Nº46 diciembre 2020

CompartiMOSS

REVISTA ESPECIALIZADA EN TECNOLOGÍAS MICROSOFT

Entrevista José Manuel Alarcón

Evolución de SQL Server en múltiples plataformas

Procesando formularios con Form Recognizer y Azure Functions

Troubleshooting del networking de Azure mediante Network Watcher

Contenido

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
https://jcgonzalezmartin.wordpress.com/
http://blog.siderys.com
https://adiazcan.github.com

REDES SOCIALES

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

03

Editorial

Este año que se está terminando quedará marcado en la historia de todos nosotros. En este 2020 que se está yendo nos azotó una pandemia mundial, el mundo se vio envuelto en una lucha contra un virus que nos desbastó, nos aisló y nos encerró en nuestras propias prisiones, nuestras casas. A lo largo de todos estos meses muchas personas perdieron seres queridos sin poder despedirse, otras se pusieron el uniforme que les correspondía y salieron a dar pelea a este virus que nos estaba ganando, y hoy en día nos sigue ganando; a todos ellos queremos darles las gracias, ¡gracias por su lucha incondicional

En este 2020 que termina, empresas como Microsoft, Google y otras tantas, dejaron sus intereses de lado y pusieron su estructura de servicios a disposición de cada uno de nosotros. Sin ir más lejos, Microsoft, cuando el virus empezó hacer estragos a nivel mundial, libero Microsoft Teams para que todos, estuviéramos donde estuviéramos, pudiéramos estar conectados, comunicados y trabajar de forma remota. El teletrabajo, en menos de una semana, se transformó en una modalidad mundial para todos. Y muchas empresas, que jamás pensaron en implementar está forma de trabajar, tuvieron que adaptarse rápidamente, y a través de los servicios de la nube encontraron la solución.

Este es el último número del año y como hace más de 10 años, desde el equipo de CompartiMOSS queremos desearles que terminen este 2020 lo mejor posible y que el 2021 sea un año diferente, nos encuentre a todos más unidos.

Disfruten este nuevo número tanto como nosotros, los editores y autores, disfrutamos creándolo.

El Equipo Editorial de CompartiMOSS

04

Pon un Bot en tu vida… y en Microsoft Teams

Cada vez más nos estamos dando cuenta que la “era de la velocidad” nos está poniendo a prueba en muchos ámbitos. Uno de ellos en estos últimos meses ha sido el teletrabajo y la rapidez con la que nos hemos visto forzados a adaptar este método de colaboración e interacción virtual con nuestros clientes y compañeros.

Uno de los desafíos más importantes ha sido la falta de tiempo para implementar sistemas de ayuda y soporte a los empleados, la incertidumbre nos ha afectado a todos tanto a nivel personal como a nivel empresarial, para que nos entendamos, nos ha pillado el toro.

Al estar en una situación así y viendo la necesidad del factor <<rapidez>> al dar respuestas, basadas en soluciones de fácil creación y mantenimiento, me viene a la cabeza una cosa: Microsoft Power Platform.

En este articulo vamos a ver desde varios enfoques (técnico y funcional) la manera de tener informados y actualizados sobre las noticias, cambios de políticas, preguntas frecuentes y una infinidad de casos de uso, a nuestros empleados utilizando estas 5 herramientas que Microsoft nos proporciona:

- Microsoft Power Virtual Agents: agente virtual que nos ayudará a conectar a los usuarios con la información de distintas fuentes gracias al lenguaje natural.
- Microsoft Power Automate: servicio de flujos de trabajo que nos ayudará a ampliar las capacidades / funcionalidades del Bot.
- SharePoint Online: espacio dedicado a la “intranet corporativa” y aplicaciones ad-hoc.
- Microsoft QnA Maker: como base de conocimiento para intenciones del usuario no identificadas
- Microsoft Teams: como canal de comunicación y colaboración entre los usuarios.

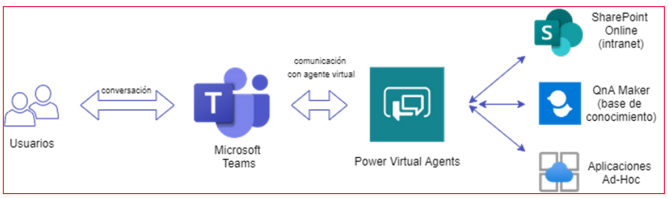


Imagen 1.- Esquema general de la solución.

Microsoft Power Virtual Agents

Después de crear nuestro Bot desde la plataforma de https://powerva.microsoft.com/ debemos establecer los temas sobre los cuales queremos dar respuesta a los usuarios. En nuestro caso vamos a crear los siguientes temas de conversación:

- Noticias: el usuario le pedirá al Bot que le informe de las ultimas noticias corporativas.
- Fichaje de horario laboral: el usuario le pedirá al Bot cerrar el horario laboral del día y este a su vez le conectará a la aplicación de fichajes.
- Preguntas frecuentes: el usuario le preguntará al Bot sobre cómo realizar gestiones dentro de la empresa (Ej.: pedir vacaciones, donde puede encontrar la última nomina, etc.…).

Cada uno de esto temas los crearemos desde la opción lateral Temas -> “+ Nuevo tema” rellenando el formulario con varios datos, así como el Nombre del tema, una breve Descripción y una serie de frases que desencadenaran este tema de conversación.

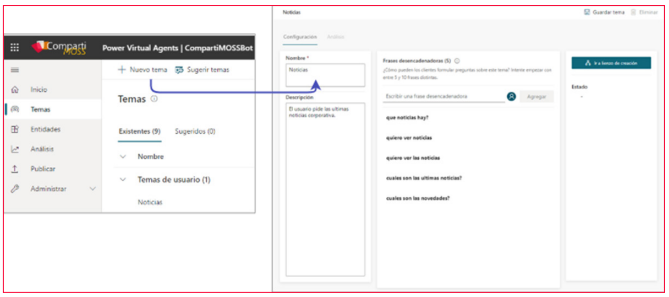


Imagen 2.- Creación de un tema dentro de Power Virtual Agents.

En este caso hemos creado el tema de conversación sobre “Noticias” donde posteriormente debemos hacer el enlace con la fuente de Noticias (colección de sitios de Sitio de comunicación en SharePoint Online). El usuario al pedirle las noticias al Bot, este a su vez iniciara un flujo que recoja los datos desde SharePoint Online y devolver los titulares de las noticias dentro de la conversación. Para ello necesitamos ir a lienzo de creación dentro del tema que acabamos de crear.

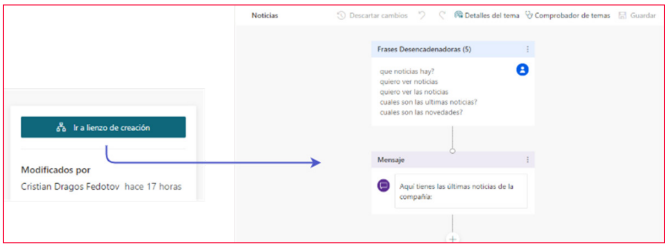


Imagen 3.- Edición del lienzo de un tema dentro de Power Virtual Agents.

Dentro del lienzo añadiremos los pasos necesarios para el dialogo especifico, en este caso hemos añadido una acción en Power Automate que se encarga de traer las noticias desde SharePoint Online y las devuelve al dialogo principal.

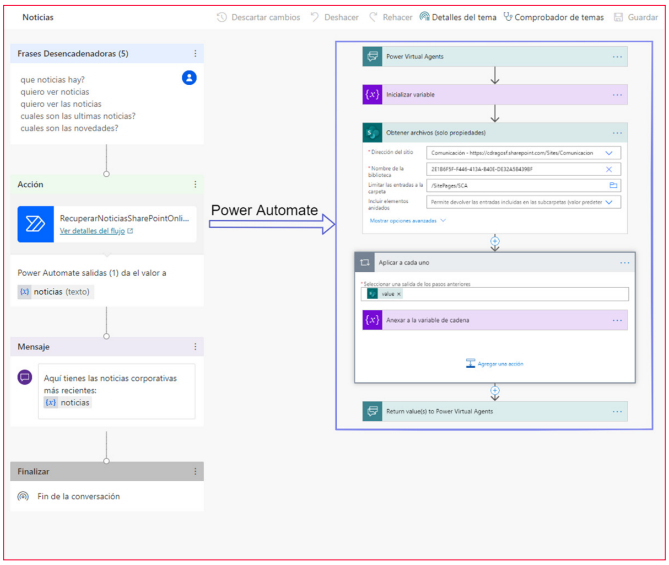


Imagen 4.- Añadir pasos en el lienzo de un tema dentro de Power Virtual Agents.

Por otro lado, debemos crear otro tema de conversación volviendo al panel de temas, pero esta vez para el fichaje de horas. De la misma manera tendremos un flujo de conversación vacío donde debemos añadir varios pasos. En mi caso he realizado una comprobación de los fichajes diarios del usuario actual para el día de hoy. Esta comprobación se ha realizado llamando a una acción de Power Automate que se encarga de comprobar mediante una petición REST el rango de horas que faltan por fichar para el día de hoy.

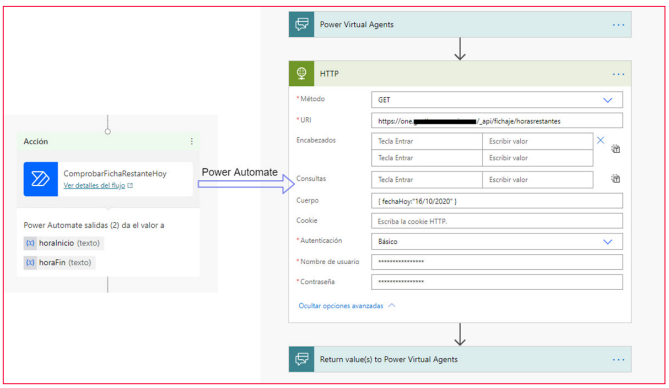


Imagen 5.- Detalle del paso de conversación con flujo en Power Automate.

"nos ayudará a conectar a los usuarios con la información de distintas fuentes gracias al lenguaje natural"

El resultado de este flujo en Power Automate serán dos parámetros, así como hora inicio de fichaje y hora fin de fichaje restantes del día. Como pasos finales para cerrar este flujo de conversación le pediremos al usuario mediante dos preguntas la hora de inicio de su fichaje y la hora fin. Almacenamos esos dos valores y posteriormente los enviamos al sistema de registro de fichajes mediante otra petición REST desde un flujo de Power Automate.

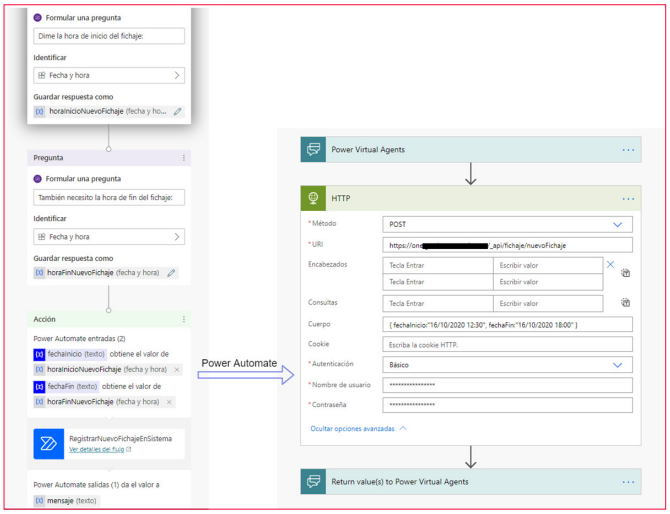


Imagen 6.- Detalle de inserción de datos desde paso de conversación y flujo en Power Automate.

Finalmente cerramos la conversación indicándole al usuario que su fichaje ha sido realizado de manera correcta. Posteriormente mostraremos al usuario un formulario de satisfacción donde puede indicar si le ha sido útil o no la ayuda.

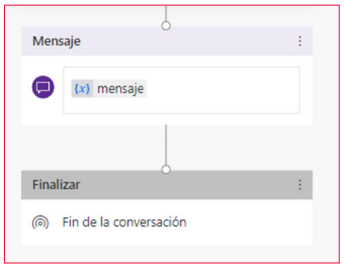


Imagen 7.- Pasos para finalización de conversación con encuesta de satisfacción.

Para las preguntas frecuentes usaremos el "Tema alternativo" del sistema. Este es un tema reservado por el mismo Bot para derivar al usuario en caso de identificar un tema que no coincida con ninguno de los ya configurados (en nuestro caso Noticias y Fichaje). De esta manera derivaremos la petición desconocida del usuario hacia el QnA Maker donde estará la base de conocimiento con las preguntas frecuentes. Para ello debemos ir a la configuración del Bot y añadir el tema alternativo. Una vez creado podremos editarlo de la misma manera que hemos editado los temas, desde el lienzo de personalización.

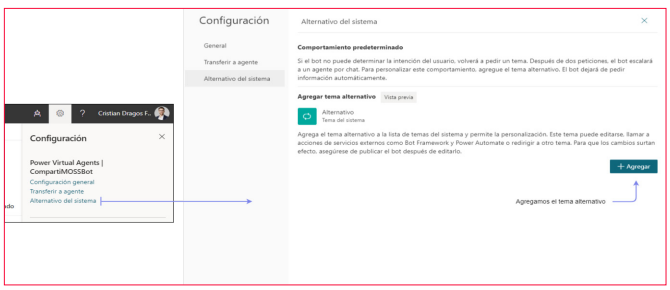


Imagen 8.- Pantalla de creación del tema alternativo del sistema.

Desde el lienzo de personalización añadimos los pasos necesarios para enviar el texto del usuario hacia el servi-

cio de QnA Maker. Al igual que en los anteriores temas lo haremos a través de una acción que desencadena la ejecución de un flujo de Power Automate. Este a su vez se encargará de enviar el texto del usuario hacia QnA Maker y recuperar las respuestas recibidas enviándolas de vuelta a la conversación. (Previamente necesitaremos tener creada una base de conocimiento en QnA Maker a la que poder conectarnos desde Power Automate.)

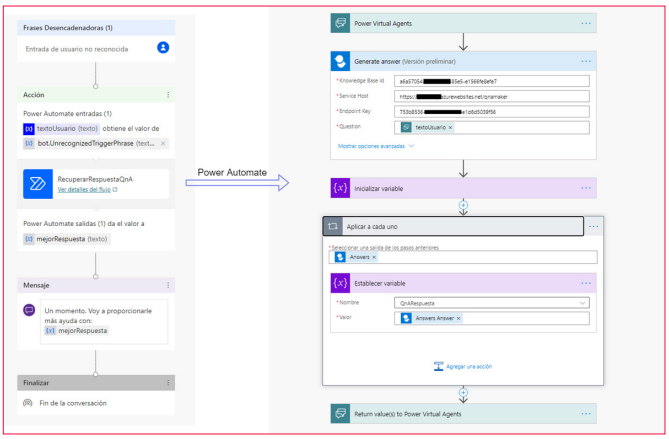


Imagen 9.- Edición del tema alternativo y llamada desde Power Automate a QnA Maker.

Como paso final devolveremos la respuesta al usuario y cerramos la conversación con una encuesta de satisfacción para saber si la respuesta proporcionada por QnA Maker ha sido de ayuda.

Microsoft Teams

Una vez finalizada la personalización del Bot debemos habilitar el canal de comunicación de Microsoft Teams y realizar una petición de publicación del Bot en la organización para que todos los usuarios lo puedan utilizar.

Para ello debemos ir al menú lateral de Power Virtual Agents, en la pestaña “Administrar”, submenú “Canales”. Dentro de esta ventana podremos modificar los detalles del Bot, así como el color y el icono de este al igual que la descripción. También podremos realizar la petición de aprobación del Bot como aplicación disponible en toda la organización.

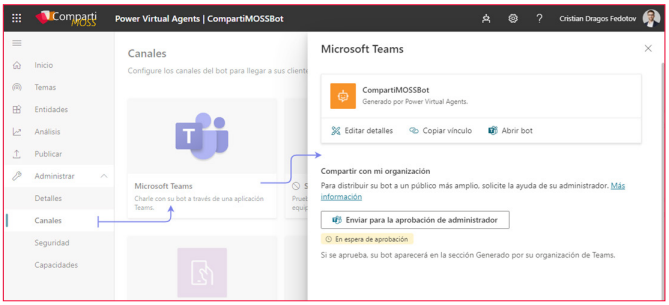


Imagen 10.- Publicación del Bot en el canal de Microsoft Teams.

A su vez los usuarios administradores desde el centro de administración de Microsoft Teams recibirán una notificación en el cuadro de aplicaciones para autorizar la distribución del Bot dentro de la Organización.

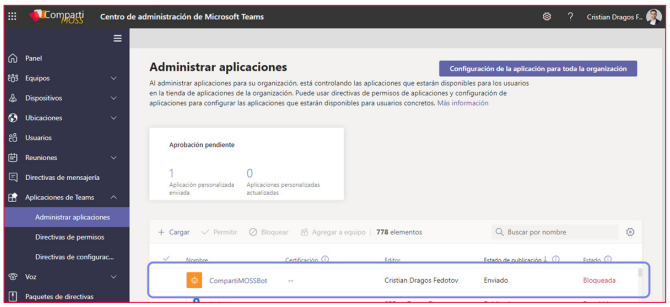


Imagen 11.- Aprobación del Bot en el centro de administración de Microsoft Teams.

"Desde el lienzo de personalización añadimos los pasos necesarios para enviar el texto del usuario hacia el servicio de QnA Maker"

Una vez aprobada esta petición, podremos encontrar nuestro Bot dentro de la pestaña de Apps de la interfaz de Microsoft Teams y así poder iniciar una nueva conversación.

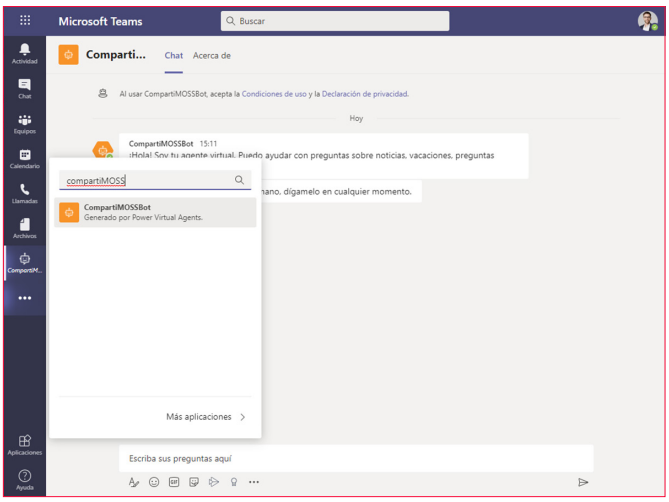


Imagen 12.- Buscar y Agregar el Bot como pestaña dentro de Microsoft Teams.

Finalmente podremos probar los tres casos de uso que habíamos expuesto:

Petición de las ultimas noticias:

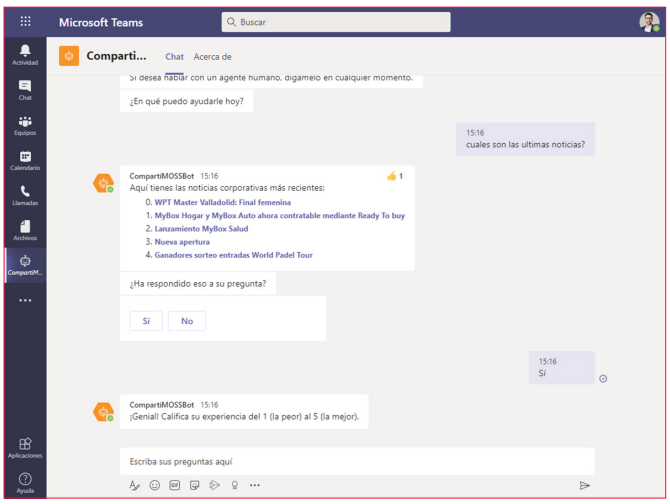


Imagen 13.- Petición de noticias desde la conversación con el Bot en Microsoft Teams.

Realizar fichaje de horario laboral diario:

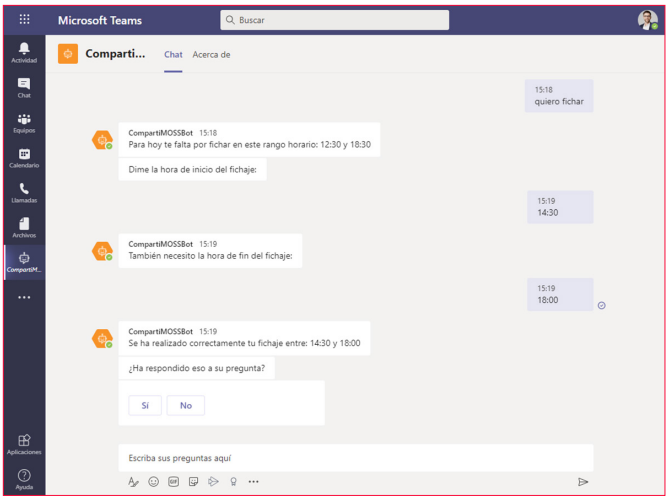


Imagen 14.- Realización de fichaje desde la conversación con el Bot en Microsoft Teams.

Derivación y resolución de preguntas frecuentes:

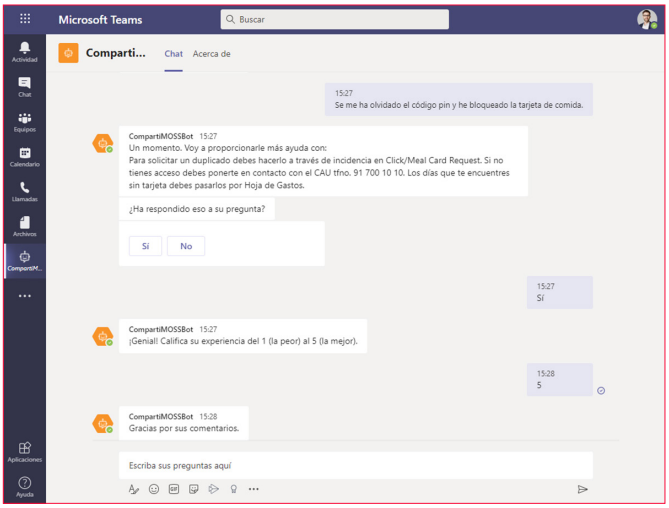


Imagen 15.- Resolución de preguntas frecuentes desde el Bot en Microsoft Teams.

Como añadido Power Virtual Agents nos permite una infinidad de posibilidades, así como autenticar nuestras peticiones, utilizar diferentes canales de comunicación para el Bot (Facebook, Correo, DirectLine, Cortana, etc....), en un futuro próximo podremos usar AdaptiveCards para enviar mensajes al usuario y un largo etc. en continua evolución.

Así que visto todo esto ya no hay excusa para no empezar a utilizar los agentes virtuales que nos ofrece Microsoft en este nuevo formato. Aquí, las necesidades de nuestros usuarios y la imaginación son nuestros aliados a la hora de crear nuevos casos de uso o temas de conversación del Bot y cada vez podamos aplicar sus funcionalidades.

CRISTIAN DRAGOS FEDOTOV

Office 365 & SharePoint Solutions Architect en Everis

cristianfedotov@gmail.com

https://www.linkedin.com/in/cristianfedotov/

08

Power Virtual Agents en Microsoft Teams

En el presente artículo nos centraremos en la utilización de Power Virtual Agents en Microsoft Teams.

Hace ya algunos meses desde que se desato el problema del Covid 19 que vengo buscando diferentes alternativas para seguir con las comunicaciones efectivas de cara a los colaboradores de las organizaciones, pero que estas a su vez sean de forma virtual y automática. Por tal motivo venias viendo el tema de asistentes virtuales como un workaround para que dicha comunicación fuera efectiva e instantánea.

Al comienzo utilizamos Microsoft Bot Framework para desarrollar los asistentes virtuales teniendo diferentes recursos de Azure y luego se asociaba con un Bot en Teams y se publicaba para la organización. Esto requería del equipo de desarrollo para la implementación del Bot y conocimientos de Azure para el manejo de recursos

Después comenzamos a utilizar Power Virtual Agents dado que son una forma rápida y efectiva de crear un Asistente Virtual con poco esfuerzo en un entorno visual sin necesidad de código. Esto nos permitía que las personas de las áreas no tecnológicas pudieran también participar de los diferentes desarrollos. Este formato tenía un pequeño inconveniente de cara a las organizaciones y era el costo de licenciamiento que para algunas organizaciones resultaba elevado y en la incertidumbre provocada por la pandemia no estaban dispuestos a invertir en eso.

Hace unas semanas Microsoft liberó Power Virtual Agents para el uso exclusivo en Teams, esto significa que puedo utilizar todo el potencial de Power Virtual Agents gratis con la restricción de que tiene que ser desarrollado y utilizado en Microsoft temas.

En este articulo mostraremos como en pocos pasos puedo generar un Asistente Virtual de Preguntas frecuentes para mi organización, como ejemplo utilizaremos las preguntas frecuentes de MS Teams que resultan ser de gran ayuda para aquellos que recién comienzan a utilizarlo.

Creando Power Virtual Agents en Teams

Primero que nada, debemos agregar la aplicación de Power Virtual Agents en Microsoft Teams:

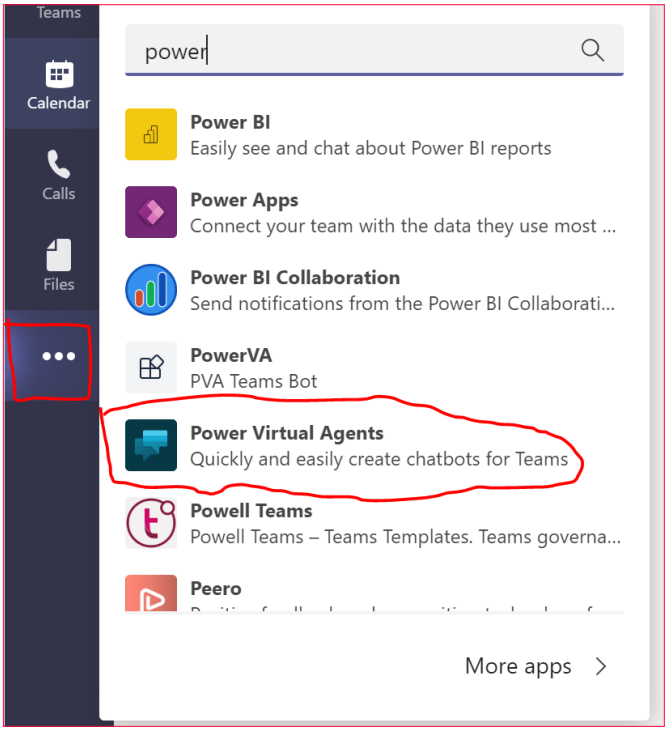


Imagen 1.- Añadiendo Power Virtual Agents en Microsoft Teams.

Una vez agregada a nuestras Apps, vamos a la pestaña de Chatbots y creamos uno nuevo haciendo clic en “New chatbot”:

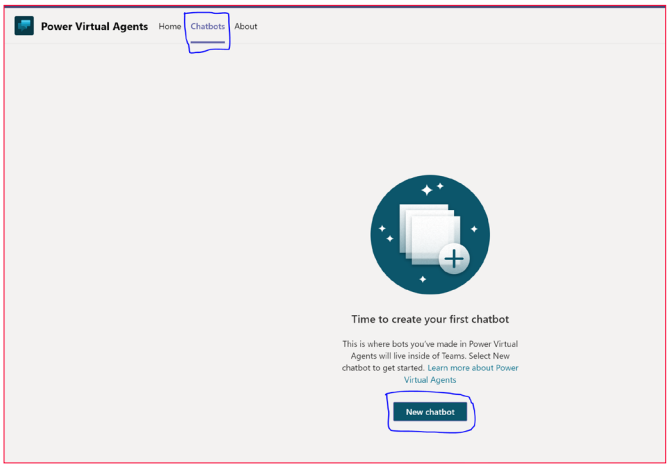


Imagen 2.- Creando un primer chatbot.

Una vez en allí seleccionamos el Team al cual queremos agregar nuestro Bot y luego “Continue” (Nota: Esto se debe a que solo se puede agregar PVA a Teams de Microsoft Teams).

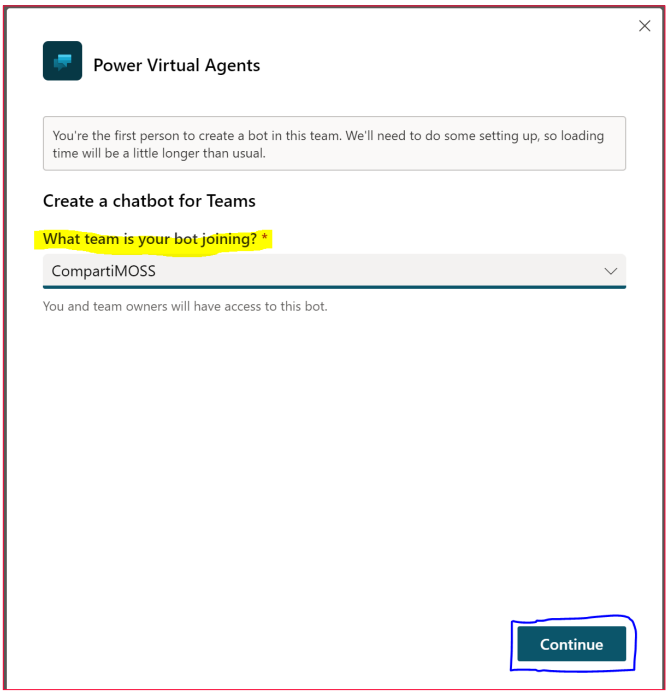


Imagen 3.- Seleccionado el Team al que vamos a añadir el chatbot.

Nos aparecerá la siguiente pantalla donde debemos esperar a que todo quede listo:

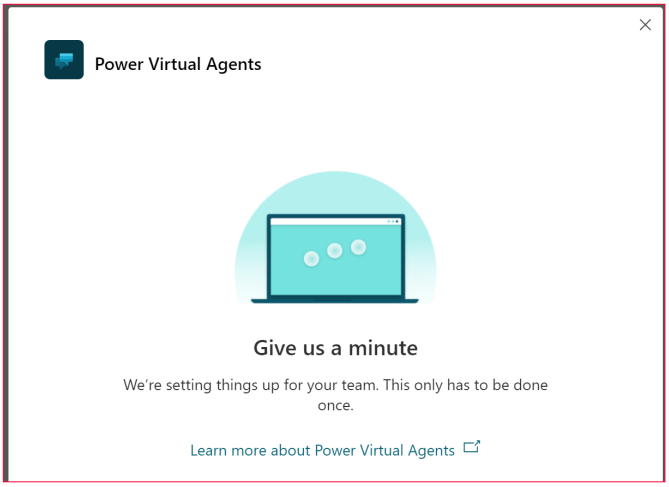


Imagen 4.- Pantalla de transición a la creación del Chatbot.

Nota: Podemos seguir realizando otras tareas hasta que termine, Teams nos avisara cuando haya finalizado.

Una vez finalizada la creación de Bot debemos darle un nombre, seleccionar el idioma y luego "Create"

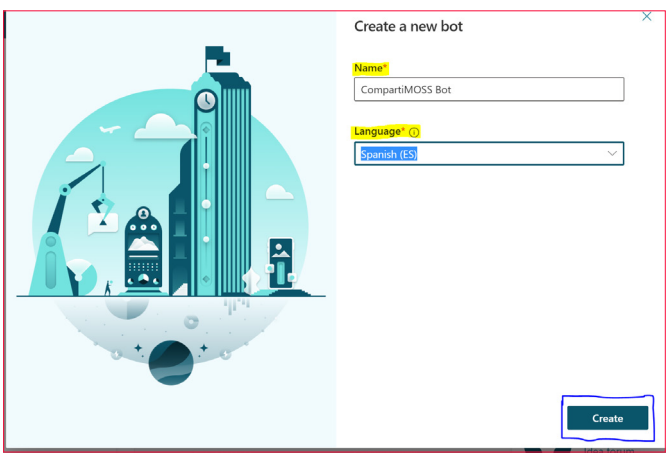


Imagen 5.- Parámetros iniciales de creación del chatbot.

Imagen 6.- Pantalla de transición para la creación del chatbot.

Una vez finalizado estaremos en la siguiente pantalla listos para comenzar a trabajar con el Bot

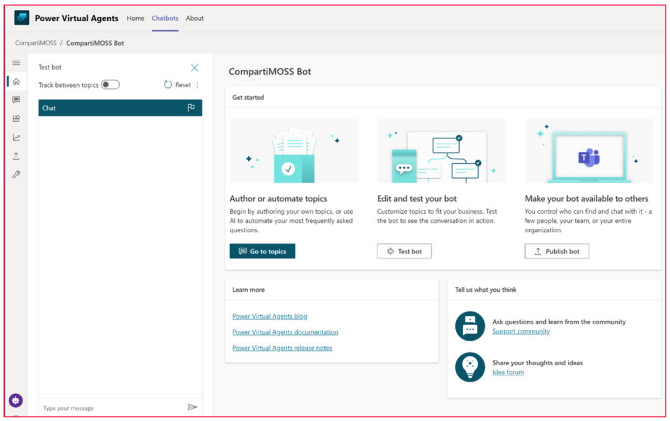


Imagen 7.- Diseñador del Chatbot en Microsoft Teams.

Ahora debemos agregarle nuestras preguntas frecuentes, vamos a "Go to topics"

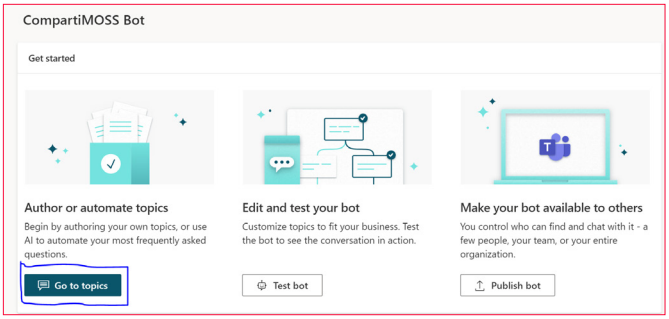


Imagen 8.- Acceso a la creación de temas (topics) para el Chatbot.

En la siguiente pantalla debemos ir a "Suggested" y luego a Get started:

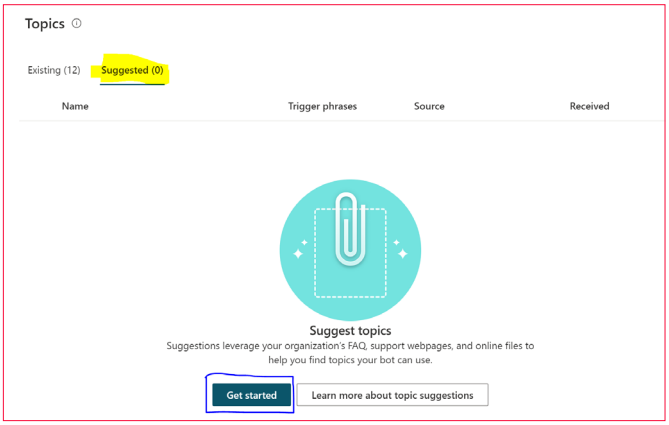


Imagen 9.- Sección Suggested en la configuración de temas para el chatbot.

"una forma rápida y efectiva de crear un Asistente Virtual con poco esfuerzo en un entorno visual sin necesidad de código"

En esta sección debemos indicar el origen de nuestras preguntas, este caso utilizamos el sitio de preguntas frecuentes de Teams: Ayuda y aprendizaje de Microsoft Teams - Soporte técnico de Microsoft

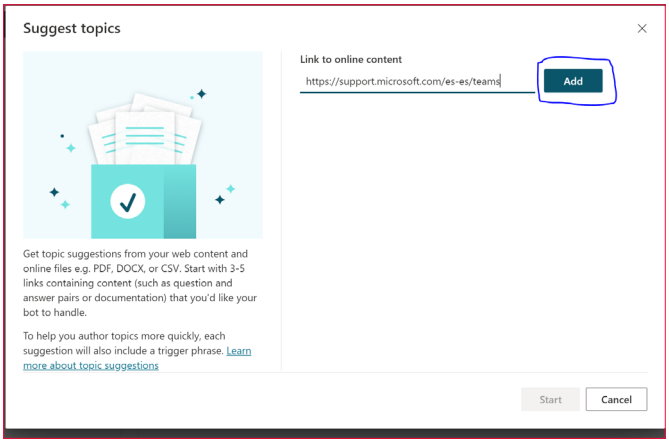


Imagen 10.- Configuración del origen de información para los temas sugeridos.

Hacemos clic en "Start". Una vez procesado nos sugerirá diferentes consultas y debemos ir agregando las que consideremos correctas

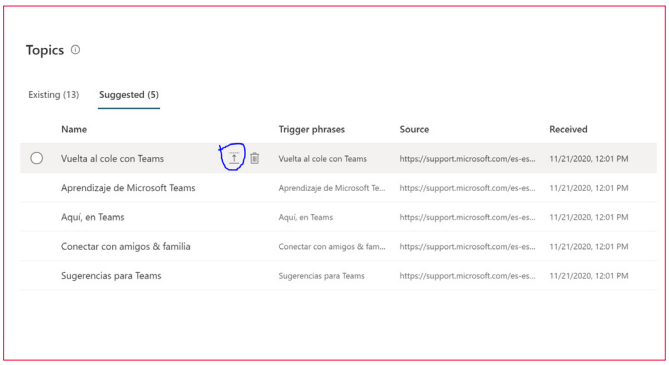


Imagen 11.- Selección de los temas sugeridos a utilizar.

Una vez agregada los topics debemos habilitarla "turn on"

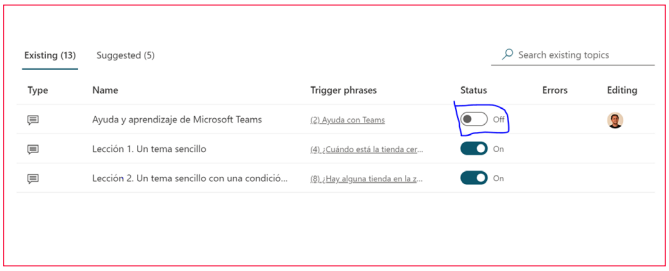


Imagen 12.- Habilitando temas.

Ya con el topic habilitado podemos probar nuestro chatbot y verificar que funcione correctamente directamente desde el cuadro de texto en la parte inferior del chatbot.

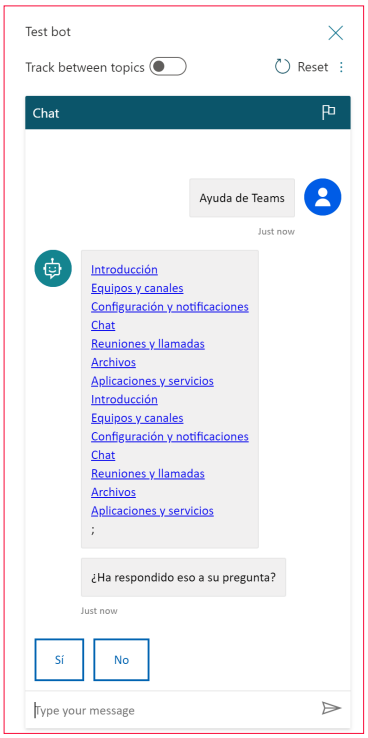


Imagen 13.- Probando el chatbot.

Por último, debemos publicarlo para que los demás integrantes del grupo de teams seleccionado al inicio lo puedan utilizar

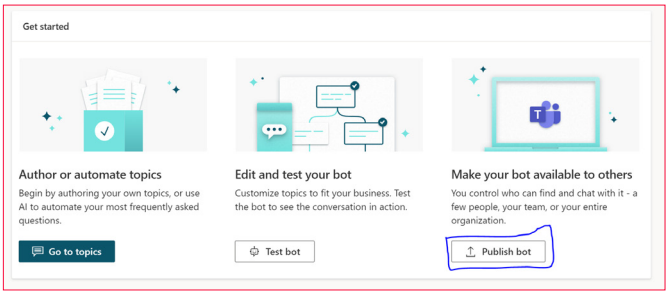


Imagen 14.- Publicando el chatbot.

Vamos a publish:

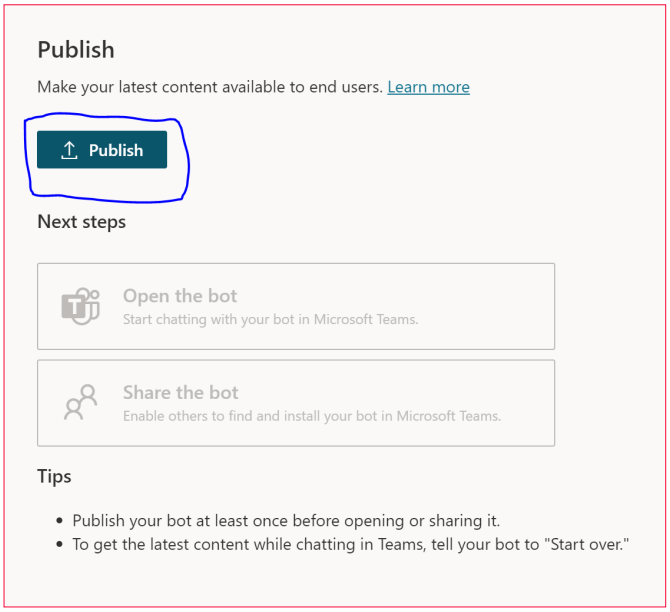


Imagen 15.- Acción de publicar el chatbot.

Confirmamos la publicación del chatbot

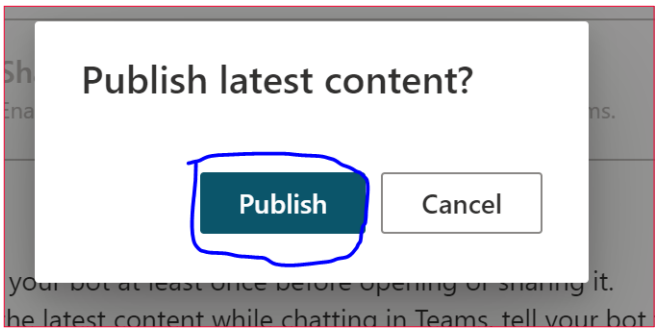


Imagen 16.- Confirmación de publicación del chatbot.

Luego también podemos agregarlo como una app en nuestro Teams

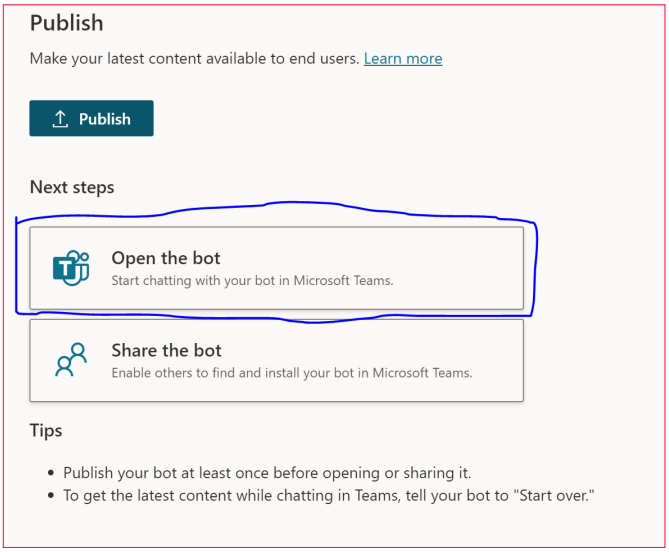


Imagen 17.- Agregando el chatbot como una App de Teams.

"también podemos agregarlo como una app en nuestro Teams"

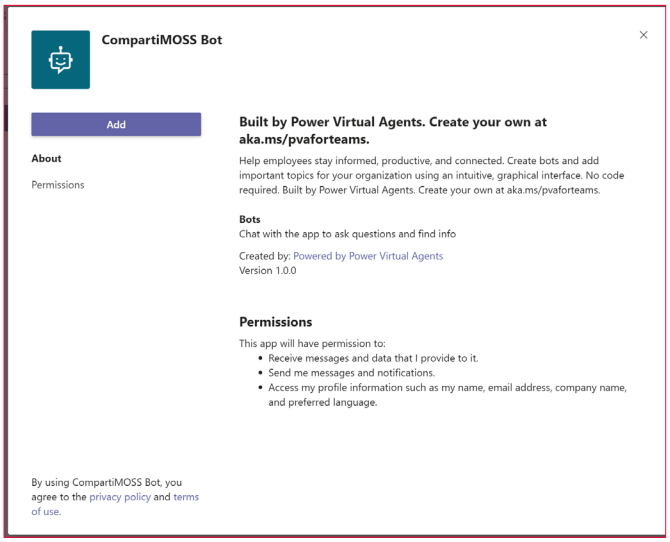


Imagen 18.- Añadiendo el chatbot como una App.

Conclusión

Esta forma de crear chatbot y agregarle preguntas frecuentes resulta ser super rápida y sencilla, brindando respuestas y soluciones de forma rápida y ágil a los colaboradores. Estos chatbots pueden se extender utilizando Power Automate y sus conectores a Cognitive Services y diferentes servicios permitiéndonos realizar innumerables automatismos.

ALEX ROSTÁN
Business Applications MVP
Country Manager/ AI & Smarts Applications / Arkano Paraguay
rostanker@msn.com
Twitter: @rostanker
https://www.linkedin.com/in/alexrostan/
http://www.arkanosoft.com

17 Entrevista José Manuel Alarcón

Me llamo José Manuel Alarcón Aguín, y soy de Vigo (Pontevedra).

En el año 2000 fundé la empresa, Krasis, que está especializada en eLearning. En la actualidad tenemos dos áreas de negocio: la plataforma de eLearning SELF, uno de los LMS más avanzados del mercado para empresas de formación y para formación empresarial, y campusMVP.es que es quizá el proyecto de referencia en formación online en español para desarrolladores.

Fui articulista profesional durante muchos años, escribiendo para revistas en papel, mientras existieron, como RPP (Revisa Profesional para Programadores), PCWorld, iWorld, Windows TI Magazine... También soy autor de unos cuantos libros, y con los años he escrito literalmente miles de artículos en la Web.

Desde 2004 hasta la actualidad, Microsoft me ha reconocido todos los años como MVP (Most valuable Professional) en el área de desarrollo Web. Debo de ser de los más antiguos.

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

Yo, en realidad, estudié ingeniería industrial y, de hecho, me especialicé en máquinas y mecanismos. Pero a mediados de los años 90 descubrí Internet, y fue una revelación: aluciné con las posibilidades que había y vi que iba a ser el futuro. Así que dejé a un lado para siempre la grasa y los engranajes, y los cambié por los bits y los bytes, que son infinitamente más maleables.

Generalmente, cuando me meto en algo me meto a fondo, así que estudié (soy autodidacta) y practiqué un montón de manera muy intensa, y enseguida me puse a desarrollar en serio. Primero pequeños encargos para conocidos en la Universidad, y luego ya de manera profesional, consiguiendo mucha experiencia en poco tiempo y en un momento en el que la demanda era brutal. He hecho casi de todo: desde pequeños sitios web allá por los 90, hasta frameworks de desarrollo, pasando por gestores de contenidos, sistemas de marketing online, simuladores, programas de gestión, utilidades de todo tipo...

¿Cuáles son tus principales actividades tecnológicas hoy en día?

En la actualidad y desde hace años debo compaginar mi pasión técnica con la gestión empresarial y la dirección de proyectos, por lo que no programo todo lo que me gustaría. Aún así, como soy muy mal empresario, me meto demasiado en el día a día de muchas cosas en la empresa, y desarrollo y defino técnicamente mucho más de lo que debería.

También, en mi tiempo libre, me dedico a investigar nuevas tecnologías y a probarlas. Al fin y al cabo me gusta estar siempre al día en todo lo que hay, aunque cada vez es más complicado (yo diría que imposible). También desarrollo proyectos propios que a veces se acaban convirtiendo en parte de nuestros productos, o bien en proyectos Open Source. Aunque, en este caso, no lo hago todo lo que me gustaría.

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

En la empresa, al ser pequeña, hago un poco de todo: planificación, gestión de proyectos, trabajo con los autores, revisión de contenidos, escribir para el blog, gestión de

marketing... Como digo, soy muy mal empresario pero es lo que hay.

¿Cuáles son tus hobbies?

A mí me gusta mucho aprender. Así que leo mucho, veo documentales, me documento... Aparte de la ciencia, me interesa mucho la sociología y la psicología, pero también la historia. Como ves, el comportamiento humano en general. En los últimos años he desarrollado un interés especial por Roma y en particular por la república romana y sus logros durante los tres siglos antes de nuestra era. Pero me parecen interesantes un montón de cosas. Lo cual es un problema a veces porque uno no sabe a qué atender, pero tiene la ventaja de que es casi imposible aburrirse.

Aparte de eso disfruto mucho de la música, la fotografía y el cine, del que me encanta además su parte más técnica de imagen, producción y dirección. Y por supuesto las series de televisión. Todos los días veo un capitulito de alguna.

¡Ah, y me encanta probar y descubrir software nuevo!

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

Una de las tendencias existentes que no me parecen un "bluff" como sí lo son otras, es la introducción cada vez mayor de sistemas basados en IA y Deep Learning, en todos los aspectos de las empresas y de la sociedad. Estamos aún empezando, pero en los próximos años será ubicuo y transparente, es decir, aunque muchos usuarios no lo van a ver, estará ahí. Y es precisamente por eso por lo que creo que va a tener una adopción grande y rápida. Pero se necesitarán técnicos especializados que dominen bien los conceptos subyacentes y no se limiten a usar un par de APIs que le dan hechas (que también están muy bien, ojo).

Y luego, en el mundo del desarrollo de software veo dos tendencias opuestas, pero totalmente relacionadas, que son: la mayor complejidad por un lado y la simplificación por otro. Una mayor complejidad porque cada vez es más difícil, no ya profesionalizarse como desarrollador, sino incluso seguir el ritmo a las novedades y a las tecnologías que van apareciendo. Pero por otro lado existe una sencillez cada vez mayor para crear software, gracias a varias tendencias que han explotado en los últimos años como la disponibilidad brutal que hay ahora de Open Source, las herramientas Low-Code o No-Code, herramientas online como Zapier o Airtable (que no tienen que ver funcionalmente, pero a efectos de lo que hablo, están relacionadas), y las plataformas Serverless como Azure Functions o CloudFlare Workers. Todas ellas conducen a que gente con conocimientos básicos técnicos o de programación pueden crear en poco tiempo aplicaciones de calidad y escalables.

En mi opinión, estas dos tendencias antagónicas van a dar lugar en la próxima década a dos modos muy diferentes de hacer programación. Si me permites la expresión trillada, crearán una polarización del mercado laboral del desarrollo. Por un lado, tendremos gente muy especializada que dominará una determinada tecnología y que tendrán salarios altos y poder en las empresas, como pasa ahora con algunos desarrolladores, pero con más barreras de entrada para formarse por la dificultad y la entrega necesarias. Pero por otro veremos que mucha gente no técnica podrá crear sus propias soluciones a medida y que habrá programadores con bajo nivel de preparación haciendo muchos pequeños desarrollos en las empresas basados en "juntar piezas", integraciones o crear pequeños programas, que serán más que suficientes. Estos perfiles no estarán tan bien pagados y tendrán menos necesidad de formación y menos estabilidad laboral, pero darán trabajo a mucha gente. Estoy hablando de programación pura y dura, ojo, no de otro tipo de perfiles.

De todos modos, es muy probable que esté equivocado, pero yo lo vislumbro así.

JOSÉ MANUEL ALARCÓN
Director de eLearning – Krasis/CampusMVP
@jm_alarcon
https://www.jasoft.org

Todos los *secretos* del desarrollo de software para Microsoft Office 365

Un nuevo libro de **Gustavo Velez** que expone cómo desarrollar software para Microsoft Office 365: Exchange, SharePoint, Word, Excel, Teams, etc.

El libro evoluciona con Office 365, de manera que los lectores reciben mensualmente las **últimas actualizaciones** de Microsoft y nuevo contenido.

Encuéntrelo en

https://guitaca.com

güitaca publishers

15

Procesando formularios con Form Recognizer y Azure Functions

Es evidente que poco a poco va desapareciendo el papel físico en cuanto a la gestión de facturas y recibos, pero estamos muy muy lejos de conseguirlo aún. Y aunque lo consigamos a medio plazo, veo muy difícil que no sigamos recibiendo Pdf's o words con el detalle de nuestras compras. Esto provoca un coste incalculable de dinero y de tiempo en las Empresas, no solo ya para los departamentos financieros y de administración, sino para los propios empleados que tienen que perder bastante tiempo procesando facturas en aplicaciones de gastos que en muchos casos son muy complicadas de procesar.

Nuestro objetivo al terminar este artículo es comprobar si Form Recognizer nos puede ayudar a automatizar los procesos de volcado de formularios y facturas a nuestras Bases de datos del CORE financiero, aunque vamos a comprobar que podemos aplicar el servicio para lectura de formularios de cualquier tipología.

Nuestro objetivo, automatización e integración

Para situarnos un poco más, y antes de adentrarnos en destripar el servicio de Azure Form Recognizer que sé que tenemos todos muchas ganas, vamos a plantearnos el escenario de uso y así nos será mucho más sencillo llegar al éxito. Partiendo del problema de que contabilizar las facturas de nuestros proveedores puede ser un trabajo muy costoso para nuestra compañía, vamos a construir un proceso que automatice lo máximo posible el insertar nuestra facturas en el ERP.

El proceso se va a basar en:

- Usaremos Azure Functions para integrar un Blob Storage de Ficheros, donde dejaremos nuestras facturas a procesar, con el servicio cognitivo de Form Recognizer.
- Con Form Recognizer procesaremos una lectura automática de las facturas, obteniendo un fichero JSON controlado en base a un entrenamiento personalizado.
- Utilizaremos una Logic App para insertar este JSON en un Commno Data Service, que hará las veces de ERP en nuestro ejemplo.

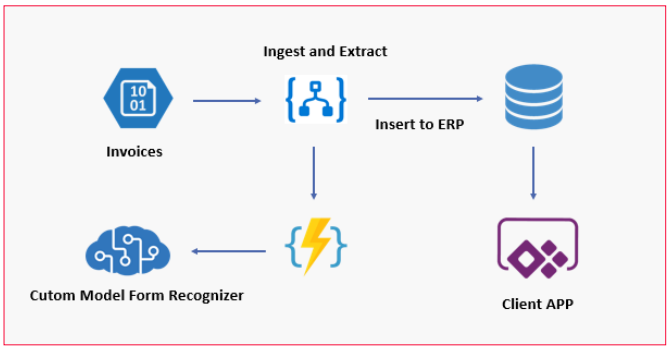


Imagen 1.- Proceso de extracción de facturas.

Form Recognizer, servicio Cognitivo de la familia VISION

Desde el pasado mes de Julio tenemos 100% disponible un nuevo servicio en General Available como Form Recognizer, con el cual vamos a poder procesar formularios personalizados, recibos o tarjetas de visita entre otros modelos. Este servicio se apoya en modelos pre-compilados que nosotros mismos podemos extender con entrenamiento, y en la lectura de ficheros por OCR para poder extraer en formato JSON los datos de los formularios o de los recibos, que se definen como pares "Clave -valor".

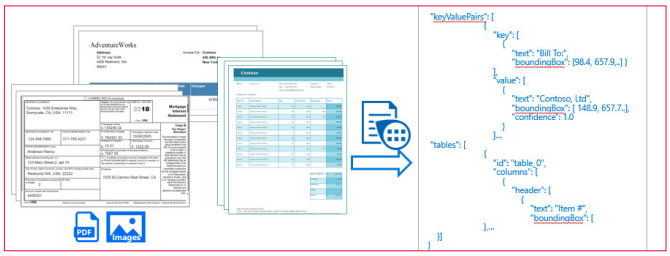


Imagen 2.- Extracción clave valor.

Por ahora el formato de los ficheros que puede procesar el servicio son PDF y ficheros del tipo Imagen, aunque están trabajando en incluir más tipos de ficheros. Para nuestro reto de hoy nos vamos a centrar en trabajar con "Formularios personalizados" y en como debemos "entrenar nuestro servicio" para poder tener una extracción exacta.

Primer contacto con el API y utilización de la librería cliente en C#

Dado que vamos a utilizar un Azure Function para procesar los formularios, deberemos conocer la librería cliente que

podemos utilizar con C#, JavaScript o Python para construir nuestro código serverless, pero en este caso los ejemplos están en C#. Antes de entrar a construir con C#, es bueno que entendamos que nos devuelve el API con los modelos pre-compilados, y con la lectura de OCR básica.

Los siguientes fragmentos de código que vamos a ir nombrando los podemos localizar en el siguiente repositorio https://github.com/shmancebo/FormRecognizer

Para empezar a trabajar con el servicio desde código, a parte de dejar el servicio creado en el portal de Azure (omitimos este paso dado que es una creación muy básica como cualquier otro servicio cognitivo), debemos crearnos una librería de clases en Net Core por ejemplo, y añadir el siguiente paquete NuGet:

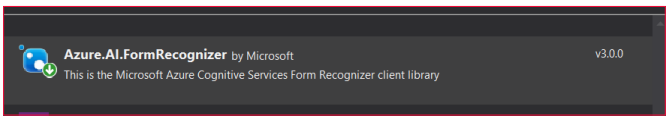


Imagen 3.- Paquete Nuget.

Si accedemos al ejemplo de Github veremos un proyecto Shared en el que encontramos una clase FormRecognizerService.cs , que nos sirve para comunicarnos con el servicio de Form Recognizer. Lo primero es construir la conexión con el servicio, y eso se realiza en el constructor:

```
public FormRecognizerService(string endpointService, string keyService)
{
    var credentials = new AzureKeyCredential(keyService);
    _frClient = new FormRecognizerClient(new Uri(endpointService), credentials);
}
```

Imagen 4.- Conectando con el servicio.

Para poder conectar con el servicio le debemos pasar la url del servicio y la clave de suscripción, y ambos datos los encontramos en el Portal de Azure entrando en la configuración de claves del servicio creado. Una vez conectado el servicio, vamos a hacer un método que permita analizar un formulario sin "moledo personalizado", y así podremos ver que nos devuelve de base el servicio.

```
public async Task<FormPageCollection> AnalyzeFormFromStream(Stream form)
{
    FormPageCollection collection = await _frClient.StartRecognizeContent(form).WaitForCompletionAsync();
    return collection;
}
```

Imagen 5.- Analizando un formulario.

Si analizamos el código anterior, vemos que el servicio espera un Stream del formulario a procesar, y con esto invocando al método "StarRecognizeContent", obtenemos una evaluación completa. Para probar si esto nos devuelve un "clave-valor" válido para poder trabajar, vamos a utilizar la consola del ejemplo del respositorio TestFR, que invoca a este método AnalyFromFromStream que acabamos de definir.

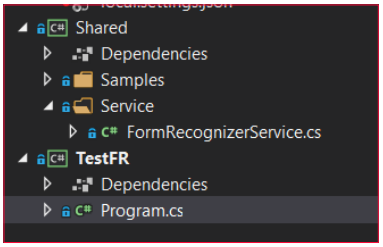


Imagen 6.- Console test.

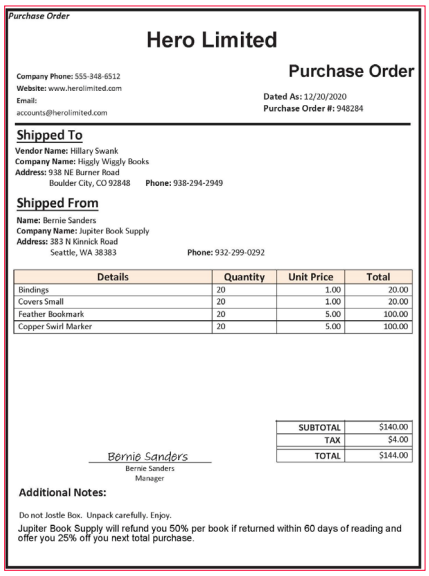
Si lanzamos la consola, nos va a pedir si queremos analizar un formulario o un recibo, seleccionamos un formulario, y dejamos en blanco el "id de modelo", ya que así vamos a forzar que pase por el método que acabamos de construir.

```
Console.WriteLine("Introduzca el modelo de entrenamiento:");
var model = Console.ReadLine();
if (string.IsNullOrEmpty(model))
{
    var formContent = await frClient.AnalyzeFormFromStream(form);
    PrintForm(formContent);
}

else
{
    var formContent = await frClient.AnalyzeCustomFormFromStream(form, model);
    PrintCustomForm(formContent);
}
```

Imagen 7.- Código console.

Vamos a asegurarnos que el código de la consola apunta al formulario de ejemplo "Form_1.jpg", que tiene un formulario como el de la imagen siguiente, y que contiene una orden de pedido:



Purchase Order

Hero Limited

Company Phone: 555-348-6512
Website: www.herolimited.com
Email: accounts@herolimited.com

Purchase Order

Dated As: 12/20/2020
Purchase Order #: 948284

Shipped To

Vendor Name: Hillary Swank
Company Name: Higgly Wiggly Books
Address: 938 NE Burner Road
Boulder City, CO 92848 Phone: 938-294-2949

Shipped From

Name: Bernie Sanders
Company Name: Jupiter Book Supply
Address: 383 N Kinnick Road
Seattle, WA 38383 Phone: 932-299-0292

Details	Quantity	Unit Price	Total
Bindings	20	1.00	20.00
Covers Small	20	1.00	20.00
Feather Bookmark	20	5.00	100.00
Copper Swirl Marker	20	5.00	100.00

SUBTOTAL	$140.00
TAX	$4.00
TOTAL	$144.00

Bernie Sanders
Bernie Sanders
Manager

Additional Notes:

Do not Jostle Box. Unpack carefully. Enjoy.
Jupiter Book Supply will refund you 50% per book if returned within 60 days of reading and offer you 25% off you next total purchase.

Imagen 8.- Orden de pago.

Si ejecutamos la consola con el formulario anterior, vamos a obtener un resultado muy parecido al siguiente:

```
C:\Program Files\dotnet\dotnet.exe
Quiere procesar un formulario (1) o un recibo (2)
1
Introduzca el modelo de entrenamiento:

Form Page 1 has 54 lines.
    Line 0 has 2 words, and text: 'Purchase Order'.
    Line 1 has 2 words, and text: 'Hero Limited'.
    Line 2 has 3 words, and text: 'Company Phone: 555-348-6512'.
    Line 3 has 2 words, and text: 'Purchase Order'.
    Line 4 has 2 words, and text: 'Website: www.herolimited.com'.
    Line 5 has 1 word, and text: 'Email:'.
    Line 6 has 3 words, and text: 'Dated As: 12/20/2020'.
    Line 7 has 1 word, and text: 'accounts@herolimited.com'.
    Line 8 has 4 words, and text: 'Purchase Order #: 948284'.
    Line 9 has 2 words, and text: 'Shipped To'.
    Line 10 has 4 words, and text: 'Vendor Name: Hillary Swank'.
    Line 11 has 5 words, and text: 'Company Name: Higgly Wiggly Books'.
    Line 12 has 5 words, and text: 'Address: 938 NE Burner Road'.
    Line 13 has 4 words, and text: 'Boulder City, CO 92848'.
    Line 14 has 2 words, and text: 'Phone: 938-294-2949'.
    Line 15 has 2 words, and text: 'Shipped From'.
    Line 16 has 3 words, and text: 'Name: Bernie Sanders'.
    Line 17 has 5 words, and text: 'Company Name: Jupiter Book Supply'.
    Line 18 has 5 words, and text: 'Address: 383 N Kinnick Road'.
    Line 19 has 3 words, and text: 'Seattle, WA 38383'.
    Line 20 has 2 words, and text: 'Phone: 932-299-0292'.
    Line 21 has 1 word, and text: 'Details'.
    Line 22 has 1 word, and text: 'Quantity'.
    Line 23 has 2 words, and text: 'Unit Price'.
    Line 24 has 1 word, and text: 'Total'.
```

Imagen 9.- Primer análisis.

Si veis la consola de salida, el servicio hace una lectura de la imagen, y saca una lectura del mismo basado en líneas del documento, y de estas obtiene las palabras. Esto quiere decir que sin hacer nada ya puedo sacar palabras clave

y valores, con lo que por ejemplo alimentar un índice de búsqueda. Imaginemos que “Purchase Order” fuera una palabra reservada para posicionar este tipo de facturas, pues solo con este ejemplo ya podríamos localizar todas las ordenes de pago de este tipo. Aunque claro está para nuestro reto de insertar este formulario de orden de pedido en el ERP, este modelo de datos no nos vale, porque nuestra Azure Function quedaría con un código muy muy complejo, y procesando demasiadas líneas y palabras que pueden variar.

Entrenando el formulario con un modelo personalizado

Para poder mejorar la extracción de “clave -valor”, vamos a entrenar el formulario anterior, de cara a obtener un código mucho más limpio de mapeo. Para ello debemos subir a un blob storage el contenido de la carpeta “Formularios para entrenar” de la solución de Visual Studio.

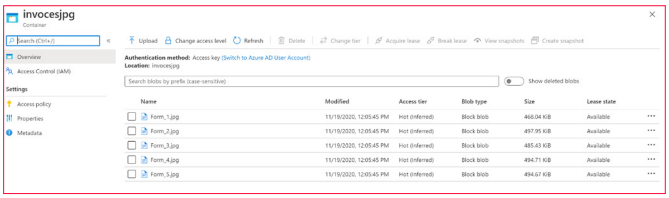


Imagen 10.- Blob storag.e

Una vez subidos los formularios, debemos generar un “Shared Access signature” de lectura, para que nuestro API de Form Recognizer pueda llegar a los formularios.

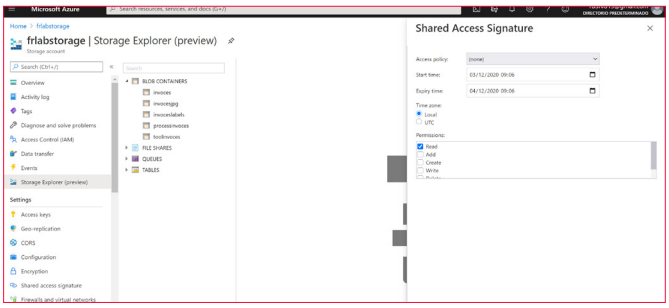


Imagen 11.- Dando permisos a Form Recognizer.

Una vez generada la firma de acceso, nos quedamos con el campo Uri que debe ser del tipo

https://frlabstorage.blob.core.windows.net/invocesjpg?sp=rl&st=2020-12-03T08:06:56Z&se=2020-12-04T08:06:56Z&sv=2019-12-12&sr=c&sig=K6tcxgSEYOneKJFBeMyKk0KyC7%2BBnu816S%2Brx5VBXwk%3D

Ahora vamos a ejecutar desde el Postman por ejemplo la siguiente petición:

```
POST
{Url form recognizer}/formrecognizer/v2.0/custom/models
BODY
{
    “source”:  “https://frlabstorage.blob.core.windows.net/invocesjpg?sp=rl&st=2020-12-03T08:06:56Z&se=2020-12-04T08:06:56Z&sv=2019-12-12&sr=c&sig=K6tcxgSEYOneKJFBeMyKk0KyC7%2BBnu816S%2Brx5VBXwk%3D”,
  “sourceFilter”: {
    “includeSubFolders”: false
  },
  “useLabelFile”:false
}

HEADERS
Ocp-Apim-Subscription-Key : {Key Form Recognizer}
Content-Type : Application/json
```

Si todo es correcto nos va devolver la petición un resultado “201” tal y como vemos en la siguiente imagen, pero deberemos coger de la cabecera Location la URL que nos devuelve, y realizar una petición GET a esa url, donde veremos si el entrenamiento ha finalizado o no.

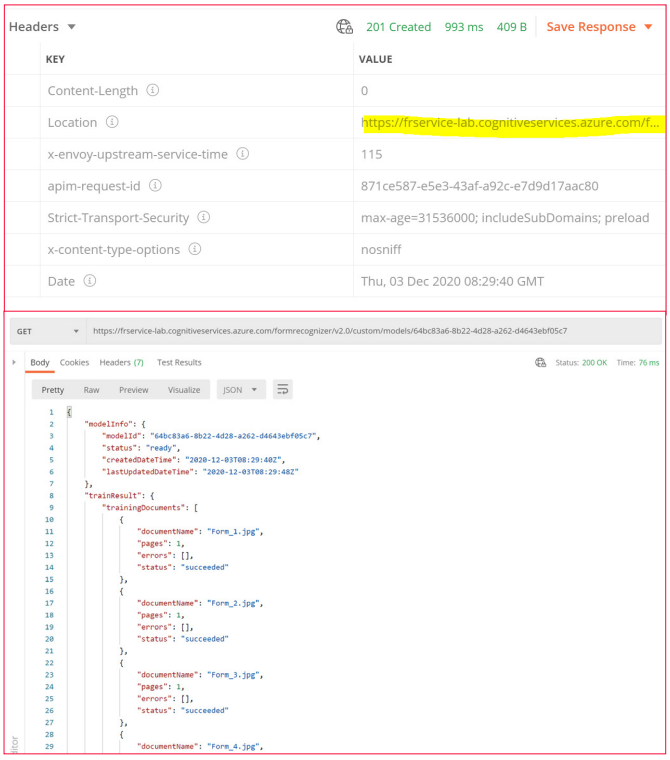


Imagen 12.- Resultado de la petición.

Como vemos en la anterior imagen, el resultado del entrenamiento nos devuelve “modelId”, que debemos anotar para poder pasárselo a nuestro servicio de Form Recognizer en el próximo análisis. Debemos recordar que, a mayor número de formularios aportados en el entrenamiento, mejor eficiencia tendrá nuestro modelo, aunque puedes comprobar con el ejemplo que con tan solo 5 formularios podemos empezar a probar el modelo.

Ejecutando un análisis con un modelo personalizado

A diferencia del primer análisis, ahora le vamos a pasar al análisis el ID del modelo entrenado en el paso anterior. Para eso vamos a revisar FromRecognizerService en el proyecto Shared, ya que debe tener un método que permita añadir el “model id” como parámetro al análisis.

```
2 references | Sergio Hernández Mancebo, 16 days ago | 1 author, 1 change
public async Task<RecognizedFormCollection> AnalyzeCustomFormFromStream(Stream form,string modelId)
{
    RecognizedFormCollection collection = await _frClient.StartRecognizeCustomFormsAsync(modelId,form)
    return collection;
}
```

Imagen 13.- Analizando con modelo.

Como se aprecia en el código anterior, la propia librería cliente de Form Recognizer nos expone un método para "Custom Form", en el que le podemos pasar el modelId, además del propio fichero. Si volvemos a la consola, ahora le debemos pasar el modelId al wizard para que nos devuelva correctamente el análisis:

```
C:\Program Files\dotnet\dotnet.exe
Quiere procesar un formulario (1) o un recibo (2)
1
Introduzca el modelo de entrenamiento:
64bc83a6-8b22-4d28-a262-d4643ebf05c7
Form of type: form-0
Field 'field-0:
    Label: 'Hero Limited
    Value: '
    Confidence: '0,5
Field 'field-1:
    Label: 'Company Phone:
    Value: '555-348-6512
    Confidence: '1
Field 'field-2:
    Label: 'Website:
    Value: 'www.herolimited.com
    Confidence: '1
Field 'field-3:
    Label: 'Dated As:
    Value: '12/20/2020
    Confidence: '1
Field 'field-4:
    Label: 'Email:
    Value: 'accounts@herolimited.com
    Confidence: '1
Field 'field-5:
    Label: 'Purchase Order #:
    Value: '948284
    Confidence: '1
Field 'field-6:
```

Imagen 14.- Prueba de custom form.

Como podemos revisar el cambio es sustancial, ya que ya no nos devuelve filas y palabras, si no que ya nos devuelve un "clave-valor" basado en "campos", los cuales podemos mapear mucho más fácil. De hecho, para que entendáis aún mejor lo que os digo, podemos revisar los métodos "Print" del proyecto de consola, y en el caso de Custom Form con modelo entrenado, el código es mucho más limpio y sencillo que si no le pasamos un modelo.

```
public static void PrintCustomForm(RecognizedFormCollection forms)
{
    foreach (RecognizedForm form in forms)
    {
        Console.WriteLine($"Form of type: {form.FormType}");
        foreach (FormField field in form.Fields.Values)
        {
            Console.WriteLine($"Field '{field.Name}: ");

            if (field.LabelData != null)
            {
                Console.WriteLine($"    Label: '{field.LabelData.Text}");
            }

            Console.WriteLine($"    Value: '{field.ValueData.Text}");
            Console.WriteLine($"    Confidence: '{field.Confidence}");
        }
    }
}
```

Imagen 15.- Print de un análisis.

Aunque este código ya nos podría servir para mapear el modelo a nuestro ERP, le encuentro aun una pega, y es que "Field" sigue teniendo valores genéricos como "Field-1", "Field-2" …, y deberíamos tirar de comparación de cadenas por el campo "field.LabelData.Text", y aunque con LINQ por ejemplo podríamos intentar realizar un mapper, el rendimiento de esto no es lo mejor del mundo, sin contar que a la mínima que nos cambien el formulario ni nos vamos a enterar, ya que esto sigue sacando todos los campos del formulario, pero la "clave" digamos no es única.

Para conseguir esta última mejora, antes de hacer nuestra Azure Functions vamos a realizar un entrenamiento con "Labels".

Entrenamiento con Labels, Claves-Valor más controladas en Form Recognizer

El último paso que nos queda una vez hemos visto como entrenar nuestro servicio y los frutos que recogemos en forma de una extracción mucha más limpia, es hacer uso del "Labeling" para poder tener claves únicas dentro de nuestro formulario, y así poder ver como mejoramos la salida.

Para conseguir añadir un entrenamiento con labels, yo os recomiendo utilizar la herramienta de etiquetado de ejemplo https://docs.microsoft.com/es-es/azure/cognitive-services/form-recognizer/quickstarts/label-tool?tabs=v2-0, la cual podemos desplegar desde un Docker en nuestro tenant o utilizar la versión de prueba que nos proporciona Microsoft https://fott-preview.azurewebsites.net/ .

En cualquier caso, la herramienta nos va a permitir entrenar nuestros formularios y añadirle con una herramienta muy visual labels; en los enlaces anteriores viene muy explicado como realizar la conexión al blob storage y como añadir etiquetas.

Yo si me voy a parar en que necesitais añadir las etiquetas necesarias para poder procesar los formularios que tenemos en la carpeta Samples, y que nos devuelva las siguientes etiquetas para que nuestra Azure Functions sepa luego mapearlas.

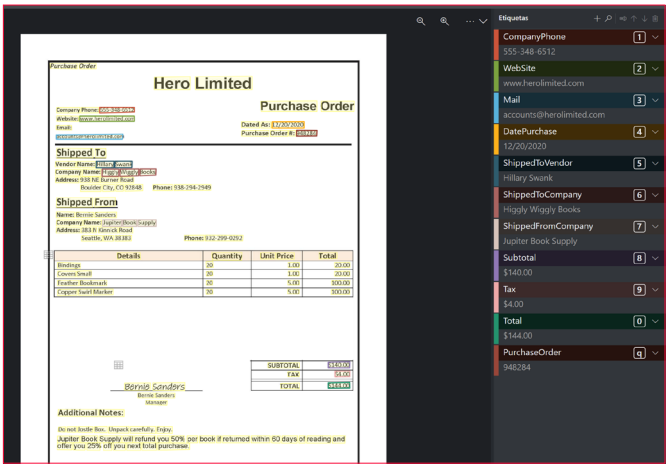


Imagen 16.- Tooling label.

Como veis en la imagen, necesitamos indicarle a la herramienta los "labels anteriores" con esos nombres concretos de cara a que por código podamos luego controlar esas claves únicas.

"la herramienta nos va a permitir entrenar nuestros formularios y añadirle labels"

Una vez añadido todos los labels en los 5 formularios, en la pestaña entrenar podemos generar un modelo de entrenamiento, tal y como hicimos vía Postman en el ejemplo anterior.

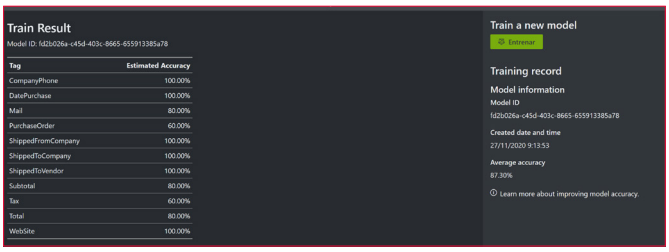


Imagen 17.- Entrenando con la herramienta.

Vamos a realizar una nueva ejecución de la consola de ejemplo en nuestra solución de Visual Studio, pero pasandole el "Model ID" , que obtenemos de la herramienta de labeling.

```
C:\Program Files\dotnet\dotnet.exe
Quiere procesar un formulario (1) o un recibo (2)
1
Introduzca el modelo de entrenamiento:
fd2b026a-c45d-403c-8665-655913385a78
Form of type: custom:form
Field 'WebSite:
    Value: 'www.herolimited.com
    Confidence: '1
Field 'Subtotal:
    Value: '$140.00
    Confidence: '1
Field 'Total:
    Value: '$144.00
    Confidence: '1
Field 'Tax:
    Value: '$4.00
    Confidence: '1
Field 'ShippedFromCompany:
    Value: 'Jupiter Book Supply
    Confidence: '1
Field 'PurchaseOrder:
    Value: '948284
    Confidence: '1
Field 'DatePurchase:
    Value: '12/20/2020
    Confidence: '1
Field 'ShippedToCompany:
    Value: 'Higgly Wiggly Books
    Confidence: '1
Field 'CompanyPhone:
```

Imagen 18.- Ejecucion con labels.

Ahora vemos que si obtenemos nombres de campos "únicos" como "Field Purchase Order" en vez de "Field -1 ", y estamos listos para construir un Azure Functions genérico, que recoja formularios y devuelva este clave valor que hemos entrenado.

Construyendo nuestra Azure Function de Integración

Estamos muy cerca de tener un proceso que integre nuestro blob y el servicio de Form Recognizer, y para ello vamos a construir un Azure Function. En el ejemplo del repositorio ya tenéis creada la Azure Function, que es un proyecto base y tiene una function CustomFormFunctions con un trigger del tipo HTTP.

Necesitamos una clase que contenga el modelo base de nuestro formulario y la encontramos en InvoceModel.cs :

```
public class InvoceModel
{
    public string CompanyPhone { get; set; }
    public string WebSite { get; set; }
    public string Email { get; set; }
    public string Date { get; set; }
    public string ShippedToVendor { get; set; }
    public string ShippedtoCompany { get; set; }
    public string ShippedFromCompany { get; set;}
    public string Subtotal { get; set; }
    public string Tax { get; set; }
    public string Total { get; set; }

    public string PurchaseOrder { get; set; }
}
```

Imagen 19.- Model Invoices.

Que decir que esta clase debe coincidir con los Labels que definíamos en la herramienta de Form Recognizer. Volviendo a la Functions, que es muy muy sencilla gracias a que Form Recognizer hace todo el trabajo por nosotros, vamos a tener el siguiente código:

```
[FunctionName("Analyze")]
0 references | Sergio Hernández Mancebo, 13 days ago | 1 author, 1 change
public static async Task<IActionResult> Run(
    [HttpTrigger(AuthorizationLevel.Function, "get", "post", Route = null)] HttpRequest req,
    ILogger log)
{

    string fileName = req.Query["fileName"];
    string model = req.Query["model"];
    var container = Environment.GetEnvironmentVariable("BlobUrl");
    var sass = Environment.GetEnvironmentVariable("Sassblob");
    var file = await GetDocumentFromStorage(fileName, container, sass);
    var streamFile = new MemoryStream(file);

    var frService = Environment.GetEnvironmentVariable("FormRecognizerService");
    var suscriptionKey = Environment.GetEnvironmentVariable("SuscriptionKeyForms");

    FormRecognizerService frClient = new FormRecognizerService(frService, suscriptionKey);
    var dataForm = await frClient.AnalyzeCustomFormFromStream(streamFile,model);
    var mappingForm = MappingForm(dataForm.FirstOrDefault());

    return new OkObjectResult(mappingForm);
}
```

Imagen 20.- Azure Functions.

> ***"Toda la magia de la function reside en los métodos AnalyzeCustomFormFromStream, que ya hemos visto en los ejemplos anteriores"***

Analizandolo vemos que en la settings de la Functions necesitamos almacenar una firma de acceso para poder llegar al blog storage, la url del blob donde vamos a ir subiendo las imágenes y las ordenes de pedido a procesar. Además necesitamos los datos de conexión al servicio de form recognizer.

```
{
 "IsEncrypted": false,
 "Values": {
  "AzureWebJobsStorage": "UseDevelopmentStorage=true",
  "FUNCTIONS_WORKER_RUNTIME": "dotnet",
  "Sassblob": "{SASS}",
  "BlobUrl": "{BLOB URL}",
  "FormRecognizerService": "{Form Recognizer URL}",
  "SuscriptionKeyForms": "{key}"
 }
}
```

Toda la magia de la function reside en los métodos AnalyzeCustomFormFromStream, que ya hemos visto en los ejemplos anteriores y que hemos utilizado en la consola, y además por novedosos debemos analizar los métodos de la propio función:

- GetDocumentFromStorage

  Este método realiza una petición HTTP al blob para obtener un array de bytes de este, con el que construiremos un Stream y desde el cual analizaremos con Form Recognizer.

  Lo curioso es la construcción de la URL en la variable "Uri" para realizar el get, y que se apoya en la firma del Blob para poder posicionar el fichero dentro del mismo.

```
private static async Task<byte[]> GetDocumentFromStorage(string fileName,string containerUri,strir
{

    var uri = $"{containerUri}/{fileName}{sasToken}";
    using (var client = new HttpClient())
    {
        using (var request = new HttpRequestMessage())
        {
            request.Method = HttpMethod.Get;
            request.RequestUri = new Uri(uri);

            var response = await client.SendAsync(request).ConfigureAwait(false);

            if (response.IsSuccessStatusCode)
            {
                return await response.Content.ReadAsByteArrayAsync();
            }
            else
            {
                return new byte[] { };
            }
        }
    }
}
```

Imagen 21.- Obteniendo el fichero.

- MappingForm

```
private static InvoceModel MappingForm(RecognizedForm form)
{
    var result = new InvoceModel()
    {
        CompanyPhone = form.Fields.Where(f => f.Key == "CompanyPhone").FirstOrDefault().Value?.ValueData?.Text,
        Date = form.Fields.Where(f => f.Key == "DatePurchase").FirstOrDefault().Value?.ValueData?.Text,
        Email = form.Fields.Where(f => f.Key == "Mail").FirstOrDefault().Value?.ValueData?.Text,
        PurchaseOrder = form.Fields.Where(f => f.Key == "PurchaseOrder").FirstOrDefault().Value?.ValueData?.Text,
        ShippedFromCompany = form.Fields.Where(f => f.Key == "ShippedFromCompany").FirstOrDefault().Value?.ValueData?.Text,
        ShippedToCompany = form.Fields.Where(f => f.Key == "ShippedToCompany").FirstOrDefault().Value?.ValueData?.Text,
        ShippedToVendor = form.Fields.Where(f => f.Key == "ShippedToVendor").FirstOrDefault().Value?.ValueData?.Text,
        Subtotal = form.Fields.Where(f => f.Key == "Subtotal").FirstOrDefault().Value?.ValueData?.Text,
        Tax = form.Fields.Where(f => f.Key == "Tax").FirstOrDefault().Value?.ValueData?.Text,
        Total = form.Fields.Where(f => f.Key == "Total").FirstOrDefault().Value?.ValueData?.Text,
        WebSite = form.Fields.Where(f => f.Key == "WebSite").FirstOrDefault().Value?.ValueData?.Text

    };

    return result;
}
```

Imagen 22.- Mapping.

Esta es la magia de todo el ejemplo, si analizais este mapeo es muy sencillo, respecto a los métodos de print que tenemos en la consola por no utilizar Labels.

El parámetro de entrada “RecognizedForm”, nos lo devuelve el propio servicio de Form Recognizer, y este tiene unos campos con unas claves únicas, que podemos mediante una consulta en LINQ extraer de forma controlada. Al ser claves únicas podemos controlar si viene a null, si no viene bien conformada, o si el formulario no tiene la estructura correcta en el proceso de análisis.

Automatizando la carga en el ERP con Logic App y nuestra Azure Function

Una vez terminado la Azure Function, tal y como la hemos visto en el punto anterior, la podemos integrar como queramos de cara a automatizar nuestro proceso de extracción de ordenes de pago.

Debemos recordar que nuestra función, recibe por Query tanto el nombre del fichero a procesar dentro del blob de facturas, como el Id de modelo, y los dos son necesarios para que el proceso no falle, tal y como vemos en el siguiente código:

```
public static async Task<IActionResult> Run(
[HttpTrigger(AuthorizationLevel.Function, “get”, “post”, Route
= null)] HttpRequest req,
ILogger log)
{

string fileName = req.Query[“fileName”];
string model = req.Query[“model”];
```

En mi caso he optado por construir una Logic App que tiene como trigger la subida de un fichero a un blob, invoca a nuestra Azure Function que ya está correctamente desplegada en mi tenant, y por último graba en una entidad de Common Data Service los datos de la factura o de la orden de pago. Este punto es un poco libre, pero si analizamos una ejecución podremos ver la composición del flujo de forma correcta:

Flujo muy simple

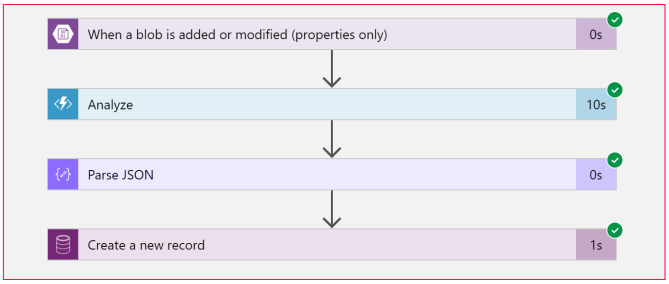


Imagen 23.- Logic App de inserción de facturas.

Como podemos ver es un flujo de apenas 4 acciones, ya que toda la lógica de negocio la tiene nuestra función, que a su vez podemos opinar también que es muy sencilla, lo cual nos habla mucho de todo lo que ha aportado el entrenamiento con Labels, y la potencia de extracción del servicio de Form Recognizer.

> ***"Una vez terminado la Azure Function, la podemos integrar como queramos de cara a automatizar nuestro proceso de extracción de ordenes de pago.***
> ***"***

La primera acción es un trigger de Blob Storage, que conecta el blob con la Logic App, debe ser el mismo blob que configuremos en las settings del Azure Functions, ya que si recordamos nuestra Azure Function solo recibe el nombre del fichero y el id del modelo.

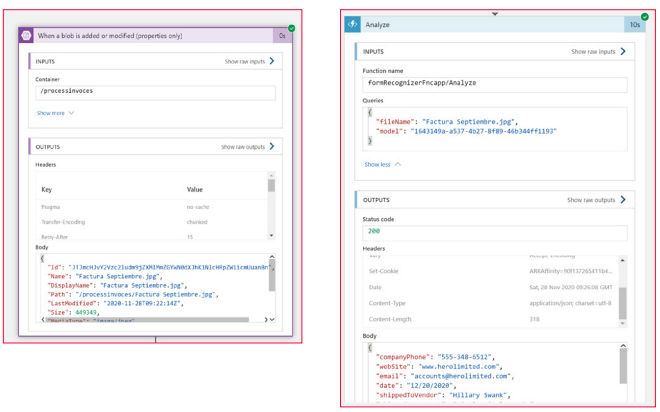


Imagen 24.- Trigger de la Logic App.

Como vemos en las imágenes anteriores, nuestra función recibe de forma correcta el nombre del fichero, que obtenemos desde el trigger contra el Blob Storage. Las otras dos acciones, son simplemente una acción de parseo a un modelo JSON que nuestra Logic App pueda procesar, y por último una escritura en la entidad Invoces de nuestro Common Data Service.

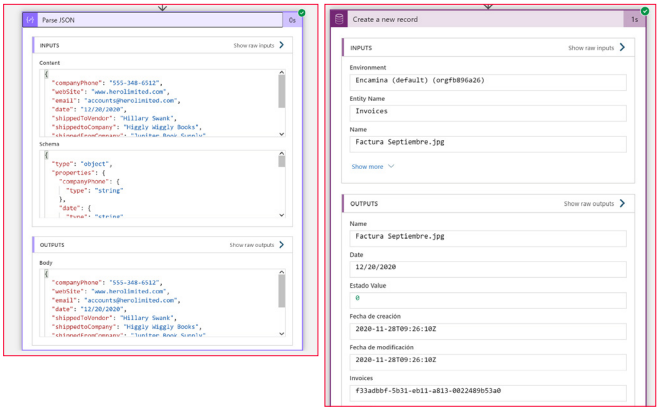


Imagen 25.- Parseo de JSON y escritura en el CDS.

El resultado de este proceso es que cada vez que subamos un fichero a un Blob, se genera un registro en nuestro Common Data Service, derivado de extraer un conjunto de claves y valores desde los formularios de ordenes de pago, y todo sin necesidad de que un ser humano revise los documentos. Desde aquí podemos empezar a crear aplicaciones frontales que se alimenten de nuestra entidad de Common Data Service y hacer todos los procesos de negocio necesarios.

¡¡¡¡Mira que igual esto funciona y todo!!!!

Este servicio me ha demostrado dos cosas, uno que esta muy maduro, y que, si hacemos unos entrenamientos limpios, y organizamos bien nuestra información, los procesos de extracción van a ser muy muy livianos y fácil de mantener, dejando toda la carga al servicio cognitivo.

Y la segunda conclusión que me llevo, es que cada vez salen más estables los servicios en Azure, si bien hace unos años había que coger con pinzas cada nuevo servicio desplegado, hoy en día tienen una madurez la plataforma, que casi no te cuesta integrar lo nuevo que va saliendo, y tiene una fiabilidad muy muy alta.

Yo recomiendo empezar a utilizar este servicio desde ya, me parece super interesante el hueco que tiene en el mercado tanto del Knowlege mining y como se puede complementar con un Cognitive Search, o para procesos de integración y extracción tan demandado en nuestras organizaciones. Felicidades a Azure y Microsoft por sacar un servicio tan interesante y útil!!!

SERGIO HERNÁNDEZ MANCEBO
Azure MVP
@shmancebo

22

Troubleshooting del networking de Azure mediante Network Watcher

En el número 45 de CompartiMOSS (http://www.compartimoss.com/revistas/numero-44/uso-de-azure-monitor), pudimos ver una introducción a Azure Monitor, en esta estrega hablaremos sobre Network Watcher, para los que no lo conozcáis, puede ser considerado como una navaja suiza para solucionar temas relacionados con el networking. Ya sea para monitorizar o bien solucionar problemas que podamos tener en nuestra red y componentes relacionados con el networking.

Habilitar Network Watcher

Primero de todo, lo que debemos de hacer es habilitar el Network Watcher. Para ello, debemos de hacerlo en cada una de las regiones que tenemos una VNET desplegada o bien que vaya a ser objeto en un futuro. Para hacer esto, bien podemos hacerlo por la UI del portal o bien por PS, navegando al servicio de Network Watcher (buscando el servicio en la barra de búsquedas).

Una vez hayamos encontrado el servicio, hacemos clic derecho sobre la región que necesitamos habilitar el servicio y le damos a la opción de “Enable network watcher”

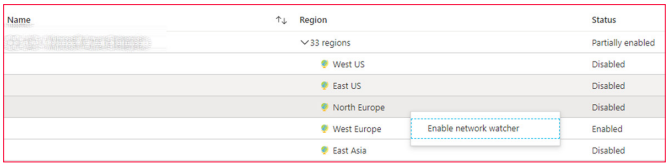


Imagen 1.- Habilitar Network watcher.

La primera vez que hagamos esto, nos creará un grupo de recursos llamado “NetworkWatcherRG”, si entramos en este RG y mostramos los elementos ocultos, veremos que contiene recursos del tipo “microsoft.network/networkwatchers”.

"Network Watcher, para los que no lo conozcáis, puede ser considerado como una navaja suiza en para solucionar temas relacionados con el networking…"

Tened en cuenta que si estáis siguiendo un plan de gobierno para vuestros recursos de Azure (deberíais de tener uno si no es así), este grupo de recursos se puede eliminar y podéis habilitar el network watcher en un RG existente o bien crear uno con la nomenclatura que necesites (PS es vuestro amigo)

Una vez hayamos habilitado este servicio, ya estamos preparados para consumir otros servicios que nos darán la posibilidad de tener características para el troubleshooting.

IP Flow Verify

Esta característica nos ayuda a saber el tráfico IP entre dos direcciones, teniendo en cuenta dirección o bien puerto. Es muy útil para conocer si el tráfico es capaz de fluir o bien está bloqueado.

Simplemente, seleccionamos la VM, la interfaz de red que queremos checkear, veremos que la dirección IP Local se autopopula, simplemente deberemos de introducir el resto de los detalles como la dirección IP remota, puerto y hacer clic en check, una vez hecho esto, la herramienta simulará el tráfico y nos arrojará los resultados.

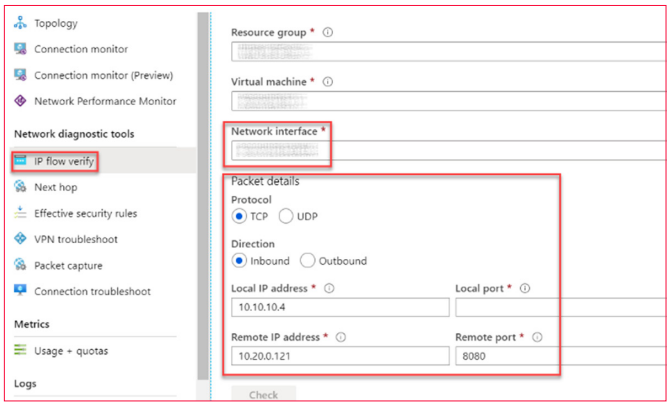


Imagen 2.- IP Flow verify.

Next Hop

En el caso de que en vuestra Sub Net tengáis varias tablas de rutas asociadas, y necesitéis verificar como el tráfico está fluyendo hacia una ruta en particular, “Next Hop” nos ayudará a ello. Con esta herramienta podemos verificar si el tráfico desde un origen a un destino pasará por un virtual appliance o no, basándose en las rutas que estén definidas en la propia tabla de rutas.

Basta con seleccionar origen y destino, las direcciones IP

se completan en función de la NIC seleccionada. Simplemente nos quedará rellenar la dirección destino, una vez introducidos los datos, hacemos un check y nos mostrará los siguientes saltos junto con la ruta que está redirigiendo el tráfico. Muy útil para escenarios de Hub&Spoke o bien para aquellos que utilizamos FW de 3rd parties.

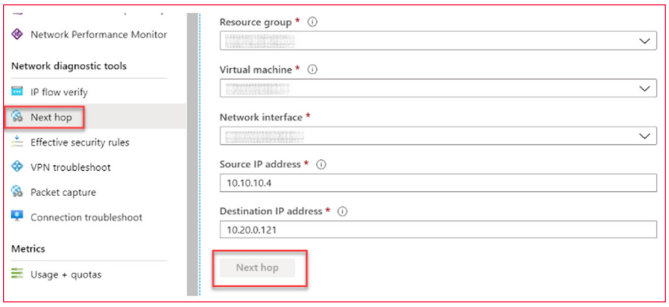


Imagen 3.- Next hop.

Effective Routes

Pese a que esta herramienta no esté incluida dentro del Network Watcher, va muy en la línea del anterior punto, nos permite mostrar todos y cada uno de los puntos de conexión que tenemos para una NIC en particular, de manera que de una forma simple podemos ver como fluye el tráfico y si tenemos alguna tabla de rutas que nos esté enrutando el tráfico de una manera u otra. Puede ser muy útil cuando estamos en escenarios con FW de por medio…

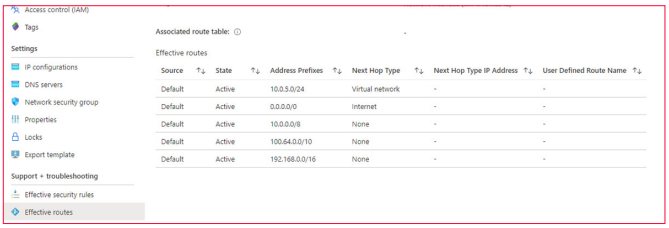


Imagen 4.- Effective Routes.

Effective Security Rules

Se puede dar el caso de que estemos protegiendo el tráfico mediante varios NSG aplicados a una VM, o bien uno aplicado a nivel de NIC y otro a nivel de SubNet. Por lo que esta herramienta nos ayudará a conocer que NSG se están aplicando a nuestra VM y qué reglas se están aplicando en todo momento:

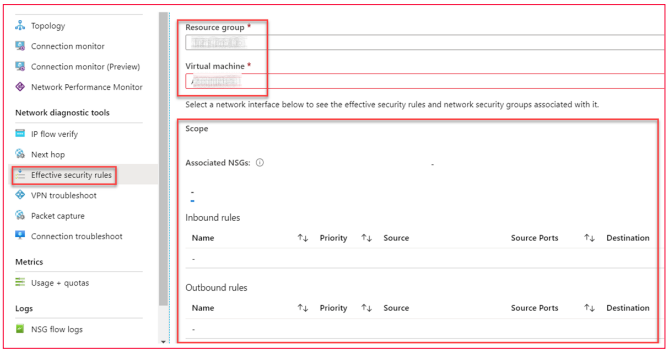


Imagen 5.- Effective Security Rules.

VPN Troubleshoot

Si tenemos problemas con nuestra VPN, este es el punto donde deberemos acudir, esta herramienta nos ayudará a conocer cuál es el estado de la VPN mediante el estudio de las trazas de comunicación entre el Gateway de Azure y el otro extremo (ya sea un dispositivo VPN local o de otro proveedor Cloud).

La única diferencia, es que deberemos de tener una cuenta de almacenamiento previamente creada, ya que ahí se almacenarán todas las trazas que se generen. Una vez acabado el proceso, podemos acceder al txt generado, y empezar a ver por dónde puede venir el posible problema.

"nos permite mostrar todos y cada uno de los puntos de conexión que tenemos para una NIC en particular, de manera que de una forma simple podemos ver como fluye el tráfico"

Packet Capture

Nos permite capturar sesiones para seguir el tráfico desde y a una VM, nos permite diagnosticar anomalías que pueda haber en la red, así como estadísticas de la red, intrusiones, debugar comunicaciones cliente-servidor, etc.

Debemos de tener en cuenta que esta solución se instala como una extensión de la VM en particular, por lo que, si vuestra VM no es compatible con el agente de Azure o bien tiene algún problema, es posible que no os funcione, por lo que deberéis de verificar eso antes.

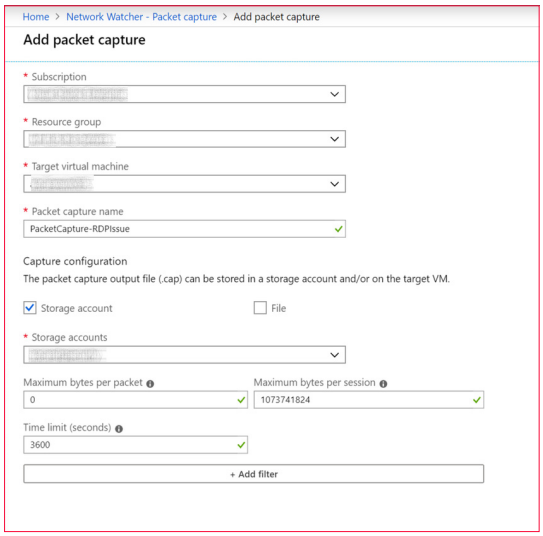


Imagen 6.- Packet Capture.

Topology

Personalmente, esta es una característica que me encanta, y que cuando entro a un proyecto sobre Azure, le suelo dar uso, ya que nos permite ver en forma de diagrama la topología de red y de interconexión de los recursos en Azure. En esta representación veremos, VNets, VMs, NICs e incluso NSG que estén aplicando.

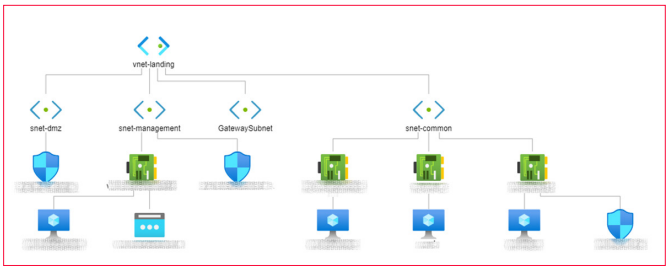


Imagen 7.- Topology.

Traffic analytics

Nos permite tener una visualización gráfica del tráfico de la red, así como su distribución, incluyendo todos los objetos, regiones, VNets, VPNs, NSGs y SubNets, y lo más chulo de todo, es que todos los objetos tienen dashboards interactivos:

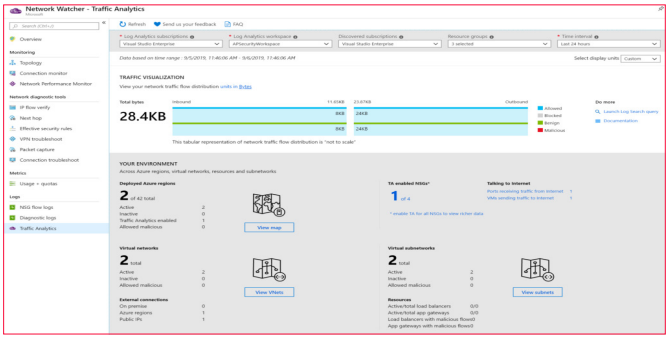


Imagen 8.- Traffic Analytics.

"Network Watcher es una herramienta muy válida para hacer troubleshooting de nuestros entornos, conocer que está pasando en cada una de las piezas del NW en un despliegue de Azure"

Una de las opciones que tenemos, es la visualización del mapa, en ella podemos ver las direcciones IP y podemos ver más opciones, como información de capa 7 y más:

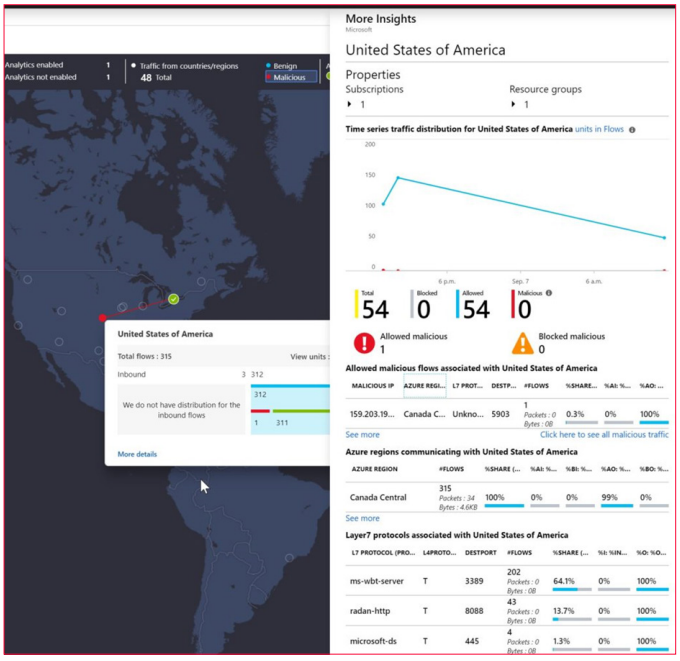


Imagen 9.- Visualización del mapa.

Resumen

Ya vemos que Network Watcher es una herramienta muy válida para hacer troubleshooting de nuestros entornos, conocer que está pasando en cada una de las piezas del NW en un despliegue de Azure, así como identificar riesgos potenciales que podamos estar teniendo.

ALBERTO ANDRÉS RODRÍGUEZ
Cloud Solutions Architect @ Seidor
@albandrod

25

El módulo de SecretManagement de PowerShell

El SecretManagement es un módulo de PowerShell que almacena "secretos" de forma segura, tales como credenciales de usuario y tokens de APIs. El módulo dispone de un conjunto de cmdlets que se pueden usar para administrar los secretos desde PowerShell. Esto resuelve algunos problemas comunes, como tener que introducir manualmente secretos en cada sesión o almacenarlos en un archivo de texto.

"SecretManagement Preview 3, versión que ya es prácticamente definitiva, pero que todavía no es recomendado utilizarlo para sistemas de producción"

Los componentes del módulo son extensibles, por lo que cualquier almacén de secretos con una API / CLI puede hacerse compatible con el SecretManagement. Esto simplifica la integración de depósitos de secretos ya existentes (Azure Key Vault [https://azure.microsoft.com/en-us/services/key-vault/], KeePass [https://keepass.info], Hashicorp Vault [https://www.vaultproject.io], SecureStore [https://github.com/neosmart/SecureStore], etc.).

Recientemente Microsoft ha lanzado el SecretManagement Preview 3, versión que ya es prácticamente definitiva, pero que todavía no es recomendado utilizarlo para sistemas de producción. Adicionalmente, Microsoft también lanzó el Preview 1 del SecretStore, que es un almacén de extensiones local multiplataforma que funciona en todas las plataformas compatibles con PowerShell, incluidos Linux y Mac.

Prerrequisitos

El SecretManagment está hecho para funcionar con PowerShell Core (desde la versión 7), y no funciona en el PowerShell tradicional (hasta la versión 5). Las versiones Core y tradicional de PowerShell pueden convivir simultáneamente en un computador, por lo que, si no tiene instalado la versión Core, puede hacerlo sin esperar problemas con la versión tradicional.

PowerShell Core puede ser utilizado desde cualquier plataforma: Windows 32 y 64 bits, Linux, Centos, Debian, Ubunto, Mac, etc. A la fecha de escribir este artículo, PowerShell Core se encuentra en versión 7.1.0 Release candidato 1 (RC), y se puede descargar desde la página de releases del sitio de GitHub de PowerShell (https://github.com/PowerShell/PowerShell/releases). Descargue la versión adecuada e instálela en el computador a utilizar.

Microsoft proporciona instrucciones detalladas sobre instalación, migración y uso de PowerShell Core en su sitio Web https://docs.microsoft.com/en-us/powershell/scripting/install/installing-powershell?view=powershell-7.

Instalación del módulo de SecretManagement

Para instalar el módulo de SecretManagement, abra una consola de PowerShell 7 como administrador y ejecute el siguiente comando (acepte la instalación del componente desde "PSGallery" cuando un mensaje de advertencia aparezca):

```
Install-Module Microsoft.PowerShell.SecretManagement -AllowPrerelease
```

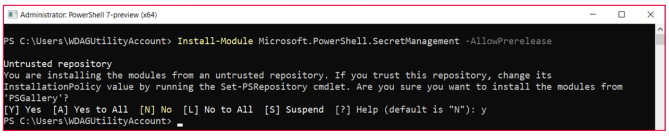


Imagen 1.- Instalación del módulo de SecretManagement.

El módulo de SecretStore, como se indicó anteriormente, permite crear y mantener un repositorio local de secretos en el computador del usuario. Para instalar este módulo, ejecute el siguiente comando:

```
Install-Module Microsoft.PowerShell.SecretStore -AllowPrerelease
```

Creación y manejo de un almacén local de secretos

El comando:

```
Get-Command -Module Microsoft.PowerShell.SecretStore | select name
```

Muestra los cmdlets disponibles para el SecretStore:

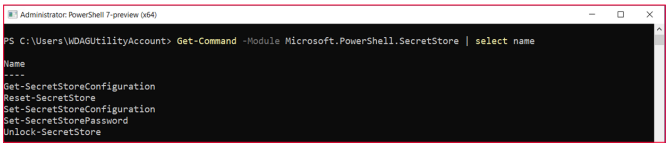


Imagen 2.- Detalle de los comandos disponibles para SecretStore.

Note que no hay cmdlets para remover un Store.

- Reset-SecretStore borra todos los secretos almacenados en todos los almacenes.
- Set-SecretStorePassword permite crear o configurar la clave para los almacenes.
- Set-SecretStoreConfiguration y Get-SecretStoreConfiguration permiten ver y cambiar los parámetros de configuración de un almacén, por ejemplo, el tiempo que la clave permanece activa antes de que el usuario tenga que ingresarla de nuevo (900 segundos por defecto).
- Unlock-SecretStore desbloque un SecretStore para el usuario actual con la contraseña proporcionada. Se puede utilizar para desbloquear un almacén cuando la configuración requiere una contraseña y la opción de configuración de solicitud está deshabilitada. Por ejemplo, cuando es necesario ejecutar scripts sin interacción del usuario. La contraseña proporcionada se aplicará a la sesión actual y dejará de ser válida después de que transcurra el tiempo de 'PasswordTimeout'. Si el parámetro no proporciona ninguna contraseña, se le pedirá al usuario que la introduzca.

El siguiente comando crea un almacén local de secretos en el computador:

```
Register-SecretVault -Name CompartimossSecretStore -ModuleName Microsoft.PowerShell.SecretStore -DefaultVault
```

En este ejemplo se registra el módulo de almacén Microsoft.PowerShell.SecretStore (almacén local) para el usuario actual. El SecretStore se instala en una ruta de acceso de PowerShell conocida por el módulo, por lo que solo se necesita su nombre. Utiliza el modificador de parámetro 'DefaultVault' para convertirlo en el módulo predeterminado para el usuario.

"El SecretManagment está hecho para funcionar con PowerShell Core (desde la versión 7), y no funciona en el PowerShell tradicional (hasta la versión 5)"

El comando 'Get-SecretVault' da una lista de todos los almacenes registrados para el usuario y comprueba que el almacén se registró y estableció como el almacén predeterminado. En el siguiente ejemplo, mostrado en la imagen, se ve que hay dos almacenes registrados para el mismo usuario.

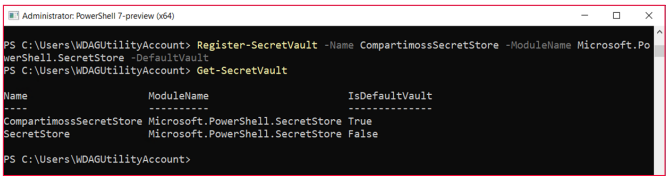


Imagen 3.- Comandos Register-SecretVault y Get-SecretVault.

Creación y manejo de secretos en un almacén local de secretos

El comando:

```
Get-Command -Module Microsoft.PowerShell.SecretManagement | select name
```

Muestra los cmdlets disponibles para manejar secretos:

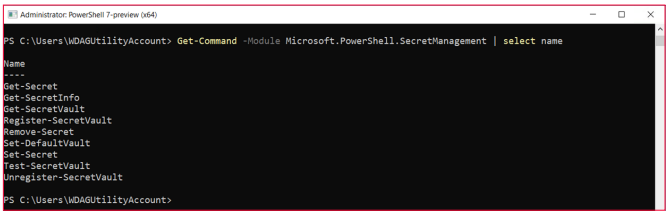


Imagen 4.- Comandos disponibles para manejo de secretos.

Antes de poder crear secretos, es necesario registrar el almacén con el cmdlet Register-SecretVault, como se indicó en la sección anterior. Para crear un secreto, utilice la siguiente sintaxis:

```
Set-Secret -Name "CompartimossKey" -Secret "LlaveMuySegura"
```

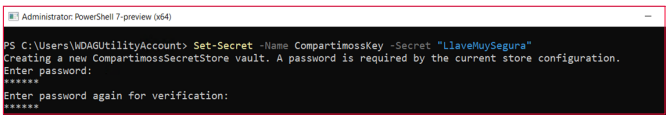


Imagen 5.- Creación de un secreto.

Los dos parámetros de entrada indican el nombre que identifica al secreto, y su valor. El parámetro -Vault (con el nombre de uno de los almacenes definidos anteriormente, no usado en el ejemplo) no es obligatorio, pero sirve para almacenar secretos en cualquier almacén deseado. Use el cmdlet Get_Secret para recobrar el valor de un secreto (utilizando el nombre del secreto):

Imagen 6.- Comando Get-Secret.

El valor recobrado es directamente un objeto del tipo SecureString que puede ser utilizado de inmediato para, por ejemplo, loguearse con SharePoint. Si se desea ver el valor del secreto, use el parámetro -AsPlainText:

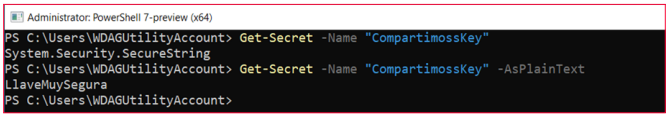


Imagen 7.- Visualización del valor de un secreto.

Después de que pasa el tiempo predefinido en la configuración para mantener la contraseña activa (900 segundos por defecto), o si se inicia una nueva sesión de PowerShell, es necesario entrar de nuevo la llave del almacén.

Usando SecretManagement con SharePoint

El módulo de SecretManagement se puede utilizar para guardar el usuario y la clave necesarios para usar SharePoint PnP PowerShell. Como el SecretManagement se puede utilizar solamente con PowerShell Core, es necesario utilizar también el módulo Core de SharePoint PnP PowerShell. Primero cree tres secretos en el almacén como se indica en seguida. Aquí se guardan como secretos los tres parámetros necesarios para loguearse solamente como un ejemplo; probablemente el primero no se debe guardar como secreto para poderlo manejar fácilmente en la configuración del script:

```
Set-Secret -Name "CompartimossUrl" -Secret "https://dominio.sharepoint.com/sites/TestCompartiMOSS"
Set-Secret -Name "CompartimossUser" -Secret "user@dominio.onmicrosoft.com"
Set-Secret -Name "CompartimossPw" -Secret "MyPassword"
```

Normalmente en una rutina de PnP PowerShell para el DotNet Framework, logearse en SharePoint y obtener el objeto de SharePoint Site significa utilizar código similar al del siguiente script:

```
$spUserPw = "MyPassword"
$spUserName = "user@dominio.onmicrosoft.com"
$spUrl = "https://dominio.sharepoint.com/sites/Test_Compartimoss"

[SecureString]$securePW = ConvertTo-SecureString -String $spUserPw -AsPlainText -Force
$myCredentials = New-Object -TypeName System.Management.Automation.PSCredential -argumentlist $spUserName,

$securePW
Connect-PnPOnline -Url $spUrl -Credentials $myCredentials

Get-PnpSite
```

La rutina es similar utilizando secretos del SecretManagement. Después de instalar el módulo de PnP Core para SharePoint PowerShell, utilice el siguiente script para utilizar los secretos guardados en el almacén:

```
$spUserPw = Get-Secret -Name "CompartimossPw"
$spUserName = Get-Secret -Name "CompartimossUser" -AsPlainText
$spUrl = Get-Secret -Name "CompartimossUrl" -AsPlainText

$myCredentials = New-Object -TypeName System.Management.Automation.PSCredential -argumentlist $spUserName, $spUserPw
Connect-PnPOnline -Url $spUrl -Credentials $myCredentials

Get-PnpSite
```

Note que se está utilizando la variable $spUserPw directamente para el objeto de credenciales, porque el módulo de SecretManagement lo entrega directamente como un SecureString. De esta forma, el script se puede ejecutar de forma más segura sabiendo que los datos sensibles no son transmitidos en plain text.

GUSTAVO VELEZ
Offices Apps & Services MVP
gustavo@gavd.net
https://guitaca.com
http://www.gavd.net

Mentoring

Comparti*MOSS*

Un servicio experto alrededor de su SharePoint

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

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

29

Microsoft Bookings: Gestión sencilla de citas en tiempos de pandemia y escenarios de Remote Work

Microsoft Bookings es una aplicación disponible de serie en distintos planes de Microsoft 365 que originalmente estaba pensada para que los pequeños negocios pudiesen contar con una solución que facilitase la reserva de citas en sus establecimientos u oficinas. Sin embargo, con la llegada de Microsoft Teams y sobre todo con el COVID-19, Microsoft está posicionando a Bookings como una solución simple para facilitar no solo la reserva de citas en negocios de distinta naturaleza, sino planificar consultas médicas por parte de pacientes y/o doctores, facilitar que padres y alumnos puedan tener reuniones con profesores, etc. En este artículo haremos un repaso a los escenarios de uso de Microsoft Teams, a sus también a sus principales características y a como configurar la solución.

Casos de Uso de Microsoft Bookings y características principales

Sin duda, los escenarios de uso de una solución como Microsoft Bookings son variados en cuanto a que facilita una gestión sencilla de citas tanto en escenarios in-person (no posibles hoy en día) como virtuales. A modo de ejemplo, en la Imagen 1 se pueden ver algunos de los casos de uso más habituales para Microsoft Bookings.

Imagen 1.- Algunos casos de uso de Microsoft Bookings.

"pensada para que los pequeños negocios pudiesen contar con una solución que facilitase la reserva de citas en sus establecimientos u oficinas"

En cuanto a los beneficios que Bookings aporta a una organización, la Imagen 2 muestra un resumen de estos en base a las principales características de la aplicación.

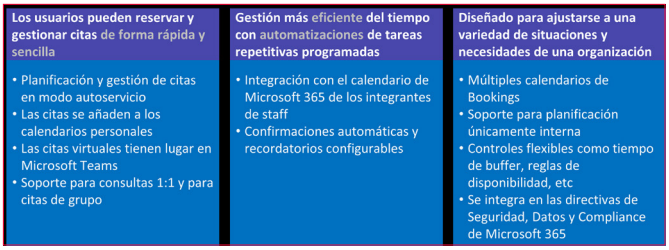


Imagen 2.- Características principales de Microsoft Bookings.

Disponibilidad de Microsoft Bookings en Microsoft 365

A nivel de disponibilidad de Microsoft Bookings, normalmente la aplicación está disponible para casi todos los planes de Microsoft 365 aunque dependiendo de como se haya adquirido la suscripción correspondiente, Bookings estará disponible de serie o será necesario añadir el Add-on de Business Apps. El resumen de disponibilidad de la aplicación en Microsoft 365 es el siguiente:

- Disponible por defecto en los siguientes planes de Microsoft 365:
  - Office 365 Edu A3, Office 365 Edu A5, Microsoft 365 Business Standard, Microsoft 365 Business Premium.
  - Desde agosto de 2020 en tenants GCC G3 y GCC G5.
  - Subscripciones de Microsoft 365 compradas de forma directa (Desde mayo de 2020).
- Disponible de forma indirecta a través del correspondiente Add-On:
  - Suscripciones adquiridas mediante un EA (Enterprise Agreement) o bajo modelo CSP.
  - Esta era la forma de disponer de Bookings hasta mayo de 2020 en suscripciones Office 365 E3, E5.

Configuración y uso de Microsoft Bookings

En primer lugar, a nivel de controles de administración

Bookings únicamente viene con un par de controles disponibles en la Settings → Org settings → Bookings:

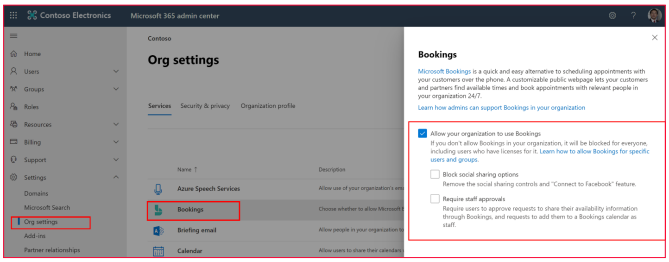


Imagen 3.- Controles de administración de Bookings.

En lo que a habilitar/deshabilitar Bookings se refiere, un administrador puede hacerlo a través de comandos PowerShell de Exchange Online provistos para este propósito. Pero en este artículo partimos de la base de que nos interesa utilizar Bookings, por lo que lo tendremos habilitado y solo tendremos que asignar la correspondiente licencia a los usuarios que vaya a utilizar Bookings:

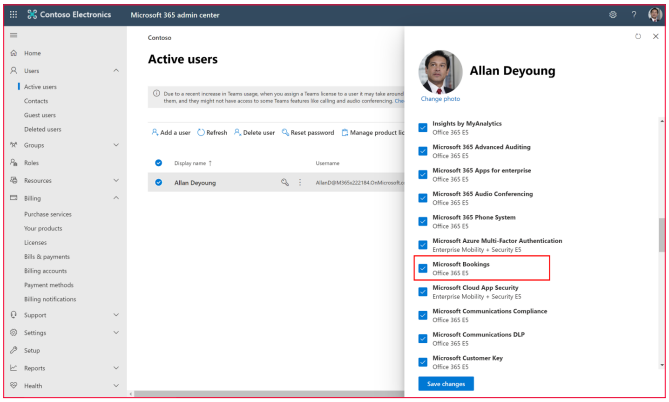


Imagen 4.- Licencia de Bookings asignada a un usuario.

A partir de este momento, podemos empezar a revisar como configurar Bookings accediendo a la solución desde el lanzador de aplicaciones (Nota: Bookings está construido sobre la base de elementos de Exchange Online como iremos viendo a lo largo de las siguientes páginas). Empezaremos revisando cada una de las opciones del menú de navegación de Bookings que tendremos disponible por cada calendario de que demos de alta en la solución:

- Business Information permite como su nombre indica completar información relativa a la organización que ofrece servicios reservables mediante Bookings: Nombre, dirección, teléfono, logo, etc. Además, desde esta sección podremos indicar los horarios de atención al público.

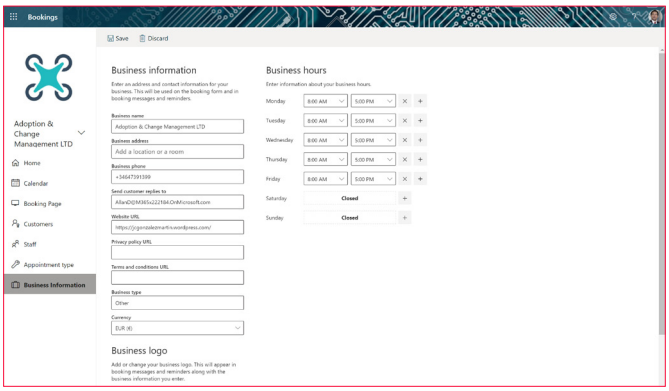


Imagen 5.- Sección Business Information.

- Appointment type permite establecer los tipos de citas que se pueden reservar a través de Bookings Por cada appointment podremos definir datos como nombre, descripción, ubicación, si se va a usar Microsoft Teams para generar un enlace para la cita, etc. También se puede asignar staff concreto para el appointment en concreto, la configuración de las notificaciones y los campos de información que se solicitarán a cualquier cliente, paciente o alumno que realice una reserva a través de la pçagima publica de Bookings. Desde la sección Appointmet type veremos todos los tipos appointments creados y tendremos la posibilidad de actualizarlos.

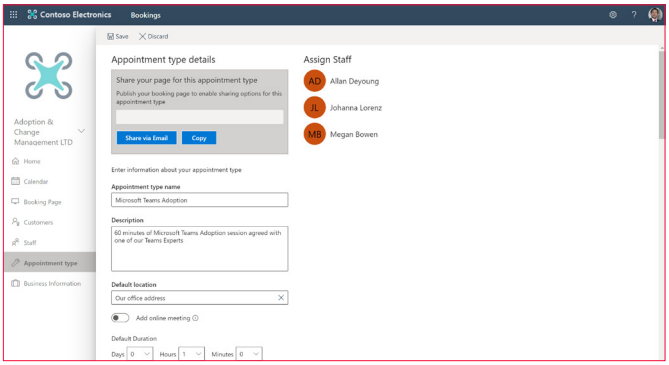


Imagen 6.- Formulario de detalle de un Appointment.

- Staff, proporciona acceso al listado de Staff asociado al calendario de Bookings actual. Podremos añadir nuevo Staff o actualizar Staff existente indicando datos como la persona concreta de la organización, su e-mail o las horas de atención al público (que pueden ser diferentes a las definidas globalmente para el calendario).

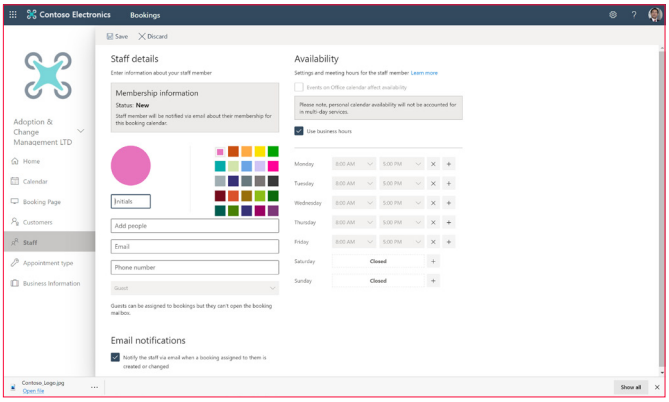


Imagen 7.- Detalle de un miembro del staff en Bookings.

- Customers, permite mantener un listado de clientes o contactos en Bookings sin pretender reemplazar a un CRM.

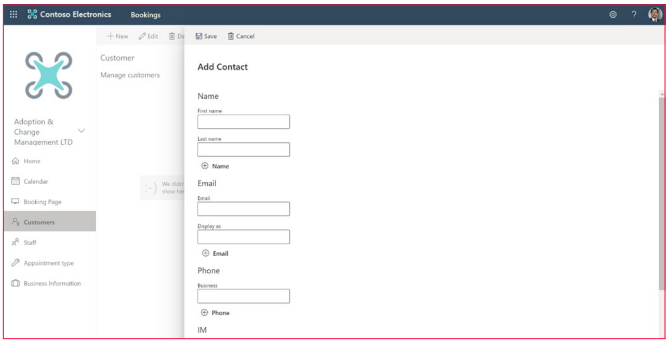


Imagen 8.- Gestión de contactos en Microsoft Bookings.

- La sección “Booking Page” es la que nos permite configurar la página pública (con acceso anónimo o no) desde la que facilitar la reserva de servicios y del staff correspondiente. Podremos aplicar branding mínimo a dicha página como elegir un tema a aplicar, establecer la política de planificación en aspectos como los incrementos de tiempo, la configuración regional a utilizar o el consentimiento de datos cuando se realiza una reserva.

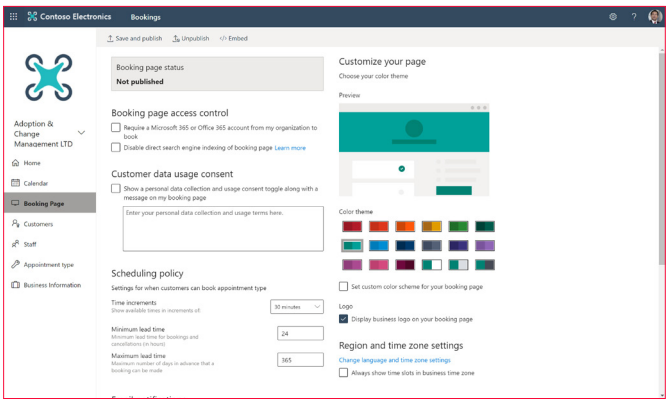


Imagen 9.- Configuración de la página pública de Bookings.

A partir de aquí, podemos decir que ya tenemos las herramientas necesarias para configurar Bookings y comenzar a utilizar la aplicación comenzando por la página pública de Bookings que tendrá una apariencia similar a la que se muestra en la Imagen 10. Como vemos:

- Se puede seleccionar entre los servicios disponibles.
- Para una fecha en concreta, podremos elegir entre el Staff disponible y de acuerdo con su disponibilidad.
- Además, se solicitarán los campos de información definidos en el tipo de servicio que se va a reservar.

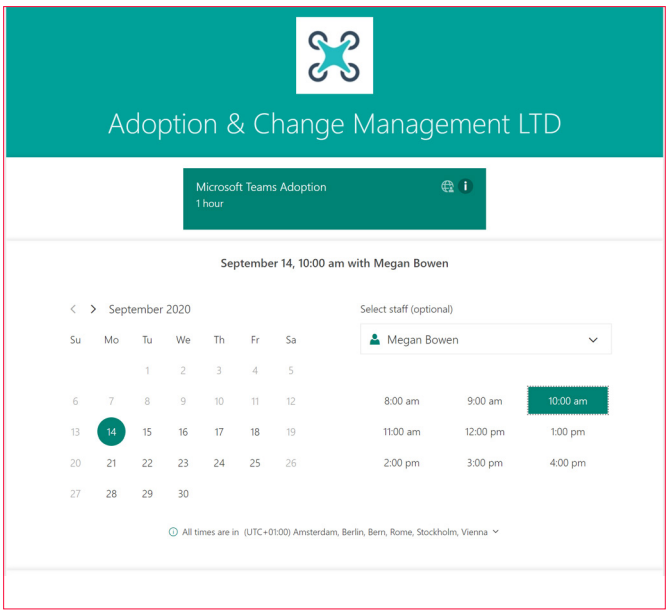


Imagen 10.- Página pública de Bookings.

Una vez realizada la reserva, tanto el staff reservado como el cliente/paciente/alumno recibirán un e-mail de confirmación de la reserva en el que se incluye también en enlace de la sesión de Microsoft Teams.

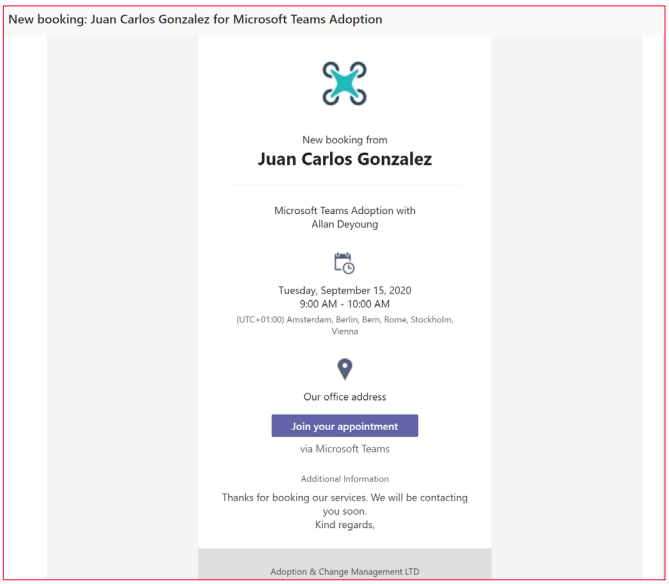


Imagen 11.- E-mail de confirmación enviado.

Por supuesto, las distintas citas, así como la disponibilidad del Staff se podrá ver en todo momento a través del calendario de Bookings integrado en la App.

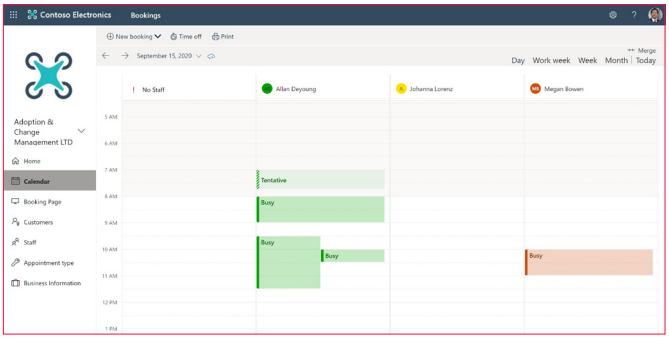


Imagen 12.- Calendario de Bookings integrado con el calendario del Staff.

Integración en Microsoft Teams

Además de la Aplicación de Microsoft Bookings, disponemos de una versión “ligera” de esta en la forma de una App para Microsoft Teams que, por defecto, como se puede comprobar en el Teams Admin Center, está configurada para que cualquier usuario pueda añadir la App en Teams.

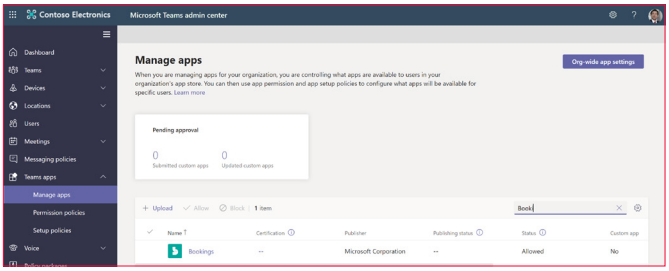


Imagen 13.- App de Bookings para Microsoft Teams.

" las distintas citas, así como la disponibilidad del Staff se podrá ver en todo momento a través del calendario de Bookings integrado en la App."

De esta forma, añadir la App de Bookings en Teams es tan sencillo como localizarla en el Catálogo de Aplicaciones de Teams. Como es esperable, previamente a añadir la App veremos los detalles de esta.

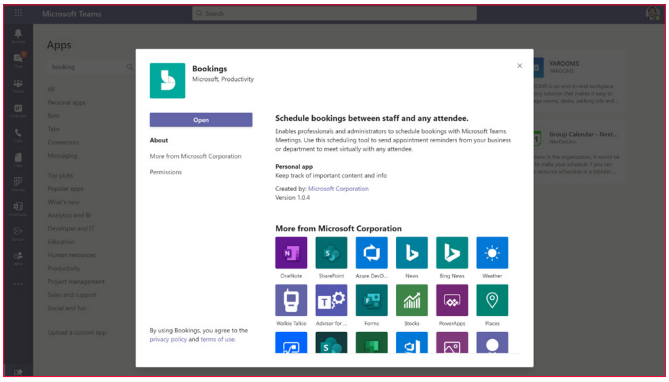


Imagen 14.- App de Bookings en el cliente de Teams.

Las principales funcionalidades de la App de Bookings para Microsoft Teams son las siguientes:

- Por defecto se muestra uno de los Calendarios de Bookings definidos ya que desde la App es posible gestionar distintos calendarios desde Bookings. En ese calendario podremos ver las citas reservadas y la disponibilidad del Staff.

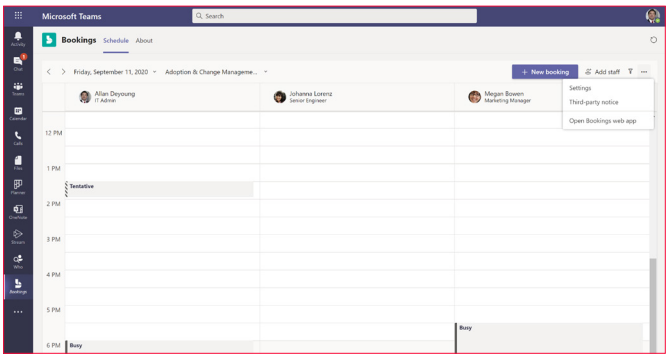


Imagen 15.- Calendario de Bookings en Teams.

- Desde la App podremos modificar algunas de las configuraciones de Bookings (no todas), añadir Staff o simplemente reservar una cita.

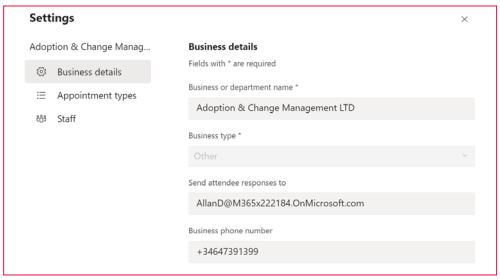


Imagen 15.- Acceso a las configuraciones de Bookings en la App de Bookings en Teams.

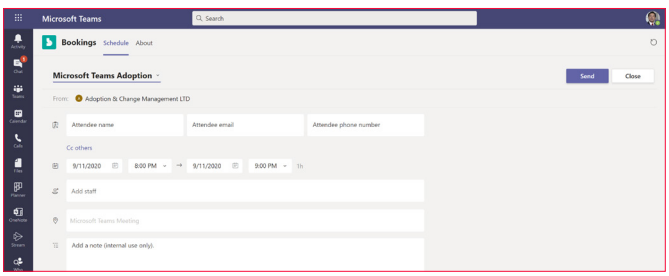


Imagen 16.- Creación de una Cita de Bookings desde la App de Teams.

Si os preguntáis si hoy hay paridad entre la App Web de Bookings y la App de Teams, la respuesta es que no y en la siguiente Tabla podéis ver las principales diferencias a nivel de funcionalidad entre ambas Apps.

CARACTERÍSTICA	APP WEB DE BOOKINGS	APP DE BOOKINGS EN TEAMS
Creación de nuevos calendarios de Bookings	Sí	Sí
Añadir / Eliminar staff de un calendario de Bookings	Sí	Sí
Crear nuevos tipos de citas	Sí	Sí
Planificar citas online	Sí	Sí
Editar los detalles de negocio	Sí	Sí
Añadir Staff con permisos de invitado para personas de fuera de la organización	Sí	No
Planificar citas In-Person u Offline	Sí	No
Planificar Bookings de Grupo o de varios clientes	Sí	No
Indicar un logo para la organización	Sí	No
Indicar horas de atención al público	Sí	No
Publicar página de petición de cita en modo autoservicio	Sí	No
Administrar contactos de clientes	Sí	No
Establecer tiempo de descanso para el Staff	Sí	No

Tabla 1.- Diferencias entre la App Web de Bookings y la App de Teams de Bookings.

Conclusiones

Para finalizar este artículo, os dejo algunas de las principales conclusiones relativas a Microsoft Bookings en Microsoft 365:

- Facilita planificar y administrar citas de forma simple y sencilla
- Permite administrar vía web calendarios de reservas y se integra con Outlook y Microsoft Teams para optimizar el calendario del Staff
- Facilita flexibilidad a los clientes para reservar el mejor slot de tiempo
- Permite pasar a un modelo de reuniones virtuales que tienen lugar en Microsoft Teams
- Es una solución ideal para diferentes escenarios y casos de uso en Educación, Soporte a Cliente, Sanidad, etc.

JUAN CARLOS GONZÁLEZ
Office Apps & Services MVP
Microsoft 365 SME & Delivery Manager en RICOH España
@jcgm1978

33

Cómo volcar información desde las Azure Table Storage a Microsoft Lists usando Logic Apps

En este artículo os mostraré cómo podemos volcar la información contenida en una tabla de una cuenta de almacenamiento alojada en Azure a una de las nuevas listas de Microsoft Lists que, como sabéis, son listas de SharePoint.

Antecedentes

Hace poco, estuve en conversaciones con el Cloud Enablement Desk de Microsoft para poder incluir mi compañía (ILUNION IT Services) en el marketplace comercial de Microsoft con una oferta SaaS. El caso es que, para poder hacerlo, un requisito es conectar la oferta con un CRM para volcar los Leads que se vayan generando. En caso de no disponer de un CRM, se puede hacer mediante una tabla de almacenamiento en Azure (Azure Table Storage) que es la opción que yo utilicé.

NOTA: Aquí está el enlace de Microsoft sobre cómo crear esta tabla: https://docs.microsoft.com/es-es/azure/marketplace/partner-center-portal/commercial-marketplace-lead-management-instructions-azure-table

En esa página hay un tutorial sobre cómo crear un flujo de Power Automate para enviar un correo con los registros de la tabla. Tened en cuenta que los conectores de este tipo son Premium.

NOTA: Aquí os dejo un enlace por si queréis ver más información sobre las Azure Table Storage: https://azure.microsoft.com/en-us/services/storage/tables/

Una vez creada la tabla en mi entorno de Azure y asociada correctamente al Marketplace, queda la parte de cómo acceder a la información alojada en esa tabla. Para ver los registros de estas tablas, podemos utilizar la herramienta "Microsoft Azure Storage Explorer" y ahí podremos comprobar que las tablas de Azure Storage almacenan los datos con formato JSON.

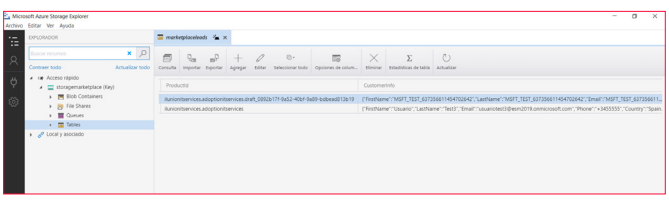


Imagen 1.- Azure Storage Explorer.

Para obtener los datos de una forma más visual y además evitar el tener que acceder recurrentemente a la herramienta del Storage Explorer para ver si teníamos nuevos registros (leads) en la tabla, se me ocurrió el implementar una Logic App en un entorno de Azure que volcase el contenido de mi tabla en una lista de Microsoft Lists. A continuación, os muestro cómo lo he implementado.

Proceso completo

PASO 1: Crear la Lista y la columna PrimaryKey

Para almacenar el contenido y probar las características de Microsoft Lists, lo primero que hice es crear una lista de este tipo en uno de los equipos que utilizo habitualmente en Teams. Para ello, basta con agregar una nueva pestaña en el canal, seleccionar la aplicación de Lists, pulsar en Guardar y después pulsar en 'Crear una lista'. Desde ahí, podréis elegir entre crear una lista en blanco, o escoger una plantilla de las que ya vienen predefinidas:

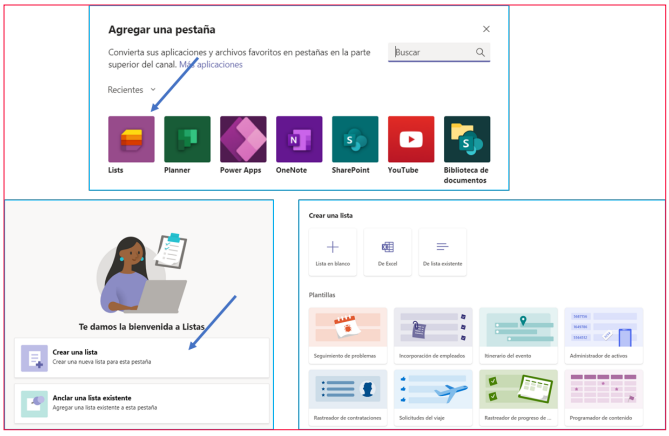


Imagen 2.- Creación de una Lista de Microsoft Lists en Microsoft Teams.

Las listas de Microsoft Lists no son más que listas de SharePoint a las que se les han aplicado una serie de estilos para que tengan un formato más amigable. Así pues, la lista que generé en mi equipo de Teams, se encuentra alojada en la colección de sitios de SharePoint de dicho equipo.

Una vez creada la lista, procedí a agregarle una columna nueva de tipo 'Una línea de texto' que llamé 'PrimaryKey' que me servirá posteriormente para identificar si el registro ya existe en la lista. Para ello, basta con situarse en la última columna de la lista y seleccionar la opción 'Agregar columna'.

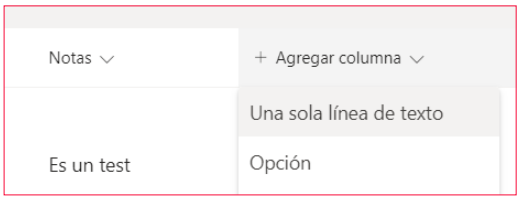


Imagen 3.- Añadiendo una columna a la lista.

PASO 2. Crear la Logic App

Dado que los conectores de Azure Storage de Power Automate son de tipo Premium, decidí utilizar mi entorno Azure para implementar allí una aplicación lógica que se encargase del proceso de volcado de información. Para crear la Logic App, podéis buscar en el buscador del portal de Azure el texto 'Logic apps' y seleccionar el Servicio para acceder a la página correspondiente. Desde allí, podréis ver todas vuestras aplicaciones lógicas que hayáis creado anteriormente y podréis crear una nueva seleccionando la opción Add.

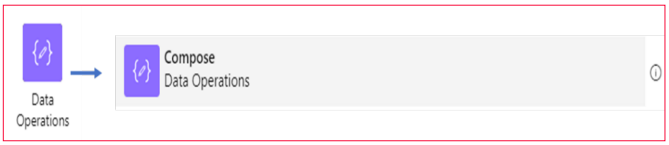


Imagen 4.- Creación de la Logic App.

Una vez pulsado el botón Add, bastará con elegir suscripción, grupo de recursos donde queréis alojarla (en mi caso, el mismo donde tengo la Azure Table Storage), ubicación (región), el nombre que queráis que tenga y pulsáis en 'Review + create'.

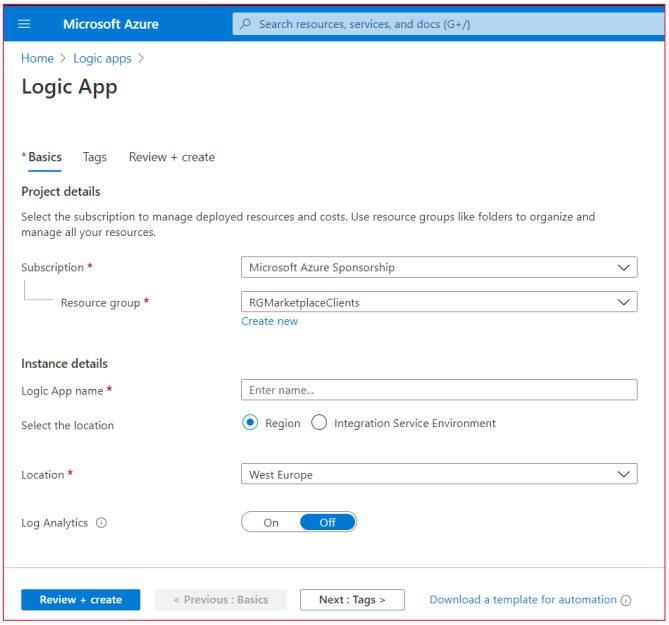


Imagen 5.- Configuración de la Logic App a crear.

Una vez creada la Logic app, podréis comenzar con su configuración utilizando la opción 'Logic app designer'.

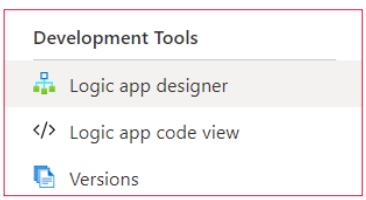


Imagen 6.- Accediendo al Logic app designer.

La primera vez que accedamos, podremos seleccionar alguna plantilla que se adapte a vuestras necesidades o partir de un disparador. También tenéis la opción de partir directamente de cero utilizando la opción Blank Logic App (que es la que yo utilicé)

Imagen 7.- Creación de la Logic App desde cero.

Si es la primera vez que utilizáis las Logic apps, veréis que el diseñador es exactamente igual que el que utiliza Power Automate. No en vano, Power Automate está montado sobre el motor de Logic apps.

PASO 3. Diseño de la Logic App

Una vez en la pantalla del diseñador, configuraremos nuestra aplicación lógica utilizando los siguientes pasos:

TRIGGER (DISPARADOR)

Dado que no hay un trigger que se dispare cada vez que se detecta un nuevo registro en la Azure Table Storage (al menos yo no lo he encontrado), lo que hice es utilizar un disparador de tipo recurrente. Es decir, que se ejecute con una frecuencia determinada. Para ello, buscaremos el disparador de tipo recurrente en la sección Programación.

Imagen 8.- Disparador de la Logic App.

En mi caso, lo configuré para ejecutarse todos los días a las 7 a.m.

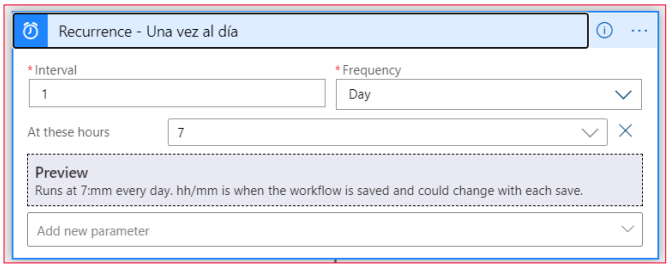


Imagen 9.- Configuración del disparador.

VARIABLES

Las variables no pueden ser inicializadas dentro de los bloques de un flujo. Por esta razón, lo primero que hace el proceso al ejecutarse es inicializar las variables que usaré más adelante. En este caso, utilizo una variable para guardar el cuerpo de una consulta de SharePoint y otra con la dirección de correo donde enviaré los mails de aviso. La opción de inicializar variable la encontraréis en la sección Variables:

Imagen 10.- Variables para la Logic App.

Y las inicializaremos de la siguiente forma:

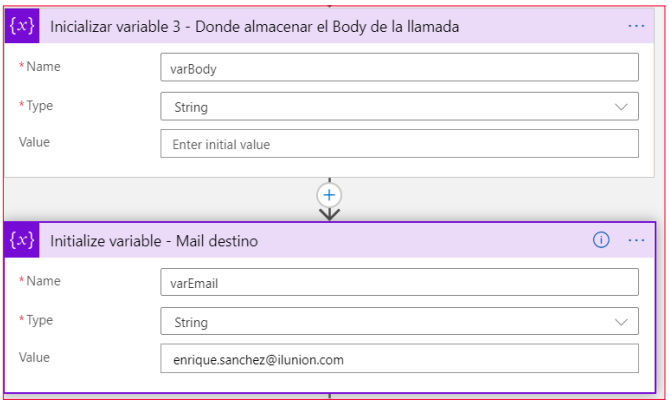


Imagen 11.- Inicialización de las Variables.

ÁMBITO (TRY)

Para poder controlar los errores que se produzcan en la ejecución de nuestro proceso, todos los pasos los vamos a incluir dentro de un contenedor que vamos a denominar Try. Para ello, introduciremos un Ámbito que se encuentra dentro de la sección Controles.

Imagen 12.- Bloque "Scope" (Ámbito).

OBTENER TODOS LOS REGISTROS DE LA TABLA

Ahora es el momento de consultar todos los registros almacenados en mi tabla de Azure. Para ello, dentro de nuestro contenedor Try, agregaremos una acción y buscaremos la denominada como 'Obtener entidades' de la sección 'Azure Table Storage'.

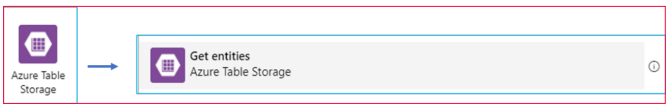


Imagen 13.- Acción Get entities.

Una vez seleccionada, tendremos que hacer la conexión con la tabla. Para ello, pulsaremos en la opción de rellenar la información manual e introducimos los siguientes datos:

- Nombre de la conexión: nombre descriptivo para la conexión que va a establecer entre este flujo y la tabla.
- Nombre de cuenta de almacenamiento: nombre de la cuenta de almacenamiento para la tabla. Puede encontrarlo en la página Claves de acceso de la cuenta de almacenamiento.
- Clave de almacenamiento compartida: valor de clave de la cuenta de almacenamiento de la tabla. Puede encontrar este valor en la página Claves de acceso de la cuenta de almacenamiento.

Tras establecer la conexión, ya podréis seleccionar la tabla en concreto de la que queréis extraer las entidades (registros). En mi caso, la tabla es la denominada marketplaceleads.

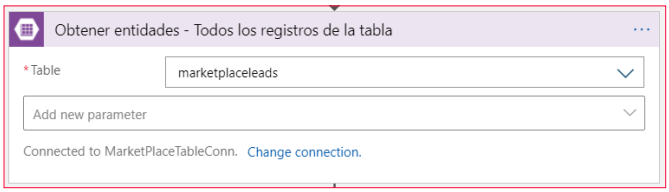


Imagen 14.- Selección de la tabla con los registros a extraer.

Si queréis, podéis establecer algún tipo de filtro para que sólo os devuelva los últimos registros, aunque, en mi caso, como hay pocos registros en la tabla, obtengo todos y, posteriormente, hago la comprobación utilizando la columna PrimaryKey. Para agregar el filtro, podéis utilizar el desplegable 'Add new parameter' y seleccionar 'Filter Query'.

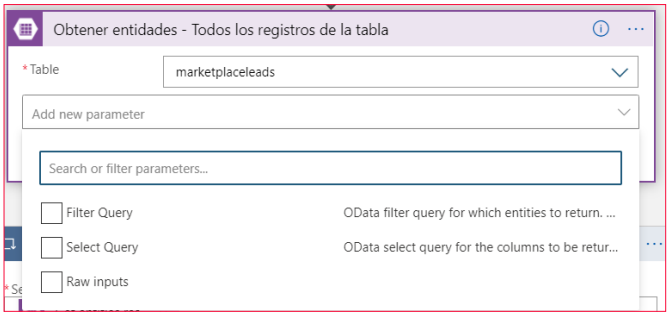


Imagen 15.- Indicando filtros en la consulta a la tabla.

"podemos volcar la información contenida en una tabla de una cuenta de almacenamiento alojada en Azure a una de las nuevas listas de Microsoft Lists"

RECORRER LOS REGISTROS (ENTIDADES) Y OBTENER LA INFORMACIÓN (PARSEO)

Ahora es el momento de recorrer los registros obtenidos de la tabla y verificar si ya lo habíamos insertado en nuestra lista anteriormente. Como la información que tenemos en las Azure Table Storage están en formato JSON, vamos a formatearla (parseo) para poder extraerla en un modo más estructurado. De hecho, la acción de obtener entidades nos devolverá los registros con formato JSON. Por eso, lo primero que vamos a utilizar es la operación de 'Parse JSON' que hay en la sección de 'Data Operations'.

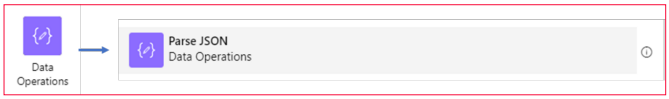


Imagen 16.- Acción de parseo de JSON.

Una vez agregada la acción, en la parte del contenido incluiremos el elemento actual (Current ítem) y en la parte de Schema debemos introducir el esquema de los datos que obtenemos. Para sacar nuestro esquema, podéis hacer una ejecución previa del flujo y en la vista de ejecución, copiar los datos de un registro (sólo uno) del apartado Outputs de la acción de 'Obtener entidades' y pegarlo en la zona de 'Use sample payload to generate schema':

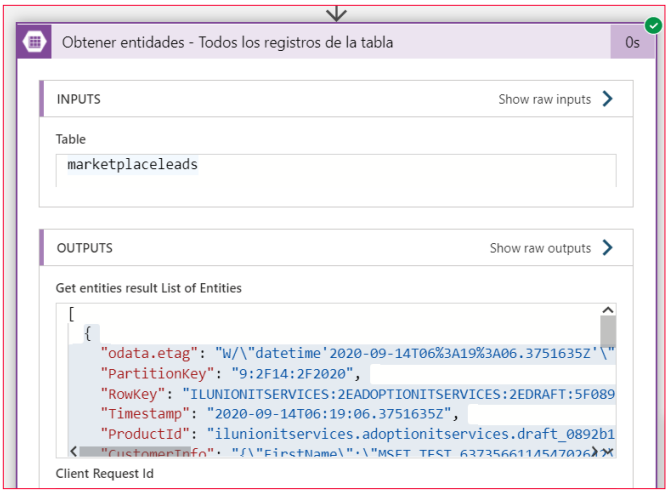


Imagen 17.- JSON a configurar.

Así, la acción quedará similar a esta:

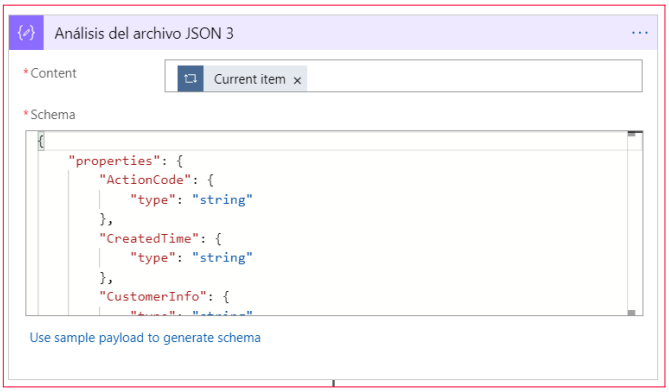


Imagen 18.- Acción configurada.

Ahora que tenemos la información de nuestro registro, queremos obtener los datos del campo "CustomerInfo" que se almacenan en la tabla en formato JSON. Por ese motivo, vamos a hacer un nuevo parseo utilizando un procedimiento como el anterior. Para el esquema utilizaremos el Output de nuestro campo en cuestión (CustomerInfo), de forma que la acción quedará de la siguiente forma:

Imagen 19.- Configuración de nuevo parseo.

Con estas dos acciones, dispondremos de todos los datos de nuestro campo CustomerInfo en un formato legible para poder proceder a la inserción en nuestra lista.

COMPROBAR SI YA HEMOS INTRODUCIDO ESE ELEMENTO EN NUESTRA LISTA.

Ahora que tenemos los campos JSON en un formato adecuado, podemos hacer una consulta a nuestra lista de SharePoint para ver si ya habíamos creado ese elemento con anterioridad. Para ello vamos a utilizar la columna PrimaryKey que creamos en la lista. Esta columna guarda la concatenación de los campos "PartitionKey" y "RowKey" de cada registro, por lo que nos bastará con hacer una búsqueda en la lista utilizando un filtro OData con esa columna. Por tanto, cogeremos la acción de 'Obtener elementos' que hay en la sección SharePoint y establecemos la conexión con nuestra lista de SharePoint:

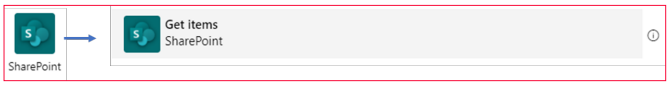


Imagen 20.- Obtención de los elementos de la lista.

Una vez establecida la conexión con nuestra lista, introducimos el filtro de la siguiente forma:

```
PrimaryKey eq '@(trim(body('Análisis_del_archivo_JSON_3')?['PartitionKey']))-@(trim(body('Análisis_del_archivo_JSON_3')?['RowKey']))'
```

De forma que nuestra acción quedará como sigue:

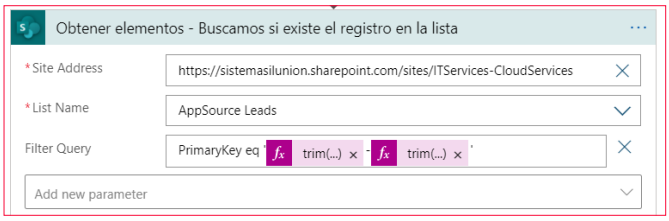


Imagen 21.- Configuración de la acción de Get items.

El resultado de esta consulta lo vamos a almacenar en la variable varBody que inicializamos al comienzo del flujo. Para ello seleccionamos la opción 'Establecer variable' que hay en la sección Variables.

Imagen 22.- Establecer variable.

Seleccionamos nuestra variable y le asignamos el valor del body de la consulta anterior (List of ítems)

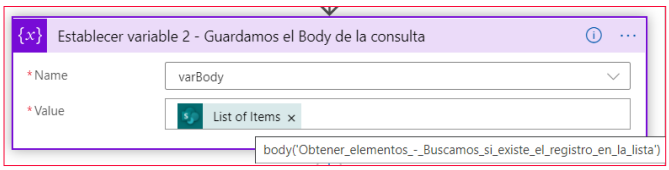


Imagen 23.- Inicialización de variable.

Una vez que tenemos el cuerpo de la respuesta en nuestra variable, es el momento de volver a ejecutar un parseo de esa variable para poder leer su contenido que se recibirá en formato JSON. Así pues, utilizaremos de nuevo la operación de 'Parse JSON', pero ahora con la variable varBody y sacando el esquema del Outputs de un lanzamiento previo:

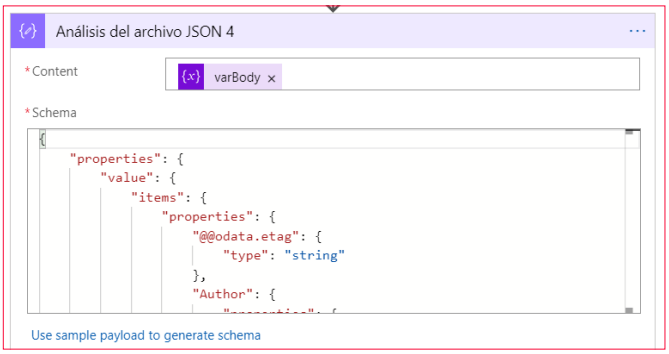


Imagen 24.- Nuevo parseo.

Ahora, utilizando la acción de Redactar que hay en la sección de 'Operaciones de Datos', vamos a poder verificar la longitud del valor devuelto por la consulta para ver si nos ha devuelto algún elemento. Para ello utilizaremos la siguiente expresión: @length(body('Análisis_del_archivo_JSON_4')?['value'])

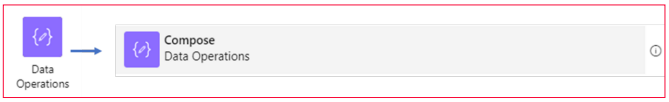


Imagen 25.- Acción compose.

Quedando la operación de la siguiente forma:

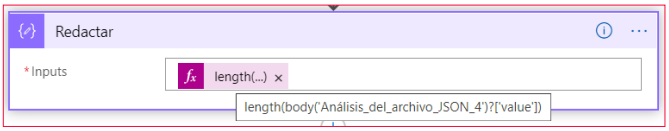


Imagen 26.- Configuración de compose.

Ahora ya podemos establecer la condición de si el resultado es igual a 0, o no. En caso afirmativo, indicaría que es un registro nuevo ya que la consulta no devolvió resultado para esa PrimaryKey. De lo contrario, indicaría que ya existe un elemento en nuestra lista con esa PrimaryKey, por lo que no tenemos que crearlo de nuevo y podremos concluir el proceso para este registro. Por tanto, tendremos la siguiente condición:

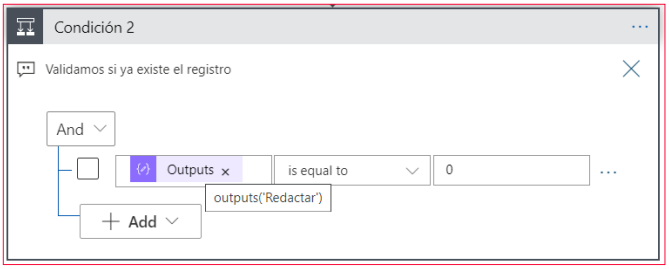


Imagen 27.- Condición.

En caso afirmativo de esta condición, vamos a crear un nuevo elemento en nuestra lista de SharePoint con todos los valores que hemos extraído en los parseos de JSON anteriores. Por tanto, en la rama del True de nuestra condición, agregaremos un paso nuevo utilizando la opción 'Crear elemento' de la sección SharePoint.

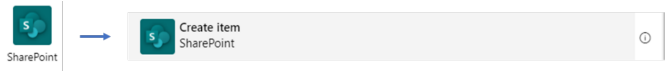


Imagen 28.- Acción de creación de un elemento de lista.

Y estableceremos todos los campos y valores que queremos almacenar. Es muy importante en este paso que establezcamos correctamente el campo PrimaryKey para poder hacer las consultas en próximas ejecuciones. La acción quedará configurada de esta forma:

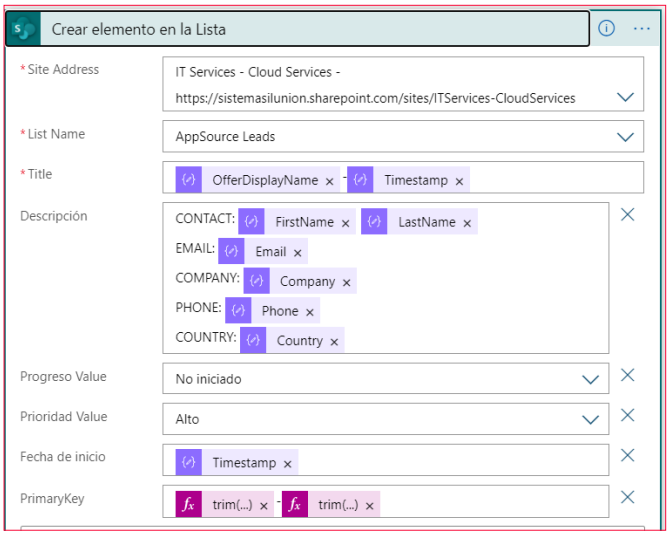


Imagen 29.- Configuración de la acción de creación de elemento de lista.

Ahora sólo nos queda incluir una nueva acción para enviar un correo de aviso de la llegada de este nuevo lead a nuestra tabla. Para ello escogeremos la opción 'Enviar correo (V2)' de la sección 'Office 365 Outlook' y estableceremos la conexión.

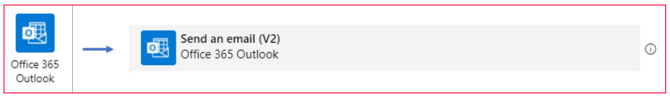


Imagen 30.- Acción de enviar e-mail.

Configuraremos la acción con el texto que deseemos para que quede con un aspecto similar al siguiente:

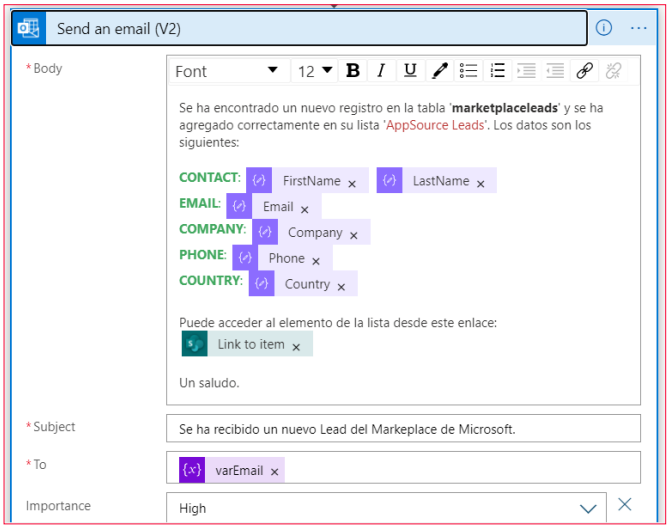


Imagen 31.- Configuración de la acción de enviar e-mail.

Y, con este paso, podemos dar por concluido el proceso de inserción del registro.

ÁMBITO (CATCH)

El último paso del proceso es definir qué ocurrirá si nuestro proceso falla. Para ello, volveremos a agregar un

nuevo ámbito que denominaremos Catch y que estará después del ámbito Try que definimos inicialmente.

Una vez agregado, incluiremos en él las acciones que queramos que se realicen cuando haya un error en el flujo. En mi caso, incluí el envío de un correo de aviso.

Para definir que este ámbito sólo se ha de lanzar cuando haya un error, nos situamos en las opciones del ámbito, seleccionamos la opción de 'Configurar ejecución posterior' y marcamos que sólo se ejecute si al ámbito Try tiene errores (has failed).

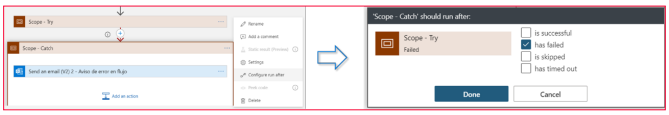


Imagen 32.- Ámbito Catch.

Ya sólo nos falta realizar la ejecución de nuestra Logic App, para comprobar que se ejecuta correctamente y se vuelca el contenido en la lista.

NOTA: Cuando establezcáis las conexiones, es importante que utilicéis usuarios cuyas contraseñas no expiren para evitar tener que actualizarlas cada vez que éstas cambien.

PASO 4. Probar nuestra Logic App

Para lanzar nuestro flujo de la Logic App, basta con pulsar el botón Run ubicado en la parte superior del diseñador

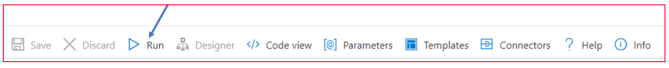


Imagen 33.- Probando la Logic App.

Una vez pulsado, veremos cómo se ejecuta el proceso:

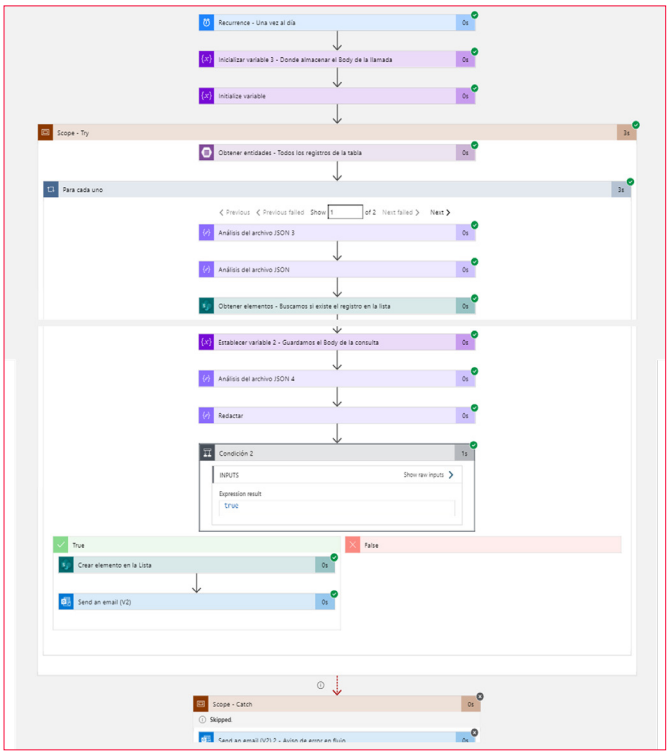


Imagen 34.- Ejecución de la Logic App.

Y una vez finalizado el proceso, podremos ver cómo nuestros registros se han cargado correctamente en la lista de Microsoft Lists:

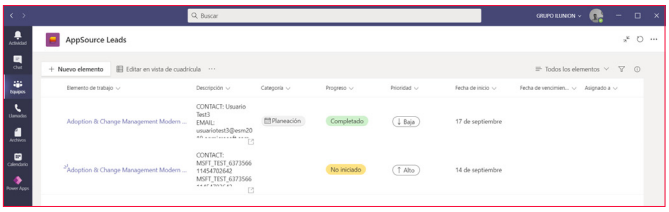


Imagen 35.- Carga de los registros en la lista de Microsoft Lists.

Con esto concluyo el artículo. Espero que os haya resultado interesante y podáis aprovechar la información si alguna vez tenéis una necesidad parecida.

ENRIQUE SÁNCHEZ MORENO
Cloud Services Manager en ILUNION IT Services
www.esanchezm.com

39

Entrevista Aura

Aura IRC, Madrid, España

Aura es una empresa tecnológica, especializada en tecnología Microsoft, posicionada siempre un paso por delante en aplicación de las últimas tecnologías y estándares. No se conforma con conocer las tecnologías del mercado, sino que cree realmente en la transformación de las empresas a través de las tendencias más vanguardistas y novedosas aterrizando conceptos y enfoques con una visión pragmática.

El equipo de Aura lleva más de 20 años liderando proyectos pioneros en innovación y fomentando el cambio digital de las empresas, ayudando así a sus clientes a superar sus objetivos, a mejorar su productividad y su competitividad. Todo ello es posible gracias a su equipo humano que supone su razón de ser y trabajar, valorando por encima de todo las relaciones humanas y el talento y …. siendo el trabajo en equipo el eje de su éxito. Su lema:

" … predice lo que te puede afectar y reacciona antes de que suceda … "

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

Aura es joven como compañía, pero con un equipo que acumula una gran experiencia en el sector tecnológico.

El gran equipo que forma actualmente Aura se debe en parte a la unión de perfiles polivalentes, conocidos ampliamente en el sector y que han aunado su experiencia, basada en distintas trayectorias profesionales en diferentes empresas, que en muchas ocasiones han coincidido en diversas etapas de su evolución, lo que hace que el nivel de confianza y compromiso sea pleno, con objetivo de actualización, trayectoria y trabajo en equipo, Aura ha aunado estas inquietudes personales dando forma a un modelo organizativo y de trabajo distinto.

El perfil que Aura busca para incorporar a su plantilla es un perfil que se ajuste a su forma de trabajo donde prima el compañerismo, la confianza, la transparencia, la implicación y el desarrollo avanzado tanto profesional como personal, cada persona establece a su medida la trayectoria en Aura. Esos son los valores que se necesitan para formar parte de Aura.

¿Cuáles son las principales actividades tecnológicas hoy en día?

Aura se enfoca en las siguientes unidades de especialización:

Modern Workplace: Conexión y apoyo a los diferentes perfiles de una compañía u organización para fomentar la productividad y el compromiso.

Business Applications: Desarrollo de software a medida para proporcionar una funcionalidad empresarial.

Cloud Solutions: Los servicios de infraestructura hacen a las compañías más productivas, rápida adaptación a necesidades y ahorro de costes.

Data & AI: Integración más avanzada en los procesos de negocio con Inteligencia Artificial.

Security: Tecnología al servicio de la seguridad para generar entornos de trabajo seguros.

Transformación Digital: Digitalización de procesos para mejorar la eficiencia en los clientes con aporte de valor.

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

La gran familia de Aura está compuesta por personas apasionadas, con experiencia y mucha dedicación por lo que hacen. Para conseguirlo, desde Aura se apuesta por:

- La identidad: Valora la implicación y el sentido de grupo, concibiendo todo el equipo humano de Aura como

un conjunto global y no por áreas o de forma individual.

- El compromiso: La dedicación en lo que se hace es un requisito fundamental en sus consultores y consultoras para conseguir la calidad, el desarrollo y la mejora continua dentro de Aura.
- El reconocimiento: Aura con su programa de AuraCredits reconoce y premia el esfuerzo y la dedicación de sus equipos en función de acciones llevadas a cabo por sí mismos o acciones que valora la compañía (antigüedad, cumpleaños, felicitaciones de clientes, actividades personales …).
- El apoyo: A través de su programa AuraCare, facilita "cuidado" a la persona, acompañándola desde distintos estamentos o áreas de actividad y con diversas iniciativas desde el mismo proceso de incorporación.
- El trabajo en equipo: Prima la labor conjunta y la integración de todos los consultores dentro de su modelo AuraTeam. La colaboración entre personas es fundamental dentro de la cultura Aura.
- La transparencia y comunicación bidireccional: Aura apuesta por una comunicación bidireccional y transparente entre personas y de éstas con los diversos estamentos organizativos de la compañía de forma periódica.
- La Igualdad: con programas de diversa índole que garantizan la igualdad de oportunidades entre las personas que forman Aura.

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

Desde el comienzo de Aura, apuesta por la especialización dentro de áreas estratégicas y aplicaciones tecnológicas en las empresas, para ello apoya con patrocinio de diferentes eventos sobre tecnología Microsoft, productos o tecnologías en las que Aura es reconocida como puntera, para su difusión y debate de novedades y lanzamientos. Ha colaborado patrocinando eventos de SharePoint, Microsoft365 o SQL entre otros, así como publicaciones técnicas en foros especializados.

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

La tecnología evoluciona cada año más rápido y se redirige continuamente con enfoques disruptivos como se ha plasmado en este último año tan complejo. Vemos aplicaciones tecnológicas en todos los ámbitos y sobre todos los dispositivos, esto está haciendo que las personas estén más conectadas, informadas e incluso realicen sus labores diarias más rápido de lo que lo hacían años atrás, hasta los dispositivos se acelera su interconexión y aporte de información con la llegada del 5G y de sistemas Cloud híbridos; si se añade a la ecuación modelos de aprendizaje en sistemas basados en Inteligencia Artificial las perspectivas para conjugar todo este ecosistema son prácticamente infinitas. Este es el camino hacia dónde va Aura, tecnología como una herramienta o vehículo para mejorar continuamente, ayudando a personas y empresas a ser cada día más eficientes. Aura no cree que exista un límite, cree que hay muchas cosas todavía por ser aplicadas, conjugadas y la tecnología en la que Aura está especializada es la base de todas ellas.

AURA IRC, MADRID, ESPAÑA

41

Evolución de SQL Server en múltiples plataformas

SQL Server “Everywhere”

Haciendo un poco de investigación para este artículo, encontré que hace varios años Microsoft desarrolló una versión de SQL Server inicialmente conocida como “Everywhere Edition”, que luego saco al mercado como SQL Server 2005 Compact Edition. La idea no es hablar de esa versión, que ya ha quedado largamente en desuso. Por el contrario, nos enfocaremos en las distintas plataformas en donde podemos correr nuestro motor de base de datos y explicar porque hoy en día SQL Server corre en todos lados “Everywhere”

Inicios

A mediados de los años 90, cuando empecé a involucrarme con SQL Server en sus versiones 6.0 y 6.5, la única opción disponible era la plataforma Windows. En ese entonces Windows Server 3.5 y Windows NT 4. Todavía recuerdo los enormes servidores NEC y Compaq Proliant donde podíamos tener entornos con discos mecánicos compartidos corriendo las primeras versiones de Windows Clúster para así, poder entrar al mundo de la alta disponibilidad.

Imagen 1.- Versiones originales de SQL Server.

La historia continua de la misma forma por muchísimos años más, ya no con esas versiones “de museo” de Windows y SQL, pero si con las versiones de 2016.

Virtualización y plataformas

Por esa época, la virtualización ya era una realidad, y recuerdo muchas personas que me consultaban por la posibilidad de correr SQL Server en esos entornos virtuales de VMWare y Hyper-V pero mi respuesta no era la que ellos querían escuchar. Hasta esos tiempos mi mensaje fue que debíamos correr SQL Server en un servidor físico dedicado, y a modo de broma y como para reforzar la respuesta agregaba que debíamos tenerlo cerca nuestro para comprobar que genere mucho calor mientras corríamos nuestras cargas de trabajo transaccionales.

> ***"la posibilidad de poder utilizar dicha versión sobre las distribuciones mas importantes de Linux"***

Luego la recomendación paso a ser que “tal vez” podrían considerar estos entornos virtuales para trabajar con ambientes no productivos

La otra consulta que me hacían ocasionalmente era si veía la posibilidad de que alguna vez SQL Server se podría correr en entornos Linux. (dado que la competencia disponía de esta opción) También mi respuesta era negativa, dado que Microsoft siempre había sido sinónimo de Windows

Bueno, la historia cambio radicalmente y hoy en día esas afirmaciones del pasado han quedado totalmente fuera de tiempo y lugar.

La tecnología de virtualización continúo avanzando a grandes pasos, convirtiéndose en la base de la mayoría de las implementaciones productivas, incluso siendo parte fundamental de las plataformas de nube. Hoy en día cuando necesitamos desplegar una solución SQL Server en Azure (u en otra opción de Cloud), lo que obtenemos es una implementación basada en máquinas virtuales

Con la presentación de SQL Server 2017, Microsoft nos dejó a todos gratamente sorprendidos al darnos, por primera vez, la posibilidad de poder utilizar dicha versión sobre las distribuciones mas importantes de Linux (como Redhat , Suse y Ubuntu)

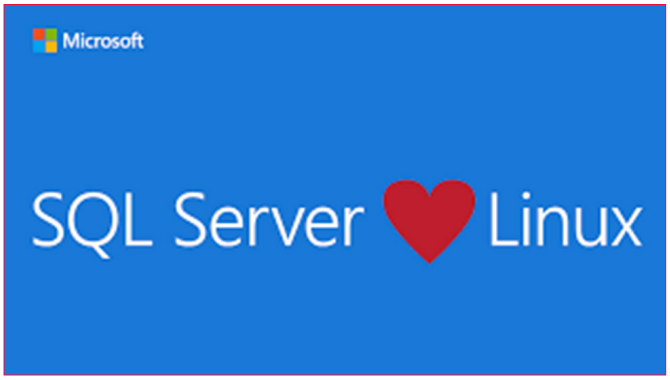


Imagen 2.- SQL & Linux.

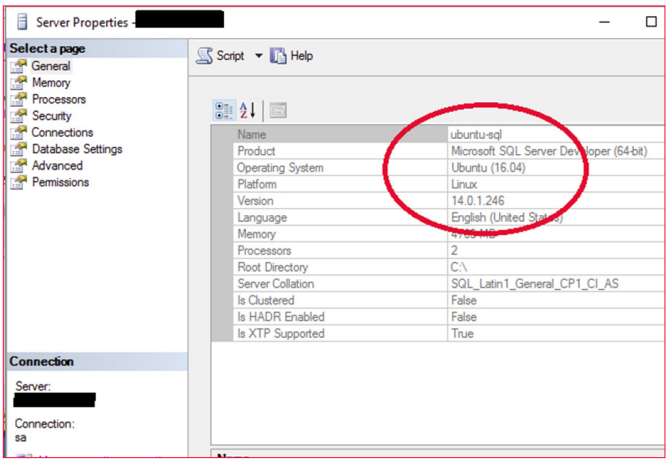


Imagen 3.- SQL Server en Ubuntu Linux.

SQL Server sobre Linux más el avance acelerado de la tecnología de visualización abrió otra gran puerta para tener la posibilidad de tener Contenedores SQL Server "Containers". De esta forma ya no teníamos que virtualizar el "Host o Servidor" sino solamente la aplicación, en este caso nuestro motor de base de datos

Imagen 4.- SQL & Docker.

Así fue como comenzamos a explorar Docker Desktop y tener la posibilidad de hacer un despliegue de un container y poder tener instalado SQL Server en cuestión de pocos minutos. Con Docker Desktop ahora también tenemos otra opción nunca antes posible dado que puede correr en sistemas operativos Mac OS ahora también podemos tener SQL Server en equipos de Apple en formato de containers

"la ultima opción disponible es SQL Server en dispositivos ARM (Como Rasberry Pi)"

La tecnología de contenedores nos dio la posibilidad de comenzar a desplegar entornos con varias aplicaciones virtualizadas (por supuesto incluyendo SQL Server) así que Kubernetes nos dio la posibilidad de orquestar múltiples containers y así tener alta disponibilidad junto con la posibilidad de tener nodos on-premises y en la nube para formar una solución hibrida

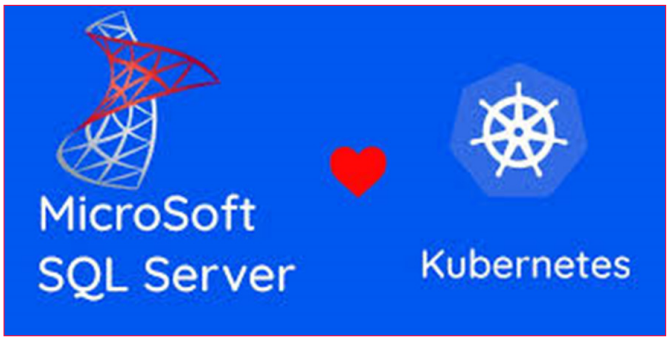


Imagen 5.- SQL & Kubernetes.

Finalmente la ultima opción disponible es SQL Server en dispositivos ARM (Como Rasberry Pi). Esta opción de Azure SQL Edge nos permite tener el motor de base de datos disponibles en dispositivos IoT para poder realizar análisis analítico en tiempo real en escenarios de Edge. Si observamos el sitio donde podemos descargar SQL Server podemos comprobar que ahora no solo podemos seleccionar Windows, sino que tenemos un amplio abanico de posibilidades en relación con la plataforma. Windows, Linux y containers, On-premises o Azure, y la posibilidad de usarlo en dispositivos IoT con Edge.

Imagen 6.- SQL Server como multi-plataforma.

En todos los casos SQL Server es siempre el mismo, por lo que podemos utilizar herramientas como Management Studio, Azure Data Studio, Visual Studio Code e incluso SQLCMD.

Los invito a explorar todas estas opciones

Esta nueva "Microsoft" en donde Linux y Open Source forman parte esencial del Core de la compañía nos deja la puerta abierta para más innovación.

JAVIER VILLEGAS
IT Director Data y BI @ MSC
Microsoft MVP , Data Platform
javier.ignacio.villegas@gmail.com
@javier_vill
/javiervillegas

43

Introducción a Synapse Analytics – integración con Power BI

En este articulo hablaremos sobre esta plataforma de datos en Azure, que aún están en Vista Previa Publica pero que viene pisando fuerte en el mundo de datos a nivel Enterprise. En términos generales Microsoft ha encontrado la manera de generar una atractiva opción para todos aquellos profesionales que trabajan con datos, para mejorar no solo la integración sino la experiencia de usuario.

Conceptos

Comencemos con conceptos básicos de esta nueva plataforma. Al dirigirnos al portal de Azure y utilizando la búsqueda con la consigna de Synapse como palabra clave encontramos:

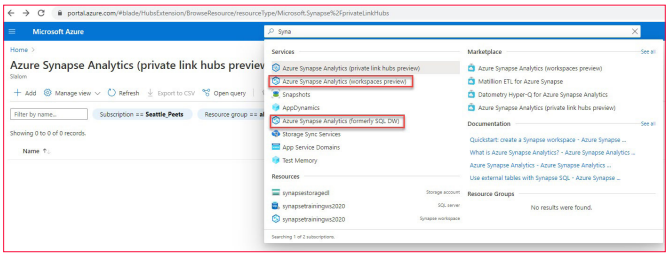


Imagen 1.- Servicios de Azure Synapse Analytics disponibles.

La opción de Azure Synapse Analytics - Workspaces, que es la que nos dará acceso a un portal de manejo general de nuestra plataforma:

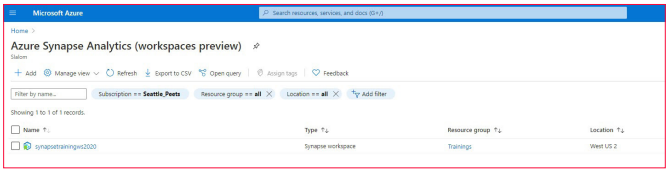


Imagen 2.- Azure Synapse Analytics (workspaces preview).

Por otro lado, la opción de Azure Synapse Analytics – previamente SQL DW nos da acceso a la creación de Synapse SQL pools.

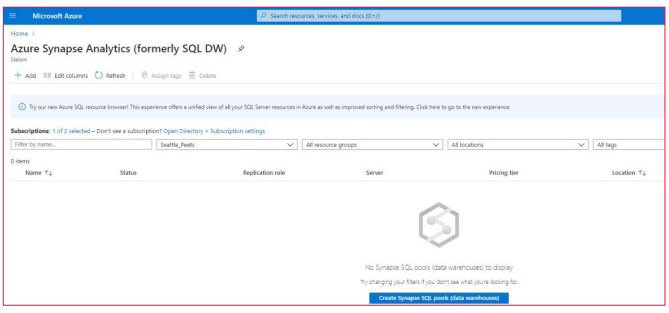


Imagen 3.- Azure Synapse Analytics (formerly SQL DW).

La visión que anteriormente teníamos de la generación de un Datawarehouse moderno era a través de varias capas en la arquitectura como: Ingestión de datos, Preparación, Transformación y Limpieza, Almacenamiento, Servicio y Visualización. Todas ellas se lograban a través de diferentes servicios de Azure como: Azure Data Factory, Azure Databricks, Azure SQL DW, Azure Data Lake Storage, Power BI, como vemos en la siguiente imagen:

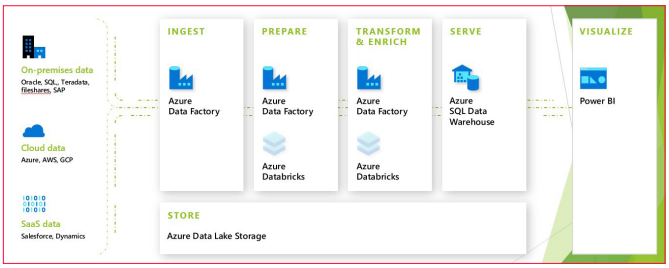


Imagen 4.- Proceso clásico de generación de un Datawarehouse.

Capacidades

El concepto detrás de la nueva interfaz es el de integrar los servicios en una única experiencia de usuario donde accedamos rápidamente para monitorear e interconectar los mismos:

Imagen 5.- Nuevo concepto bajo la interfaz de Azure Synapse Analytics.

De esta manera tenemos un ecosistema integrado de plataforma de datos para cada escenario: Inteligencia Artificial, Machine Learning, Internet of Things, Aplicaciones Inteligentes, Analítica de Datos. A su vez esta nueva interfaz se agregan piezas muy interesantes como la de contar con una plataforma basada en servicios con muy poca necesidad de codificación. En caso de ser necesario codificar además contamos con la opción de agregar código en nuestro lenguaje favorito: SQL, Python, .NET, JAVA, Scala y R.

Otras de las mejoras de esta plataforma es contar con

tecnología serverless: SQL Analytics para procesamiento batch, y streaming, como también Spark para procesamiento de big data con Python, Scala, R y .NET.

La plataforma además cuenta con una instancia de Azure Data Lake integrada y acceso directo Data Verse de nuestra Power Platform.

Otros de los puntos a destacar es SQL On-Demand como experiencia serverless que nos permite auto escalamiento y manejo automático para tener menor costo de mantenimiento y la capacidad de pagar solo y efectivamente por uso (no hay recursos reservados).

Accedemos de una forma rápida a los nodos de SQL donde la performance es óptima para la ingestión de datos. La inferencia de schemas también es un punto para destacar, donde la lectura es automática en una variedad de formatos (CSV, JSON, TXT, Parquet).

En ciertos puntos también cabe la pena destacar la tecnología Multicapa para mejorar la performance en visualizaciones de cara al usuario de negocio:

- Capacidad de acceder a los datos vía DirectQuery.
- Cache ResultSet – pool de cómputo y con elasticidad para manejo de clúster .
- Cache In-Memory.
- Materialized Views: opciones de hacer joins y agregaciones predefinidas de datos para garantizar consistencia transaccional y optimización automática de queries.

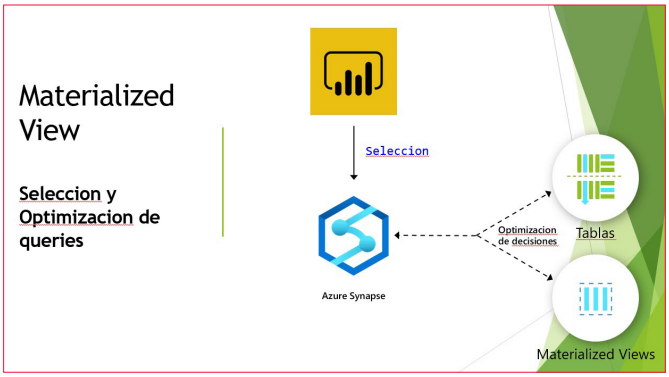


Imagen 6.- Material views en Synapse Analytics.

A continuación, un ejemplo de creación de Materialized Views, que además podremos combinar con modelos composites en Power BI:

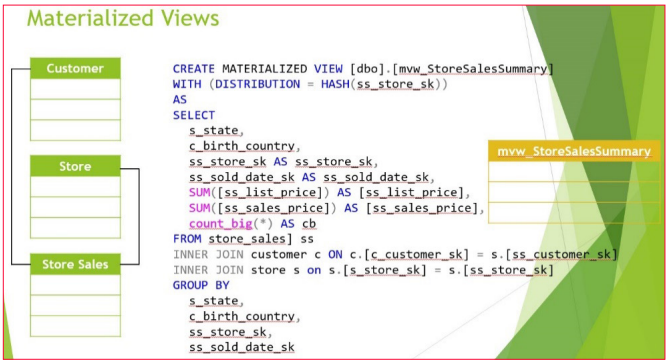


Imagen 7.- Ejemplo de creación de Materlized Views.

Por otro lado, la misma interfaz general de Azure Synapse Analytics nos permite desarrollar pipelines de ingestión de datos, integrando no solo SQL Scripts, Notebooks de Apache Spark, Dataflows, y exportando directamente la información generando datasets de Power BI:

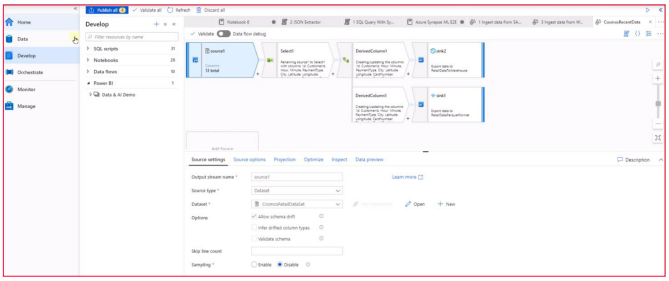


Imagen 8.- Ejemplo de pipeline de ingestión de datos.

Como vemos en la imagen anterior la misma interfaz nos da la capacidad de navegar entre las diferentes capas de acceso a datos (data), de ingestión de datos (develop), orquestación de datos (pipelines) y de monitoreo de la plataforma general (monitor).

"SQL On-Demand como experiencia serverless que nos permite auto escalamiento y manejo automático para tener menor costo de mantenimiento"

Integración con Power BI

Además, contamos con el beneficio de integrar Synapse con Power BI de una forma sencilla, permitiendo crear reportes en un Workspace, actualizando reportes en tiempo real, con la capacidad de visualizar, explorar y analizar los datos:

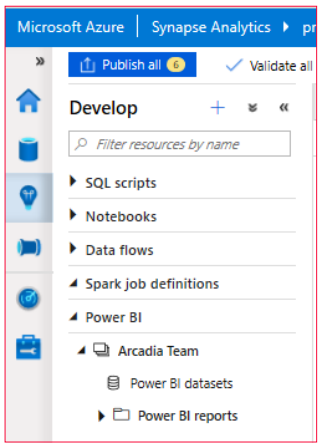


Imagen 9.- Integración con Power BI en Workpsaces de Synapse Analytics.

Como podemos ver a continuación, desde nuestra capa de Desarrollo en Synapse accedemos a un Workspace de Power BI donde podemos explorar nuestros datos en forma directa sin salir de la plataforma permitiendo que el usuario pueda ver en tiempo real el impacto de las transformación, limpieza e ingestión de datos desde diferentes fuentes:

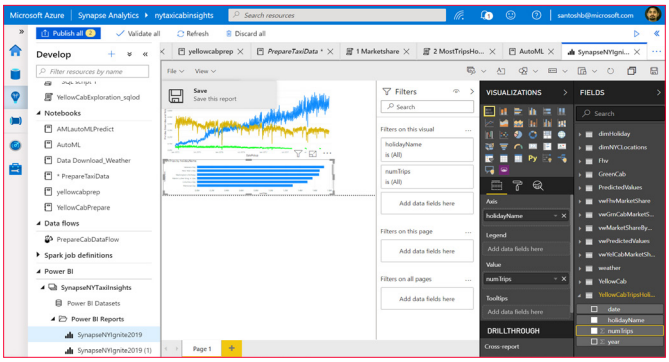


Imagen 10.- Ejemplo de integración con Power BI en Workpsaces de Synapse Analytics.

De la imagen anterior se desprenden varias conclusiones interesantes al momento de trabajar en Synapse: No solo trabajar en la conexión de diferentes servicios como Power BI, otras fuentes de información, sino la capacidad de trabajar con SQL scripts para la manipulación de datos, Notebooks en nuestro Spark pool para la preparación de datos, dataflows para limpieza de datos, y al conectarnos con un área de trabajo de Power BI la directa visualización de nuestro conjunto de datos, hasta el punto de generemos un reporte y lo visualicemos de forma inmediata.

Conclusión

A modo de conclusión, se vienen muchas funcionalidades a esta plataforma que busca de esta manera integrar el trabajo de Arquitectos, Ingenieros de Datos, y Científicos de datos en una interfaz muy amigable y que está en constantes actualizaciones.

GASTÓN CRUZ
Data Platform MVP | Solution Principal at Slalom

46

Ahorrando código: Funcionalidades de C# para no Programar (de) más (pero programar mejor)

En un artículo anterior explorábamos una forma de hacer llegar conceptos de programación básicos y avanzados a programadores que están en formación (http://www.compartimoss.com/revistas/numero-44/el-comic-como-medio-de-ensenanza-de-conceptos-de-programacion). En este artículo también pretendemos hacer llegar conceptos de programación a un sector de programadores, pero con la diferencia de que están enfocados al único propósito de ahorrar escribir código y que nos dirigimos a personas con una determinada experiencia. Y es que muchos programadores, especialmente si llevan mucho tiempo en el mercado, adquieren hábitos de uso a la hora de codificar que, bien por desconocimiento o por inercia, son muy difíciles de desterrar aun cuando existen en el lenguaje alternativas mejores para hacer el mismo trabajo. C# no es una excepción.

Es importante aclarar que no estamos diciendo que se hagan cosas mal, sino que se invierte más tiempo y esfuerzo del necesario, ya que un adecuado conocimiento de nuevas características de un lenguaje permitiría realizar ciertas tareas con una inversión considerablemente menor. Este artículo pretende hacer un recorrido por algunos de los casos que creemos más típicos en los que hemos visto que ocurre, destacando tanto elementos del lenguaje que ya tienen unos años como otros mucho más modernos y, por tanto, potencialmente más desconocidos por un mayor número de personas.

Este artículo se divide en dos partes: formas de ahorrar código a la hora de hacer declaraciones de clases y variables, y formas de ahorrar código en cuerpos de métodos. En donde sea aplicable, se proporciona un ejemplo de "antes" y "después" que trate de ilustrar la técnica y facilite que pueda aplicarse en otros contextos. Todas las pruebas mostradas se han creado bajo un Visual Studio Community 2019 Preview (versión 16.8.0 Preview 4.0) con un proyecto que usa la última versión disponible de .Net 5.0 en el momento de la escritura de este artículo (octubre 2020).

Ahorrando Código en Declaraciones de Tipos

Getters y setters: encapsulando "sin traumas"

Muchos programadores en C#, especialmente si vienen del mundo Java, están acostumbrados a generar métodos getter y setter para los atributos de sus clases en función de cómo quieren que estos sean encapsulados. Esto hace que el código de las clases se llene de pequeños métodos, dificultando su mantenimiento, además de invertir tiempo en una tarea tediosa y repetitiva. Para paliar estos problemas, se necesita la ayuda del IDE (autogeneración de estos métodos) o bien, en caso de Java, de librerías externas como el proyecto Lombok (https://projectlombok.org/).

C# soluciona este problema dentro del propio lenguaje, con una serie de alternativas de generación de métodos get / set de uso muy sencillo y adaptado a diferentes requisitos de encapsulación, gracias al uso de propiedades. En la imagen 1 vemos algunas aproximaciones que ahorran escribir código, destacando una nueva de C# 9 que permite propiedades de solo lectura y de escritura solo en inicialización de la instancia de la clase (init), un caso de uso muy común.

```
class Computer {
    // Propiedad de lectura y escritura estándar, con métodos autogenerados
   public string Brand {get; set;}

   //Propiedad de solo lectura, como expresión (C# 6.0+), necesita menos código
     public string ID => $"Standard Computer {Brand} ({CPU}, {Mhz})";

   /*    Si es necesario incluir una lógica especial en los get/set, hay que crear una propiedad privada de respaldo */
   private double _mhz;
   //Miembros con forma de expresión (C# 7.0+)
   public double Mhz {
      get => _mhz;
      set => _mhz = (value>0)? value: throw new ArgumentException ("...");
   }

    //Esta propiedad se puede leer. Escribir solo está permitido en constructores.
   public string CPU {get; init;}
}
```

Poner tipos está "pasado de moda"

A la hora de crear nuevas instancias de objetos, la forma de creación estándar típica Computer computer = new Computer ("Ryzen 1700"); admite variantes que requieren escribir menos código. Esto es especialmente útil en clases con nombres largos y complejos. Por ejemplo:

- Desde C# 3 en adelante, el uso de var deja que el compilador infiera el tipo de la declaración: var computer =

new Computer (“Ryzen 1700”);

- Como nueva característica de C# 9, el tipo ahora puede omitirse del lado derecho de la asignación también, lo cual tiene unos efectos de ahorro de código interesantes que se pueden ver en la imagen 2:

```
public static Computer ConfigureComputer(string configura-
tionName, Computer baseComputer) {
    ...
}
...
//Se omite el tipo en la creación, pues ya se conoce (C# 9+)
Computer computer = new ();
//Se puede combinar con propiedades tipo init
Computer computer2 = new() {CPU = „Ryzen 2700“};
//Se puede usar para pasar instancias de una clase que se ad-
mita como parámetro
var configured = ConfigureComputer(“Standard_conf”,
new());
```

Métodos “de quita y pon”: Extension methods y partial classes

Una de las cosas que más puede dificultar la mantenibilidad de un código es que una clase crezca desmesuradamente debido a que posea una gran cantidad de atributos y métodos. Si bien esto puede evitarse habitualmente con un buen diseño, en ocasiones no es así, y la clase crece irremediablemente. No obstante, eso no quiere decir que no haya formas de solucionar el problema, y C# provee varias:

- Componer una clase “a trozos”: Mediante la palabra reservada partial un programador puede crear partes de una misma clase en varios ficheros, de forma que cada uno de ellos contenga solo aquellas partes de la clase que tengan que ver con una funcionalidad concreta. A la hora de compilarla, todos los “trozos” parciales de la clase se integrarán en una sola, y no habrá diferencia (más allá de tener un código más mantenible) respecto a haberlo hecho en un solo fichero. Esta palabra reservada también se admite en métodos, de manera que un fichero puede declarar la cabecera de un método y otro puede declarar su implementación. Esto último puede ayudar cuando necesitamos tener la certeza de la existencia de un método que se llama desde otro, pero dicho método está en otro fichero al haber usado clases parciales, como se muestra en la imagen 3.

Fichero1

```
public partial class Network {
  public int Mask {get; init;}
  public partial string
      GetNetWorkDescriptor()
=>
        BaseIP + “/” + Mask;
}
```

Fichero2

```
public partial class Network {
  public string BaseIP {get; init;}
  public partial string
      GetNetWorkDescriptor();
}
```

```
...
//Todas las partes de la clase están disponibles sin problema
var n = new Network () {BaseIP = “156.45.34.0”, Mask = 24};
Console.WriteLine(n.GetNetWorkDescriptor());
```

- Extender la clase externamente: Otra de las formas de disminuir el tamaño de las clases particionando su contenido son los extension methods. Estos permiten “añadir” métodos a una clase fuera de su declaración, en un fichero aparte, sin que en el momento de usarlos el programador note alguna diferencia respecto a los métodos que la clase realmente declara. No obstante, a la hora de usarlos hay que tener en cuenta una serie de consideraciones:
  - Deben ser métodos estáticos, declarados en clases estáticas y cuyo primer parámetro sea un objeto del tipo de la clase que extiende, precedido de la palabra reservada this (ver imagen 4).
  - Aunque a efectos de uso no es así, a efectos de creación funcionan como métodos externos a una clase. Esto quiere decir que no se tiene acceso directo a propiedades no públicas (habría que usar los métodos de acceso disponibles), no pueden sobrescribir métodos declarados en las clases y, si el fichero donde se declaran pertenece a un ensamblado/namespace distinto al de la clase principal y este no se referencia o se usa, entonces no estarán disponibles.
  - Las limitaciones anteriores sin embargo hacen que puedan declararse extension methods de cualquier tipo de C# existente, incluidos tipos básicos predefinidos (int, string…). La posibilidad de que no estén disponibles no es algo necesariamente malo: si una funcionalidad no es necesaria, no debería estar accesible. Por ejemplo, en el caso de Linq, si no hacemos uso del namespace System.Linq ninguna clase que implemente IEnumerable tendrá acceso a los métodos de esta librería, ya que están implementados como métodos de extensión de esta interfaz.

```
//No se declara dentro de class Network, sino en otra clase
aparte
public static class NetworkExtensions {
  public static string GetDHCPServerIP (this Network network)
  {
    return network.BaseIP.Replace(“.0”, “.1”);
  }
}
...
//Indistinguible de un método declarado
var dchp = n.GetDHCPServerIP();
```

¿Que declare mi clase? No, yo soy un tipo anónimo

En ocasiones necesitamos tener objetos de forma temporal para recoger resultados, hacer composiciones de objetos existentes y otras acciones que, debido a su carácter transitorio, ocasional y temporal, quizá no merezca la pena que tengan una clase declarada solo para ellas.

El motivo es que, si tenemos muchos casos así, se incrementaría el tamaño del código fuente de la aplicación y su complejidad, dificultándose su mantenibilidad debido a la existencia de clases que solo se usarían una o muy pocas veces. Esto es algo que se puede dar frecuentemente cuan-

do se recogen resultados de expresiones Linq (de las que hablaremos más abajo).

Para estos casos C# tiene la opción de crear tipos anónimos: objetos de clases sin nombre que se crean en un punto del programa para algo temporal, y que pueden estar compuestos de los miembros que queramos o nos hagan falta en un momento dado. De esta forma, es mucho más fácil crear algo que nos dé servicio en un punto concreto del programa y que no tendrá vida más allá del mismo. Los tipos anónimos se comportan como clases estándar (el acceso a sus miembros se hace de forma idéntica) puesto que por debajo no son más que clases AnonymousType'<Numero> que el compilador crea según sea necesario. De esta manera, usando los ejemplos anteriores, si en un momento dado nos hiciera falta asociar un ordenador con su red para hacer alguna operación concreta, podríamos hacer algo como lo que se muestra en la imagen 5.

```
/* Tipo anónimo creado en este punto. Los nombres de los atributos pueden omitirse y se usaría el nombre de la instancia usada para darle valor (var redPC = new {n, computer2} sería valido) */
var redPC = new {
  Network = n,
  Computer = computer2,
};
...
Console.WriteLine(redPC.Network + ": " + redPC.Computer);
```

Registros (records): clases "prefabricadas" (para ciertos contextos)

Los record types de C# son un añadido muy reciente de la versión 9 del lenguaje. Son básicamente clases (tipos referencia), pero con semántica de tipos por valor por defecto. Esto quiere decir que deberíamos usarlos de manera que sus atributos sean de solo lectura y que solo puedan inicializarse en construcción (pudiendo usar el modificador init visto anteriormente para sus propiedades). En otras palabras, una vez creados, sus atributos no deberían cambiar de valor.

Los tipos inmutables evitan tener efectos laterales en la computación, lo cual les hace perfectos para ser usados en escenarios donde se use programación funcional pura y código concurrente. En este último caso, evitarían el uso de mecanismos de sincronización que perjudicarían el rendimiento global del programa.

¿Cómo se gestionan estos tipos entonces? Pues al ser básicamente clases, su uso no es muy distinto al estándar. No obstante, dado su carácter inmutable, hay que tener en cuenta una serie de propiedades que poseen. La ventaja en el contexto de este artículo de los tipos record es que, si quisiéramos conseguir estas mismas propiedades con clases estándar (lo cual es un caso de uso bastante común), tendríamos que programar código adicional. Los tipos record nos ahorrarían todo ese código.

- Sus propiedades deberían declararse como get; init; (solo lectura, inicialización en el constructor)
- Si tenemos la necesidad de cambiar el valor de alguno de sus atributos, mantener su semántica inmutable requiere la creación de un nuevo record con esos valores cambiados, quedando el objeto original inalterado. Esto se simplifica en C# 9 mediante el uso de la palabra reservada with como se ve en la imagen 6:

```
public record ComputerUser {
  public string Name {get; init;}
  public string Department {get; init;}

  public ComputerUser(string name, string department)
      => (Name, Department) = (name, department);
}
...
var user2 = new ComputerUser("C#erlock HolMS", "IT");

/* Creamos un nuevo usuario user3, que tiene los mismos valores que user2 salvo los de las propiedades que enumeremos aquí. Se admiten varias propiedades separadas por comas */
var user3 = user2 with {Department = "Accounting"};
```

- Se implementa por defecto una semántica de comparación por valor, es decir que, a diferencia de los objetos de las clases tradicionales que no redefinen su propio Equals, dos registros son iguales si los valores de todas sus propiedades son iguales. Nótese que esto nos ahorra la implementación tanto de un método Equals típico como de un método HashCode, ya que el lenguaje lo hace por nosotros. Por ejemplo, dada la clase anterior, vemos en la imagen 7 como la comparación de los record types devuelve true cuando los valores de todas las propiedades de los objetos son iguales, aunque sean objetos distintos:

```
var user1 = new ComputerUser("C#erlock HolMS", "IT");
var user2 = new ComputerUser("C#erlock HolMS", "IT");
var user3 = user2 with {Department = "Accounting"};
Console.WriteLine(user1.Equals(user2)); // Imprime true
Console.WriteLine(user3.Equals(user2)); // Imprime false
```

- Se implementan automáticamente métodos Copy y Clone, ahorrándonos ese trabajo.
- Adicionalmente, se implementa un método ToString que muestra los valores de cada instancia de un record type de una forma clara (ver imagen 8), y que también puede servir para ahorrarnos implementar este método en muchos escenarios.

```
var user1 = new ComputerUser("C#erlock HolMS", "IT");
...
//Imprime: ComputerUser {Name = C#erlock HolMS, Department = IT}
Console.WriteLine(user1);
```

- Finalmente, un último detalle de los record types es su especial relación con las tuplas (tipo de datos introducido en C# 7+). Ya no solo es el hecho de que se admita la asignación de propiedades usando la sinta-

xis de tupla (ver la línea del constructor en la imagen 6), sino que hay una relación aún más estrecha en el caso de que declaremos los record con la sintaxis de positional record. Un positional record es equivalente al que hemos definido, pero usando aún menos código a cambio de tener una serie de comportamientos por defecto. Los miembros de un positional record se tratan automáticamente como si se declarasen con get; init; y en este caso, además de las propiedades vistas anteriormente (ToString, Copy, Clone, Equals…) implementan automáticamente un método Deconstruct que, al igual que con las clases estándar que cuentan con él, permite asignar sus propiedades a una tupla muy fácilmente de la siguiente forma:

```
//Declaración de ComputerUser como positional record
public record ComputerUser (string Name, string Department);
...
//Deconstrucción como tupla automática
var (name, department) = user1;
Console.WriteLine(name + "," + department);
```

Por tanto, podemos ver los tipos record como una forma rápida de implementar clases típicas que necesitamos que tengan una semántica inmutable (o que el hecho de que la tengan no conlleva ninguna penalización o problema). Esto nos da automáticamente una serie de operaciones y semántica que nos ahorraría implementar bastante código que tradicionalmente se añade a las clases que necesitan este conjunto de funcionalidades (especialmente en el caso de los positional records). En el resto de los sentidos, los records funcionan como clases estándar: admiten herencia (pueden ser marcadas como "sealed" para prevenirlo), permiten la redefinición de métodos, métodos extensores…

Ahorrando Código en Cuerpos de métodos

¿Por qué sobrecargar con varios métodos si solo puedes tener uno?

En ocasiones nos vemos obligados a implementar varios métodos con un mismo nombre, pero diferente número y tipo de parámetros. Esto nos lleva a un método con un gran número de sobrecargas que, salvo que seamos muy cuidadosos, puede hacernos repetir trozos de código y tener potenciales problemas de mantenibilidad. Esto puede evitarse en C# si usamos el mecanismo de paso de parámetros con valores por defecto, que un parámetro tomará si hacemos una llamada sin dar un valor a la posición que ocupa. En la imagen 10 podemos ver las dos formas de hacerlo, y en ambos casos Method1 se puede llamar con 1, 2 o 3 parámetros.

```
public string Method1 (int p1) {
  ...
}
public string Method1(int p1, int
p2) {
  return null;
}
public string Method1(int p1, int
p2, int p3) {
  return null;
}
```

```
public string Method1B (int p1, int
p2=0, int p3= 0) {
  return null;
}
```

¿De verdad hace falta todo esto para empezar un programa?

Otra de las cosas nuevas que ha incorporado C# 9 es la necesidad de escribir una serie de código que tradicionalmente se usaba para crear el punto de entrada Main de un programa. Ahora el compilador admite top-level statements, de manera que un solo fichero de toda la solución puede usar sentencias que estén en el ámbito global, sin pertenecer a ningún método, clase o namespace. Dado que esta característica está pensada para hacer la declaración del Main más sencilla, este código tiene acceso a una variable args (array de string con los parámetros pasados al Main, ver imagen 11) y en caso de que devuelva un entero se usará como valor que se suministra al sistema operativo.

```
public static class Application {
  public static int Main (string
args) {
    Console.WriteLine(args);
    ...
    return 0;
  }
}
```

```
Console.WriteLine(args);
```

¿Comprobar null antes de acceder a una variable? Sí, pero más fácil

Una de las fuentes de errores en tiempo de ejecución más típicas a la hora de programar es la falta de comprobación de si una determinada variable es null antes de acceder a su valor. Si bien esto tradicionalmente se cubre mediante if a modo de precondición, la posibilidad de olvidarse de hacer estas comprobaciones puede llegar a ser frustrante y tediosa, además de hacer el código más grande si es algo que se comprueba a menudo. Para evitar estos casos, C# contempla dos opciones:

- El operador ?, que hace que si la variable a la que está asociada tiene el valor null entonces el resultado de la expresión es también null, lo que evitaría problemas a la hora de acceder a su valor en ciertos escenarios típicos.
- El null-coalescing operator (??), que retorna el valor de la parte izquierda si no es null y, en caso contrario, evalúa la parte derecha y retorna el valor obtenido en dicha evaluación. Hay que tener en cuenta que este operador no evalúa su parte derecha si la parte izquierda no es null.

La imagen 12 ilustra el uso de ambos. En ambos casos se pueden lograr comportamientos aceptables sin falta de ha-

cer manualmente comprobaciones previas con if.

var user1 = new ComputerUser("C#erlock HolMS", "IT");
var user2 = new ComputerUser("C#erlock HolMS", "IT");
 	
//Imprime el nombre
Console.WriteLine(user1?.Name);
user1 = null;
//Imprime null
Console.WriteLine(user1?.Name);
	user1 = new ComputerUser(null, "IT");
	//Imprime el nombre del departamento
	Console.WriteLine(user1.Department ?? "Department not defined");
	
	//Imprime "Name not defined"
	Console.WriteLine(user1.Name ?? "Name not defined");

No a los mini-métodos: Expresiones lambda

Una de las situaciones que más podría "ensuciar" el código de una aplicación viene derivada de la existencia de una serie de métodos de tamaño muy pequeño (una línea o poco más), cuya finalidad es usarse para determinadas operaciones de manera muy esporádica. En estas situaciones, lo mejor es optar por eliminar estos métodos del lugar donde se encuentran declarados y convertirlos en expresiones lambda. Las expresiones lambda son algo así como "pedazos de código" que pueden usarse a voluntad en aquellos puntos donde sean necesarios. Esto crea una filosofía de trabajo más ágil y mantenible, puesto que solo estarán declarados aquellos elementos de la clase que tengan un uso más constante (e importante) dentro de un programa. Cualquier método puede convertirse a una expresión lambda siguiendo el procedimiento indicado en la imagen 13. Aunque todos los pasos intermedios mostrados en la misma llevan a código válido ejecutable, lo mejor es quedarse siempre con la evolución final (la expresión lambda).

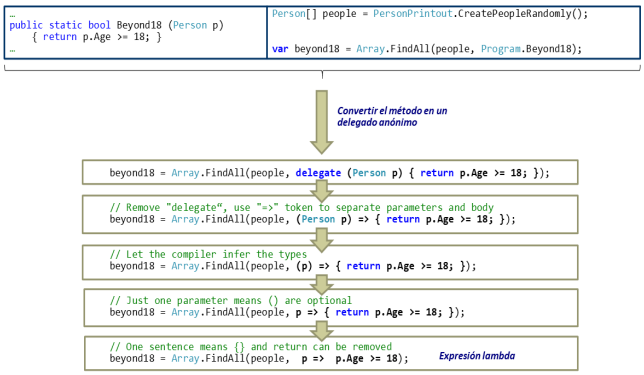


Imagen 1.- Conversión de un método en una expresión lambda.

El uso frecuente de expresiones lambda introduce el paradigma funcional en nuestro código, lo que puede usarse para ahorrar código en muchos otros escenarios que se listan a continuación. En C# 9 se ha introducido además la posibilidad de usar _ para descartar algún parámetro que no tenga sentido en algún uso particular de una expresión lambda, de forma que, aunque un parámetro no se use en un momento dado, la expresión siga siendo compatible con un delegado que las contenga determinado (ver https://docs.microsoft.com/es-es/dotnet/csharp/language-reference/operators/lambda-expressions).

¿Comparadores? ¿Equals? Eso es muy "clásico"

Como se ve en la imagen siguiente, el uso en este contexto de expresiones lambda junto con la librería Linq convierte en una línea de código algo que antes necesitaba una clase auxiliar y una instancia de esta.

Forma Orientada a Objetos
class Person {
public string Name {get; set;}
public string NIF {get; set;}
public override string ToString() {
return Name + "; " + NIF;
}
}
Tenemos que implementar una clase que implemente IComparer distinta por cada criterio de comparación usado (Nombre, Apellidos...):
class PersonByNameComparator : IComparer<Person> {
public int Compare(Person x, Person y) {
return x.Name.CompareTo(y.Name);
}
}
...
Array.Sort(persons, new PersonByNameComparator());
	Forma Funcional
	(solo tenemos que indicar el criterio de ordenación como una expresión lambda)
	var sorted = persons.OrderBy(person => person.NIF);

¿Bucles? ¡Eso es muy orientado a objetos!

Existen formas de sustituir operaciones típicas que se realizan con bucles tradicionalmente en equivalentes usando programación funcional que cubren las mismas necesidades. Esto es gracias a las funciones de orden superior Select, Where y Aggregate, que son los nombres en C# de las clásicas implementaciones de Map, Filter y Reduce de la programación funcional. En los siguientes ejemplos se pueden ver varios usos típicos a efectos ilustrativos de cómo se podría hacer la sustitución de un código orientado a objetos clásico en uno equivalente funcional. Por ejemplo, en la imagen 15 se hace para convertir el conjunto de números naturales menor que 100 a string.

"lo mejor es optar por eliminar estos métodos del lugar donde se encuentran declarados y convertirlos en expresiones lambda"

Forma Orientada a Objetos
var numbers = OOPNumberGenerator(100);
var temp = new string[numbers.Count()];
for(int i = 0; i<100; i++)
temp[i] = numbers[i].ToString();
Show(temp);
	Forma Funcional
	Show(LazyNumberGenerator(100).Select(number => number.ToString()));

La imagen 16 ilustra cómo extraer de una colección aquellos elementos que cumplan con una determinada condi-

ción, en este caso, que sea un número primo.

Forma Orientada a Objetos	Forma Funcional

numbers = OOPNumberGenerator(100);
var tempInt = new int[numbers.Length];
int counter = 0;
foreach (var number in numbers)
  if (IsPrime(number))
  tempInt[counter++] = number;
Array.Resize(ref tempInt, counter);
Show(tempInt);
	Show(LazyNumberGenerator(100).Where(number => IsPrime(number)));

Finalmente, la imagen 17 ilustra cómo se puede usar la programación funcional para hacer cálculos sobre los elementos de una colección, en este caso la suma de todos los primos menos que 100.

Forma Orientada a Objetos	Forma Funcional

numbers = OOPNumberGenerator(100);
var result = 0;
foreach (var number in numbers)
  if (IsPrime(number))
  result += number;
Console.WriteLine(result);
	Console.WriteLine(LazyNumberGenerator(100).
	  Aggregate((accum, number) => {
	  if (IsPrime(number))
	  return accum + number;
	  return accum;
	  }));
	Aunque en este caso concreto también se admitiría:
	Console.WriteLine(LazyNumberGenerator(100).
	  Where(n => IsPrime(n)).Sum());

Cabe decir que esta forma de trabajo no solo aporta ventajas a la hora de simplificar el código del programa, sino que también puede traer ventajas de rendimiento: muchas de las funciones Linq se comportan de forma lazy, es decir, solo calculan los elementos a medida que se le van pidiendo. Esto puede hacer que su rendimiento aumente en muchos escenarios (especialmente aquellos en los que no sea necesario recorrer todos los elementos de la colección). Por otro lado, en el cálculo de valores a partir de los elementos de una colección (imagen 17), el uso de tipos anónimos (vistos anteriormente) para recoger los resultados puede simplificar aún más el código de los cálculos.

¡IEnumerable en una línea!

Otro de los escenarios donde la programación funcional nos permite un ahorro sustancial de código es si tenemos que hacer que una determinada clase implemente la interfaz IEnumerable. En lugar de la aproximación clásica, donde hay que implementar una clase auxiliar que tenga una serie de métodos dada, la programación funcional, mediante el concepto de generador, permite hacerlo de una forma muy sencilla haciendo además que el IEnumerable generado siga automáticamente una política lazy, con las ventajas ya mencionadas (ver imagen 18).

Forma Orientada a Objetos	Forma Funcional

class MyList<T>: IEnumerable<T> {
 public IEnumerator<T> GetEnumerator() {
 return new MyListEnumerator<T>(this);
 }
 IEnumerator IEnumerable.GetEnumerator()
  { return GetEnumerator(); }
}
 	
Necesitamos otra clase que implemente IEnumerator<T>:
class MyListEnumerator<T>:IEnumerator<T>{
  public MyListEnumerator(MyList<T> listToEnumerate) {...}
  public void Dispose() {...}
  public bool MoveNext() {...}
  public void Reset() {...}
  public T Current {get;}
  object IEnumerator.Current {
  get {return Current;}}
}
	Usando programación funcional no tenemos que crear otra clase que implemente IEnumerator<T>:
	class MyListEnumeratorFunctional<T>:
	  IEnumerable<T> {
	 public T GetElement(int pos) {...}
	 public int Length {get; init;}
	
	 public IEnumerator<T> GetEnumerator() {
	  for (int i = 0; i<this.Length; i++)
	  yield return GetElement(i);
	 }
	
	IEnumerator IEnumerable.GetEnumerator() { return GetEnumerator(); }
	
	}

Esta misma aproximación se puede usar también para la generación de colecciones de números u otros elementos que sigan un determinado patrón (en la imagen 19, números de 0 a un límite determinado, en el caso funcional además sería lazy).

Forma Orientada a Objetos	Forma Funcional

int [] OOPNumberGenerator (int limit) {
  int counter = 0;
  int [] temp = new int[limit];
  while (counter < limit)
  temp[counter] = counter++;
  return temp;
}
	IEnumerable<int> LazyNumberGenerator (int limit) {
	  int counter = 0;
	  while (counter < limit)
	  yield return counter++;
	}

Patrones para comparaciones avanzadas y ahorro de código (C#9)

Como ejemplo de nuevas características de C# 9 que ahorran código junto con elementos de programación funcional, tenemos los nuevos conjunctive, disjunctive y negated patterns, que básicamente permiten hacer pattern matching con expresiones lógicas and, or y not respectivamente. Estos determinan, por ejemplo, si el valor de una determinada variable cumple con unas determinadas características o encaja con un patrón determinado, pudiendo hacer una expresión lógica con estos encajes. En la imagen 20 se muestra un ejemplo para determinar si una password tiene una complejidad mínima, usándose este tipo de construcciones para simplificar mucho el código frente a su equivalente orientado a objetos.

```
public bool IsAcceptableEnoughPassword (string pwd) {
    return pwd.Count(c => c is >= 'a' and <= 'z' or >= 'A' and <= 'Z') > 5 &&
    pwd.Count(c => c is >= '1' and <= '9') > 3;
}
```

Finalmente, la imagen 21 contiene un ejemplo de uso de pattern matching que haría la misma labor que una estructura clásica if-else anidada, pero usando mucho menos código.

```
enum Cathegory {Bronze, Silver, Gold}
...
public Cathegory DetermineCathegory(int points) => points
switch {
    < 50 => Cathegory.Bronze,
    < 70 => Cathegory.Silver,
    _ => Cathegory.Gold,
};
```

Conclusión

Si bien existen más esquemas que permiten ahorrarse código (la palabra using para ahorrarse cerrar ciertos tipos de objetos (ej.: ficheros) una vez finalicemos de usarlos, los nuevos rangos soportados por el lenguaje...), creemos que en este artículo se ha hecho un recorrido por aquellos que consideramos más útiles en un número de casos más elevados. La intención de dar a conocer este tipo de mecanismos es demostrar que C# es un lenguaje vivo, en constante crecimiento y adquisición de nuevas características (con énfasis en el aspecto funcional y de lenguajes dinámicos), y que el conocimiento de dichas características puede darnos una ventaja sustancial a la hora de mejorar la mantenibilidad y legibilidad de nuestros programas, un mejor uso del tiempo disponible y solventar algunos errores en tiempo de ejecución.

JOSÉ MANUEL REDONDO LÓPEZ
Doctor en Ingeniería Informática
redondojose@uniovi.es
@The_Rounded_Man
https://www.researchgate.net/profile/Jose_Redondo8

ANTONIO PAYÁ GONZÁLEZ
Graduado en Ingeniería Informática
uo251065@uniovi.es
@AntonioPaya22
https://www.researchgate.net/profile/Antonio_Paya_Gonzalez

ALBA COTARELO TUÑÓN
Graduada en Ingeniería Informática
uo251336@uniovi.es
@Albact7
https://www.researchgate.net/profile/Alba_Cotarelo

53

Desarrollando Microsoft Teams Messaging Extensions desde SPFx

En este artículo, vamos a ver cómo podemos reutilizar nuestro conocimiento del SharePoint Framework, para poder desarrollar messaging extensions para Microsoft Teams.

Introducción a MS Teams messaging extensions

Si todavía no sabéis lo que es una messaging extension de Teams, en el siguiente enlace oficial de MS se explica perfectamente:

https://docs.microsoft.com/en-us/microsoftteams/platform/messaging-extensions/what-are-messaging-extensions

Básicamente, messaging extensions nos permite extender Teams con nuestro propio código, tanto a la hora de componer mensajes, como sobre mensajes ya existentes. La siguiente imagen describe claramente en qué puntos podemos incluir nuestras propias acciones:

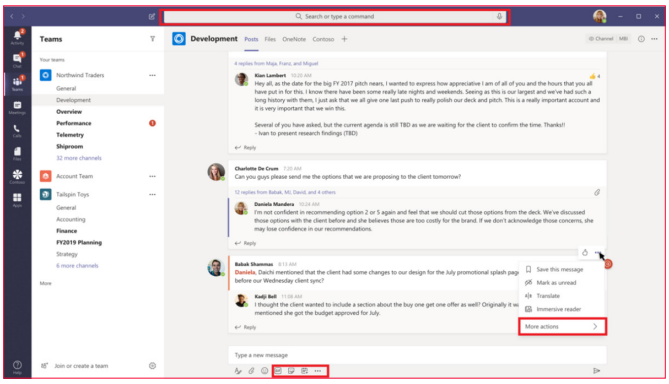


Imagen 1.- Ejemplos de Message Extensions en Microsoft Teams.

Existen 3 tipos de extensiones:

- Comandos de acción: Permiten presentar a los usuarios un elemento emergente modal para recopilar o mostrar información. Cuando envían el formulario, el servicio web puede responder insertando un mensaje en la conversación directamente o insertando un mensaje en el área de mensaje de redacción y permitiendo al usuario enviar el mensaje. Incluso puede enlazar varios formularios entre sí para flujos de trabajo más complejos.
- Comandos de búsqueda: Permiten a los usuarios buscar información en un sistema externo y, a continuación, insertan los resultados de la búsqueda en un mensaje.
- Apertura de vínculos: Permiten invocar a nuestro servicio cada vez que se pega una URL con un dominio dado en la caja de mensaje.

Ahora que ya sabemos lo que son las messaging extensions, viene la buena noticia, y es que para desarrollarlas (si no por completo, una gran parte de ellas), podemos reutilizar lo que ya sabemos de desarrollo SPFx (desde la versión 1.11).

Escenario

Lo que vamos a desarrollar como parte del artículo, será una extensión que estará disponible desde la caja de escribir un mensaje, y que nos listará todos los Teams de nuestra Tenant (bueno, un número fijo limitado, por simplificar), y que una vez seleccionado un Team, incrustará una Adaptive Card en la caja de chat, con la información detallada de dicho Team, para poderla enviar por el chat.

La idea es que puede que esté chateando con algún compañero que anda buscando cierta información, que tú no le sabes dar exactamente, pero le puedes orientar en qué Team (o Teams) puede encontrarla. La siguiente imagen muestran el resultado final:

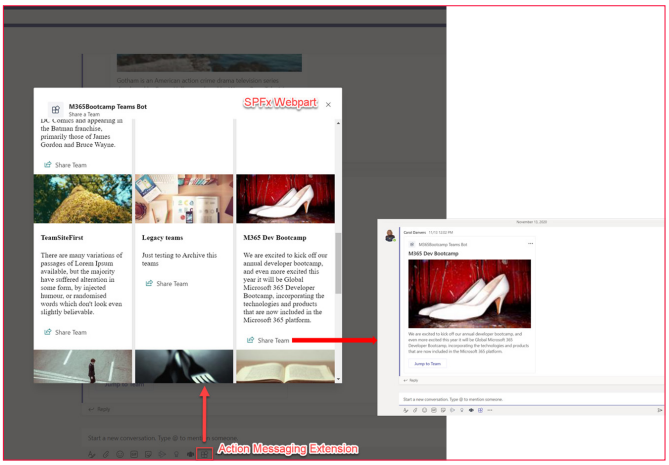


Imagen 2.- Message Extensions a desarrollar.

En nuestra solución, tendremos 2 piezas principales:

- SPFx WebPart: Será un WebPart de SPFx que consultará la MS Graph API para obtener el listado de los Teams en la Tenant.
- Bot: Será el encargado de recoger la información en-

viada por el WebPart SPFx (básicamente el Teams seleccionado de la lista), y componer una Adaptive Card con dicha información, para poder ser enviada por el chat. A día de hoy, desde el WebPart SPFx no podemos enviar nada por el chat, así que todavía necesitamos esta parte, pero como veremos, el código aquí será mínimo.

Desarrollando nuestro WebPart SPFx para Teams extensions

No vamos a entrar en mucho detalle en el código del WebPart. Vamos a usar la grandísima librería del PnP JS para acceder a MS Graph y sacar los Teams de la tenant. El snippet principal se muestra en la siguiente imagen:

```
graph.groups
  .setEndpoint("beta")
  .top(20)
  .select("id, displayName, description")
  .filter("resourceProvisioningOptions/Any(x:x eq 'Team')") // only Teams
  .get()
  .then((teams) => {
    this.setState({
      teams: teams.map((team: IGraphTeam, index: number) => {
        index += 10;
        return {
          displayName: team.displayName,
          id: team.id,
          description: team.description,
          thumbnailUrl: `https://picsum.photos/id/${index}/200/100`,
        };
      }),
    });
  });
```

Imagen 3.- Snippet principal del WebPart.

Lo importante viene ahora, donde necesitamos especificar que nuestro WebPart debe funcionar como extensión de Teams, para ello, necesitamos crear un archivo "manifest.json" en la carpeta "teams" de nuestro proyecto SPFx.

Tenéis la base de ese fichero manifest en la siguiente URL: https://docs.microsoft.com/en-us/sharepoint/dev/spfx/web-parts/guidance/creating-team-manifest-manually-for-WebPart

La parte principal de ese manifest, es el nodo "composeExtensions":

```
"composeExtensions": [
  {
    "botId": "** ESTE ES EL ID DE LA APP CREADA CUANDO REGISTRATE EL BOT CHANNEL **",
    "canUpdateConfiguration": true,
    "commands": [
      {
        "id": "shareTeam",
        "type": "action",
        "title": "Share Team info",
        "description": "Find and share a Team",
        "initialRun": false,
        "fetchTask": false,
        "context": [
          "commandBox",
          "compose"
        ],
        "taskInfo": {
          "title": "Share a Team",
          "width": "700",
          "height": "600",
          "url": "https://{teamSiteDomain}/_layouts/15/TeamsLogon.aspx?SPFX=true&dest=/_layouts/15/teamstaskhostedapp.aspx%3Fteams%26personal%26componentId=WEB-PART_ID%26forceLocale={locale}"
        }
      }
    ]
  },
```

Dicho nodo define las diferentes extensiones que queremos crear en Teams, así como el Identificador del WebPart que queremos que se abra cuando se pulse esa acción desde Teams. Además, debemos establecer el Identificador del Bot que manejará el evento disparado desde el WebPart de SPFx cuando se selecciona un Team de la lista. Posteriormente veremos cómo registrar esa App y obtener el ID.

Nuestro WebPart, para disparar el evento que recogerá el Bot, hará uso del mismo Framework de SPFx, que nos proporciona un objeto para ello. Lo vemos en el siguiente snippet:

```
private groupClicked = (group: IGraphTeam): void => {
  if (this.props.isTeamsMessagingExtension) {
    this.props.teamsContext.teamsJs.tasks.submitTask(group);
  }
}
```

Imagen 4.- Disparar el evento que recogerá el Bot.

Registro del Bot Channel en Azure

Como decíamos, vamos a necesitar un Bot para recoger la información y evento lanzado desde el WebPart, y mostrar esa información en el chat. Para ello registraremos el Bot Channel en Azure. Tenéis detalle de cómo hacer esto en el siguiente link: https://docs.microsoft.com/en-us/azure/bot-service/bot-service-quickstart-registration?view=azure-bot-service-4.0

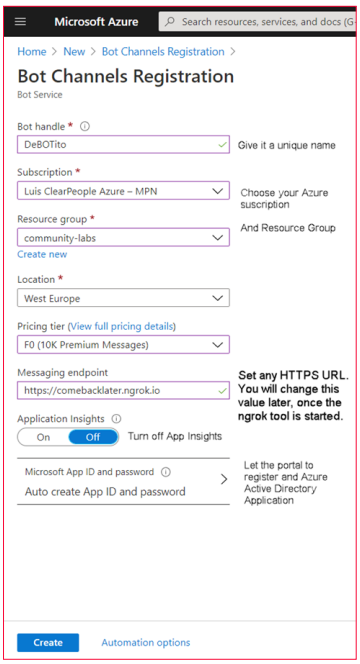


Imagen 5.- Registro del Bot.

Una vez registrado el Bot, nos creará una aplicación en Azure Active Directory, y es el ID de esa aplicación, el que necesitamos configurar en el "manifest" del WebPart SPFx.

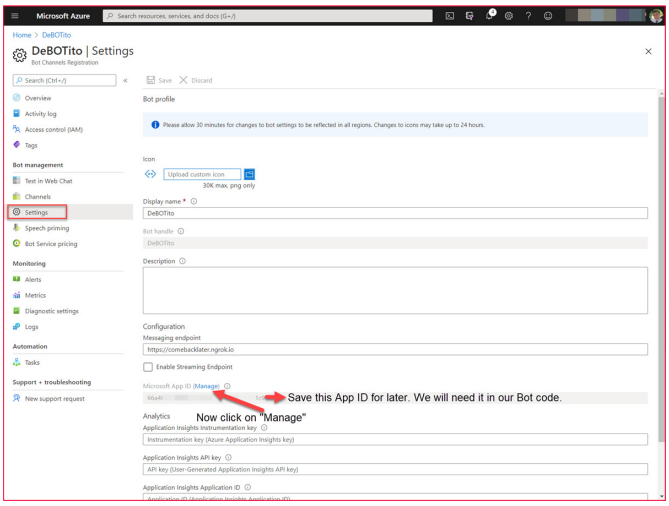


Imagen 6.- Aplicación creada en Azure AD.

Desarrollando nuestro Bot

```
npm install yo gulp-cli --global
npm install generator-teams --global
```

El Bot necesita una URL pública, así que para desarrollo, haremos uso de ngrok, que además, el propio "Yo Teams" tiene un comando para arrancar ngrok ya configurado para nuestro bot. Ten en cuenta que ngork, en su versión gratuita, asigna una URL nueva cada vez que se arranca, por lo que hay que cambiarla URL en el registro del Bot Channel. A nivel de código, una vez el scaffolding hecho por la herramienta "Yo Teams", necesitamos poco código para incrustar la información enviada por el WebPart, a la caja del chat.

```
protected async handleTeamsMessagingExtensionSubmitAction(
    context: TurnContext,
    action: MessagingExtensionAction): Promise<MessagingExtensionActionResponse> {

    const group: any = action.data;

    const groupCard = CardFactory.heroCard(group.displayName, group.description, [group.thumbnailUrl], undefined, undefined);

    const response: MessagingExtensionActionResponse = {
        composeExtension: {
            type: 'result',
            attachmentLayout: 'list',
            attachments: [groupCard]
        }
    }

    return Promise.resolve(response);
}
```

Imagen 7.- Scaffolding necesario.

Como vemos, sobrescribimos uno de los métodos del SDK de Teams, donde en su parámetro "action", recibimos la información enviada por el WebPart de SPFx (el Teams seleccionado del listado). Luego hacemos uso de otro método proporcionado por el framework, y construimos la Adaptive Card con la información del Teams. Finalmente, devolvemos esa Adaptive Card, que se incrusta en el chat de Teams.

Y hasta aquí el artículo. Tenéis todo el código completo disponible, además de los pasos detallados para el registro del Bot, en el siguiente repositorio de GitHub:

https://github.com/CompartiMOSS/Microsoft-365-Developer-Bootcamp-Virtual/tree/master/Track1/TeamsMessagingExtensionsWithSpfx

¡Hasta el próximo artículo!

LUIS MAÑEZ
Cloud Architect en ClearPeople LTD
@luismanez
https://github.com/luismanez

Alberto Diaz

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

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

Email: adiazcan@hotmail.com
Twitter: @adiazcan

Fabián Imaz

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

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

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

Gustavo Velez

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

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

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

Juan Carlos González Martín

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

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

Santiago Porras

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

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

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

¿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