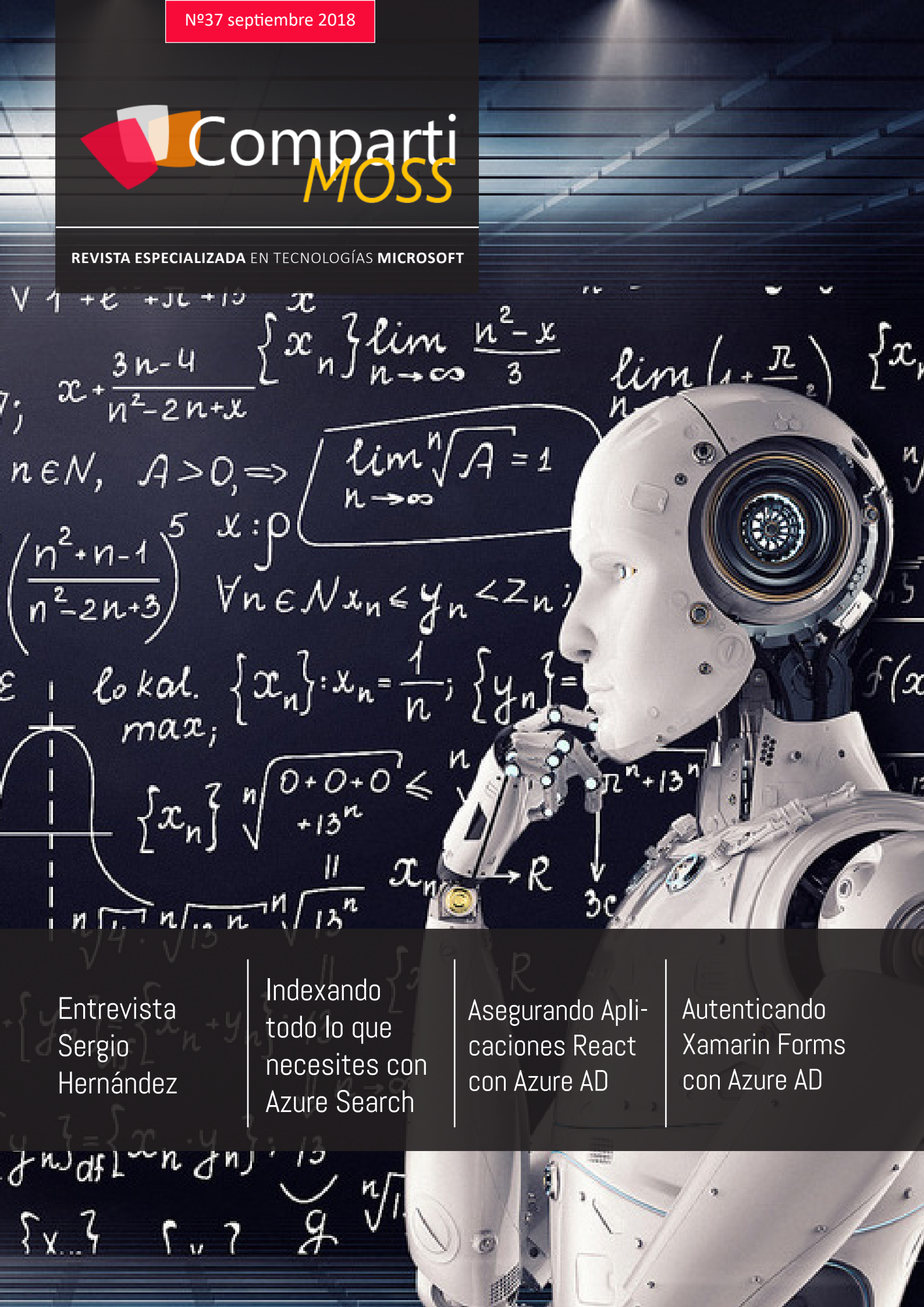
Nº37 septiembre 2018

Comparti MOSS

REVISTA ESPECIALIZADA EN TECNOLOGÍAS MICROSOFT



Entrevista Sergio Hernández

Indexando todo lo que necesites con Azure Search

Asegurando Aplicaciones React con Azure AD

Autenticando Xamarin Forms con Azure AD

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

Nos gustaría reflexionar sobre dos frases acerca de los datos y su futuro:

Brian Krzanich, CEO de Intel, dijo "*Oil changed the world in the 1900s. It drove cars, it drove the whole chemical industry," Krzanich explains. "Data, I look at it as the new oil. It's going to change most industries across the board.*"

Satya Nadella, CEO de Microsoft, en la Keynote del último Build "*We need to ask ourselves not only what computers can do, but what computers should do, Data privacy is a human right*"

La tecnología de la que hablamos en esta Revista no deja de ser un medio para procesar y analizar los datos. Nuestros artículos, directa o indirectamente, tratan de explorar medios para trabajar con esos datos y potenciar la transformación de las personas. Datos es el concepto del futuro, el nuevo petróleo con el que tenemos que trabajar con ética y responsabilidad social.

Nuestros autores son expertos en tecnología Microsoft que comparten sus conocimientos para hacer un mundo mejor, que escriben sus artículos para que podamos todos cambiar el modo en que trabajamos con los datos. Disfruten leyendo este número de septiembre y los invitamos a compartir su conocimiento en el próximo número de la revista y, por su puesto, a seguir leyéndonos.

El Equipo Editorial de CompartiMOSS

04

Explorando la programación multi-idioma con SharePoint Framework

Estimados colegas programadores SharePoint es un gusto para mí poder colaborar para esta revista nuevamente después de varios años de ausencia. En fin, vámonos directo al tema en cuestión. Estaremos dando a conocer las características multilenguaje de SharePoint Framework mediante una revisión de los elementos de código necesarios para implementar soporte multi idioma.

Si eres un programador experimentado sabrás que mediante archivos de recursos y con una programación completamente libre de textos escritos en controles y/o código es que podemos lograr que nuestras aplicaciones funcionen a manera de multi idioma.

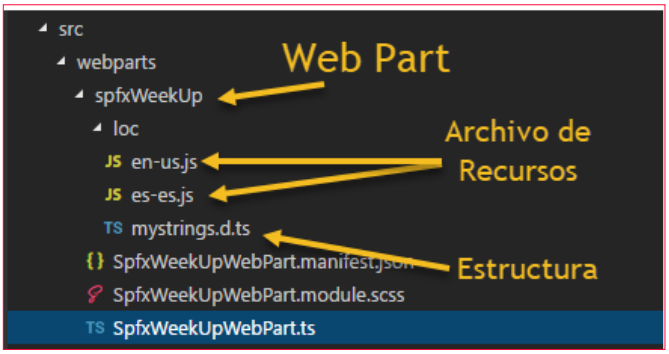


Imagen 1.- Elementos a considerar para soportar multi-idioma en nuestros WebParts y Extensiones de SharePoint Framework.

La buena noticia es que en SharePoint Framework no es la excepción, samos archivos JavaScript por cada lenguaje que queremos implementar en nuestra aplicación, cada archivo de lenguaje contendrá los textos en el idioma correspondiente. Para nuestra suerte la plomería de configuración generada por la plantilla de proyecto Yeoman mediante el famoso comando "yo @microsoft/sharepoint" ya incluye el soporte multi idioma en cada WebPart o extensión generada, solo es cuestión de que nosotros:

1.– Definimos la estructura que represente los datos que queremos manejar, claro que esta estructura de datos será utilizada por los archivos de recursos que generaremos por cada idioma y que además será importada para poder hacer referencia a los valores correspondientes desde nuestro código del WebPart. En el archivo mystrings.d.ts ya tiene la interface que podemos utilizar, solo agregamos nuestras variables de tipo string por cada valor que necesitamos manejar en multi idioma.

```
declare interface ISpfxWeekUpWebPartStrings {
 WPTitleText:string;
 WPSubtitleText:string;
 WPSubtitleSuffixText:string;
 WPFormat:string;
}
```

2.– Crear los archivos de recursos por cada idioma a implementar, en este caso en-us.js para idioma ingles y es-es.js para idioma español, pt-br.js para portugués, etc. Cada WebPart y extensión creados con el generador de Yeoman cuenta con una carpeta llamada "loc" en donde ya reside un archivo de recurso en ingles en-us.js, solo tenemos que crear archivos adicionales por cada idioma que queremos implementar utilizando como nombre la típica nomenclatura de región como es-es, pt-br, en-us, etc. En el siguiente fragmento vemos un ejemplo de dos archivos de recursos usando la estructura de datos definida en la interface mencionada anteriormente:

```
en-us.js
define([], function() {
 return {
  "WPTitleText":"Good day, today is",
  "WPSubtitleText":"week",
  "WPSubtitleSuffixText":"of the year",
  "WPFormat":"dddd MMMM D"
 }
});
es-es.js
define([], function() {
  return {
   "WPTitleText":"Buen día, hoy es",
   "WPSubtitleText":"semana",
   "WPSubtitleSuffixText":"del año",
   "WPFormat":"dddd D MMMM"
  }
 });
```

"Office 365 ya cuenta con el soporte multi idioma necesario para hacer de esta plataforma sea accesible a diversas audiencias geográficamente"

Ya dentro del código de nuestro WebPart se encuentra la

instrucción que importa la definición de nuestra estructura de datos:

```
import * as strings from 'SpfxWeekUpWebPartStrings';
```

En SharePoint Framework no es la excepción, usamos archivos JavaScript por cada lenguaje que queremos"

Y gracias a ello podemos hacer referencia programáticamente a los valores de nuestra estructura de datos que existen dentro de cada archivo de recursos. La magia sucede en tiempo de ejecución ya que el SharePoint Framework analiza el idioma con el que se está ejecutando nuestro WebPart y retorna los valores del archivo de recursos JavaScript como instancia de la interface, de esa forma hacemos referencia a los campos de nuestra estructura de datos mediante el código y obtenemos su valor en el idioma en cuestión:

```
${strings.WPTitleText}
```

Importante mencionar que podemos recurrir a clases y métodos JavaScript existentes en el contexto de ejecución que nos permiten saber el idioma con el que se está ejecutando nuestro código, por ejemplo:

```
this.context.pageContext.cultureInfo.currentCultureName.
toLowerCase()
```

Si quieres hacer pruebas en tiempo de desarrollo puedes utilizar el archivo config\write-manifests.json para forzar al Workbench que se ejecute en el idioma que necesitas probar, solamente agrega una entrada más al código json llamada "debugLocale" y define el idioma que necesitas:

```
{
  "$schema": "https://developer.microsoft.com/json-sche-
mas/spfx-build/write-manifests.schema.json",
  "debugLocale": "es-es",
  "cdnBasePath": "<!-- PATH TO CDN -->"
}
```

Necesitaras detener la ejecución de gulp serve e inicializarla de nuevo.

Conclusión

Ahora si como dicen las personas de Estados Unidos, "in a nutshell" que más o menos se refieren "en resumen", esto es lo que tienes que saber para tomar ventaja de las características de programación multi idioma en SharePoint Framework.

1.– Definir la estructura de datos.

2.– Crear archivos de recursos.

3.– Invocar la estructura en el código en tu WebPart o extensión.

"solo tenemos que crear archivos javascript por cada idioma a imlementar"

Bien colegas, con esto me despido esperando que haya sido de tu agrado esta breve explicación. Si necesitas ver el ejemplo paso a paso que mencione al iniciar este articulo te invito a que:

1.– Veas este video de Youtube donde demuestro paso a paso como construir un WebPart con soporte multi idioma - https://tinyurl.com/video-multi-idioma

2.– Que descargas de mi perfil de GitHub una copia del código de ejemplo - https://tinyurl.com/code-multi-idioma

HAARON GONZALEZ
Consultor SharePoint y Office 365
haaron.gonzalez@sharepoint.com.mx
@helpdsp

06

Usando Microsoft Graph API Schema Extensions desde SPFx

En el número anterior de la revista, vimos como podemos extender MS Graph para almacenar nuestros propios datos (http://www.compartimoss.com/revistas/numero-36/microsoft-graph-api-extensions). Partiendo de lo que aprendimos en ese artículo, vamos a ver como podemos utilizar una Schema Extension de MS Graph desde una WebPart del SharePoint Framework (SPFx).

Un posible escenario de negocio sería la necesidad de extender la información relativa a un sitio de SharePoint

Un posible escenario de negocio sería la necesidad de extender la información relativa a un sitio de SharePoint, con algunos metadatos adicionales. Por ejemplo, imaginemos que creamos un Team site de SharePoint para toda la información relativa a una oferta para un proyecto que nuestra empresa quiere conseguir. Sería bastante útil si en el sitio de SharePoint podemos almacenar información relativa a la "Bid", como por ejemplo el Budget estimado, la Unidad de negocio donde entraría el proyecto, o la fecha prevista de cierre de la bid.

Para conseguir esto, podemos crear un Schema Extension en Graph, específico para Grupos de Office 365 (recuerda que todo Team Site moderno de SharePoint tiene por debajo un grupo de Office 365), y almacenar esos datos en la extensión. Para mostrar y editar esa información haríamos uso de un webpart SPFxx.

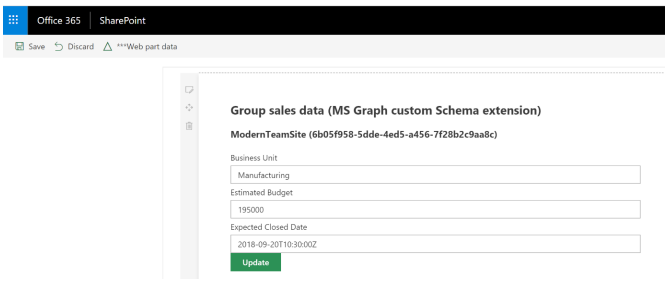


Imagen 1.- SPFx WebPart para lectura y edición de un custom Schema Extension en MS Graph.

Limitación de permisos a la hora de usar Spfx GraphHttpClient

A la hora de usar la MS Graph API desde SPFx, el Framework nos provee de un objeto GraphHttpClient, dentro del contexto del webpart.

```
context.graphHttpClient
```

Sin embargo, hay ciertas limitaciones en los permisos, por lo que solo determinadas acciones son permitidas para ese cliente. Al fin y al cabo, el GraphHttpClient obtiene un Token oAuth2 para comunicarse con Graph. Dicho token se obtiene utilizando una Aplicación registrada en el Azure Active Directory del tenant de Office 365. Dicha aplicación viene pre-registrada por Microsoft, aunque no está visible desde el portal. Sin embargo, podemos utilizar las Developer tools del navegador para cazar la petición que hace el GraphHttpClient, y descodificar el Token utilizado. La siguiente imagen muestra parte de la información disponible en un Token utilizado por GraphHttpClient


```
{
  "aud": "https://graph.microsoft.com",
  "iss": "https://sts.windows.net/[redacted]-4631-
a196-[redacted]243/",
  "iat": 1530519684,
  "nbf": 1530519684,
  "exp": 1530523584,
  "acr": "1",
  "aio": "Y2dgYAhb+0Ftjf3bRNN49vuP1qcdBQA=",
  "amr": [
    "pwd"
  ],
  "app_displayname": "Office 365 SharePoint Online",
  "appid": "00000003-0000-0ff1-ce00-000000000000",
  "appidacr": "2",
  "auth_time": 1530517819,
  "family_name": "Mañez",
  "given_name": "Luis",
  "ipaddr": "95.23.22.152",
  "name": "Luis Mañez (Inherits Cloud)",
  "oid": "51ced5b0-8f60-4d5e-8aa2-35e6d7db6606",
  "platf": "5",
  "puid": "10037FFEA08B9864",
  "scp": "Group.ReadWrite.All Reports.Read.All
User.Read.All",
  "sub": "ZlEZKYRR75fWfLg7tUHg5JbG0Ajh5AH9_rA9y5mr7JA",
  "tid": "cd8b37d9-53de-4631-a196-2d01dc814243",
  "unique_name": "luis@inheritscloud.com",
  "upn": "luis@inheritscloud.com",
  "uti": "Ivmqsz1Rr0OYll9Pu_kWAA",
  "ver": "1.0",
  "wids": [
    "62e90394-69f5-4237-9190-012177145e10"
  ]
}
```


Imagen 2.- oAuth2 Token utilizado por GraphHttpClient.

En la imagen podemos ver el nombre de la Aplicación de Azure AD junto a su ID, y si nos fijamos en el Scope, vemos que los permisos disponibles son:

- Group.ReadWrite.All
- Reports.Read.All
- User.Read.All

Básicamente, lo que podemos realizar es obtener información del usuario logado, y realizar operaciones con Grupos de Office 365, tanto de lectura como escritura. Gracias a este último permiso, podemos extender la información del Grupo utilizando una custom Schema extensión.

Nota: podemos actualizar la extensión de un grupo, pero NO podemos crear la custom Schema Extension, para ello tendremos que utilizar otra Aplicación, bien desarrollada por nosotros, o podemos hacer uso del sitio web de MS Graph Explorer (https://developer.microsoft.com/en-us/graph/graph-explorer).

En el artículo del número anterior (http://www.compartimoss.com/revistas/numero-36/microsoft-graph-api-extensions), podemos ver cómo crear la custom Schema Extension.

Para obtener la información de una Schema Extensión, podemos utilizar el siguiente código:

```
private async _getCustomExtension(): Promise<ISalesDataSchemaExtension> {
    const groupId: Guid = this.props.context.pageContext.site.group.id;

     const response: HttpClientResponse = await this.graphClient.get(
      `v1.0/groups/${groupId}/?$select=id,displayName,inheritscloud_SalesCustomData`,
      GraphHttpClient.configurations.v1);

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

    const groupSchemaExtenion: ISalesDataSchemaExtension = {
      id: responseJson.id,
      displayName: responseJson.displayName,
       expectedClosedDate: responseJson.inheritscloud_SalesCustomData &&
        responseJson.inheritscloud_SalesCustomData.expectedClosedDate,
      estimatedBudget: responseJson.inheritscloud_SalesCustomData &&
       responseJson.inheritscloud_SalesCustomData.estimatedBudget,
      businessUnit: responseJson.inheritscloud_SalesCustomData &&
       responseJson.inheritscloud_SalesCustomData.businessUnit
    };

    return groupSchemaExtenion;
  }
```

Básicamente, tenemos que hacer una petición GET al endpoint de groups, pasando el ID del grupo específico, que está disponible en la información de contexto de la WebPart, y asegurándonos de que incluimos la extensión como un campo más, usando $select

NO podemos crear la custom Schema Extension, para ello tendremos que utilizar otra Aplicación

Con la información retornada, simplemente la devolvemos usando una Interfaz custom que modela dicha información

```
export interface ISalesDataSchemaExtension {
  id: string;
  displayName: string;
  expectedClosedDate?: Date;
  estimatedBudget?: number;
  businessUnit?: string;
}
```

Para actualizar el valor de la Schema Extension del grupo, podemos usar el siguiente snippet:

```
private async _updateExtensionInGroup(): Promise<any> {
    console.log("About to update Extension with data: ", this.state.data);

   const httpClientOptions: IGraphHttpClientOptions = {
    method: "PATCH",
    body: JSON.stringify({
     "inheritscloud_SalesCustomData": {
      "businessUnit": this.state.data.businessUnit,
      "estimatedBudget": this.state.data.estimatedBudget,
       "expectedClosedDate": this.state.data.expectedCloseDDate
     }
    })
   };

    const groupId: Guid = this.props.context.pageContext.site.group.id;

     const response: HttpClientResponse = await this.graphClient.fetch(
      `v1.0/groups/${groupId}`,
      GraphHttpClient.configurations.v1,
      httpClientOptions);

   return response.status;
  }
```

En este caso, se trata de enviar una petición PATCH a la url del grupo (misma URL que para obtener la información), y enviar en el Body un string JSON que especifique el ID de la extensión, y los valores de cada propiedad.

Tenéis disponible el ejemplo concreto incluido en el repositorio del PnP de SPFx webparts:

https://github.com/SharePoint/sp-dev-fx-webparts/tree/master/samples/react-graph-schema-extensions

Además, tuve la suerte de poder hacer una demo del proyecto durante una de las Community Call del PnP, así que podéis ver la grabación en el siguiente link (idioma inglés).

https://youtu.be/44Er83s9SW8?t=1526

Nada más, de esta forma podemos incluir nuestra propia información personalizada en grupos de Office 365, y hacerlo disponible desde un WebPart de SPFx.

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

08

Indexando todo lo que necesites con Azure Search

Múltiples orígenes de datos, múltiples opciones

La intención de este artículo es la de explorar las opciones de Indexado y rastreo en Azure Search, desde orígenes de datos diversos como bases de datos SQL en Azure, Azure Cosmos DB, ficheros alojados en un blob o bien desde un proceso propio en .Net.

Aunque repasaremos conceptos básicos del servicio, para los que no estén del todo familiarizado con el mismo recomiendo echar un vistazo a artículos anteriores de la revista que detallan a más bajo nivel el uso del servicio desde el propio portal de Azure.

Un poco de repaso, ¿Qué es Azure Search?

Si pudiéramos resumir en muy pocas líneas que es y para que podemos usar este servicio englobado en el catálogo de servicios PaaS de Azure, podríamos definir las siguientes características:

- Servicio de búsqueda en la nube.
- Ofrece a los desarrolladores las APIS y herramientas necesarias para agregar una experiencia de búsqueda enriquecida y de fácil integración en las aplicaciones.
- Filtros y búsquedas de texto en función de como se configure el índice.
- Permite aplicar algoritmos de inteligencia artificial a nuestros rastreos y búsquedas mediante el uso de Cognitive services de Azure.
- Integración con datos de cualquier origen siempre que se envíe como una estructura JSON.
- Utilización de Indexadores para configurar rastreos automáticos desde orígenes de datos concretos como Azure SQL, Azure Cosmos DB o Azure Blob Storage.
- Configuración del servicio para mejorar la experiencia de usuario con funciones de autocompletar, sugerencias de búsqueda, ofrecimiento de sinónimos, ordenaciones o navegación por facets por categorías para el filtrado dirigido por intervalos.

En resumen, es un servicio PaaS de Azure, que podemos integrar en nuestras aplicaciones vía bien API Rest o SDK en .Net, el cual recibe documentos JSON con la información a Indexar y que permite dar forma a un buscador con el comportamiento que lleguemos a necesitar.

Empezando a trabajar con el servicio: Indexando los premios OSCARS

Con los siguientes ejemplos vamos a intentar ver las diferentes opciones de indexación que nos aporta el servicio, para preparar nuestros datos de cara a montar un buscador en nuestras aplicaciones. Para ello he dejado una sencilla solución de Visual Studio en GitHub, que podéis descargar y trabajar con ella.

La solución va a trabajar con dos ficheros que contienen la información histórica de los premios Oscars:

- Data_oscars.csv
- Data_oscars.json

Por ver varios con los ejemplos, el primer fichero está en formato CSV para poder indexarlo desde un servicio de Cosmos DB en Azure Search con un indexador automático, y el segundo está en formato JSON para poder crear y subir un índice vía SDK de forma sencilla con un proyecto 100% en Visual Studio.

Creando y cargando nuestro primer Índice vía SDK de .Net en Azure

La primera opción y la más sencilla para empezar a trabajar con el servicio, es construir un pequeño proceso de Indexado con .Net Core y el SDK de Azure, con el que podremos subir nuestros datos vía JSON para posteriormente indexarlos en el servicio.

> ***La intención de este artículo es la de explorar las opciones de Indexado y rastreo en Azure Search, desde orígenes de datos diversos***

Vamos a crear un proceso que realice las siguientes tareas:

- Serializar toda la información de nuestro fichero JSON de datos en una clase AzureSearchModel.cs

- Crear un índice en un servicio de Azure Search (previamente creado).
- Subir un documento JSON con la información serializada.

Una vez concluido el proceso podremos ir al portal de Azure y realizar una pequeña consulta, para ver que la información esta correctamente indexada. Del proyecto descargado debemos fijarnos en la clase AzureSearchService del proyecto Services que nos va a permitir interactuar con el servicio de búsqueda de Azure, y la clase AzureSearchModel del proyecto Models que define la naturaleza de nuestro índice y como vamos a poder consultarlo.

AzureSearchService

Contiene dos métodos con los que crearemos el índice y subiremos el modelo generado desde nuestro fichero JSON de datos.

- Create index: Nos permite generar un nuevo index en el servicio, en el cual subiremos documentos con una estructura JSON y sobre los cuales haremos las consultas.

```
public async Task<Index> CreateIndexAsync<T>(string indexName, bool overwriteIfExists, List<Suggester> suggesters) where T : class
```

- Upload Document: Este método permite subir los modelos obtenidos desde el fichero JSON con los premios Oscars desde el origen de la academia.

```
public async Task<KeyValuePair<bool, IndexBatch<T>>> UploadDocuments<T>(string indexName, T[] array) where T : AzureSearchModel
```

AzureSearchModel

```
public class AzureSearchModel
  {
    static int NextId = 0;
    [IsRetrievable(true), IsSearchable, IsSortable]
    public string Category { get; set; }

    [System.ComponentModel.DataAnnotations.Key]
    public string Id { get; private set; }


    [IsRetrievable(true), IsSearchable, IsSortable]
    public string Entity { get; set; }

    [IsRetrievable(true)]
    public bool Winner { get; set; }

    [IsRetrievable(true), IsSearchable, IsSortable]
    public string Year { get; set; }

    public AzureSearchModel()
    {
      this.Id = NextId++.ToString();
    }
  }
```

Como se ve en la clase, le podemos especificar al servicio de búsqueda que campos de nuestro modelo son filtrables, o buscables. Para implementar este proyecto de forma correcta necesitaremos hacer uso del paquete nuget:

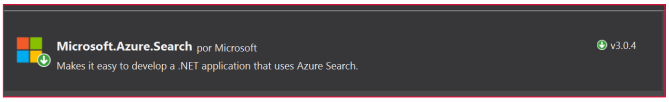


Imagen 1.- Paquete Microsft.Azure.Search-

Para poder de forma correcta el proceso y subir nuestro juego de datos al servicio de búsqueda, debemos configurar de forma correcta nuestro proyecto de consola, y para ello debemos editar los datos de conexión al servicio de búsqueda, en el Program.cs del proyecto Compartimoss. Indexation.Console

```
static void Main(string[] args)
  {
    string searchServiceName = "*****";
    string adminApiKey = "*****";

  }
```

Los datos necesarios los podremos obtener desde el portal de Azure, accediendo a nuestro servicio de Búsqueda en el apartado de Keys.

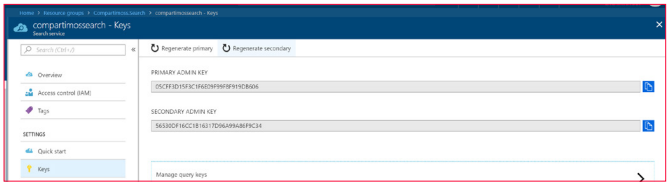


Imagen 2.- Claves de acceso al servicio.

Una vez configurada la consola, podemos lanzar una ejecución desde local y comprobar que termina con éxito la subida, accediendo al portal de Azure y realizando una simple consulta, por ejemplo, por Tom Hanks.

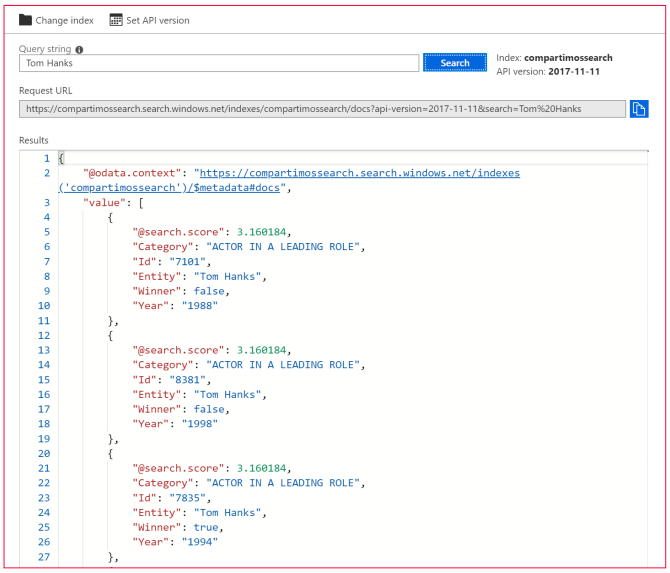


Imagen 3.- Prueba del índice.

Por terminar este apartado, aclarar el funcionamiento del proceso. Dado que utilizamos la clase IndexBatch en el método UploadDocuments y este tiene un límite de 1000

elementos por subida, vamos a subir los documentos agrupando por año del evento, tal y como vemos en el siguiente código de nuestra consola.

```
class Program
{
    0 referencias
    static void Main(string[] args)
    {
        string searchServiceName = "compartimossearch";
        string adminApiKey = "05CFF3D15F3C1F6E09F99F8F919DB606";
        //Get data
        string text = System.IO.File.ReadAllText(@"../../../Data/dataOscars.json");
        List<AzureSearchModel> data = JsonConvert.DeserializeObject<List<AzureSearchModel>>(text);


        //Create index
        var azureSearch = new AzureSearchService(searchServiceName, adminApiKey);
        var indexCreate = azureSearch.CreateIndexAsync<AzureSearchModel>("compartimossearch", false, null).Result;

        // Group by year - One document by year

        var groupByYear = data.GroupBy(e => e.Year).ToList();

        foreach(var element in groupByYear)
        {
            var currentData = element.ToList<AzureSearchModel>();
            var uploadDocument = azureSearch.UploadDocuments<AzureSearchModel>("compartimossearch", currentData.ToArray()).Result;
        }
    }
}
```

Imagen 4.- Consola en .Net Core para crear y subir el índice.

Así podremos subir todos nuestros datos a nuestro índice de pruebas.

Imagen 5.- Resumen del índice, documentos y peso.

Indexadores, rastreando e indexando desde un origen de datos

Hemos visto como cargar nuestro índice desde código con el SDK de .Net Core, esto nos va a ser muy útil para poder hacer procesos que requieran cierta lógica de transformación del dato, o que tengamos que integrar sistemas que poco tengan que ver con Azure, por ejemplo. Pero si nuestro dato se encuentra en sistemas más conocidos de bases de datos como Azure SQL o Azure DocumentDB, vamos a poder hacer uso de los llamados Indexadores.

El indexador es un rastreador (crawler) que extrae datos y metadatos útiles para nuestras búsquedas, y se puede usar bajo petición de rastreo o por el contrario dejar una programación del mismos, para que se ejecute de forma desatendida.

Se puede utilizar el indexador como único medo de ingesta de un índice desde un origen, o bien solo escoger unos pocos campos del índice.

Actualmente con estos indexadores podemos obtener datos desde:

- SQL de Azure.
- Azure Cosmos DB.
- Azure Table Storage.
- Azure Blob Storage.

Para crear un indexador, como veremos en el siguiente punto podremos hacer uso del portal de Azure, del API Rest de Azure Search o del SDK de .Net

Conectando un CosmosDB con Azure Search vía Portal de Azure

Vamos a ver en este ejemplo como subir nuestros datos sobre los Premios Oscars a un CosmosDB y posteriormente mediante un indexador de Azure Search, indexar esta información como hemos conseguido en el primer ejemplo.

Antes de empezar necesitamos tener creado:

- Un servicio de Cosmos DB del tipo SQL API (la única que tiene soporte en producción ahora mismo).
- Subir la información de nuestro Cosmos DB desde el CSV de prueba o bien vía SDK de .Net subir el JSON vía la herramienta de migración de datos.
- Crear un nuevo índice sobre el servicio de Search que utilizamos en el primer ejemplo.

Importar el CSV a CosmosDB

En este caso vamos a optar por utilizar la herramienta de importación de datos que nos podemos descargar desde el portal de Azure, y que nos va a permitir subir nuestro documento CSV a una colección de Cosmos DB que llamaremos OscarAdwards.

Es una herramienta bastante intuitiva, deberemos configurar la subida para que no tengamos error, de la siguiente forma:

- Datos de origen: Seleccionamos el fichero CSV que descargamos con la solución.

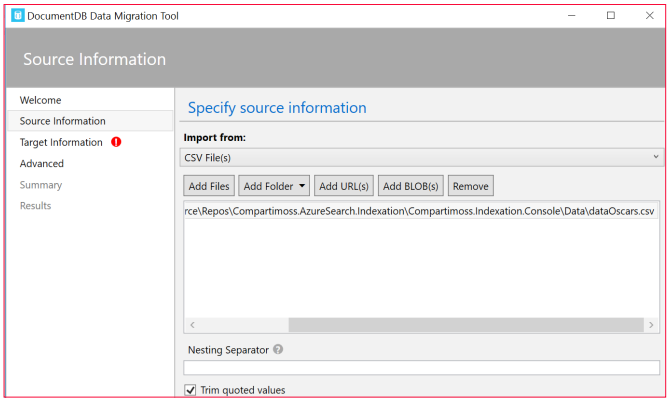


Imagen 6.- Datos de origen.

- Datos de destino: Debemos rellenar tanto la cadena de conexión que podemos obtener desde el portal de Azure con el servicio de Cosmos DB en el apartado Keys, como el nombre de la colección que en este caso decidimos llamarla compartimossSearch.

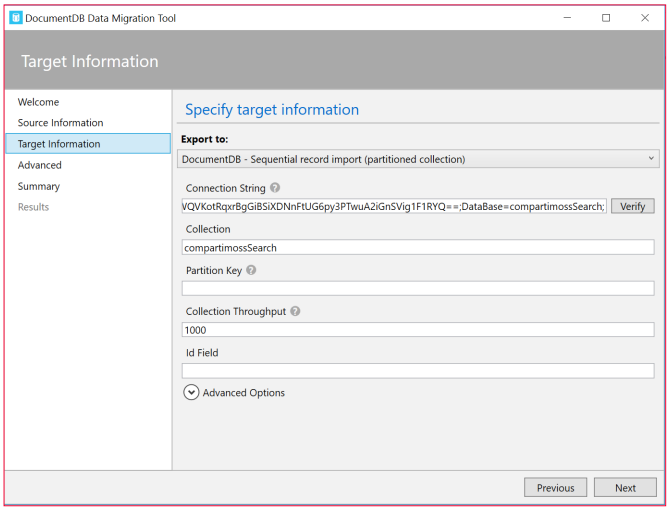


Imagen 7.- Datos de nuestro Cosmos DB.

Como observación, si copiamos la cadena de conexión desde las claves del portal decir que este no añade el nombre de la base de datos y es necesario que tenga una estructura similar a la siguiente:

```
AccountEndpoint=<Endpoint CosmosDB>;AccountKey=<Clave primaria>;Database=<Nombre database>;
```

Al terminar el proceso de subida, si accedemos a la colección de cosmos DB y lanzamos una query podremos ver que nuestros datos se han subido de forma correcta. Vamos a obtener todos los registros de Tom Hanks, tal y como se ve en la siguiente imagen.

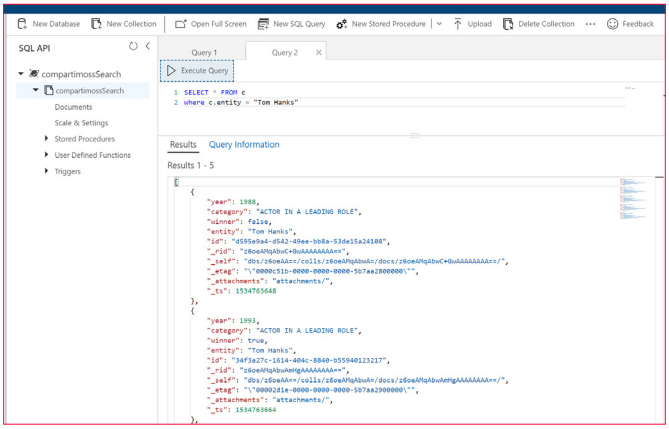


Imagen 8.- Query de prueba sobre Cosmos DB.

Crear el indexador con Cosmos DB como Origen

Una vez se han cargado los datos desde el CSV de forma correcta, vamos a proceder a configurar el indexador, de forma que los datos se repliquen de forma correcta en un nuevo índice del servicio de búsqueda. Para crear el indexador, necesitamos seleccionar dentro del servicio de Cosmos DB la opción "Add Azure Search" dentro del panel de opciones del servicio.

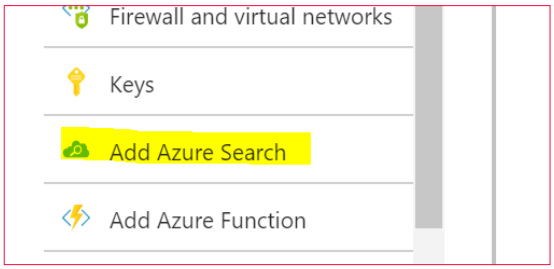


Imagen 9.- Añadir Azure Search a Cosmos DB.

En esta nueva pantalla debemos seleccionar primero el servicio de búsqueda que hemos utilizado hasta ahora.

aplicar algoritmos de inteligencia artificial a nuestros rastreos y búsquedas mediante el uso de Cognitive services de Azure

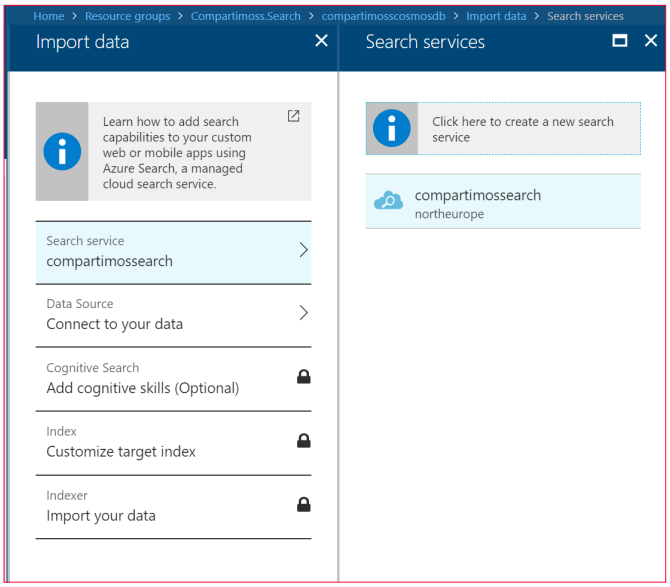


Imagen 10.- Configurando el indexer paso 1.

El segundo paso seleccionamos la instancia y la colección de Cosmos DB en la cual hemos importado los datos desde el CSV de datos.

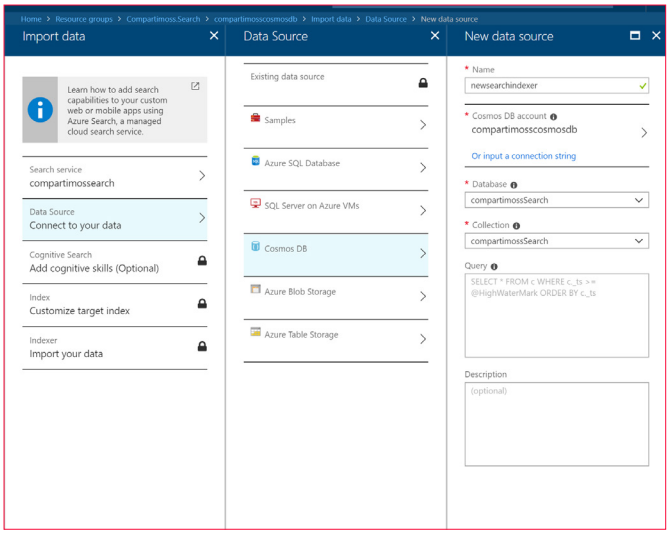


Imagen 11.- Configurando el index paso 2.

Podríamos insertar alguna Query si queremos por ejemplo que no suban todos los datos Indexados, por ejemplo, filtrar por un año, o por un actor en concreto, pero en nuestro caso por dejarlo igual que el primer ejemplo no vamos a filtrar nada. No vamos a incluir servicios cognitivos en el indexador, pero recordar que ahora mismo solo están disponibles en las regiones de South Central y West Europe, por tenerlo en cuenta.

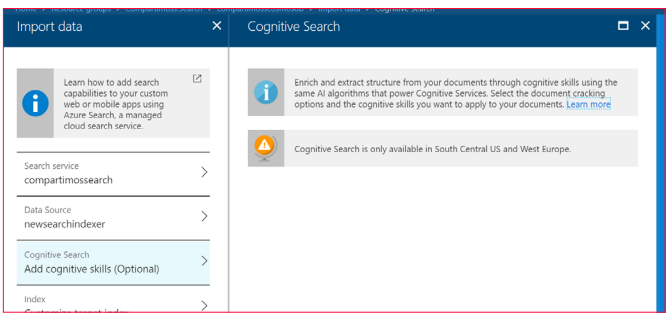


Imagen 12.- Configurando el index paso 3.

Por último, debemos configurar el índice nuevo que vamos a crear con los datos recuperados desde Cosmos DB, y para que sea idéntico al de nuestro proceso batch lo dejaremos como en la siguiente imagen.

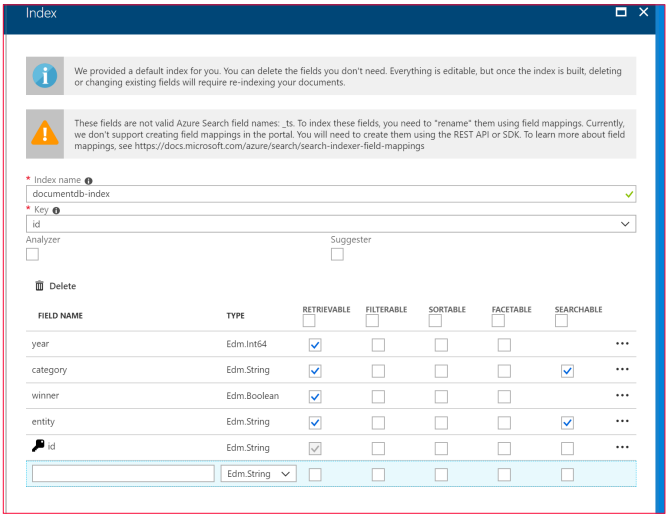


Imagen 13.- Configurando el index paso 4.

El último paso y no el menos importante, es configurar el proceso de rastreo. Como hemos nombrado con anterioridad lo podemos lanzar bajo demanda o bien programar una ejecución, en este caso lo dejaremos en modo manual.

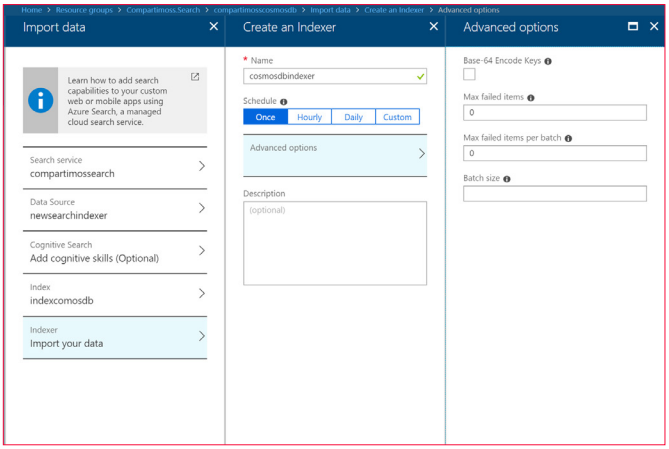


Imagen 14.- Configurando el index paso 5.

Además, podemos configurar el número de fallos que permitimos en el proceso de indexado, vamos a pensar que no falla y que no permitimos ninguno. Una vez configuremos el indexado se va a lanzar una primera ejecución, para ver si ha funcionado o no, deberemos acceder al propio servicio de búsqueda y en la pestaña Overview encontraremos los Indexadores que tenemos configurados.

Si hemos seguido la configuración de la pantalla anterior, deberéis tener una ejecución que ha fallado ya que ha excedido el número de documentos por batch, que si recordamos tiene un máximo de 1000 elementos por lote.

si accedemos a la colección de cosmos DB y lanzamos una query podremos ver que nuestros datos se han subido

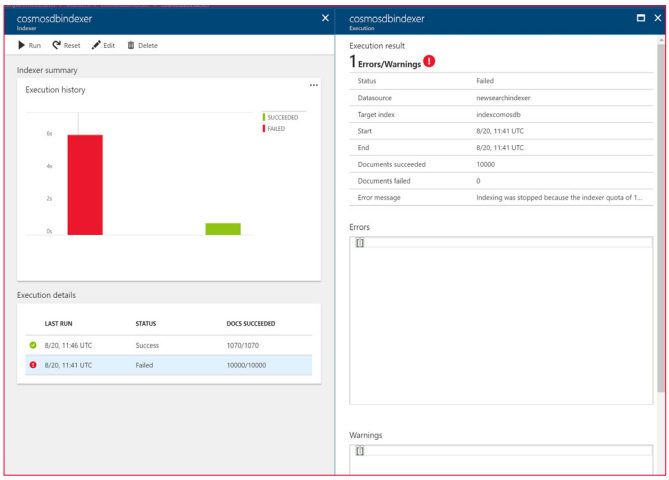


Imagen 15.- Fallo por exceso de elementos por lote.

Para resolver este problema debemos volver a editar el indexador y dejar el tamaño máximo del batch en 1000 elementos. Si volvemos a lanzar el proceso podremos ver que ahora si es correcto y ha dejado en el índice todos los documentos.

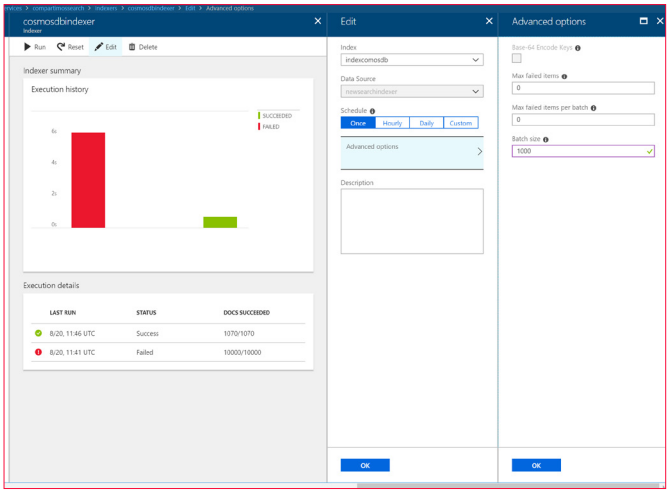


Imagen 16.- Ejecución correcta.

Si analizamos el número de documentos cargados tanto desde el CSV como desde el .JSON con el proceso de consola vemos que el resultado es totalmente el mismo.

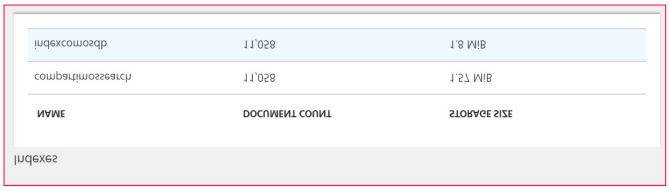


Imagen 17.- Comparando índices.

Es cierto que el peso del índice del proceso automático es ligeramente superior, ya que por ejemplo el campo ID usa un hexadecimal el proceso para rellenarlo y nosotros en el batch hemos optado por un entero, pero el número de documentos son los mismos, y por tanto no hemos perdido datos.

Para verificarlo vamos hacer la misma consulta sobre Tom Hanks y vemos que obtenemos el mismo resultado.

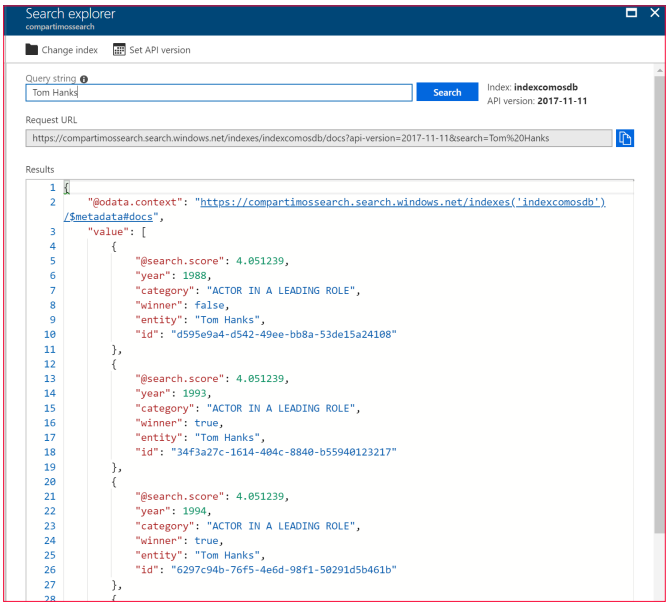


Imagen 18.- Validando el indexer.

Conclusiones de nuestros rastreos

Siempre es bueno empezar por un rastreo o carga manual para conocer el servicio, es más en casos puntuales lo tendremos que hacer así ya que se cargarán desde orígenes de datos poco accesibles o amigables. ¿Quién no sufre de vez en cuando algún que otro fichero desde un FTP y que es capital para la organización?

Pero por otro lado esto nos deja un mundo de posibilidades para por un lado tener una programación de nuestros datos y una actualización de los mismos de forma desatendida, que permita hacer un incremental y un ALM correcto no solo de nuestros desarrollos sino también del dato que exponemos.

Además, poder tener en medio un Azure SQL o un Azure Cosmos DB y usarlo como backup de nuestros maestros de datos, nos permite entre otras cosas tener servicios de búsqueda en alta disponibilidad y en regiones diferentes, consumiendo una misma o distintas bases de datos. Este no es un modelo para nada novedoso en tecnología, pero del que si carecíamos no hace mucho para este tipo de servicios en Azure.

En mi opinión con los indexadores y los rastreadores que nos aporta Azure Search vamos a poder importar miles de datos de forma segura, controlada y con un rendimiento altísimo, y al otro lado de la balanza el esfuerzo para conectar nuestros maestros a nuestros índices es mínimo si lo comparamos con hacer procesos de transformación en .Net.

Como siempre digo programar es necesario, y casi obligatorio, pero hagámoslo donde aportemos valor, y que el código que tiremos sea para mejorar si cabe los servicios que ya nos aporta el portal de Azure.

SERGIO HERNÁNDEZ MANCEBO
Principal Team Leader en ENCAMINA
Microsoft Azure MVP
@shmancebo

14

Azure Management Groups

¿A quién no le ha pasado? Un cliente pide ayuda con el gobierno de Azure, tu vas y le cuentas RBAC, las directivas y el cumplimiento, además de buenas prácticas. Al cliente le va gustando, y como tiene un contrato EA (Enterprise Agreement) pregunta que cómo puede aplicar todo eso a varias suscripciones a la vez. No puedes, tienes que repetirlo todo en cada suscripción. Gesto de decepción. Pasemos a otro tema.

Pues eso acaba de cambiar radicalmente con los Azure Management Groups, o Grupos de Administración de Azure

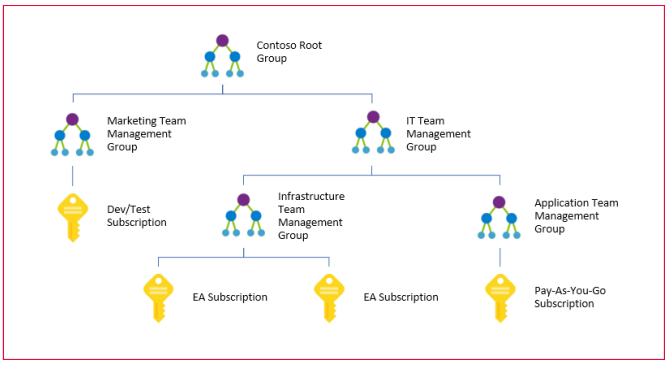


Imagen 1.- Jerarquía de los grupos de gestión.

¿Qué es?

Los Grupos de Administración de Azure permiten agrupar suscripciones para aplicar directivas o roles de RBAC a varias suscripciones a la vez. Todas las suscripciones (y sus recursos) dentro de un Grupo de Administración heredan automáticamente los controles aplicados al Grupo de Administración. Además de suscripciones, podemos organizar los Grupos de Administración de manera jerárquica.

los Management Groups para facilitar el gobierno de múltiples suscripciones de Azure

Utilizando la jerarquía de grupos de la Imagen 1 podemos, por ejemplo, aplicar una directiva que permita la creación de máquinas virtuales sólo en West Europe para el grupo "Infrastructure Team Management Group". Esta directiva se heredará a las dos suscripciones EA y aplicará a todas las máquinas virtuales de esas suscripciones. Al ser una directiva heredada, ningún administrador de las suscripciones podrá alterar estas directivas, a menos que tenga el rol RBAC adecuado en el Grupo de Administración.

Los Grupos de Administración de Azure también son útiles para asignar permisos en varias suscripciones a la vez. Si damos a un usuario un rol RBAC sobre un determinado Grupo de Administración, todas las suscripciones y recursos bajo ese Grupo de Administración heredarán los permisos.

Grupo Raíz

Cada directorio (tenant) de Azure AD tiene un Grupo de Gestión Raíz, del cual "cuelgan" todos los demás grupos o suscripciones. El Grupo Raíz permite aplicar roles RBAC y directivas de forma global a todo el directorio.

Para acceder al Grupo Raíz, el administrador global debe elevar sus propios permisos, yendo a las propiedades de Azure AD y seleccionando la opción de gestionar suscripciones y grupos de administración. Al hacerlo, está elevando permisos a sí mismo, y no a todos los administradores globales. Se recomienda que esta elevación de privilegios sobre el Grupo Raíz sea temporal y sólo por necesidad.

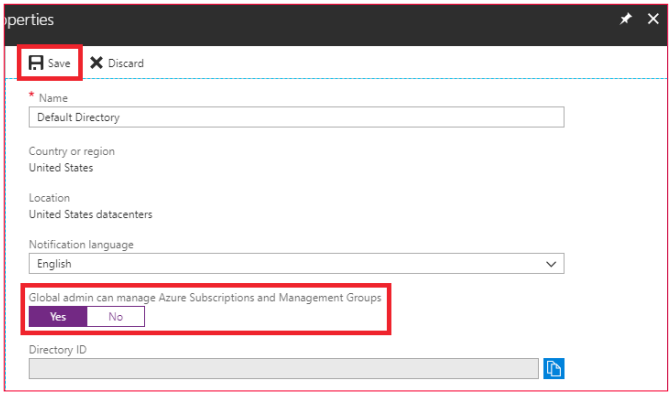


Imagen 2.- Habilitar el acceso al grupo raíz.

El Grupo Raíz no se puede editar, borrar, ni mover.

Crear Grupos de Administración

La creación del primer Grupo de Administración puede llevar varios minutos, ya que al activar esta funcionalidad se disparan varios procesos, por ejemplo, la creación del Grupo Raíz, y la asignación de todas las suscripciones existentes en el tenant como herederas del Grupo Raíz.

Para crear un nuevo Grupo de Administración en el portal

de Azure, buscamos en Todos los servicios -> Grupos de Administración y damos a "Agregar grupo de administración"

El ID del grupo es un identificador único que se utilizará para lanzar comandos a este Grupo de Administración.

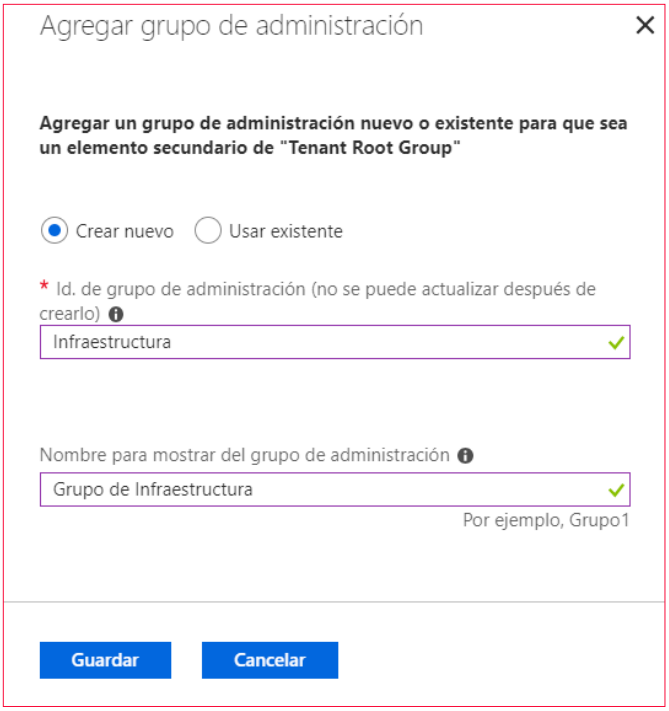


Imagen 3.- Crear nuevo grupo.

La opción de crear un nuevo grupo la encontramos tanto a nivel de Grupo Raíz, como en cualquier otro Grupo de Administración, para poder ubicar el nuevo grupo en el lugar deseado dentro de la jerarquía. Ingresando a los detalles del grupo se nos presentan más opciones, por ejemplo agregar suscripciones al grupo, renombrar o mover el grupo a otro nodo de la jerarquía (el grupo raíz no se puede renombrar ni mover).

como en cualquier otro Grupo de Administración, para poder ubicar el nuevo grupo en el lugar deseado

Para crear un grupo con Powershell:

```
New-AzureRmManagementGroup -GroupName Madrid -DisplayName "Infra Madrid" -ParentId Infraestructura
```

Con este comando hemos creado un nuevo grupo bajo el grupo "Infraestructura"

Para cambiar el nombre a un grupo (sólo podemos cambiar el Display Name, el ID no se puede actualizar):

```
Update-AzureRmManagementGroup -GroupName Madrid -DisplayName "Infraestructura Madrid"
```

Para poder borrar un Grupo de Administración, no debe tener ningún grupo o suscripción en un nivel inferior.

```
Remove-AzureRmManagementGroup -GroupName Madrid
```

Para mover suscripciones entre Grupos de Administración, debemos ser Owner de la suscripción, y Owner o Contributor de los Grupos de Administración.

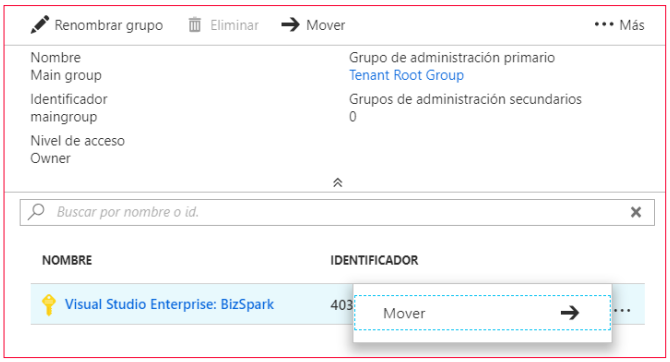


Imagen 4.- Mover suscripción.

En Powershell harían falta dos pasos, en primer lugar, quitar la suscripción del grupo actual, y luego agregarla al nuevo grupo.

```
Remove-AzureRmManagementGroupSubscription -GroupName Madrid -SubscriptionId 12345678-1234-1234-1234-123456789012
New-AzureRmManagementGroupSubscription -GroupName Barcelona -SubscriptionId 12345678-1234-1234-1234-123456789012
```

También Podemos mover Grupos de Administración a otro lugar en la jerarquía. Al mover grupos, todos los recursos, grupos y suscripciones heredados también se moverán.

```
Update-AzureRmManagementGroup -GroupName Madrid -ParentId Desarrollo
```

Especificaciones

- 10,000 grupos de Administración por tenant.
- Cada tenant puede contener 6 niveles de grupos anidados (Excluyendo el grupo raíz y las suscripciones).
- Cada grupo y cada suscripción pertenece a un solo Grupo de Administración.

Permisos

La siguiente tabla muestra los roles RBAC y los permisos sobre los Grupos de Administración.

Nombre de rol de RBAC	Crear	Cambiar nombre	Move	Eliminar	Asignar acceso	Asignar directiva	Lectura
Propietario	X	X	X	X	X	X	X
Colaborador	X	X	X	X			X
Colaborador MG*	X	X	X	X			X
Lector							X
Lector MG*							X
Colaborador de directivas de recursos						X	
Administrador de acceso de usuario					X		

Tabla 1.- Roles y permisos.

Conclusión

Hacía falta ya una forma de gestionar diferentes suscripciones de manera centralizada. Los Grupos de Administración vendrán de maravillas a los administradores, no sólo para reducir las tareas de gobierno, sino también para evitar errores en la asignación de controles, y crear un solo punto de mantenimiento.

Hacía falta ya una forma de gestionar diferentes suscripciones de manera centralizada

Desde Microsoft dicen que esto es solo el principio, y que más funcionalidades llegarán pronto alrededor de los Grupos de Administración. ¡Estaremos atentos!

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

17

SharePoint y Azure: Reconocimiento Óptico de Caracteres (OCR)

¿Qué es OCR?

Optical Character Recognition (OCR, Reconocimiento Óptico de Caracteres) es la operación de conversión de imágenes que contienen texto impreso en texto codificado para computadores. OCR se utiliza extensamente en todo tipo de procesos de IT, variando desde el reconocimiento de texto impreso en papel y escaneado electrónicamente, hasta lectura de documentos de identificación, de tal forma que puedan ser almacenados en sistemas de intercambio de información tales como SharePoint, indexados para facilitar su búsqueda, y clasificados por medio de metadatos.

OCR ha sido históricamente una de las primeras tareas que sistemas de computación ha tenido que realizar, desde los tiempos en que se utilizaban telégrafos mecánicos. Desde la aparición de los teléfonos inteligentes, OCR ha tomado un impulso mayor debido a todo tipo de software que reconoce texto desde las imágenes que los aparatos móviles son capaces de generar.

OCR en Azure

Microsoft ha ofrecido durante muchos años diferentes tipos de software para OCR, integrándolo inclusive en algunos de sus programas, tales como Word y PowerPoint, especialmente para archivos de PDF. Azure ofrece OCR a través del servicio "Computer Vision API" en 25 lenguajes (en el momento de escribir este artículo): Árabe, Chino Simplificado, Chino Tradicional, Checo, Danés, Holandés, Ingles, Fines, Frances, Alemán, Griego, Húngaro, Italiano, Japonés, Coreano, Noruego, Polaco, Portugués, Rumano, Ruso, Serbio (Cirílico y Latino), Eslovaco, Español, Sueco y Turco.

Las imágenes tienen que tener una resolución de entre 40x40 y 3200x3200 pixeles y no ser mayores de 10 megapíxeles. El texto dentro de las imágenes puede estar girado en cualquier ángulo, y el API se encarga de rotarlo para que pueda ser leído horizontalmente. No todos los tipos de letras pueden ser reconocidos con exactitud, especialmente los estilos "artísticos", con imágenes de fondo complejas, con letras muy pequeñas o muy grandes, o con textos tachados, pero el índice de reconocimiento es muy alto, y, por ser un sistema dinámico, Azure "aprende" a reconocer cada vez más texto con el tiempo de utilización.

Azure tiene otros dos servicios con funcionalidad parecida a OCR, pero ambos están en preview en el momento: el "Text Recognition" del Computer Vision API que hace un trabajo muy similar al OCR API, pero reconociendo texto escrito sobre superficies complejas, y el "Azure Media OCR" que puede extraer texto desde videos y películas.

Azure OCR y SharePoint

El servicio de OCR del Azure Computer Vision API se puede utilizar en SharePoint para enriquecer automáticamente la información en el sistema. Cuando se suben imágenes a SharePoint, en realidad el sistema no puede indexar su contenido, solamente los metadatos que posee. Por medio de OCR es posible extraer información directamente desde la imagen, y agregarla al sistema como metadatos, permitiendo clasificarla más exactamente, y obtener resultados de búsqueda más precisos.

OCR es la operación de conversión de imágenes que contienen texto impreso en texto codificado para computadores

En el ejemplo que se va a desarrollar enseguida se utiliza el OCR del Computer Vision API para extraer información de una imagen y agregarla a un campo en la Librería a donde se sube. La Biblioteca de SharePoint dispone para esto de un campo de texto extra para insertar el texto encontrado en la imagen y otro campo de texto para el identificador del lenguaje.

Configuración del Azure Computer Vision API

Para utilizar el Computer Vision API es necesario crear primero el servicio en Azure, aunque también es posible utilizar una cuenta temporal de prueba que se puede crear desde la página de Microsoft https://azure.microsoft.com/en-us/try/cognitive-services/.

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 “Computer Vision API”:

- En el Resource Group, utilice el botón de “+Add” para crear un recurso, busque por “Computer Vision” en la casilla de búsqueda y seleccione “Computer Vision” en los resultados.
- Asígnele un nombre al servicio y utilice el Grupo de Recursos deseado. En la casilla de “Pricing tier” seleccione un nivel dependiendo de la cantidad de consultas a esperar por segundo, lo que determina el precio del servicio (por bloques de mil consultas).

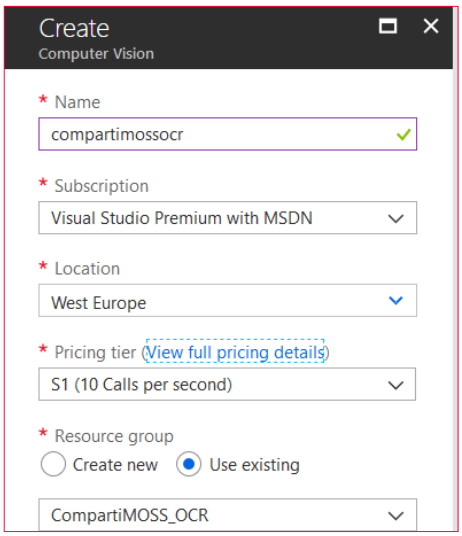


Imagen 1.- Creación del servicio de Computer Vision API.

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 OCR con SharePoint

En el siguiente ejemplo, como se indicó anteriormente, se va a utilizar una Biblioteca de SharePoint con un campo de texto extra llamado “OCRText” y otro para el identificador del lenguaje (“Language”). Cuando se sube una imagen a la Biblioteca, un WebHook hace que una Función de Azure comience a funcionar, extrae el texto y lenguaje en la imagen utilizando el OCR del Computer Vision API y modifica los campos adicionales del documento agregándoles el texto e idioma encontrados.

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).

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

2.– Utilizando Visual Studio 2017 (o Visual Studio 2015 con el AddIn para programar Funciones de Azure), cree una nueva solución del tipo “Azure Function”. Una vez creada la solución, agréguele una Función del tipo “Http Trigger” con derechos de acceso anónimos.

3.– Agréguele a la solución los paquetes NuGet “AppForSharePointOnlineWebToolkit” y “Microsoft.Azure.CognitiveServices.Language” (chequee la casilla de “Include prerelease”).

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

```
[FunctionName(“FunctionOCR”)]
public static async Task<HttpResponseMessage> Run([HttpTrigger(AuthorizationLevel.Anonymous, “post”, Route = null)]
HttpRequestMessage req, TraceWriter log)
{
    log.Info(“FunctionOCR trigger 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);
}
```

es posible extraer información directamente desde la imagen, y agregarla al sistema como metadatos

Esta rutina primero se encarga de hacer el registro del We-

bHook (si la consulta contiene un parámetro “validation-token” 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;
}
```

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 Biblioteca (por medio de la rutina “GetChanges”). En la variable “whSiteListUrl” del App Settings de la función se encuentra el URL del sitio en donde está la Lista a examinar (“https://[Dominio].sharepoint.com/sites/[NombreSitio”)

5.– 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(Lis-
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);
      Microsoft.SharePoint.Client.File myFile = myItem.File;
       ClientResult<System.IO.Stream> myFileStream = myFile.
OpenBinaryStream();
      SPClientContext.Load(myFile);
      SPClientContext.ExecuteQuery();

      // The picture as Byte Array
        byte[] myFileBytes = ConvertStreamToByteArray(myFi-
leStream);

      // Analyze the text
          TextAnalyzeOCRResult myResult = GetAzureTextA-
nalyzeOCR(myFileBytes).Result;
      log.Info($”---- Text Analyze OCR Result : “ + JsonConvert.
SerializeObject(myResult));

      // Insert the values back in the List
      myItem[“Language”] = myResult.language;
      string myText = string.Empty;
       for (int oneLine = 0; oneLine < myResult.regions[0].lines.
Count(); oneLine++)
      {
          for (int oneWord = 0; oneWord < myResult.regions[0].
lines[oneLine].words.Count(); oneWord++)
        {
          myText += myResult.regions[0].lines[oneLine].words[o-
neWord].text + “ “;
        }
      }
      myItem[“OCRText”] = myText;
      myItem.Update();
      SPClientContext.ExecuteQuery();
        log.Info($”---- Text Analyze OCR 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 la imagen agregada, que se convierte en un array de bytes por medio de la rutina “ConvertStreamToByteArray”:

```
public static Byte[] ConvertStreamToByteArray(ClientResul-
t<System.IO.Stream> myFileStream)
{
  Byte[] bytReturn = null;

    using (System.IO.MemoryStream myFileMemoryStream =
new System.IO.MemoryStream())
  {
    if (myFileStream != null)
    {
      myFileStream.Value.CopyTo(myFileMemoryStream);
      bytReturn = myFileMemoryStream.ToArray();
    }
  }

  return bytReturn;
}
```

En la misma rutina se llama a “GetAzureTextAnalyzeOCR”, la que se encarga de hacer la consulta en Azure, utilizando

como parámetro de entrada el array de bytes de la imagen. Esta rutina entrega de regreso un objeto del tipo "TextAnalyzeOCRResult" que contiene los resultados de la consulta y que tiene la forma:

```
public class TextAnalyzeOCRResult
{
  public string language { get; set; }
  public float textAngle { get; set; }
  public string orientation { get; set; }
  public Region[] regions { get; set; }
}

public class Region
{
  public string boundingBox { get; set; }
  public Line[] lines { get; set; }
}

public class Line
{
  public string boundingBox { get; set; }
  public Word[] words { get; set; }
}

public class Word
{
  public string boundingBox { get; set; }
  public string text { get; set; }
}
```

Como Azure retorna un resultado con varios elementos, uno por cada línea de texto encontrada, por medio de un loop se encadenan los valores. Finalmente se utilizan los valores de "language" y "myText" para insertarlos en los campos de "Language" y "OCRText" del Documento de SharePoint.

6.– La rutina "GetAzureTextAnalyzeOCR" recibe como parámetros de entrada el array de bytes de la imagen y retorna un objeto con los valores encontrados por Azure:

```
public static async Task<TextAnalyzeOCRResult> GetAzure-
TextAnalyzeOCR(byte[] myFileBytes)
{
    TextAnalyzeOCRResult resultReturn = new TextAnalyzeO-
CRResult();

  HttpClient client = new HttpClient();

  // Request headers
     client.DefaultRequestHeaders.Add("Ocp-Apim-Subscrip-
tion-Key", ConfigurationManager.AppSettings["azVisionApi-
ServiceKey"]);

  // Request parameters.
   string requestParameters = "language=unk&detectOrienta-
tion=true";

  // Assemble the URI for the REST API Call.
   string uri = ConfigurationManager.AppSettings["azVisionA-
piOcrEndpoint"] + "?" + requestParameters;
  string contentString = string.Empty;

  HttpResponseMessage response;

   using (ByteArrayContent content = new ByteArrayConten-
t(myFileBytes))
  {
    content.Headers.ContentType = new MediaTypeHeaderVa-
lue("application/octet-stream");

     // Execute the REST API call
     response = await client.PostAsync(uri, content);

     // Get the JSON response
    contentString = await response.Content.ReadAsStringAsy-
nc();

         resultReturn = JsonConvert.DeserializeObject<TextA-
nalyzeOCRResult>(contentString);
     return resultReturn;
  }
}
```

Cada consulta 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 "azVisionApiOcrEndpoint" y que es "https://westeurope.api.cognitive.microsoft.com/vision/v1.0/ocr"), utilizando como parámetros en el QueryString el "language" ("unk", para forzar un resultado de retorno) y si es necesario detectar la orientación de la imagen ("detectOrientation"). En la App Settings "azVisionApiServiceKey" se mantiene el valor de la llave mencionada en el punto 4.

7.– 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.Igno-
re)]
  public string Id { get; set; }

    [JsonProperty(PropertyName = "clientState", NullValue-
Handling = 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", NullValueHand-
ling = NullValueHandling.Ignore)]
  public string Resource { get; set; }
}
```

8.– Registre el WebHook en la Biblioteca de SharePoint y suba una imagen que contenga texto. El WebHook hará que la Función realice su trabajo, entregue los resultados encontrados por Azure y muestre el idioma y el texto en la imagen:

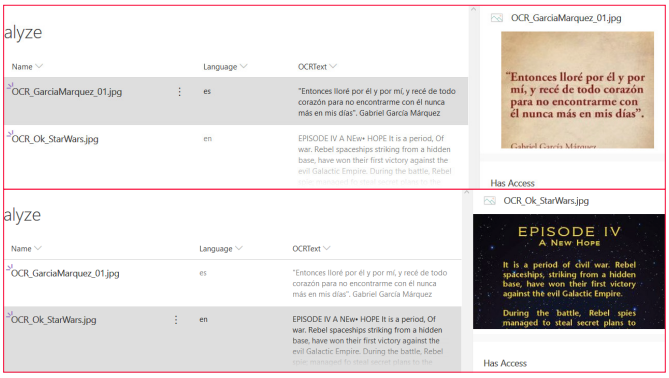


Imagen 2.- Imágenes con texto recobrado por el servicio de OCR.

El servicio de OCR del Azure Computer Vision API permite enriquecer la informacion que los usuarios suben a SharePoint

Conclusiones

El servicio de OCR del Azure Computer Vision API permite enriquecer la información que los usuarios guardan en SharePoint extrayendo el texto que se encuentra en imágenes. El servicio de OCR 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 Apps and Services
gustavo@gavd.net
http://www.gavd.net

i S 22

Mi nombre es Sergio Hernández Mancebo, soy natural de Madrid, pero con sangre leonesa por parte de padre. Actualmente vivo en un barrio con mayúsculas como es Vallecas, pero en verdad me he criado siempre en la sierra de Madrid, en concreto en una bonita localidad del parque natural del Guadarrama que se llama Collado Mediano.

Ingeniero Informático de profesión, llevo cerca de 8 años de carrera profesional, siempre vinculado al mundo Microsoft, en la actualidad disfrutando de una bonita familia como es Encamina, en la que me han dado total libertad para liderar el área de proyectos y metodología desde mi puesto de Principal Team Leader.

Si no me podía quejar absolutamente de nada en lo profesional, ahora mucho menos ya que este pasado mes de agosto, Microsoft me ha otorgado el premio MVP en Azure, ¡¡¡regalazo para mí, 32 cumpleaños y un sueño cumplido!!!

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

Empecé posiblemente como mejor se puede empezar, por vocación y mejor aún sin saber absolutamente nada o casi nada. Como casi todos empecé muy joven a romper ordenadores en casa, a los cuales mi padre acompañaba con una ligera bronca, y una sustitución de piezas cuando se le pasaba el mosqueo (no es broma más de una fuente de alimentación me llevé por delante haciendo pruebas).

Al salir de la carrera, con 24 años ya llevaba muchos años, noches y fines de semana con horas de trabajo a mis espaldas, pero no en nada relacionado con la informática o algo que se parezca, en realidad trabajaba en el mundo de la hostelería. Por este motivo, creo que cogí con unas ganas increíbles la oportunidad de empezar en el mundo de la tecnología, pero a la vez con la madurez necesaria para saber encarar un cliente o un proyecto complejo.

Como mi conocimiento no era muy amplio, y casi nulo en .NET, cuando me ficharon para el primer Partner de Microsoft, mi respuesta a casi toda pregunta respecto a ¿Sabes de .NET? ¿C#? ¿SharePoint?, fue un categórico NO. Aun así, el responsable que ahora es amigo y al que le debo mucho por esto, me fichó y con el tiempo me admitió que tanta sinceridad nunca pueda ser mala, y eso muchas veces pesa más que el control o no en una materia.

¿Cuáles son tus principales actividades tecnológicas hoy en día?

Mi perfil es tan camaleónico que a veces me pierdo yo mismo. En la actualidad reparto mi tiempo entre liderar equipos de trabajo, ya que llevo a mi cargo el equipo de Team Leader de Encamina, intentar poner orden en la Cartera de proyectos (así como en el buen hacer de los mismos), involucrarme en tareas de preventa, y como no, acercarme a los proyectos (aunque así en la intimidad debo decir que me gusta también mucho), haciendo de Team Leader, de desarrollador o de lo que se necesite siempre que el equipo lo necesite.

Estas actividades las compagino con mi verdadera pasión a nivel laboral (aunque ya se ha convertido en un hobbie) que es dedicarme a la comunidad. Ya sea con pequeños grupos de usuarios en los que participo como Crossdevelup, o acudiendo a grandes eventos técnicos cada fin de semana. Estas actividades me permiten estar al día de todo, y disfrutar un poco de esta profesión, que a veces se hace un poco dura.

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

Pues la verdad es que no tengo término medio. Mi principal pasión es pasar tiempo con mi mujer, que además entre otras cosas es mi mejor amiga y con la que más disfruto el tiempo.

Si tenemos la oportunidad nos encanta escaparnos y si es cruzando el charco a nuestro querido USA mejor que mejor (somos cowboys frustrados he de admitirlo), pero si no, siempre nos gusta pasar una buena tarde de domingo, con una manta y nuestro atracón de series de Netflix, ¡¡¡umm que más se puede pedir!!!

Además, tengo otra adicción en esta vida y es el futbol, sí lo reconozco, no me pega nada con todo lo anterior o sí no sé. Soy además fan incondicional del Deportivo de la Coruña (desde muy pequeño) y del Rayo Vallecano (por adopción en mi barrio actual). Ahora me he moderado muchísimo, pero en origen, me podía ver fácil 10 partidos al fin de semana, y conocerme la alineación de todos los equipos de las grandes ligas o no tan grandes.

¡¡¡Y por si fuera poco me fascina todo tipo de deportes, bien sea viéndolo o practicándolo!!!

Esas tardes de tour (con siesta incluida), de la hierba de Wimblendon entre Nadal y Fededer, o noches en vela viendo a Mike Tyson dando mamporros.

¿Cuáles son tus hobbies?

Como ya he dicho pasar tiempo con la gente que merece la pena, mi mujer, mi familia y mis amigos. Tengo la suerte de mantener esas amistades que haces siendo crio, y aun con todo lo que ha pasado estos últimos años siguen ahí, somos una verdadera piña!!!

Más así en plan friki pues como casi todos del gremio, soy un apasionado de las consolas (me da igual el fabricante, pero me decanto un poco más por XBOX claro está), y mi otra gran pasión es el CINE. Si lo pongo en mayúsculas porque no se la de pelis que he podido ver en el cine, en casa, o escuchar, si porque disfrutar de una buena banda sonora de Hans Zimmer o John Williams no tiene desperdicio. Reto a quien quiera a jugar a las películas poniendo unos pocos acordes de cada banda sonora, tengo que decir que tengo un don para averiguar de qué película y canción se trata (si la he visto claro, que hay mucho cine).

Y un hobbie que he perdido y debo recuperar es la lectura, antes me pasaba horas leyendo novela fantástica, tengo mi record personal en 7 lecturas del señor de los anillos y no muchas menos del Hobbit, ya que Tolkien es mi perdición, y no me pregunten por juego de tronos, no es un foro para entrar en disputas!!

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

Pues yo creo y espero que vaya como hasta ahora, para mejor. Creo que hay fabricantes muy potentes en el mercado, no solo Microsoft que están permitiendo tener un desarrollo tecnológico brutal. Espero que esto se invierta en verdaderas mejoras para la humanidad, ya que los WhatsApp o los Facebook son muy necesarios para salir y evadirnos del trabajo y disfrutar un poco, pero también tenemos medios para invertir más en I + D, en desarrollo para la gente Invidente o con minusvalías, y sobre todo podemos hacer de esta sociedad un mundo más cómodo para todos y no para unos pocos.

Ya más centrado en lo más cercano, creo que nos está tocando vivir un momento precioso, la expansión hacia el Cloud nos está dejando liberar la mente, pensar en soluciones que hace unos años no estaban ni al alcance de plantearse, y nos hace crecer a todos en lo técnico y en lo personal.

Creo que el Cloud es el presente, el futuro cercano es el AI y no perdamos de vista al mundo IoT, de la Robótica o la Bioinformática.

La próxima década puede ser la definitiva, no se si en modo Blade Runner y replicantes, si más en modo Skynet y Terminator o como a mí más me gustaría, Jurassic Park, pero lo que es seguro es que esto va a cambiar mucho mucho, espero desde mi pequeño espacio poder aportar algo a todo esto.

SERGIO HERNÁNDEZ MANCEBO
Principal Team Leader
Azure MVP
@shmacenbo

Modernizando sitios clásicos de SharePoint Online con Grupos de Office 365

Por defecto, en SharePoint Online (SPO) podemos tener distintos tipos de sitios en función del tipo de plantilla que se ha elegido en el momento de creación. Podemos decir que las plantillas disponibles se ubican en dos grandes categorías: plantillas para sitios clásicos de SPO y plantillas para sitios modernos de SPO. En el caso de sitios modernos de SPO, actualmente tenemos dos tipos de plantillas: Sitio moderno de Grupo y Sitio de Comunicación. Ahora bien: ¿Es posible "modernizar" un sitio clásico de SPO de forma rápida de forma que podamos tener todas las características que aportan a los sitios modernos, pero en nuestros sitios clásicos? La respuesta es sí y en este artículo veremos una de las posibilidades de modernización de un sitio clásico de SPO a través de añadir un nuevo Grupo de Office 365.

Preparando el escenario

Lo primero que necesitaremos para hacer uso de la característica de añadir un nuevo Grupo de Office 365 a un sitio clásico de SPO es disponer de ese sitio clásico:

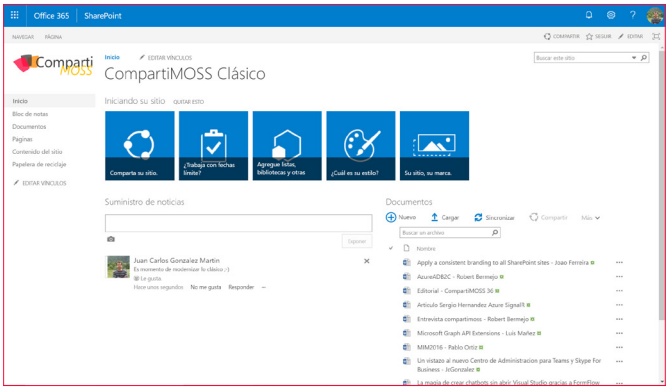


Imagen 1.- Sitio clásico de SPO.

Este sitio lo tendremos configurado de acuerdo con nuestros requerimientos. Por ejemplo, en mi caso he configurado dos usuarios adicionales en el sitio:

- Un usuario con rol de propietario del sitio.
- Otro usuario con rol de integrante del sitio.

Como veréis más adelante, os detallo este setup en la configuración del sitio porque el proceso de añadir un nuevo Grupo de Office 365 al sitio clásico tiene que tener en cuenta (y respetar) las configuraciones de mi sitio clásico. En concreto:

- Cuando se añada el Grupo de Office 365 al sitio, los usuarios propietarios del sitio pasarán a ser propietarios del Grupo de Office 365.
- De la misma forma, los usuarios integrantes del sitio pasarán a ser integrantes del Grupo de Office 365.

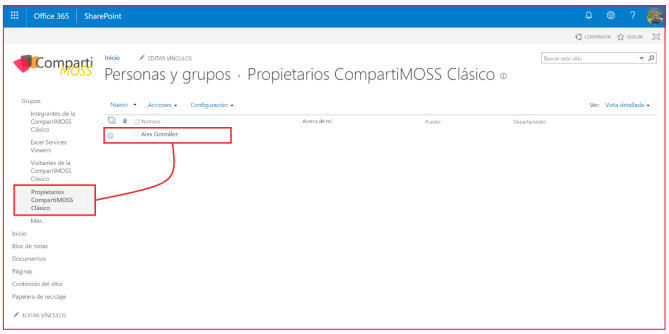


Imagen 2.- Configuraciones particulares del sitio clásico de SPO.

Añadiendo un nuevo Grupo de Office 365 a un sitio clásico de SPO

Una vez que tenemos nuestro escenario de sitio clásico de SPO a conectar con un Grupo de Office 365, ya estamos listos para añadir un Grupo al sitio. Para hacer esta operación, tenemos dos posibilidades:

- La interfaz de usuario.
- PowerShell.

Para conectar un sitio clásico (Nota: Aunque estoy en todo momento hablando de sitio clásico, es importante tener en cuenta que sólo es posible añadir Grupos de Office 365 a Colecciones de Sitios de SPO, no a subsitios de una colección) de SPO por medio de la interfaz de usuario:

- Desde cualquier página del sitio, hacemos clic en el icono de configuración de la barra superior y a continuación en "Conectar a un nuevo grupo de Office 365":

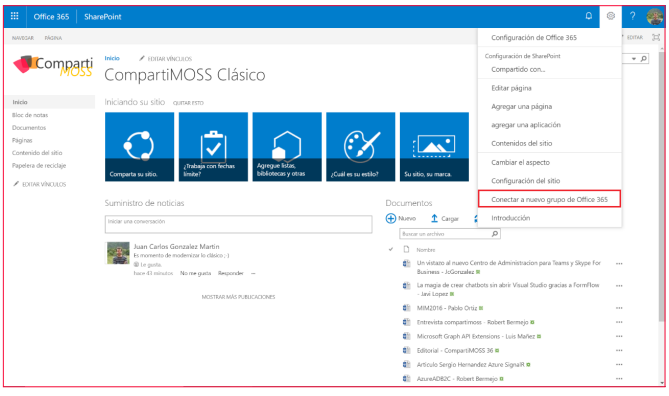


Imagen 3.- Opción "Conectar un nuevo grupo de Office 365" en un sitio clásico de SPO.

- A continuación, se muestra un panel que da inicio al

asistente para configurar el nuevo Grupo de Office 365 a añadir al sitio. Como se aprecia en la Imagen 4, el primer panel del asistente muestra información relativa a lo que supone añadir un nuevo Grupo de Office 365 al sitio:

- El contenido del sitio y configuraciones se mantiene.
- Se crea una nueva página principal para el sitio que es una página moderna.
- Se crea un Grupo de Office 365 que aporta todos los elementos propios de Grupos:
  - Un buzón de correo dónde tienen lugar las conversaciones del Grupo.
  - Un calendario del Grupo de Office 365 para realizar convocatorias de este.
  - Un Plan de Planner para planificar el trabajo de los integrantes del Grupo.
  - La posibilidad de añadir un Team de Microsoft Teams a partir del Grupo.

podemos tener distintos tipos de sitios en función del tipo de plantilla que se ha elegido en el momento de creación.

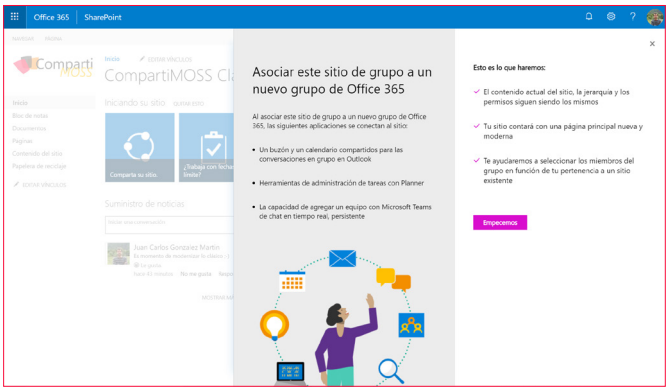


Imagen 4.- Primera pantalla del asistente para añadir un nuevo Grupo de Office 365 al sitio.

- Para iniciar el siguiente paso del asistente, hacemos clic en “Empecemos” de forma que se muestra un nuevo panel en el que podemos configurar aspectos relativos tanto al Sitio de SPO como al Grupo de Office 365 a crear. En concreto (Imagen 5) podremos configurar lo siguiente:
  - El nombre del Grupo de Office 365 a crear.
  - El alias del Grupo de Office 365 a crear.
  - La descripción del Sitio de SPO.
  - La privacidad del Grupo de Office 365 a crear (Público o privado).
  - En el caso en el que se haya configurado en el tenant de Office 365, el grado de confidencialidad de los datos del Grupo.

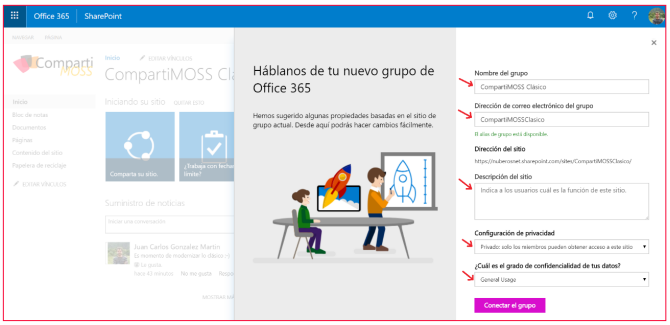


Imagen 5.- Parámetros que se pueden configurar en el Grupo de Office 365.

- A continuación, hacemos clic en “Conectar el grupo” para desencadenar la creación del Grupo de Office 365. Esta acción supone que se muestre un nuevo panel en el que podemos configurar los propietarios e integrantes del Grupo que se va a crear. Como se aprecia en la Imagen 6, el proceso por defecto configura a los usuarios en el grupo de propietarios del sitio como propietarios del Grupo y a los usuarios del grupo de integrantes del sitio como integrantes del Grupo:

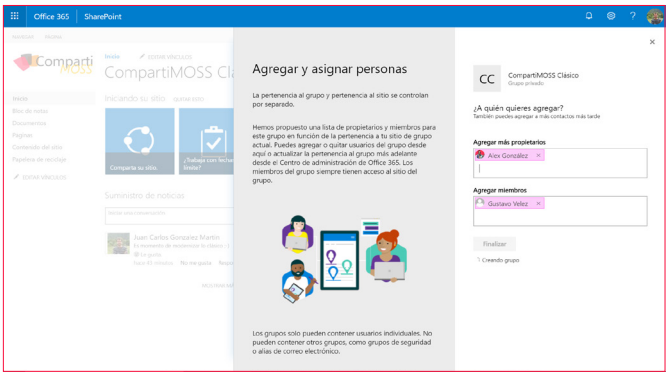


Imagen 6.- Proceso de creación del Grupo de Office 365.

- Para finalizar, haremos clic en el botón “Finalizar” del panel de forma que se muestra la nueva página principal (y moderna) del sitio desde la que podremos acceder al resto de recursos del Grupo creado:

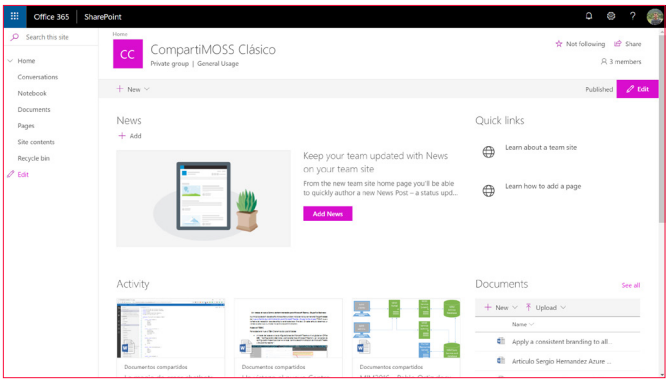


Imagen 7.- Nueva página principal del sitio.

Como se puede apreciar, la página principal cuenta con varias páginas modernas que se han añadido de forma automática en el proceso de creación del Grupo (Noticias, Enlaces rápidos, Actividad y Documentos).

Elementos que se añaden al sitio moderno cuando se crea el Grupo

Además de la nueva página principal moderna añadida al

Sitio, podremos comprobar que los siguientes elementos asociados al Grupo de Office 365 han sido creados/configurados:

- Un buzón de correo para las conversaciones del Grupo y un calendario para las reuniones en las que participen los miembros del Grupo (Nota: Es posible que el buzón del Grupo no esté disponible de forma automática una vez acabe el proceso. Eso se debe a que el proceso de creación del Grupo y los elementos asociados se realiza en hilos diferentes):

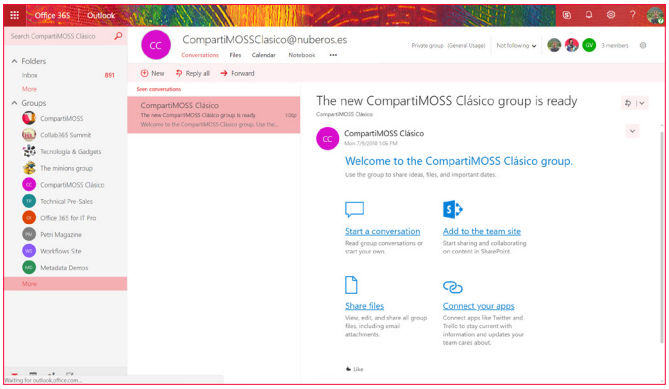


Imagen 8.- Buzón del Grupo creado.

- Un plan de Planner.

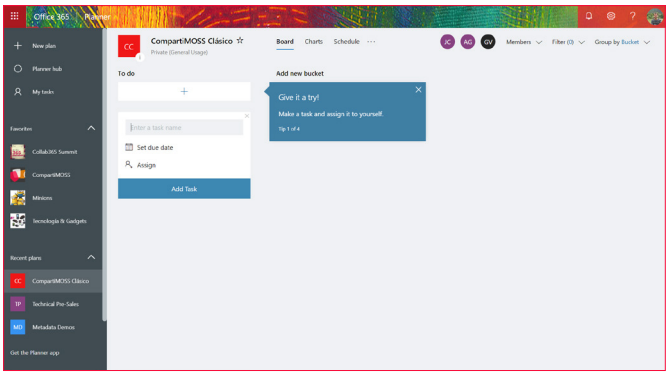


Imagen 9.- Plan de Planner asociado al Grupo.

- Los usuarios propietarios e integrantes del sitio se añaden como propietarios e integrantes del Grupo respectivamente.

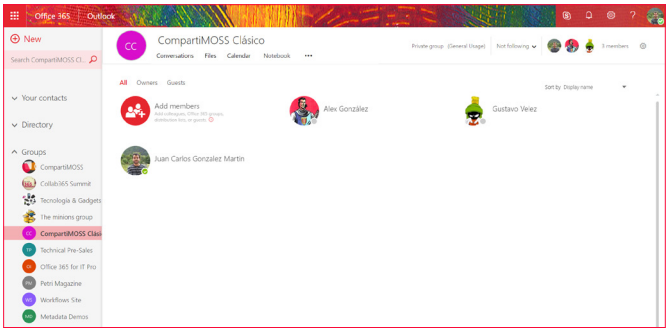


Imagen 10.- Configuración de Membership del Grupo.

Añadiendo un nuevo Grupo de Office 365 a un sitio clásico de SPO con PowerShell

Como alternativa a añadir un nuevo Grupo de Office 365 a un sitio clásico existente mediante la interfaz de usuario, podemos hacer uso de PowerShell:

- En primer lugar, necesitaremos descargar e instalar la última versión disponible del SharePoint Online Management Shell del siguiente enlace: https://www.microsoft.com/en-us/download/details.aspx?id=35588
- A continuación, bien en el propio SharePoint Online Management Shell o bien en nuestro entorno PowerShell favorito (Visual Studio Code por ejemplo), pegamos el siguiente código PowerShell que permite añadir un Grupo de Office 365 a un sitio existente de SPO:

```
$sUserName="<Office365_User>@<Office365_Domain>.onmicrosoft.com"
$sMessage="Type your Office 365 Credentials"
$sSPOAdminCenterUrl="https://<Office365_Domain>-admin.sharepoint.com"
$sClassicSPOSite="https://<Office365_Domain>.sharepoint.com/sites/<Site_Name>/"
$O365Cred=Get-Credential -UserName $sUserName -Message $sMessage
Connect-SPOService -Url $sSPOAdminCenterUrl -Credential $O365Cred
$sGroupDisplayName="<Group Display Name>"
$sGroupAlias="<GroupAlias>"
Set-SPOSiteOffice365Group -Site $sClassicSPOSite -DisplayName $sGroupDisplayName -Alias $sGroupAlias -IsPublic $true
```

- Y como resultado, obtendremos los mismos elementos ya vistos al añadir un Grupo de Office 365 a un sitio de SPO haciendo uso de la interfaz de usuario.

Algunos puntos de mejora en la creación del Grupo

Algunos puntos de mejora en la creación del Grupo que podréis comprobar son los siguientes:

- Si el Sitio a conectar cuenta con un logo para el sitio, este logo no se traslada al Grupo de Office 365 como cabría esperar.

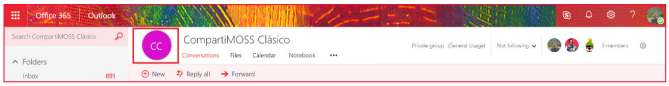


Imagen 11.- Logo del Grupo de Office 365.

- En la parte de Membership, inicialmente veréis que los usuarios propietarios del sitio no aparecen como propietarios del Grupo de Office 365.

Conclusiones

Tal y como se había anunciado en Microsoft Ignite 2017, desde hace unos meses tenemos de forma nativa la posibilidad de modernizar sitios clásicos de SPO a través de añadir un nuevo Grupo de Office 365 a dichos sitios. Microsoft proporciona dos mecanismos para añadir un nuevo Grupo de Office 365 a un sitio: la interfaz de usuario y PowerShell.

JUAN CARLOS GONZÁLEZ
Cloud & Productivity Advisor
Office Apps & Services MVP
jcgm1978
https://jcgonzalezmartin.wordpress.com/

27

Apply SPFx extensions to SharePoint Hub Sites using PnP PowerShell

Hub sites bring a set of common functions to all sites that belong the hub, but extensions are not propagated to all sites automatically. With PnP PowerShell is possible to automate the installation of Application Customizers through the sites, the script below search for all sites associated with hub and apply the custom action to all of them.

Before running the script, you will need to have PnP PowerShell installed, the latest version can be installed from here. The scripts will not install the extension using ALM APIs, to get this working, you need to install the extension manually on the global app catalog.

Installing

Before running the script, you need to adjust the main variables to your own values:

- tenantAdmin = "https://contoso-admin.sharepoint.com"
- hubSite = "https://contoso.sharepoint.com/sites/hub"
- extensionGUID = "6da1a9e8-471d-4f39-80e6-a7ded02e8881"
- extensionName = "Extension Name"
- extensionTitle = "Extension Title"

Hub sites bring a set of common functions to all sites that belong the hub

You can get your extension GUID from the extension project, open the manifest.json and copy the id value.

```
cls

$tenantAdmin = "https://contoso-admin.sharepoint.com"
$hubSite = "https://contoso.sharepoint.com/sites/hub"
$extensionGUID = "6da1a9e8-471d-4f39-80e6-a7ded02e8881"
$extensionName = "Extension Name"
$extensionTitle = "Extension Title"

try
{
    Connect-PnPOnline -Url $tenantAdmin -UseWebLogin
} catch {
    Write-Host "Unable to connect."
    exit
}

$HubSite = Get-PnPHubSite $hubSite
$HubSiteId = $HubSite.Id
$ModernSites = (Get-PnPTenantSite -Template 'GROUP#0') + (Get-PnPTenantSite -Template 'SITEPAGEPUBLISHING#0')
$SitesFromHub = New-Object System.Collections.ArrayList

Write-Host ("Searching {0} sites:" -f $HubSite.Title) -BackgroundColor Gray
foreach ($ModernSite in $ModernSites){
    $site = Get-PnPHubSite $ModernSite.Url
    if($site.SiteUrl){
        if($site.SiteId -eq $HubSiteId){
            Write-Host ("* {0} - {1}" -f $ModernSite.Title, $ModernSite.Url)
            $SitesFromHub.Add($ModernSite) | Out-Null
        }
    }
}

Write-Host ""
Write-Host "Installing at:" -BackgroundColor Gray
foreach ($SiteHub in $SitesFromHub){
    Write-Host ("* {0} - {1} ... " -f $SiteHub.Title, $SiteHub.Url) -NoNewline
    Connect-PnPOnline -Url $SiteHub.Url -UseWebLogin
    Add-PnPCustomAction -ClientSideComponentId $extensionGUID -Name $extensionName -Title $extensionTitle -Location ClientSideExtension.ApplicationCustomizer -Scope site
    Write-Host "Done" -BackgroundColor Green
    Disconnect-PnPOnline
}

Write-Host "All Done"
```

The execution of the script will identify all the sites belonging to the Hub and will print if the installation was achieved with success.

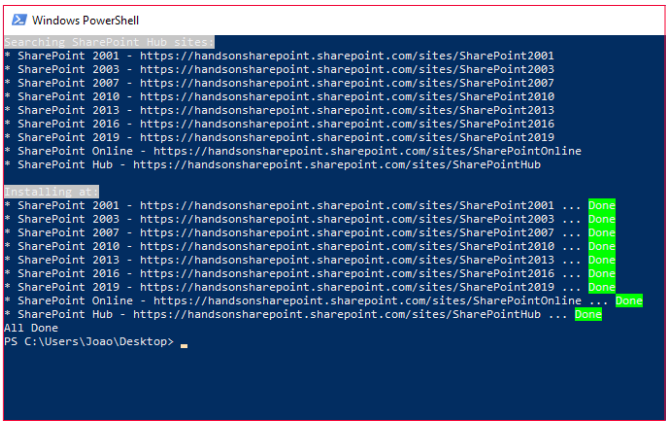


Image 1.- Installing Extension on Hub sites

Uninstalling

In case you want to remove the extensions from all the hub sites I'm also providing a script to do it, but the extension must be installed with the same name in all sites. Every

time an extension is installed a new identifier for the extension is generated and the removal requires that id, it can be retrieve using PowerShell if we know in advance what was the installation name.

PnP PowerShell comes handy to automate processes on modern SharePoint

Before running the script, you need to adjust the main variables to your own values:

- tenantAdmin = "https://contoso-admin.sharepoint.com"
- hubSite = "https://contoso.sharepoint.com/sites/hub"
- extensionName = "Extension Name"

```
cls

$tenantAdmin = "https://contoso-admin.sharepoint.com"
$hubSite = "https://contoso.sharepoint.com/sites/hub"
$extensionName = "Header"

try
{
        Connect-PnPOnline -Url $tenantAdmin -UseWebLogin
} catch {
        Write-Host "Unable to connect."
        exit
}

$HubSite = Get-PnPHubSite $hubSite
$HubSiteId = $HubSite.SiteId
$ModernSites = (Get-PnPTenantSite -Template 'GROUP#0') +
(Get-PnPTenantSite -Template 'SITEPAGEPUBLISHING#0')
$SitesFromHub = New-Object System.Collections.ArrayList

Write-Host ("Searching {0} sites:" -f $HubSite.Title) -Back-
groundColor Gray

foreach ($ModernSite in $ModernSites){
        $site = Get-PnPHubSite $ModernSite.Url
        if($site.SiteUrl){
                if($site.SiteId -eq $HubSiteId){
                        Write-Host ("* {0} - {1}" -f $Modern-
Site.Title, $ModernSite.Url)
                        $SitesFromHub.Add($ModernSite)
| Out-Null
                }
        }
}

Write-Host ""
Write-Host "Uninstalling from:" -BackgroundColor Gray
foreach ($SiteHub in $SitesFromHub){
        $removed = $false
        Write-Host ("* {0} - {1} ... " -f $SiteHub.Title, $SiteHub.Url)
-NoNewline
        Connect-PnPOnline -Url $SiteHub.Url -UseWebLogin

        $customActions = Get-PnPCustomAction -Scope site
        foreach ($customAction in $customActions){
                if($customAction.Name  -eq  $extensionNa-
me){
                        Remove-PnPCustomAction -scope
site -identity $customAction.Id -Force
                        $removed = $true
                }
        }
        if($removed){
                write-host "Done" -BackgroundColor Green
        } else {
                write-host "Not found" -BackgroundColor Ye-
llow -ForegroundColor Black
        }
        Disconnect-PnPOnline
}


Write-Host "All Done"
```

The execution of the script will identify all the sites belonging to the Hub and will print if the uninstall was achieved with success.

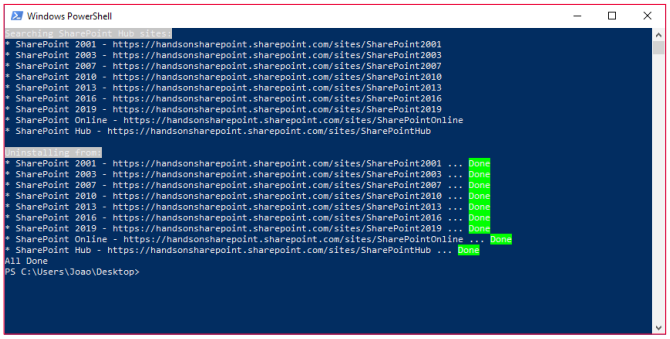


Image 2.- Uninstall Extension from hub sites

Conclusion

PnP PowerShell comes handy to automate processes on modern SharePoint especially when there are no graphical interfaces available, a must-have for every SharePoint administrator.

JOAO FERREIRA
SharePoint Team Lead at BindTuning
@joao12ferreira

29

Asegurando Aplicaciones React con Azure AD

La mayoría de los desarrolladores, actualmente han estudiado las tecnologías ReactJS, Angular y VUE como fundamentales para la creación de nuevas aplicaciones. Esta información es de conocimiento de programadores incluso de los que trabajan en ASP.NET, para crear aplicaciones que hagan más ágil la transmisión de información, ya que la aplicación no se refrescaría constantemente del lado del servidor. Desde mi punto de vista, crear aplicaciones ASP.NET MVC ya no es suficiente para tener una buena experiencia de usuario, es necesario tener un framework del lado del cliente para hacer una experiencia que agrade al usuario final.

A pesar de la diversidad de textos, artículos, blogs y manuales, aún existen lagunas en la información. Al no encontrar alguna publicación con las explicaciones que necesitaba surgió la idea de este artículo. El presente texto es creado con el objetivo de mostrar cómo se configurar una aplicación ReactJS que consume un Web API para que se autentique con usuarios creados en el directorio Activo de Azure (el WebAPI también está protegido por Azure AD).

Lo primero que debemos hacer es crear un registro de aplicación en el directorio activo de Azure, como se muestra en las imágenes siguientes:

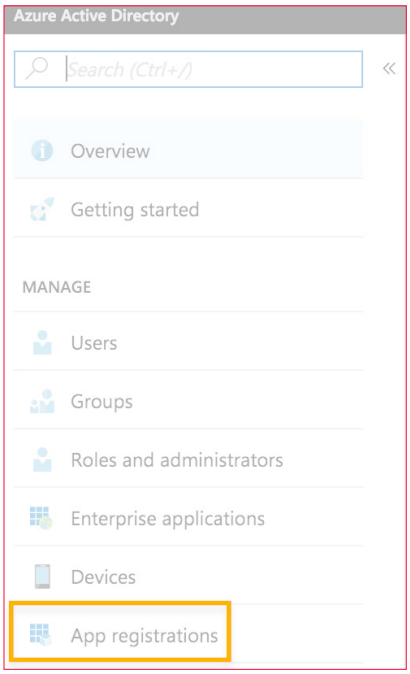


Imagen 1.- Acceso al registro de aplicaciones en Azure AD.

Luego creamos una aplicación con los siguientes datos:

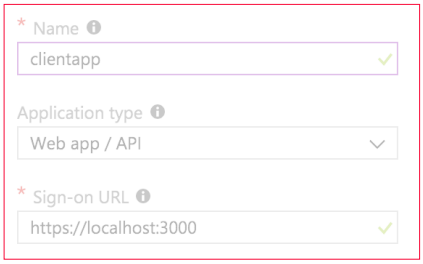


Imagen 2.- Creación del registro de aplicación.

Al terminar de crear la aplicación, debemos tomar nota del App Id:

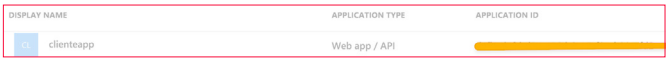


Imagen 3.- App ID de la aplicación creada.

Para este articulo asumimos que el lector tiene conocimientos básicos en ReactJS y que ya tiene una aplicación básica creada.

Ahora en nuestra app, tenemos que instalar el paquete React-adal, las instrucciones se encuentran en este sitio web: https://github.com/salvoravida/react-adal

Después de haber instalado el paquete React-adal, debemos configurarlo, para esto debemos crear un archivo adal-config.js

```
import { AuthenticationContext, adalFetch, withAdalLogin }
from 'react-adal';

export const adalConfig = {
 tenant: 'aaaaaaaa-c220-48a2-a73f-1177fa2c098e',
 clientId: 'aaaaaaa-bd54-456d-8aa7-f8cab3147fd2',
 endpoints: {
  api:'aaaaaaaa-abaa-4519-82cf-e9d022b87536'
 },
   'apiUrl':  'https://ourfuturewebappi-app.azurewebsites.net/
api',
 cacheLocation: 'localStorage'
};

export const authContext = new AuthenticationContext(adal-
Config);

export const adalApiFetch = (fetch, url, options) =>
    adalFetch(authContext,  adalConfig.endpoints.api,  fetch,
adalConfig.apiUrl+url, options);

export const withAdalLoginApi = withAdalLogin(authContext,
adalConfig.endpoints.api);
```

En este archivo, debemos cambiar los valores de:

- Tenant: es el id del directorio activo.
- clientId: es el id del registro de la app en el directorio activo.
- Endpoints: es un arreglo de objetos clave valor, en este arreglo se listan los API que queremos consumir desde la aplicación React, y que también están protegidos por Azure AD. (es el ID de la App registrada en el directorio activo, no es el ID del App Service).

Después de esto y, para terminar, apiURL es la URL del web api que vamos a consumir (En un momento detallaremos mas la parte de configuración del web api). Volviendo a la aplicación React, debemos crear un archivo index.js, es en este archivo es donde utilizamos el paquete instalado anteriormente en el texto.

```
import React from 'react';
import ReactDOM from 'react-dom';
import DashApp from './dashApp';
import registerServiceWorker from './registerServiceWorker';
import 'antd/dist/antd.css';
import { runWithAdal } from 'react-adal';
import { authContext } from './adalConfig';

const DO_NOT_LOGIN = false;
runWithAdal(authContext, () => {
   ReactDOM.render(<DashApp />, document.getElementById('root'));
 // Hot Module Replacement API
 if (module.hot) {
   module.hot.accept('./dashApp.js', () => {
     const NextApp = require('./dashApp').default;
      ReactDOM.render(<NextApp />, document.getElementById('root'));
   });
 }

},DO_NOT_LOGIN);

registerServiceWorker();
```

En el código anterior se puede observar la utilización del método runWithAdal, y dentro del este se coloca el componente principal desarrollado en ReactJS y así concluye el proceso.

Al correr nuestra aplicación por primera vez, la app se direccionará a la página de login de Microsoft, una vez autenticados, Microsoft nos enviará de vuelta a nuestra aplicación local de React con el Bearer token, el cual será guardado en el almacenamiento local del navegador según lo configuramos en los pasos anteriores.

Hasta esta sección solo hemos hecho la parte de la autenticación de la aplicación del lado del cliente, es decir el frontend, pero como es sabido las aplicaciones tienen también funcionalidad del lado del servidor, es decir la aplicación ReactJS al procesar algo del lado del servidor a través de un Web API. Es importante mencionar que este Web API también tiene que estar protegido para que solo puedan utilizarlos usuarios o aplicaciones registradas en el directorio activo de Azure.

Para poder realizar este procedimiento es necesario regresar al portal de Azure para registrar una segunda aplicación, esta vez, será el Reply URL, el Web API que tiene que existir previo a este procedimiento, si no se tiene se puede referir a la página: (Como crear un Web API)

Es importante que al final de la URL de nuestro Web API publicado, se agregue lo siguiente: .auth/login/aad/callback. Esto es necesario para que la autenticación del directorio activo de Azure también funcione con nuestro web api.

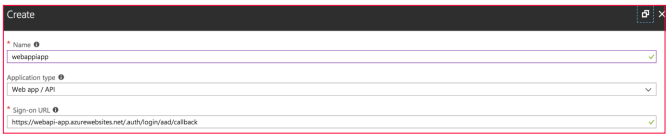


Imagen 4.- Registro de la nueva aplicación.

Después de esto se debe editar el manifiesto de nuestra aplicación registrada y es importante cambiar la configuración que se muestra a continuación, a través de KnownClientApplications. KnownClientApplications, es un arreglo de aplicaciones que pueden consumir nuestro Web API. En nuestro caso es el id de la aplicación registrada anteriormente.

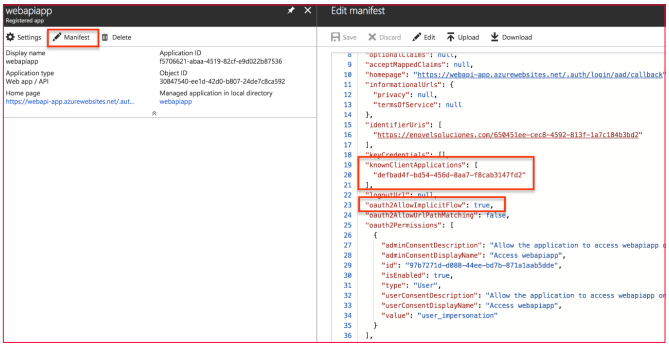


Imagen 5.- Valor del KnowClientApplications.

No se debe dejar de lado la configuración de nuestro Web API para autenticación con el directorio activo, seleccionamos nuestra web app en el portal. Y nos aseguramos de que el Web API este configurado como se muestra a continuación:

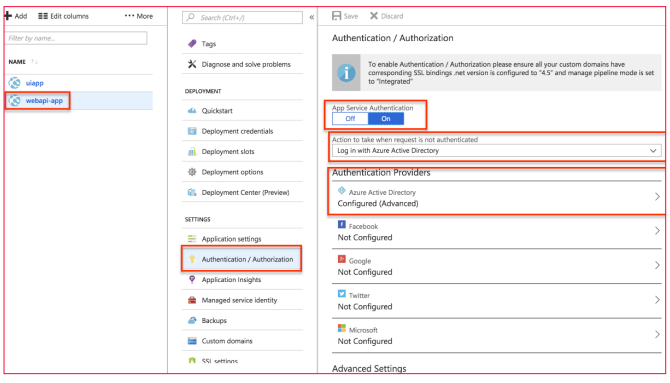


Imagen 6.- Configuración de la autenticación del Web API.

Para proseguir, se necesita el ClientID, ya que es el id de la Web API registrada en el directorio activo de Azure, y así el Issuer Url termina con el ID del directorio activo de Azure.

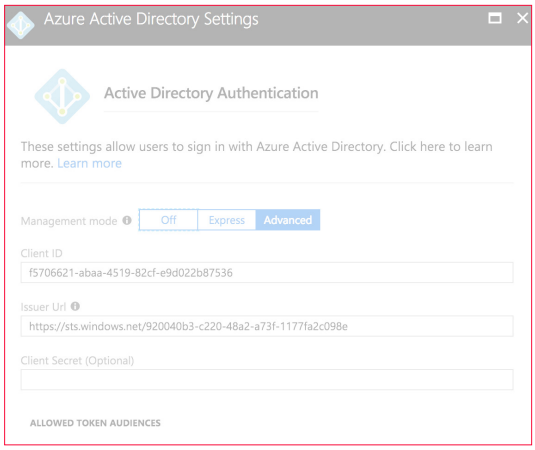


Imagen 7.- Client ID e Issuer Url.

Como ya se había mencionado anteriormente, en el archivo adalconfig.js debemos configurar el id del Web API registrado y la Url del api, como se observa a continuación.

```
import { AuthenticationContext, adalFetch, withAdalLogin } from 'react-adal';

export const adalConfig = {
  tenant: '920040b3-c220-48a2-a73f-1177fa2c098e',
  clientId: 'defbad4f-bd54-456d-8aa7-f8cab3147fd2',
  endpoints: {
    api:'f5706621-abaa-4519-82cf-e9d022b87536'
  },
  'apiUrl': 'https://webapi-app.azurewebsites.net/api',
  cacheLocation: 'localStorage'
};

export const authContext = new AuthenticationContext(adalConfig);

export const adalApiFetch = (fetch, url, options) =>
  adalFetch(authContext, adalConfig.endpoints.api, fetch, adalConfig.apiUrl+url, options);

export const withAdalLoginApi = withAdalLogin(authContext, adalConfig.endpoints.api);
```

Una vez realizamos esto, ya podemos utilizar un controlador del API de manera segura como en el siguiente componente:

```
import React, { Component } from 'react';

import { Row, Col } from 'antd';
import PageHeader from '../../components/utility/pageHeader';
import Box from '../../components/utility/box';
import LayoutWrapper from '../../components/utility/layoutWrapper.js';
import ContentHolder from '../../components/utility/contentHolder';
import basicStyle from '../../settings/basicStyle';
import IntlMessages from '../../components/utility/intlMessages';
import { adalApiFetch } from '../../adalConfig';

export default class extends Component {
 constructor(props) {
  super(props);
  this.state = {
    data: []
  };

 }

 fetchData = () => {
  adalApiFetch(fetch, "/values", {})
   .then(response => response.json())
   .then(responseJson => {
    if (!this.isCancelled) {
      this.setState({ data: responseJson });
    }
   })
   .catch(error => {
    console.error(error);
   });
 };

 componentDidMount(){
  this.fetchData();
 }

 render() {
  const { data } = this.state;
  const { rowStyle, colStyle, gutter } = basicStyle;
  const radioStyle = {
    display: 'block',
    height: '30px',
    lineHeight: '30px'
  };
  const plainOptions = ['Apple', 'Pear', 'Orange'];
  const options = [
   { label: 'Apple', value: 'Apple' },
   { label: 'Pear', value: 'Pear' },
   { label: 'Orange', value: 'Orange' }
  ];
  const optionsWithDisabled = [
   { label: 'Apple', value: 'Apple' },
   { label: 'Pear', value: 'Pear' },
   { label: 'Orange', value: 'Orange', disabled: false }
  ];

  return (
   <div>
    <LayoutWrapper>
     <PageHeader>{<IntlMessages id="pageTitles.TenantAdministration" />}</PageHeader>
     <Row style={rowStyle} gutter={gutter} justify="start">
      <Col md={12} sm={12} xs={24} style={colStyle}>
       <Box
        title={<IntlMessages id="pageTitles.TenantAdministration" />}
        subtitle={<IntlMessages id="pageTitles.TenantAdministration" />}
       >
        <ContentHolder>
        <ul>
          {data && data.map(item => (
            <li>{item}</li>
          ))}
        </ul>
        </ContentHolder>
       </Box>
      </Col>
     </Row>
    </LayoutWrapper>
   </div>
  );
 }
}
```

En este último ejemplo se observa cómo se puede consumir un web api que simplemente devuelve dos valores (value1, value2), pero lo interesante de esto, es que este web api está protegido con Azure AD así como la aplicación que lo consume. El token que se obtiene al autenticarse por primera vez en la aplicación ReactJS, es reutilizado al consumir el Web API, y por lo tanto no es necesario una segunda autenticación.

Este articulo sirve como referencia para la construcción de una aplicación ReactJS la cual consume toda su funcionalidad a través de un web api, para poder realizar cualquier aplicación, espero que sea de utilidad para el lector. En próximos textos se revisará como utilizar este conocimiento para utilizar funcionalidades de SharePoint Online con los paquetes de OfficePnP.

LUIS VALENCIA
Office Development MVP
www.luisevalencia.com
@levalencia

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.

33

Dynamic Data que son y como empezar a utilizarlo en nuestros desarrollos

SI echamos la vista atrás, hace ya unos 6 años que empecé a colaborar con la revista. Mi primer artículo http://www.compartimoss.com/revistas/numero-13 fue sobre como conectar mediante programación distintos WebParts existentes en una página de SharePoint. En aquel momento estábamos empezando a intentar automatizar todo lo posible los despliegues de SharePoint sin hacer ningún clic. De aquellos tiempos, los recuerdos que me quedan es que esta forma de conectar era cuanto menos peculiar como si no estuviera pensada para hacerlo mediante desarrollo, sino que solo se podían hacer por la interfaz.

Con dicha introducción está claro que en el momento en el que se anunció esta característica iba a empezar a intentar ver si ahora sí se había aprendido de los errores del pasado y teníamos una funcionalidad pensada para desarrolladores o bien iba a ser un auténtico infierno. Se que a mucha gente (sobre todo los más noveles en el desarrollo de SharePoint) quizás le sobraba. Principalmente por el motivo que al estar el desarrollo en JavaScript existen técnicas más o menos ortodoxas para poder pasar los datos de un WebPart a otro. Sin embargo, desde mi punto de vista, es un acierto esta funcionalidad ya que abre un abanico de posibilidades para nuestros desarrollos, además de facilitar la forma en la que hacemos.

Arquitectura de la solución

Los Dynamic Data se han implementado utilizando un sistema de notificaciones. Existe un emisor que es el que expone que datos se van a comunicar. Y como posibles receptores puede ser cualquier otro WebPart o Extension, desde dichos WebParts se han de subscribir al emisor y de esta forma cuando se producen cualquier cambio en el emisor este se notifica a todos sus subscriptores.

Términos clave

Emisor-> El emisor es el elemento SPFX que va a exponer sus datos para que se puedan consumir desde otros elementos. Para declarar un emisor se debe de implementar la interfaz IDynamicDataController. Esta interfaz obliga a implementar a implementar dos métodos:

- getPropertyDefinitions devuelve una matriz de tipos de datos que devuelve el origen de datos dinámicos particular.
- getPropertyValue devuelve el valor que vamos a utilizar cuando notifiquemos un cambio.

En estos dos tipos podemos definir más de una forma de devoluciones, es decir, tenemos un WebPart que tiene como datos las Empresas y los empleados de dicha empresa. Podemos indicar que dependiendo a que acción se subscriban devolvamos un dato u otro. Esto hace que podamos hacer elementos más reutilizables y no tener que crear varios Dynamic data para cada tipo de datos.

es el elemento SPFX que va a exponer sus datos para que se puedan consumir desde otros elementos

Ejemplo de definición:

```
public getPropertyDefinitions(): ReadonlyArray<IDynamicDa-
taPropertyDefinition> {
  return [
   {
    id: “text”,
    title: “Text”
   }
  ];
 }
 /**
  * Return the current value of the specified dynamic data set
  * @param propertyId ID of the dynamic data set to retrieve
the value for
  */
 public getPropertyValue(propertyId: string): IQuery {
  switch (propertyId) {
   case “text”:
    return this._selectedText;
  }

  throw new Error(“Bad property id”);
```

Ahora bien, para poder hacer que otros elementos SPFx se puedan subscribir, en el método OnInit debemos inicializar el Dynamic data agregando esta línea:

```
this.context.dynamicDataSourceManager.initializeSource(-
this);
```

Una vez, ya permitimos que otros artefactos se puedan subscribir, debemos definir cuándo vamos a notificar nuestros subscriptores de que se ha producido un cambio en los datos que exponemos. Para notificar a nuestros

subscriptores deberemos utilizar la siguiente línea:

```
this.context.dynamicDataSourceManager.notifyProperty-
Changed("text");
```

El lugar de la notificación obviamente dependerá de cada desarrollo, pero por buen uso, cada vez que la variable que hemos indicado en el GetPropertyValue se modifique, deberemos lanzar este método para que se notifique.

Receptor -> El receptor es aquel elemento que necesita de los datos de otro para empezar a realizar su trabajo. Podemos poner múltiples ejemplos de su uso: Combos en cascada, buscador, mapas, etc. Para poder subscribirnos a un Origen de datos deberemos invocar este método Los tres parámetros son:

```
this.context.dynamicDataProvider.registerPropertyChange-
d(this.properties.sourceId,  this.properties.propertyId,  this.
render);
```

- sourceId => WebPart que lo referencia.
- propertyId => Identificador del origen de datos.
- método=> Y método que se invoca cuando se actualiza los datos.

Para poder obtener el sourceId y el propertyId disponemos de varios métodos en el Framework:

```
this.context.dynamicDataProvider.getAvailableSources().ma-
p(source => {
    return {
      key: source.id,
      text: source.metadata.title
    };
  });

 const source: IDynamicDataSource = this.context.dynamic-
DataProvider.tryGetSource(selectedSource);
   if (source) {
       propertyOptions = source.getPropertyDefinitions().ma-
p(prop => {
      return {
        key: prop.id,
        text: prop.title
      };
    });
```

Para evitar tener que poner estos valores "fijos" en nuestra WebParts se recomienda hacer uso de sus propiedades para que se puedan seleccionar y de esta forma además se queden almacenadas en la propia página y sean persistentes las subscripciones.

Consideraciones para tener en cuenta

- Cada página puede tener múltiples orígenes de datos dinámicos y consumidores.
- Cada componente puede proporcionar datos dinámicos a otros componentes y consumir datos dinámicos de otros componentes.
- Los componentes pueden consumir datos de múltiples orígenes de datos dinámicos.
- Para que persistan las suscripciones a orígenes de datos dinámicos, almacene la información de suscripción en las propiedades del elemento web.

Aspectos Positivos

Una de las grandes mejoras que nos proporciona los Dynamic data es que podemos optimizar las consultas que realizamos sobre las APIS que consumimos (sean de SharePoint o no). Por ejemplo, si tenemos un WebPart que necesita una lista y esta hace falta en muchos más componentes pues optimizamos las peticiones. Otra de las ventajas es que evitamos crear un Mega WebPart o añadir alguna técnica un tanto curiosa (acceder a la cache del navegador) para poder conectar varios desarrollos.

Aspectos Negativos

Ahora mismo en las SPFx Extensions no podemos consumir datos dinámicos. El motivo es que las subscripciones del propio elemento se pierden. Aunque es bien cierto que hay pocos casos de uso en el que desde una Extensión queramos consumir algún dato de otro componente.

Conclusión

Los Dynamic Data tal y como se han implementado me parecen un gran acierto y algo necesario que estuviera incorporado en el Framework. La primera vez que supe de su incorporación, pensé para mis adentros si lo han implementado como en las WebPart clásicas, no lo van a utilizar y será una característica que caerá en desuso. Tal y como esta implementado me parece que en muchos escenarios simplifica el desarrollo y además al conectarlo hace que la carga de estos se vea de una forma natural y no con unos extraños golpes de vista (nosotros le llamamos que los componentes van saliendo como "setas").

Si queréis ver los Dynamic Data en funcionamiento en el Repositorio oficial existen dos ejemplos que os pueden valer para aplicar los explicado en dicho artículo:

- https://github.com/SharePoint/sp-dev-fx-extensions/blob/master/samples/react-application-search-dynamicdata
- https://github.com/SharePoint/sp-dev-fx-webparts/tree/master/samples/react-events-dynamicdata

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

35

Desarrollando Aplicaciones de Big Data con Cloudera en Azure

Desde el inicio de los tiempos, los seres humanos han querido organizar diferentes tipos de información de muchas maneras distintas. Desde pintando en cavernas con piedras de colores, utilizando papiro, haciendo cálculos con un ábaco, y construyendo bibliotecas de proporciones bíblicas. Conforme la humanidad ha ido avanzando, la tecnología ha ido mejorando a pasos de gigante.

Ahora hay calculadoras que valen menos que un refresco y tienen más capacidad y velocidad de cálculo que un mini ejército de matemáticos. Y ni qué decir de las computadoras, algunas iniciaron siendo del tamaño de un cuarto y ahora caben en la palma de la mano.

Y tratando de no entrar en mucho detalle para no aburrirlos y brincándome una gran parte de la historia, viajemos al punto de la historia en donde nació la inteligencia de negocios, conocida comúnmente como business intelligence o simplemente BI.

Aunque al principio no era tan sofisticado. Había aplicaciones de todo tipo, algunas complejas, algunas sencillas. Algunas tiraban tablas de texto, otras presentaban algunos gráficos. Pero lo que siempre se tenía en común era que a partir de la información que se presentaba, se tomaban decisiones informadas. Y algunas de estas herramientas utilizaban hojas de cálculo, no obstante, cuando se llegaba a cierta cantidad de datos era necesario una bases de datos.

Y las bases de datos cambiaron por completo la cantidad de información que podía ser almacenada, procesada, y clasificada!

Y hay bases de datos de todo tipo. Desde las más pequeñas como Access, algunas comerciales como Oracle y SQL Server, otras open source como MySQL y PostgreSQL. Había para todo tipo de gustos, colores, sabores, billeteras y capacidad de procesamiento. ¡Entonces llegamos a un punto de la vida en la que hacer una aplicación para inteligencia de negocios con grandes cantidades de datos era “fácil”! Los pasos eran algo como así:

1.– Se buscaba cuál era la maquina más grande que se podía comprar. Se conseguía presupuesto y se mandaba a pedir.

2.– Luego se escribía el cheque con más ceros posibles que aguantara el presupuesto para comprar la base de datos comercial más poderosa posible.

3.– Y se encerraba unos cuantos desarrolladores con mucha pizza y mucho café en un cuarto a hacer magia.

4.– “Profit”

Bueno, tal vez este cuarto paso podría variar un poco, pero esa es la idea en general. No todas las aplicaciones eran un éxito, pero aquellas que sí, daban un apoyo tremendo para tomar decisiones informadas. Pero conforme pasó el tiempo, la cantidad de datos fue creciendo de manera exponencial. Había ya aplicaciones en las cuales una sola máquina no le hacía ni cosquillas. ¿Como qué? Me imagino que te estás preguntando. Pues el mejor ejemplo es los motores de búsqueda de internet.

No sé si usted sabía, pero al inicio de los 80s, se tenía un archivo llamado hosts.txt en Stanford que tenía los IPs y los nombres de todas las máquinas conectadas a la internet de esa época. Ya para finales de los 90s e inicios del siglo 21 ya internet había crecido tanto que para indexar toda la información de todas las páginas era imposible hacerlo con una sola máquina, no importa el tamaño ni el presupuesto disponible.

cómo olvidarse de la infraestructura y enfocarse en sus aplicaciones utilizando infraestructura manejada

Y entonces quitando y poniendo algunos detalles, resumiendo y tomando un poco de libertad artística en la historia - por ahí hay unas publicaciones de Google, el trabajo hecho en Yahoo, una plataforma que se llama Nutch, y algo de Lucene entre otros detalles - pero llegamos al punto importante de la historia, el nacimiento de una de las plataformas de procesamiento paralelo más importante que han existido: Hadoop.

Hadoop cambió la manera en la que se procesaban datos para siempre. Ya no era necesario comprar servidores de millones de dólares. ¡Ahora era posible comprar servidores más estándares, conocidos principalmente como “commodity hardware”, pero comprar muchos de ellos y ponerlos a trabajar juntos! Y, además, como el proyecto fue donado

de parte de Yahoo a Apache, al ser open source, ya no era necesario pagar por licencias con costos astronómicos.

Al principio se trabajaba creando aplicaciones en Java, con MapReduce. Pero esto fue evolucionando con el tiempo.

Más allá de una sola plataforma, se fue creando un ecosistema de plataformas para proveer a los distintos tipos de desarrolladores con herramientas que se ajusten a su caso de uso. De ahí nacieron o se refinaron Pig, Hive, HBase, Solr y más. Pero había un detalle, todas estas plataformas son complejas. Llevan muchas configuraciones que deben de tener los valores correctos para poder trabajar en conjunto. Y aún más allá, se hay ciertas configuraciones que tienen ciertos valores óptimos que no son fáciles "adivinar".

Y en ese momento, llegó una empresa a cambiar la manera en la que se trabajaba con Hadoop. En resumen, su misión fue la de democratizar el mundo de Hadoop. ¿Y cómo lo lograron? Pues crearon la primera distribución de Hadoop, todavía open source, en la cual ellos se encargaban de hacer todas las configuraciones adecuadas y uno como usuario podía enfocarse en utilizar Hadoop, no en instalar cada plataforma por separado y ver cómo hacer para que trabajaran las plataformas en conjunto.

El nombre era CDH, que significa Cloudera Distribution including Hadoop.

Otra cosa que caracteriza a Cloudera es su compromiso con el open source. Cloudera ha contribuido a través de su historia dedicando una gran cantidad de recursos a mejorar las plataformas actuales y crear nuevas. Además, muchas de las mentes más brillantes del mundo de la programación trabajan o han trabajado en Cloudera. Muchos de ellos han sido contratados para crear mejores a plataformas como Hadoop o Solr, y su trabajo luego se incluye en el código open source de Apache. Entre ellos podemos con el hombre que creó Hadoop, Mr. Doug Cutting. Tengo que confesarles que puedo ver venir por la calle a Cristiano Ronaldo, Shakira, David Bisbal o alguna Kardashian y ni me pasar por la mente ir a saludarlos o pedirles un autógrafo.

¡Pero a don Doug Cutting... hay que saludarlo!

Imagen 1.- Doug Cutting.

Volvamos a Big Data. Crear un cluster (un grupo de máquinas que trabajan en conjunto) de Hadoop ahora es mucho más sencillo. Les voy a contar como se puede hacer de unas cuantas maneras. Veamos que sencillo. Una nota es que los pasos que voy a detallar ahora están explicados a detalle en un entrenamiento gratis que desarrollo Cloudera a petición de Microsoft, el cual pueden ver en el siguiente link:

- Deploying and Scaling Cloudera Enterprise on Microsoft Azure:

(https://ondemand.cloudera.com/courses/course-v1:Cloudera+Azure+180601/info)

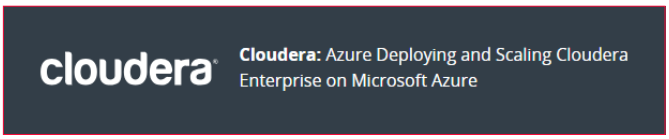


Imagen 2.- Azure Deploying and Scaling Cloudera Enterprise on Microsoft Azure.

Si le interesa aprender cómo instalar on-premises en vez de Azure o incluso en AWS, tengo algunos cursos disponibles en Pluralsight que talvez le puedan interesar: https://www.pluralsight.com/search?q=xavier+cloudera&categories=all

Pero bueno, vamos a lo importante. Ahora sí, veamos cuáles son las maneras en las que se puede instalar un cluster, y que tipo de clusters hay.

#1 Instalación desde Cero Utilizando Cloudera Manager (Para Desarrollo)

Esta es la manera más fácil para instalar un cluster de desarrollo. Digo de desarrollo pues es una instalación automatizada, que utiliza una base de datos "embedded", es decir que trae como parte de la instalación.

Lo primero que ocupamos es una máquina Linux. La recomendación es CentOS y acá hay un link con las instrucciones de cómo crear una máquina virtual en Microsoft Azure con dichas características desde ekl portal de Azure (portal.azure.com): https://docs.microsoft.com/en-us/azure/virtual-machines/linux/quick-create-portal (el articulo indica Ubuntu, pero como les comente, les recomiendo utilizar CentOS).

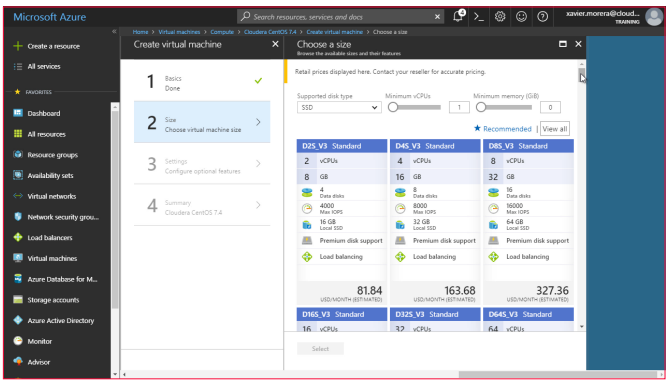


Imagen 3.- Selección del tipo de máquina virtual a crear.

Una vez que la máquina está corriendo y se han conectado

vía SSH, vamos a proceder a seguir las instrucciones para instalar Cloudera, con lo cual podemos después proceder a crear un cluster. Vamos a seguir estas instrucciones:

https://www.cloudera.com/documentation/enterprise/latest/topics/cm_ig_non_production.html

Siendo tres los comandos que nos interesan:

```
$ wget https://archive.cloudera.com/cm5/installer/5.15.0/cloudera-manager-installer.bin
$ chmod u+x cloudera-manager-installer.bin
$ sudo ./cloudera-manager-installer.bin
```

todos los pasos anteriores hacen la vida dentro del mundo de Big Data más fácil

En este punto la instalación automatizada inicia, esto es lo que se llama Path A Install. Cuando ya está instalado Cloudera Manager, inicia el proceso de "bootstrap", en el cual se utiliza este servidor para instalar agentes en otras máquinas.

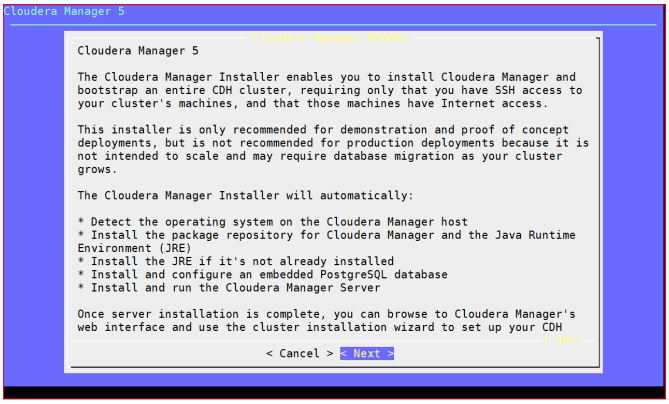


Imagen 4.- Asistente para la instalación de Cloudera.

Se sigue el "wizard" de instalación, y se seleccionan cuáles son los servicios deseados:

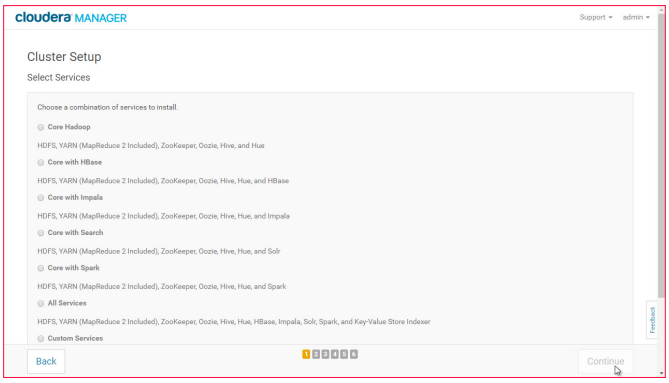


Imagen 5.- Setup de instalación de Cloudera.

Y finalmente tendremos un cluster donde podemos ya usar Hive, Impala, Spark, Hue, Oozie, MapReduce 2, Solr, o cualquiera de los otros servicios disponibles.

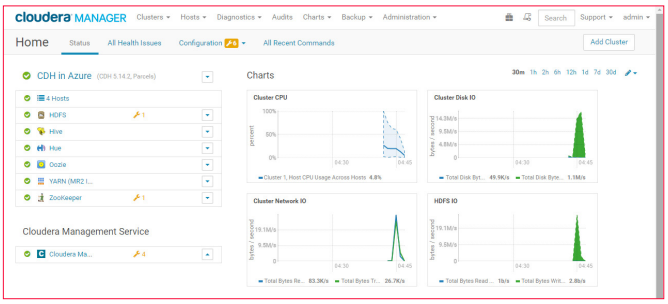


Imagen 6.- Cloudera Manager.

Y así de fácil, logramos instalar un cluster de desarrollo en Microsoft Azure. ¿Y qué tal si quisiéramos un cluster ya de producción? Los pasos están bien documentados, aunque son un poco más largos. Esto es lo que se llama un Path B Install, y la principal diferencia con el paso anterior es que es necesario crear una base de datos que puede ser MySQL, MariaDB, PostgreSQL, u Oracle. Estos son los pasos por si les interesa conocer más a fondo: https://www.cloudera.com/documentation/enterprise/latest/topics/installation.html.

No obstante, hay una forma todavía más sencilla de instalar un cluster de producción, el Enterprise Data Hub, en Microsoft Azure. Esta metodología utiliza Azure Resource Manager para crear un cluster a base de una plantilla. Acá está el link por si desean hacer la instalación ustedes mismos: https://azuremarketplace.microsoft.com/en-us/marketplace/apps/cloudera.clouderaedh

Pero, en resumen, los pasos son sencillos. Se carga la plantilla de Azure Resource Manager, se especifica los detalles de las máquinas a utilizar, se le da un nombre a un cluster, se seleccionan los servicios y al finalizar de llenar la información, se espera aproximadamente 30 minutos para tener un cluster de Hadoop en la nube.

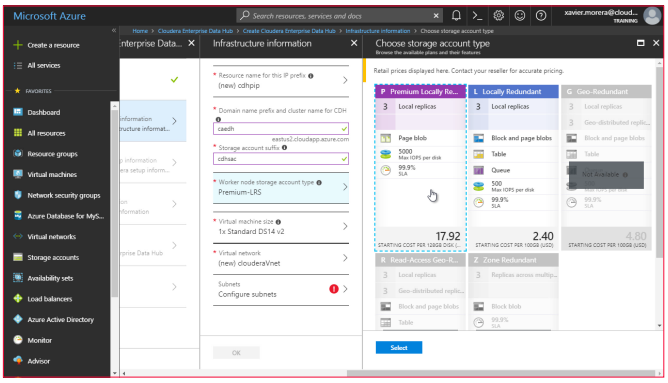


Imagen 7.- Despliegue del Enterprise Data Hub.

Y luego, se puede ir uno a tomar un café, ¡y al regresar tiene un cluster corriendo!

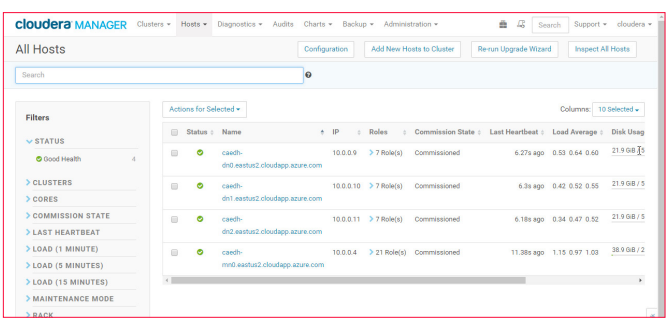


Imagen 8.- Cluster operativo.

Y todos los pasos anteriores hacen la vida dentro del mundo de Big Data más fácil, un cluster a la vez. Qué tal si uno lo que necesita es tener una forma fácil de crear clusters bajo demanda. Es decir, cada vez que un equipo o departamento ocupa un cluster, ojalá fuera tan fácil como apretar un botón y tener un cluster nuevo. Pues eso, damas y caballeros, es lo que se llama Cloudera Director. Con Cloudera Director se automatiza la creación de clusters en Microsoft Azure. Se configura las credenciales apropiadas, se crean unas plantillas para especificar el tipo de instancia que se desea utilizar y con solo apretar un par de botones, un cafecito, y luego ya se tiene uno o varios clusters.

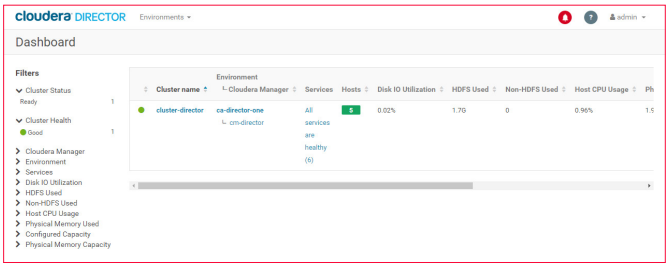


Imagen 9.- Cloudera Director.

Y aunque es fácil crear (y por ende) destruir un cluster, o incluso muchos clusters, generalmente es necesario dar el mantenimiento necesario para que funcione de manera apropiada. Acá es generalmente donde se necesita personal de IT que instale actualizaciones, monitoree espacio disponible, y en general se encargue de todas las tareas normales cuando se administran servidores.

Pero hay un detalle que tenemos que tomar en cuenta. Si uno ve una gran cantidad de los servicios disponibles ahora, la gran mayoría ofrecen software-as-a-service, lo cual es fenomenal. Es un concepto tan fácil de entender utilizando la analogía de alquilar un carro. Uno lo reserva para el periodo que lo necesita, lo utiliza y luego lo devuelve. No tiene que preocuparse del mantenimiento del mismo ni de otros aspectos mundanos.

Y es así como Cloudera cambió la forma en la que se trabaja con Big Data, al llegar a la era de Big Data como platform-as-a-service, ¡una era de managed Big Data! Y esta plataforma se llama Cloudera Altus.

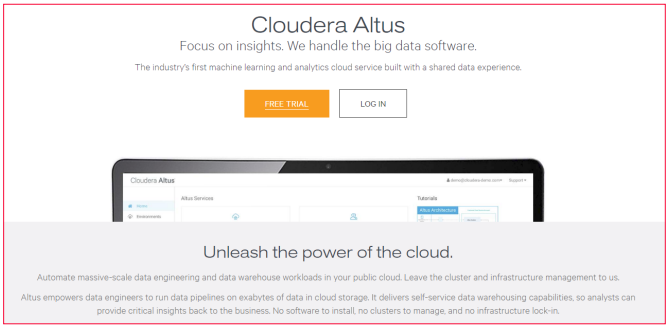


Imagen 10.- Cloudera Altus.

Con Altus, el foco de una aplicación de Big Data es el "job", es decir la ejecución de una aplicación en un cluster. Los "jobs" pasaron a ser el ente principal de una aplicación. Ya no es necesario darle mantenimiento a un cluster. Simplemente se crean para ejecutar código y se desechan al terminar o incluso se especifica que un cluster nuevo tiene que crearse durante la duración de la ejecución del código.

un concepto tan fácil de entender utilizando la analogía de alquilar un carro

Lo primero que se necesita con Altus es tener un ambiente, o un "environment". Dentro de un environment especificamos las credenciales necesarias para conectarse a Microsoft Azure y poder hacer instalaciones de clusters. Por dicha hay un "wizard" que es fácil de seguir.

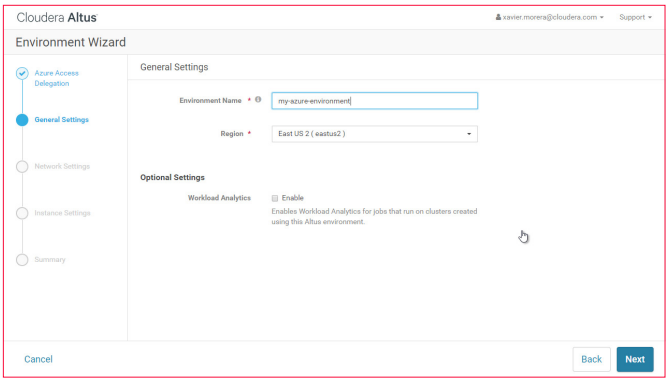


Imagen 11.- Asistente para.

Una vez que se tiene un "environment", y verificamos que tenemos las credenciales apropiadas, podemos crear un cluster. ¡El proceso de iniciar la creación de un cluster dura menos de 1 minuto! Solo se ocupa proveer un nombre, el tamaño, cual es el motor de ejecución - se puede seleccionar MapReduce v2/Spark/Hive/entre otros - y una información de seguridad. Después de eso, nada más es cuestión de dejarlo correr y al final tendrá un cluster de Hadoop del tamaño deseado. Pueden ser 3 nodos o 10 o 50. No importa cuantos sean, eso es solo un parámetro que uno especifica. No obstante, acuérdense de tener cuidado pues esos nodos cuestan dinero al correr en la nube.

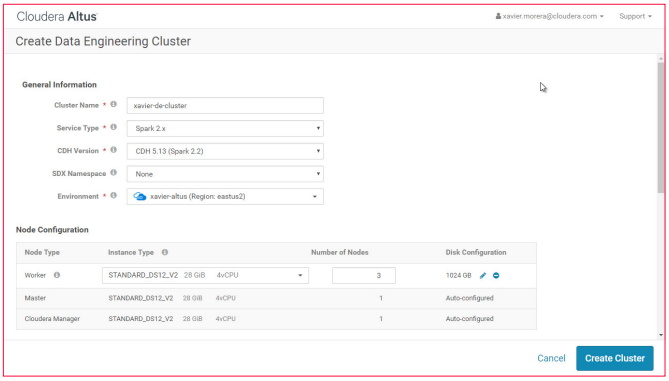


Imagen 12- Cluster a crear.

Ya con un cluster arriba podemos pasar a la siguiente etapa, le de ejecutar una aplicación. Ahora podemos enviar un "job" a ser ejecutado. Este es el "secreto" de Altus, su valor agregado, el motivo principal por el que utilizamos Altus. El cluster está corriendo, pero no nos tenemos que preocupar de que este bien configurado. Los expertos de Cloudera configuran el cluster de acuerdo con su experiencia en el

campo, con los parámetros más convenientes para el tipo de código a ejecutar, por ejemplo, Apache Spark.

Y este paso es muy sencillo, se indica cual es el código a ejecutar, se especifica cualquier dependencia, y en cual cluster se va a ejecutar. Mejor aún, si no tiene un cluster corriendo, se puede especificar que el "job" corre sobre un cluster que se levanta exclusivamente para la ejecución y al terminar se destruye. Así se paga únicamente por el tiempo en el que la aplicación corre.

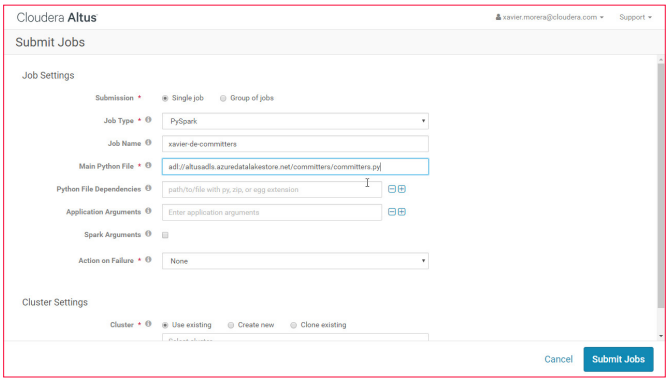


Imagen 13.- Job de Altus

Y hablando de aplicaciones, en este caso estoy ejecutando código Python. Pero también podría ser Scala, o MapReduce, o Hive, o, en fin, cualquiera de los motores de ejecución disponibles. Y me imagino que se están preguntando, si estoy levantando un cluster, utilizándolo y destruyéndolo, ¿Dónde estoy guardando mi información? Bueno, la respuesta es muy fácil. Hay muchas opciones, pero en este caso en particular estoy utilizando Azure Data Lake Store, HDFS en la nube.

Conclusión

Y esas son damas y caballeros, las múltiples opciones que hay para correr Hadoop con Cloudera, ya sea on-premises o en la nube. Todas son opciones útiles y funcionales, no obstante, la posibilidad de utilizar Big Data manejado en la nube, es decir platform-as-a-service, nos da la opción de olvidarnos por completo de la parte de administración y enfocarnos exclusivamente en el código a ejecutar y en nuestros datos.

¡Para mi eso es innovación y de la buena!

XAVIER MORERA
xavier@familiamorera.com
@xmorera
http://www.xaviermorera.com
*https://app.pluralsight.com/profile/author/*xavier-morera

40

Autenticando Xamarin Forms con Azure AD

Nuestras aplicaciones móviles para conseguir los datos a mostrar llaman a las APIs que hayamos desarrollado para obtenerlas. Estas APIs están aseguradas para que no todo el mundo pueda consumirlas, sino solo aquellas aplicaciones que tengan permiso para hacerlo.

Por otro lado, para autenticar los usuarios en nuestra aplicación móvil lo más normal es usar un sistema de Identidad que nos permita implementar la autenticación mediante OAuth o OpenId Connect, y que nos permita obtener el token correspondiente para poder enviar a nuestras APIs.

Un sistema de identidad fácil de usar y bastante extendido es Azure AD, que nos permite realizar todas las operaciones anteriormente mencionadas y muchas otras (https://docs.microsoft.com/es-es/azure/active-directory/). Vamos a ver cómo podemos autenticar mediante OAuth una aplicación Xamarin Forms, solo en versión Android, y que llame a una API alojada en una WebApp de Azure asegurada mediante Azure AD.

Creando las aplicaciones en Azure AD

Primero deberemos crear dos aplicaciones en Azure AD que serán las que provean la identidad a nuestra aplicación móvil y a nuestra API REST. Empezaremos creando la aplicación para nuestra aplicación móvil.

Para ello iremos a Azure Active Directory – App Registrations y le daremos a New Application Registration.

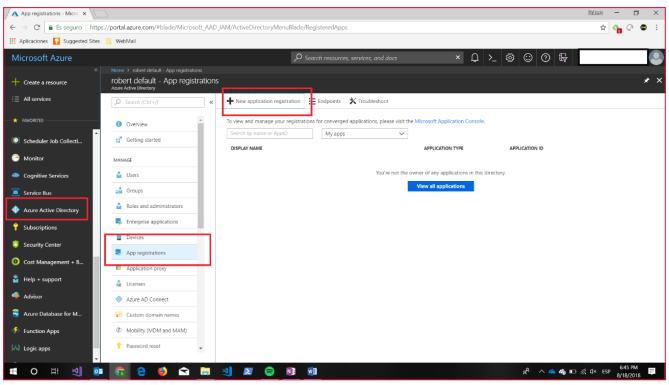


Imagen 1.- Creación de la aplicación en Azure AD.

Nos aparecerá la pantalla para añadir la información de nuestra aplicación:

- Name > Nombre de nuestra aplicación.

Nuestras aplicaciones móviles para conseguir los datos a mostrar llaman a las APIs que hayamos desarrollado para obtenerlas

- Application Type > Combo con dos opciones. En este caso seleccionaremos la opción de Native.
- Redirect URI > Dirección de retorno, en nuestro caso pondremos una url cualquiera con formato válido.

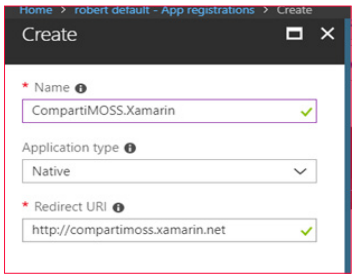


Imagen 2.- Parámetros de la aplicación.

Ahora nos guardaremos el Application ID que lo necesitaremos más adelante.

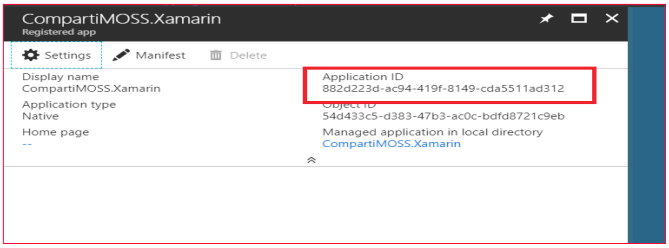


Imagen 3.- Application ID.

Ahora vamos a crear la aplicación para la API REST. Seguimos los mismos pasos que en el caso anterior para crearla. Y en el punto de añadir la información en la opción de Application Type seleccionamos la opción WebApp / WebAppi y en la rediret URI ponemos la URL donde alojaremos nuestra API Rest

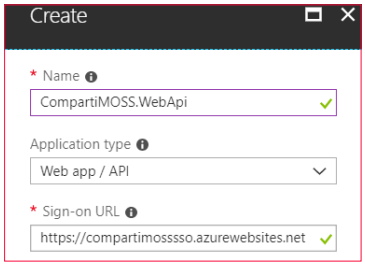


Imagen 4.- Creación de la API REST en Azure AD.

Una vez creado nos guardaremos el Application ID como anteriormente, y además crearemos una key que será la secret key para la Web API.

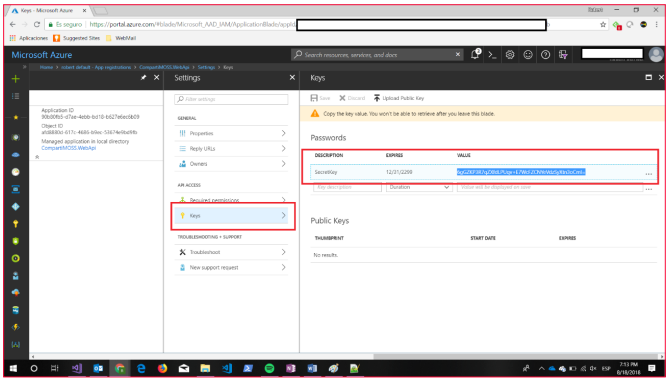


Imagen 5.- Creación del secret key.

Estos valores nos servirán para configurar la autenticación de nuestra API REST. Ahora le daremos permiso a Read Directory Data, para ello le damos a la opción Required Permisions, seleccionamos Windows.Azure.Active.Directory y en Delegated Permissions seleccionamos Read Directory Data.

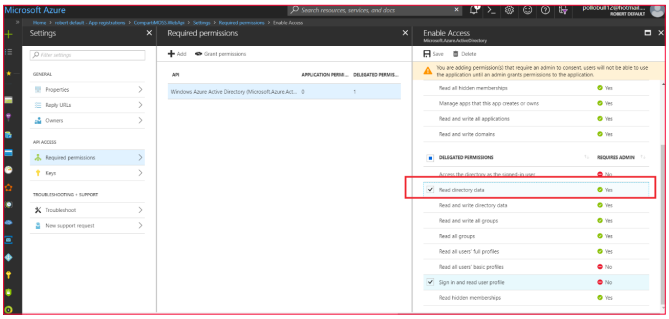


Imagen 6.- Configuración de los permisos para la API REST.

Después de ello le daremos Grant Permissions a la aplicación.

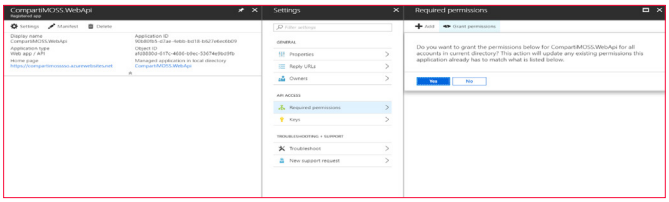


Imagen 7.- Concediendo permisos a la API REST.

el punto de añadir la información en la opción de Application Type seleccionamos la opción WebApp

Ahora lo que debemos hacer es dar permisos a las aplicaciones la una a la otra para que tengan acceso entre ellas. Primero le daremos permisos de la aplicación de la API a la aplicación Mobile. Para ello, iremos a la aplicación que hemos creado anteriormente CompartiMOSS.Xamarin, seleccionaremos Settings y le daremos a la opción Required Permissions

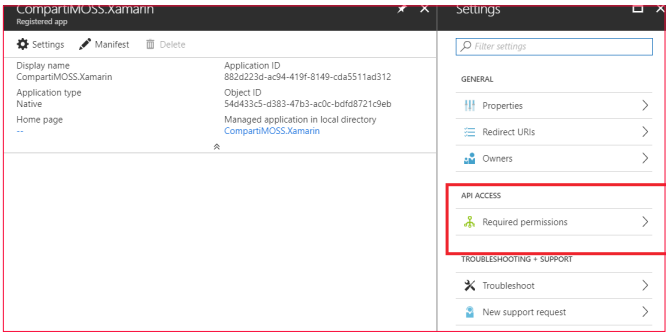


Imagen 8.- Concesión de permisos entre aplicaciones.

Le daremos a Add, buscaremos nuestra aplicación API, la seleccionamos y le daremos los permisos.

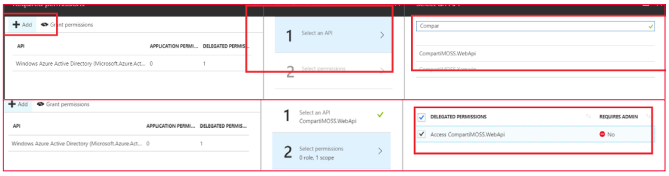


Imagen 9.- Concesión de permisos.

Ahora haremos lo miso con la aplicación de la API, pero el procedimiento es diferente. Buscaremos la aplicación y le daremos a Manifest, y en la sección knownClientApplications. Añadiremos el Application Id de la aplicación Mobile.

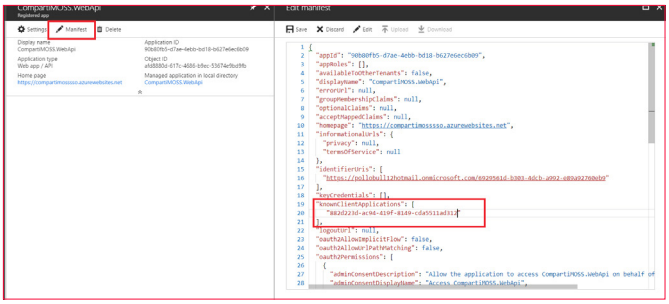


Imagen 10.- Configuración del manifestó de la aplicación.

Una vez esto hecho, solo nos faltará conocer un parámetro para nuestra configuración, el Id de nuestro tenant de Azure AD. Para ello en la opción de Azure Active Directory, seleccionamos Properties y nos apuntamos el Directory Id

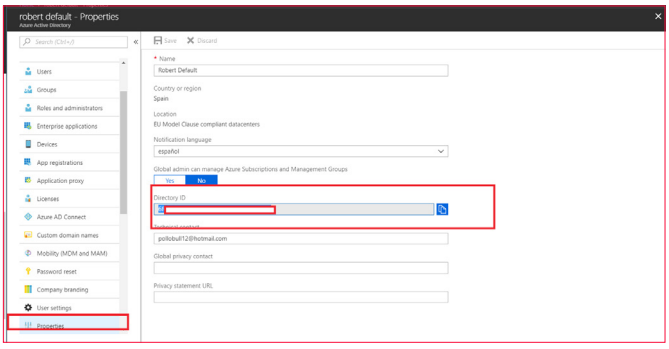


Imagen 11.- Propiedad Directory ID.

Añadiendo autenticación en Xamarin.Forms

Primero crearemos una aplicación Xamarin.Forms (shared Project), para este ejemplo seleccionamos únicamente Droid, ya que solo lo veremos para Android.

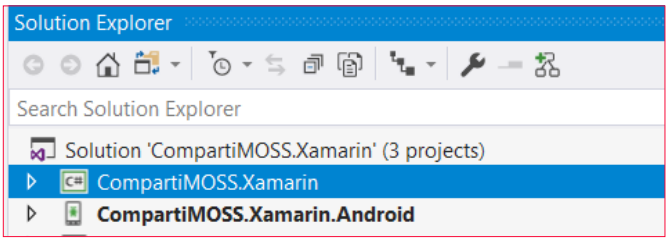


Imagen 12.- Proyectos Xamarin.Forms.

En la imagen anterior vemos la estructura de proyectos que se nos crea, ahora debemos añadir el siguiente paquete Nugget en ambos proyectos:

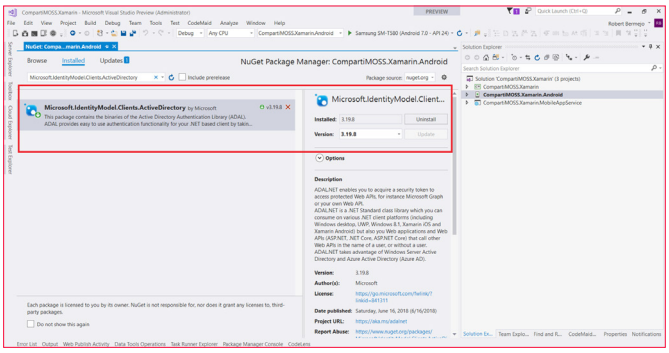


Imagen 13.- Paquetes Nugget a añadir.

En el proyecto Android, en la clase MainActivity añadiremos la sobreescritura del método OnActivityResult:

```
protected override void OnActivityResult(int requestCode, Result resultCode, Intent data)
{
      base.OnActivityResult(requestCode, resultCode, data);
        AuthenticationAgentContinuationHelper.SetAuthenticationAgentContinuationEventArgs(requestCode,    resultCode, data);
}
```

En el proyecto Shared, en este ejemplo Xamarin, crearemos la interface que especificará el los métodos para realizar la autenticación en este ejemplo:

```
public interface IADALAuthenticator
{
    Task<ADToken> AuthenticationAsync();
}
```

Ahora en el proyecto de Android, implementaremos la inteface anterior:

```
public class ADALAuthenticator : IADALAuthenticator
  {
    private const string TenantUrl = "https://login.microsoftonline.com/common";
       public static string ADClientId = "882d223d-ac94-419f-8149-cda5511ad312";
     public static string tenant = "xxxx-8106-48a2-a786-xxx";
       public static Uri returnUrId = new Uri("http://compartimoss.xamarin.net");
      public static string WebApiADClientId = "90b80fb5-d7ae-4ebb-bd18-b627e6ec6b09";

     public async Task<ADToken> AuthenticationAsync()
     {
       try
       {
          var platformParams = new PlatformParameters(CrossCurrentActivity.Current.Activity);
           var authContext = new AuthenticationContext(TenantUrl);

          if (authContext.TokenCache.ReadItems().Any())
          {
             authContext = new AuthenticationContext(authContext.TokenCache.ReadItems().FirstOrDefault().Authority);
          }

           var authResult = await authContext.AcquireTokenAsync(WebApiADClientId, ADClientId, returnUrId, platformParams);

           return new ADToken()
           {
             AccessToken = authResult.AccessToken,
             TokenType = authResult.AccessTokenType,
             Expires = authResult.ExpiresOn.Ticks,
             UserName = authResult.UserInfo.DisplayableId
           };
         }
         catch(Exception ex)
         {
           throw ex;
         }
      }
   }
```

Del código anterior, lo importante es que en el método AcquireTokenAsync el primer parámetro que es el resource donde haremos el login, ponemos el Application Id de la WebApi, de esta forma obtendremos un token válido para poder enviarlo en las peticiones. Ahora lo único que nos queda es en nuestra "page" llamar a esté método.

para este ejemplo seleccionamos únicamente Droid, ya que solo lo veremos para Android

En este ejemplo, nuestra página principal se llama XamarinAdalPage, donde hay un botón que al hacer clic llama al método de autenticación mostrando el login de Azure AD

```
public partial class XamarinAdalPage : ContentPage
{
     public XamarinAdalPage()
     {
       InitializeComponent();
     }

    async void Login_Clicked(object sender, System.EventArgs e)
     {
       await LoginAsync();
     }
     private async Task LoginAsync()
     {
 var response = await DependencyService.Get<IADALAuthenticator>().AuthenticationAsync();

       if (response != null)
       {
App.BearerToken = response.AccessToken;
          await Navigation.PushModalAsync(new MainPage());
       }
     }

}
```

Una vez se ha realizado la autenticación ya hemos obtenido el Bearer Token y ya lo podemos enviar en nuestras llamadas a la API.

```
   HttpClient client = new HttpClient();
   client.BaseAddress = new Uri($"{App.AzureBackendUrl}/");
   client.DefaultRequestHeaders.Authorization = new  AuthenticationHeaderValue("Bearer", App.BearerToken);
```

Ahora lo único que nos queda por hacer es autenticar la API que hayamos creado con Azure AD, para ello hacemos botón derecho en Connected Services y seleccionamos Authentication with Azure Active Directory.

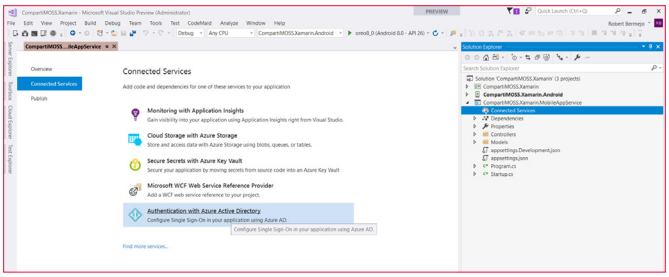


Imagen 14.- Configuración de la autenticación con Azure AD.

todo esto lo único que queda es desplegar la API en Azure y llamarla desde la aplicación Xamarin

Seleccionamos la segunda opción:

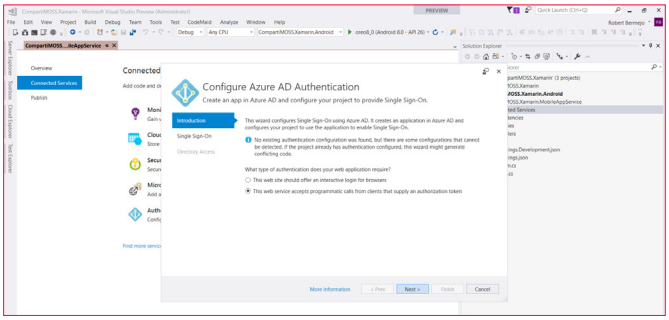


Imagen 15.- Configuración de la segunda opción.

Seleccionamos la opción de usar settings ya existentes.

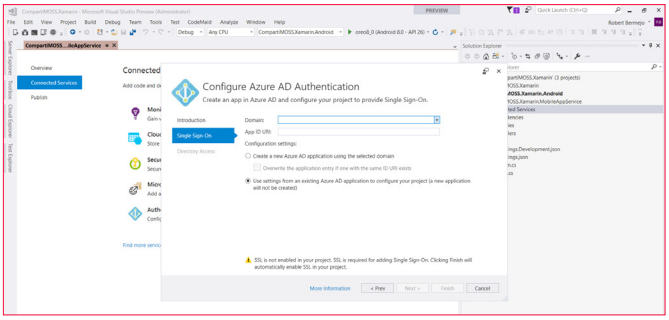


Imagen 16.- Opción de app settings seleccionada.

A partir de este punto lo único que se debe hacer es añadir toda la información que anteriormente hemos ido apuntado. Si mediante el transcurso de esta operación os diera algún error, podéis realizar todo el proceso creando una aplicación nueva, y acto seguido en appSettings cambiar la configuración por la que hemos creado al principio.

Con todo esto lo único que queda es desplegar la API en Azure y llamarla desde la aplicación Xamarin de forma segura.

Podéis ver todo el código completo en: https://github.com/bermejoblasco/CompartiMOSSXamarinAAD

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

44

Instalación de SCOM 2016 para la monitorización de nuestra infraestructura (Parte 2)

Introducción

Tras haber visto que es System Center Operations Manager (SCOM) y que ventajas nos aporta su instalación, así como haber tratado brevemente los requisitos que tenemos que instalar previamente para poder desplegar esta solución de monitorización en nuestra infraestructura, en este pequeño artículo vamos a hablar de algunos temas peliagudos que debemos solventar y tener en cuenta para que no nos de problemas la instalación de la solución en sí.

En concreto, en este artículo vamos a tratar la configuración de Reporting Services, o mejor dicho, la revisión de la configuración de Reporting Services que habremos hecho previamente durante la instalación del SQL Server que usa SCOM.

Es importante que comprobemos esto porque RS es uno de los principales quebraderos de cabeza de los administradores de SCOM cuando están en fase de despliegue.

Sin ir más lejos, hace unos años, cuando salió la versión SCOM 2012 existía un pequeño bug que hacía que Reporting Services se comportara de manera anómala en caso de darse una serie de factores en concreto. Aunque el bug fue reparado por Microsoft, muchas organizaciones siguieron teniendo problemas por no haber actualizado sus sistemas debidamente.

Premisas

En este caso suponemos que los requisitos del sistema han sido instalados ya que se vieron en la revista número 34 y que hemos instalado dentro de esos requisitos el Reporting Services de SQL. De todas maneras, estas configuraciones pueden ser tenidas en cuenta durante el proceso de instalación de SQL Server.

Nota:

En este caso vamos a dividir el artículo en las distintas secciones que tiene el wizard de configuración de Reporting Services.

Acceso a la configuración de Reporting Services

Para comprobar que la configuración de Reporting Services de SCOM o reconfigurar aquellas partes que sea necesario modificar, podemos acceder a través del administrador de configuración de Reporting Services. Para ello vamos a todos los programas y buscamos “Microsoft SQL Server 2016 – Administrador de configuración de Reporting Services”.

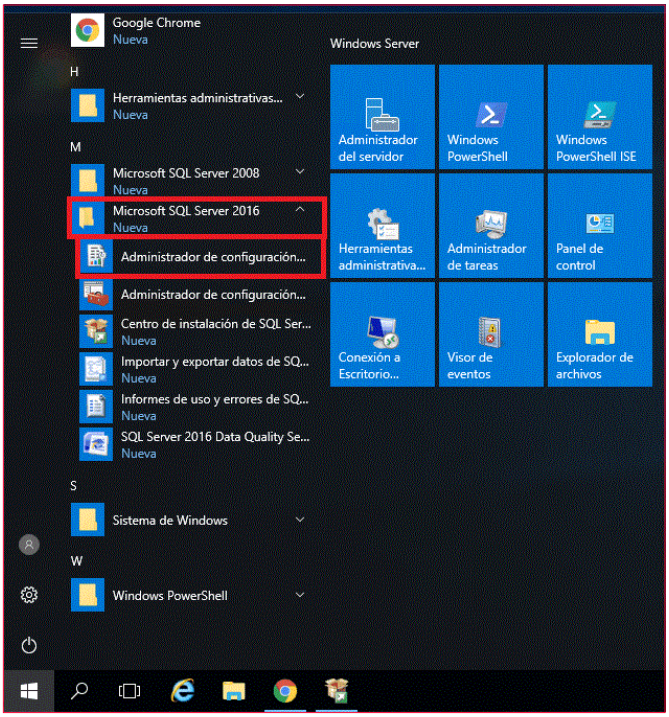


Imagen 1.- Acceso a la Administración de Configuración de Reporting Services.

Para acceder a la aplicación de configuración de SQL Server 2016 Reporting Services debemos seleccionar el nombre del servidor y el nombre de la instancia que usa el servidor de informes. A continuación, haremos clic en “Conectar” y tras un breve momento estaremos conectados a la instancia de RS del servidor para poder comprobarla y reconfigurarla.

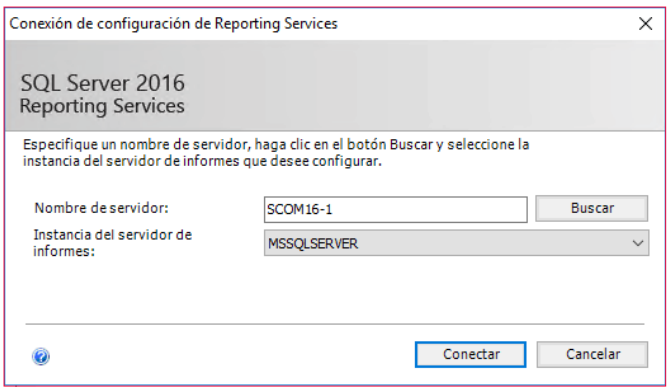


Imagen 2.- Selección de la Instancia de Reporting Services.

como haber tratado brevemente los requisitos que tenemos que instalar previamente para poder desplegar esta solución

En cuanto nos validamos en el configurador de SQL Server 2016 Reporting Services deberemos recorrer distintos apartados para comprobarlos y configurarlos de la manera apropiada para que puedan dar ser vicio a SCOM. De entrada, aterrizamos en una ventana de bienvenida donde se muestra el estado del servidor y además se muestran datos del producto, informando de la versión y la edición. También obtenemos el nombre de la instancia, el nombre de la base de datos, el modo en que se está ejecutando el servicio y el estado del mismo. Por último, desde esta ventana podemos gestionar el estado del servicio; arrancándolo o deteniéndolo. Esto puede ser recomendable tras los cambios que realizaremos a continuación.

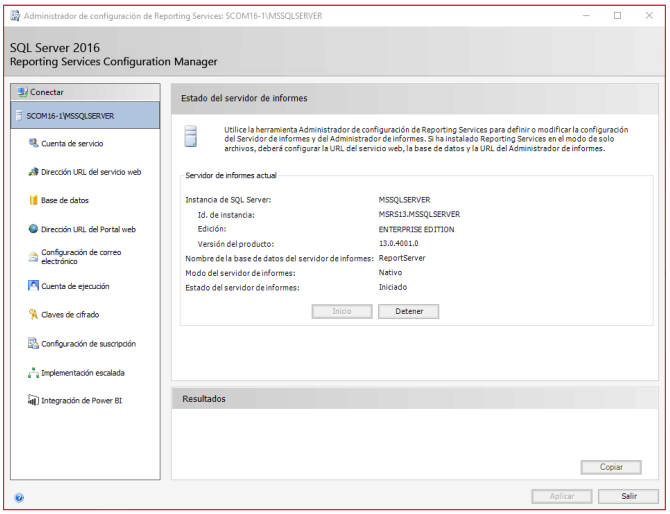


Imagen 3.- Estado del Servidor de Informes.

Configuración de cuenta de usuario

En la sección “Cuenta de servicio” que podemos seleccionar a través del menú de parte izquierda de la ventana, podemos especificar la cuenta usada para ejecutar el servicio del servidor de informes de Reporting Services para SQL Server. Disponemos de dos opciones que deben satisfacer cualquier modelo de RS que ejecutemos, fuera cual fuere su propósito.

Las opciones disponibles son:

- Usar cuenta integrada, que a su vez nos permitirá seleccionar una de las siguientes opciones: Sistema local, servicio de red, cuenta de servicio virtual.
- Usar otra cuenta: Para la que tendremos que dar una cuenta de dominio con el formato “dominio\cuenta” junto con su contraseña.

En nuestro caso dejamos la cuenta integrada que es la opción por defecto.

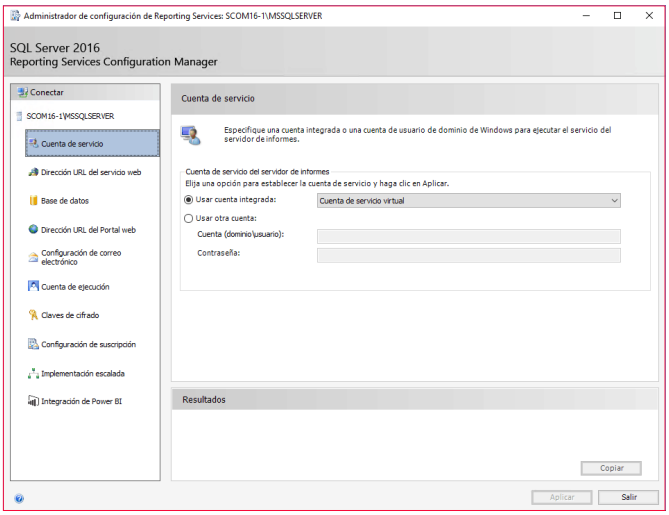


Imagen 4.- Configuración de cuentas de usuario.

Configuración de web de Reporting Services

Después de esto, nos desplazamos a la sección “Dirección URL del Servicio Web” situada en la parte lateral izquierda de SQL Server 2016 Reporting Services Configuration Manager. Dentro de esta sección cambiamos “Directorio virtual” y podemos ver que conforme lo cambiamos se cambia la dirección URL. Lo podremos configurar para que se adapte a las necesidades de nuestra infraestructura, pudiendo cambiar el puerto, o asignando un certificado para que se pueda acceder de manera segura.

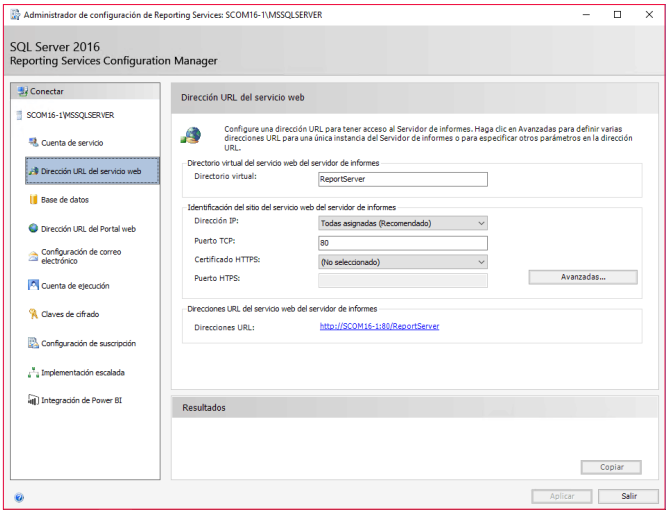


Imagen 5.- Configuración del Servicio Web de Reporting Services.

Básicamente, en este apartado podemos configurar distintas opciones:

- Dirección IP: En este apartado configuraremos la IP a través de la que dará servicio Reporting Services. Por defecto está configurado como “Todas asignadas (Recomendado)”. Después tendremos que tener en cuenta esto de cara a las configuraciones de los Firewall de nuestra infraestructura.
- Puerto TCP: Al igual que hemos configurado la IP, debemos configurar el puerto a través del que da servicio el servicio web. En nuestra infraestructura, en vez de trabajar con el puerto 80 solemos trabajar con el 8080.

- Certificado HTTPS: Podemos seleccionar el trabajar con puertos seguros, pero en nuestro caso, para empezar y ver que todo está correcto solemos trabajar con HTTP para evitar una capa de problemas, pasando a HTTPS posteriormente.
- Puerto HTPS: No lo utilizamos en esta primera fase, aunque lo cambiaremos posteriormente. Hay que recordar que ésta es una capa más que necesitará de certificados y configuraciones adicionales.

Si hacemos modificaciones y con el objetivo de que el cambio sea efectivo deberemos hacer clic en “Aplicar”.

Configuración de Base de Datos

Una vez terminada la configuración de la dirección de servicio pasamos a revisar la configuración de la base de datos de Reporting Services y si es necesario a cambiar algunos parámetros. La “base de datos del servidor de informes actual” tiene tres propiedades básicas, igual que cualquier BBDD de RS y la identifican de manera unívoca. Las opciones son:

- Nombre de SQL Server.
- Nombre de la base de datos.
- Modo del servidor de informes.

En este caso, todas estas opciones pueden ser modificadas mediante los botones “Cambiar base de datos”. En la sección de “credencial actual de la base de datos del servidor de informes” tenemos los siguientes puntos a revisar, que deberemos tener claros de la parte en la que planificamos la configuración de SCOM y que hablamos en el artículo anterior. En caso de ser necesario deberemos cambiarlo, y en el proceso de cambio se producirá un test destinado a evitar malos funcionamientos posteriores. Las opciones de configuración son sencillas:

- Credencial.
- Inicio de sesión.
- Contraseña.

Para realizar el proceso de cambio bastará con hacer clic en “Cambiar credenciales” y seguir las indicaciones.

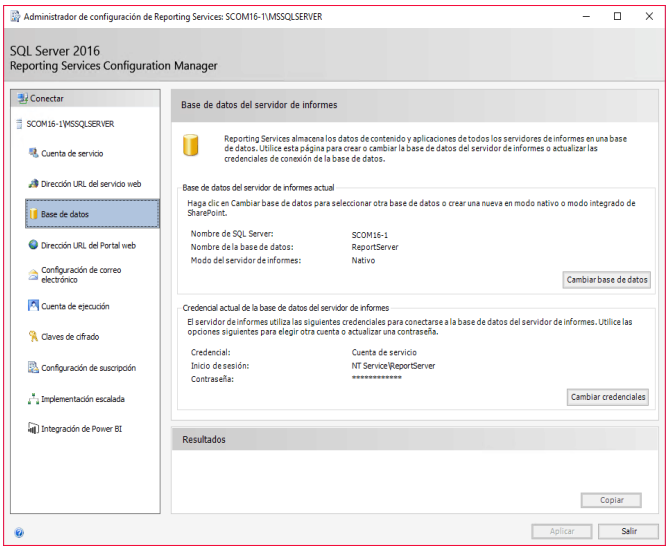


Imagen 7- Configuración de la BD de Reporting Services.

Configuración de portal de Reporting Services

Seguimos con las configuraciones dentro de SQL Server 2016 Reporting Services Configuration Manager y en este caso accedemos a la sección “Dirección URL del Portal Web”, donde queremos revisar y si es necesario cambiar el “Directorio virtual”. En este caso, conforme lo vamos modificando, podremos ver como a su mismo tiempo va cambiando la dirección URL a la que hace referencia. Este cambio será efectivo tras hacer clic en “Aplicar”. Esta parte es donde se realizan las consultas y se muestran los informes de RS y deberemos tenerla en cuenta también de cara a las configuraciones del cortafuegos cuando queramos que se publique de cara a las distintas redes de nuestra organización. Dentro de las opciones avanzadas podemos encontrar configuraciones adicionales sobre puertos, certificados, etc.

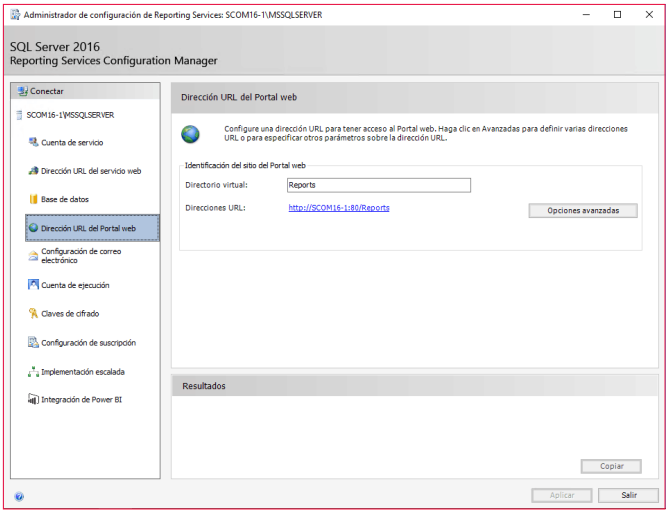


Imagen 8.- Configuración del Sitio Web de Reporting Services.

Configuración de correo electrónico

Ahora pasamos a la “Configuración de correo electrónico” que nos permitirá configurar una cuenta de comunicación de cara a informaciones varias de RS. En esta sección, como no podía ser de otra manera trataremos de configurar la cuenta de correo SMTP. Para llevar a cabo esta tarea tenemos que rellenar los siguientes campos:

- Dirección del remitente.
- Método de entrega SMTP actual.
- Servidor SMTP.
- Autenticación.

No hace falta mucha más explicación ya que son campos de sobra conocidos por todos. Nuestra organización utiliza correo de Office 365 por lo que mi servidor de SMTP es smtp.office365.com, pero no da problema con ningún correo que haya probado, a diferencia de SCSM, que para la generación de incidencias automáticas desde correos electrónicos es bastante más sensible.

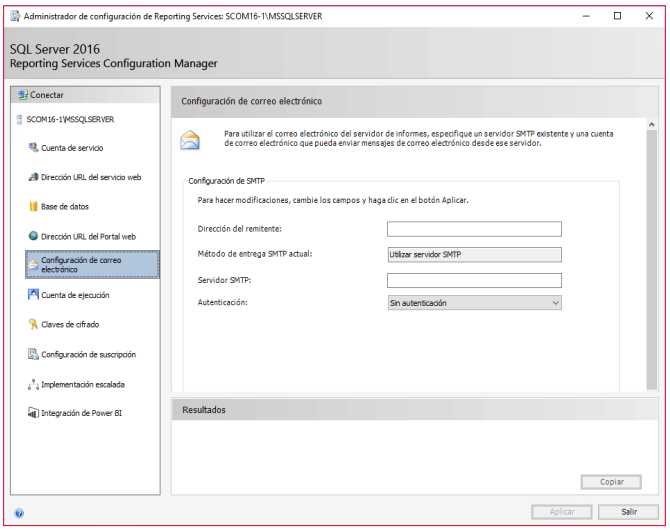


Imagen 9.- Configuración de Correo electrónico.

Tal como se puede ver en la captura, podemos seleccionar distintos tipos de "autenticación". Concretamente podemos seleccionar una de las tres opciones siguientes:

- Sin autenticación.
- Nombre de usuario y contraseña (básico).
- Cuenta de servicio del servidor de informes (NTLM).

Podemos ver que se nos da la opción de no utilizar autenticación. Esta opción es poco segura y no está soportada por muchos servidores de correo. Por estas dos razones y sobre todo por la seguridad, no dejaremos esta opción como no autenticada. Una vez configurada la opción que más se acerque a nuestras necesidades y para que el cambio sea efectivo, deberemos hacer clic en "Aplicar".

configuramos la "cuenta de ejecución" que nos permite especificar la cuenta que se utilizará para el uso de orígenes de datos

Cuenta de ejecución

A continuación, configuramos la "cuenta de ejecución" que nos permite especificar la cuenta que se utilizará para el uso de orígenes de datos de informes que no requieren validación y también para que se puedan extraer informaciones adicionales de servidores para la generación de los informes presentados. En esta parte debemos asegurarnos de especificar una cuenta de usuario de dominio con los permisos mínimos para realizar sólo aquellas operaciones de lectura requeridas. De esta manera evitaremos el acceso a información que no queramos que sea accedida. Debemos recordar que System Center está diseñado para la gestión de la seguridad de nuestra infraestructura y este tipo de temas debemos cuidarlos siempre. Deberemos especificar la cuenta de ejecución, proporcionando el nombre de usuario y la contraseña.

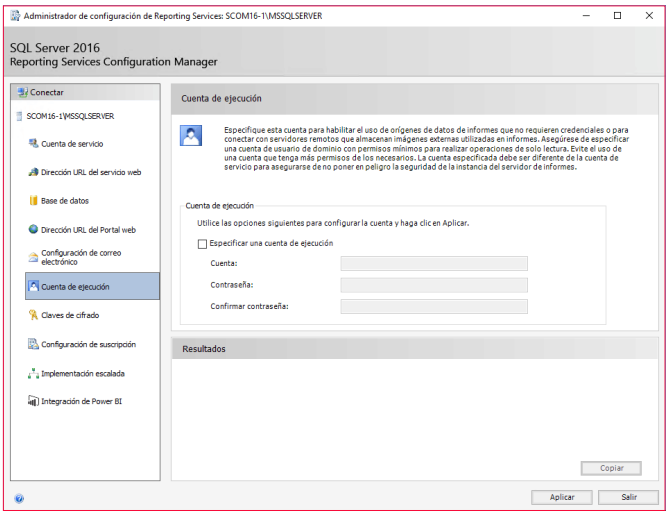


Imagen 10.- Configuración de la cuenta de ejecución.

Claves de cifrado

Ahora pasamos a configurar las "claves de cifrado" que Reporting Services utiliza para cifrar las credenciales, cadenas de conexión y otra información confidencial almacenadas en la base de datos del servidor de informes. Reporting Services utiliza claves simétricas. Durante la configuración podemos realizar una copia de seguridad de la clave de cifrado, restaurar una clave, cambiar la clave en uso por otra o eliminar el contenido cifrado.

Hay que tener cuidado con la última opción, ya que, en caso de eliminar el contenido cifrado, se eliminarán también las cadenas de conexión, las credenciales y todos los valores cifrados que se han almacenado en la suscripción, lo cual puede ocasionarnos problemas en caso de no hacerlo de manera consciente. Además, si nosotros eliminamos el contenido cifrado debemos definir conexiones, suscriptores, etc. de nuevo.

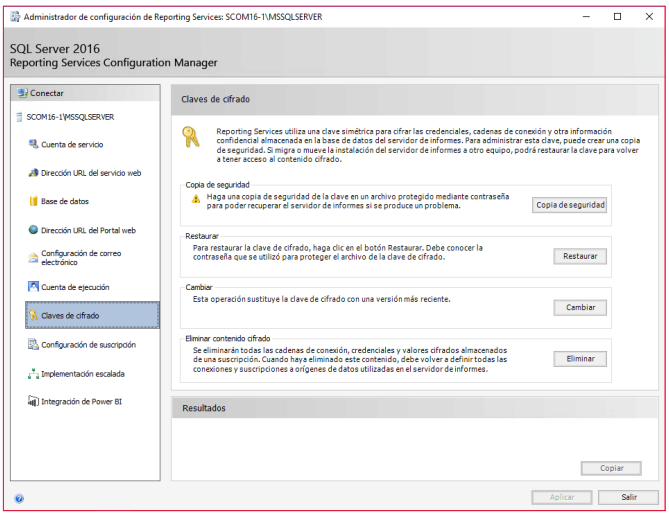


Imagen 11.- Configuración de la clave de cifrado.

Configuración de suscripción

La "Configuración de suscripción", que es el siguiente paso a configurar, sirve como cuenta de recurso compartido de archivos y nos permite ver la información de una implementación escalada. Este tipo de despliegues puede con-

tener datos cifrados en una base de datos de informes común, por lo que necesitaremos este tipo de suscripciones. Dicho de otra manera, esta cuenta usará las suscripciones que tenga configuradas para tener acceso a los recursos compartidos de archivos. Nuevamente, tenemos que tener en cuenta el usar una cuenta con pocos privilegios para cumplir con las necesidades de seguridad de nuestra organización. En este caso debemos dar un nombre de usuario, una contraseña y la validación de esta.

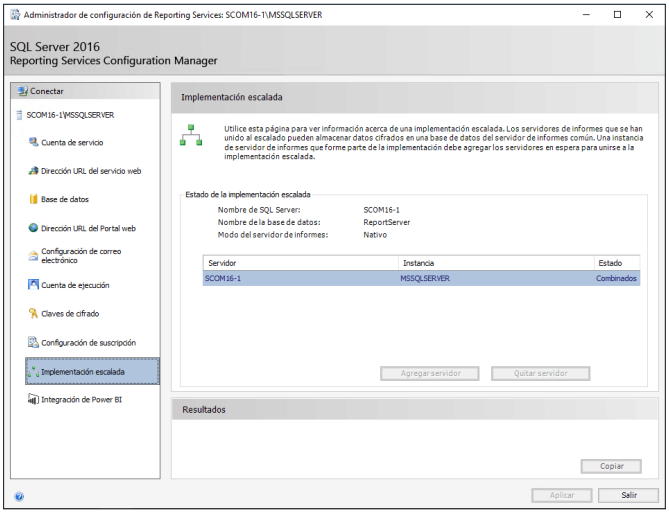


Imagen 12.- Configuración de suscripción.

esto permite que los usuarios puedan anclar elementos del informe a los paneles de Power BI

Integración de Power BI

En la sección de “Integración de Power BI” tenemos la posibilidad de establecer el registro del servidor de informes contra el Power BI, esto permite que los usuarios puedan anclar elementos del informe a los paneles de Power BI. En esta fase del proyecto no estamos interesados en este punto, aunque puede ser interesante de cara a estudiar y analizar los datos de nuestras incidencias y sobre todo buscar tendencias. En nuestro caso lo dejamos por defecto.

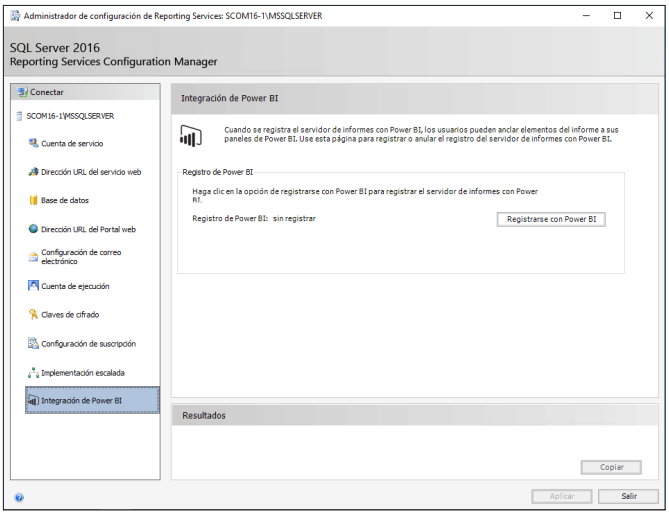


Imagen 13.- Configuración de la integración de Power BI.

Conclusión

El Reporting Services de System Center Operations Manager es de sumo interés para el correcto funcionamiento de nuestra organización ya que nos permite generar informes, hacer estudios de incidencias, buscar tendencias, etc. Debemos tener sumo cuidado en las modificaciones ya que es uno de los puntos críticos en la configuración y mantenimiento de SCOM y más, tras pequeños bugs que se han dado en el pasado.

JUAN IGNACIO OLLER AZNAR
MVP Cloud and Datacenter Management
jioller@live.com
@jioller
http://blogs.itpro.es/jioller
http://dtc2mobility.com

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

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

Santiago Porras

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

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

Sitio Web: http://www.santiagoporras.es
Email: santiagoporras@outlook.com
Blogs: http://geeks.ms/santypr
Twitter: @saintwukong

Coordinadores de sección

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

¿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