

```
export default class SentimentIcon extends React.Component<ISentimentIconProps, {}> {

    private ERROR_ICON: string = '
    private LOADING_ICON: string = '
    private VERY_HAPPY_FACE_ICON: string = '
    private HAPPY_FACE_ICON: string = '
    private NEUTRAL_FACE_ICON: string = '
    private SAD_FACE_ICON: string = '
    private VERY_SAD_FACE_ICON: string = '

    @override
    public render(): React.ReactElement<{}> {
      const image = this._scoreToIcon(this.props.score);
      return (
        <img src={image} data-sentiment={this.props.score} />
      );
    }

    private _scoreToIcon(score: number): string {
        const percentage = score * 100;

        if (percentage < 0) return this.ERROR_ICON;
        if (percentage == 0) return this.LOADING_ICON;
        if (percentage > 90) return this.VERY_HAPPY_FACE_ICON;
        if (percentage < 90 && percentage > 60) return this.HAPPY_FACE_ICON;
        if (percentage < 60 && percentage > 20) return this.NEUTRAL_FACE_ICON;
        if (percentage < 20 && percentage > 10) return this.SAD_FACE_ICON;
        if (percentage < 10) return this.VERY_SAD_FACE_ICON;
      }
}
```

Como podéis ver, primero hemos definido unas constantes con los diferentes iconos que queremos utilizar (en formato base64, pero también podrían ser URLs a imágenes).

> *"El proyecto también ha sido aceptado como contribución en el repositorio de SharePoint"*

Finalmente, definimos las diferentes escalas que queremos aplicar a cada icono, según la puntuación de sentimiento obtenida.

Código fuente disponible en GitHub

Tenéis todo el proyecto en mi cuenta de GitHub:

https://github.com/luismanez/office365-developer-boot-camp-2017

El proyecto también ha sido aceptado como contribución en el repositorio de SharePoint, así que también podéis encontrarlo en el siguiente link:

https://github.com/SharePoint/sp-dev-fx-extensions/tree/master/samples/react-command-vision-api

---

LUIS MAÑEZ
SharePoint/Cloud Solutions Architect en ClearPeople LTD
@luismanez
http://geeks.ms/lmanez/

11

Trigger serverless en una BBDD no SQL en Azure

Durante muchos años los modelos relacionales han sido el pan nuestro de cada día, y alrededor de ellos se han ido incorporando infinidad de procesos para poder tanto insertar como extraer datos de forma sencilla. Hablamos de las famosas ETL, los servicios de Reporting, los Jobs de nuestros SQL Server, procedimientos almacenados y como no nuestros queridos Trigger.

Los años han pasado, las arquitecturas han cambiado, pero no las necesidades de saber si un evento se ha producido sobre nuestros datos, esa necesidad de controlar los cambios no ha desaparecido, pero por suerte si han evolucionado las técnicas.

¿Qué entendemos por Trigger serverless?

Actualmente y no solo en el mundo de la tecnología, tenemos la necesidad de "imponer" nombres a cosas que antes dábamos por supuestas. Esto genera un fenómeno de novedad, que a veces asusta, ya que por ejemplo conceptos como DevOps, ALM o Serverless no son tan novedosos como el mercado pinta, ya que en menor o mayor medida todo desarrollador o arquitecto serio ya los aplicaba en la fase de diseño e implementación de sus aplicaciones.

> *"En Azure si hablamos de servicio de base de datos no relacional sin duda debemos pensar en Cosmos DB"*

Me voy a centrar en "definir" a mi manera lo que entiendo como Trigger serverless, ya que es muy importante que todos entendamos lo mismo para luego darle el mismo uso.

Para mí el serverless en general (ya sea una arquitectura, un evento o un proceso) me lleva en primera instancia a pensar en un sistema en el cual no nos tiene que preocupar la parte de infraestructura del mismo, que no sabemos donde se aloja, sobre que maquinas se ejecuta, y que tipo de hardware necesita, simplemente funciona y funciona como nosotros queremos.

Pero como soy un romántico, esto sería muy sencillo y poco clasificatorio en mi opinión, por lo que a mí me gusta pensar que serverless es la definición de un sistema del cual no sabemos sus límites de escalado, volúmenes de ejecución por minuto ni numero de usuarios que lo van a consumir, y aun así estoy 100% tranquilo de que va a funcionar un 99,99% de las veces.

Por tanto, uniendo todo un poco, para mi un evento serverless, es un evento que se produce bajo un cambio en un sistema X y se ejecuta sobre un proceso muy potente en un sistema serverless Y.

Conectando tu CosmosDB con Azure Function

En Azure si hablamos de servicio de base de datos no relacional sin duda debemos pensar en Cosmos DB, que nos permite entre otros muchos modelos crear una un Cosmos DB SQL, que no es más que el antiguo Document DB pero que permite realizar consultas SQL sobre nuestras colecciones de documentos. Por otro lado, el mayor exponente de un servicio serverless en Azure son las Azure Functions, y como es de esperar es muy sencillo conectar una Comos DB Document DB API a un Azure Functions sin tener que desarrollar nada de inicio.

Imagen 1.- Cosmos y Functions.

Lo primero vamos a crear una Azure Functions en blanco y la vamos a llamar compartimossFunctions. Accedemos a nuestra suscripción de Azure y seleccionamos y creamos una nueva Azure Functions tal y como se ve en la imagen.

En mi caso he elegido un plan por consumo, pero es indiferente para el ejemplo.

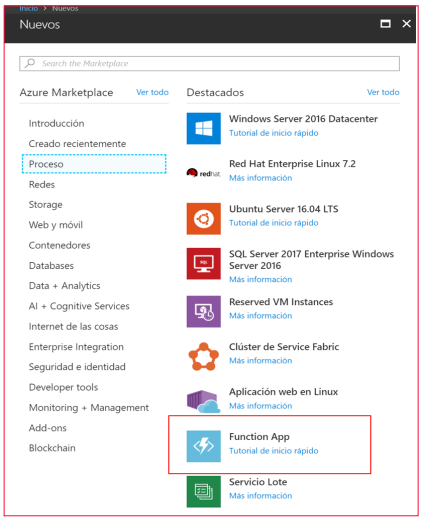


Imagen 2.- Creando una función.

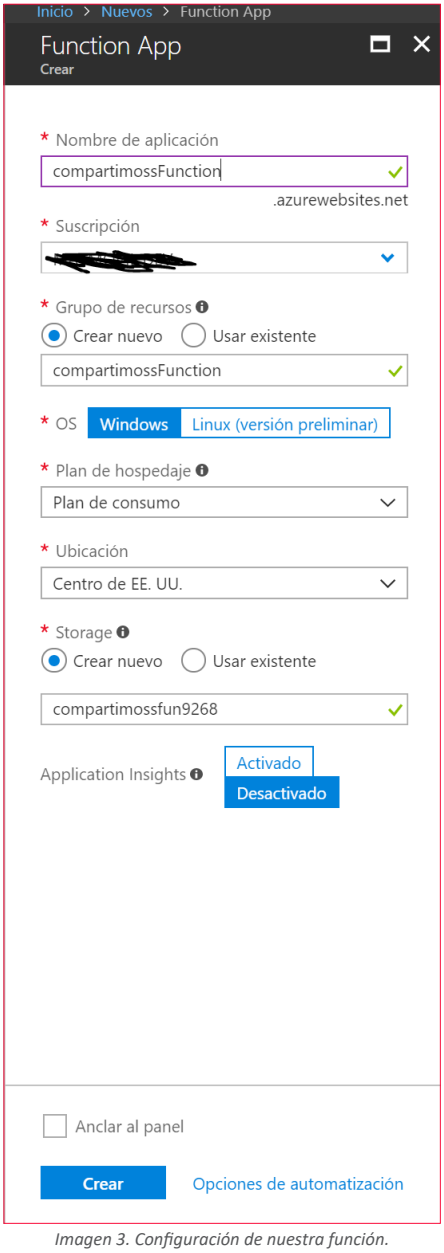


Imagen 3. Configuración de nuestra función.

Una vez creado el servicio de Azure Function CompartimossFunctions, si ya tenemos creada una collection de Document DB o SQL API (como se puede ver en este ejemplo), debemos acceder a nuestro servicio de SQL API , y en el menú generar seleccionamos “Agregar Función de Azure”.

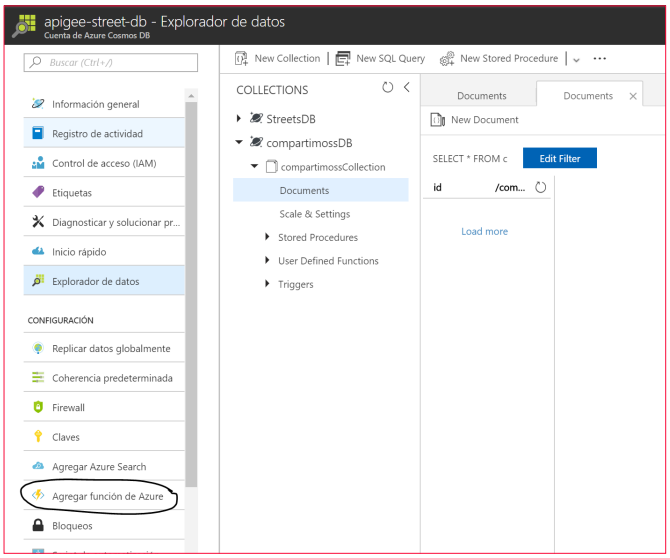


Imagen 4.- Agregando la función.

Como se ve en la figura anterior, en mi servicio he creado una collection CompartimossCollection. Debemos seleccionar la colección sobre la que vamos a crear el evento, la functions que hemos creado en el paso anterior, y por último el lenguaje.

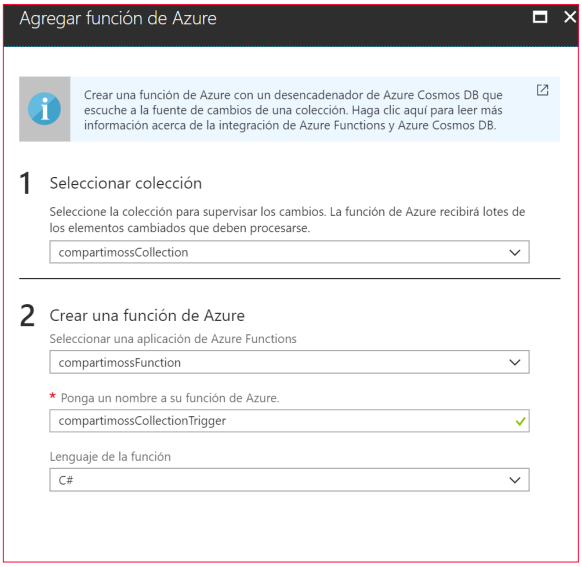


Imagen 5.- Configuración de la función.

En cuanto termine el proceso de creación que es instantáneo, se nos redirige a la functions CompartimossFunction que hemos enlazado. Podemos comprobar que se ha creado una función nueva “compartimossCollectionTrigger”, con el siguiente código de ejemplo:

```
#r “Microsoft.Azure.Documents.Client”
using Microsoft.Azure.Documents;
using System.Collections.Generic;
using System;
public static async Task Run(IReadOnlyList<Document> input,
TraceWriter log)
{
  log.Verbose(“Document count “ + input.Count);
  log.Verbose(“First document Id “ + input[0].Id);
}
```

Además, se ha creado un trigger, que vincula a nuestra colección de CosmosDB con nuestra functions. La ventaja de

hacer este ejemplo así es que nos evitamos tener que configurar las conexiones de nuestro trigger de forma manual desde el Azure Functions.

Para testear que todo funciona correctamente, deberemos acceder al explorador de datos de nuestro Cosmos, y añadir un documento nuevo a nuestra colección.

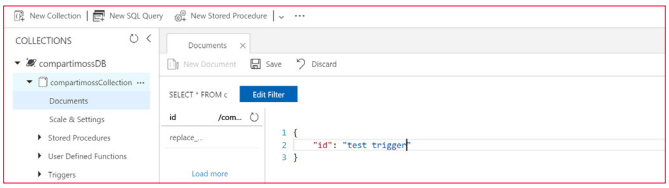


Imagen 6.- Agregando un documento.

Si accedemos a "supervisar" nuestra función veremos que se ha ejecutado correctamente y ha dejado traza de la subida del documento.

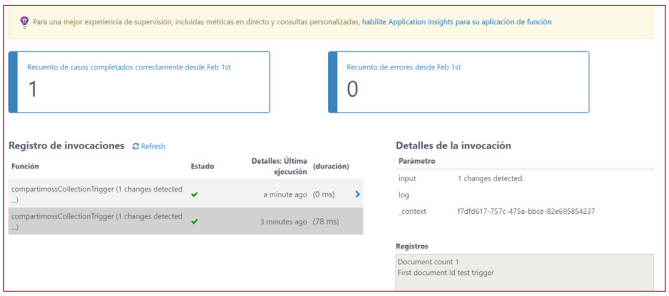


Imagen 7.- Revisión de la ejecución de la función.

Este proceso se podría hacer tanto desde la propia creación de la función, como desde Visual Studio, teniendo el mismo resultado. Aunque para el api de SQL funciona correctamente usando colecciones de documentos, no podemos decir lo mismo para por ejemplo un TABLE API, ya que el código C# generado de ejemplo no funciona correctamente, teniendo que adaptarlo para soporta un CosmosDB Trigger.

Logic App, la opción "no code" para Jobs de integración

Para el que no conozca lo que es una Logic App, se puede resumir como un servicio de Azure PaaS que nos permite diseñar workflows con una interfaz muy potente, y mediante conectores y llamadas a servicios realizar procesos de integración entre diferentes sistemas.

Hoy en día Logic App no tiene un desencadenador o un trigger por defecto contra CosmosDB, pero si tenemos un conector que nos permite entre otras cosas manipular documentos de nuestras colecciones. Por eso podríamos definir un proceso que integre nuestro CosmosDB SQL del ejemplo anterior, con cualquier otro sistema como un Sharepoint o un Dynamics entre otros.

Vamos a ver un ejemplo de cómo obtener todos los documentos de una colección de Cosmos DB.

Para ello vamos a seguir los siguientes pasos:

1.– Elegimos como desencadenador una periodicidad de 3 minutos.

2.– Añadimos una nueva acción "ComosDB" para consultar un documento:

- Nos pedirá que creemos una nueva conexión como se ve en la siguiente imagen.

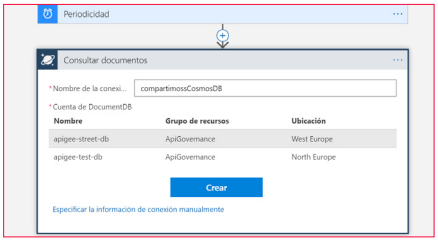


Imagen 8.- Conexión de Cosmos DB en Logic Apps.

- Una vez configurada la conexión seleccionamos la acción "Consultar documentos de CosmosDB", y configuramos la acción según la imagen.

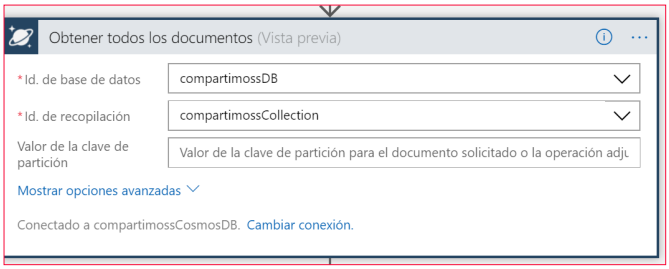


Imagen 9.- Get documents desde Cosmos DB en Logics Apps.

- Para poder mapear la consulta a una estructura JSON utilizamos la acción REDACTAR que nos permite mapear el cuerpo de la consulta a una estructura más legible. Quedando un flujo como el siguiente:

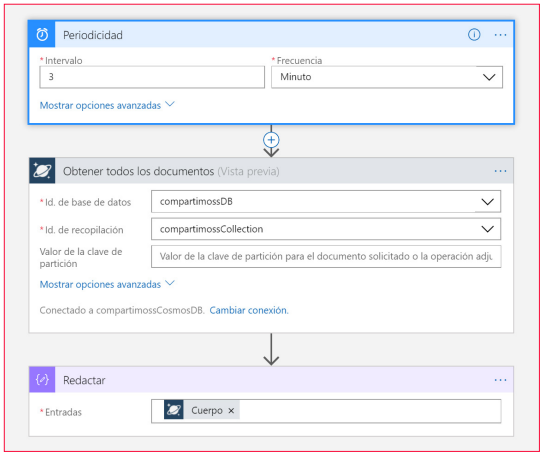


Imagen 10.- Ejemplo final de Logic Apps.

"una Logic App se puede resumir como un servicio de Azure PaaS que nos permite diseñar workflows con una interfaz potente, y mediante conectores y llamadas a servicios realizar procesos de integración entre diferentes sistemas"

Si guardamos el flujo, y seleccionamos ejecutar, veremos el resultado de la ejecución y en concreto obtendremos todos los documentos de la colección.

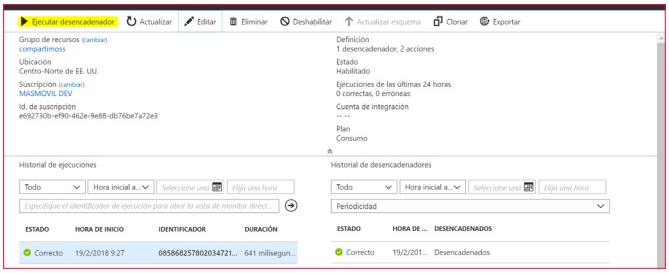


Ilustración 11.- Ejecuciones de tu Logic Apps.

Entrando en el detalle de la ejecución, podemos ver los resultados de cada Acción o Paso, y en concreto investigar el contenido de nuestra estructura JSON de resultado.

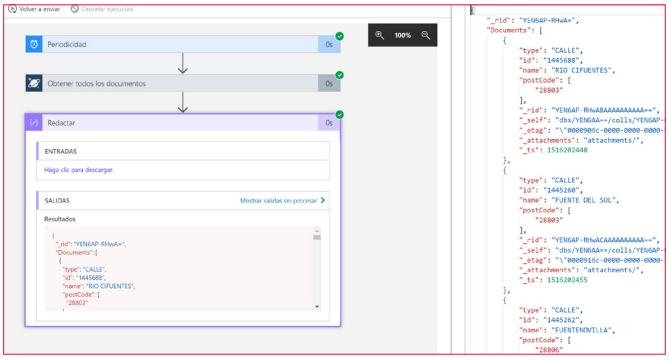


Imagen 12.- Detalle del json de resultado.

¿Cómo se produce el evento?: Fuente de cambios

Hemos visto un par de ejemplos de cómo integrarnos con nuestro servicio de Cosmos DB, permitiendo extender nuestra base de datos no relacional con procesos serverless, dotando de potencia a nuestras aplicaciones. Pero si recordamos con Azure Functions si hemos podido crear un evento y suscribirnos por así decirlo a nuestro Cosmos DB pero con nuestro workflow en Logic App no, ¿Esto por qué sucede?

A día de hoy el sistema que queremos sincronizar con el servicio de CosmosDB debe ser "compatible" con la llamada Fuente de Cambios. La fuente de cambios guarda cada uno de los cambios de nuestros documentos, de forma ordenada y se pueden procesar de forma asíncrona. Los cambios están disponibles para cada intervalo de claves de partición dentro de una colección de documentos.

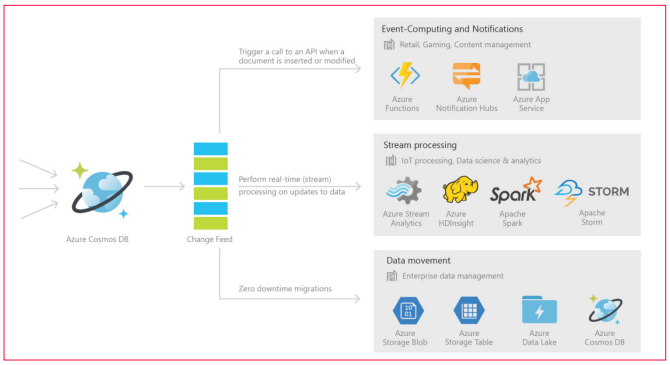


Imagen 13.- Fuente de cambios de Cosmos DB.

Si observamos el anterior diagrama, para Azure Functions si existe un trigger desde la Fuente de Cambios, y es la propia fuente la que realiza una llamada a nuestra función cuando un documento es modificado o insertado.

> *"el sistema que queremos sincronizar con el servicio de CosmosDB debe ser "compatible" con la llamada Fuente de Cambios"*

Mejorando el pasado, disfrutando el presente, en busca del futuro

Hemos oído muchas veces que plagar de procedimientos almacenados, o de triggers nuestras bases de datos era peligroso o incluso una muy mala práctica.

Esto provocó que viviésemos durante años peleados con servicios de integración y de reporting, que eran difíciles de implementar y por si fuera poco aun más de monitorizar.

Bueno, pues el presente es evidente que nos permite recuperar esos trabajos de integración y de extracción, incluso añadir lógica de negocio a nuestros datos, sin necesidad de hacer aplicaciones complejas, con estos pequeños procesos serverless que ya Azure nos proporciona podemos crear aplicaciones complejas y sobre todo asegurando una performance muy alta.

¿Dónde está el futuro?, en mi opinión orquestando estos eventos, teniendo control de tus datos y de sus procesos, uniendo estas piezas para crear una verdadera arquitectura serverless a prueba de bombas.

SERGIO HERNÁNDEZ MANCEBO
Principal Team Leader en Encamina
@shmancebo

¡Skype ha muerto!!Viva MS Teams!

Desde las primeras aproximaciones de Microsoft a las comunicaciones en tiempo real, se ha intentado cubrir todas las necesidades que pudiera requerir el ámbito empresarial. Naciendo con un chat y presencia, hasta ir agregando todas esas funcionalidades que tanto apreciamos (o tal vez no tanto) de nuestro Skype for Business.

Pero para decirle adiós primero tendríamos que entender desde donde vino y con ello entender porque se va.

¿Quién escucho de LCS? ¿y de OCS? ¿Y muchos más ya están familiarizados con Lync y Skype for Business… pero que tienen en común todas estas herramientas? (el nombre no es algo que compartan) Todos ellos han sido evoluciones de un mismo producto (Communication Server) de Microsoft, si bien el re-branding no ha sido del todo acertado, lo que si lo ha sido es la funcionalidad que va agregando versión tras versión. No solo es cambiarle el nombre y el look-and-feel, sino que realmente iban agregando valor al pasar de una versión a otra, y no solo para mantenerte actualizado.

Entonces estamos hablando que desde LCS 2003 hasta la versión SFB 2015 se ha venido evolucionando un producto, hasta el cliente que tenemos hoy en día. Pero ya han pasado 3 años desde la última versión. También estamos en la revolución de la nube, en la lucha por los XaaS (X as a Service) , serverless, cloud cloud cloud, etc.

Si toda la infraestructura ha ido evolucionando, y llevándonos a soluciones que anteriormente no se veían prometedoras. ¿Las comunicaciones tendrían que mantenerse tal y como son hoy en día?

Si bien varios aun desean tener todo por escrito en un correo, los mensajes instantáneos y las llamadas lograron quitarle un poco el trabajo a todo ese engorroso trabajo de redactar un correo para pedir una autorización, para pedir un Vo.Bo.

La comunicación unilateral no tenía más futuro, necesitábamos colaborar y comunicarnos todos.

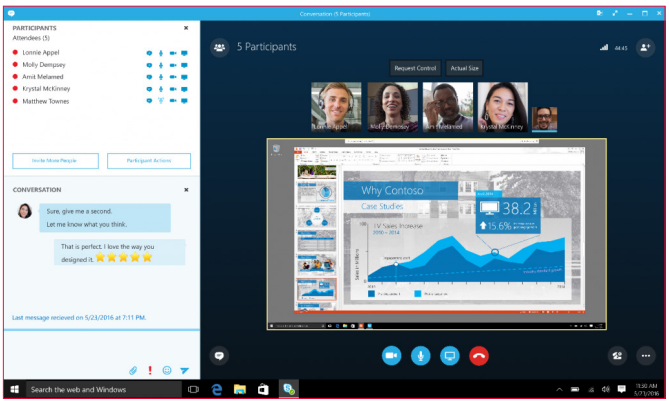


Imagen 1.- Reunión en Skype for Business.

También esto fue debido a un cambio generacional, la mentalidad va cambiando de acuerdo a la generación en la que crecimos y nos desarrollamos, lo podemos ver simplemente en el hogar, con algo tan sencillo como la música de nuestros padres, la nuestra y la de nuestros hijos, cada generación tiene diferente foco, diferentes necesidades. Pues todas estas mentalidades también se ven reflejadas en las empresas.

La brecha generacional también va cambiando las necesidades tecnológicas y va requiriendo que la tecnología se adapte.

Ahora estamos en una sociedad móvil, una sociedad conectada a Internet para todo. Para las compras, para el clima, para la TV, para ver a los niños por streaming en la escuela. Con redes sociales para pedir un transporte, para pedir alimentos, para felicitar a la tía que no vemos desde quien sabe cuánto tiempo.

Era de esperarse que no tardara mucho en socializar el entorno empresarial. Se tuvieron sus primeros intentos con Yammer (servicio que en lo personal no todos usan) y pues empezaron a salir los primeros en establecer una nueva forma de comunicación (Slack, Spark).

Y es en este punto donde “nace” Teams, ¿y porque hago énfasis en “nace”? porque Teams llega con la idea de una nueva forma de comunicación, pero tomando como base toda la experiencia que se tiene de la familia de Communication Server, entonces Teams no empieza de cero, al contrario, Teams tiene toda la parte de comunicación en tiempo real cubierta.

Presencia, IM, Conferencias, Audio, Video, Movilidad, Multiplataforma, Integración con otros servicios (aun en roadmap).

Tiene mas de 10 años de experiencia en este ámbito, sumándole todas estas nuevas funcionalidades de crear teams, agregar aplicaciones, tabs, menciones, GIF's, Memes …. Si, aunque parezcan cosas sin importancia son el "toque" especial de la nueva generación …

Si bien se ha tratado de que los clientes de la familia CS sean amigables al usuario, Microsoft Teams puede ser visto como un entorno mas sociable, menos rígido, donde tenias que estar en completo control de que no dijeras una palabra de más y no pongas un punto donde no va.

Esta comunicación es mas informal, mas espontanea (OJO nunca he dicho que deja de ser una herramienta empresarial, y debe siempre de ser utilizada de manera respetuosa y responsable) una comunicación que los usuarios llevan en el día a día en otras plataformas.

Pasamos a las comunicaciones inteligentes, comunicaciones que no son solamente "unificadas" sino que son colaborativas, son incluyentes e innovadoras. Cuanto trabajo no nos simplifica el uso de bots, cuando trabajo nos simplifica el tener las herramientas en una sola interfaz y poder compartirla y colaborar con todo el equipo de trabajo. Hacer mención para que pongan foco en la conversación, adjuntar ahí mismo el documento y trabajar directamente sobre el, etc.

En conclusión, Skype morirá (en su momento) pero para dejar paso a una tecnología con Skype en su ADN. Así que, amigos lectores, por ahora pueden estar tranquilos de que su servicio de Skype aun es funcional, pero la recomendación es que empiecen a adoptar Microsoft Teams para algunos casos de uso particulares, y poco a poco vayan conociendo sus bondades.

Espero que esto les sea de ayuda y si desean saber mas de lo que viene para Skype y MS Teams, síganme en mis redes sociales y únanse al único User Group en Espanol de Skype & Teams.

RODOLFO CASTRO AGUILAR
MVP Office Server and Services
Twitter : @ucblogmx
facebook.com/groups/SkypeTeamsUG/
ucblogmx.com

17

Setup Google Analytics on modern SharePoint using SPFx Extensions

Having an analytics tool monitoring your intranet is essential to understand the engagement of the users and how they are interacting with the content that is published. Google Analytics is widely used in the internet, it provides a concise analysis with reports of everything that happens in your site as well as real time reports, mostly targeted for public sites it also works on SharePoint intranets. Google Analytics is added to the sites using a script provided by Google, while this is not a problem for classic SharePoint Sites, with the modern no script site things are a bit different.

In this article, I explain step by step how to build and deploy an Application Customizer using the SharePoint Framework Extensions to add Google Analytics to the modern SharePoint following the best practices recommended by Microsoft.

How to setup Google Analytics

First things first, before we get into the SharePoint bits let's set up google analytics and get everything you need.

1.– Sign in to your Analytics account.
2.– Select the Admin tab.
3.– Select an account from the menu in the ACCOUNT column, or CREATE NEW ACCOUNT if you don't have one already.
4.– Select a property from the menu in the PROPERTY column.
5.– Under PROPERTY, click Tracking Info -> Tracking Code.
6.– Save the Tracking Id.

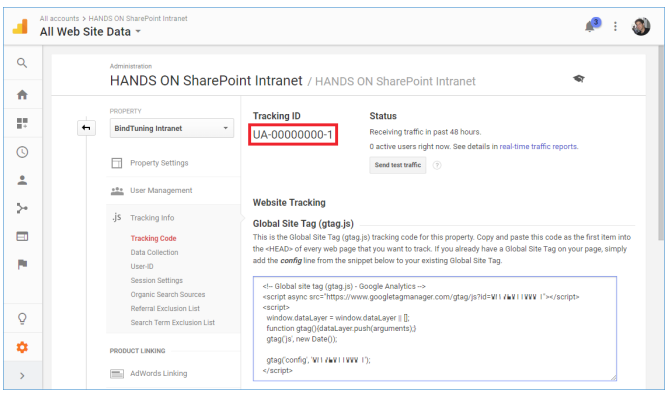


Image 1.-Tracking ID

Create the Application Customizer Extension project

The Application Customizer is one of the available extension types provided by the SharePoint Framework and allows you to add custom JavaScript to every page in a site or web, to achieve the steps below you need to have installed the SPFx v1.4 or higher.

1.– Create a folder with the name of the project e.g. analytics.
2.– Open the console window in the new directory.
3.– Type the command yo @microsoft/sharepoint.
4.– When prompted:
- Accept the default app-extension as your solution name, and press Enter.
- Choose SharePoint Online only (latest), and press Enter.
- Choose Use the current folder, and press Enter.
- Choose Y to make the extension available to be added without activating any features.
- Choose Extension as the client-side component type to be created.
- Choose Application Customizer as the extension type to be created.
- Provide a name to the extension. e.g. analytics
- Provide a description to the extension. e.g. Google Analytics for SharePoint modern pages

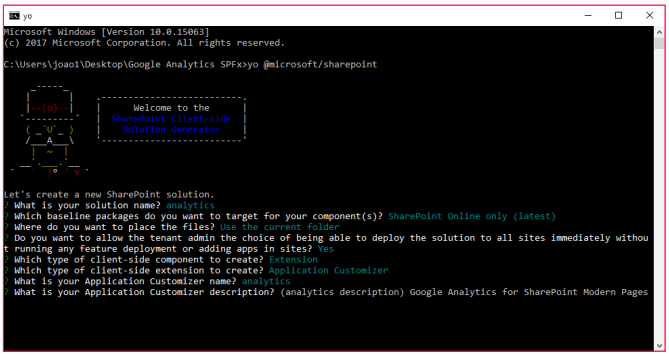


Image 2 .- SPFx Solution Setup.

> *"The Application Customizer is one of the available extension types provided by the SharePoint Framework"*

The download of all the requirements might take a few minutes, once it's done you will see a message indicating the success of the operation.

Build the extension

Now that you have the extension project created let's proceed and modify it to include the Google Analytics JavaScript code.

1.– Type code . to open the project (this will open visual studio code but you can use another editor).
2.– On your solution go to src/extensions/analytics and open the AnalyticsApplicationCustomizer.ts
3.– Locate the interface it IAnalyticsApplicationCustomizerProperties and replace it by the code below, this property will be used to store your tracking ID.

```
export interface IAnalyticsApplicationCustomizerProperties {
 trackingID: string;
}
```

4.– Locate the OnInit method and replace it by the code below

```
public onInit(): Promise<void> {
        let trackingID: string = this.properties.trackingID;
        if (!trackingID) {
                Log.info(LOG_SOURCE, "Tracking ID not
provided");
        }else{
                var gtagScript = document.createEle-
ment("script");
                gtagScript.type = "text/javascript";
                gtagScript.src = `https://www.googletag-
manager.com/gtag/js?id=${trackingID}`;
                gtagScript.async = true;
                document.head.appendChild(gtagScript);

                eval(`
                        window.dataLayer = window.
dataLayer || [];
                        function gtag(){dataLayer.
push(arguments);}
                        gtag('js', new Date());
                        gtag('config', '${trackingID}');
                `);
        }
        return Promise.resolve();
}
```

This snippet is a modified version of the code provided by Google, it was converted from JavaScript to TypeScript to use the Tracking Id dynamically.

> *"SharePoint Framework extensions must be specifically associated to sites, lists, and fields programmatically"*

Package the analytics solution

To deploy the analytics solution to all the users it needs to be packaged and installed on SharePoint Online. The instructions below are specific for SPFx 1.4 or higher and will make use of the Asset Packaging functionality.

1.– On your project go to the config folder, open the package-solution.json and confirm if the property includeClientSideAssets exists in the solution, if it doesn't exist it means that you are not using SPFx 1.4.

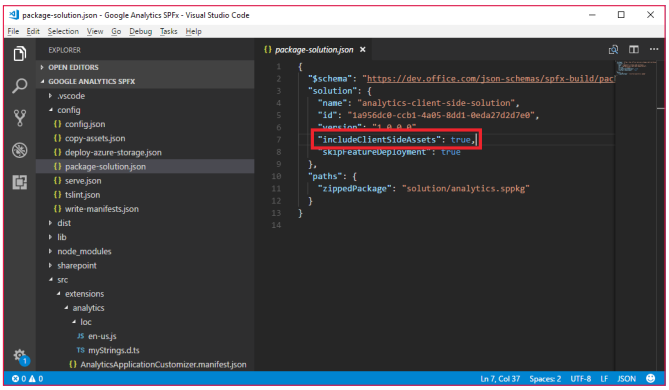


Image 3.- Package Solution.

2.– To get the basic structure for the packaging run the command gulp bundle –ship.
3.– To get the installation package run the command gulp package-solution –ship.
4.– On your project structure navigate to sharepoint/solution, in this folder you will find the *.sppkg installation file.

Install Google Analytics on Modern SharePoint

Extensions won't be automatically enabled. SharePoint Framework extensions must be specifically associated to sites, lists, and fields programmatically to be visible to site users.

To achieve the steps described in this section you will need to install the SharePoint PnP PowerShell.

1.– Open your App Catalog and upload the sppkg file
2.– Check the box Make this solution available to all sites in the organization

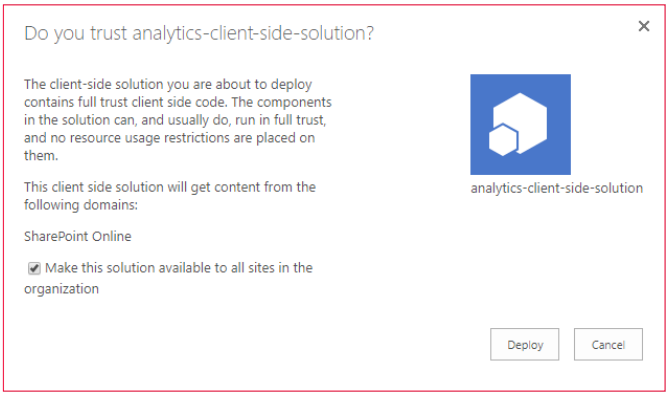


Image 4.- Install Solution.

3.– On your project go to src/extensions/analytics and open the AnalyticsApplicationCustomizer.manifest.json
4.– Copy the id value.
5.– Open the PowerShell command line.

6.– To establish a connection, execute the command Connect-PnPOnline -UseWebLogin -Url https://yourtenant.sharepoint.com/.

7.– To enable the extension, execute the command Add-PnPCustomAction -ClientSideComponentId “id” -Name “Analytics” -Title “Analytics” -Location ClientSideExtension.ApplicationCustomizer -ClientSideComponentProperties: ‘{“trackingID”:”UA-00000000-1”} -Scope site’ Replace the “id” by the client component id Replace the “UA-00000000-1” by your own google analytics tracking id.

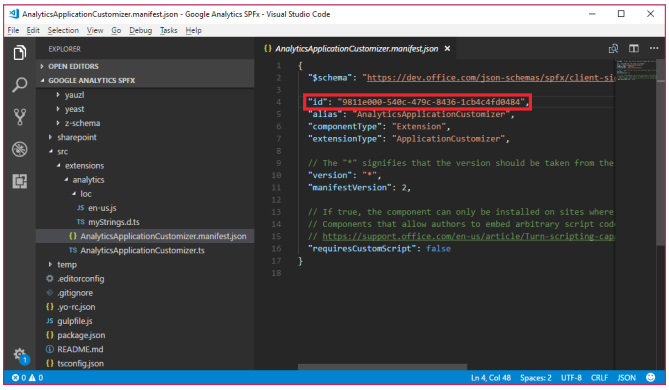


Image 5.- Solution ID.

“you learned how to add Google Analytics to the Modern SharePoint sites and how to create an application customizer step by step”

Conclusion

In this article, you learned how to add Google Analytics to the Modern SharePoint sites and how to create an application customizer step by step. If you were already using Google Analytics to monitor the SharePoint usage now you can combine your existent solution with this approach to get a complete overview of all sites and pages.

JOAO FERREIRA
SharePoint Developer

20

Introducción a Azure Durable Functions

A estas alturas todo el mundo conoce el concepto serverless y conoce que nos ofrecen las Azure Functions y también las mayores limitaciones que tienen:

- Timeout máximo de 10 minutos.
- Dificultad para orquestar diferentes funciones.

Para poder dar salida a estas dos limitaciones y abrir un nuevo mundo en el serverless, Microsoft ha sacado la versión preview de las Azure Durable Functions.

¿Qué es una Durable Function?

Las Durable Functions son una extensión de Azure Functions y por tanto una extensión de Azure WebJobs, que nos permite escribir workflows mediante código. Entonces, si podemos escribir workflows y no tenemos servidores, y estos escalan automáticamente y no tenemos control ninguno sobre estos ¿Cómo se coordinan los workflows?

Event Sourcing

Azure Durabel Functions implementa el patrón event sourcing (https://docs.microsoft.com/es-es/azure/architecture/patterns/event-sourcing), lo que permite saber y controlar el estado de la función en cada momento y poder interactuar con ella.

En este punto ya podemos inferir algunas de las ventajas que nos proporcionan las Azure Durable Functions:

- Worflows implementados con código sin necesidad de diseñadores o especificaciones JSON como por ejemplo Logic Apps.
- Al usar event sourcing nunca perdemos el estado de la función si por ejemplo el proceso se recicla o la VM donde se está ejecutando se reinicia o se cae. Siempre tenemos un punto de control y sabemos el estado exacto.
- Se pueden llamar a otras funciones de forma síncrona o asíncrona e interactuar con el resultado de estas.

Triggers, Flujos y Clases

¿Como desencadenamos la ejecución de una Durable Function? Pues sí, has acertado mediante triggers como funciones normales. En este caso hay una diferencia, que la orquestación de la función no se lanza directamente, sino que se requiere que se lance mediante una función superior, en el siguiente esquema se ve esta jerarquía:

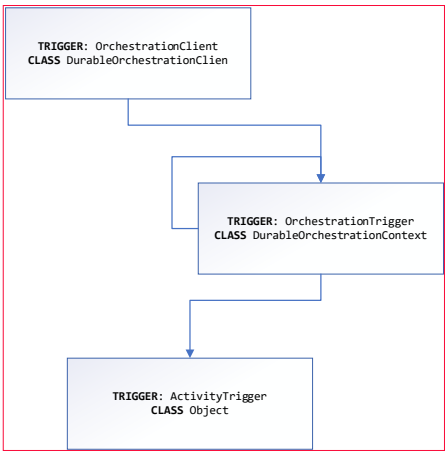


Imagen 1.- Esquema flujo de Duration Functions.

Vamos a explicar el esquema y como funciona mediante un ejemplo. El ejemplo consiste en insertar en forma masiva datos en Dynamics 365. El problema que nos encontramos en este caso es que solo se puede ejecutar 2 ExecuteMultipleRequest simultáneamente y 1.000 elementos en cada una de ellas, estos valores son limitaciones por defecto.

En nuestro caso y después de pruebas, para el buen rendimiento y evitar errores en un tenant trial insertamos 50 elementos a la vez.

Pasos:

1.– Creamos 1.000 ficheros con los 50 elementos que queremos insertar.
2.– Los subimos a un blob storage.
3.– Lanzamos la ejecución de la Durable Functions de forma que haya dos orquestaciones a la vez ejecutándose.
4.– La orquestación hace lo siguiente:
   - Comprueba que el fichero exista.
   - Si existe lo descarga e inserta en Dynamics
   - Añada un elemento, con un 1, en la cola para que se relanza la Durable Function con una nueva ejecución
5.– Si el fichero no existe se finaliza la ejecución.

De esta forma siempre tenemos dos instancias ejecutando el workflow y aprovechando al máximo la potencia tanto de las funciones como los límites Dynamics 365.

Empezamos explicando el lanzador:

```
[FunctionName("CounterStart")]
 public static async Task Run(
 [QueueTrigger("operation2")] string instanceId,
 [OrchestrationClient] DurableOrchestrationClient client, Trace-
Writer log)
{
  var count = Convert.ToInt32(instanceId);
  if (count <= 2)
  {
    for (int i = 1; i <= count; i++)
    {
      log.Info($"Client Operation 2 : Start Counter {i}");
      await client.StartNewAsync("Counter", i);
    }
  }
  else
  {
    log.Info($"Client Operation 2 : Start Counter {count}");
    await client.StartNewAsync("Counter", count);
  }
 }
```

Como podemos observar esta función está esperando una entrada en una cola llamada Operation2, y según el valor que nos llegue se lanzarán tantas funciones orquestadoras como indique el valor. En este caso el primer elemento deberá ser un 2 para lanzar dos orquestaciones y después 1 para ir lanzando nuevas orquestaciones.

En el código anterior podemos ver el trigger OrchestrationClient y la clase vinculada DurableOrchestrationClient. Esta clase os permitirá realizar lo siguiente:

- Ejecutar una nueva orquestación: StartNewAsync.
- Consultar el estado: GetStatusAsync.
- Enviar eventos a las orquestaciones para realizar acciones: RaiseEventAsync.
- Finalizar orquestaciones: TerminateAsync.

El código de la orquestación es el siguiente:

```
[FunctionName("Counter")]
public static async Task Run(
  [OrchestrationTrigger] DurableOrchestrationContext counter-
Context,
  TraceWriter log)
{
  int numberOfExecutions = 0;
  try
  {
    numberOfExecutions = counterContext.GetInput<int>();
    log.Info($"********{counterContext.InstanceId}: Cur-
rent counter state is {numberOfExecutions}. isReplaying:
{counterContext.IsReplaying} Waiting for next opera-
tion.**************");

    log.Info($"*********{counterContext.InstanceId}: Call activ-
ity ExistFile from {numberOfExecutions}*************");
    var existsFile = await counterContext.CallActivityAsync<-
bool>("ExistFile", numberOfExecutions.ToString());

    if (existsFile)
    {
      log.Info($"*********{counterContext.InstanceId}: EXISTS
FILE {numberOfExecutions}.json *************");

      log.Info($"*********{counterContext.InstanceId}: Call
activity AddCRM from {numberOfExecutions}*************");
      await counterContext.CallActivityAsync("AddCRM", num-
berOfExecutions.ToString());

      log.Info($"*********Add element to queue
*************");
    }
    else
    {
      log.Info($"*********{counterContext.InstanceId}: NO
EXIST FILE {numberOfExecutions}.json *************");
    }

    log.Info($"*********Return {counterContext.InstanceId}:
FINISH from {numberOfExecutions}*************");
  }
  catch (Exception ex)
  {
    log.Error($"**********ERROR General execution: {number-
OfExecutions} -  {counterContext.IsReplaying} - {counterCon-
text.InstanceId} *********", ex.InnerException != null ? ex.Inner-
Exception : ex);
    if (!counterContext.IsReplaying)
    {
      log.Info($"**********RETRY execution: {numberOfExecu-
tions} - {counterContext.InstanceId} *********");
      counterContext.ContinueAsNew(numberOfExecutions);
    }
  }
}
```

Esta es la función orquestadora, donde tenemos OrchestrationTrigger y la clase para este trigger DurableOrchestrationContext. En este método podemos ver como se ejecutan varias functions de forma encadenada. Esta clase contiene todos los métodos necesarios para poder orquestar llamadas a funciones como:

- Ejecutar una función: CallActivityAsync, CallActivityWithRetryAsync.
- Ejecutar otras funciones orquestadoras: CallSubOrchestratorAsync, CallSubOrchestratorWithRetryAsync.
- Continuar con una nueva ejecución de la orquestación manteniendo la instancia actual: ContinueAsNew.
- Espera eventos externos: WaitForExternalEvent.

En el ejemplo podemos ver que lo único que hacemos, como hemos dicho, encadenar diferentes llamadas a funciones una detrás de otra cuando acaban su ejecución mediante el método CallActivityAsync, que tiene dos parámetros: el nombre de la función a ejecutar, y el valor a pasar.

Por ejemplo, una de las que estamos ejecutando es la de añadir un elemento en la cola cuando se ha acabado correctamente todas las demás ejecuciones:

```
[FunctionName("AddQueueTrigger")]
[return: Queue("operation2", Connection = "AzureWebJobsS-
torage")]
public static string AddQueueTrigger(
  [ActivityTrigger] string numberOfExecution,
  TraceWriter log)
{
  try
  {
    var nextNumber = Convert.ToInt32(numberOfExecution);
    nextNumber = nextNumber + 2;
    log.Info($"Add Element To queue: {nextNumber}");
    return nextNumber.ToString();
  }
  catch (Exception ex)
  {
    log.Error($"\n************* ERROR execute CRM {number-
OfExecution} ******************", ex);
    return "0";
  }
}
```

Aquí observamos el trigger AcitivityTrigger y en este caso

como parámetro pásanos un string. Esta función se ejecuta y una vez ejecutada la orquestación recoge el resultado y continua su proceso. Todo proceso se guarda en una Azure Table, que es donde se van guardando los diferentes estados para poder saber el estado.

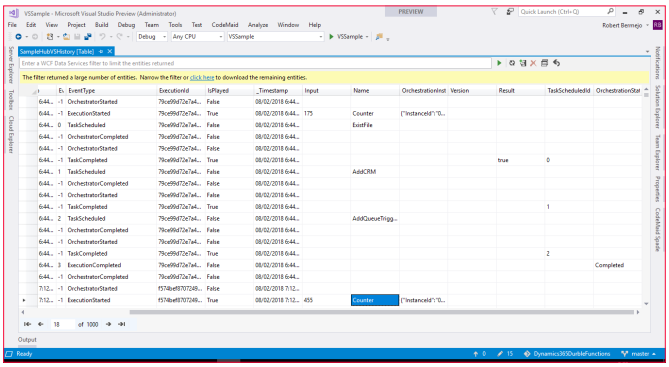


Imagen 2: Estados de una Durable Function.

En la imagen anterior, podemos ver el flujo de una ejecución de una orquestación y de los registros que se van dejando en esta tabla.

Patrones

Con todo lo explicado hasta el momento se pueden implementar cinco patrones:

1.– Encadenamiento de Funciones: Es el patrón que hemos implementado en el ejemplo,
se van encadenando funciones a medida que la anterior finaliza.

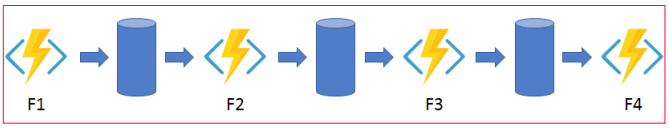


Imagen 3.- Encadenamiento de funciones.

2.– Fan-out/ Fan-in: Se lanzan funciones en paralelo y la orquestación espera que finalicen todas para realizar alguna acción con el resultado devuelto.

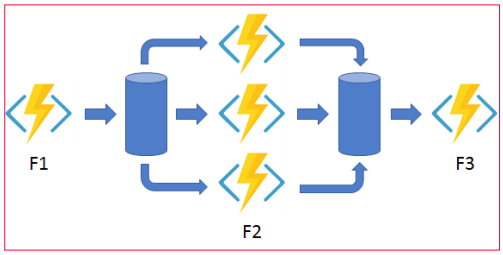


Imagen 4.- Fan-out/ Fan-in.

3.– Async Http APIs: Este patrón nos sirve cuando queremos coordinar el estado de las operaciones dese una aplicación externa.

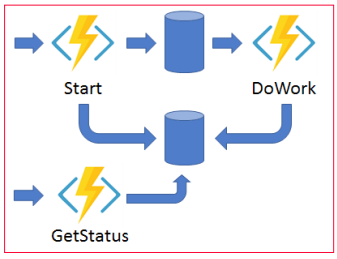


Imagen 5.- Async Http APIs.

4.– Singletons con estado: Cuando queremos ejecutar bucles infinitos.

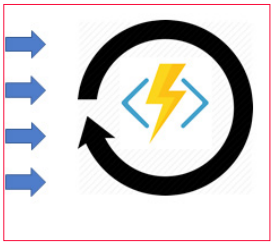


Imagen 6.- Singletons.

5.– Interacción humana: En ocasiones necesitamos una interacción de un externo, como por ejemplo un humano para poder continuar con el flujo de trabajo, un ejemplo sería cuando queremos hacer una doble autenticación en un registro donde esperamos que el usuario introduzca un código generado para acabar el registro.

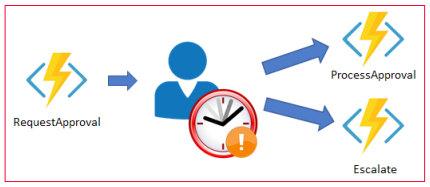


Imagen 7.- Interacción humana.

Restricciones

Pero no todo va a ser tan bonito, hay algunas consideraciones o restricciones que hay que tener en cuenta:

- El código debe ser determinista, ya que se reproducirá varias veces y siempre ha de devolver el mismo resultado.
- En el orquestador no se deben producir bloqueos, es decir, no se deben hacer llamadas de E/S, Thread.Sleep o equivalentes.
- En el orquestador se deben evitar bucles infinitos ya que podemos quedarnos sin memoria

Pequeñas consideraciones que se deben tener en cuenta para poder utilizar esta gran nueva versión de Azure Functions.

El código completo de los ejemplos los podéis ver en:

https://github.com/bermejoblasco/Dynamics365DurbleFunctions

Referencias

https://docs.microsoft.com/es-es/azure/azure-functions/durable-functions-overview

ROBERT BERMEJO
Team Leader en ENCAMINA
Microsoft Azure MVP
bermejoblasco@live.com
@robertbemejo
www.robertbermejo.com

23

GDPR Microsoft Office 365: Introducción

El nuevo Reglamento Europeo de Protección de Datos (RGPD o más conocido por sus siglas en inglés GDPR) entró en vigor en mayo de 2016 y es aplicable a partir de mayo de 2018, es decir se ha incluido un período de transición de dos años.

Imagen 1.- Fecha aplicable GDPR

En este período transitorio, los responsables y encargados del tratamiento de datos deben ir preparando y adoptando las medidas necesarias para cumplir con el reglamento en el momento en que sea de aplicación.

GDPR tiene como objetivo primordial, unificar los estándares de la Unión Europea en protección de datos y lograr que los ciudadanos europeos tengan mayor control sobre sus datos personales, entendiéndose estos como toda información relativa a una persona física identificada o identificable cuya identidad pueda determinarse directa o indirectamente.

Para más información:

http://www.agpd.es/portalwebAGPD/temas/reglamento/index-ides-idphp.php

"GDPR tiene como objetivo primordial unificar los estándares de la Unión Europea en protección de datos"

Microsoft Office 365

Muchos clientes se preguntan cómo en Office 365 pueden cumplir con el reglamento GDPR y cuáles son las medidas que tienen que tomar.

Office 365 es un buen sitio para cumplir con el reglamento, además de cumplir tanto con la mayoría de los estándares internacionales como los específicos de la industria como ENISA IAF, ISO/IEC 27001, 27018, FedRAMP, SOC 1 y SOC 2, HIPAA, sin olvidar el Esquema Nacional de Seguridad / ENS o la AEPD.

Hay que tener en cuenta cuales son las responsabilidades en el ámbito GDPR, para ello en el contrato de términos de los servicios online del 1 de febrero de 2018 ya se incluye el Anexo 4: Términos de conformidad con el Reglamento General de Protección de Datos de la Unión Europea.

Se puede destacar el segundo apartado dentro de roles y ámbito de aplicación.

"A los efectos de los presentes Términos RGPD, el Cliente y Microsoft convienen en que el Cliente es el responsable del tratamiento de los Datos Personales del Cliente y que Microsoft es el encargado del tratamiento de dichos datos, con la salvedad de que, si el Cliente está actuando en calidad de encargado del tratamiento, entonces Microsoft es un sub encargado del tratamiento"

Para más información http://www.microsoftvolumelicensing.com/DocumentSearch.aspx?Mode=3&DocumentTypeId=31

DAPI (Descubrir, Administrar, Proteger e Informar)

La preparación para cumplir GDPR puede ser compleja y es necesario sentar unas bases claras para poder adjuntarnos al nuevo reglamento, Microsoft recomienda centrarse en cuatros pasos claves conocido como DAPI (Descubrir, Administrar, Proteger e Informar).

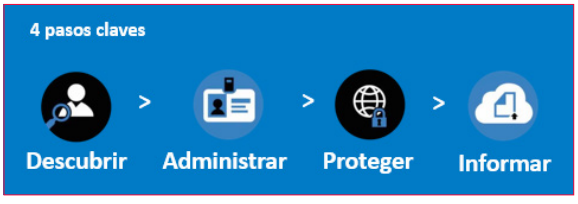


Imagen 2.- DAPI.

Descubrir

El primer paso es analizar los datos personales que poseemos y el lugar donde se encuentran.

La definición de "datos personales" es bastante amplia

conforme GDPR, ya que estos corresponden a cualquier dato que tenga relación con una persona natural identificada o que sirva para identificar a una persona natural.

Es importante realizar un inventario de los datos, esto ayuda a entender cuales son personales e identifica los sistemas donde se almacenan, procesan, comparten y el tiempo de conservación de los mismos.

> *"GDPR establece nuevos estándares en materia de transparencia, responsabilidad y mantenimiento de registros"*

Administrar

Para cumplir GDPR es necesario gestionar y controlar los datos personales, por ese motivo es necesario tener un plan de gobernanza de datos ya que nos ayuda a definir directivas, roles y responsabilidades para el acceso, la administración y el uso de datos personales, además de ayudar a asegurarse que su práctica cumple con el reglamento.

Proteger

GDPR eleva el nivel de exigencia en la seguridad de la información, requiere que las organizaciones adopten las medidas adecuadas para proteger los datos personales contra la pérdida, el acceso o la divulgación no autorizada.

Es necesario disponer de medidas de seguridad para proteger los datos personales.

Informar

GDPR establece nuevos estándares en materia de transparencia, responsabilidad y mantenimiento de registros, siendo necesario mantener registro sobre las categorias de datos personales, identidades de terceros con los que se comparten los datos, si hay paises extranjeros que reciben los datos personales, las medidas de seguridad organizacionales/técnicas y los tiempos de conservación de los datos.

Una forma de lograr esto es mediante el uso de herramientas de auditoría, lo que puede ayudar a asegurar que cualquier procesamiento de datos personales se rastrea y registra.

Herramienta de evaluación del GDPR de Microsoft

Microsoft proporciona una herramienta para ayudar a evaluar la madurez del cumplimiento en nuestra organización, esta disponible en dos modalidades:

- Versión Online

  La podemos encontrar en https://www.gdprbenchmark.com/es/questions

  Actualmente, se realizan 26 preguntas agrupadas por DAPI, como se puede aprececiar en la siguiente imagen:

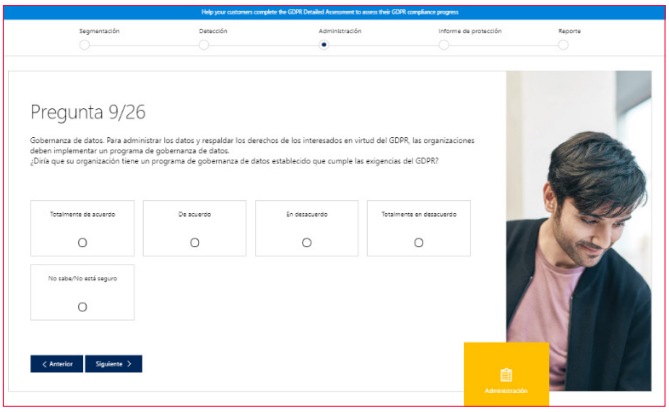


Imagen 3.- Herramienta de evaluación GDPR online.

  Al finalizar el cuestionario, se presenta como resultado el nivel de madurez que tenemos en el cumplimiento del reglamento y algunas recomendaciones para su mejora, como se puede apreciar en la siguiente imagen:

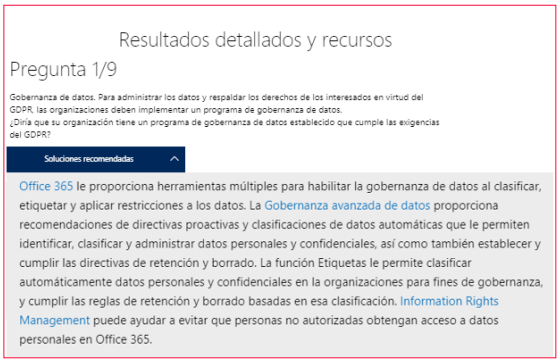


Imagen 4.- Herramienta de evaluación GDPR online – resultado.

- Versión local en Excel

  La podemos descargar de https://assets.microsoft.com/en-us/gdpr-detailed-assessment.zip. Actualmente, se realizan 162 preguntas agrupadas por DAPI, como se pude apreciar en la siguiente imagen:

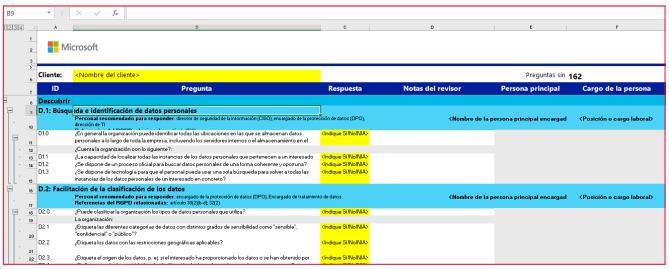


Imagen 5.- Herramienta de evaluación GDPR local.

También presenta en el resultado el nivel de madurez y recomendaciones, como se puede apreciar en la siguiente imagen.

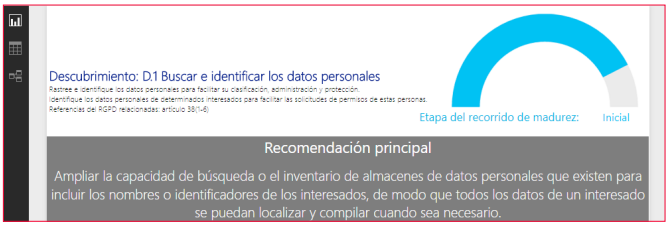


Imagen 6.- Herramienta de evaluación GDPR online - resultado

Nota: para ver el resultado es necesario tener instalado Power BI Desktop en el equipo.

> *"Mediante PowerShell podemos realizar muchas acciones DAPI, pero lo importe es que podemos realizar acciones para identificar, ratificar, eliminar, informar y exportar datos personales"*

Herramientas para ayudar a cumplir GDPR en Office 365

Microsoft ofrece una serie de herramientas para ayudar a cumplir con el reglamento GDPR en Office 365, a continuación se enumeran algunas de ellas:

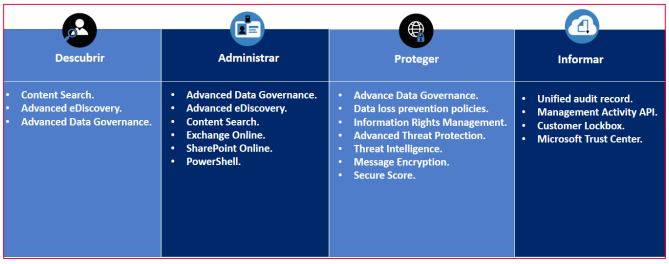


Imagen 7.- Herramienta ayuda de GDPR Office 365.

- Content Search.

  Herramienta de búsqueda de contenidos en todas las carpetas públicas de Exchange y SharePoint Online y OneDrive de la empresa, nos ayuda a encontrar e identificar los datos personales que pueden ser relevantes para el cumplimiento.

- Advanced eDiscovery.

  Permite identificar los datos relevantes con mayor agilidad y precisión que las búsquedas tradicionales por palabra clave y encuentra archivos prácticamente idénticos, reconstruye hilos de correo e identifica temas clave y relaciones entre datos, ademas de propocionar a los interesados una copia exportable de sus datos personales en caso que sea solicitado.

- Advanced Data Governance.

  Proporciona recomendaciones de políticas proactivas y clasificaciones de datos automáticas que ayudan a identificar, clasificar y administrar datos e información confidencial, así como a aplicar políticas de retención y supresión. La función de etiquetas permite clasificar automáticamente los datos personales y la información confidencial de toda la organización, así como aplicar reglas de retención y supresión basadas en dicha clasificación.

- Exchange Online.

  Utiliza reglas de flujo de correo de Exchange Online para enrutar el correo con determinadas palabras clave, como permisos o supresión, a buzones específicos. Esto permite crear un proceso personalizado para recibir, administrar y responder a estas solicitudes.

- SharePoint Online.

  Podemos utilizarlo para efectuar un seguimiento y administración de los manuales de las solicitudes de permisos de las personas a las que se refieren los datos personales.

- PowerShell.

  Mediante PowerShell podemos realizar muchas acciones DAPI, pero lo importe es que podemos realizar acciones para identificar, ratificar, eliminar, informar y exportar datos personales de forma totalmente personalizada.

- Data loss prevention.

  Permiten aplicar automáticamente restricciones de acceso a los datos y restringir su uso compartido.

- Information Rights Management.

  Ayuda a evitar el acceso a la información personal en Office 365 por parte de personas no autorizadas.

- Advanced Threat Protection .

  Protege correo electrónico, archivos y las aplicaciones de Office 365 contra ataques desconocidos y sofisticados.

- Threat Intelligence.

  Ayuda a detectar de forma proactiva y a protegerse contra amenazas avanzadas en Office 365.

- Message Encryption.

  El cifrado de mensajes de Office 365 complementa la Azure Information Protection, y hace que sea más fácil compartir correos electrónicos protegidos con cualquier persona, tanto de dentro como de fuera de tu organización.

- Secure Score.

  Analiza la seguridad de la organización en Office 365 en función de actividades normales y la configuración de seguridad y asigna una puntuación.

- Unified audit record.

  Permite hacer un seguimiento y registrar las actividades en nuestro entorno de Office 365, incluyendo actividades de usuario y de administrador en Exchange Online, SharePoint Online y OneDrive para la Empresa. Se puede utilizar el registro de auditoría unificado para registrar la resolución de las solicitudes de permisos de las personas a las que se refieren los datos y registrar eventos asociados con la modificación, supresión o transferencia de datos personales.

- Management Activity API.

  Servicios de informes que permiten obtener transacciones agregadas, pudiendo ser consumida por otras herramientas.

- Customer Lockbox.

Permite controlar si se permite el acceso a nuestros datos office 365 por parte del ingeniero de soporte técnico Microsoft, dicho permiso puede ser temporal con una fecha de expiración, para que solo tenga vigencia en el proceso de evaluación por parte del soporte técnico.

- Microsoft Trust Center.

  Es el centro de Microsoft donde se encuentra información detallada sobre seguridad, privacidad y ofertas de cumplimiento, políticas, características y prácticas de los productos de Microsoft en la nube.

"es obligatorio cumplir con el Reglamento Europeo de Protección de Datos GDPR. Office 365 es un buen sitio para cumplir con el reglamento"

Conclusiones

Ya es obligatorio cumplir con el Reglamento Europeo de Protección de Datos GDPR. Office 365 es un buen sitio para cumplir con el reglamento, además de cumplir con la mayoría de los estándares internacionales como específicos de la industria. Microsoft demuestra que tiene un gran compromiso de informar y ayudar a sus clientes para cumplir con el reglamento de la Unión Europea, como se puede apreciar en

https://www.microsoft.com/en-us/TrustCenter/Privacy/gdpr, además de recomendar los pasos claves (DAPI) que nos tenemos que centrar.

Office 365 dispone de un gran número de herramientas que nos facilitan conseguir el cumplimiento del reglamento: Content Search, Advanced eDiscovery, Advanced Data Governance, Data loss prevention policies, Advanced Threat Protection, Threat Intelligence, Message Encryption, Secure Score, etc.

MARTIN LUIS LOPEZ REQUENA
SharePoint Solution Architect & Trainer at everis
martinluislopez@hotmail.com

27

Aspectos que me hubiera gustado saber antes de abordar un proyecto de ReactJS - Parte I

Los tiempos en el desarrollo sobre Office 365 han cambiado, del desarrollo tradicional centrado en tecnología en servidor hemos pasado a un desarrollo en el que la gran mayoría de nuestros esfuerzos se centran en el FronEnd. Como se ha comentado en anteriores artículos publicados, el tooling de herramientas ha cambiado totalmente: Visual Studio Code, Typescript, Gulp, WebPack, NodeJS son algunas de las que ahora no deben de faltar para los "nuevos" Office Developers. Estas herramientas están claras, donde vienen las dudas principalmente es en la elección del framework JavaScript que se va a utilizar. Naturalmente el equipo de producto comenta que se puede utilizar cualquiera, sin embargo, vemos que hacen un especial hincapié en ReactJS (siendo un producto de Facebook). ¿Que tiene ReactJS para que sea nuestra elección? ¿Cuáles son sus principales características? ¿Qué cosas debemos de conocer de React para poder adoptarlo y que sea un framework en el que nos sintamos cómodos con él? A lo largo de una serie de artículos voy a intentar explicar todo lo que hubiera necesitado cuando hace un año empezamos a adoptarlo como Framework para nuestros desarrollos tanto dentro de SharePoint como fuera ellos.

"ReactJS NO es un framework JavaScript como tal, es una librería que se encarga solamente del renderizado/visualización de nuestro componente"

¿Qué es ReactJS?

Aunque parezca una pregunta obvia no lo es tanto, ReactJS NO es un framework JavaScript como tal, es una librería que se encarga solamente del renderizado/visualización de nuestro componente. Estamos acostumbrados al Framework JavaScript que se encarga de todo, Route, Services, MVC …como pueda ser Angular, esto hace que tengamos un gran acoplamiento/dependencia con la evolución de nuestro desarrollo. Al decir esto no quiero decir que ReactJS no tengan otras librerías auxiliares que se encarguen de esto, React-Route, React-Dom, React-Redux, simplemente que las elecciones de estas librerías ya son decisión del desarrollador y no impuestas por el Framework.

Vamos a centrarnos en su principal virtud que es el renderizado, uno de los grandes problemas que ha ocurrido principalmente cuando desarrollamos en el Front, es que la manipulación de los elementos del DOM es lento y costoso por lo tanto si estamos continuamente modificando sus valores eso hace que la interfaz de usuario quede un poco fea, mostrando la carga de los elementos en varios pasos, ocultando diversas pantallas antes. En definitiva, el acceso al DOM continuo tiene problemas de rendimiento y además de diseño. ¿Cómo soluciona ReactJS este problema? Para solucionarlo ellos han creado un DOM Virtual que es el que nosotros modificamos directamente en nuestros desarrollos, ahora bien, el éxito no es solamente este DOM virtual sino la implementación de un algoritmo que se encarga de comparar el DOM virtual con el DOM real y solamente modifica los nodos que se han modificado desde su último renderizado. Esto ha sido a nivel de eficiencia el principal motivo por el que ha crecido ReactJS.

¿Qué trae de nuevo ReactJS respecto a otras librerías/Frameworks?

Junto con este algoritmo de renderizado del DOM virtual, ReactJS implementa en cierta forma la especificación de Web Components. El estándar, en realidad, se compone de 4 subelementos complementarios, pero independientes entre sí: Custom Elements (elementos personalizados), Templates (plantillas), Shadow DOM (DOM oculto) y HTML Imports. Este estándar lo que trata es de poner un poco de orden a todo el desarrollo que se realiza en el Front y que no sea un mundo totalmente diferente al resto de desarrollos.

Este estándar lo implementan todos los nuevos Frameworks JavaScript que se han desarrollado en los últimos años. Sin embargo, la particularidad de ReactJS es que va unido a una nuevo paradigma de arquitectura/patrón para implementar aplicaciones grandes. Este nuevo paradigma es lo que se llaman arquitecturas Flux (no confundir con Redux del cual hablaremos en el siguiente artículo) estas arquitecturas se basan en que la aplicación sigue un flujo lógico de ejecución y cualquiera persona es capaza de seguirlo (siempre que se implemente bien). Para ver un ejemplo de este tipo de arquitectura podemos observar como Microsoft implementó Delve basándose en este patrón https://medium.com/@delveeng/how-we-use-the-flux-architecture-in-delve-effc551f8fbc

Para que tengamos claro que es Flux vamos a poner un ejemplo simple, ahora mismo en una aplicación MVC tene-

mos lo que se llama el “two binding” es decir algo similar a la siguiente imagen:

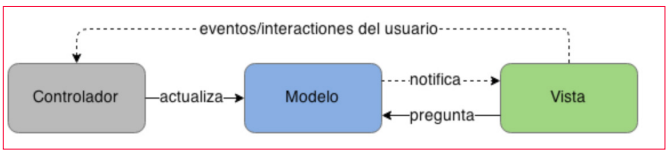


Imagen 1.- Modelo de Two Binding.

Conforme vamos añadiendo más controladores a la aplicación nuestro modelo se complica mucho más:

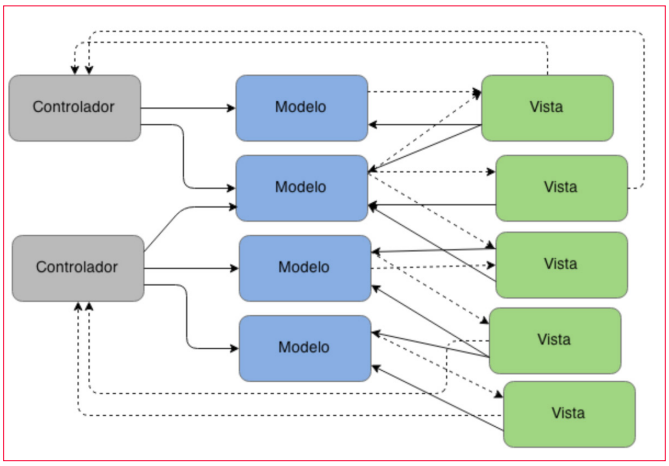


Imagen 2.- Modelo con varios controladores.

Con una arquitectura Flux la secuencia de la aplicación se simplifica mucho más quedando de la siguiente forma:

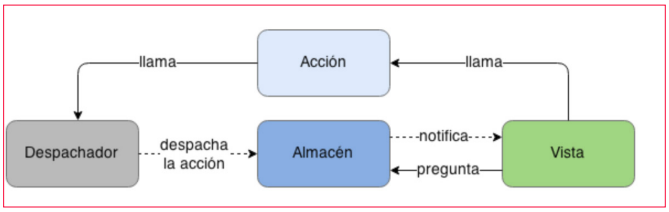


Imagen 3.- Secuencia de la aplicación con una arquitectura Flux.

Y aunque incrementemos el número de vistas y de “Dispatch/Controladores” la aplicación no se complica más de la cuenta:

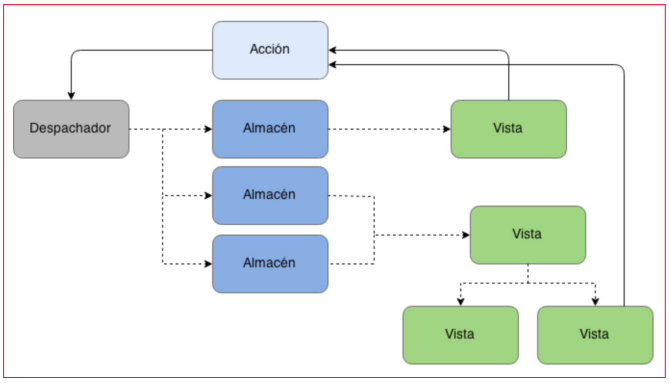


Imagen 4.- Aplicación con un mayor número de Dispatch / Controladores.

La arquitectura flux tiene propiedades que la hacen única y provee importantes garantías, todas giran alrededor de un flujo de datos explícito y fácil de entender, aumentando la capacidad de seguir, reproducir y realizar pruebas en estados de aplicación específicos. Sus principales características:

- Síncronas: El despachador de acciones y las funciones dentro de los almacenes son síncronos. Todas las operaciones asincrónicas deben invocar una acción le comunica al sistema el resultado de la operación. Los creadores de acciones pueden llamar APIs asincrónicamente, los almacenes idealmente no lo deben hacer. Esta regla hace que el flujo de información sea extremadamente explícito y en caso de errores fácilmente se puede identificar la acción ese estado erróneo de la aplicación.
- Inversión del Control: Los almacenes se auto actualizan en respuesta a acciones en lugar de ser actualizados por un controlador o modulo similar, ningún otro componente de la aplicación contiene lógica sobre como actualizar el estado. Como los almacenes se auto actualizan en respuesta a acciones y únicamente sincrónicamente, realizar pruebas es tan sencillo como inicializar con un estado específico, invocar una acción y verificar que el estado final es el esperado.
- Cero acciones en cascadas: Flux no permite despachar una segunda acción como resultado de una primera acción, esto ayuda a prevenir actualizaciones en cascada que son difíciles de mantener y debuggear. Además, ayuda a pensar en las interacciones de la aplicación en una forma más semántica.

El ciclo de vida de un componente

Para empezar a crear un componente tiene dos parámetros que se le pasan al componente, el primer parámetro son las Props (propiedades) y el segundo parámetro es el State o estado del componente.

```
export class Pagination extends React.Component<IPaginationProps, IPaginationState> {
```

Esta es una de las primeras dudas que nos planteamos a la hora de empezar el propio desarrollo cuando utilizo las Props y cuando utilizo el State. Para saber cuándo hay que utilizar cada una de ellas hay que saber su definición. Por un lado, las Props son propiedades inmutables dentro de un componente es decir que su valor no se modifica en dicho componente. Mientras el State es el estado que tiene el propio componente en un determinado momento, este State se puede modificar en el propio componente.

“Junto con saber diferenciar bien el State y las Props es necesario entender bien el ciclo de vida de un componente”

Lo ideal a la hora que empezamos a desarrollar componentes es ir creando un componente principal en el que tenemos las propiedades de nuestro desarrollo y estas propiedades se van inyectado en los estados de los componentes hijos. Cuando incorporemos una librería como Redux veremos la importancia de saber muy bien estos dos parámetros.

Junto con saber diferenciar bien el State y las Props es necesario entender bien el ciclo de vida de un componente, para ello un componente React tiene un método "Render" que es el encargado de "pintar" el HTML del propio componente, pero junto con este método principal hay una serie de métodos que se lanzan antes de este render: ComponentWillMount, ComponenteDidMount, ComponentDidMount, ComponenteWillMount, ComponentReceiveProps. Estos métodos podríamos verlos como un símil al típico OnLoad de un WebPart clásico.

Dentro de estos métodos se hacen operaciones como pueda ser llamar a una API para cargar los datos de un componente. El flujo del componente es que una vez nos traemos los datos, se notifica esta modificación al componente y una vez estos datos se modifican el componente se vuelve a renderizar. Por eso, aunque la primera vez os suene a chino, dentro de estos métodos NO se debe de modificar el estado del propio componente, el motivo: puedes entrar en un bucle infinito.

"dentro de la clase que extiende nuestro WebPart, tenemos diversas propiedades como es el contexto"

Show me the code

Tras toda esta introducción sobre React vamos a dirigirnos y abrimos una solución de SPFX. Si nos damos cuenta en la clase en la que se renderiza el WebPart se renderiza un objeto de ReactJS, dentro de este componente se le pasan los elementos que se van a encargar de pintar nuestro componente. Como aspectos que son curiosos es que dentro de la clase que extiende nuestro WebPart, tenemos diversas propiedades como es el contexto donde encontramos todas propiedades de dicho componente. Si este Contexto lo queremos utilizar en nuestro Componente deberemos de inyectárselo como Propiedad a nuestro componente. De tal forma que quedaría de la siguiente forma:

```
export default class HelloWorldWebPart extends BaseClien-
tSideWebPart<IHelloWorldWebPartProps> {

 public render(): void {
  const element: React.ReactElement<IHelloWorldProps > =
React.createElement(
    HelloWorld,
    {
     description: this.properties.description,
     context: this.context
    }
  );

  ReactDom.render(element, this.domElement);
 }
```

Una vez ya tenemos el punto de arranque nos quedaría la forma de como planteamos la comunicación entre los diversos componentes que lo utiliza y como organizar nuestro código de tal forma que nuestros desarrollos no sean código spaghetti.

¿Pero, ahora bien, en SharePoint necesitamos utilizar solamente React o necesitamos de librerías auxiliares Flux, React Router, etc? Está es otra de las cuestiones que necesitamos saber que opción y no dejarnos llevar por las modas sino por las necesidades que tenemos y sobre todo el contexto en el que estamos que no hay que olvidarlo es dentro de un sitio de SharePoint Online. Todo esto lo abordaremos en el siguiente número.

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

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.

31 Entrevista CompartiMOSS

Hola a todos.

Mi nombre es CompartiMOSS, y no tengo apellido. Y para que vean como soy de original, tengo dos padres naturales en lugar de un padre y una madre, como todos ustedes.

Además, a pesar de mi juventud y de mi modestia reconocida mundialmente, tengo que decir que soy experta en todas las tecnologías de Microsoft, no en una o dos como, de nuevo, todos ustedes.

Tampoco soy Microsoft MVP pues nadie se ha molestado en contarle a Microsoft de mi labor de comunidad, mi experiencia con todos los productos de la compañía, y mi ampliamente apreciada capacidad para compartir mis vastos conocimientos sobre el tema. A pesar de eso, no le tengo resentimiento a Microsoft y, por el contrario, intento regularmente trabajar con la compañía para organizar conferencias, seminarios y otras actividades... casi siempre termino sola organizándolo todo...

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

Comencé hace diez años, cuando mis dos padres, Héctor Insua y Gustavo Velez, en una tarde de irresponsabilidad decidieron que era necesario crear un medio de información en español sobre SharePoint. Si, SharePoint, un servidor de Microsoft que hace diez años era una revolución en la informática, y hoy en día es prácticamente ignorado y está escondido como uno de los componentes de otro producto llamado Office 365 (aunque parece que Microsoft no lo quiere dejar morir del todo, y dizque quieren sacar una nueva versión, SharePoint 2019). Durante años y años nos divertimos muchísimo modificando a SharePoint a nuestro gusto y publicando información al respecto, hasta que a un señor desconocido, un tal Satya Nadella, se le metió la idea en la cabeza que todo había que ponerlo en la nube, en lugar de tener los pies metidos en el barro, como debería ser.

¿Cuáles son tus principales actividades tecnológicas hoy en día?

Después de que, gracias a Microsoft, SharePoint prácticamente desapareció del mercado, yo, CompartiMOSS, entré en una crisis de identidad y de lectores. El padre que todavía cuidaba de mí, Gustavo (mi otro padre me abandonó cuando estaba recién nacida), y mis nuevos padres adoptivos, Fabián, Juan Carlos y Alberto, decidieron que había dos posibilidades: dejarme morir de inanición, es decir, publicarme cada vez menos artículos, o ampliar mis horizontes técnicos y permitirme aprender nuevas capacidades. Afortunadamente estoy en buenas manos y la decisión fue dejar de ser una revista especializada en un servidor moribundo, y abrirme a todo lo que Microsoft ofrecía. Y así estoy ahora, rejuvenecida, no gracias a botox y estiramientos de piel, sino a las ideas refrescantes que vienen de mis autores y editores.

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

En realidad, no me queda mucho tiempo libre. Mi servidor de internet está demasiado ocupado renderizando miles y miles de páginas por día, y mi ancho de banda casi que no me alcanza más debido a la cantidad de descargas que tengo que satisfacer. Así que mi CPU está funcionando casi al 100% todo el tiempo, y no me queda mucha más capacidad de cálculo para hacer otras cosas. Tengo algunos amigos, por ejemplo, un servidor de CRM que esta virtualizado aquí a mi lado y, por supuesto, los servidores de parches de Microsoft que hacen una fiesta conmigo todos los segundos martes de cada mes, pero a eso se reduce toda mi vida social.

¿Cuáles son tus hobbies?

Como acabo de comentar: me he cansado de pedirle cons-

tantemente a mis administradores (sin que me hagan caso) que me aumenten mi CPU y RAM, y no me queda tiempo real para hobbies. Cada tres meses tengo que publicar un montón de artículos nuevos, procesar más imágenes, publicar nuevo código, renderizar páginas, servir pdfs. Tiempo para cosas tan banales como hobbies no me queda. Y tampoco para hacerme propaganda con Microsoft para que me nombren MVP.

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

A decir verdad, no tengo ni visión ni me interesa el futuro de la tecnología. Yo solo público lo que me dicen que publique. Como el homo-sapiens es un homo-técnicus, siempre habrá alguien que quiera decir algo al respecto y siempre tendré algo para contarle a mis lectores. Lo que me preocupa más es que mis editores consigan el suficiente dinero con que mantener mis servidores en el aire y pagarle a mi salón de belleza (perdón, quiero decir a mi diseñador gráfico). Solo me da curiosidad saber que van a hacer conmigo en los próximos diez años...

CompartiMOSS *(ayudada para escribir este artículo por Gustavo, Juan Carlos, Fabián y Alberto para evitar que después no me lo quieran publicar)*
Sitio: http://www.compartimoss.com
Correo: revista@compartimoss.com

33

Implementando PowerShell con Power BI

Hay varios requerimientos que surgen en un ambiente corporativo con la finalidad de tener mayor control y poder automatizar ciertas actividades de despliegue. Para comenzar, existen lo que se definen como módulos de PowerShell. Se debe tomar en cuenta que no es muy común hablar de automatizar tareas a través de PowerShell en lo referente a la tecnología Power BI pero veremos durante este artículo que es sumamente potente, y flexible. En primer lugar, como referencia, podemos partir por la siguiente descarga:

https://www.powershellgallery.com/packages/Microsoft.PowerBI.PowerShell/1.2

Veremos a continuación cuales son los componentes necesarios para comenzar y configurar la conexión e integración:

Requerimientos:

- Contar con un tenant de Azure Active Directory y un usuario corporativo.
- Registrar un App de Power BI.
- Registrar dicha App de Power BI en Azure.
- Aplicar permisos para la aplicación en Azure Active Directory.
- Instalar el Módulo de PowerShell para Power BI.

Registro de Power BI App:

- Ingresar dev.powerbi.com/apps.
- Ingresar con una cuenta existente de Power BI.
- Definir un Nombre de App en el cuadrante correspondiente.
- Para PowerShell selección App Nativa.
- Registrar un App de Power BI
- Registrar dicha App de Power BI en Azure.
- Aplicar permisos para la aplicación en Azure Active Directory.
- Instalar el Módulo de PowerShell para Power BI.
- Ingresar lo siguiente como URL de Redirección: https://login.microsoftonline.com/nombreorganizacion.onmicrosoft.com/oauth2
- Agregar todas las APIS que correspondan para nuestro caso. Se puede chequear un poco mas en detalle la información de permisos desde el siguiente link: https://powerbi.microsoft.com/en-us/documentation/powerbi-developer-power-bi-permissions/
- Dar click en el botón de Registrar App.
- Tomar nota del Client ID ya que nos será solicitado.

Paso a paso del Registro de Power BI App

A continuación, se detalla como registrar paso a paso la Power BI App:

- Registro de la App de Power BI

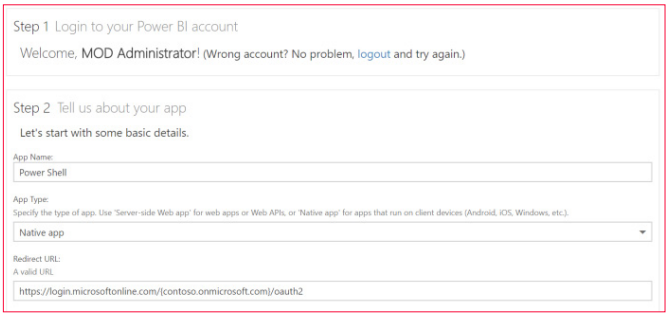


Imagen 1.- Datos iniciales de la App.

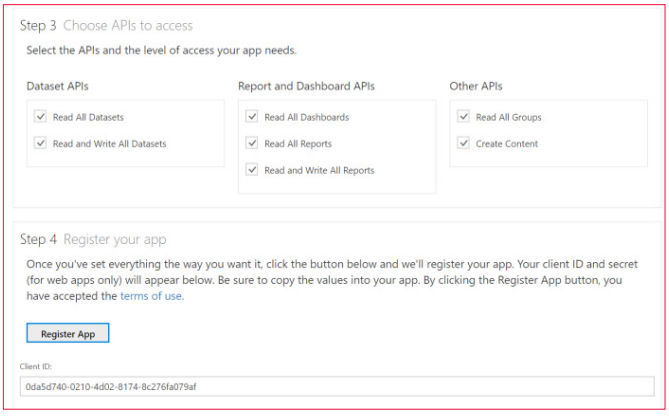


Imagen 2.- APIs a las que se va a acceder y Client ID.

- Registro de la App de Power BI en Azure:

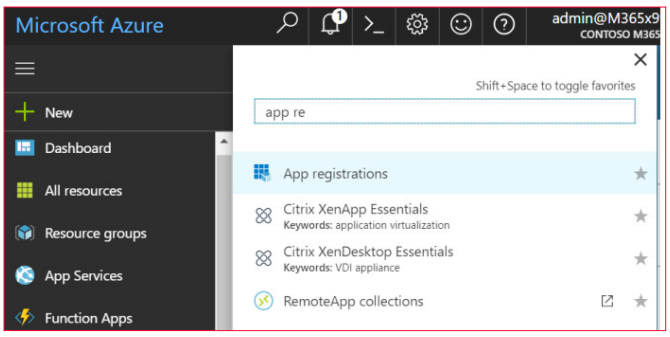


Imagen 3.- Registros de la App de Power BI en Azure.

- Seleccionar nuevo registro de aplicación.
- En el formulario siguiente que aparece agregar la siguiente información:
  - Agregar un nombre en el cuadro de dialogo.
  - Seleccionar Nativo en el Tipo de Aplicación, desde el cuadro desplegable.
  - Agregar como URL de redireccionamiento: https://login.microsoftonline.com/contoso.onmicrosoft.com/oauth2

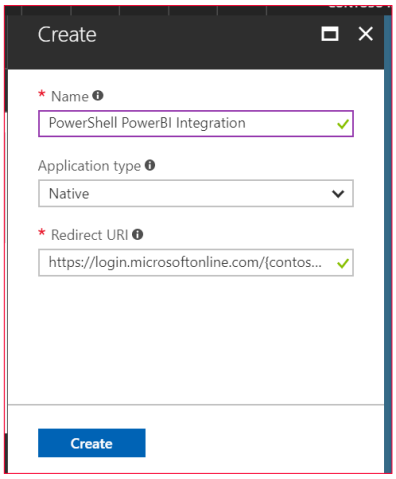


Imagen 4.- Información de registro de la App.

- Dar click en la opción de Crear.

Aplicar permisos para nuestra aplicación en Azure Active Directory

- En la sección de Registro de Aplicaciones seleccionar nuestra nueva App:

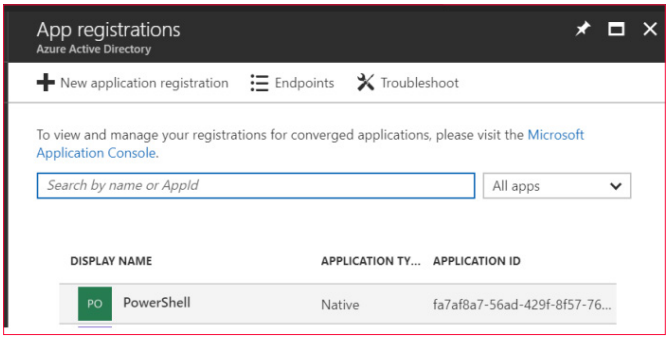


Imagen 5.- Selección de la App en Registro de Aplicaciones.

- Abrir la Ventana de Windows Azure Active Directory API, y seleccionar la opción de Acceso a Directorio con usuario logueado. Luego guardar:

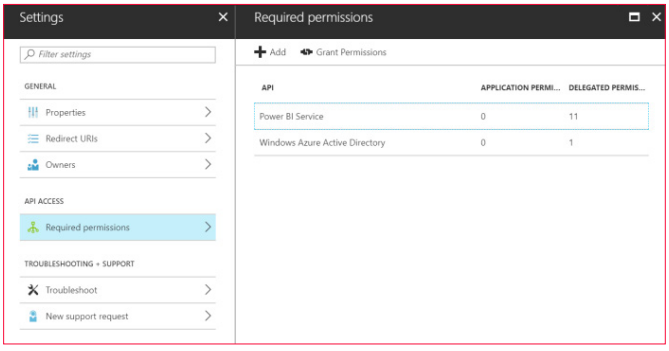


Imagen 6.- Configuración de permisos para la Aplicación.

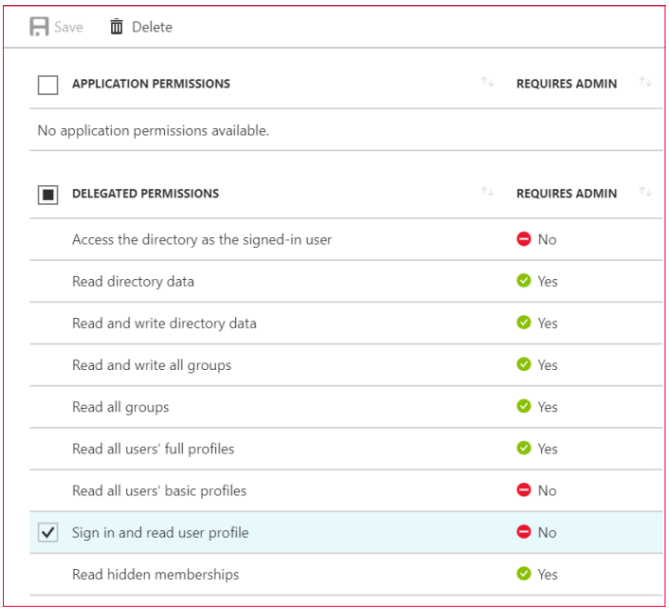


Imagen 7.- Permisos configurados para la App.

"no es muy común hablar de automatizar tareas a través de PowerShell en lo referente a la tecnología Power BI pero veremos que es sumamente potente, y flexible"

- Abrir la API del Servicio Power BI, seleccionar todos los permisos debajo de Permisos Delegados y guardar

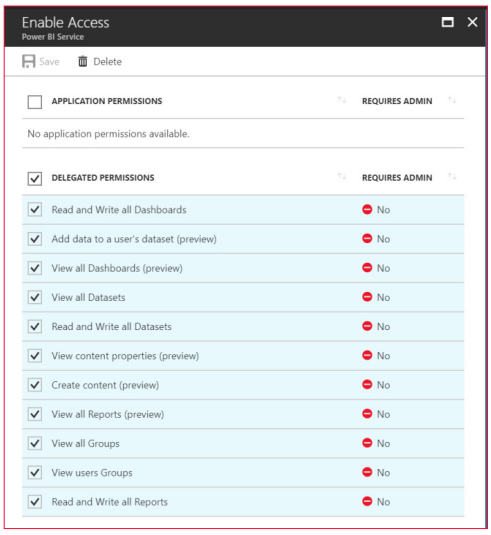


Imagen 8.- Permisos seleccionados en Delegated Permissions.

- Seleccionar Otorgar Permisos

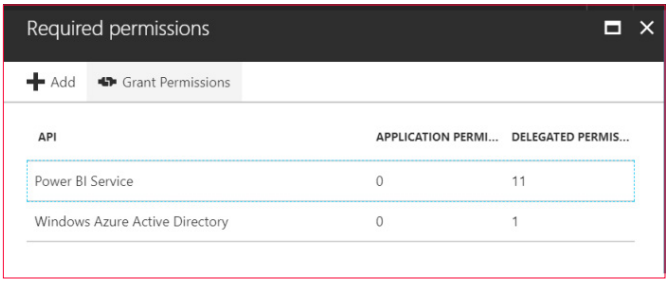


Imagen 9.- Otorgando permisos.

Instalación del Módulo de Power BI y dependencias

Ahora instalaremos el Módulo de Power BI y dependencias. En primer lugar, lo descargamos desde la galería de PowerShell. Solo debemos abrir una ventana de PowerShell e ingresar los siguientes comandos:

```
Install-Module -Name Microsoft.ADAL.PowerShell
Install-Module -Name Microsoft.PowerBI.PowerShell
```

En estos momentos ya estamos en condiciones de comenzar a utilizar la integración que hemos realizado. Por lo tanto, empezaremos a utilizar el Módulo de Power BI y visualizar la información de salida en un Reporte de Power BI. Los pasos que daremos:

- Qué podemos lograr con el Modulo de Power BI (comandos).
- Escribir un script de reporting para enviar datos a Power BI.
- Crear un reporte a partir de un DataSet en Power BI.

Comandos disponibles en el Módulo de Power BI en PowerShell

Los comandos disponibles en el Módulo de Power BI para PowerShell son los siguientes:

- Connect-PowerBi
- Switch-PowerBIContext
- Add-PowerBIDataSet
- Get-PowerBIDataSets
- Get-PowerBITables
- Update-PowerBITableSchema
- Add-PowerBIRows
- Remove-PowerBIRows
- Get-PowerBIGroups
- New-PowerBIDataSet
- New-PowerBITable
- New-PowerBIColumn

Con estos comandos somos capaces de crear y editar un dataset. Recordemos que podemos darle un vistazo para detalles a la página oficial del Módulo de Power BI aquí:

https://www.powershellgallery.com/packages/Microsoft.PowerBI.PowerShell/1.2/Content/Microsoft.PowerBI.PowerShell.psm1%0b%0b

Escribir un Script para envío de datos a Power BI

En una primer, y simple etapa podemos crear un script para enviar datos a Power BI. El script se encargará de recopilar información de Grupos de Office 365, Administradores y Miembros y publicar los mismos en Power BI.

- Script de Configuración: Esta primer etapa recoge información de Office 365 y crea un dataset en Power BI donde podremos alojar dicha información.

```
$Username = "MyAccount@mydomain.onmicrosoft.com"
$Password = "**********"

#Define $info array
$info = @()

#Get all groups
$Groups = Get-UnifiedGroup | Select-Object Alias,Access-
type,ManagedBy,PrimarySmtpAddress,Displayname,-
Notes,GroupMemberCount,GroupExternalMemberCount,When-
Changed

foreach($Group in $Groups) {

  Write-Host -Object "Number of Groups left to process $Group-
sCount" -ForegroundColor Green
  $Members = Get-UnifiedGroupLinks -Identity $Group.alias
-LinkType members
  $Owners = Get-UnifiedGroupLinks -Identity $Group.alias
-LinkType owners
  $MembersCount = $Members.count
  $OwnerCount = $Group.ManagedBy

  foreach($Owner in $Owners){
    $Object=[PSCustomObject]@{
      Name = $Group.Displayname
      Group = $Group.Alias
      Email = $Group.PrimarySmtpAddress
      UserName = $Owner.name
      NumberOfMembers = $Group.GroupMemberCount
      MemberOrOwner = 'Owner'
      NumberOfOwners = $OwnerCount.count
      GroupType = $Group.AccessType
      ExternalMemberCount = $Group.GroupExternalMember-
Count
      WhenChanged = $Group.WhenChanged | Get-Date -For-
mat 'yyyy.MM.dd hh:mm'
      Description = $Group.Notes
      }#EndPSCustomObject
    $info+=$object
  }

  foreach($Member in $Members){
    $Object=[PSCustomObject]@{
      Name = $Group.Displayname
      Group = $Group.Alias
      Email = $Group.PrimarySmtpAddress
      UserName = $Member.name
      NumberOfMembers = $Group.GroupMemberCount
      MemberOrOwner = 'Member'
      NumberOfOwners = $OwnerCount.count
      GroupType = $Group.AccessType
      ExternalMemberCount = $Group.GroupExternalMember-
Count
      WhenChanged = $Group.WhenChanged | Get-Date -For-
mat 'yyyy.MM.dd hh:mm'
      Description = $Group.Notes
      }#EndPSCustomObject
$info+=$object
  }
  $GroupsCount--
}

#Connects to PowerBI
Connect-PowerBI -AuthorityName MyAccount.onmicrosoft.com
-ClientId 'fa7af8a7-56ad-429f-8f57-76b1bd2087e1' -UserName
$username -Password $password
###

#Groups reporting data.
#Defines collums in the table you are going to create.

$col1 = New-PowerBIColumn -ColumnName Name -Column-
Type String
$col2 = New-PowerBIColumn -ColumnName Group -Column-
Type String
$col3 = New-PowerBIColumn -ColumnName Email -Column-
Type String
$col4 = New-PowerBIColumn -ColumnName UserName
-ColumnType String
$col5 = New-PowerBIColumn -ColumnName NumberOfMem-
bers -ColumnType Int64
$col6 = New-PowerBIColumn -ColumnName MemberOrOwner
-ColumnType String
$col7 = New-PowerBIColumn -ColumnName NumberOfOwners
-ColumnType Int64
```

```
$col8 = New-PowerBIColumn -ColumnName GroupType -Column-
Type String
$col9 = New-PowerBIColumn -ColumnName ExternalMemberCount
-ColumnType Int64
$col10 = New-PowerBIColumn -ColumnName WhenChanged
-ColumnType DateTime
$col11 = New-PowerBIColumn -ColumnName Description -Column-
Type String

#Creates table from defined collums.
#Comment out after first time setup.
$table1 = New-PowerBITable -TableName GroupReport -Columns
$col1,$col2,$col3,$col4,$col5,$col6,$col7,$col8,$col9,$col10,$col11

#Creates dataset from defined table.
#Comment out after first time setup.
$dataset = New-PowerBIDataSet -DataSetName GroupReport -Ta-
bles $table1

#Adds dataset and get datasetid.
#Comment out after first time setup.
$datasetid = Add-PowerBIDataSet -DataSet $dataset

#Take note of datasetid so you have it for when you are updating the
dataset.
$datasetid

#Remove commeting after firsttime setup on line below.
#Remove-PowerBIRows -DataSetId datasetid -TableName $table1

#Set datasetid manualy after first time setup
Add-PowerBIRows -DataSetId $datasetid -TableName GroupReport
-Rows $info
```

- Crear un reporte a partir de un DataSet en Power BI. Los pasos por seguir para crear un primer reporte a partir del Dataset creado son:
  - Ingresamos a https://powerbi.microsoft.com y nos logueamos con nuestra cuenta.
  - Nos dirigimos a Mi ambiente de trabajo y seleccionamos desde el cuadro desplegable en Datasets, y desde Acciones seleccionamos la opción de Crear Reporte

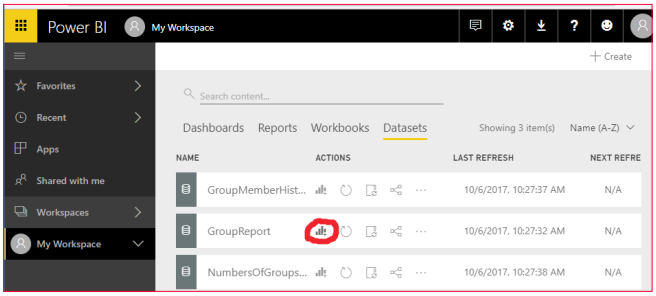


Imagen 10.- Acceso a la creación del informe.

- Seleccionamos Stack Columns como formato visual.
- Al tener seleccionado este formato grafico seleccionamos los campos Group, y NumberofMembers
- Seleccionamos la opción Contar debajo de Valores.

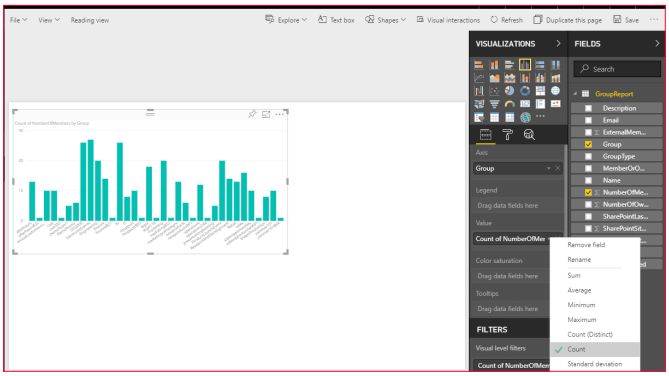


Imagen 11.- Seleccionar la opción de Contar.

"empezaremos a utilizar el Módulo de Power BI y visualizar la información de salida en un Reporte de Power BI"

- Dejamos de seleccionar la gráfica, y expandimos la entidad UserName debajo de Campos, Ahora tenemos una tabla con todos los usuarios que son miembros de nuestros grupos de Office 365.

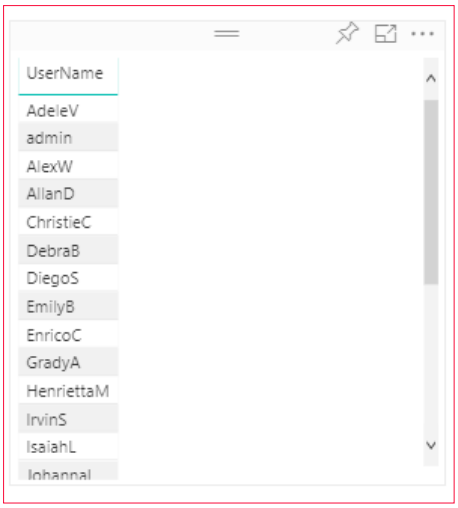


Imagen 12.- Tabla con los usuarios miembros de grupos de Office 365.

- Al seleccionar en la tabla de UserName y dar click sobre un nombre y podremos ver la magia de Power BI. Podemos ver ahora en la gráfica de columnas a que grupos pertenece el usuario seleccionado

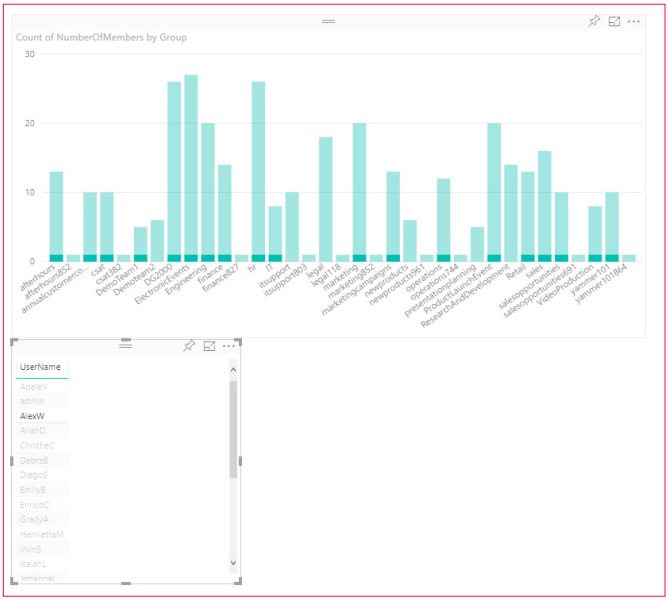


Imagen 13.- Informe operativo.

GASTÓN CRUZ
Data Plaform MVP
Business Intelligence Technical Manager
@GastoFCruz

37

SharePoint y Azure: El Microsoft Translator API

Que es el Microsoft Translator API

El servicio de Traducción de Azure permite, entre otras cosas, traducir textos de un idioma a otro automáticamente, permitiendo integrar textos en múltiples idiomas desde cualquier tipo de aplicación. El servicio forma parte del conjunto de Cognitive Services de Azure, que es una colección de algoritmos de Inteligencia Artificial que se pueden utilizar en la nube de Microsoft.

El servicio de Traducción ha utilizado desde su principio un algoritmo de Traducción Estadística Mecánica (SMT, Statistical Machine Translation), pero como esta plataforma ya ha alcanzado su máximo desarrollo y no es posible mejorarla, Microsoft ha comenzado a utilizar otro tipo de algoritmo basado en Inteligencia Artificial llamado DNN, Deep Neural Networks. El cambio entre los dos algoritmos se esta haciendo incrementalmente, pero los usuarios solamente notaran las mejoras en cuanto a la calidad de las traducciones pues la API de comunicación no cambia. DNN ofrece mejores traducciones porque utiliza el contexto de toda la frase para crear el texto traducido, no solamente el contexto de algunas palabras antes y después de la palabra a traducir.

El Azure Translator API es una herramienta ideal para utilizar en SharePoint: una tarea recurrente en la creación de Portales es poder ofrecer la misma información en varios idiomas simultáneamente. Aunque en este momento la calidad de las traducciones no es totalmente perfecta (y probablemente nunca lo será), el nivel es suficientemente alto como para poder reducir considerablemente los costos de traducción manual ya que los traductores solamente necesitan revisar los textos, no traducirlos desde cero. La calidad de las traducciones varia dependiendo de los idiomas: traducir de un idioma cualquiera al inglés tiene generalmente mucha mas calidad que traducir desde el inglés a otro idioma. También se puede observar que la calidad depende del idioma: lenguajes muy utilizados como el español y francés producen resultados de traducción superiores a idiomas menos usados. En este momento el servicio de traducción soporta 62 idiomas, incluyendo dos dialectos de Klingon, Yucatec Maya y Hmong Daw. Microsoft trabaja constantemente para aumentar el numero de idiomas, pero hay que tener en cuenta que se necesita aproximadamente un millón de palabras en textos originales para entrenar los algoritmos de IA, lo que no facilita el trabajo.

Configuración del Azure Translator API

Para utilizar el Translator API es necesario crear primero el servicio en Azure:

1.– Entre al portal de manejo de Azure (https://portal.azure.com) utilizando sus credenciales.
2.– Vaya a la sección de “Resource Groups” y cree un nuevo Grupo de Recursos (también es posible reutilizar un grupo ya existente).
3.– Cree un servicio de “Translator Text API”:
- En el Resource Group, utilice el botón de “+Add” para crear un recurso, busque por “translator” en la casilla de búsqueda y seleccione “Translator Text API” en los resultados.
- Asígnele un nombre al servicio y utilice el Grupo de Recursos deseado. En la casilla de “Pricing tier” seleccione un servicio dependiendo de la cantidad de traducciones a esperar por mes, lo que determina el precio del servicio (gratis si se utilizan menos de 2.000.000 de caracteres por mes). Acepte el anuncio de privacidad que aparece en la configuración (Microsoft utilizara los datos enviados para mejorar automáticamente los algoritmos de traducción).

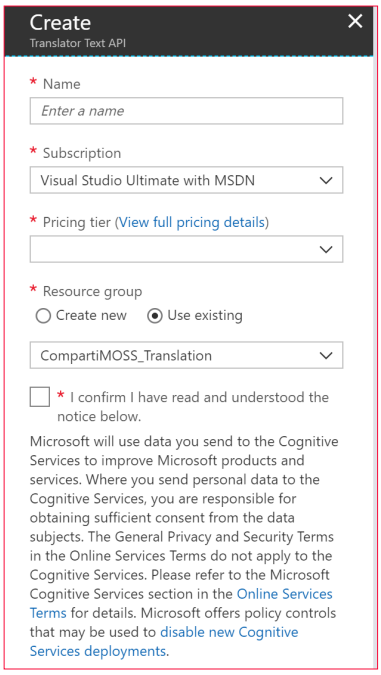


Imagen 1.- Creación del servicio de Traducción API.

4.– Una vez creado el servicio, haga clic sobre su nombre en la lista de recursos del Resource Group, vaya a “Keys” y copie el valor de “Key 1”.

“El servicio de Traducción ha utilizado desde su principio un algoritmo de Traducción Estadística Mecánica (SMT, Statistical Machine Translation)”

Utilizando el Azure Translation API con SharePoint

En el siguiente ejemplo se van a utilizar dos Listas de Anuncios de SharePoint en donde el campo de “Body” de un elemento nuevo creado en la Lista en Español será traducido al inglés y utilizado para crear otro anuncio en la Lista de Anuncios en inglés. Cuando se introduce un nuevo elemento en la Lista de Anuncios en español, un WebHook hace que una Función de Azure comience a funcionar, utilice el texto del “Body” para hacer una llamada al Azure Translation API y cree el elemento en la Lista en Ingles.

Nota: la creación y configuración de una Función de Azure se puede encontrar el en artículo “SharePoint y Azure – Azure Functions” (http://www.compartimoss.com/revistas/numero-30/sharepoint-y-azure-azure-functions). La configuración y utilización de WebHooks de SharePoint se puede encontrar en el artículo “Eventos sobre SharePoint Online con Webhooks” (http://www.compartimoss.com/revistas/numero-32/eventos-sobre-sharepoint-online-con-webhooks).

5.– Cree una cuenta de Funciones básica en el Grupo de Recursos, asignándole un nombre, Plan de Servicios y cuenta de Azure Storage.

6.– Utilizando Visual Studio 2017 (o Visual Studio 2016 con el AddIn para programar Funciones de Azure), cree una nueva solución del tipo “Azure Function”. Una vez creada la solución, agréguele una Función del tipo “Http Trigger” con derechos de acceso anónimos.

7.– Agregue a la solución los paquetes NuGet, “AppForSharePointOnlineWebToolkit” y “Newtonsoft.Json”.

8.– Reemplace toda la rutina “Run” con el siguiente código:

```
[FunctionName(“TranslationFunction”)]
public static async Task<HttpResponseMessage> Run([HttpTrigger(AuthorizationLevel.Anonymous, “post”, Route = null)]
HttpRequestMessage req, TraceWriter log)
{
  // Registration
  string validationToken = req.GetQueryNameValuePairs()
    .FirstOrDefault(q => string.Compare(q.Key, “validationtoken”, true) == 0)
    .Value;
  if (validationToken != null)
  {
    var myResponse = req.CreateResponse(HttpStatusCode.OK);
    myResponse.Content = new StringContent(validationToken);
    return myResponse;
  }

  // Changes
  var myContent = await req.Content.ReadAsStringAsync();
  var allNotifications = JsonConvert.DeserializeObject<ResponseModel<NotificationModel>>(myContent).Value;

  if (allNotifications.Count > 0)
  {
    foreach (var oneNotification in allNotifications)
    {
      // Login in SharePoint
      string baseUrl = “https://gustavo.sharepoint.com”;
      string myUserName = “gustavo@gustavo.onmicrosoft.com”;
      string myPassword = ConfigurationManager.AppSettings[“gustavoPW”];

      SecureString securePassword = new SecureString();
      foreach (char oneChar in myPassword) securePassword.AppendChar(oneChar);
      SharePointOnlineCredentials myCredentials = new SharePointOnlineCredentials(myUserName, securePassword);

      ClientContext SPClientContext = new ClientContext(baseUrl + oneNotification.SiteUrl);
      SPClientContext.Credentials = myCredentials;

      // Get the Changes
      GetChanges(SPClientContext, oneNotification.Resource, log);
    }
  }

  return new HttpResponseMessage(HttpStatusCode.OK);
}
```

Esta rutina primero se encarga de hacer el registro del WebHook (si la consulta contiene un parámetro “validationtoken” en el Query String). Después de registrado el WebHook, cada consulta es procesada para extraer las notificaciones que contiene. En cada notificación de la colección de notificaciones se hace un logeo en SharePoint para obtener los cambios detectados en la Lista (por medio de la rutina “GetChanges”).

9.– La rutina “GetChanges” recibe el contexto de SharePoint y el identificador de la Lista, y tiene la forma:

```
static void GetChanges(ClientContext SPClientContext, string ListId, TraceWriter log)
{
  // Get the List
  Web spWeb = SPClientContext.Site.RootWeb;
  List listaAnunciosES = spWeb.Lists.GetByTitle(“AnnouncementsES”);
  SPClientContext.Load(listaAnunciosES);
  List listaAnunciosEN = spWeb.Lists.GetByTitle(“AnnouncementsEN”);
  SPClientContext.Load(listaAnunciosEN);
  SPClientContext.ExecuteQuery();

  // Create the ChangeToken and Change Query
  ChangeToken lastChangeToken = new ChangeToken();
  lastChangeToken.StringValue = string.Format(“1;3;{0};{1};-1”, ListId, DateTime.Now.AddMinutes(-1).ToUniversalTime().Ticks.ToString());
  ChangeToken newChangeToken = new ChangeToken();
  newChangeToken.StringValue = string.Format(“1;3;{0};{1};-1”, ListId, DateTime.Now.ToUniversalTime().Ticks.ToString());
  ChangeQuery myChangeQuery = new ChangeQuery(false, false);
  myChangeQuery.Item = true;  // Get only Item changes
  myChangeQuery.Add = true;  // Get only the new Items
```

```
    myChangeQuery.ChangeTokenStart = lastChangeToken;
    myChangeQuery.ChangeTokenEnd = newChangeToken;

    // Get all the Changes
    var allChanges = listaAnunciosES.GetChanges(my-
ChangeQuery);
    SPClientContext.Load(allChanges);
    SPClientContext.ExecuteQuery();

    foreach (Change oneChange in allChanges)
    {
      if (oneChange is ChangeItem)
      {
        // Get what is changed
        ListItem anuncioES = listaAnunciosES.GetItemBy-
Id((oneChange as ChangeItem).ItemId);
        SPClientContext.Load(anuncioES);
        SPClientContext.ExecuteQuery();

        string textoParaTraducir = anuncioES["Body"].ToString().
Replace("\"", "'");
        string lenguajeDe = "es";
        string lenguajeA = "en";
        string apiKey = ConfigurationManager.AppSet-
tings["apiKey"];
        string textoTraducido = getTraduccion(textoParaTraducir,
lenguajeDe, lenguajeA, apiKey);

        // Insert the values in the EN List
        ListItemCreationInformation newAnuncioENInfo = new
ListItemCreationInformation();
        ListItem anuncioEN = listaAnunciosEN.AddItem(newAnun-
cioENInfo);
        anuncioEN["Title"] = anuncioES["Title"];
        anuncioEN["Body"] = textoTraducido;
        anuncioEN.Update();
        SPClientContext.ExecuteQuery();
      }
    }
}
```

Primero se crean dos objetos que contienen las Listas de Anuncios de SharePoint. Luego se crea una consulta de cambio (variable "myChangeQuery") que especifica que se requieren los cambios ocurridos en el ultimo minuto, que ocurren en elementos de la Lista y que sean del tipo "Add", es decir, elementos nuevos. Luego de ejecutar la consulta, se examina cada uno de los cambios y se obtiene un objeto con el Anuncio agregado (el que contiene, a su vez, el texto del "Body").

"Después de registrado el WebHook, cada consulta es procesada para extraer las notificaciones que contiene"

En la misma rutina se llama a la rutina "getTraduccion", la que se encarga de hacer la consulta en Azure para traducir el texto, utilizando como parámetros de entrada el texto a traducir, el idioma de origen, el idioma de destino y la llave obtenida en el punto 4. Cuando ya se ha obtenido el texto traducido, se crea un nuevo elemento en la Lista de Anuncios en inglés.

10.– La rutina "getTraduccion" recibe como parámetros de entrada el texto a traducir, los idiomas de origen y de destino y la clave del servicio de Traducción:

```
static string getTraduccion(string TextoParaTraducir, string
LenguajeDe, string LenguajeA, string ApiKey)
{
    string txtTraducido = string.Empty;

    AzureAuthToken authTokenSource = new AzureAuthTo-
ken(ApiKey.Trim());
    string authToken = authTokenSource.GetAccessTokenAsync().
Result;

    string uriTraduccion = "https://api.microsofttranslator.com/
v2/Http.svc/Translate?text=" +
            HttpUtility.UrlEncode(TextoParaTraducir) +
            "&from=" + LenguajeDe + "&to=" + LenguajeA;
    HttpWebRequest httpWebRequest = (HttpWebRequest)We-
bRequest.Create(uriTraduccion);
    httpWebRequest.Headers.Add("Authorization", authToken);
    using (WebResponse myResponse = httpWebRequest.Get-
Response())
    using (Stream stream = myResponse.GetResponseStream())
    {
      DataContractSerializer mySerializer = new DataContractSe-
rializer(Type.GetType("System.String"));
      txtTraducido = (string)mySerializer.ReadObject(stream);
    }

    return txtTraducido;
}
```

Cada consulta al Translation API se realiza por medio de una llamada REST a un URL pre-especificado del servicio de Traducción, utilizando como parámetros los dos valores de los idiomas y el texto a traducir. En este caso se está utilizando el método "Translate" en la llamada REST, pero existen otros 12 métodos que se pueden utilizar, incluyendo métodos para traducir no un texto solo sino también un array de textos en una sola llamada, detectar el idioma original, enumerar los idiomas que se pueden utilizar y separar palabras en un texto completo.

Esta rutina llama inicialmente el método "GetAccessTokenAsync" de la clase "AzureAuthToken" para obtener el token de acceso al servicio de traducción utilizando la llave obtenida al momento de configuración descrito en el punto 4. Esta clase tiene un método sincrónico y otro asincrónico para obtener el token, pero Microsoft especifica que es preferible utilizar el método asincrónico. La clase ha sido creada por Microsoft, se puede encontrar en la información de Azure y tiene la forma:

```
using System;
using System.Net;
using System.Net.Http;
using System.Threading.Tasks;

namespace TranslationFunction
{
  /// <summary>
  /// Client to call Cognitive Services Azure Auth Token service in
order to get an access token.
  /// Exposes asynchronous as well as synchronous methods.
  /// </summary>
  public class AzureAuthToken
  {
    /// URL of the token service
    private static readonly Uri ServiceUrl = new Uri("https://api.
cognitive.microsoft.com/sts/v1.0/issueToken");

    /// Name of header used to pass the subscription key to the
token service
    private const string OcpApimSubscriptionKeyHeader =
"Ocp-Apim-Subscription-Key";

    /// After obtaining a valid token, this class will cache it for
this duration.
```

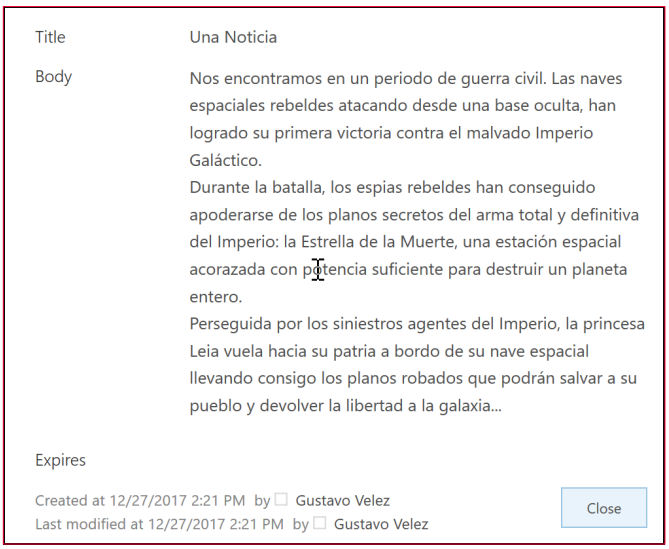
```
    /// Use a duration of 5 minutes, which is less than the actual token lifetime of 10 minutes.
    private static readonly TimeSpan TokenCacheDuration = new TimeSpan(0, 5, 0);

    /// Cache the value of the last valid token obtained from the token service.
    private string _storedTokenValue = string.Empty;

    /// When the last valid token was obtained.
    private DateTime _storedTokenTime = DateTime.MinValue;

    /// Gets the subscription key.
    public string SubscriptionKey { get; }

    /// Gets the HTTP status code for the most recent request to the token service.
    public HttpStatusCode RequestStatusCode { get; private set; }

    /// <summary>
    /// Creates a client to obtain an access token.
    /// </summary>
    /// <param name="key">Subscription key to use to get an authentication token.</param>
    public AzureAuthToken(string key)
    {
      if (string.IsNullOrEmpty(key))
      {
        throw new ArgumentNullException(nameof(key), "A subscription key is required");
      }

      this.SubscriptionKey = key;
      this.RequestStatusCode = HttpStatusCode.InternalServerError;
    }

    /// <summary>
    /// Gets a token for the specified subscription.
    /// </summary>
    /// <returns>The encoded JWT token prefixed with the string "Bearer ".</returns>
    /// <remarks>
    /// This method uses a cache to limit the number of request to the token service.
    /// A fresh token can be re-used during its lifetime of 10 minutes. After a successful
    /// request to the token service, this method caches the access token. Subsequent
    /// invocations of the method return the cached token for the next 5 minutes. After
    /// 5 minutes, a new token is fetched from the token service and the cache is updated.
    /// </remarks>
    public async Task<string> GetAccessTokenAsync()
    {
      if (string.IsNullOrWhiteSpace(this.SubscriptionKey))
      {
        return string.Empty;
      }

      // Re-use the cached token if there is one.
      if ((DateTime.Now - _storedTokenTime) < TokenCacheDuration)
      {
        return _storedTokenValue;
      }

      using (var client = new HttpClient())
      using (var request = new HttpRequestMessage())
      {
        request.Method = HttpMethod.Post;
        request.RequestUri = ServiceUrl;
        request.Content = new StringContent(string.Empty);
        request.Headers.TryAddWithoutValidation(OcpApimSubscriptionKeyHeader, this.SubscriptionKey);
        client.Timeout = TimeSpan.FromSeconds(2);
        var response = await client.SendAsync(request);
        this.RequestStatusCode = response.StatusCode;
        response.EnsureSuccessStatusCode();
        var token = await response.Content.ReadAsStringAsync();
        _storedTokenTime = DateTime.Now;
        _storedTokenValue = "Bearer " + token;
        return _storedTokenValue;
      }
    }

    /// <summary>
    /// Gets a token for the specified subscription. Synchronous version.
    /// Use of async version preferred
    /// </summary>
    /// <returns>The encoded JWT token prefixed with the string "Bearer ".</returns>
    /// <remarks>
    /// This method uses a cache to limit the number of request to the token service.
    /// A fresh token can be re-used during its lifetime of 10 minutes. After a successful
    /// request to the token service, this method caches the access token. Subsequent
    /// invocations of the method return the cached token for the next 5 minutes. After
    /// 5 minutes, a new token is fetched from the token service and the cache is updated.
    /// </remarks>
    public string GetAccessToken()
    {
      // Re-use the cached token if there is one.
      if ((DateTime.Now - _storedTokenTime) < TokenCacheDuration)
      {
        return _storedTokenValue;
      }

      string accessToken = null;
      var task = Task.Run(async () =>
      {
        accessToken = await this.GetAccessTokenAsync();
      });

      while (!task.IsCompleted)
      {
        System.Threading.Thread.Yield();
      }
      if (task.IsFaulted)
      {
        throw task.Exception;
      }
      if (task.IsCanceled)
      {
        throw new Exception("Timeout obtaining access token.");
      }
      return accessToken;
    }
  }
}
```

11.– Otras tres clases definen objetos utilizados por el WebHook:

```
public class ResponseModel<T>
{
  [JsonProperty(PropertyName = "value")]
  public List<T> Value { get; set; }
}

public class NotificationModel
{
  [JsonProperty(PropertyName = "subscriptionId")]
  public string SubscriptionId { get; set; }

  [JsonProperty(PropertyName = "clientState")]
  public string ClientState { get; set; }

  [JsonProperty(PropertyName = "expirationDateTime")]
  public DateTime ExpirationDateTime { get; set; }

  [JsonProperty(PropertyName = "resource")]
  public string Resource { get; set; }

  [JsonProperty(PropertyName = "tenantId")]
  public string TenantId { get; set; }
```

```
    [JsonProperty(PropertyName = "siteUrl")]
    public string SiteUrl { get; set; }

    [JsonProperty(PropertyName = "webId")]
    public string WebId { get; set; }
}

public class SubscriptionModel
{
    [JsonProperty(NullValueHandling = NullValueHandling.Ig-
nore)]
    public string Id { get; set; }

    [JsonProperty(PropertyName = "clientState", NullValueHan-
dling = NullValueHandling.Ignore)]
    public string ClientState { get; set; }

    [JsonProperty(PropertyName = "expirationDateTime")]
    public DateTime ExpirationDateTime { get; set; }

    [JsonProperty(PropertyName = "notificationUrl")]
    public string NotificationUrl { get; set; }

    [JsonProperty(PropertyName = "resource", NullValueHandling
= NullValueHandling.Ignore)]
    public string Resource { get; set; }
}
```

12.– Registre el WebHook en la Lista de Anuncios en español (utilizando Postman, por ejemplo, como se indica en el artículo mencionado al principio del artículo) y cree un anuncio nuevo en la Lista. El WebHook hará que la Función realice su trabajo, traduzca el texto, y cree un nuevo Anuncio en la Lista de Anuncios en inglés:



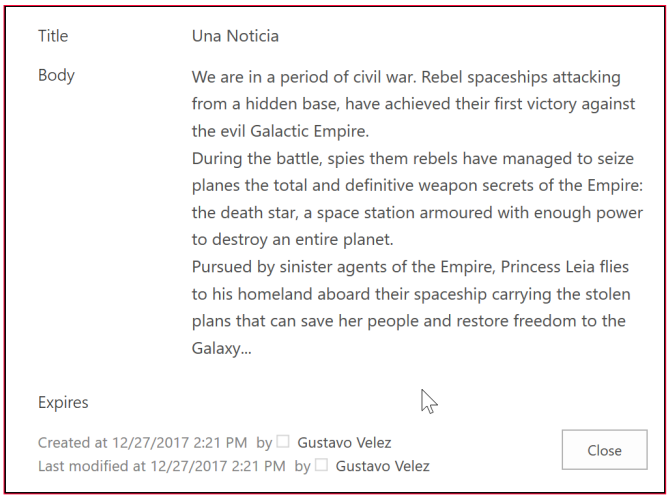


Imagen 2.- Anuncio original y traducido utilizando el servicio de Traducción API.

"puede llegar a ser una herramienta muy valiosa en SharePoint, especialmente en lo que se refiere a la creación de contenido traducido automáticamente"

Conclusiones

El servicio de Traducción de Texto de Azure puede llegar a ser una herramienta muy valiosa en SharePoint, especialmente en lo que se refiere a la creación de contenido traducido automáticamente a diferentes idiomas. El Azure Translation API es fácil de utilizar desde cualquiera lenguaje de programación, y produce resultados confiables rápida y seguramente. El API utiliza algoritmos de Inteligencia Artificial que se mejoran con el uso, por lo no es necesario crear ni entrenar algoritmos propios.

GUSTAVO VELEZ
MVP Office Servers and Services
gustavo@gavd.net
http://www.gavd.net

42

Patrones de Diseño de Typescript aplicados a SharePoint Framework : Contructor (Builder Pattern)

El patrón constructor nos permite construir objetos complejos usando objetos simples de una manera sencilla y paso a paso. Este patrón es de la categoría de patrones creaciones y nos provee una de las maneras mas sencillas para simplificar la creación de objetos.

"es de la categoría de patrones creaciones y nos provee una de las maneras mas sencillas para simplificar la creación de objeto"

La idea de este ejemplo es contruir un objeto complejo a partir de objetos sencillos, una Comida a partir del objeto hamburgesa, papas frías, gaseosa). Supón que tienes una lista en SharePoint para almacenar hamburgesas, otra para gaseosas, otra para postres, y quieres construir diferentes tipos de menu, eso seria el ejemplo perfecto.

UML

El diagrama UML de lo que vamos a construir es el siguiente (Figura 1):

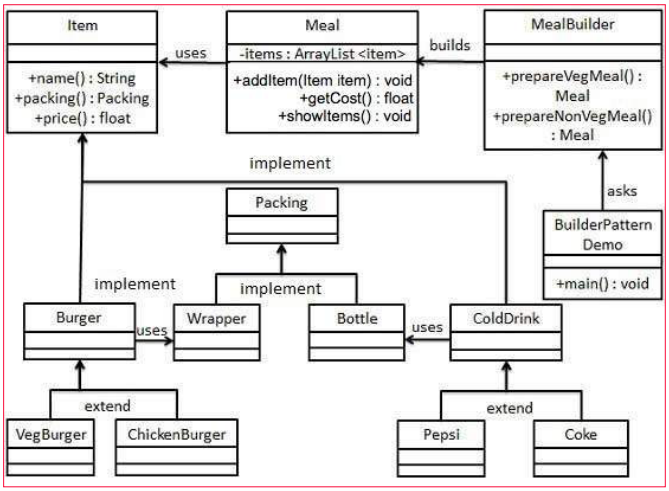


Imagen 1.- Digrama UML.

Estructura del Proyecto

Para el ejemplo hemos creado un componente con todas las clases necesarias y vamos a discutir las clases una por una.

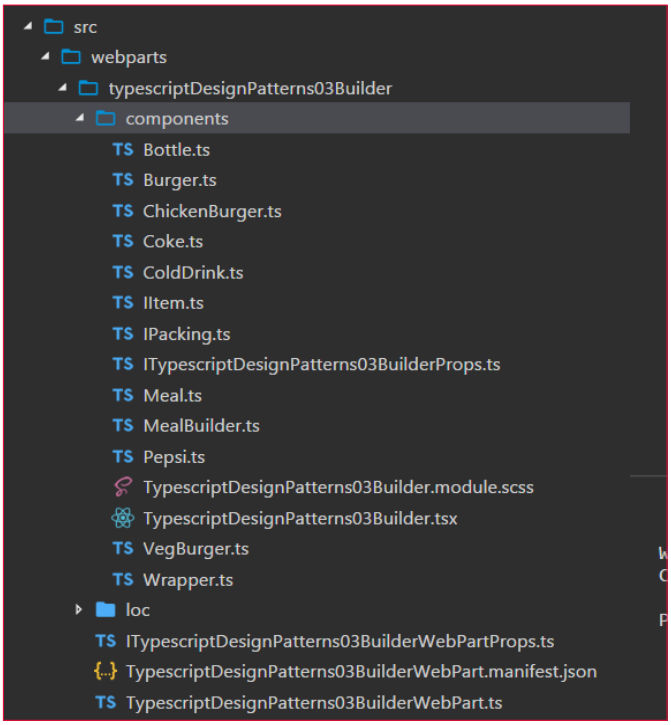


Imagen 2.- Estructura del Proyecto.

IItem.ts

Esta interfaz es la que necesitan implementar todos los productos para que estos tengan una estructura común.

```
Import Ipacking from "./Ipacking";

interface Iitem {
  name(): string;
  packing(): Ipacking;
  price(): number;
}

export default Iitem;
```

Ipacking.ts

Esta interfaz es la que necesitan implementar todos los tipos de Paquetes, por ejemplo botellas, bolsa, etc. Es la forma que podríamos definir comportamiento y propiedades communes segun el tipo de paquete usado por el producto.

```
interface IPacking {
  pack(): string;
}

export default IPacking;
```

Bottle.ts

Este es un tipo de paquete, implementa la interfaz IPacking.

```
import IPacking from "./IPacking";

class Bottle implements IPacking {
  public pack(): string {
    return "Bottle";
  }
}

export default Bottle;
```

Wrapper.ts

```
import IPacking from "./IPacking";

class Wrapper implements IPacking {
  public pack(): string {
    return "Wrapper";
  }
}

export default Wrapper;
```

Burger.ts

Esta es la clase abstracta de la cual nuestras clases especificas tienen que implementar, esta clase existe para que todas las hamburguesas tengan una estructura y comportamiento comun.

```
import IItem from "./IItem";
import Wrapper from "./Wrapper";
import IPacking from "./IPacking";

abstract class Burger implements IItem {
  public name(): string {
    throw new Error("Method not implemented.");
  }

  public packing(): IPacking {
    return new Wrapper();
  }

  public abstract price(): number ;

}

export default Burger;
```

ChickenBurger.ts

La hamburguesa de pollo, implementada los métodos necesarios.

```
import Burger from "./Burger";

class ChickenBurger extends Burger {
  public price(): number {
    return 15;
  }

  public name(): string {
    return "Chicken Burger";
  }
}

export default ChickenBurger;
```

VegBurger.ts

La hamburguesa vegetariana, implementando los métodos necesarios.

```
import Burger from "./Burger";

class VegBurger extends Burger {
  public price(): number {
    return 11;
  }
  public name(): string {
    return "Veg Burger";
  }
}

export default VegBurger;
```

Colddrink.ts

Todas las geseosas implementarían la clase abstracta de ColdDrink que define las propiedades y comportamiento común entre ellas.

```
import IItem from "./IItem";
import IPacking from "./IPacking";
import Bottle from "./Bottle";

abstract class ColdDrink implements IItem {
  public name(): string {
    throw new Error("Method not implemented.");
  }
  public packing(): IPacking {
    return new Bottle();
  }
  public abstract price(): number ;
}

export default ColdDrink;
```

Coke.ts

Una implementación de la clase anterior para definir su precio y nombre.

```
import ColdDrink from "./ColdDrink";

class Coke extends ColdDrink {
  public price(): number {
    return 2.5;
  }
  public name(): string {
    return "Coca Cola";
  }
}

export default Coke;
```

Pepsi.ts

```
import ColdDrink from "./ColdDrink";

class Pepsi extends ColdDrink {
  public price(): number {
    return 1.5;
  }
  public name(): string {
    return "Pepsi Cola";
  }
}

export default Pepsi;
```

Meal.ts

Esta clase representa el comportamiento completo para la construcción de un Menu, tiene métodos para agregar ítems al Menu, obtener el costo total, y mostrar los productos del menú.

```
import IItem from "./IItem";

class Meal {
  private items: IItem[];

  public addItem(item: IItem): void {
    this.items.push(item);
  }

  public getCost(): number {
    let cost: number  = 0;
    for(let item of this.items) {
      cost+= item.price();
    }

    return cost;
  }

  public showItems(): string {
    let returnStr: string;
    for(let item of this.items) {
      returnStr +="Item:" + item.name;
      returnStr +=", Packing:" + item.packing().pack();
      returnStr +=", Price: " + item.price();
    }
    return returnStr;
  }
}

export default Meal;
```

MealBuilder.ts

Esta clase es la que construye los Menus, retornando un objeto tipo Meal para cada tipo de Menu, el cual tiene diferentes productos. Por simplicidad se han creado solo 2 menus.

```
import Meal from "./Meal";
import VegBurger from "./VegBurger";
import Coke from "./Coke";
import ChickenBurger from "./ChickenBurger";

class MealBuilder {
  public prepareVegMeal(): Meal {
    let meal: Meal= new Meal();
    meal.addItem(new VegBurger());
    meal.addItem(new Coke());
    return meal;
  }
  public prepareNonVegMeal(): Meal {
    let meal: Meal= new Meal();
    meal.addItem(new ChickenBurger());
    meal.addItem(new Coke());
    return meal;
  }
}

export default MealBuilder;
```

ITypescriptDesignPatterns03BuilderProps.ts

Hemos creado una propiedad que nos guardara la decisión tomada por el usuario final (selectedMeal), y la agregamos a las "Props" del componente React.

```
export interface ITypescriptDesignPatterns03BuilderProps {
 description: string;
 selectedMeal: string;
}
```

TypescriptDesignPatterns03Builder.tsx

Esta es nuestra clase de componente, aquí tenemos un constructor y en el constructor llamamos el método SetMeal, en el cual enviamos la opción elegida por el usuario como parámetro. Una vez el menú esta construido, en el método render podemos utilizar el método showItems para mostrar lo seleccionado.

```
import * as React from "react";
import styles from "./TypescriptDesignPatterns03Builder.module.scss";
import { ITypescriptDesignPatterns03BuilderProps } from "./ITypescriptDesignPatterns03BuilderProps";
import { escape } from "@microsoft/sp-lodash-subset";
import MealBuilder from "./MealBuilder";
import Meal from "./Meal";
import { IPropertyPaneConfiguration } from "@microsoft/sp-webpart-base/lib/propertyPane/propertyPane/IPropertyPane";
import {
 PropertyPaneDropdown
} from "@microsoft/sp-webpart-base";
import Version from "@microsoft/sp-core-library/lib/Version";

export default class TypescriptDesignPatterns03Builder extends React.Component<ITypescriptDesignPatterns03BuilderProps, {}> {
 private mealBuilder: MealBuilder ;
 private items: string;
 private meal: Meal;
 constructor(props: ITypescriptDesignPatterns03BuilderProps, state: any) {
  super(props);
  this.setMeal(props.selectedMeal);
  this.mealBuilder = new MealBuilder();
 }
 public render(): React.ReactElement<ITypescriptDesignPatterns03BuilderProps> {
  return (
    <div className={styles.typescriptDesignPatterns03Builder}>
      <div className={styles.container}>
       <div className={`ms-Grid-row ms-bgColor-themeDark ms-fontColor-white ${styles.row}`}>
        <div className="ms-Grid-col ms-lg10 ms-xl8 ms-xlPush2 ms-lgPush1">
         <span className="ms-font-xl ms-fontColor-white">Welcome to Burger Company!</span>
         <p className="ms-font-l ms-fontColor-white">You have selected the following.</p>
          <span className={styles.label}>{this.meal.showItems()}</span>
        </div>
       </div>
      </div>
    </div>
   );
  }

 protected get dataVersion(): Version {
  return Version.parse("1.0");
 }
 private setMeal(selectedMeal: string): void {
  if(selectedMeal === "VegMeal") {
    this.meal = this.mealBuilder.prepareVegMeal();
  }
  if(selectedMeal === "NonVegMeal") {
   this.meal = this.mealBuilder.prepareNonVegMeal();
 }
 }
}
```

Y finalmente

TypescriptDesignPatterns03BuilderWebPart.ts

Esta es la clase del webpart que genera Sharepoint Framework, aquí lo único que hacemos es definir la propiedad de tipo DropDown, y luego utilizar el componente con la opción seleccionada.

```
import * as React from "react";
import * as ReactDom from "react-dom";
import { Version } from "@microsoft/sp-core-library";
import {
  BaseClientSideWebPart,
  IPropertyPaneConfiguration,
  PropertyPaneTextField,
  PropertyPaneDropdown
} from "@microsoft/sp-webpart-base";

import * as strings from "TypescriptDesignPatterns03BuilderWebPartStrings";
import TypescriptDesignPatterns03Builder from "./components/TypescriptDesignPatterns03Builder";
import { ITypescriptDesignPatterns03BuilderProps } from "./components/ITypescriptDesignPatterns03BuilderProps";
import { ITypescriptDesignPatterns03BuilderWebPartProps } from "./ITypescriptDesignPatterns03BuilderWebPartProps";

export default class TypescriptDesignPatterns03BuilderWebPart extends
  BaseClientSideWebPart<ITypescriptDesignPatterns03BuilderWebPartProps> {

  public render(): void {
    const element: React.ReactElement<ITypescriptDesignPatterns03BuilderProps > = React.createElement(
      TypescriptDesignPatterns03Builder,
      {
        description: this.properties.description,
        selectedMeal: this.properties.selectedMeal
      }
    );

    ReactDom.render(element, this.domElement);
  }

  protected get dataVersion(): Version {
    return Version.parse("1.0");
  }

  protected getPropertyPaneConfiguration(): IPropertyPaneConfiguration {
    return {
      pages: [
        {
          header: {
            description: "Header"
          },
          groups: [
            {
              groupName: "Group",
              groupFields: [
                PropertyPaneDropdown("meal", {
                  label: "Select meal",
                  options: [
                    { key: "Veg", text: "Veg" },
                    { key: "Nonveg", text: "Nonveg" }
                  ],
                  selectedKey: "Nonveg"
                })
              ]
            }
          ]
        }
      ]
    };
  }
}
```

"TypescriptDesignPatterns03BuilderWebPart.ts es la clase del webpart que genera Sharepoint Framework"

El Proyecto se encuentra en mi repositorio de github: https://github.com/levalencia/TypescriptDesignPatterns03-Builder

LUIS VALENCIA
MVP SharePoint
levm38@outlook.com
@levalencia
http://www.luisevalencia.com

52

Nosotros

Alberto Diaz

Alberto Díaz es SharePoint Team Lead en Encamina, liderando el desarrollo de software con tecnología Microsoft. Para la comunidad, ha fundado TenerifeDev (www.tenerifedev.com) con otros colaboradores, un grupo de usuarios de .NET en Tenerife, y coordinador de SUGES (Grupo de Usuarios de SharePoint de España, www.suges.es) y colaborador con otras comunidades de usuarios. Microsoft MVP de SharePoint Server desde el año 2011 y asiduo conferenciante en webcast y conferencias de tecnología de habla hispana.

Sitio Web: http://blogs.encamina.com/negocios-sharepoint/
Email: adiazcan@hotmail.com
Blogs: http://geeks.ms/blogs/adiazmartin
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 12 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 Servers & Services y co-fundador del Grupo de Usuarios de SharePoint de España (SUGES, www.suges.es), del Grupo de Usuarios de Cloud Computing de España (CLOUDES) y de la Comunidad de Office 365. Hasta la fecha, ha publicado 9 libros sobre SharePoint & Office 365, así como varios artículos en castellano y en inglés sobre ambas plataformas.

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

Coordinadores de sección

DAVID RIVERA FERNÁNDEZ
Coordinador de System Center
driverafer@hotmail.com

GEOVANY ACEVEDO
Coordinador de Exchange Server
geovany.acevedo@hotmail.com

GASTÓN CRUZ
Coordinador de PowerBi
gastoncruz@gmail.com

XAVIER MORERA
Coordinador de .Net
xavier@familiamorera.com

PABLO PERALTA
Coordinador de Dynamics CRM
pablop2006@gmail.com

JOHN BARRETO
Coordinador de System Center
john.barreto22@hotmail.com

RODOLFO CASTRO AGUILAR
Coordinador de Skype for Business
rodolfo-castro@live.com

ESTEBAN SOLANO GRANADOS
Coordinador de Xamarin
stvansolano@outlook.com

MAXI ACCOTTO
Coordinador de SQL Server
maxi.accotto@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