Nº36 junio 2018

CompartiMOSS

REVISTA ESPECIALIZADA EN TECNOLOGÍAS MICROSOFT

Entrevista Robert Bermejo

SharePoint y Azure: El Microsoft Bing Search API

Contenedores en Azure

Microsoft Identity Manager 2016

Contenido

Comparti MOSS

Staff

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

DIRECCIÓN GENERAL

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

DISEÑO Y DIAGRAMACIÓN

- Santiago Porras Rodríguez

Contacte con nosotros

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

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

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

03

Editorial

Después de nuestro décimo aniversario, les presentamos un nuevo número de la revista en el que de nuevo nuestros autores comparten su experiencia y conocimiento en las tecnologías y plataformas punteros de Microsoft en la que son referentes. En este nuevo número, el 36 desde el nacimiento de la revista, hablaremos, por mencionar algunos de los hot topics, sobre desarrollo para SharePoint y Azure, como la combinación entre ambas plataformas es poderosa, novedades sobre Teams o lo que podemos llegar a hacer con las extensiones de Microsoft Graph.

Desde el equipo editorial de CompartiMOSS estamos encantados con el esfuerzo que número a número realizan nuestros autores y que hacen posible que la revista sea referente, nos atrevemos a decir que, a nivel mundial, en tecnologías y productos de Microsoft. Somos conscientes de que ser una revista de referencia supone una gran responsabilidad y también mucho trabajo de todos los que la conformamos (Equipo de CompartiMOSS, autores y lectores) tratando de seguir el rápido ritmo con el que Microsoft sigue innovando y del que hemos sido una vez más testigos de excepción en las pasadas conferencias de Microsoft Build, SharePoint Virtual Summit y SharePoint Conference North America.

Pero al mismo tiempo, esa gran responsabilidad es todo un reto que nos animará a tod@s a seguir, experimentar, utilizar y compartir todas esas innovaciones que Microsoft traerá en campos como la Inteligencia Artificial o la Mix-Reality, y a plataformas y tecnologías que desde CompartiMOSS siempre estamos siguiendo (SharePoint, Office 365, Azure por mencionar algunas).

Esperamos que disfruten de los artículos de este número como hemos disfrutado el equipo de CompartiMOSS durante el proceso de edición de la revista.

El Equipo Editorial de CompartiMOSS

04

Microsoft Graph API Extensions

Este artículo lo vamos a dividir en dos partes. En la primera, aprovechando que recientemente hemos tenido el Build 2018, vamos a ver algunas de las novedades presentadas alrededor de Graph. En la segunda parte, hablaremos de cómo podemos extender Graph almacenando información que puede venir de cualquier otro origen.

Novedades en Graph presentadas en el Build 2018

Como os decía, el pasado 7-9 de mayo se celebró el Build, donde Microsoft ha presentado todas sus novedades para desarrolladores. MS Graph fue de los que más atención ha recibido por parte de MS, así que tenemos bastantes novedades que os resumo en las 2 siguientes imágenes:

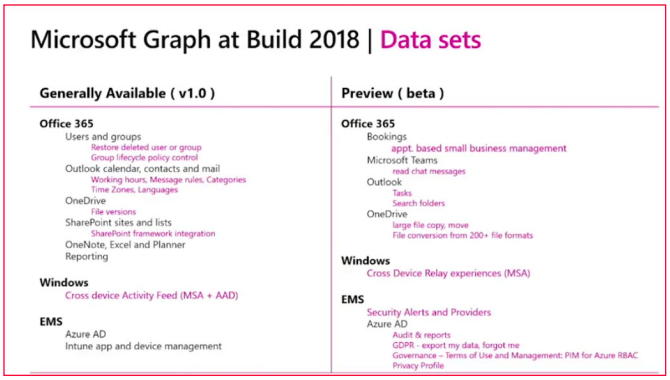


Imagen 1.- Data Sets de Microsoft Graph.

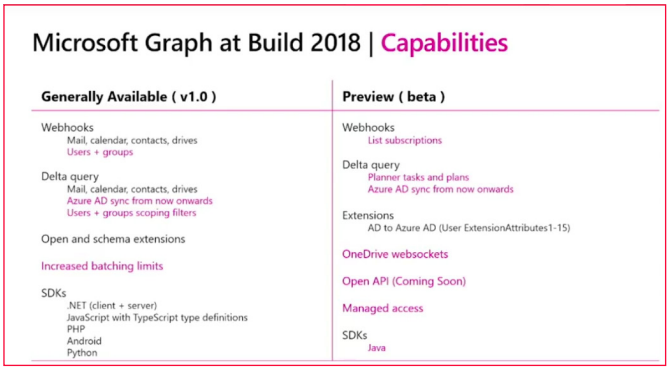


Imagen 2.- Microsoft Graph Capabilities.

Todas las sesiones del Build están disponibles en la web de Chnnel 9. En el siguiente link podéis ver todas las sesiones relacionadas con Graph:

https://channel9.msdn.com/Events/Build/2018?sort=status&direction=desc&tag%5B0%5D=microsoft%2Bgraph

Microsoft Graph Extensions

Graph Extensions en el mecanismo que nos proporciona Graph para extender la información almacenada en Graph, con datos provenientes de otros sistemas o escenarios. Por ejemplo, es bastante común tener un ID de empleado, que suele venir de algún sistema de RRHH, y queremos añadirlo a la información de perfil de Graph… o bien digamos que tenemos Grupos de Office 365, y queremos guardar Tags asociados a los grupos… para todo este tipo de escenarios, podemos usar Graph Extensions.

Existen dos tipos distintos de Extensions: Open extensions / Schema extensions

> ***"Graph Extensions en el mecanismo que nos proporciona Graph para extender la información almacenada en Graph"***

Antes de entrar en detalle, conviene saber que no todos los recursos de Graph soportan extensions. Por el momento, la siguiente tabla contiene los recursos que podemos extender:

Resource	Version
Administrative unit	Preview only
Calendar event	GA
Group calendar event	GA
Group conversation thread post	GA
device	GA
group	GA
message	GA
organization	GA
Personal contact	GA
user	GA

Imagen 3.- Recursos que se pueden extender.

MS Graph Open Extensions

Las extensiones de tipo Open, nos van a permitir añadir información a un recurso utilizando simplemente un listado de Key-Values. Por ejemplo, digamos que queremos añadir a mi perfil, el código de empleado, así como la fecha de la última revisión médica que hice a través de la empresa.

Para ello, hacemos un "POST" al endpoint:

https://graph.microsoft.com/v1.0/me/extensions

En el cuerpo del mensaje, compondremos un JSON con la siguiente información

NAME	VALUE
@odata.type	Microsoft.Graph.OpenTypeExtension
extensionName	Cualquier cadena única. Como convenio se suele invertir el dominio de la tenant: com.contoso.rrhhExtraData
Key – Value list	Una lista con clave-valor (nombre de la propiedad: valor asignado)

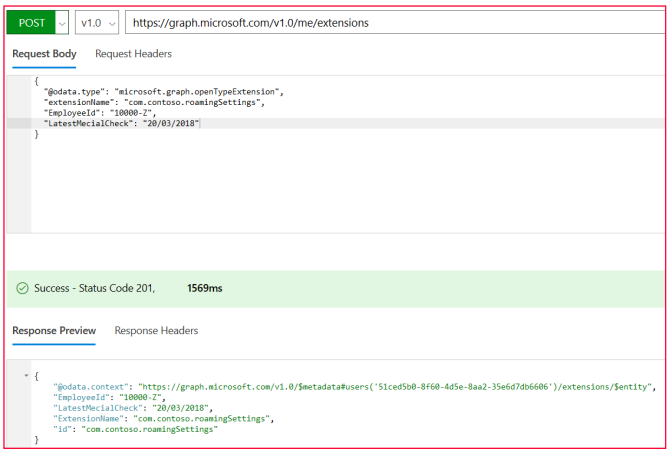


Imagen 4.- JSON con una Graph extension.

Para recuperar esa información, podemos hacer un GET al endpoint de Extensions, o expandir a Extensions:

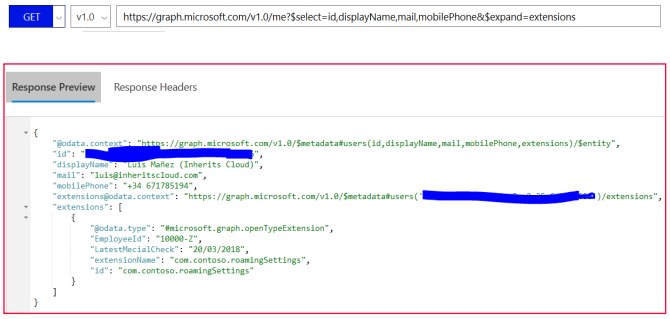


Imagen 5.- Como recuperar la información de una Graph Extension.

Ten en cuenta que al ser Open extensions, solo tendrán un valor en el caso que hayan sido asignadas, es decir, que, si hacemos esa misma query a un usuario que previamente no hemos asignado la extensión, ésta simplemente devolverá vacía, no esperes encontrar un EmployeId y LatestMedicalCheck con valor null, si no vacío a nivel de Extension.

MS Graph Schema extensions

Schema extensions nos va a permitir ampliar la información asociada a un recurso, pero de una manera más "tipada". Es decir, vamos a poder definir un tipo para nuestra Extensión. Por ejemplo, siguiendo con el ejemplo anterior, vamos a crear un tipo para extender nuestro perfil con algunos datos más que podemos tener en otros sistemas. Para ello necesitamos hacer un POST como el siguiente:


```
{
    "id":"inheritscloud_RRHHSchema",
    "description": "Data from RRHH system integration",
    "owner": "1d659d2d-9f28-4ffc-bb54-00f65b9cf0ec",
    "targetTypes": [
        "User"
    ],
    "properties": [
        {
            "name": "employeeId",
            "type": "Integer"
        },
        {
            "name": "payeCode",
            "type": "String"
        },
        {
            "name": "latestMedicalReview",
            "type": "DateTime"
        },
        {
            "name": "onLeave",
            "type": "Boolean"
        }
    ]
}
```

En esta URL puedes encontrar el detalle de las propiedades a establecer en el JSON

https://developer.microsoft.com/en-us/graph/docs/api-reference/beta/api/schemaextension_post_schemaextensions

Nota: Fíjate bien en como componer el valor para la propiedad "id", así como asegurarte de que proporcionas un valor válido para la propiedad "owner" (que debe ser el ID de una Azure AD App registrada en esa tenant). En el siguiente link tienes los diferentes tipos soportados en las properties (básicamente: Binary, Boolean, DateTime, Integer, String):

https://developer.microsoft.com/en-us/graph/docs/api-reference/beta/resources/extensionschemaproperty

Una vez definido el Schema, podemos actualizar nuestro usuario con dicha información. Para ello haremos un PATCH:

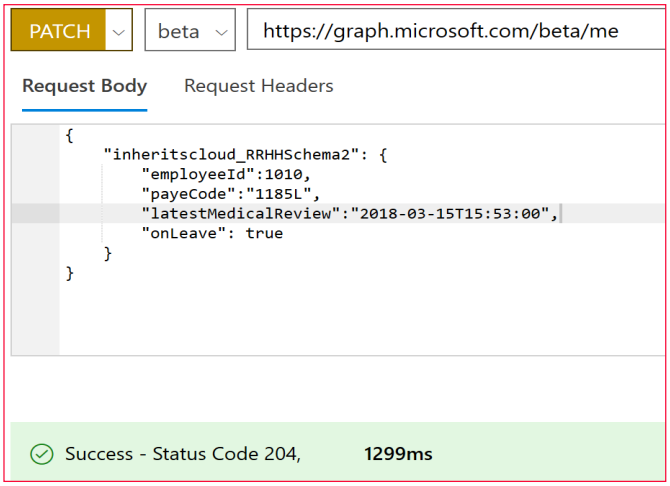


Imagen 6.- Actualización del usuario mediante PATCH

Llegado este punto, ya sólo tenemos que recuperar esa información. Para ello, y a diferencia de las Open extensions, en este caso podemos hacer un $select de la extensión, igual que si fuera un campo más como DisplayName, etc:

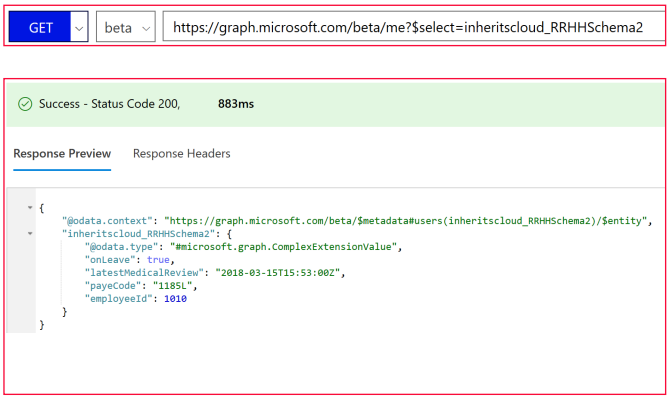


Imagen 7.- Haciendo un select de una Graph Extension

"Las extensiones de tipo Open, nos van a permitir añadir información a un recurso utilizando simplemente un listado de Key-Values"

Como hemos visto en el artículo, MS Graph nos ofrece un mecanismo muy potente para poder guardar información de otros sistemas en la propia Graph.

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

07

Eventos Real-Time con Azure SignalR en ASP.NET Core

Como cambian los tiempos, las personas y los pueblos, no podíamos esperar menos de nuestra amada tecnología, que si cabe evoluciona y se transforma más rápido que nadie. Si hablamos de arquitecturas software, podemos decir que es el momento de potenciar el uso de las librerías cliente (React, Angular,Vue…) y cómo no, el momento de pensar en arquitecturas serverless con objetivos muy ambiciosos de rendimiento y escalado.

El servicio del que os voy a hablar, creo que representa muy bien esta situación, mejora tanto el rendimiento en cliente como el rendimiento en backend, es llevar a un nivel más nuestro querido SignalR.

¿Qué es SignalR?

Para el que no conozca ASP.NET SignalR, es una librería que permite conectar en nuestras aplicaciones, la capa de backend con el lado cliente, en ambas direcciones, y sobre todo en tiempo real sin necesidad de recargar pantalla. Para conseguirlo, necesitábamos montar un SignalR Hub en el backend de nuestra web, y mediante una conexión Websocket, podíamos enviar mensajes a cada uno de nuestros clientes.

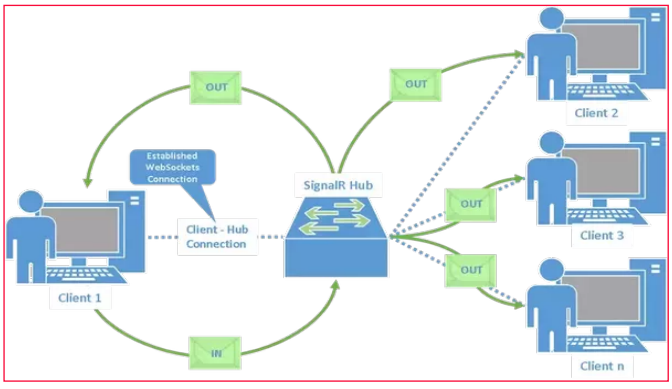


Imagen 1.- Arquitectura SignalR.

> ***"cambian los tiempos, las personas y los pueblos, no podíamos esperar menos de nuestra amada tecnología"***

Como vemos en la siguiente figura, cada mensaje que se envía al Hub de SignalR, este lo distribuye por el canal indicado en la petición, a cada uno de los clientes conectados (Hub Proxy).

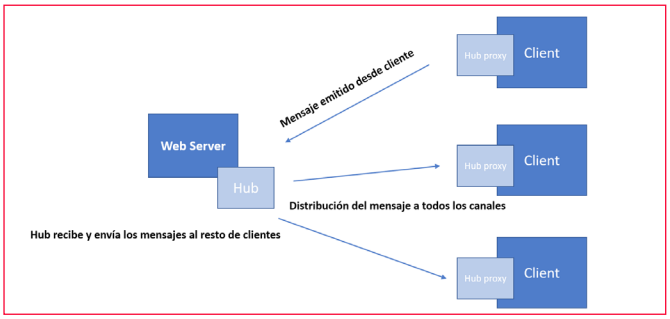


Imagen 2.- Mapa lógico de envíos.

Este sistema funcionaba bien, pero siguiendo un poco la tendencia actual, ¿Si empezamos a tener un tráfico excesivo de mensajes en nuestro Hub como dimensionamos el backend? ¿Qué coste tendría? ¿Por qué repetir este proceso de arquitectura para cada una de mis aplicaciones, no puedo tener un servicio único y escalable?

Para resolver todas estas preguntas / peticiones, entra en juego Azure SignalR.

¿Qué aporta Azure SignalR?

Este servicio es 100% compatible con la librería de SignalR, y lo que viene es a proporcionarnos un servicio 100% administrado, distribuido y multi-servidor, que nos va a permitir dimensionar y escalar tal y como queramos nuestra solución con SignalR. La principal diferencia con las soluciones que hemos visto anteriormente es que:

- Descargamos de trabajo al backend, el nuevo backend de SignalR estará alojado en un servicio en Azure, o lo que es lo mismo el Hub ya no se ejecutará desde la web. Ahora el mensaje se le enviará al servicio de Azure, en el que tendremos definido el Hub, el mensaje ahora puede venir de un Azure Function, de un App Service o de cualquier otro de origen de datos ya que es una simple llamada HttpPost.

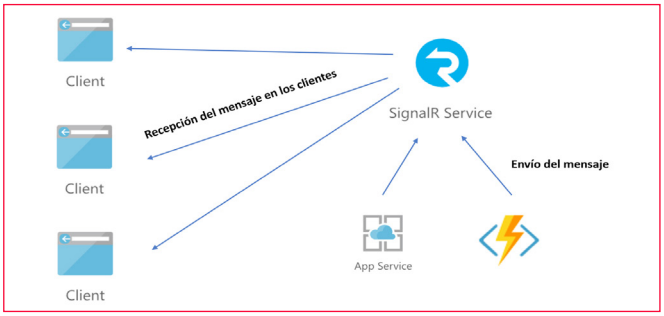


Imagen 3.- Nueva arquitectura de envíos, Azure como backend.

- Podremos conectar a un servicio tanto clientes como digamos, pudiendo escalar el servicio bajo demanda.
- Las conexiones al servicio se realizan con autenticación multifactor, gracias a la clave de acceso, que nos devuelve un token de sesión válido para que las aplicaciones clientes consuman el servicio.

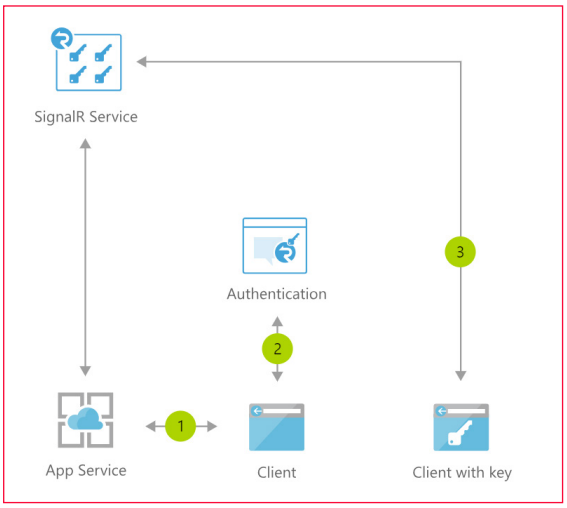


Imagen 4.- Autenticación en el servicio.

ASP.NET Core y SignalR, tu primer ejemplo

Una vez hemos visto las características del servicio, vamos a realizar un pequeño ejemplo en ASP.NET Core, creando un servicio de SignalR en Azure, y una aplicación de consola que envíe datos al servicio, para que la aplicación cliente que desplegaremos en Azure reciba los mensajes. Tanto los clientes, como la aplicación estarán autenticados contra el servicio de Azure SignalR mediante la cadena de conexión y la clave de acceso que daremos de alta en el portal de Azure.

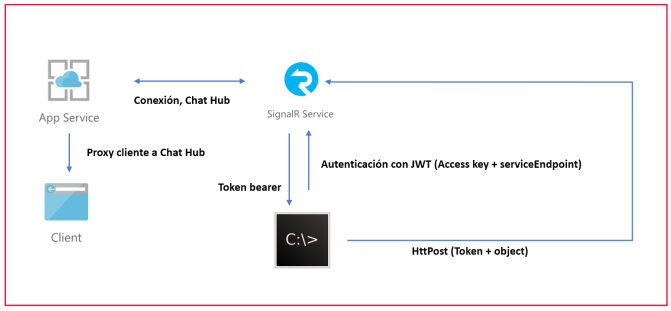


Imagen 5.- Ejemplo completo de envío.

El ejemplo completo nos lo podemos descargar de GitHub ya terminado, aunque con este artículo podremos crearlo desde cero.

Configurando el servicio en una aplicación web

Antes de empezar recordar que el ejemplo está desarrollado con Visual Studio 2017 y .Net Core 2.0, que nos podremos descargar de https://www.microsoft.com/net/download/windows. Además necesitaremos node.js instalado para poder descargar paquetes npm.

1.– El primer punto es crear una solución en Visual Studio, y en concreto en .NET Core 2.0 una Web MVC.

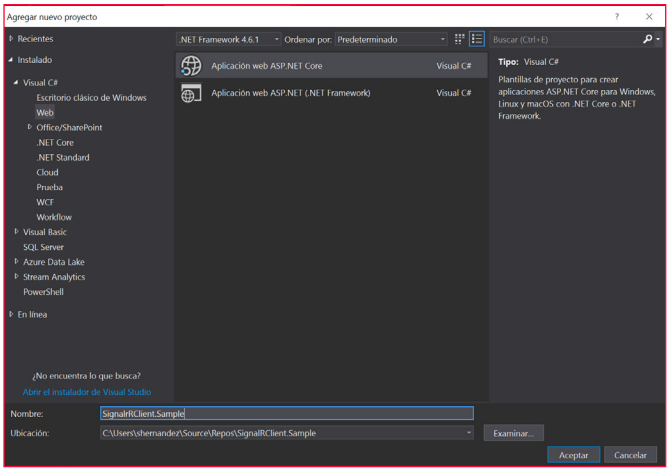


Imagen 6.- Creando la solución.

2.– Como el ejemplo está desarrollado en una web ASP.NET con MVC, seleccionamos la opción "Aplicación Web (controlador vista y modelos)", aunque podríamos optar por hacer una solución Angular o React, eso sí con JavaScript, porque al menos por mi experiencia actual, al ser un servicio en preview, la autenticación con TypeScript, al conectar nuestro proxy hub con el servicio de Azure da error.

3.– Una vez creada la solución, necesitamos descargarnos los siguientes paquetes:

- Paquete Nuget - Microsoft.Azure.SignalR -v 1.0.0-preview1-10009

  Para instalarlo deberemos desde una consola, situándonos en el path del proyecto MVC que acabamos de crear,ejecutar el comando "dotnet add package Microsoft.Azure.SignalR -v 1.0.0-preview1-10009"

- Paquete npm aspnet/signalr
  Para instalarlo deberemos ejecutar desde una consola la instrucción "npm install @aspnet/signalr"

4.– Una vez instalado los paquetes, deberemos copiar en la carpeta /webroot/js la librería signalR.js que nos acabamos de descargar, además de añadir al layout de la web el siguiente código, para poder instanciar el fichero:

```
<environment include="Development">
    <script src="~/lib/jquery/dist/jquery.js"></script>
    <script src="~/lib/bootstrap/dist/js/bootstrap.js"></script>
    <script src="~/js/signalr.js"></script>
    <script src="~/js/site.js" asp-append-version="true"></script>
</environment>

<environment exclude="Development">
    <script src="~/lib/jquery/dist/jquery.js"></script>
    <script src="~/lib/bootstrap/dist/js/bootstrap.js"></script>
    <script src="~/js/signalr.js"></script>
    <script src="~/js/site.js" asp-append-version="true"></script>
</environment>
```

Imagen 7.- Registrando signalR.js.

5.– En el fichero startup.cs deberemos modificar los métodos:

- ConfigureServices:

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddMvc();
    var conn = Configuration.GetConnectionString("SignalR").ToString();
    services.AddSignalR().AddAzureSignalR(conn);
}
```

Imagen 8.- Configurando la app y los servicios a inyectar.

Con este código registramos el servicio de SignalR en nuestra aplicación, y ya podremos hacer uso de este, además .NET Core, se encargará de inyectarlo en los controllers, para que podamos instanciar entre otras cosas un HubContext de signalR como veremos luego. Además, el servicio instanciado es nuestro servicio de Azure mediante la cadena de conexión que deberemos obtener desde el portal de Azure, una vez creemos el servicio.

- Configure:

```
    app.UseFileServer();
    app.UseAzureSignalR(routes =>
    {
        routes.MapHub<Chat>("/chat");
    });
```

Imagen 9.- Definiendo el hub.

Este código es muy importante, ya que nos permitirá instanciar el Hub en el contexto, y definir el proxy que vamos a utilizar, en este caso "/chat". Como vemos en el código, debemos mapear una clase Chat (aun por crear) a nuestro servicio de SignalR.

6.– Deberemos crear una clase Chat.cs, que va a actuar de Hub en nuestra web y que permitirá definir los ProxyHub en el lado cliente, eso si consumiendo el servicio de Azure tal cual hemos especificado en la configuración de la app.

```
using Microsoft.AspNetCore.SignalR;

namespace chattest
{

    3 referencias | shernandez, Hace 15 horas | 1 autor, 1 cambio
    public class Chat : Hub
    {
        0 referencias | shernandez, Hace 15 horas | 1 autor, 1 cambio | 0 excepciones
        public void BroadcastMessage(string name, string message)
        {
            Clients.All.SendAsync("new_msg", name, message);
        }

    }
}
```

Imagen 10.- Clase hub de SignalR.

7.– Ya solo nos falta crear el servicio cliente que instancie el hub "/Chat" y que utilice el canal "new_msg" que hemos definido en nuestro hub. Para ello añadimos a la vista Index de nuestra web un código muy sencillo que va a mostrar un alert con el dato recibido en el evento.

```
@section Scripts{

    <script type="text/javascript">
        var connection = new signalR.HubConnectionBuilder()
            .withUrl('/chat')
            .build();
        connection.start().catch(function (error) {
            console.error(error.message);
        });
        connection.on("new_msg", function (msg) {
            alert(msg);
        });
    </script>
}
```

Imagen 11.- Script cliente, proxyHub.

Si analizamos un poco el código, tenemos que creamos una conexión con HubConnectionBuilder, y lo apuntamos nuestro Hub "/chat". Además, necesitamos configurar con "connection.on" las entradas por el canal "new msg". Con este código, ya tenemos nuestro "proxyHub", y la conexión por websocket abierta, para que podamos recibir los mensajes por el canal. En el momento de carga de la vista si hacemos uso de la developer tools del navegador deberíamos obtener algo parecido a la siguiente imagen, si no es así deberemos revisar nuestro código.

Imagen 12.- traza en el navegador, websokect.

"Para resolver todas estas preguntas / peticiones, entra en juego Azure SignalR."

Creando el servicio en Azure

La aplicación ya la tenemos preparada, a falta de indicarle la cadena de conexión al nuevo servicio de Azure de SignalR. Para ello accedemos al portal de Azure y creamos un nuevo servicio del tipo SignalR Service, aclarar que es un servicio aun en "Preview" por lo que debemos estar predispuestos a que esto puede cambiar de un día para otro.

El servicio es muy sencillo, simplemente deberemos rellenar los datos habituales:

- Nombre del servicio: MySignalRSampleService.
- Elegir suscripción de Azure.
- Elegir o crear un grupo de recursos – Compartimoss-Sample.
- Location o región – West Europe.
- Price – Tier : Aquí podemos escoger entre la versión gratuita sin escalado (suficiente para el ejemplo), o escoger la básica que ya podemos definir unidades de escalado, y que tiene un coste de 20,68 € mes (precio estimado).

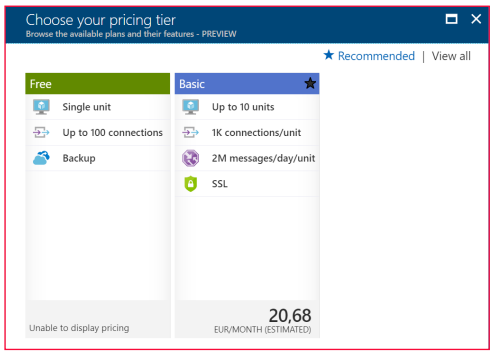


Imagen 13.- Servicio en azure, plan de precios.

Si todo va como debe, y tenemos creado el servicio, deberemos acceder al mismo, y en el apartado key podremos ver las cadenas de conexión, y copiar la cadena primaria, para poder incluirla en el proyecto.

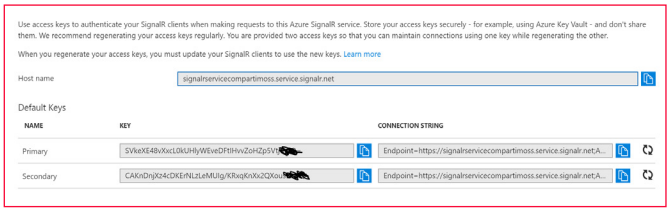


Imagen 14.- Cadenas de conexión.

> ***"falta de indicarle la cadena de conexión al nuevo servicio de Azure de SignalR"***

Por último, solo nos quedaría volver al proyecto, y añadir al fichero appsetting.json (para cada uno de los entornos) el siguiente JSON de conexión.

```
{
  "ConnectionStrings": {
    "SignalR": "Endpoint=https://signalrservicecompartimoss.service.signalr.net;AccessKey=SVkeXE48v
  },
  "Logging": {
    "IncludeScopes": false,
    "LogLevel": {
      "Default": "Warning"
    }
  }
}
```

Imagen 15.- Fichero de configuración en .NET Core.

Comprobando la conexión, enviando datos desde la web

En este punto tenemos todo configurado, pero es difícil saber si ha ido todo correcto, para ello vamos a hacer una pequeña prueba añadiendo a un controller la inserción de un mensaje en el canal de SignalR, para comprobar que el cliente está configurado correctamente y obtenemos conexión y contexto de autenticación con el servicio de Azure.

```
0 referencias | shernandez, Hace 16 horas | 1 autor, 1 cambio | 0 solicitudes | 0 excepciones
public void AddMsg(string msg)
{
    context.Clients.All.SendAsync("new_msg", msg);
}
```

Imagen 16.- Prueba desde MVC Controller.

Pero para poder instanciar el contexto de SignalR, vamos a hacer uso de la inyección de dependencias de .NET Core, y a modificar el constructor de nuestro HomeController.

```
IHubContext<Chat> context;
0 referencias | shernandez, Hace 16 horas | 1 autor, 1 cambio | 0 excepciones
public HomeController(IHubContext<chattest.Chat> hub)
{
    context = hub;
}
```

Imagen 17.- Inyectando el HubContext.

Si con la aplicación compilada y lanzada desde nuestro Visual Studio, invocamos la petición GET http:localhost:portApp/Home/AddMsg?msg=mensajeatrasmitir, deberíamos ver en la Home/Index un alert con el mensaje pasado por querystring. Hasta aquí hemos dejado la aplicación funcional, pero sigue siendo el backend el que desde el controller envía el mensaje al lado cliente, aún nos queda un poco de trabajo.

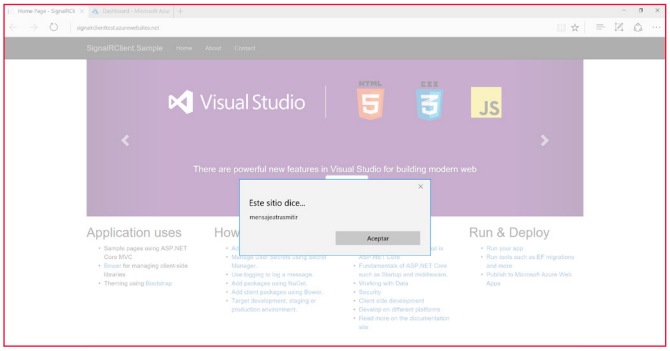


Imagen 18.- Resultado de la demo, ok.

Enviando datos directamente desde el servicio en Azure

Vamos a dar por supuesto, que ya tenemos la APP 100% funcional y desplegada en un Azure App Service. Para que no tengamos problemas con SignalR en Azure, deberemos dejar en las settings del app service activados los websocket, o no funcionará correctamente.

Imagen 19.- Habilitando websocket.

Ahora vamos a crear una aplicación de consola, y la añadiremos a nuestra solución con el siguiente código:

```
namespace TestSignalR
{
    0 referencias | shernandez, Hace 16 horas | 1 autor, 1 cambio
    class Program
    {
        0 referencias | shernandez, Hace 16 horas | 1 autor, 1 cambio | 0 excepciones
        static void Main(string[] args)
        {

            var signalR = new AzureSignalR("Endpoint=https://signalrservicecompartimoss.service.signalr
            var msg = Console.ReadLine();
            var result = signalR.SendAsync("chat", "new_msg",msg).ConfigureAwait(false);
        }
    }
}
```

Imagen 20.- Program.cs.

Si revisamos el código, nos falta por crear una clase "AzureSignalR" que ahora veremos, y que recibe como parámetro de entrada el mismo endpoint que hemos venido utilizando en el ejemplo anterior (Azure SignalR service).

> *“queda trabajo sobre todo con la integración con TypeScript, documentación y ejemplos”*

La clase AzureSignalR que deberemos crear dentro del proyecto de consola, se va a encargar de generar el JWT para obtener el token de seguridad, utilizando la cadena de conexión.

```
2 referencias | shernandez, Hace 16 horas | 1 autor, 1 cambio | 0 excepciones
private string GenerateJwtBearer(string issuer, string audience, ClaimsIdentity subject, DateTime? expires, string signingKey)
{
    SigningCredentials credentials = null;
    if (!string.IsNullOrEmpty(signingKey))
    {
        var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(signingKey));
        credentials = new SigningCredentials(securityKey, SecurityAlgorithms.HmacSha256);
    }

    var token = jwtTokenHandler.CreateJwtSecurityToken(
        issuer: issuer,
        audience: audience,
        subject: subject,
        expires: expires,
        signingCredentials: credentials);
    return jwtTokenHandler.WriteToken(token);
}
```

Imagen 21.- JWT para obtener bearer.

Además, va a realizar un POST con el mensaje a trasmitir, que en este caso recoge por consola.

```
1 referencia | shernandez, Hace 16 horas | 1 autor, 1 cambio | 0 excepciones
private Task<HttpResponseMessage> PostJsonAsync(string url, object body, string bearer)
{
    var request = new HttpRequestMessage
    {
        Method = HttpMethod.Post,
        RequestUri = new Uri(url)
    };

    request.Headers.Authorization = new AuthenticationHeaderValue("Bearer", bearer);
    request.Headers.Accept.Clear();
    request.Headers.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));
    request.Headers.AcceptCharset.Clear();
    request.Headers.AcceptCharset.Add(new StringWithQualityHeaderValue("UTF-8"));

    var content = JsonConvert.SerializeObject(body);
    request.Content = new StringContent(content, Encoding.UTF8, "application/json");
    return httpClient.SendAsync(request);
}
```

Imagen 22.- Post del mensaje al servicio.

La diferencia con el ejemplo anterior es que el dato que recibe ahora la web si es enviado desde el hub de Azure SignalR, que ahora si actúa como backend.

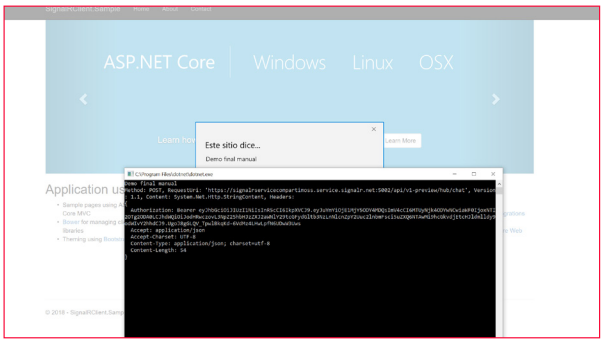


Imagen 23.- Demo final desde console.

Conclusiones

Aun siendo un servicio en preview, y que le queda trabajo sobre todo con la integración con TypeScript, documentación y ejemplos; animo a todo el mundo a empezar a utilizarlo en entornos de desarrollo, y a probar lo sencillo que es integrarlo con las apps que necesiten de eventos en tiempo real como son estos ejemplos de SignalR.

Cada vez más va a ser necesario aligerar los backend de las webs, para incluir nuevas piezas serverless, y eso equivale a mucho más trabajo asíncrono, en background y por lo tanto intentar que la parte cliente esté 100% conectada y no veo una forma mejor que empezar con este tipo de servicios.

Como pista, si queremos hacer una verdadera solución “real time” os animo a que combinéis Azure SignalR y Azure Event Grid, ya que están condenados a entenderse y podremos monitorizar en tiempo real todos los eventos de nuestros backends en una solución cliente.

SERGIO HERNANDEZ MANCEBO
Principal Team Leader
@shmancebo

12 Entrevista Robert Bermejo

Me llamo Robert Bermejo, Barcelonés hasta la médula, nací en esa maravillosa ciudad en 1977.

Soy Licenciado en Ingeniería informática por la Universidad Autónoma de Barcelona.

Actualmente trabajo en TOKIOTA como Cloud Solutions Consultant que me permite desarrollar una de mis pasiones que es el Cloud, y en concreto Azure.

Organizo la comunidad de Azure de Barcelona, CATzure, donde mensualmente organizamos meetups para hablar, compartir y aprender sobre Azure, además organizado junto a muchos amigos y compañeros de la comunidad la Global Azure Bootcamp en Barcelona.

A parte de mi trabajo y mi comunidad, también participo en todos aquellos eventos que me aceptan, escribo en mi blog (muy poquito) y en esta maravillosa revista llamada CompartiMOSS.

También soy MVP de Azure desde hace un año premio del que me siento muy orgulloso por lo que representa y por la responsabilidad que conlleva.

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

La “culpa” fue de mi padre, que cuando tenía 6-7 años trajo un Spectrum 48k a casa, recuerdo conectar el Spectrum a una TV y un radiocasete y rezar para que no se cortase la carga del juego después de 10 minutos cargando.

De allí pasé ha hacer un curso de Basic cuando tenía 14 años, y no entendí nada de programación veía los Goto y alucinaba, ya que yo era más “gamer” que developer.

Pero fue en ese momento que supe que me vida estaría ligada al mundo de la informática y fue mi única opción para entrar en la Universidad, si no hubiera entrado no sé que hubiera estudiado la verdad.

Lo más curioso es que durante la carrera no me gustaba programar, con el tiempo me he dado cuenta de que era porque no tenía ni idea, y mi objetivo era dedicarme al management más que a la parte técnica.

Y así que cuando empecé mi carrera laboral, enseguida además de programar intentaba coger experiencia y pedía oportunidades más de gestión que de desarrollo, seguía sin saber programar, hasta que un día se cruza en tu camino alguien con pasión.

Las crisis económicas son muy duras, pero a veces también dan oportunidades, y ese fue mi caso, ya que durante la crisis me vi obligado a buscar un cambio de trabajo y en mi camino se cruzó Roberto Grassi que me realizó una entrevista de trabajo.

Recuerdo salir de aquella entrevista confirmando aquello que ya sabía: no tengo ni idea y tengo que ponerme las pilas y una frase lapidaria: “Si entras no volverás a leer novelas como esa”, en aquella época me estaba leído una trilogía de romanos, dicho y hecho, no he vuelto a leer una novela.

Sorprendentemente me llamaron y entré a forma parte de su equipo y a partir de ese momento la pasión y el conocimiento que trasmitía me enganchó y empecé a estudiar e intentar mejorar mis habilidades como developer y a devorar libros y blogs de tecnología.

Y ahí realmente empezó todo, el principio de un camino que todavía no ha acabado.

¿Cuáles son tus principales actividades tecnológicas hoy en día?

Mi día a día es muy variado, principalmente paso la parte de mi tiempo diseñando soluciones en Azure para los clientes, que les ayuden a solucionar sus problemas e incrementar su productividad y liderando estos proyectos técnicamente También ayudo en las preventas, realizando las ofertas junto a los compañeros de comercial y operaciones.

Y durante todas esas tareas intento ayudar a crecer a los compañeros ayudándoles en su día a día e intentando poder transmitir algo de mis conocimientos y sobre todo intentando que ellos me transmitan los suyos, porque para mi actividad diaria prioritaria es aprender algo.

Y una vez acabo con la parte profesional empieza mi jornada de pasión dedicada a la comunidad, artículos, preparar sesiones, meetups, etc.

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

Fuera de la tecnología dedico mi tiempo a la familia, a mi mujer y mis hijos. Mis dos hijos me absorben el máximo del tiempo posible, y son lo mejor de mi vida: llegar a casa y que te vengan corriendo, te abracen y te den un beso es la mejor cura del estresante día a día.

¿Cuáles son tus hobbies?

La música, no sé vivir sin ella, siempre que puedo escucho música, y las series, es lo único que veo en la tela.

¿Alguna anécdota en el mundo de la informática, que recuerdes con buen sentido del humor?

Una que no hace mucho tiempo que me pasó.

Empecé un proyecto para cliente, un proyecto muy sencillito sin mucha lógica de negocio, simplemente unos maestros para realizar unas configuraciones de unas aplicaciones y poca cosa más. Lógicamente la arquitectura de esta aplicación era muy sencilla, con una API REST muy básica.

A las tres semanas de proyecto el cliente nos comenta que tenemos que tener una reunión con el responsable de arquitectura (que hasta la fecha no había aparecido) y sin saber de que iba empezó a explicar como quería que fuera la arquitectura: que si todo en microservicios, que porque no habíamos implementado un proxy para las APIS…y yo intentando explicarle que la bazuca no hacía falta para esa pequeña mosca.

Después de la conversación le pedí que entorno para Microservicios tenían para orquestar y administrar contenedores y la respuesta fue que tenía en su máquina un entorno de test con Linux y OpenShift (el desarrollo era sobre Windows y necesitaba Windows Authentication) y me adjuntó el libro de Microservicios de Microsoft escrito por Cesar de la Torre de como quería la solución.

Os podéis imaginar las risas que nos dimos a su costa, y no se entregaron ni contendores ni microservicios.

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

Difícil, además de que no se me da nada bien predecir el futuro, Aramis Fuster seguro que acertaría más que yo.

Pienso que el futuro está en ayudar y facilitar las decisiones y acciones de las personas en su día a día, por eso creo que el IoT que más que un futuro es una realizad y la IA son los dos puntales del presente y del futuro de la tecnología.

Pero a niel empresarial creo que el futuro de la tecnología pasa por el cloud. Pasa porque cada vez más empresas quieren reestructurar su forma de trabajar, integrar sus aplicaciones, desarrollarlas más rápido, evolucionarlas más rápido, gastar menos en infraestructura, y todo ello con el objetivo de que dar un mejor servicio a sus clientes y hacer más productivos a sus empleados. Para ello creo que el futuro pasa también por la mejora continua de los servicios cloud, de nuevas formas de desarrollar aplicaciones que evolucionar con los nuevos servicios que se ofrezcan y de nuevas formas de poder migrar las aplicaciones al cloud de una forma más sencilla y rápida.

ROBERT BERMEJO
Cloud Specialist Consultant en TOKIOTA
Microsoft Azure MVP
bermejoblasco@live.com
@robertbemejo
www.robertbermejo.com

14

SharePoint y Azure: El Microsoft Bing Search API

Que es el Microsoft Bing Search API

El servicio de búsqueda de Bing en Azure proporciona una experiencia similar a las búsquedas que se pueden realizar con Bing.com, devolviendo los resultados de una consulta que Bing determina son relevantes para el usuario. Los resultados incluyen páginas web, imágenes, vídeos y mucho más, como por ejemplo noticias, consultas relacionadas, corrección de errores de escritura, conversión de unidades, traducciones y cálculos matemáticos. El servicio forma parte del conjunto de Cognitive Services de Azure, que es una colección de algoritmos de Inteligencia Artificial que se pueden utilizar en la nube de Microsoft.

El Bing API es un servicio que consiste en un servicio general, el Bing Web Search, que devuelve resultados generales, de noticias, imágenes, videos, etc. relacionados con la consulta del cliente, y servicios especializados para cada una de esas partes que se pueden llamar por separado y retornan solamente los resultados relevantes (solo noticias, o solo videos, etc.). El servicio está basado en consultas REST, en donde la consulta se envía a un URL especializado (el Endpoint de Bing Search, https://api.cognitive.microsoft.com/bing/v7.0/search), con un parámetro "q" en el Query String con la cadena de consulta, y la llave del usuario de Azure en la cabecera del URL. La respuesta retorna en formato JSON.

"una experiencia similar a las búsquedas que se pueden realizar con Bing.com"

Adicionalmente es posible refinar los resultados especificando un país, lo que devuelve los resultados de la búsqueda basándose en los intereses de ese país (para la parte de noticias, por ejemplo). La API de Bing Web Search soporta más de tres docenas de países, muchos con más de un idioma. El país se especifica mediante el parámetro de consulta "cc". Si se especifica un país, también se debe especificar uno o varios códigos de idioma mediante el encabezado HTTP Accept-Language. Los lenguajes soportados varían según el país, y se indican para cada país según su "mercado", que a su vez se puede especificar utilizando el parámetro de consulta "mkt" y un código especifico. Utilizando un mercado, simultáneamente se especifica un país y un idioma preferido. El parámetro de consulta "setLang" puede establecerse en un código de idioma en este caso; usualmente este es el mismo idioma especificado por "mkt" a menos que el usuario prefiera ver Bing en otro idioma. En castellano se pueden utilizar, por ejemplo, los países Argentina (AR), Chile (CL), México (MX) y España (ES), y los mercados Argentina-Español (es-AR), Chile-Español (es-CL), México-Español (es-MX) y España-Español (es-ES).

El Azure Bing Search API se puede utilizar en SharePoint para enriquecer automáticamente la información en el sistema. Aunque SharePoint dispone de su propio motor de búsqueda, él se reduce a indexar información dentro de SharePoint mismo, dentro de SharePoint OnPrem y un tenant de SharePoint 365 (en un sistema hibrido), o indexar sistemas conectados por medio del Business Connectivity Service (y, en ciertos casos, información en file-shares), pero no es capaz de relacionar información interna con información disponible en internet. Por medio del Bing Search API es posible crear este tipo de relaciones y, si es necesario, agregarle también funcionalidad adicional, como por ejemplo hacer cálculos matemáticos.

En el ejemplo que se va a desarrollar enseguida se utiliza el Bing Search API para mostrar información adicional sobre un nombre de un personaje. Una Lista Personalizada de SharePoint dispone de tres campos: uno (el Titulo por defecto) para el nombre del personaje, otro para mostrar un resumen de información al respecto, y un tercero con el vinculo correspondiente a la Wikipedia.

Configuración del Azure Bing Search API

Para utilizar el Bing Search API es necesario crear primero el servicio en Azure, aunque también es posible utilizar una cuenta temporal de prueba desde la página de Microsoft https://azure.microsoft.com/en-us/try/cognitive-services/#-search. Para crear un servicio (de pago) en Azure:

1.– Entre al portal de manejo de Azure (https://portal.azure.com) utilizando sus credenciales.

2.– Vaya a la sección de "Resource Groups" y cree un nuevo Grupo de Recursos (también es posible reutilizar un grupo ya existente).

3.– Cree un servicio de “Bing Search API”:

- En el Resource Group, utilice el botón de “+Add” para crear un recurso, busque por “Bing Search” en la casilla de búsqueda y seleccione “Bing Search v7 APIs” en los resultados.
- Asígnele un nombre al servicio y utilice el Grupo de Recursos deseado. En la casilla de “Pricing tier” seleccione un servicio dependiendo de la cantidad de consultas a esperar por segundo, lo que determina el precio del servicio (por mil consultas). Acepte el anuncio de privacidad que aparece en la configuración (Microsoft utilizara los datos enviados para mejorar automáticamente los algoritmos de Bing).

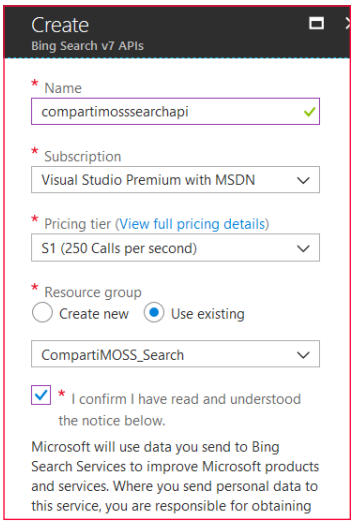


Imagen 1.- Creación del servicio de Bing Search API.

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

Utilizando el Azure Bing Search API con SharePoint

En el siguiente ejemplo, como se indicó anteriormente, se va a utilizar una Lista Personalizada de SharePoint en donde el campo de “Titulo” de un elemento nuevo creado en la Lista, inicia una consulta de Bing. Cuando se introduce un nuevo elemento con el nombre de una personalidad en el Titulo de un elemento de la Lista, un WebHook hace que una Función de Azure comience a funcionar, utilice el texto del “Titulo” como consulta para hacer una llamada al Azure Bing Search API y modifique el elemento en la Lista agregándole el resumen de la información y el URL del articulo correspondiente de Wikipedia.

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

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

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

7.– Agréguele a la solución el paquete NuGet “AppForSharePointOnlineWebToolkit”.

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

```
[FunctionName(“SearchBingWeb”)]
public static async Task<HttpResponseMessage> Run([HttpTrigger(AuthorizationLevel.Anonymous, “post”, Route = null)]
HttpRequestMessage req, TraceWriter log)
{
  log.Info(“*** SearchBingWeb function processed a request ***”);

  // Registration
  string validationToken = GetValidationToken(req);
  if (validationToken != null)
  {
    log.Info($”---- Processing Registration”);
     var myResponse = req.CreateResponse(HttpStatusCode.OK);
      myResponse.Content = new StringContent(validationToken);
    return myResponse;
  }

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

  if (allNotifications.Count > 0)
  {
    log.Info($”---- Processing Notifications”);
     string siteUrl = ConfigurationManager.AppSettings[“whSiteListUrl”];
    foreach (var oneNotification in allNotifications)
    {
      // Login in SharePoint
       ClientContext SPClientContext = HelpFunctions.LoginSharePoint(siteUrl);

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

  return new HttpResponseMessage(HttpStatusCode.OK);
}

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

Esta rutina primero se encarga de hacer el registro del WebHook (si la consulta contiene un parámetro “validationtoken” en el Query String) utilizando la rutina “GetValidationToken”:

```
public static string GetValidationToken(HttpRequestMessage
req)
{
  string strReturn = string.Empty;
  strReturn = req.GetQueryNameValuePairs()
       .FirstOrDefault(q => string.Compare(q.Key, "validation-
token", true) == 0)
    .Value;

  return strReturn;
}
```

En el código, después de registrado el WebHook, cada consulta es procesada para extraer las notificaciones que contiene. En cada notificación de la colección de notificaciones se hace un logeo en SharePoint para obtener los cambios detectados en la Lista (por medio de la rutina "GetChanges"). En la variable "whSiteListUrl" del App Settings de la función se encuentra el URL del sitio en donde se encuentra la Lista a examinar ("https://[Dominio].sharepoint.com/sites/[NombreSitio").

"Como Bing retorna un resultado con varios elementos, por medio de un loop se busca el resultado que se refiere a la Wikipedia"

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

```
static void GetChanges(ClientContext SPClientContext, string
ListId, TraceWriter log)
{
  // Get the List
  Web spWeb = SPClientContext.Web;
   List myList = spWeb.Lists.GetByTitle(ConfigurationManager.
AppSettings["whListName"]);
  SPClientContext.Load(myList);
  SPClientContext.ExecuteQuery();

  // Create the ChangeToken and Change Query
   ChangeQuery myChangeQuery = GetChangeQueryNew(List-
tId);

  // Get all the Changes
  var allChanges = myList.GetChanges(myChangeQuery);
  SPClientContext.Load(allChanges);
  SPClientContext.ExecuteQuery();

  foreach (Change oneChange in allChanges)
  {
    if (oneChange is ChangeItem)
    {
      int myItemId = (oneChange as ChangeItem).ItemId;

      // Get what is changed
      log.Info($"---- Changed ItemId : " + myItemId);
      ListItem myItem = myList.GetItemById(myItemId);
      SPClientContext.Load(myItem);
      SPClientContext.ExecuteQuery();

      // Search result in Bing Web
      SearchBingWebResult myResultBing = GetBingWebSear-
ch(myItem["Title"].ToString()).Result;

      // Insert the values back in the List
      int indexWeb = 0;
       for (int oneWeb = 0; oneWeb < myResultBing.webPages.
value.Count(); oneWeb++)
      {
        if (myResultBing.webPages.value[oneWeb].name.Con-
tains("Wikipedia") == true)
        {
          indexWeb = oneWeb;
          break;
        }
      }
        myItem["LinkWikipedia"] = myResultBing.webPages.
value[indexWeb].displayUrl;
        myItem["Description"] = myResultBing.webPages.va-
lue[indexWeb].snippet;
      myItem.Update();
      SPClientContext.ExecuteQuery();
        log.Info($"---- Search Bing Web added to SharePoint
Item");
    }
  }
}
```

Primero se crea un objeto que contienen la Lista a utilizar en SharePoint. Luego se crea una consulta de cambio (variable "myChangeQuery") que especifica que se requieren los cambios ocurridos en el ultimo minuto, que ocurren en elementos de la Lista y que sean del tipo "Add", es decir, elementos nuevos:

```
public static ChangeQuery GetChangeQueryNew(string ListId)
{
  ChangeToken lastChangeToken = new ChangeToken();
   lastChangeToken.StringValue = string.Format("1;3;{0};{1};-1",
ListId, DateTime.Now.AddMinutes(-1).ToUniversalTime().Ticks.
ToString());
  ChangeToken newChangeToken = new ChangeToken();
  newChangeToken.StringValue = string.Format("1;3;{0};{1};-1",
ListId, DateTime.Now.ToUniversalTime().Ticks.ToString());
    ChangeQuery myChangeQuery = new ChangeQuery(false,
false);
  myChangeQuery.Item = true;  // Get only Item changes
  myChangeQuery.Add = true;  // Get only the new Items
  myChangeQuery.ChangeTokenStart = lastChangeToken;
  myChangeQuery.ChangeTokenEnd = newChangeToken;

  return myChangeQuery;
}
```

Luego de ejecutar la consulta, se examina cada uno de los cambios y se obtiene un objeto con el Elemento agregado. En la misma rutina se llama a "GetBingWebSearch", la que se encarga de hacer la consulta en Azure Bing, utilizando como parámetro de entrada el nombre del personaje (el Titulo del Elemento). Esta rutina entrega de regreso un objeto del tipo "SearchBingWebResult" que contiene los resultados de la consulta y que tiene la forma:

```
public class SearchBingWebResult
{
  public Webpages webPages { get; set; }
}

public class Webpages
{
  public string webSearchUrl { get; set; }
  public int totalEstimatedMatches { get; set; }
  public Value[] value { get; set; }
}

public class Value
{
  public string id { get; set; }
  public string name { get; set; }
  public string url { get; set; }
  public About[] about { get; set; }
  public bool isFamilyFriendly { get; set; }
  public string displayUrl { get; set; }
  public string snippet { get; set; }
```

```
    public Deeplink[] deepLinks { get; set; }
    public DateTime dateLastCrawled { get; set; }
    public string language { get; set; }
    public string thumbnailUrl { get; set; }
}

public class About
{
    public string name { get; set; }
}

public class Deeplink
{
    public string name { get; set; }
    public string url { get; set; }
}
```

Como Bing retorna un resultado con varios elementos, por medio de un loop se busca el resultado que se refiere a la Wikipedia. Finalmente se utilizan los valores de "snipped" y "displayUrl" para insertarlos en los campos de "Description" y "LinkWikipedia" del Elemento.

"El servicio de Bing de Azure permite enriquecer la información que los usuarios guardan en SharePoint"

10.– La rutina "GetBingWebSearch " recibe como parámetros de entrada el texto del título con el nombre del personaje y retorna un objeto con los valores encontrados por Bing:

```
public static async Task<SearchBingWebResult> GetBingWebSearch(string mySearchQuery)
{
    SearchBingWebResult resultReturn = new SearchBingWebResult();

  // Construct the URI of the search request. Using the "en-US" market to force responce in english only
    var uriQuery = ConfigurationManager.AppSettings["azSearchBingWebEndpoint"] + "?mkt=en-US&q=" + Uri.EscapeDataString(mySearchQuery);
    string contentString = string.Empty;

    // Perform the Web request and get the response
    WebRequest request = HttpWebRequest.Create(uriQuery);
    request.Headers["Ocp-Apim-Subscription-Key"] = ConfigurationManager.AppSettings["azBingSearchApiServiceKey"];
    HttpWebResponse response = (HttpWebResponse)request.GetResponseAsync().Result;
    string json = new StreamReader(response.GetResponseStream()).ReadToEnd();

    // Create result object for return
    var searchResult = new SearchResult()
    {
      jsonResult = json,
      relevantHeaders = new Dictionary<String, String>()
    };

    // Extract Bing HTTP headers
    foreach (String header in response.Headers)
    {
        if (header.StartsWith("BingAPIs-") || header.StartsWith("X-MSEdge-"))
        searchResult.relevantHeaders[header] = response.Headers[header];
    }

    resultReturn = JsonConvert.DeserializeObject<SearchBingWebResult>(searchResult.jsonResult);
    return resultReturn;
}

// Used to return search results including relevant headers
struct SearchResult
{
    public String jsonResult;
    public Dictionary<String, String> relevantHeaders;
}
```

Cada consulta al Bing API se realiza por medio de una llamada REST a un URL pre-especificado del servicio de búsqueda (dado en el valor de la App Settings "azSearchBingWebEndpoint" y que es "https://api.cognitive.microsoft.com/bing/v7.0/search"), utilizando como parámetros en el QueryString el "Mercado" ("mkt", si es necesario forzar resultados en un idioma determinado) y la cadena de la consulta ("q"). En la App Settings "azBingSearchApiServiceKey" se mantiene el valor de la llave mencionada en el punto 4.

El Bing Web API es mucho más completo de lo que se muestra en el ejemplo. Por medio de los parámetros de la llamada REST se pueden especificar mucho más finamente los resultados, incluyendo determinar qué tan "fresca" es la información, filtrar el tipo de resultados, etc. Igualmente, el resultado que se usa en el ejemplo es solamente el objeto "Webpage", pero la respuesta puede contener muchos otros tipos incluyendo imágenes, videos, noticias, etc. Toda la definición de los parámetros de entrada y los objetos de salida se puede encontrar en la información de referencia de Microsoft en https://docs.microsoft.com/en-us/rest/api/cognitiveservices/bing-web-api-v7-reference. Para utilizar otros parámetros de entrada, modifique la consulta en la variable "uriQuery" de la rutina "GetBingWebSearch". Para entregar otros tipos de objetos de respuesta, modifique la clase "SearchBingWebResult" para que el objeto pueda capturar los valores deseados; la rutina "GetBingWebSearch" se encarga de deserializarlos automáticamente.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

12.– Registre el WebHook en la Lista de SharePoint y cree un Elemento indicando en el titulo el nombre de un personaje. El WebHook hará que la Función realice su trabajo, entregue los resultados de Bing y muestre la descripción y el URL del artículo de la Wikipedia:

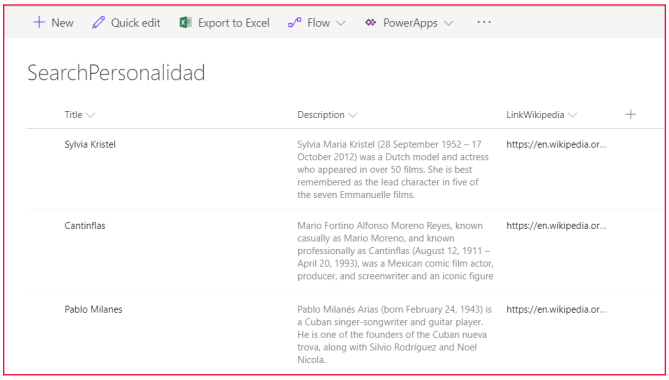


Imagen 2.- Personajes y su información utilizando el servicio de Bing API.

Conclusiones

El servicio de Bing de Azure permite enriquecer la información que los usuarios guardan en SharePoint. El Azure Bing API es fácil de utilizar desde cualquiera lenguaje de programación, y produce resultados confiables rápida y seguramente. El API utiliza algoritmos de Inteligencia Artificial que se mejoran con el uso, por lo no es necesario crear ni entrenar algoritmos propios.

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

19

Apply a consistent branding to all SharePoint sites

SharePoint modern experience is available for Lists, Document Libraries and Pages and while it delivers new functionalities to the end users it lacks some of the customization options available in the classic SharePoint.

The modern experience is provided with a theme that is responsible to format the colours of the elements in the page. On modern site collections like team and communication sites there's a native interface that allows the administrator to select the theme, but on classic site collections things are a little bit different and the modern experience will receive a random color applied automatically.

Requirements

To achieve the steps described in this article you will need to install the SharePoint Color Pallet Tool, SharePoint PnP PowerShell and SharePoint Online Management Shell. SharePoint Color Pallet Tool provides the color palette functionality to use with SharePoint designs. SharePoint PnP PowerShell and SharePoint Online Management Shell provide a set of commands that allows you to perform complex operations towards SharePoint simplifying the process using PowerShell.

"The modern experience is provided with a theme that is responsible to format"

How to change the color of SharePoint modern experience within a classic site collection

To change the color of SharePoint modern experience within a classic site collection just follow bellow steps:

1.– Open the SharePoint Color Pallet Tool.

2.– Click on the color tile next to the Recolor button.

3.– Introduce your main color for the theme, it needs to be provided in RGB.

4.– Click Recolor, modify the individual color slots as needed.

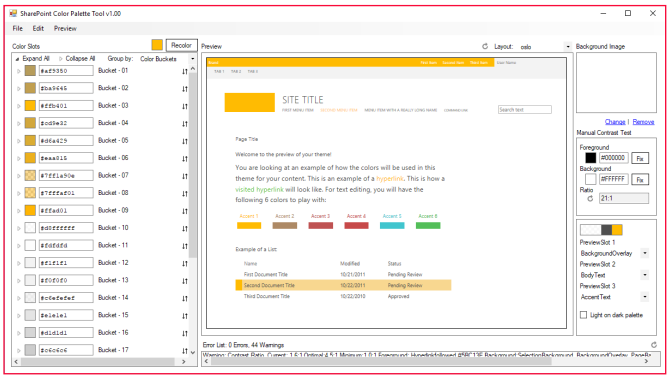


Image 1 - SharePoint color pallete tool.

5.– Go to File and Save the color scheme.

6.– Copy or download the PowerShell script bellow.

```
cls
$themeName = "Yellow"

#URL of the site collection
$targetSiteCollection = "https://contoso.sharepoint.com/"

#Path of the source folder containing your color file (ends with \)
$filePath = "C:\path\to\file\"

#File name including extension
$fileName = "yellow.spcolor"

#Autenticate on SharePoint Online site collection, credentials might be requested
Connect-PnPOnline -UseWebLogin -Url $targetSiteCollection

#Get the relatative url of the site collection
$relativeWebUrl = Get-PnPWeb
$relativeWebUrl = $relativeWebUrl.ServerRelativeUrl

#Path to the theme gallery
$targetDir = "/_catalogs/theme/15"

#Upload the theme file to the themes library
Add-PnPFile -Path ($filePath+$fileName) -Folder $targetDir

#Register the theme on the Composed Looks list
Add-PnPListItem -List "ComposedLooks" -ContentType "Item" -Values @{"Title"=$themeName; "Name"=$themeName; "MasterPageUrl"=$relativeWebUrl+"/_catalogs/masterpage/seattle.master, "+$relativeWebUrl+"/_catalogs/masterpage/seattle.master"; "ThemeUrl"=$relativeWebUrl+"/_catalogs/theme/15/"+$fileName+", "+$relativeWebUrl+"/_catalogs/theme/15/"+$fileName+""; "DisplayOrder"="1"}

#Reset all sites in the site collection to use the uploaded theme
$pathToColorFile = $relativeWebUrl+"/_catalogs/theme/15/"+$fileName
write-host "Applying the theme..."
Set-PnPTheme -ColorPaletteUrl $pathToColorFile -ResetSubwebsToInherit
```

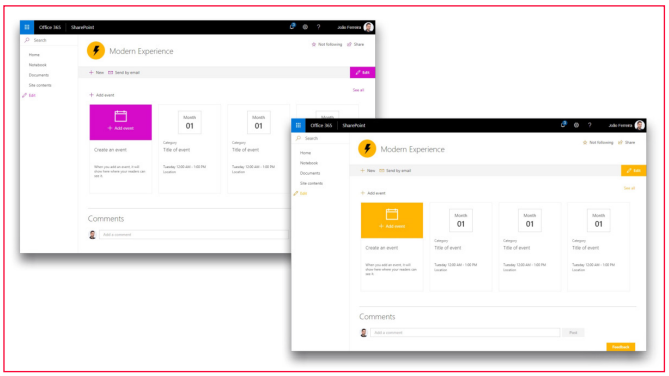
7.– Modify the variables $themeName, $targetSiteCollection and $filePath to reflect your values The script automates the following tasks:

- Upload the color file to the Theme library.
- Register the theme on the Composed Looks list.
- Apply the theme to all the sites of the site collection.

8.– Execute the script on the PowerShell console e.g. ./ApplyNewColorSiteCollection.ps1.

9.– If requested authenticate on SharePoint Online.

The process might take a few minutes depending of the number of sites you have in your site collection; the theme will be applied to all the existent sites but unfortunately it will not be applied to sites created after the execution of the script.



Iamgen 2 - Before and after.

How to change the color of SharePoint modern Team and Communication sites

If you are using exclusively modern team or communication sites, Microsoft as a solution that will allow you to pick a theme from a graphical user interface, the new theming experience allow site owners to apply themes to all modern pages in the site collection, to deliver new engaging and familiar looks.

"there are 8 themes available but you as an Administrator can create and deploy yours"

By default, there are 8 themes available but you as an Administrator can create and deploy yours. To create the color themes Microsoft provides the Theme Generator, an online tool that generates the color palette to be deployed in the tenant.

In the tool you will be able to introduce the primary theme color, body text color and body background color. The Fabric palette is generated in three formats, JSON, SASS and PowerShell, as an Admin you will need to use the PowerShell version.

1.– Open the Theme Generator.

2.– Define your colors.

3.– Save the PowerShell version of the schema.

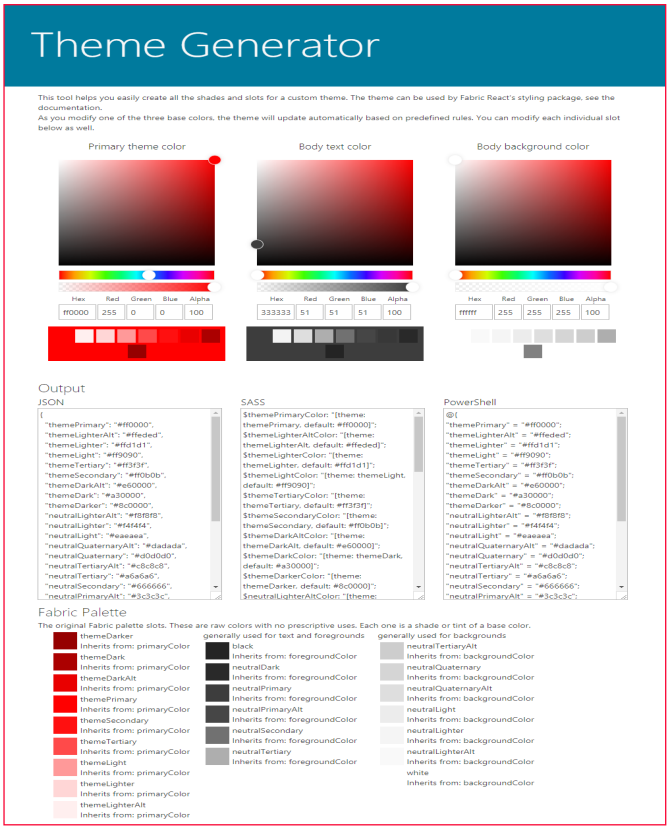


Imagen 3 - SharePoint theme generator.

Deploy the theme

To deploy new SharePoint themes, you will need to use the SharePoint Online Management Shell, and you will need to be a SharePoint Admin.

1.– To connect to your SharePoint Online environment open PowerShell and execute the command:

```
Connect-SPOService -Url https://contoso-admin.sharepoint.com
```

2.– In the console create a PowerShell variable with the generated color pallet.

```
$customTheme = @{
“themePrimary” = “#ff0000”;
“themeLighterAlt” = “#ffeded”;
“themeLighter” = “#ffd1d1”;
“themeLight” = “#ff9090”;
“themeTertiary” = “#ff3f3f”;
“themeSecondary” = “#ff0b0b”;
“themeDarkAlt” = “#e60000”;
“themeDark” = “#a30000”;
“themeDarker” = “#8c0000”;
“neutralLighterAlt” = “#f8f8f8”;
```

```
"neutralLighter" = "#f4f4f4";
"neutralLight" = "#eaeaea";
"neutralQuaternaryAlt" = "#dadada";
"neutralQuaternary" = "#d0d0d0";
"neutralTertiaryAlt" = "#c8c8c8";
"neutralTertiary" = "#a6a6a6";
"neutralSecondary" = "#666666";
"neutralPrimaryAlt" = "#3c3c3c";
"neutralPrimary" = "#333333";
"neutralDark" = "#212121";
"black" = "#1c1c1c";
"white" = "#ffffff";
"primaryBackground" = "#ffffff";
"primaryText" = "#333333";
"bodyBackground" = "#ffffff";
"bodyText" = "#333333";
"disabledBackground" = "#f4f4f4";
"disabledText" = "#c8c8c8";
}
```

3.– To deploy the theme run the command below, define the custom Name that will be visible on SharePoint online and in the Palette argument reference the variable created on step #2

```
Add-SPOTheme -Name "Contoso Theme" -Palette $customTheme -IsInverted $false -Overwrite
```

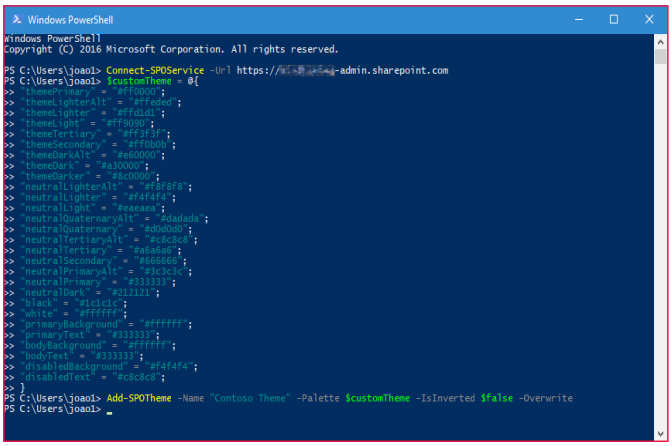


Image 4 - Installing theme.

"There are several ways to apply a consistent branding to SharePoint sites depending the version and type of sites"

4.– To validate if the theme was successfully deployed run the command below and validate if it is listed

```
Get-SPOTheme
```

Apply the theme

To apply the theme deployed:

1.– Open your modern site and click on the cog icon.

2.– Click on Change the Look.

3.– Select your custom theme and click Apply.

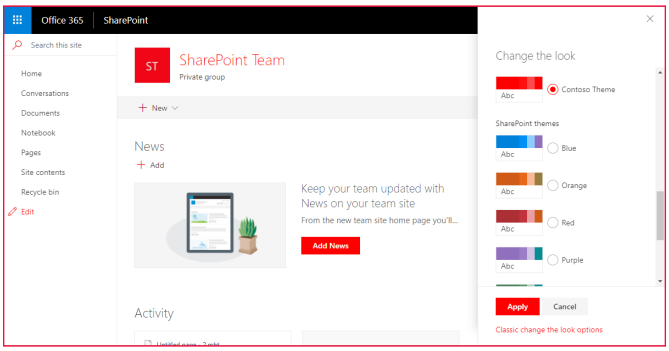


Imagen 5.- Change the look.

Conclusion

There are several ways to apply a consistent branding to SharePoint sites depending the version and type of sites used but with the methods explained in this article you will be able to automate the process and maintain your intranet with a same look and feel across multiple site collections.

Note: When compared with the classic themes, the modern version will only change the colors, the option to change the font and the background image is only available to the classic themes.

JOÃO FERREIRA
SharePoint developer | BindTuning
Twitter: joao12ferreira
Blog: http://handsonsharepoint.net

22

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

Introducción

En el número anterior vimos como una introducción a ReactJS para poder abordar un proyecto de este tipo. Una vez ya hemos tenido la primera toma de contacto con la librería y vamos a empezar un nuevo desarrollo utilizando este Framework nos van a surgir muchas preguntas y tendremos que tomar algún tipo de decisión (la cual puede ser acertada o no). Hay nuevos elementos nuevos que tendremos que decidir si incorporarlos o no dentro de nuestro desarrollo, su utilización puede cambiar radicalmente nuestra forma de llevar a cabo dicho desarrollo.

¿Qué aspectos deberemos de incorporar a nuestros desarrollos?

- Un "route" que se encargue de que dependiendo de la url que se muestre se cargue un componente u otro.
- Un "almacenamiento" para guardar el estado de los componentes.
- Que arquitectura vamos a implementar.

Arquitectura Flux y Redux

Empecemos por los cimientos de cómo implementar una arquitectura Flux. Si hay algo que ha cambiado realmente la forma de implementar las arquitecturas en el Front es este patrón, para mí lo más importante que ha hecho ReactJS es la incorporación de este patrón y el cambio de pensamiento a la hora de abordar un desarrollo. Flux es una arquitectura para el manejo y el flujo de los datos en una aplicación web. Fue ideada por el equipo de Facebook siendo su funcionalidad principal facilitar el manejo de datos en aplicaciones web con cierto grado de complejidad.

"ha cambiado realmente la forma de implementar las arquitecturas en el Front"

Estamos acostumbrados a las arquitecturas MVC en la que hay un flujo de datos bidireccional, es decir cualquier modificación en el servidor se modifica en la vista y viceversa, esto hace que en flujos complejos los problemas de rendimiento están a la orden del día. Con Flux esto cambia, propone una arquitectura en la que el flujo de datos es unidireccional. Los datos viajan desde la vista por medio de acciones y llegan a un Store desde el cual se actualizará la vista de nuevo.

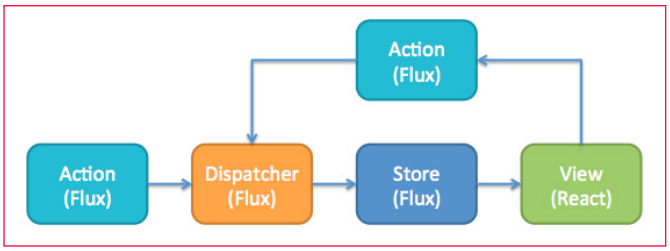


Imagen 1.- Flujo de Flux.

Teniendo todo el flujo de la aplicación centralizado es mucho más sencillo depurar las aplicaciones y encontrar los errores en la misma.

Que actores entran en juego en una arquitectura Flux:

- Vista: Serían los propios componentes de React.
- Store: Guarda los datos de la aplicación. No hay métodos en la store que permitan modificar directamente sobre ella, se tiene que hacer a través de dispatcher y acciones.
- Actions o Acciones: Una acción es simplemente un objeto que indica una intención de realizar algo y que lleva datos asociados en caso de ser necesario.
- Dispatcher: No es más que un mediador entre la Store y las acciones. Sirve para desacoplar la Store de la vista, ya que así no es necesario conocer que Store maneja una acción concreta.

El flujo que sigue la aplicación sería el siguiente:

- La vista, mediante un evento envía una acción con la intención de realizar un cambio en el estado.
- La acción contiene el tipo y los datos, y es enviada al dispatcher.
- El dispatcher propaga la acción al Store y se procesa en orden de llegada.
- El Store recibe la acción y dependiendo del tipo recibido, actualiza el estado y notifica a las vistas de ese cambio.
- La vista recibe la notificación y se actualiza con los cambios.

Este patrón se puede implementar bien de forma propia o bien utilizando alguna librería como pueda ser Redux, Re-

Flux, Fluxxor, Fluxible, etc… De todas ellas la más utilizada es Redux, es una pequeña librería de menos de 2kb y que con unos pocos métodos implementa el patrón Flux. Es agnóstica al framework por lo que esta se puede implementar en otros frameworks como Angular, Vue, etc.

¿Qué hace Redux?

Se encarga en cierta manera de desacoplar el estado global de una aplicación web de la parte visual. El estado de la aplicación pueden ser varias cosas, normalmente se trata los datos que se reciben a través de peticiones a servicios REST (consultas a listas de SharePoint). Pero también se refiere al estado de la UI en un determinado momento, por ejemplo: mostrar una información al usuario o no, un mensaje de error, ocultar desplegar un panel, etc.

Los conceptos claves de Redux:

1.– La Store=> La única fuente de datos, aunque el patrón Flux indica que pueden haber más de una store, Redux simplifica unificando todo en un único árbol.

2.– El Estado => Solo podemos modificar el estado a través de acciones

3.– Reducers=> Es básicamente una función que recibe dos parámetros, el estado inicial y una acción y dependiendo del tipo de acción realizará una operación u otra en el estado.

SPFx, ReactJS y Redux

Ahora bien, todo esto sería pensando que estamos haciendo una aplicación "normal" de ReactJS y que no estamos desarrollando un webpart de "SPFx". Cuando abordamos un desarrollo en SPFx tenemos que tener en cuenta los mismos planteamientos que para el desarrollo de un WebPart:

- Podemos poner varios WebParts en la misma página.
- La navegación la gestiona SharePoint.
- Tiene que haber comunicación entre los distintos WebParts.
- Es una SPA.

Todo esto que estamos hablando más que propio del desarrollo de React en SPFx es propio del desarrollo en SharePoint. ¿En qué circunstancias extiendo SharePoint mediante desarrollo? En el momento que la plataforma no cumple con los requerimientos del cliente y dichos requerimientos se adaptan perfectamente a SharePoint y hacen uso de alguno de los elementos out of the box de los que se componen. En caso de que estos requerimientos sean que se quiere un SharePoint que no se parezca a un SharePoint plantearos que quizás las necesidades que tiene el cliente no es el de utilizar SharePoint.

Ahora bien, ya hablando directamente de ReactJS, Redux, y SPFx. Partimos del propio contexto de SharePoint en el que en una página pueden tener más de un WebPart. ¿Es necesario utilizar Redux? Yo creo que Redux viene a cubrir un aspecto de aplicaciones complejas, y no en pequeños desarrollos de código que son lo habitual en SPFx. Además, como hemos comentado para implementar el patrón Flux no es necesario el añadir una complejidad mayor en la aplicación. Pero en el caso de que lo vayamos a utilizar vamos a ver cómo utilizarlo.

Show me the code, Talk is cheap

Dado un proyecto en ReactJS, uno de los aspectos que debemos de tener en cuenta para poder un mantenimiento óptimo de nuestro desarrollo es tener clara la organización que vamos a hacer. Por ahora en los proyectos que hemos desarrollado, hemos llegado a la siguiente convención:

- Una carpeta Reducer, donde tendremos un fichero por cada uno de los reducers que vamos a tener, un fichero index en el que se unificarán todos los reducers que vamos a utilizar.
- Una carpeta Action donde tendremos las acciones de los componentes, del mismo modo tendremos un fichero donde tendremos en constantes las acciones que vamos a utilizar (de esta forma evitamos los magic strings). Para acciones genéricas de algún componente también creamos una carpeta acction donde tendremos las acciones de dicho componente.
- Cada componente tendrá una parte de Frontnet y un contenedor que es el que se encargará de inyectar la implementación de dichos métodos.

"las arquitecturas MVC en la que hay un flujo de datos bidireccional, es decir cualquier modificación en el servidor se modifica en la vista"

Vamos a ver un ejemplo práctico, dada la plantilla de ReactJS que viene en Visual Studio en la versión de .NET Core 2.0 (la nueva versión cambia un poco ya que no utiliza TypeScript) En primer lugar lo que vamos a hacer es desde una línea de comandos instalar los siguientes paquetes y sus typings:

Nos crearemos una carpeta Reducer, donde vamos a tener un fichero UserProfile.ts

```
class UserProfileState {
 firstname : string;

 constructor() {
  this.firstname = "Default name";
 }
}

export const userProfileReducer =  (state : UserProfileState =
new UserProfileState(), action) => {
 return state;
};
```

Y como hemos comentado también tendremos un fichero index.tx en el que tendremos todos nuestros reducers, en este ejemplo solamente uno

```
import { combineReducers } from 'redux';
import { userProfileReducer } from './userProfile';

export const reducers = combineReducers({
 userProfileReducer
});
```

A continuación, crearemos un componente HelloWorldComponent.tsx de la siguiente forma:

```
import * as React from 'react';

export const HelloWorldComponent = (props : {userName : string}) => {
 return (
  <h2>Hello Mr. {props.userName} !</h2>
 );
}
```

También crearemos el contenedor de dicho componente:

```
import { connect } from 'react-redux';
import { HelloWorldComponent } from './helloWorld';

const mapStateToProps = (state) => {
 return {
  userName: state.userProfileReducer.firstname
 }
}

const mapDispatchToProps = (dispatch) => {
 return {
 }
}

export const HelloWorldContainer = connect(
 mapStateToProps,
 mapDispatchToProps
)(HelloWorldComponent);
```

El siguiente paso es en el punto de arranque de nuestra aplicación indicar que arranque con nuestro componente contenedor e indicarle la store (donde vamos a tener almacenado todo el estado de nuestra aplicación y que Redux se encarga de darnos un método para ahorrarnos dicha implementación).

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { createStore } from 'redux';
import { Provider } from 'react-redux';
import {reducers} from './reducers';
import {HelloWorldContainer} from './helloWorldContainer';

let store = createStore(reducers);

ReactDOM.render(
  <Provider store={store}>
    <HelloWorldContainer/>
  </Provider>
 , document.getElementById('root'));
```

Ahora si arrancamos la aplicación se nos mostrará la pantalla un Hello Mr. Default Name. ¿Lo veis claro no? Vamos a darle una vuelta más a toda la magia de Redux y es dado este desarrollo vamos a añadir un nuevo componente en el que el usuario pueda introducir su nombre. ¿Cómo vamos a modificar el nombre? Se lanzará una acción que será la encarga de modificar el estado que hay en la store y una vez se modifica se volverá a renderizar el componente con dicho nuevo valor.

Para empezar, crearemos un fichero common/actionsEnums.ts con las constates de la acción que vamos a implementar:

```
export const actionsEnums = {
    UPDATE_USERPROFILE_NAME : 'UPDATE_USERPROFILE_NAME'
}
```

"uno de los aspectos que debemos de tener en cuenta para poder un mantenimiento óptimo de nuestro desarrollo"

Crearemos una carpeta con donde estará la acción que se va a lanzar: actions/updateUserProfileName.ts

```
import {actionsEnums} from '../common/actionsEnums';

export const updateUserProfileName = (newName : string) => {
 return {
  type: actionsEnums.UPDATE_USERPROFILE_NAME,
  newName : newName,
 }
}
```

A continuación, modificaremos el reducer para que cuando se desencadene dicha acción se modifique el estado de la store, para ello substituiremos el siguiente código:

```
import {actionsEnums} from '../common/actionsEnums';

class UserProfileState {
 firstname : string;

 constructor() {
  this.firstname = "Default name";
 }
}

export const userProfileReducer = (state : UserProfileState = new UserProfileState(), action) => {
switch (action.type) {
  case actionsEnums.UPDATE_USERPROFILE_NAME:
   return handleUserProfileAction(state, action);
 }

 return state;
};

 const handleUserProfileAction = (state : UserProfileState, action) => {
  return {
   ...state,
   firstname: action.newName,
  };
 }
```

Una vez implementada las Actions y los Reducers nos crearemos el componente NameEdit.tsx

```
import * as React from 'react';

export const NameEditComponent = (props: {userName : string, onChange : (name : string) => any}) => {
 return (
  <div>
   <label>Update Name:</label>
   <input
    value={props.userName}
    onChange={(e : any) => props.onChange(e.target.value)}
    />
  </div>
 );
}
```

Y su contenedor correspondiente NameEditContainer.tsx

```
import { connect } from 'react-redux';
import { NameEditComponent } from './nameEdit';
import {updateUserProfileName} from './actions/updateUserProfileName';

const mapStateToProps = (state) => {
 return {
  userName: state.userProfileReducer.firstname
 }
}

const mapDispatchToProps = (dispatch) => {
 return {
   onChange: (name : string) => {return dispatch(updateUserProfileName(name))}
 }
}

export const NameEditContainer = connect(
 mapStateToProps,
 mapDispatchToProps
)(NameEditComponent);
```

"un planteamiento muy diferente a lo que estamos acostumbrados para llevar a cabo un desarrollo en el FrontEnd"

Para mostrar los dos componentes nos crearemos un Componente Padre App que tendrá la siguiente estructura:

```
import * as React from 'react';
import {HelloWorldContainer} from './helloWorldContainer';
import {NameEditContainer} from './nameEditContainer';

export const App = () => {
 return (
  <div>
   <HelloWorldContainer/>
   <br/>
   <NameEditContainer/>
  </div>
 );
}
```

Si ejecutamos dicho código, lo que vamos a ver es que cada vez que en el componente se modifica el valor automáticamente el Hello World cambia de valor según lo que se ha introducido.

Aunque dicho ejemplo es muy simple sirve como claro concepto para entender cuál es el flujo que sigue una aplicación React. El entender dicho flujo es muy importante porque cuando empecemos a añadir complejidad a nuestra aplicación tenemos que tenerlo muy claro porque si no es muy sencillo que empecemos a odiar a React.

Resumen

Hemos visto un planteamiento muy diferente a lo que estamos acostumbrados para llevar a cabo un desarrollo en el FrontEnd. El patrón Flux ha venido para quedarse y el pensar de esta forma hace que nuestras aplicaciones sean más mantenibles y podamos dotarlas de un orden que hasta hace poco carecían. Ahora bien, no todo es color de rosa y detrás del patrón hay que entender muy bien el ciclo de vida de cada componente. Del amor al odio hay un paso, y en el proceso de adopción de React es posible que los primeros días el nivel del desarrollador este muy contento y le parezca lo mejor que ha probado. Cuando se empiezan a añadir dificultad al desarrollo es posible que lo odies e intentes plantear una solución a lo "clásico" y por último acabarás por entender su ciclo de vida y entenderás toda la potencia que tiene React.

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

26

Contenedores en Azure

El mundo del desarrollo de aplicaciones no para de evolucionar y las opciones de ejecución aumentan a un ritmo endiablado, tanto que es complicado mantenerse al día y más complicado tener proyectos que hagan uso de la tecnología más adecuada en cada momento.

Uno de los principales culpables de este ritmo frenético es la nuble pública. Mes a mes, vemos cómo evoluciona y aparecen servicios nuevos que permiten exprimir al máximo nuestras aplicaciones con arquitecturas de ejecución especializadas para los requisitos actuales.

Desde el punto de vista de la arquitectura de ejecución, la llegada de los contenedores ha sido una revolución y, no sólo para el maravilloso mundo de los microservicios, sino que también para las opciones Serverless que nos ofrecen. Microsoft y el equipo de producto de Azure lo sabe y es por ello por lo que tenemos diversas opciones para ejecutar, administrar y monitorizar contenedores en el cloud de Microsoft.

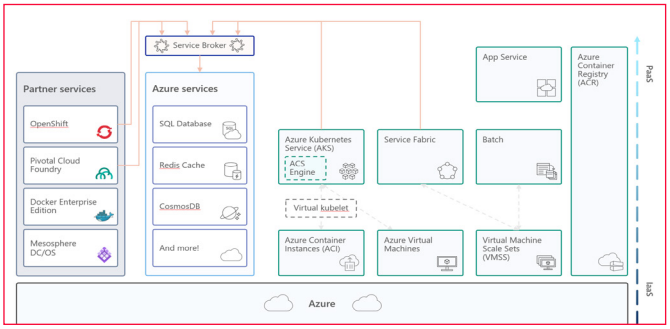


Imagen 1- Soporte de contenedores en Azure.

Como podemos ver en la imagen anterior de Microsoft, el soporte a contenedores en Azure no sólo se basa en los servicios donde ejecutar nuestros contenedores, sino que internamente existe un "Service Broker" encargado de dar soporte a las aplicaciones de otros Partners que usan contenedores.

Azure Container Registry

Este servicio es un repositorio de imágenes de contenedores del tipo DC/OS, Docker, Swarm, Kubernetes, Service Fabric, etc. Con Azure Container Registry podemos compilar, almacenar y administrar las imágenes de los contenedores de nuestras aplicaciones que estarán disponibles para que se instancien desde nuestros orquestador o servicio de contenedores favorito.

"Un servicio fundamental para publicar las versiones de nuestras aplicaciones de forma segura"

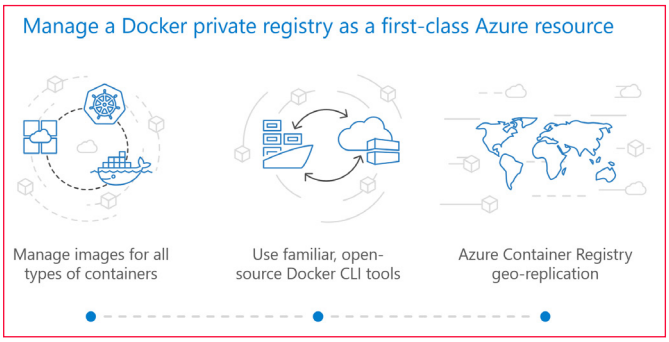


Imagen 2.- Funcionalidades base de Container Registry.

Un servicio fundamental para publicar las versiones de nuestras aplicaciones de forma segura y que se haga uso de ellas desde nuestros sistemas de microservicios.

Azure Kubernetes Service (AKS)

AKS es un servicio administrado de un cluster de Kubernetes que nos permite ejecutar aplicaciones en contenedores. Kubernetes es un sistema Open-Source para orquestar los contenedores de las aplicaciones basadas en microservicios, y Azure nos ofrece este servicio ahorrándonos todo el trabajo de instalar y configurar el cluster para preocuparnos de ejecutar nuestros contenedores y monitorizarlos.

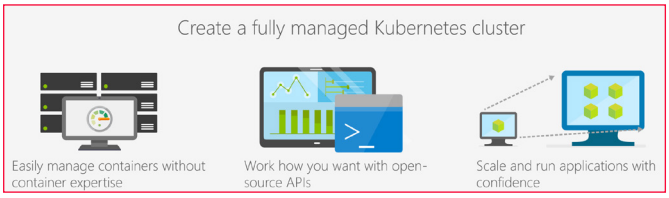


Imagen 3.- Un cluster de Kubernetes como servicio.

Por su puesto, aísla la aplicación de la infraestructura donde se ejecuta para resolver problemas escalado ante la demanda de los recursos o posibles fallos en las máquinas virtuales donde estamos ejecutando nuestros contenedores.

Azure Service Fabric

Service Fabric es la tecnología base que donde se ejecuta la infraestructura principal de Azure y otros servicios, como

Skype Empresarial, Intune, Azure Event Hubs, Azure Data Factory, Azure Cosmos DB, Azure SQL Database, Dynamics 365 y Cortana. Pensado para ejecutar sistemas distribuidos y abstraernos de la confiabilidad, la escalabilidad, la administración y la latencia que necesitan estos sistemas.

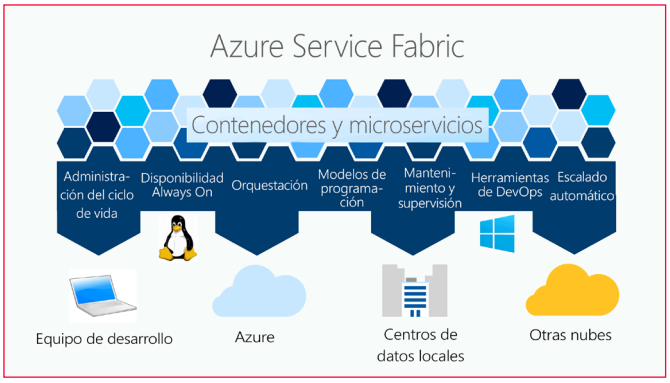


Imagen 4.- Funcionalidades de Service Fabric.

En resumidas palabras, Azure Service Fabric es el servicio administrado para ejecutar un cluster de Service Fabric en Azure, que nos permite orquestar nuestras aplicaciones de microservicios, bien mediante el uso de contenedores Docker o implementando un SDK específico para el desarrollo de estos servicios.

> ***"La ventaja de utilizar un contenedor, frente a una máquina virtual, es que podemos empaquetar todas las dependencias de ejecución en el contenedor"***

App Service Web App for Containers

Una versión de las aplicaciones web del App Service de Azure que nos ofrece la posibilidad de utilizar un contenedor donde se ejecuta, en cada instancia del App Service, nuestra aplicación. Una buena opción para ejecutar nuestra aplicación junto con sus dependencias, por ejemplo, para aplicaciones legacy que necesitamos ejecutar en Azure y sacar todo el provecho de la ejecución en un servicio PaaS, como el autoescalado.

Azure Batch

Azure Batch es un servicio de orquestación para ejecutar de procesamiento por lotes. El objetivo de este servicio es determinar el número de instancias de máquinas virtuales o nodos que se necesitan para ejecutar las tareas programadas en un trabajo predeterminado. Por ejemplo, para calcular el riesgo de una cartera de clientes mediante simulaciones de Monte Carlo, para ejecutar el renderizado de una película de animación o para ejecutar los algoritmos que permiten obtener el análisis del genoma de una muestra de un ser humano.

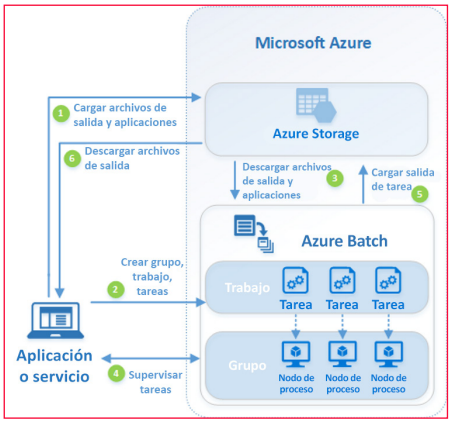


Imagen 5.- Arquitectura de ejecución de Azure Batch.

Como podemos ver en la imagen anterior, que define a alto nivel la arquitectura de ejecución, la ejecución de las tareas se delega a un nodo de proceso que no es más que una máquina virtual o un contenedor. La ventaja de utilizar un contenedor, frente a una máquina virtual, es que podemos empaquetar todas las dependencias de ejecución en el contenedor y despreocuparnos del entorno de ejecución que, al fin y al cabo, es una plantilla personalizada de máquina virtual en Azure.

Azure Service Fabric Mesh

Mesh nos ofrece la misma funcionalidad que tenemos en Service Fabric pero sin tener que preocuparnos de la administración del cluster o de las operaciones de parcheo del mismo. Azure Service Fabric Mesh permite desplegar microservicios en contenedores y escalar para satisfacer la demanda de trabajo.

Azure Container Instances

Por último, pero no por ello menos importante, tenemos la versión Serverless de contenedores. ACI nos ofrece la posibilidad de ejecutar contenedores sin preocuparnos del entorno de ejecución ni del cluster donde se ejecutará el mismo. Está pensando para la ejecución de aplicaciones basadas en eventos o para trabajos de procesamiento de datos, pero también podemos utilizarlo para extender la capacidad de AKS instancias microservicios bajo demanda que se ejecutan en este servicio.

Conclusiones

Los contenedores nos simplifican los desarrollos y facilitan la ejecución de las aplicaciones en entornos empresariales, sin embargo, no es la solución para todos los problemas. Estos servicios de Azure nos permiten ejecutar nuestros contenedores de una forma fácil, sencilla pero segura.

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

28

Un vistazo al nuevo Centro de Administración para Microsoft Teams y Skype For Business

A principios de abril de este año Microsoft anunció el inicio del rollout en tenants Target Release del nuevo Centro de Administración para Microsoft Teams y Skype For Business (TSBAC) que a la fecha de redacción de este artículo está todavía en Preview. En este artículo daremos un vistazo a este nuevo y moderno centro de administración.

Acceso al TSBAC

Para acceder al nuevo TSBAC tenemos dos posibilidades:

- A través del acceso a las configuraciones de Microsoft Teams a nivel global en Office 365: "Configuración->Servicios y complementos->Microsoft Teams" y en el panel de configuración hacemos clic en el enlace "centro de administración de Microsoft Teams y Skype Empresarial".

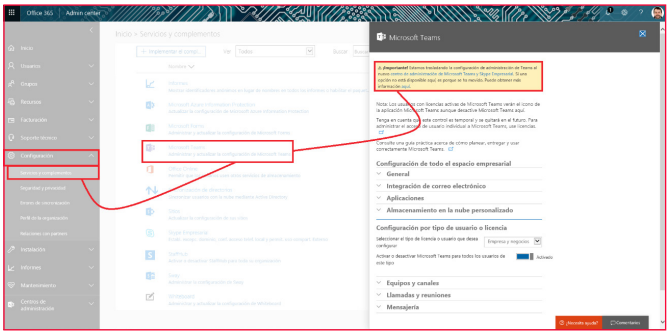


Imagen 1.- Acceso al TSBAC desde las opciones de configuración de Microsoft Teams.

- Directamente escribiendo la siguiente URL en el navegador:

https://admin.teams.microsoft.com/

Cualquiera de las dos opciones nos llevará al nuevo TSBAC que como es esperable cuenta con una apariencia moderna y en línea con la administración de Office 365 o la nueva administración de SharePoint Online.

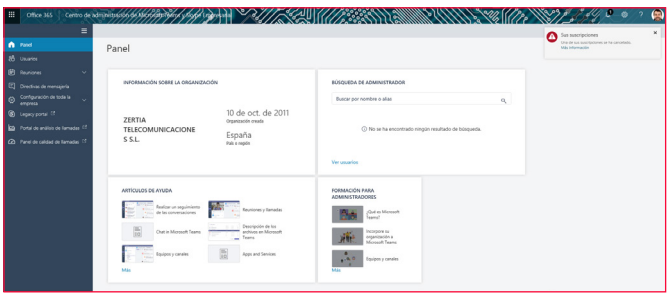


Imagen 2.- Nuevo TSBAC.

> ***"Cualquiera de las dos opciones nos llevará al nuevo TSBAC que como es esperable cuenta con una apariencia moderna"***

Secciones y opciones en el nuevo TSBAC

Lo primero que nos encontraremos en el nuevo TSBAC (Imagen 2) es una página principal con poco contenido por el momento en el que se visualiza la información de la organización, un buscador para el administrador y accesos directos a videos de Teams orientados tanto al usuario final como al administrador del servicio. Sin embargo, en el menú vertical nos encontraremos una serie de opciones interesantes para administrar tanto Microsoft Teams como Skype For Business:

- Usuarios muestra un listado de los usuarios de Microsoft Teams y de Skype For Business en el tenant. Si hacemos clic en uno de los usuarios, accederemos a la página de detalle del usuario en la que no solo se muestra información, sino que es posible realizar cambios en las configuraciones relativas a Teams y Skype for Business para el usuario como por ejemplo:
  - El puente de conferencia asignado para realizar y recibir llamadas.
  - La directiva de mensajería de Teams.
  - Configuraciones concretas de Microsoft Teams como por ejemplo indicar que Teams es la aplicación predeterminada para el usuario tanto para llamadas como para chats.

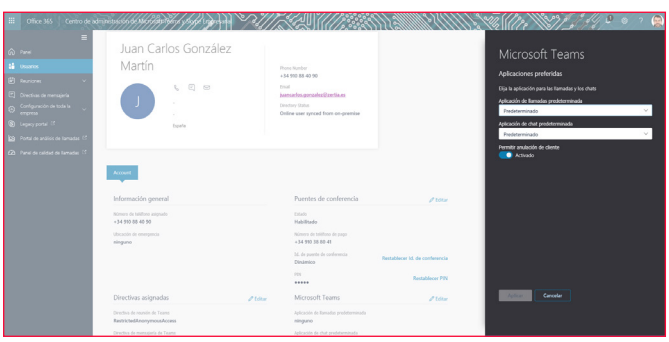


Imagen 3.- Detalle de un usuario en Microsoft Teams.

- Reuniones, por el momento solo proporciona acceso a la opción “Puentes de conferencia” que permite configurar números de teléfono tanto gratuitos como de pago que pueden ser utilizados por los usuarios para conectarse a las reuniones mediante llamada de teléfono.

“en uno de los usuarios, accederemos a la página de detalle del usuario en la que no solo se muestra información”

- Directivas de mensajería, muestra el listado de políticas de mensajería disponibles en el TSBAC y que se pueden asignar a los usuarios de Teams y de Skype For Business. El detalle de cada política se puede ver en el correspondiente panel de detalle y dependiendo de la política tendremos la posibilidad de editar la configuración de la misma o no. De las 4 directivas disponibles a la fecha de redacción de este artículo, sólo la directiva Global es modificable. Finalmente, comentar que desde esta sección es posible crear directivas personalizadas.

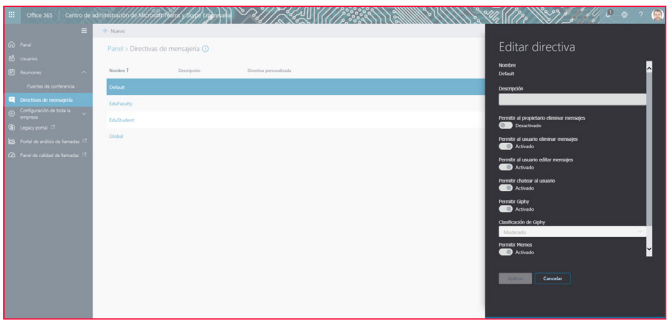


Imagen 4.- Directivas de mensajería.

- Configuración de toda la empresa, proporciona acceso a las siguientes configuraciones de Teams y Skype for Business:
  - Acceso externo que permite habilitar o no la comunicación con usuarios externos de Skype for Business o usuarios de Skype de consumo.
  - Usuarios invitados que permite configurar los permisos que van a tener usuarios invitados en Microsoft Teams.
- Legacy Portal, es simplemente un acceso directo al Centro de Administración legacy de Skype For Business.
- Portal de Análisis de Llamadas, permite acceder al portal citado en el que es posible por usuario revisar la calidad de las últimas llamadas realizadas haciendo uso de Microsoft Teams o de Skype For Business:

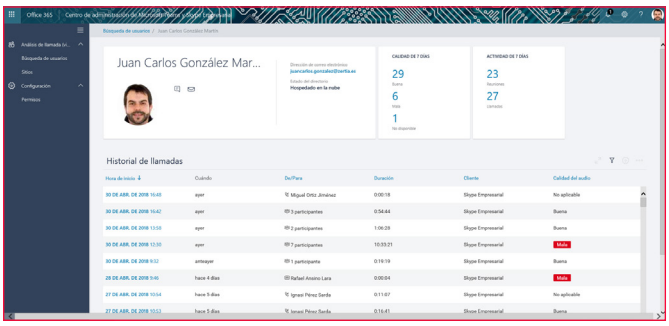


Imagen 5.- Portal de análisis de llamada.

“El nuevo Centro de Administración para Microsoft Team y Skype For Business es un nuevo centro moderno de administración”

Si hacemos clic en una llamada en concreto, podremos ver información detallada que se ha registrado de la misma como detalles del equipo usado para la conexión, características de conectividad al compartir pantalla, detalles de red, etc.

- Panel de calidad de llamadas, que nos proporciona acceso a una serie de informes de calidad de las llamadas realizadas tanto con Microsoft Teams como con Skype For Business.

Imagen 6.-Panel de calidad de llamadas.

Conclusiones

El nuevo Centro de Administración para Microsoft Team y Skype For Business es un nuevo centro moderno de administración desde el que se podrán realizar todas las configuraciones relativas tanto a Microsoft Teams como a Skype for Business en aspectos como directivas de mensajería, configuraciones de acceso externo y de invitados, etc.

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

30

Microsoft Identity Manager 2016

Aunque no sea su principal reclamo, EM+S incluye licenciamiento para Microsoft Identity Manager (MIM) 2016, una poderosa solución de gestión y sincronización de identidades locales, evolución de Forefront Identity Manager (FIM) 2010. Veamos cómo funciona y qué necesitamos para su instalación.

Sí, MIM 2016 es la evolución de FIM 2010, con el cual comparte no sólo la funcionalidad, sino que gran parte de la documentación escrita para FIM es válida también para MIM. Sobre todo en cuanto a planificación, requerimientos, e instalación, no hay que preocuparse si no encontramos mucha documentación de MIM 2016, pues la que existe de FIM 2010 es válida.

La documentación de MIM 2016 se queda corta y a veces es inexacta. No es caballo de batalla ni goza de la popularidad de otras soluciones, pero es una herramienta muy potente que se merece un poco de foco. Así que ¡vamos allá!

Imagen 1.- Microsoft Identity Manager.

¿Qués es MIM 2016?

MIM 2016, al igual que FIM 2010, es una solución de sincronización de identidades y accesos, que ayuda a gestionar usuarios, grupos, credenciales, directivas, y accesos. Además de las funcionalidades ya existentes en FIM 2010, MIM 2016 agrega:

- Informes híbridos en Azure, para visualizar datos cloud y locales en un único sitio.
- Privileged Access Management (PAM): gestión de credenciales con privilegios.
- Nuevas funcionalidades en la gestión de certificados.
- El Autoservicio ahora incluye desbloqueo de cuentas y Azure MFA.

Arquitectura

Los principales componentes de una arquitectura MIM 2016 son el Portal, el Servicio MIM, el Servicio de Sincronización, y SQL Server.

- MIM Portal: interfaz web de usuario para gestión de grupos, exploración de usuarios, reseteo de contraseñas, y tareas administrativas. Va montado sobre SharePoint.
- MIM Service: servicio web que permite implementar la funcionalidad de MIM 2016.
- MIM Sync Service: servicio de sincronización de cuentas y atributos, que se conecta con otras fuentes de identidades como Directorio Activo o el propio servicio MIM.

> ***"una poderosa solución de gestión y sincronización de identidades locales, evolución de Forefront Identity Manager"***

Todos los componentes pueden desplegarse en el mismo servidor, aunque lógicamente este escenario no contempla escalados ni balanceos, y sería más idóneo para pilotos o pruebas de concepto. Lo más común es separar los diferentes servicios por máquinas, y así obtener más flexibilidad, ya que esta arquitectura permite escalar los diferentes componentes. Por ejemplo, podemos agregar un nuevo servidor SharePoint con un balanceador para escalar el portal y servicio MIM, y además aportar alta disponibilidad. Lo mismo ocurre con las bases de datos. Podemos escalar el servidor SQL, e incluso desplegar grupos de disponibilidad Always On, dependiendo de la versión instalada.

https://blogs.technet.microsoft.com/iamsupport/2017/03/28/support-info-microsoft-identity-manager-2016-sp1-hotfix-4-4-1459-0-released/

También podemos separar las instancias de Servicio MIM de usuario y administrador, y de esta forma no penalizar la experiencia de usuario por culpa de los flujos de sincronización a nivel administrador.

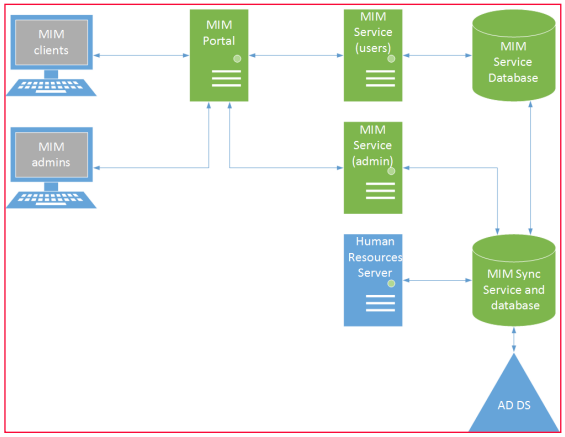


Imagen 2.- Topología de alta disponibilidad de MIM 2016.

Sea cual sea la topología elegida, podemos desplegar todos los componentes de MIM en máquinas físicas, virtuales, o en Azure.

Características especiales

Además de las características de gestión y sincronización de identidades que MIM 2016 ofrece con una instalación Standard, existen otras características opcionales que extienden su funcionalidad.

- MIM 2016 Certificate Manage (CM): gestión de smartcards, certificados de usuario, tokens de hardware.
- Privileged Access Management (PAM): control y gestión de credenciales y accesos de administradores.
- Self-service password reset (SSPR): permite a los usuarios resetear sus contraseñas y desbloquear sus cuentas en el portal de MIM.
- Password change notification service: sincronización de contraseñas de AD hacia otros sistemas de identidades sincronizados con MIM.
- Hybrid reporting: permite presentar informes de actividad de MIM en los cuadros de actividad de Azure.

Instalación

Para la sincronización de identidades necesitamos instalar MIM Sync Service y un servidor SQL para alojar sus BBDD. Si no vamos a crear flujos o reglas de sincronización, no necesitamos nada más. Crearíamos los conectores (los Management Agents de FIM 2010), los perfiles de ejecución en la consola de sincronización, y listo. ¡A sincronizar identidades!

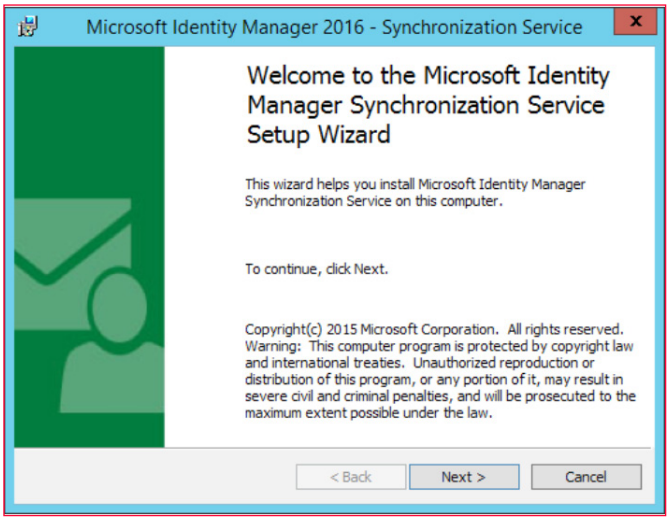


Imagen 3.- Instalación de MIM Sync Service

Para cualquier otra funcionalidad necesitaremos por lo menos el Servicio MIM y el Portal MIM, cada uno de ellos con su correspondiente instancia de SQL Server. El Portal MIM va montado sobre SharePoint, así que los requerimientos mínimos de Hardware serán los de la versión de SharePoint Server que vayamos a instalar.

https://docs.microsoft.com/en-us/microsoft-identity-manager/capacity-planning-guide

La documentación oficial de Microsoft describe el proceso de instalación sobre un único servidor, aunque aplicarlo a otra topología no será un problema.

MIM 2016 requiere SP1 para soportar SharePoint 2016, y además habrá que adaptar lo indicado en la documentación, que contempla SharePoint 2013 Foundation.

La principal modificación para desplegar MIM sobre SharePoint Server 2016 será cambiar el parámetro CompatibilityLevel de 14 a 15 al momento de crear la Colección de Sitios.

https://docs.microsoft.com/en-us/microsoft-identity-manager/microsoft-identity-manager-deploy

Conectando directorios

Los conectores enlazan diferentes fuentes de identidades, a través del servicio de sincronización de MIM.

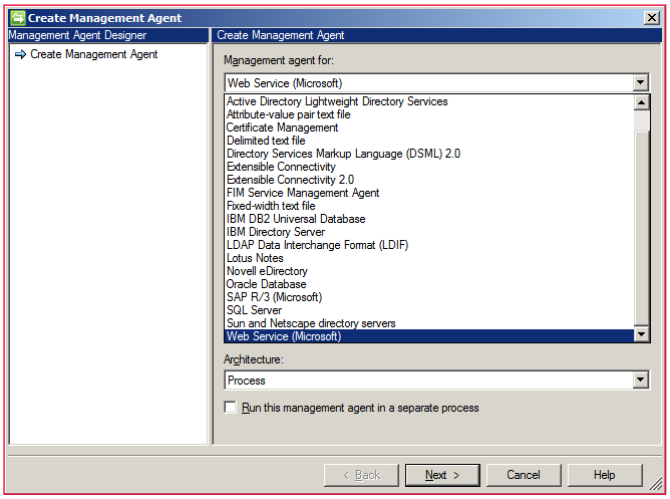


Imagen 4.- Crear conector.

El servicio MIM Sync ejecuta los trabajos de sincronización y Export para mover datos entre las diferentes fuentes, y actualizar atributos de identidades. Para que todo esto sea posible, necesitamos instalar y configurar el conector adecuado para cada origen de datos, incluso para el servicio MIM, pues desde el punto de vista de la sincronización, el servicio de MIM es una fuente de identidades más.

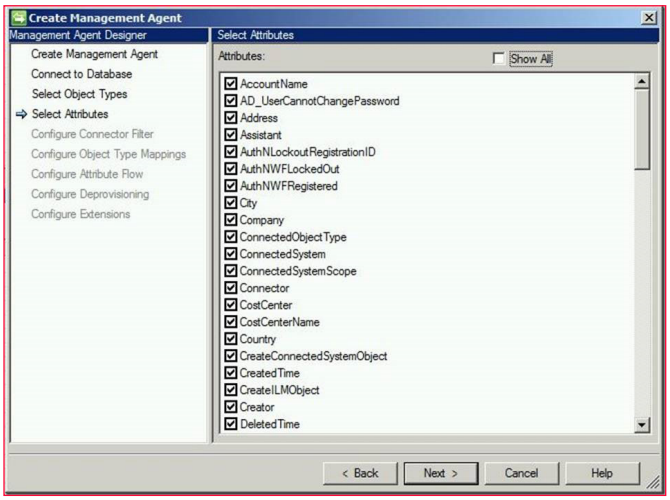


Imagen 5.- Configuraciones del conector.

"podemos separar las instancias de Servicio MIM de usuario y administrador, y de esta forma no penalizar la experiencia de usuario"

Hay una gran cantidad de conectores incluidos en MIM 2016:

https://docs.microsoft.com/en-us/microsoft-identity-manager/supported-management-agents

Además, el conector Extensible Connectivity 2.0 permite crear más conexiones, como los creados por algunos Partners:

http://social.technet.microsoft.com/wiki/contents/articles/1589.fim-2010-management-agents-from-partners.aspx

Use case: Azure AD

Pregunta: ¿Es posible sincronizar identidades con Azure AD sin utilizar Azure AD Connect? ¿Y sin AD local? Pues, aunque suene raro la respuesta es sí.

https://docs.microsoft.com/en-us/previous-versions/mim/dn511001(v=ws.10)

Utilizando el conector de Azure AD para MIM 2016 podemos sincronizar identidades de cualquier fuente conectada, hacia Azure AD. En el siguiente enlace podemos ver una comparativa de las herramientas de sincronización con Azure AD:

https://docs.microsoft.com/en-us/azure/active-directory/active-directory-hybrid-identity-design-considerations-tools-comparison

Reglas y Workflows

Pero tal vez lo más interesante de la gestión de identidades, tanto en MIM 2016 como en sus antecesores, sean los workflows y las reglas de negocio que podemos configurar para asegurar, de manera automática, que los usuarios tienen los permisos que necesitan en los diferentes sistemas.

Management Policy Rules (MPR) son reglas que actúan como reglas de negocio y tienen dos funciones fundamentales: por un lado, definen lo que ocurre cuando llega una petición a MIM. Por ejemplo, mediante estas reglas podemos permitir que los usuarios lean y/o editen sus atributos, soliciten acceso a determinados grupos, o vean el perfil de otros usuarios. Es decir, con las MPR se otorgan permisos para gestionar objetos.

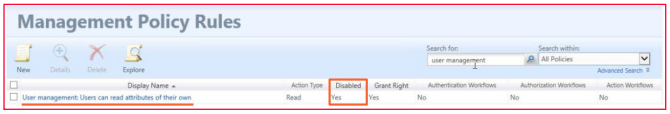


Imagen 6.- Management Policy Rules.

Por otro lado, las MPR actúan como disparadores de Workflows que realizan determinadas acciones en MIM 2016. Podemos, por ejemplo, crear una MPR que dispare un Workflow para actualizar los grupos a los que pertenece un usuario según el valor del atributo "Departamento".

https://docs.microsoft.com/en-us/previous-versions/mim/ff356871(v=ws.10)

Conclusión

Si tienes las mismas identidades repartidas en diferentes sistemas, estarás usando algún tipo de validación, ya sea manual o automatizada. En cualquiera de los dos casos, Microsoft Identity Manager 2016 ayuda a ahorrar muchísimo tiempo, a través de la sincronización, la automatización de tareas, y el autoservicio que permite a los usuarios gestionar sus propios perfiles.

PABLO ORTIZ BAIARDO
Infrastructure & Cloud Consultant @ Tokiota
ortiz.pablo@gmail.com
@portiz2017
https://www.linkedin.com/in/portiz

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.

34

Azure AD B2C

Cuando una empresa desarrolla una aplicación para sus clientes, externos a su organización, uno de los puntos claves es el desarrollo que permitirá el acceso a esta aplicación. ¿Qué puntos se deben tener en cuenta?

- Permitir varios proveedores de identidad:
  - Registro.
  - Proveedores de terceros: Twitter, Facebook, LinkedIn, etc.
- Permitir autenticación multi factor.
- Restablecimiento de contraseña.
- Posibilidad de implementar políticas según los casos.
- SSO entre diferentes.

Para poder dar salida a todos estos requerimientos tenemos dos opciones:

- Implementamos el servicio de identidad: ya sea uno custom, con identity server…
- Utilizamos un servicio ya implementado y preparado para ello.

En mi caso, a los clientes, es la segunda opción la que les aconsejo, y además les aconsejo utilizar Azure AD B2C. El objetivo de este artículo es explicar un que es Azure AD B2C, que podemos hacer con él y de esta forma entender porque utilizarlo.

Pilares de Azure AD B2C

Los pilares sobre los que se basa Azure AD B2C son cinco:

- Cuentas locales y sociales.
- Autenticación Multi-Factor.
- Políticas de: sign up, sign in, sign up o sign in, password reset y profile editing.
- Customización de la experiencia de usuario: Pantalla login, registro, edición perfil usuario y reseteo de password.
- Fácil integración en el desarrollo de aplicaciones que utilicen Azure AD B2C.

Cuentas locales y sociales

Si queremos dar un buen servicio a nuestros usuarios, una de las opciones que debemos darle es que pueda realizar el login en nuestra aplicación mediante alguna de sus cuentas de redes sociales que ya utilicé. Hay muchos usuarios que prefieren utilizar una única identidad en todas las aplicaciones para no tener infinitas cuentas, otros utilizar varias y otros que prefieren tener una por sistema.

> ***"clientes, externos a su organización, uno de los puntos claves es el desarrollo que permitirá el acceso a esta aplicación"***

Resumiendo, Azure AD B2C brinda a los usuarios finales la posibilidad de elegir entre:

- "Traer su propia identidad" (BYOI) mediante el uso de una de sus cuentas sociales como Amazon, Facebook, Google+, LinkedIn, cuenta de Microsoft (MSA), etc.
- Crear una nueva cuenta local (dirección de correo electrónico arbitraria).

Por defecto cuando creas una cuenta de Azure AD B2C solo tienes la opción de crear cuentas locales:

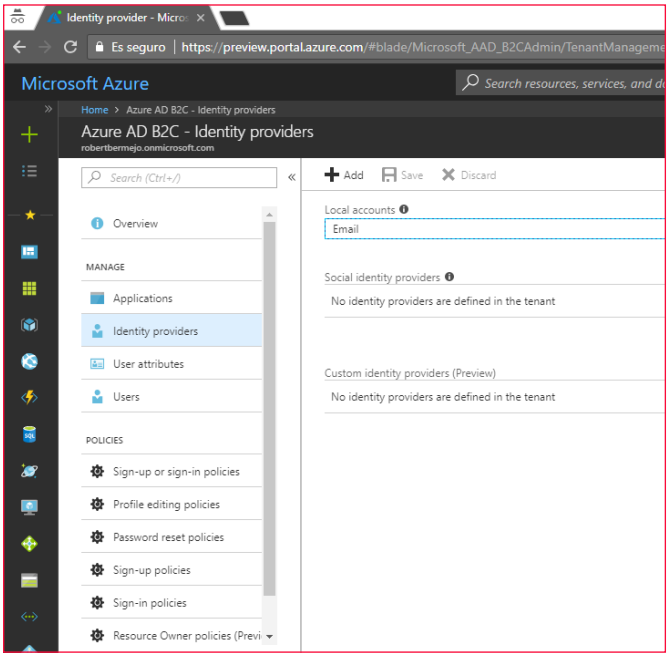


Imagen 1.- Creación de una cuenta local en Azure AD B2C.

Para dar la opción de poder interactuar con cuentas externas, hay que darle a la opción Add y activar el proveedor deseado. Actualmente las opciones son: Microsoft Account, Google, Faebook, LinkdIn, Amazon, Weibo (Preview), QQ (Preview), WeChat (Preview), Twitter (Preview), GitHub (Preview).

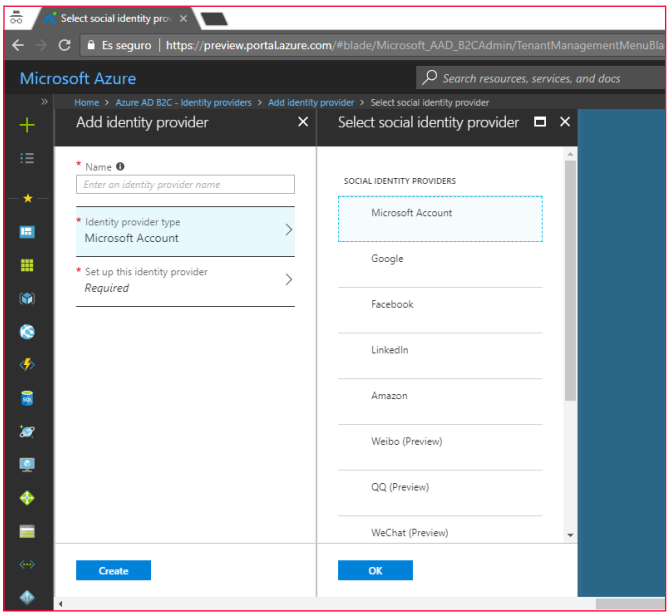


Imagen 2.- Proveedores de autenticación externo disponibles en Azure AD.

Para cada una de ellas deberás introducir el ClientId y el ClientSecret obtenido de la plataforma de origen. Una vez habilitadas a nivel de Tenant, cuando creemos o editamos una política podremos seleccionar que tipo de cuenta se requiere para esa política concreta.

Autenticación Multi-Factor

Azure AD B2C permite una segunda capa de seguridad a la hora de registrarse o logarse. Actualmente esta segunda capa solo se puede hacer mediante llamadas de teléfono y mensajes de texto. La autenticación Multi-Factor se activa o desactiva cuando se crea o edita una política. La razón por la que se hace a nivel de políticas es porque de esta forma se puede administrar el Multi-Factor por aplicación, no todas nuestras aplicaciones tienen porque realizar Multi-Factor, o incluso a nivel de partes concretas de una aplicación.

Políticas

Las políticas son la parte más importante de Azure AD B2C. En ellas se definen cuál será la experiencia del usuario cuando inicie una sesión, edite su perfil, se registre en la aplicación o cambié su contraseña. Cuando creamos una política, está nos permite configurar los siguientes puntos:

- Tipo de cuenta que se puede usar para el registro: local o alguna de terceros permitida (Facebook, Twitter…).
- Atributos que son necesarios para identificar a los clientes: Nombre, dirección, teléfono.
- Uso de autenticación Multi-Factor.
- URL de la página customizada.
- Claims: La información que se añadirán a los claims cuando el usuario esté logado.

¿Y cómo Azure B2C sabe que política aplicar en cada momento? Cuando se hace la llamada https uno de los campos que se deben enviar es la política a aplicar. Por ejemplo, si tuviéramos una policita de Sign-in con el nombre de siginpolice la llamada https de login de usuario sería:

https://login.microsoftonline.com/contosob2c.onmicrosoft.com/oauth2/v2.0/authorize?client_id={El id de tu aplicación registrada}&redirect_uri={url registrada de respuesta}&response_mode=form_post&response_type=id_token&scope=openid&nonce=dummy&&p=siginpolice(que es nuestra política definida)

Se pueden definir las siguientes políticas:

- Sign-up or Sign-in: Política que nos sirve tanto para cuando un usuario haga login como si se registra.

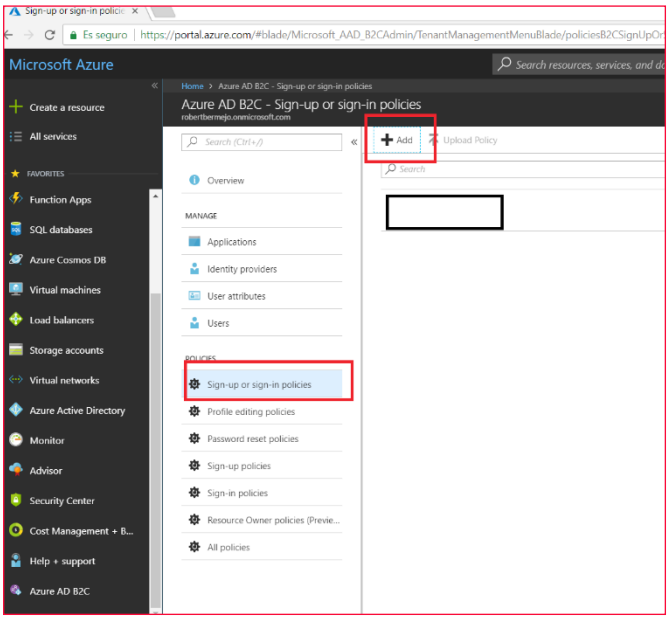


Imagen 3.- Política de Sign-up o Sign-in.

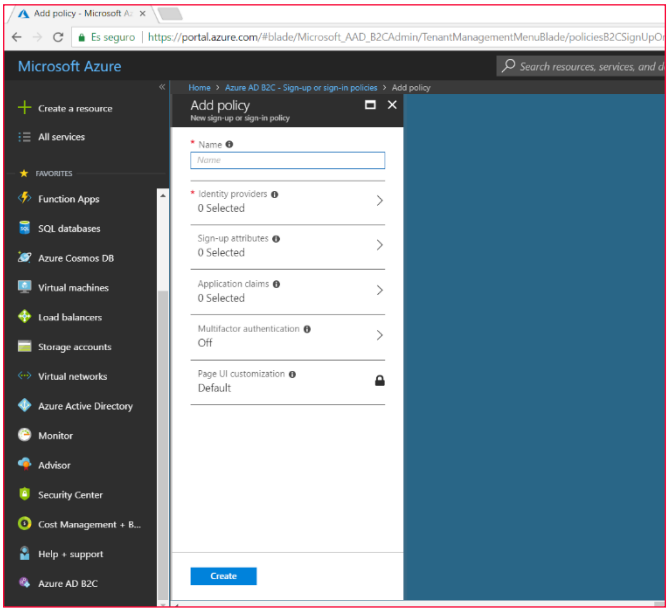


Imagen 4.- Añadiendo una política de Sign-up o Sign-in.

Los campos que rellenar son:

1.–Proveedor de identidad: las opciones que se nos mostrarán serán aquellas que habremos definido con anterioridad a nivel de servicio.

2.–Los atributos: Campos que queremos que se muestren en el formulario, las opciones que se nos mostrarán son todos los que hay por defecto más aquellos que hayamos creado de forma custom.

3.–Claims: Una vez el usuario se haya logado que atributos queremos que aparezcan en los claims.

4.–Autenticación multifacotr: Si queremos activar la autenticación multifactor o no.

5.–Page UI Custom: Si queremos utilizar una página customizada, la URL de la página cusmtomizada que será la que aparezca tanto en el login como en el registro.

- Profile editing: Nos servirá para configurar la experiencia que tendrá el usuario cuando edite su perfil.

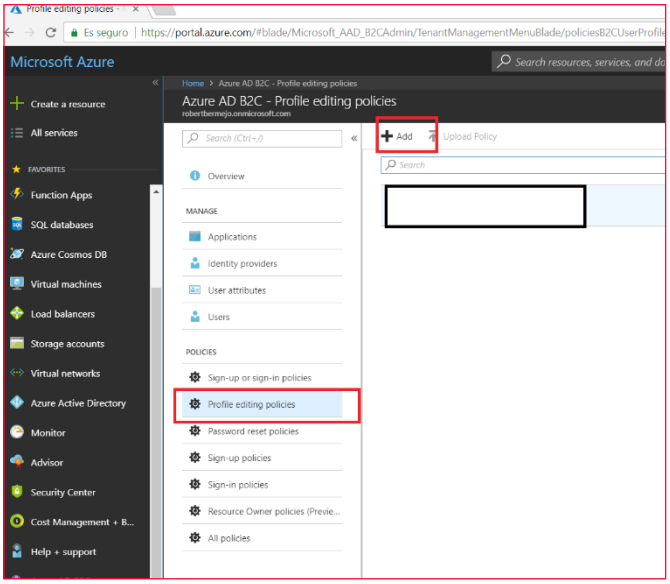


Imagen 5.- Añadiendo una Profile editing policy.

Los campos a rellenar son los mismos que en la política anterior, excepto que no existe la opción de autenticación multifactor porque no es una política de autenticación.

- Password reset: Nos servirá para configurar la experiencia que tendrá el usuario cuando quiera hacer el reset de su password.

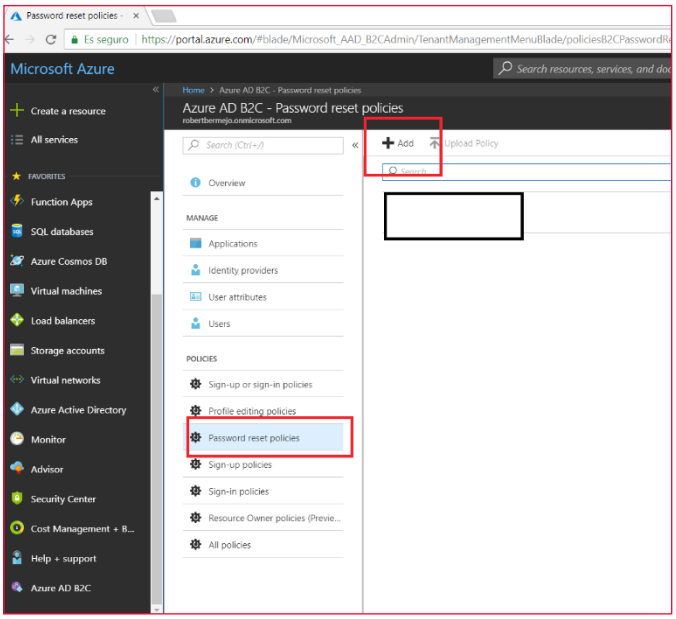


Imagen 6.- Password reset policy.

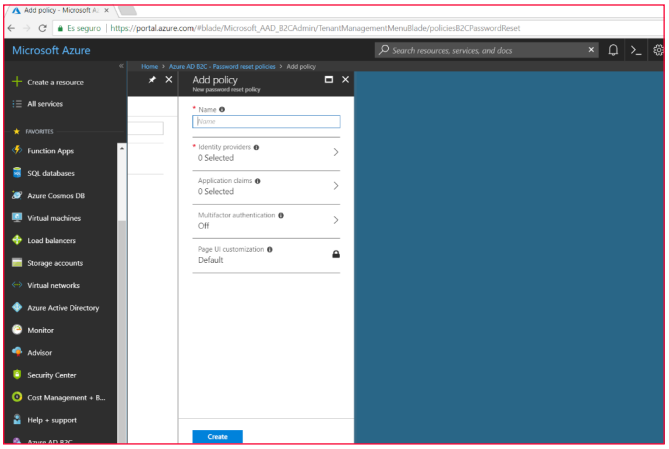


Imagen 7.- Campos a configurar.

Como se puede ver en la imagen anterior se pueden configurar todos los campos de Signin-Signup excepto la configuración de los atributos a mostrar, en este caso son por defecto.

- Sign-Up: Nos servirá para configurar la experiencia que tendrá el usuario cuando quiera registrarse en la aplicación.

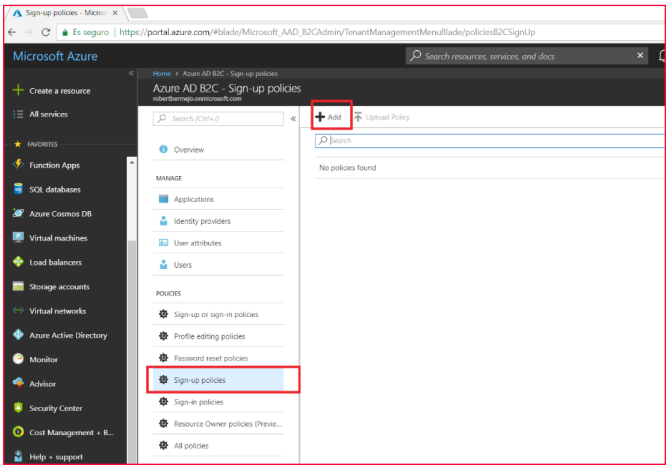


Imagen 8.- Políticas de Sign-up.

Los campos que se mostrarán para ser configurados son los mismos que en Sign-up or Sign-in

- Sign-In: Nos servirá para configurar la experiencia que tendrá el usuario cuando quiera hace el login en la aplicación.

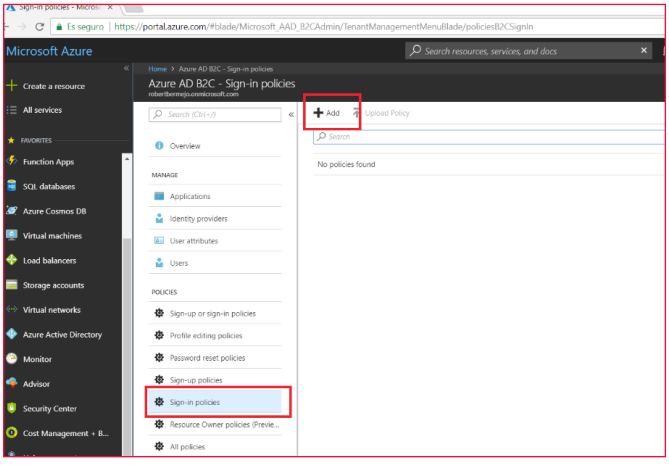


Imagen 9.- Sign-in policies.

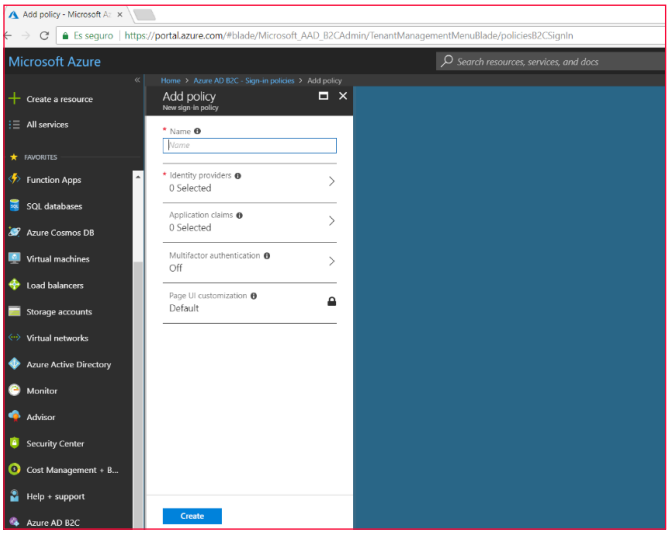


Imagen 10.- Campos para una Sign-in policy.

Como podemos observar los campos son los mismos excepto que en este caso no tiene sentido que podamos configurar los atributos que saldrán en pantalla.

- Resource Owner Policies: Esta política nos permitirá definir los claims que queremos devolver cuando se ejecutar cuando se esté ejecutando el Flow de resource owner password credentials (ROPC).

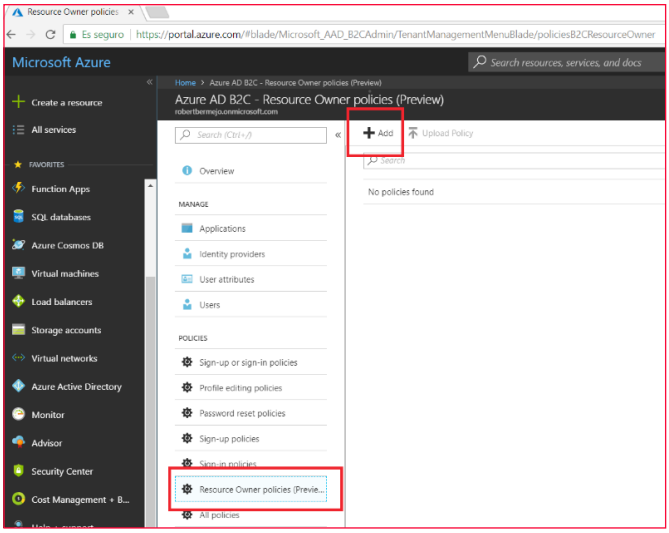


Imagen 11.- ROPC policies.

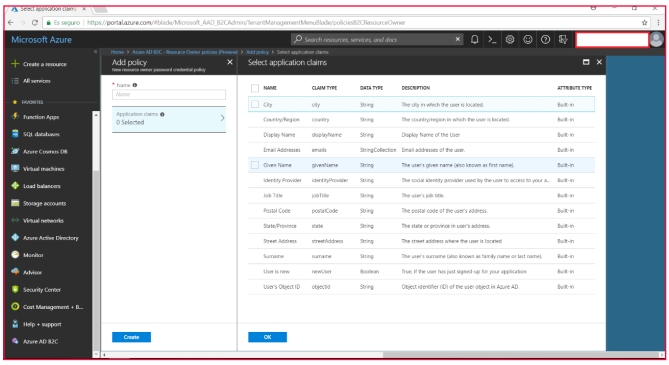


Imagen 12.- Selección de claims para la política.

A continuación, vamos a ver un ejemplo de customización Sign-Up y Sign-In. En la siguiente imagen se puede ver el resumen de la configuración:

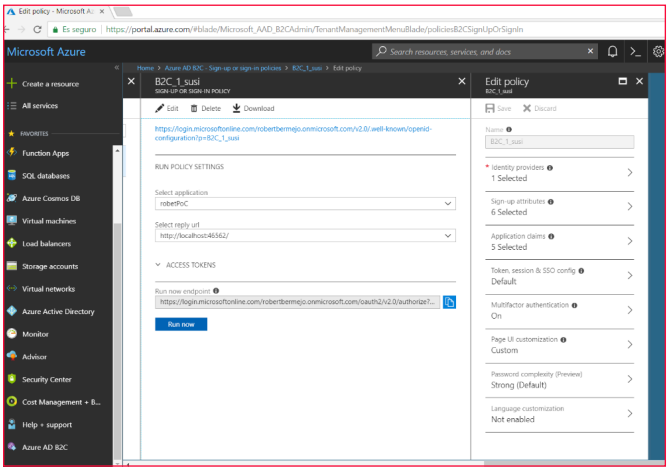


Imagen 13.- Ejemplo de customización.

Para finalizar vamos a ver cómo sería el flujo en una aplicación:

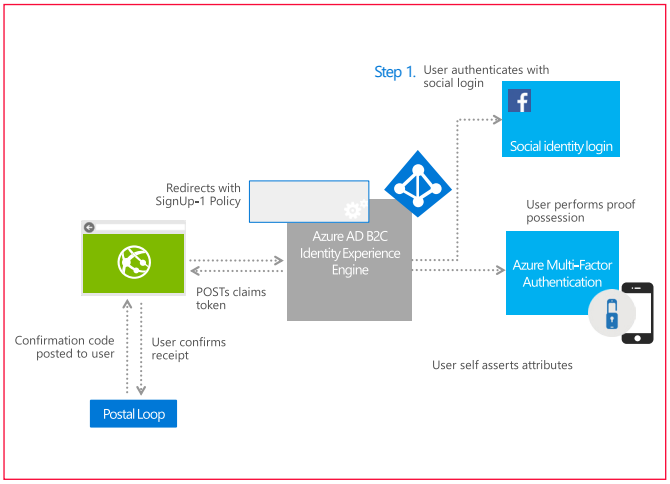


Imagen 14.- Flujo de la aplicación.

1.– El usuario realiza un Sign-up en la aplicación, y es redireccionado al proveedor de identidad configurado, en este caso Facebook. Si la autenticación tiene éxito, se cachea el token recibió y se continua con el segundo paso de autenticación como sea definido en la política.

2.– Una vez el usuario ha finalizado el ciclo de la doble autenticación, el orquestador inicia el proceso de registro donde el usuario verifica la información.

Como podemos ver Azure AD B2C es una opción muy a tener en cuenta ya que nos da una gran variedad de opciones para que nuestros clientes puedan usar nuestras aplicaciones.

Más información en: https://docs.microsoft.com/es-es/azure/active-directory-b2c/

ROBERT BERMEJO
Cloud Specialist Consultant en TOKIOTA
Microsoft Azure MVP
bermejoblasco@live.com
@robertbemejo
www.robertbermejo.com

38

La magia de crear chatbots sin abrir Visual Studio gracias a FormFlow

Los chatbots, las interfaces conversacionales, que empezaron a estar de moda hace un par de años resultan muy divertidos, pero no siempre son útiles. Requiere bastante esfuerzo diseñar una conversación teniendo en cuenta todas las opciones de respuesta que pueden plantear los usuarios. Pero: ¿es necesario controlar todas las posibles entradas para realizar una tarea de negocio? En la mayoría de las aplicaciones, la respuesta es un rotundo "no". Si así fuese, no se basarían la mayoría de apps en formularios para recoger datos con los que hacer algo.

Microsoft nos provee, dentro del Bot Framework, de un modo sencillo de modelar pequeñas funcionalidades de negocio que antes resolveríamos con un simple formulario. FormFlow es un motor que, a partir de un modelo de datos, es capaz de automatizar una conversación para recuperar toda la información necesaria para completar el modelo y que así podamos hacer lo que queramos con los datos.

Gracias a él, podemos aprovechar la omnipresencia (por estar en muchos canales en los que ya están presentes los usuarios) de un chatbot para estar más cerca de nuestro público objetivo.

"vamos a ver cómo se puede montar un chatbot completamente funcional en unos minutos"

Vamos a ver a continuación como podemos desplegar un chatbot sencillo en Azure. Veréis que el uso de FormFlow es tan sencillo que ni siquiera necesitaremos abrir Visual Studio para codificarlo. ¡Va a ser cosa de magia! En concreto, vamos a hacer un simulador del Sombrero Seleccionador de Howarts que decidía, en las historias de J. K. Rowling, a que casa iban los alumnos de la escuela de magia.

Es un ejemplo muy sencillo, pero completo, que nos servirá de base para afrontar cualquier aplicación sencilla que se base en la recopilación de un conjunto de los datos con los que después hacer algo, ya sea realizar unos cálculos (como es nuestro caso, pero podría ser el de calcular las condiciones de una hipoteca, o el precio que se aplicaría en tu seguro de coche), llamar a una API (por ejemplo para grabar la solicitud de esa hipoteca, o la petición de la llamada de un comercial de seguros), o cualquier otra cosa que se nos ocurra.

Como la mayoría del bot está basado en la definición de un modelo de datos será una tarea sencilla por muy complejo que lo queramos hacer, ya que la magia ya está codificada por Microsoft en las tripas de FormFlow.

Lo primero que tenemos que hacer, es crear en Azure un nuevo bot (Web App Bot). Para facilitarnos el trabajo, vamos a partir de un bot de ejemplo, por lo que a la hora de crear la aplicación le diremos que use la plantilla de FormFlow.

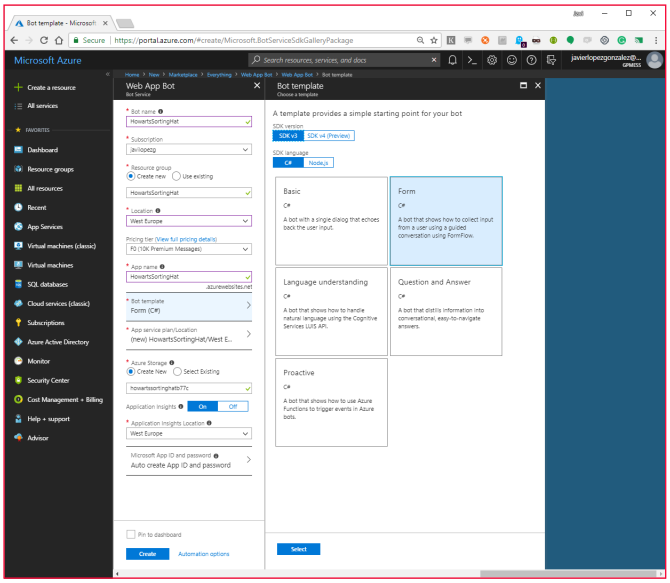


Imagen 1. - Formulario de creación de Web App Bot con detalle de selección de plantilla

Una vez desplegado, tendremos dos elementos con el mismo nombre, una Web App y el Web App Bot propiamente dicho. En el Web App Bot una de las opciones es editar el código online (dentro del menú Build), que es lo que usaremos para editar el código de nuestro bot, ya que su sencillez no requiere montar un proyecto complejo o utilizar toda la potencia de Visual Studio.

Dado que los bots que utilizan FormFlow se basan en un modelo, lo primero que tenemos que hacer será definir este. Nuestro Sombrero Seleccionador no es tan mágico como el de Howarts, el nuestro recopila distinta información sobre los alumnos para tomar una decisión de a qué casa mandarles.

Las variables del modelo pueden ser de tipos básicos (los distintos tipos de enteros, los distintos tipos de números

de coma flotantes, strings, DateTime, enumeraciones) y listas de enumeraciones. En nuestro caso, inicialmente, hemos usado sólo enumeraciones porque los valores que vamos a manejar son muy específicos de nuestro negocio.

```
        public enum Nombres {
                Godric,
                Helga,
                Rowena,
                Salazar
        };
        public enum Cualidades {
                Valentía,
                Honestidad,
                Inteligencia,
                Ambición
        };
        public enum Colores {
                Rojo,
                Amarillo,
                Azul,
                Verde
        };
        public enum ColoresDeJoyas {
                Oro,
                Negro,
                Bronce,
                Plata
        };
        public enum Animales {
                Leon,
                Tejon,
                Aguila,
                Serpiente
        };
        public enum LugaresParaVivir {
                TorreOscura,
                Bodega,
                TorreLuminosa,
                Mazmorras
        };
        public enum Titulos {
                Sir,
                Fraile,
                Dama,
                Barón
        };
        public enum Amigos {
                Ron,
                Neville,
                Hermione,
                Harry
        };
        public enum Accesorios {
                Espada,
                Copa,
                Diadema,
                Guardapelo
        }
        [Serializable]
        public class SortingTest
        {
                public Nombres? Nombre;
                public Cualidades? Cualidad;
                public Colores? Color;
                public ColoresDeJoyas? ColorDeJoyas;
                public Animales? Animal;
                public LugaresParaVivir? LugarParaVivir;
                public Titulos? Titulo;
                public Amigos? Amigo;
                public Accesorios? Accesorio;

        public static IForm<SortingTest> BuildForm()
        {
                OnCompletionAsyncDelegate<SortingTest>
evaluate = async (context, state) =>
                {
                        await context.PostAsync("OK");
                };
return new FormBuilder<SortingTest>()
                        .Message("Hola")
                        .OnCompletion(evaluate)
                        .Build();
        }
```

```
        };
```

Si ahora decidiéramos manejar nueva información como el nombre o la fecha de nacimiento, tan sólo tendríamos que añadir nuevas propiedades a nuestra clase modelo para almacenar estos datos, que posteriormente el bot se encargará de recoger y validar.

```
public string TuNombre;
public DateTime? FechaDeNacimiento;
```

Una vez que tenemos nuestra clase definida, solo la tendremos que añadir a nuestro proyecto creando un archivo online, y aprovecharemos a eliminar el modelo del ejemplo de la plantilla que no encaja para nada con nuestro mundo mágico.

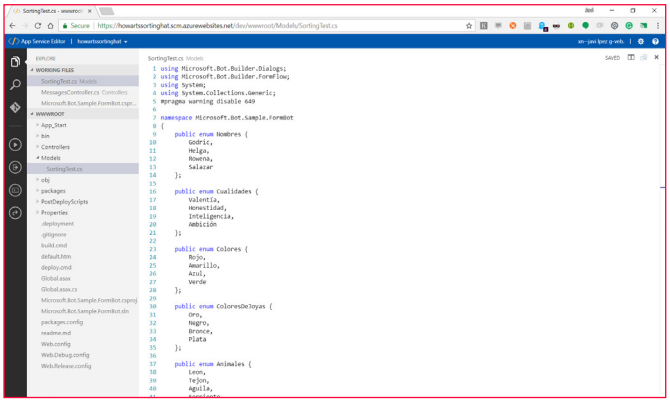


Imagen 2. - Árbol de archivos de la solución con código de modelo.

Además, tendremos que hacer unos cambios menores en el controlador para que use nuestro nuevo modelo en lugar del que hemos eliminado. Para esto editaremos el archivo MessagesController.cs y cambiaremos las referencias al modelo en el método MakeRootDialog.

```
        internal static IDialog<SortingTest> MakeRootDialog()
        {
            return Chain.From(() => FormDialog.FromForm(Sorting-
Test.BuildForm));
        }
```

A partir de aquí ya podemos compilar (build.cmd) y probar nuestro bot. En el componente del Web App Bot tenemos una opción para testear nuestro bot con un web client sin salir del portal de Azure. En cuanto le hablemos, nos responderá y podremos ver cómo nos hace preguntas para rellenar nuestro modelo.

"los bots que utilizan FormFlow se basan en un modelo, lo primero que tenemos que hacer será definir este"

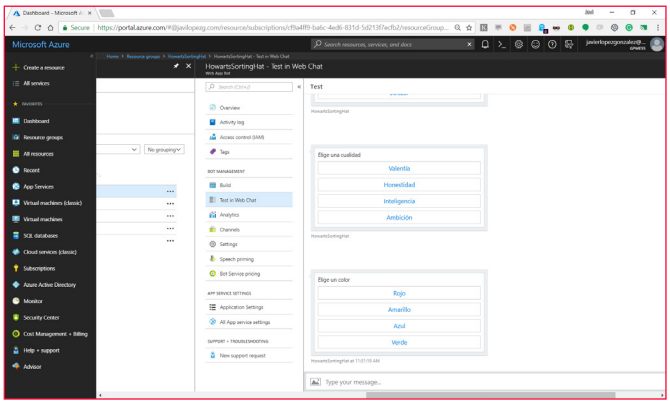


Imagen 3. - Web client para testing.

Una vez que ya estamos capturando los datos, sólo tenemos que procesarlos, para eso cambiaremos el código de BuildForm dentro de nuestro SortingTest a nuestro gusto. Si lo probamos, veremos que ya lo tenemos todo funcionando. Sin embargo, no queda muy bien que si nuestro bot está pensado para público castellano parlante le hable en inglés. FormFlow está pensado para localizarlo a distintos idiomas, pero en nuestro caso solo cambiaremos un par de detalles usando atributos sobre nuestro modelo.

Hay atributos para distintas cosas como por ejemplo para marcar qué campos son opcionales o introducir nuestras propias validaciones. Nosotros usaremos un atributo de plantilla para modificar la pregunta que se hace por cada campo.

```
[Template(TemplateUsage.EnumSelectOne, "Elige un {&} {||}",
ChoiceStyle = ChoiceStyleOptions.Auto)]
```

Hay un lenguaje completo para editar el formato de los mensajes. En nuestro caso los caracteres {&} representan el nombre del campo y los caracteres {||} las distintas opciones que tendrá el usuario. La enumeración ChoiceStyleOptions nos permite indicar cómo se mostrarán estas opciones.

> ***"pero con esto hemos podido hacer un poco de "magia de informático" en unos pocos minutos"***

Si volvemos a probar veremos que ya queda mucho más aparente, pero no queda del todo bien ya que, por ejemplo, la propiedad Cualidad es femenina y al no haber planteado una pregunta neutra se hace raro el resultado. Pasa lo mismo con las propiedades de tipo string y DateTime, para las que no hemos cambiado su plantilla. Podemos usar un atributo similar al anterior pero que se aplica sólo sobre una propiedad para cambiar estos casos concretos.

```
[Prompt("Elige una cualidad {||}")]
```

FormFlow tiene más capacidades, pero con esto hemos podido hacer un poco de "magia de informático" en unos pocos minutos, para tener algo funcional y aparente. Tan sólo nos queda elegir uno o varios canales en los que publicarlo para empezar a llegar a nuestro público objetivo justo en el canal en el que ya están trabajando a diario. Por ejemplo, si queréis saber qué opina de vosotros nuestro Sombrero Seleccionador, sólo tendréis que visitar mi web en javilopezg.com/SortingHat y charlar con él.

JAVI LÓPEZ G.
mail@javilopezg.com
@javilopezg
https://javilopezg.com

IoT Devices – Stream Analytics y PowerBI

Hace ya un tiempo Microsoft tiene un foco muy importante en la explotación de datos históricos que se vienen capturando en diferentes verticales de negocio. En esta temática hoy día contamos con múltiples dispositivos que generan información en tiempo real. Celulares, estaciones meteorológicas, sensores en plantas de producción, paneles solares, molinos, autos inteligentes, maquinaría industrial, entre otros.

Vamos por lo tanto contar en el siguiente artículo acerca de cómo implementar soluciones que nos permitan conectar dispositivos a través de servicios de Azure, volcar dicha información a BD y destinos variados, así como también redireccionarla en tiempo real para explotarla en Power BI, y permitir tomar decisiones en el momento. Podríamos también pensar en procesar la información histórica y ejecutar Machine Learning con la finalidad de entrenar un modelo que nos permita predecir comportamientos de las variables que intervienen en nuestro proceso.

No es menor el gran catálogo de opciones con las que contamos hoy en día utilizando servicios de Azure:

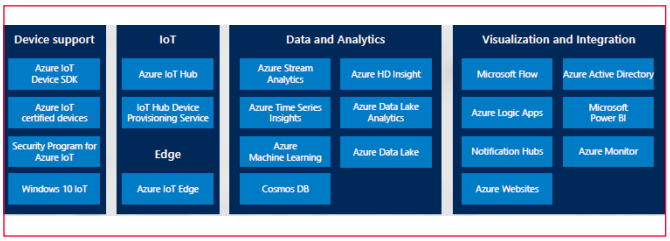


Imagen 1.- Servicios de Azure.

El siguiente ejemplo constará en una DEMO de un proceso de detección de fraude telefónico. Mediante una aplicación de consola simularemos la generación continua y automático de llamadas, pudiendo simular que un % sean fraudulentas. Nuestra aplicación de consola contará con un Config que setearemos con los datos de nuestro EndPoint:

```
<appSettings>
  <!-- Service Bus specific app setings for messaging connections -->
  <add key="EventHubName" value="gaston-eh-frauddetection-demo"/>
      <add key="Microsoft.ServiceBus.ConnectionString" value="Endpoint=sb://gaston-eh-ns-demo.servicebus.windows.net/;SharedAccessKeyName=gaston-policy-manage-demo;SharedAccessKey=n++N5Ow/jtzSXZ+G3379FJBaOwx+fmsWCKg9V7hIGHw="/>
 </appSettings>
```

"soluciones que nos permitan conectar dispositivos a través de servicios de Azure, volcar dicha información a BD y destinos variados, así como también redireccionarla en tiempo real para explotarla en Power BI"

Como podemos ver en estas configuraciones estamos definiendo como objetos a generar en Azure, lo que se denomina EventHubName, y un Connection String que se formará con objetos tales como un EndPoint, un Shared AccessKey, y un Policy definida para nuestro Device. Los pasos que debemos dar previamente en Azure, es contar con un Event Hub. Para esto, en primer lugar, se genera lo que se denomina en Azure un contenedor o un Event Hubs Namespace.

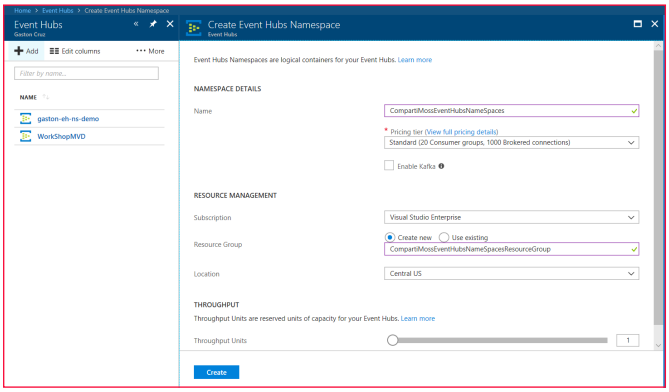


Imagen 2.- Creación del Event Hubs Namespace.

Al generar nuestro Event Hubs Namespace podremos ingresar en el mismo a crear el Event Hub:

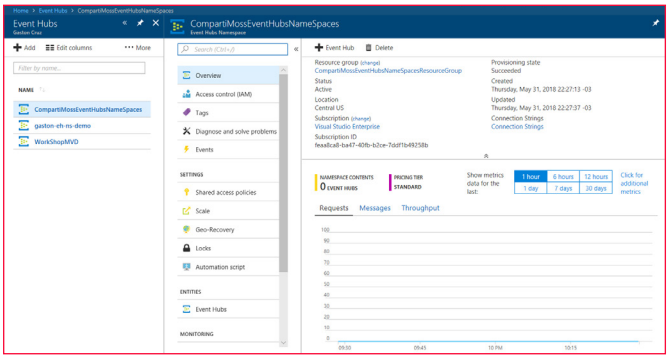


Imagen 3.- Acceso al Event Hubs Namespace creado.

Una vez creado el contenedor, lo que haremos es generar un Event Hub que permita conectar nuestro dispositivo IoT.

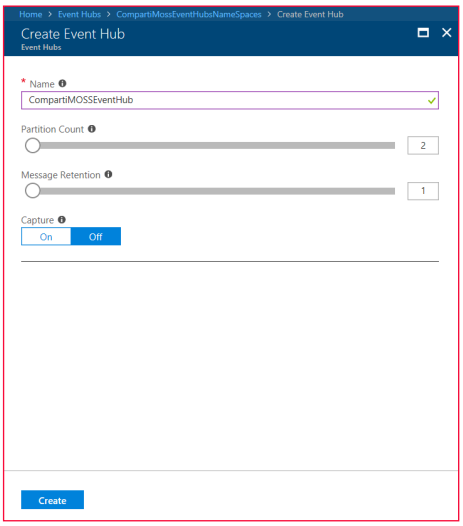


Imagen 4.- Creación del EventHub.

Una vez ya hemos creado nuestro Event Hub, debemos ingresar al mismo, y en este sentido debemos realizar la configuración de Políticas de Accesos. En este punto en particular debemos prestar atención que dicha configuración la estemos realizando sobre el Event Hub, y no sobre el NameSpace dado que lo que queremos conectar es un dispositivo a dicha salida. En la sección izquierda de esta pantalla encontraremos Shared Access Policies donde agregaremos nuestra política de acceso:

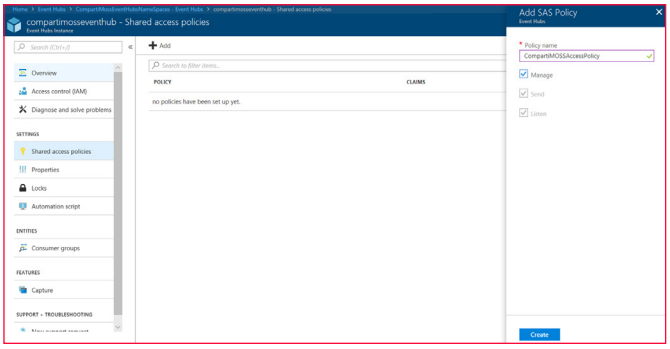


Imagen 5.- Configuración Shared Access Policies.

Como podemos ver, en la configuración de nuestro Event Hub, y en particular de la política de acceso definimos para que el hub permita enviar, y recibir mensajería. Una vez creada la política, al dar click sobre la misma veremos que encontramos definidas las Key de Accesos:

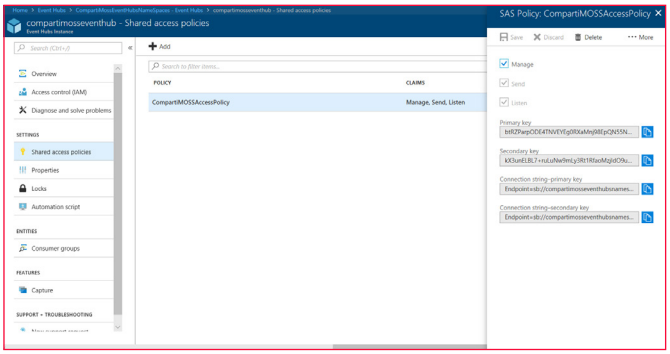


Imagen 6.- Claves de Acceso.

Las key que vemos en la captura anterior son las que configuramos en los settings de nuestra aplicación, de manera que nuestro dispositivo sepa cual es el procedimiento de acceso al Event Hub.

En estos momentos nuestra aplicación ya sabe para donde deber ir emitiendo la información. Ahora como siguiente paso generaremos un Stream Analytics Job que permitirá reunir ambos universos, los INPUT, y los OUTPUT.

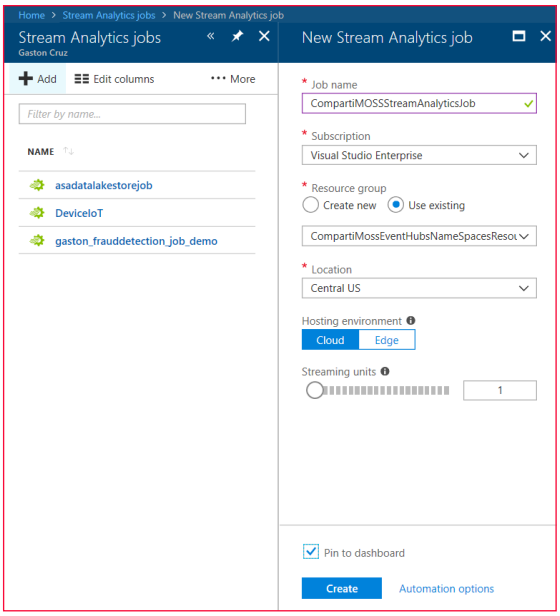


Imagen 7.- Creación de un nuevo Stream Analytics job.

Seleccionando nuestro Stream Analytics contaremos con la posibilidad en primer lugar de conectar nuestro INPUT device:

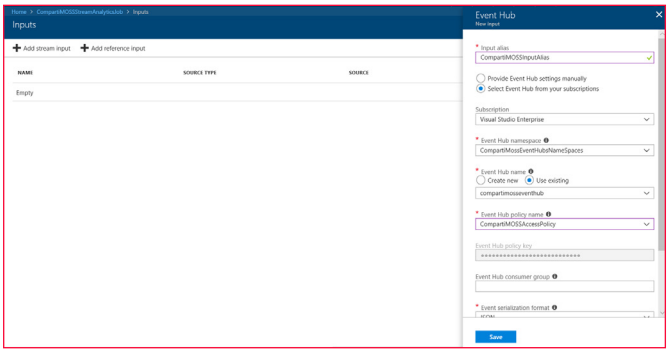


Imagen 8.- Añadiendo un nuevo Stram input.

Ahora vamos a comenzar a simular llamadas con nuestra aplicación de consola.

Ejecutaremos en una línea de comando: telcodatagen.exe 1000 .2 2

El dato 1000 es la cantidad de registros que simularemos por hora.

El .2 es el porcentaje de llamadas fraudulentas que se generarán.

El último digito 2 es la ventana de tiempo que la app está corriendo (puede ser detenida manualmente cuando lo decidamos).

La imagen a continuación nos muestra la aplicación co-

rriendo y simulando los registros de llamadas:

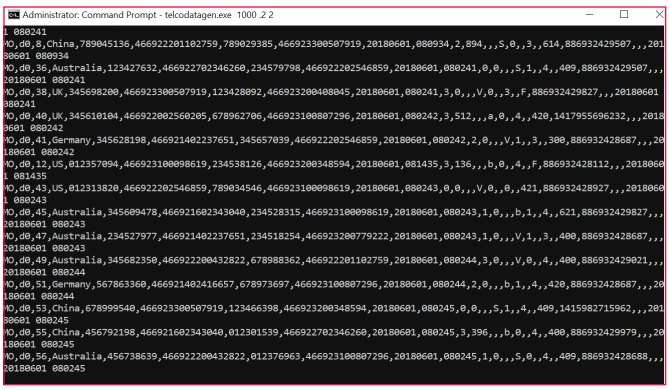


Imagen 9.- Registro de llamadas realizado.

Ahora realizaremos un testing básico que constará en saber si los datos están fluyendo a través de nuestro Event Hub y capturándose mediante el INPUT definido en nuestro Stream Analytics Job. Para esto volvemos a la pantalla principal de Stream Analytics Job, en el lateral izquierdo de la pantalla vamos a queries y seleccionamos la opción de Sample Data from Input:

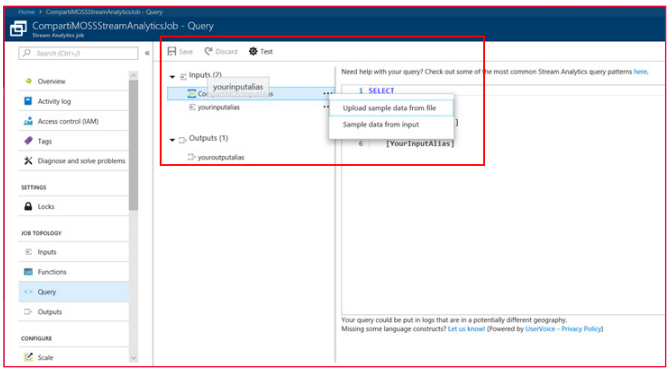


Imagen 10.- Inicio del test del Event Hub.

De esta manera, una vez ejecutamos el proceso y aguardamos algunos minutos mientras los datos se van capturando. Al finalizar tenemos la posibilidad de ejecutar una Consulta que nos permita hacer un Test de la información capturada:

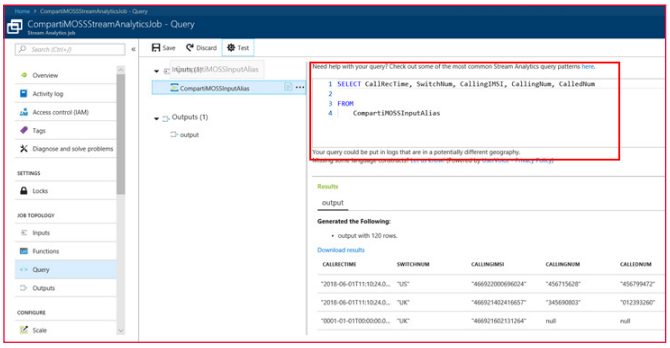


Imagen 11.- Consulta para obtener la información de test generada.

De manera de poder capturar las llamadas de tipo fraudulentas vamos a generar una Consulta que nos permita registrar dicha información:

```
/* Criterio de Fraude:
 Llamadas registradas desde un mismo punto geográfico con
una diferencia de 5 segundos entre una y otra */
```

```
SELECT System.Timestamp AS WindowEnd, COUNT(*) AS
FraudulentCalls
 INTO "CompartiMOSSStream-PowerBI"
 FROM "CompartiMOSSInputAlias" CS1 TIMESTAMP BY CallRecTime
 JOIN "CompartiMOSSInputAlias" CS2 TIMESTAMP BY CallRecTime

 /* Donde el número registrado desde donde surge el llamado
es el mismo*/
 ON CS1.CallingIMSI = CS2.CallingIMSI

 /* ...y la diferencia entre los tiempos CS1 y CS2 están entre 1 y
5 segundos */
 AND DATEDIFF(ss, CS1, CS2) BETWEEN 1 AND 5

 /* Donde la Localización es Diferente */
 WHERE CS1.SwitchNum != CS2.SwitchNum
 GROUP BY TumblingWindow(Duration(second, 1))
```

Si repetimos el proceso de generar datos de ejemplo como mencionamos en el punto anterior y volvemos a hacer un test contra la nueva consulta, obtendremos un resultado como el siguiente:

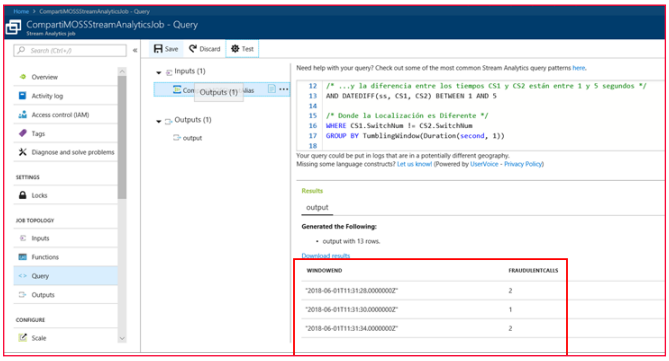


Imagen 12.- Nueva consulta de datos de test.

A continuación, vamos a definir en nuestro Stream Analytics Job, nuestro Output. Tenemos que tener presente que, al generar el output, seleccionando que la salida va a ser a Power BI debemos contar con una cuenta de Power BI con permisos que nos permita escribir sobre el servicio. De esta manera definiremos el WorkSpace, el Dataset a generar, y el DataTable que implementaremos en el servicio de Power BI:

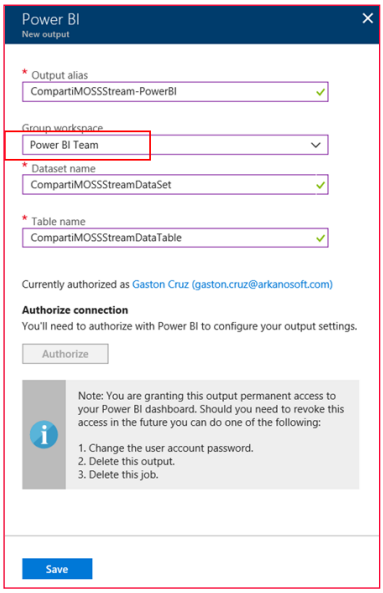


Imagen 13.- Generación del Workspace de Power BI.

Ahora le daremos Start a nuestro Stream Analytics Job (esto puede tardar unos minutos) para que comience el proceso y capturar todo el flujo.

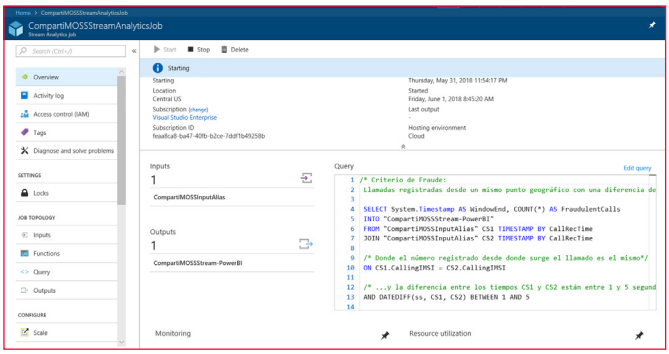


Imagen 14.- Inicio del Stream Analytics Job.

Luego de algunos minutos ya podemos ir a nuestro servicio de Power BI, loguearnos con nuestra cuenta, e ir al Workspace que seleccionamos como Output. Ahí deberíamos buscar nuestro DataSet que ya debería haber sido creado mediante nuestro Stream Analytics en forma automática:

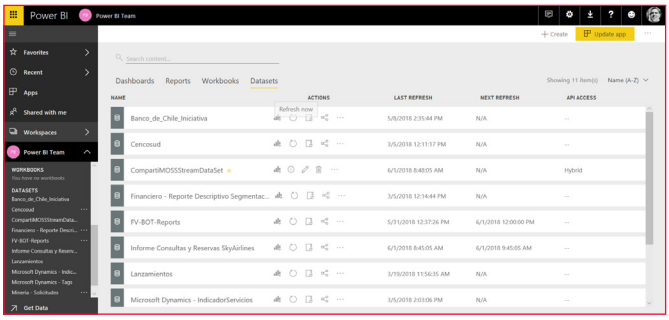


Imagen 15.- DataSet generado.

Ahora que ya tenemos generado nuestro dataset y está recibiendo datos en tiempo real desde nuestro dispositivo generaremos un Dashboard con esta información. Seleccionamos la opción de Crear y luego Dashboard en el borde superior derecho de la pantalla:

Imagen 16.- Acceso a la opción Dashboard.

Agregaremos ahora un Tile a nuestro Dashboard vacio, seleccionando la opción de Custom Streaming Data que es lo que desplegaremos como contenido:

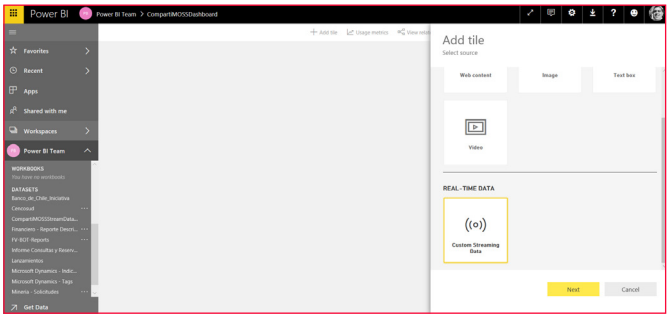


Imagen 17.- Añadiendo un Tile al Dashboard.

Al agregar como opción Custom Streaming Data, la siguiente pantalla nos permitirá seleccionar todos los DataSets que en nuestro Workspace sean del tipo Real Time DataSet. En nuestro caso solo encontraremos el definido como output del Stream Analytics Job generado desde Azure. La configuración del Tile es muy sencilla, se selecciona el formato visual, y los campos de nuestro dataset que deseamos visualizar. Como primer ejemplo definimos este Tile como CARD en el formato, y le agregamos como Valor el campo FraudulentCalls (conteo de llamadas fraudulentas). A continuación, veremos que se agrega una tarjeta que va cambiando su valor en tiempo real a medida que se van generando los datos en nuestro dispositivo.

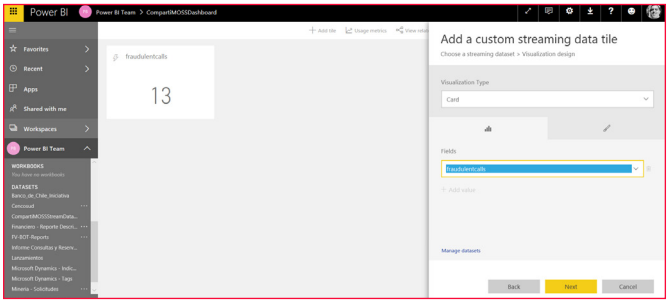


Imagen 18.- Tile configurado.

Vamos a agregar un Tile adicional con el conteo de llamados fraudulentos en una línea de tiempo. Al graficarla veremos que nuestro Dashboard queda listo y tiene un movimiento de acuerdo a los sucesos se van capturando en tiempo real:

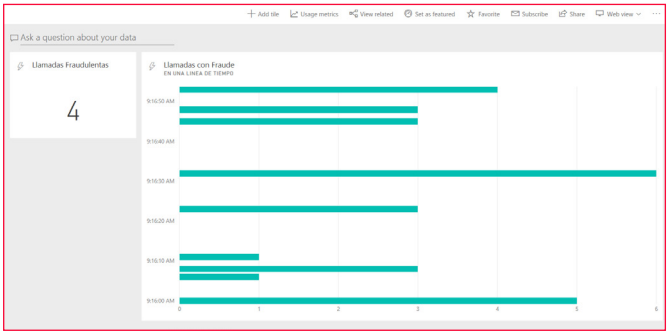


Imagen 19.- Línea de tiempo añadida.

Conclusión

Generamos en este ejemplo una DEMO que nos permite tener una visión global de una de las arquitecturas (hoy en día existen varias opciones) que podemos utilizar para proyectos de IoT. No solo utilizando varios de los servicios de Azure sino interconectándolos, y obteniendo un resultado final que nos permita tener una visualización en tiempo real.

Un dato no menor, podríamos configurar varios Outputs de nuestro Stream Analytics Jobs y de esa manera ir persistiendo la información que vamos recibiendo, en un Azure Blob Storage, Azure Data Lake, o Cosmos DB por ejemplo.

GASTON CRUZ
Data Platform MVP
@GastonFCruz

45

Nosotros

Alberto Diaz

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

Sitio Web: http://blogs.encamina.com/negocios-sharepoint/
Email: adiazcan@hotmail.com
Blogs: http://geeks.ms/blogs/adiazmartin
Twitter: @adiazcan

Fabián Imaz

Fabián Imaz, MVP de SharePoint Server trabaja en el mundo del desarrollo de software desde hace más de 10 años, teniendo la suerte de trabajar en distintas arquitecturas y tecnologías Microsoft. Pertenece a la firma Siderys, http://www.siderys.com empresa de desarrollo de Software especializada en SharePoint 2007/2010/2013 y en desarrollo de soluciones inteligentes. Desde los comienzos Fabián ha trabajado en distintitas comunidades donde organiza y promueve eventos locales para la difusión de tecnología dentro de los miembros de las mismas. Es director de la carrera SharePoint 2010 y SharePoint 2013 en Microsoft Virtual Academy, http://www.mslatam.com/latam/technet/mva2/ Home.aspx y cuenta con un sitio en CodePlex con varios desarrollos http://siderys.codeplex.com.

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

Gustavo Velez

Gustavo Velez es Ingeniero Mecánico y Electrónico; trabaja en la arquitectura, diseño e implementación de sistemas de IT basados en tecnologías de Microsoft, especialmente SharePoint, Office 365 y Azure. Propietario del sitio especializado en información sobre SharePoint en español http://www. gavd.net, autor de ocho libros sobre SharePoint y sus tecnologías y numerosos articulos y conferencias sobre el tema.

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

Juan Carlos González Martín

Ingeniero de Telecomunicaciones por la Universidad de Valladolid y Diplomado en Ciencias Empresariales por la Universidad Oberta de Catalunya (UOC). Cuenta con más de 12 años de experiencia en tecnologías y plataformas de Microsoft diversas (SQL Server, Visual Studio, .NET Framework, etc.), aunque su trabajo diario gira en torno a las plataformas SharePoint & Office 365. Juan Carlos es MVP de Office Servers & Services y co-fundador del Grupo de Usuarios de SharePoint de España (SUGES, www.suges.es), del Grupo de Usuarios de Cloud Computing de España (CLOUDES) y de la Comunidad de Office 365. Hasta la fecha, ha publicado 9 libros sobre SharePoint & Office 365, así como varios artículos en castellano y en inglés sobre ambas plataformas.

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

Coordinadores de sección

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

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

GASTÓN CRUZ
Coordinador de PowerBi
gastoncruz@gmail.com

XAVIER MORERA
Coordinador de .Net
xavier@familiamorera.com

PABLO PERALTA
Coordinador de Dynamics CRM
pablop2006@gmail.com

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

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

ESTEBAN SOLANO GRANADOS
Coordinador de Xamarin
stvansolano@outlook.com

MAXI ACCOTTO
Coordinador de SQL Server
maxi.accotto@gmail.com

¿Desea colaborar con CompartiMOSS?

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

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

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

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

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

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


Comparti
MOSS