Nº34 diciembre 2017

CompartiMOSS

REVISTA ESPECIALIZADA EN TECNOLOGÍAS MICROSOFT

Entrevista Vladimir Medina

Haciendo uso de los Cognitive Services desde spfx

Usando Cosmos DB Gremlin Graph API

4 formas de añadir un administrador secundario a ODFB

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

Editorial

Con el número 34 de CompartiMOSS cerramos el año 2017. Un año en el que, aunque no hemos visto desarrollos que podamos llamar espectaculares, si podemos asegurar que ha sido el año de la consolidación definitiva de las tecnologías en la nube de Microsoft.

Azure no solamente ha resultado ser la respuesta de Microsoft (y, especialmente, de Satya Nadella) para regresar al mundo de empresas de punto de tecnologías de IT después de años y años de decadencia, sino que está en camino a reemplazar a tecnologías "tradicionales" (Windows, SQL Server) como la parte de la empresa que produce la mayor cantidad de ingresos. Por su parte, Office 365 tiene cada vez más aceptación popular y comienza a ser el servicio de colaboración y manejo de información por defecto en todo tipo de empresas, de pequeñas a multinacionales de cualquier sector y mercado.

Tal vez el único cambio dramático del año ha sido el anuncio de Microsoft indicando que abandonan el terreno de Windows Mobile. Por un lado, un empobrecimiento del mercado, pues solamente quedan dos productores que (sobre)dominan el mundo de los teléfonos, pero por otro lado simplemente la constatación del hecho de que Microsoft nunca logró conseguir una proporción de usuarios significativa, a pesar de tener una excelente tecnología en sus manos.

Veremos que nos trae el nuevo año. En cualquier caso, desde CompartiMOSS seguiremos informado sobre el mundo de tecnologías y productos de Microsoft, tal como lo hemos hecho en los últimos diez años. Y estamos seguros que ustedes, nuestros lectores, seguirán disfrutando de la revista tanto como nosotros, los autores y editores disfrutamos creándola.

El equipo editorial de CompartiMOSS

04

Patrones de Diseño en Typescript aplicados a SharePoint Framework – Factory Method

Si tus skills vienen del lado del desarrollo del servidor como los del autor de este texto, existe una gran posibilidad de haber utilizado: C # / .NET Farm solutions o servicios de WCF, REST API o ASP.NET WebForms o MVC, principalmente el singleton. El objetivo de este texto es describir el proceso de varios meses utilizando los patrones de diseño de Typescript aplicados a los webparts de SharePoint Framework, tomando en cuenta que SPFx es 100% del lado del cliente, pero eso no significa que se tiene que crear la lógica y la UI y poner todo en solo archivo .TS sin ningún tipo de estructura, como se puede ver en numerosos blogs, artículos e incluso en la documentación oficial de Microsoft.

Los patrones de diseño nos permiten crear código limpio y comprensible que cualquiera puede entender (con cierta experiencia) e introducir cambios luego de la entrega a la producción de manera sencilla, como arquitecto de soluciones fui responsable del diseño y la entrega de grandes proyectos, con ayuda de estos patrones se pudieron llevar a cabo una gran cantidad de ellos.

Quería escribir sobre este tema hace por mucho tiempo, pero no fue fácil hacerlo, como dijo Martil Golding, “siempre debes programar pensando que la persona que va a mantener tu código es un psicópata violento que sabe dónde vives”, pero ¿Cómo poder hacerlo? Una respuesta es, con los patrones de diseño. En este articulo explicare el patrón Factory Method y cómo usarlo en SPFx.

Patrones de diseño

En ingeniería de software, un patrón de diseño es una solución de repetición general a un problema que comúnmente ocurre en desarrollo de software, un patrón de diseño no es un diseño que puede ser transformado directamente en código, es una descripción o una plantilla para saber cómo resolver un problema que puede ser utilizado en muchas situaciones diferentes.

Patrones de diseño para creación de objetos

1.– Método Fabrica (Factory Method).
2.– Abstract Factory.
3.– Builder.
4.– Prototype.
5.– Singleton.

Hay otras categorías de patrones también como: Comportamiento, estructural, principios SOLID entre otros, en este artículo me concentrare en el Método Fabrica y en próximos números de la revista veremos los patrones mencionados anteriormente.

Método de fábrica (Factory method)

En la programación basada en clases, el patrón de método de fábrica es creacional ya que usa métodos de fábrica para tratar el problema de crear objetos sin tener que especificar la clase exacta del objeto que se creará. Esto se hace construyendo objetos llamando a un método de fábrica, ya sea especificado en una interfaz e implementado por clases secundarias o implementado en una clase base, opcionalmente anulado por clases derivadas en lugar de llamar a un constructor.

Otra definición que me gustó más y se ajusta perfectamente a mi muestra hoy es la siguiente, tomada de: https://www.javatpoint.com/factory-method-design-pattern

Un Patrón de Fábrica o un Patrón de Método de Fábrica dice que solo defina una interfaz o clase abstracta para crear un objeto, pero deje que las subclases decidan qué clase instanciar. En otras palabras, las subclases son responsables de crear la instancia de la clase.

Ventaja del patrón de diseño de fábrica

El Patrón de diseño de fábrica permite a las subclases elegir el tipo de objetos para crear, promueve el acoplamiento libre al eliminar la necesidad de vincular clases específicas de la aplicación en el código, eso significa que el código interactúa únicamente con la interfaz resultante o clase abstracta, de modo que, funcionará con cualquier clase que implemente esa interfaz o que amplíe esa clase abstracta.

¿Cuando usar el patrón de diseño de fábrica?

¿Cómo saber cuándo se usa? Para comenzar con la explicación de cómo aplicar el patrón de fabrica es necesario que existan tres supuestos:

1.– Cuando una clase no sabe qué subclases serán

necesarias para crear.

2.– Cuando una clase requiera que sus subclases especifiquen los objetos que se crearán.

3.– Cuando las clases principales elijan la creación de objetos para sus subclases.

En un sitio SharePoint se pueden tener múltiples listas y todas ellas pueden tener diferentes columnas o campos, ¿por qué no crear una forma genérica para construir los objetos de la lista de elementos dependiendo de la lista seleccionada? En palabras simples un sitio web en el cual se pueda seleccionar la lista y en función de ella se mostrarán todas las columnas, esto se puede hacer de diversas maneras y con varios switch, "if", etc., pero en este caso, busque una solución más elegante, en otras palabras, traté de simplificarlo.

Partiendo de lo ya mencionado y para comenzar, los patrones de diseño que utilizo estarán en mis repositorios de github: https://github.com/levalencia

Imagen 1.- Diagrama de Patrones de diseño.

En el diagrama anterior en lugar de clases se tienen interfaces para el elemento de lista genérico IListItem y otras interfaces que extienden la base para agregar más campos según la lista, las noticias, los anuncios y el directorio.

En el lado derecho del diagrama, tenemos una interfaz IFactory que declara la firma del método getItems y lo que debe devolver "arreglo de any" , es decir, cualquier objeto, recordemos que al final todos los elementos serán de tipo IListItem porque extienden esa interfaz.

Y finalmente, en el componente FactoryMethod de React utilizamos ListItemFactory para obtener los elementos, desde el punto de vista del que llama, no importa qué devolverá, es la responsabilidad del método de fábrica crear realmente la lógica interna para saber qué tipo de instancia debería retornar.

Estructura del proyecto

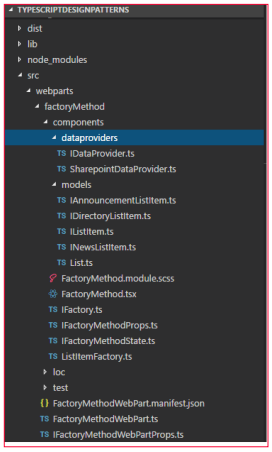


Imagen 2.- Estructura del proyecto.

Modelos

Tiene el código de los objetos de negocio que queremos que sean convertidos desde el json retornado por los servicios.

```
export  interface IListItem {
   [key: string]: any;
   id: string;
   title: string;
   modified: Date;
   created: Date;
   modifiedby: string;
   createdby: string;
}

import {IListItem} from "./IListItem";

export interface  IAnnouncementListItem extends IListItem {
   announcementBody: string;
   expiryDate: Date;
}

import {IListItem} from "./IListItem";

export interface IDirectoryListItem extends IListItem {
      firstName: string;
      lastName: string;
      mobileNumber: string;
      internalNumber: string;
}


import {IListItem} from "./IListItem";

export interface  INewsListItem extends IListItem {
   newsheader: string;
   newsbody: string;
   expiryDate: Date;
}
```

"Los patrones de diseño nos permiten crear código limpio y comprensible que cualquiera puede entender en el componente FactoryMethod de React utilizamos ListItemFactory"

Clases de fábrica

Entonces, la interfaz Factory es bastante simple, solo tenemos un método para implementar en las clases extendidas.

```
import { IListItem } from "./models/IListItem";
import { SPHttpClient, SPHttpClientResponse } from "@microsoft/sp-http";
export  interface IFactory {
   getItems(requester: SPHttpClient, siteUrl: string, listName: string): Promise<any[]>;
}
```

A continuación, es en este método donde existe una declaración switch para obtener una lista diferente con una URL diferente (Seleccionar columnas) y entonces en el retorno se obtienen diferentes tipos en concreto.

```
import { SPHttpClient, SPHttpClientResponse } from "@microsoft/sp-http";
```

```
import { IWebPartContext } from "@microsoft/sp-webpart-base";
import { IListItem} from "./models/IListItem";
import { IFactory } from "./IFactory";
import { INewsListItem } from "./models/INewsListItem";
import { IDirectoryListItem } from "./models/IDirectoryListItem";
import { IAnnouncementListItem } from "./models/IAnnouncementListItem";

export class ListItemFactory implements IFactory {
  public getItems(requester: SPHttpClient, siteUrl: string, listName: string): Promise<any[]> {
    switch(listName) {
      case "GenericList":
        let items: IListItem[];
        return requester.get(`${siteUrl}/_api/web/lists/getbytitle('${listName}')/items?$select=Title,Id,Modified,Created,Author/Title,Editor/Title&$expand=Author,Editor`,
        SPHttpClient.configurations.v1,
        {
          headers: {
            "Accept": "application/json;odata=nometadata",
            "odata-version": ""
          }
        })
        .then((response: SPHttpClientResponse): Promise<{ value: IListItem[] }> => {
          return response.json();
        })
        .then((json: { value: IListItem[] }) => {
          console.log(JSON.stringify(json.value));
          return items=json.value.map((v,i)=>(
            {
              id: v.Id,
              title: v.Title,
              created: v.Created,
              createdby: v.Author.Title,
              modified: v.Modified,
              modifiedby: v.Editor.Title
            }
          ));
        });
      case "News":
        let newsitems: INewsListItem[];
        return requester.get(`${siteUrl}/_api/web/lists/getbytitle('${listName}')/items?$select=Title,Id,Modified,Created,newsheader,newsbody,expiryDate,Author/Title,Editor/Title&$expand=Author,Editor`,
        SPHttpClient.configurations.v1,
        {
          headers: {
            "Accept": "application/json;odata=nometadata",
            "odata-version": ""
          }
        })
        .then((response: SPHttpClientResponse): Promise<{ value: INewsListItem[] }> => {
          return response.json();
        })
        .then((json: { value: INewsListItem[] }) => {
          return newsitems=json.value.map((v,i)=>(
            {
              id: v.Id,
              title: v.Title,
              created: v.Created,
              createdby: v.Author.Title,
              modified: v.Modified,
              modifiedby: v.Editor.Title,
              newsheader: v.newsheader,
              newsbody: v.newsbody,
              expiryDate: v.expiryDate
            }
          ));
        });
      case "Announcements":
        let announcementitems: IAnnouncementListItem[];
        return requester.get(`${siteUrl}/_api/web/lists/getbytitle('${listName}')/items?$select=Title,Id,Modified,Created,announcementBody,expiryDate,Author/Title,Editor/Title&$expand=Author,Editor`,
        SPHttpClient.configurations.v1,
        {
          headers: {
            "Accept": "application/json;odata=nometadata",
            "odata-version": ""
          }
        })
        .then((response: SPHttpClientResponse): Promise<{ value: IAnnouncementListItem[] }> => {
          return response.json();
        })
        .then((json: { value: IAnnouncementListItem[] }) => {
          return announcementitems=json.value.map((v,i)=>(
            {
              id: v.Id,
              title: v.Title,
              created: v.Created,
              createdby: v.Author.Title,
              modified: v.Modified,
              modifiedby: v.Editor.Title,
              announcementBody: v.announcementBody,
              expiryDate: v.expiryDate
            }
          ));
        });
      case "Directory":
        let directoryitems: IDirectoryListItem[];
        return requester.get(`${siteUrl}/_api/web/lists/getbytitle('${listName}')/items?$select=Title,Id,Modified,Created,-firstName,lastName,mobileNumber,internalNumber,Author/Title,Editor/Title&$expand=Author,Editor`,
        SPHttpClient.configurations.v1,
        {
          headers: {
            "Accept": "application/json;odata=nometadata",
            "odata-version": ""
          }
        })
        .then((response: SPHttpClientResponse): Promise<{ value: IDirectoryListItem[] }> => {
          return response.json();
        })
        .then((json: { value: IDirectoryListItem[] }) => {
          return directoryitems=json.value.map((v,i)=>(
            {
              id: v.Id,
              title: v.Title,
              created: v.Created,
              createdby: v.Author.Title,
              modified: v.Modified,
              modifiedby: v.Editor.Title,
              firstName: v.firstName,
              lastName: v.lastName,
              mobileNumber: v.mobileNumber,
              internalNumber: v.internalNumber
            }
          ));
        });
      default:
        break;
    }
  }
}
```

Propiedades y estados

Las propiedades que pasan del elemento WebPart al componente se definen en la interfaz de utilería, cosas como SPHttpclient son importantes aquí, entonces el estado, el estado es donde realmente se almacena nuestra información devuelta del servidor, porque los elementos de lista pueden ser de tipos diferentes, yo creé una interfaz contenedora y según el tipo de lista, el estado se leerá desde una propiedad diferente DetailsListItemState, DetailsNewsListItemState, etc.

```
import { SPHttpClient } from "@microsoft/sp-http";
import IDataProvider from "./dataproviders/IDataProvider";

export interface IFactoryMethodProps {
 listName: string;
 spHttpClient: SPHttpClient;
```

```
  siteUrl: string;
  dataProvider: IDataProvider;
  configureStartCallback: () => void;
}

import { IListItem } from "./models/IListItem";
import { INewsListItem } from "./models/INewsListItem";
import { IDirectoryListItem } from "./models/IDirectoryListItem";
import { IAnnouncementListItem } from "./models/IAnnounce-
mentListItem";
import {
  IColumn
} from "office-ui-fabric-react/lib/DetailsList";

export interface IFactoryMethodState {
  hasError: boolean;
  status: string;
  columns: IColumn[];
  DetailsGenericListItemState: IDetailsGenericListItemState;
  DetailsNewsListItemState: IDetailsNewsListItemState;
  DetailsDirectoryListItemState : IDetailsDirectoryListItemState;
  DetailsAnnouncementListItemState : IDetailsAnnouncement-
ListItemState;
}

export interface IDetailsGenericListItemState {
  items: IListItem[];
}

export interface IDetailsNewsListItemState {
  items: INewsListItem[];
}

export interface IDetailsDirectoryListItemState {
  items: IDirectoryListItem[];
}

export interface IDetailsAnnouncementListItemState {
  items: IAnnouncementListItem[];
}
```

El componente

El componente tiene mucha lógica UI, pero la verdadera magia está en el método que consume el método fabrica es decir readItemsAndSetStatus, donde se usa la clase Fabrica para obtener los elementos y luego se establece el estado correspondiente, esto es bastante bueno ya que se evita la redundancia en el método readItemsAndSetStatus

```
//#region Imports
import * as React from "react";
import styles from "./FactoryMethod.module.scss";
import  { IFactoryMethodProps } from "./IFactoryMethodProps";
import {
  IDetailsGenericListItemState,
  IDetailsNewsListItemState,
  IDetailsDirectoryListItemState,
  IDetailsAnnouncementListItemState,
  IFactoryMethodState
} from "./IFactoryMethodState";
import { IListItem } from "./models/IListItem";
import { IAnnouncementListItem } from "./models/IAnnounce-
mentListItem";
import { INewsListItem } from "./models/INewsListItem";
import { IDirectoryListItem } from "./models/IDirectoryListItem";
import { escape } from "@microsoft/sp-lodash-subset";
import { SPHttpClient, SPHttpClientResponse } from "@micro-
soft/sp-http";
import { ListItemFactory} from "./ListItemFactory";
import { TextField } from "office-ui-fabric-react/lib/TextField";
import {
  DetailsList,
  DetailsListLayoutMode,
  Selection,
  buildColumns,
  IColumn
} from "office-ui-fabric-react/lib/DetailsList";
import { MarqueeSelection } from "office-ui-fabric-react/lib/Mar-
queeSelection";
import { autobind } from "office-ui-fabric-react/lib/Utilities";
import PropTypes from "prop-types";
//#endregion

export default class FactoryMethod extends React.Component<I-
FactoryMethodProps, IFactoryMethodState> {
  constructor(props: IFactoryMethodProps, state: any) {
    super(props);   this.setInitialState();
  }

  // lifecycle help here: https://staminaloops.github.io/undefinedis-
notafunction/understanding-react/

  //#region Mouting events lifecycle
  // the data returned from render is neither a string nor a DOM node.
  // it's a lightweight description of what the DOM should look like.
  // inspects this.state and this.props and create the markup.
  // when your data changes, the render method is called again.
  // react diff the return value from the previous call to render with
  // the new one, and generate a minimal set of changes to be ap-
plied to the DOM.
  public render(): React.ReactElement<IFactoryMethodProps> {
    switch(this.props.listName) {
      case "GenericList":
       return <this.ListMarqueeSelection items={this.state.DetailsGe-
nericListItemState.items} columns={this.state.columns} />;
      case "News":
      return <this.ListMarqueeSelection items={this.state.Detail-
sNewsListItemState.items} columns={this.state.columns}/>;
      case "Announcements":
     return <this.ListMarqueeSelection items={this.state.DetailsAn-
nouncementListItemState.items} columns={this.state.columns}/>;
      case "Directory":
      return <this.ListMarqueeSelection items={this.state.DetailsDi-
rectoryListItemState.items} columns={this.state.columns}/>;
      default:
       return null;
    }
  }

   // invoked once, only on the client (not on the server), immediately
AFTER the initial rendering occurs.
   public componentDidMount(): void {
    // you can access any refs to your children
    // (e.g., to access the underlying DOM representation - React-
DOM.findDOMNode).
    // the componentDidMount() method of child components is
invoked before that of parent components.
    // if you want to integrate with other JavaScript frameworks,
    // set timers using setTimeout or setInterval,
    // or send AJAX requests, perform those operations in this meth-
od.
    this._configureWebPart = this._configureWebPart.bind(this);
    this.readItemsAndSetStatus("GenericList");
  }
  //#endregion

  //#region Props changes lifecycle events (after a property chang-
es from parent component)
  public componentWillReceiveProps(nextProps: IFactoryMethod-
Props): void {
    // in the parent component (webpart) when the dropdownchang-
es it triggers this event in the child component, however
    // we need to check if the selected list is different to set the state
    // setting the state will trigger a render
    if(nextProps.listName !== this.props.listName) {
      this.readItemsAndSetStatus(nextProps.listName);
    }
  }
  //#endregion

  //#region private methods
  private _configureWebPart(): void {
    this.props.configureStartCallback();
  }

  public setInitialState(): void {
    this.state = {
      status: this.listNotConfigured(this.props)
        ? "Please configure list in Web Part properties"
        : "Ready",
      columns:[],
      hasError: false,
      DetailsGenericListItemState:{
        items:[]
      },
```

```
    DetailsNewsListItemState:{
      items:[]
    },
    DetailsDirectoryListItemState:{
      items:[]
    },
    DetailsAnnouncementListItemState:{
      items:[]
    },
  };
 }

 // reusable inline component
 private ListMarqueeSelection = (itemState: {columns: IColumn[],
items: IListItem[] }) => (
    <div>
      <DetailsList
        items={ itemState.items }
        columns={ itemState.columns }
        setKey="set"
        layoutMode={ DetailsListLayoutMode.fixedColumns }
        selectionPreservedOnEmptyClick={ true }
        compact={ true }>
      </DetailsList>
    </div>
 )

 // read items using factory method pattern and sets state accord-
ingly
 private  readItemsAndSetStatus(listName: string): void {
  this.setState({
    status: "Loading all items..."
  });

   const factory: ListItemFactory = new ListItemFactory();
   factory.getItems(this.props.spHttpClient, this.props.siteUrl, list-
Name)
   .then((items: any[]) => {
     const keyPart: string = listName === "GenericList" ? "Generic" :
listName;

     var myItems = null;
     switch(listName) {
       case "GenericList":
         myItems = items as IListItem[];
         break;
       case "News":
         myItems = items as INewsListItem[];
         break;
       case "Announcements":
         myItems = items as IAnnouncementListItem[];
         break;
       case "Directory":
         myItems = items as IDirectoryListItem[];
         break;
     }

      // the explicit specification of the type argument `keyof {}` is bad
and
      // it should not be required.
      this.setState<keyof {}>({
        status: `Successfully loaded ${myItems.length} items`,
        ["Details" + keyPart + "ListItemState"] : {
          items
        },
        columns: buildColumns(myItems)
      });
  });
 }

 private listNotConfigured(props: IFactoryMethodProps): boolean {
  return props.listName === undefined ||
    props.listName === null ||
    props.listName.length === 0;
 }

 //#endregion
}
```

WebPart

Y, finalmente, el código de la WebPart se encuentra debajo, con comentarios para que el lector entienda el ciclo de vida de los eventos.

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

import * as strings from "FactoryMethodWebPartStrings";
import FactoryMethod from "./components/FactoryMethod";
import { IFactoryMethodProps } from "./components/IFactoryMethod-
Props";
import { IFactoryMethodWebPartProps } from "./IFactoryMethodWebPart-
Props";
import * as lodash from "@microsoft/sp-lodash-subset";
import List from "./components/models/List";
import { Environment, EnvironmentType } from "@microsoft/sp-core-li-
brary";
import IDataProvider from "./components/dataproviders/IDataProvider";
import MockDataProvider from "./test/MockDataProvider";
import SharePointDataProvider from "./components/dataproviders/Share-
pointDataProvider";

export default class FactoryMethodWebPart extends BaseClientSideWeb-
Part<IFactoryMethodWebPartProps> {
  private _dropdownOptions: IPropertyPaneDropdownOption[];
  private _selectedList: List;
  private _disableDropdown: boolean;
  private _dataProvider: IDataProvider;
  private _factorymethodContainerComponent: FactoryMethod;

  protected onInit(): Promise<void> {
    this.context.statusRenderer.displayLoadingIndicator(this.domElement,
"Todo");

    /*
    Create the appropriate data provider depending on where the web part is
running.
    The DEBUG flag will ensure the mock data provider is not bundled with
the web part when you package the
     solution for distribution, that is, using the --ship flag with the pack-
age-solution gulp command.
    */
    if (DEBUG && Environment.type === EnvironmentType.Local) {
      this._dataProvider = new MockDataProvider();
    } else {
      this._dataProvider = new SharePointDataProvider();
      this._dataProvider.webPartContext = this.context;
    }

    this.openPropertyPane = this.openPropertyPane.bind(this);

    /*
    Get the list of tasks lists from the current site and populate the property
pane dropdown field with the values.
    */
    this.loadLists()
      .then(() => {
        /*
        If a list is already selected, then we would have stored the list Id in the
associated web part property.
        So, check to see if we do have a selected list for the web part. If we do,
then we set that as the selected list
        in the property pane dropdown field.
        */
        if (this.properties.spListIndex) {
          this.setSelectedList(this.properties.spListIndex.toString());
          this.context.statusRenderer.clearLoadingIndicator(this.domElement);
        }
      });
```

Para concluir este artículo en donde aprendió a crear interfaces y extenderlas con otras interfaces para agregar campos, como si fuera herencia de clases.

Para obtener la última versión de la WebPart, recuerda revisar mis repositorios en github: https://github.com/levalencia

LUIS VALENCIA
MVP Office Development
le.valencia@outlook.com
@levalencia
http://www.luisevalencia.com

09

Seguridad en Azure AD

Azure AD, la solución de identidad como servicio (IdaaS) de Microsoft, es un complejo sistema de gestión de identidades y acceso (IAM) que puede desplegarse en la nube o de forma híbrida, sincronizando directorios locales con Azure AD Connect. Dependiendo del tipo de licencia, tendremos a nuestra disposición una serie de características avanzadas de seguridad y administración que facilitarán la gestión de identidades y accesos a cualquier escala.

¿Qué es AAD?

Ante esta pregunta resulta tentador afirmar que AAD es la versión cloud de AD (Directorio Activo). Pero en realidad es mucho más. Azure AD ofrece administración de identidades y control de acceso a aplicaciones SaaS y locales, experiencia de inicio de sesión único, integración con bosques de AD local, y toda una serie de "extras" dependiendo de la versión que tengamos contratada: autenticación multifactor (MFA), administración de dispositivos, autoservicio de contraseñas y grupos, administración de cuentas con privilegios (PIM), control de acceso basado en roles (RBAC), protección de identidades, acceso condicional, etc.

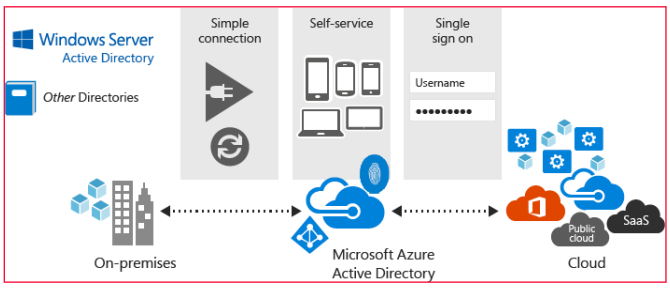


Imagen 1.- Esquema de flujo de identidades cloud y on-premises.

En este artículo vamos a profundizar en las características de seguridad avanzada que ofrece Azure AD Premium. Pero antes repasemos las opciones de licenciamiento y de arquitectura de identidades.

Identidades

Para la comprobación de identidades (autenticación) tenemos varias opciones:

- Identidad Cloud: las cuentas se administran solamente en la nube. La autenticación la realiza Azure AD. Es la configuración por defecto cuando se crea un nuevo tenant de Office 365, por ejemplo.

Las demás opciones están disponibles utilizando AD Connect para sincronizar objetos de Directorio Activo local con Azure AD.

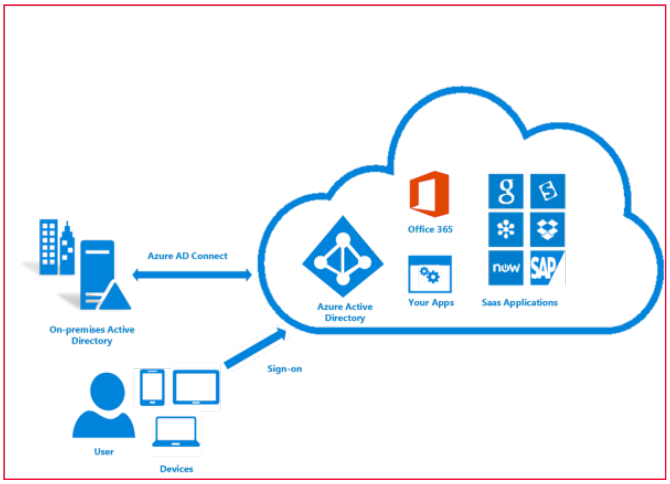


Imagen 2.- Esquema de arquitectura de sincronización con Azure AD Connect.

La ventaja de Azure AD Connect es que los usuarios pueden acceder a recursos locales y en la nube con la misma cuenta. Otra ventaja es que no se necesita crear cuentas en Azure AD, pues éstas se sincronizan desde el Directorio Activo local. Microsoft recomienda instalar Azure AD Connect en una máquina independiente con acceso al bosque de Directorio Activo local.

- Identidad sincronizada: las identidades existen tanto en la nube como en local, pero la autenticación sucede en la nube. Requiere Servicios de Dominio de Directorio Activo (ADDS), Azure AD, y AD Connect con sincronización de contraseñas. Los usuarios y las contraseñas de ADDS se sincronizan con Azure AD.
- Federación con ADFS: las identidades existen tanto en la nube como en local. La autenticación se realiza en local a través de los Servicios de Federación de Directorio Activo (ADFS) contra el Directorio Activo local. Requiere ADDS federado con ADFS, Azure AD, y AD Connect. Los usuarios se sincronizan con Azure AD, pero no así las contraseñas. Además de los servidores ADDS y ADFS, se debe configurar Azure AD Connect para utilizar ADFS, y de esta forma federar los dos directorios, lo que genera una relación de confianza entre Azure AD y ADFS. El login ocurre en la nube, pero la autenticación se redirige al ADFS.
- Pass-through authentication: las identidades existen tanto en la nube como en local. La autenticación se realiza en local a través del Directorio Activo (ADDS). Requiere Servicios de Dominio de Directorio Acti-

vo (ADDS), Azure AD, y AD Connect con la opción de pass-through authentication. Los usuarios se sincronizan con Azure AD. Se instala un agente en un Windows Server local, para recibir y responder a peticiones de validación de contraseñas. Recibe las contraseñas cifradas desde Azure AD, las descifra y las valida contra el Directorio Activo local. La comunicación se realiza a través de un Bus de Servicio de Azure.
En cuanto al login, y a diferencia de la opción ADFS donde la contraseña se ingresa en local, en este caso tanto usuario como contraseña se ingresan en la nube, y Azure AD envía la contraseña cifrada al Directorio Activo para validación. Al igual que la opción ADFS, las contraseñas no se guardan en la nube.

- Federación con solución de terceros: las identidades existen tanto en la nube como en local.
  La autenticación se realiza en local a través de un servidor de identidades de terceros contra el Directorio Activo local. Requiere Azure AD, AD Connect con la opción "do not configure", y solución de identidades de terceros. Los usuarios se sincronizan con Azure AD, pero no así las contraseñas.
  Relación de confianza entre Azure AD y la solución de terceros. El login se realiza en la nube, pero la autenticación se redirige al gestor de identidades de terceros.
  El tipo de inicio de sesión se elige durante la instalación de Azure AD Connect:

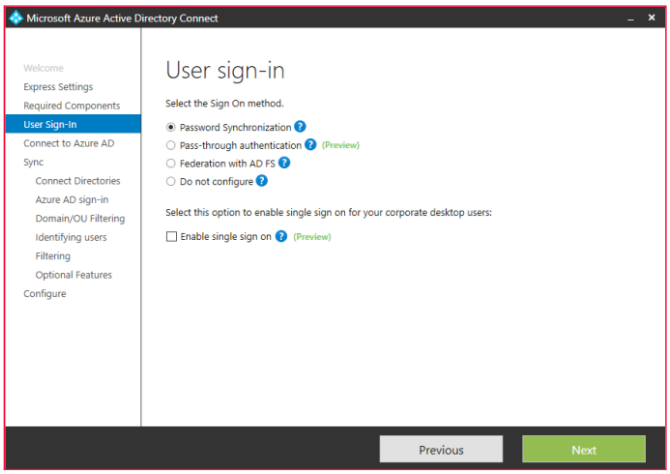


Imagen 3.- Selección de inicio de sesión en la instalación de Azure AD Connect.

Licencias

Todos los servicios de Microsoft Online dependen de Azure AD a nivel de identidades, por lo que (tal vez no lo sepas, pero…) si usas Office 365 empresarial o Dynamics CRM online, ya tienes un directorio de Azure AD. Existen cuatro niveles de licenciamiento de Azure AD: Gratuito, Básico, Premium P1, Premium P2. En realidad son cinco, si contamos el directorio que se crea con un tenant de Office 365, y que ofrece características diferentes a las cuatro opciones de licenciamiento.

Gratuito

Con la edición gratuita de AAD se pueden administrar usuarios y grupos (límite de 500K objetos), registrar dispositivos, sincronizar directorios locales con AAD Connect, configurar SSO. Además, la edición gratuita proporciona colaboración B2B con usuarios y directorios externos, y también informes básicos de seguridad y uso.

Características de Windows 10 con Azure AD join disponibles en la versión gratuita de AAD: unir dispositivos Windows 10 a Azure AD, Windows Hello para Azure AD, Bitlocker recovery.

Además, los niveles gratuito y básico de Azure AD incluyen autenticación multifactor (MFA) con el modelo de pago por uso:

https://azure.microsoft.com/es-es/pricing/details/multi-factor-authentication/

"en la edición gratuita de AAD se pueden administrar usuarios y grupos"

Básico

El nivel básico incluye todas las características de la licencia gratuita, más: administración de usuarios y grupos sin límite de objetos, acceso basado en grupo, proxy de aplicaciones (para publicar aplicaciones locales con AAD), autoservicio de restablecimiento de contraseña, personalización de páginas de login y acceso, SLA de 99,9%

Office 365

Las características del directorio de Azure AD que se crea con el tenant de Office 365 incluyen todas las del nivel básico, más: administración de usuarios y grupos sin límite de objetos de Office 365, autoservicio de restablecimiento de contraseña, personalización de páginas de login y acceso, SLA de 99,9%, y MFA sin coste para aplicaciones de Office 365.

Premium P1

Las novedades más importantes de las licencias premium, con respecto a los niveles gratuito y básico, son las características de seguridad y protección.

La licencia Premium P1 incluye todas las características de los planes básico y gratuito, más:

- Grupos dinámicos.
- Sincronización bi-direccional con directorios locales a través de Azure AD Connect.
- Microsoft Identity Manager.
- Cloud App Discovery.
- Acceso condicional según grupo y ubicación.
- Acceso condicional según dispositivo (administrado o unido a dominio).
- Inscripción automática de dispositivos.

Premium P2

La licencia Premium P2 incluye todas las características del plan Premium P1, más:

- Identity protection.
- Acceso condicional basado en riesgo de inicio de sesión o de usuario.
- Administración de cuentas con privilegios (PIM).

Seguridad en Azure AD

En cuanto a la seguridad, Azure AD protege aplicaciones, datos e identidades con varias características Premium. En las próximas líneas analizaremos cuatro de esas características:

- Autenticación Multifactor (MFA).
- Protección de identidades.
- Administración de cuentas con privilegios.
- Acceso Condicional.

MFA

La autenticación multifactor es un método que requiere más de una comprobación y agrega otro nivel de seguridad a las transacciones e inicios de sesión del usuario. Funciona mediante la solicitud de dos o más de los siguientes métodos de verificación:

- Un elemento conocido (por ejemplo, contraseña).
- Un elemento disponible (un dispositivo de confianza que no se puede duplicar con facilidad, como un móvil).
- Un elemento físico de identificación (biométrico).

Para la segunda comprobación pueden usarse los siguientes métodos:

- Llamada de teléfono.
- SMS.
- Notificación en App móvil.
- Código de verificación en App móvil.
- Tokens OATH de terceros.

Versiones de MFA

MFA para Office 365: Esta versión funciona exclusivamente con aplicaciones de Office 365 y se administra desde el portal de Office 365.

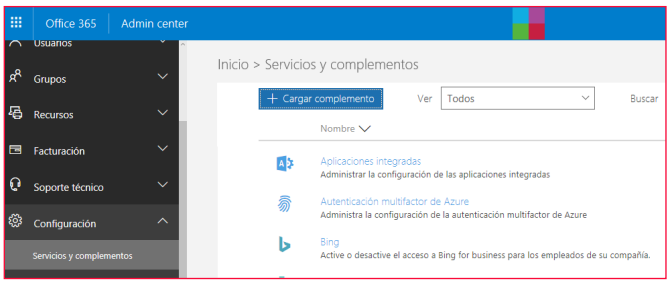


Imagen 4.- Acceso a MFA en el portal de Office 365.

En la consola de administración de Office 365, vamos a Configuración > Servicios y complementos > Azure MFA

En el panel de configuración de Azure MFA para Office 365, además de la gestión de usuarios a los que se requiere MFA, tenemos la opción de omitir IPs o rangos de IP de la autenticación multifactor.

Imagen 5.- Configuración del servicio MFA.

MFA para administradores de Azure: se puede habilitar MFA para administradores globales de Azure, sin coste adicional.

Azure MFA: Conocida como la versión “completa”, Azure Multi-Factor Authentication ofrece opciones de configuración adicionales e informes avanzados a través del Portal de Azure clásico (uno de los últimos servicios de Azure que continúan en el portal clásico). Azure Multi-Factor Authentication se incluye en los planes de Azure Active Directory Premium y en los planes de Enterprise Mobility + Security, y se puede implementar en la nube o de forma local.

Si nos decantamos por una implementación local, debemos descargar la imagen desde el portal de Azure, instalar y configurar en un servidor local:

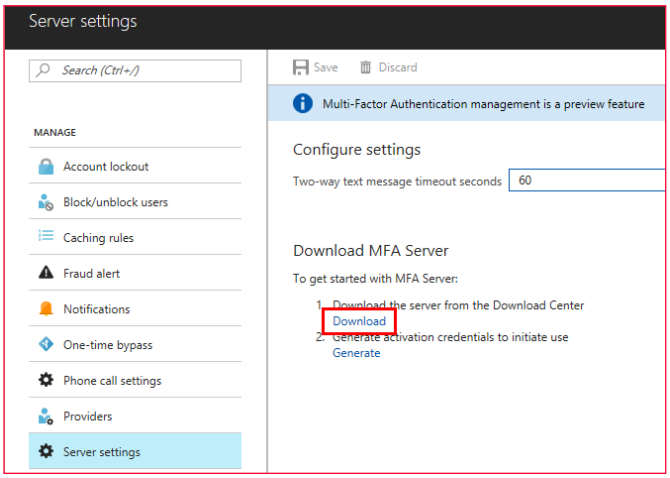


Imagen 6.- Descarga del servidor MFA.

Más detalles en el siguiente enlace:

https://docs.microsoft.com/es-es/azure/multi-factor-authentication/multi-factor-authentication-get-started-server

Protección de identidades

Azure AD Identity Protection es una característica de la ver-

sión Premium P2 que permite:

- Detectar identidades vulnerables.
- Configurar respuestas automáticas a posibles amenazas a las identidades.
- Investigar y resolver incidentes relacionados con las identidades.

Azure AD Identity Protection va más allá de una simple monitorización y reporting, permitiendo establecer directivas basadas en riesgos que puedan desencadenar una respuesta cuando se alcanza un determinado nivel de riesgo. Por ejemplo, una directiva podría bloquear accesos, forzar el reseteo de contraseñas, o exigir MFA. La detección de vulnerabilidades incluye configuración de Azure AD, como AAD Privileged Identity Management y también la presencia de aplicaciones cloud no gestionadas.

Imagen 7.- Azure AD Identity Protection.

AAD Identity Protection detecta 6 tipos de riesgos, y a cada uno le asigna un nivel: Alto, Medio, o Bajo.

- Usuarios con credenciales comprometidas.
- Inicios de sesión desde IPs anónimas.
- Viaje imposible a ubicaciones atípicas.
- Inicios de sesión desde ubicaciones inusuales.
- Inicios de sesión desde dispositivos infectados.
- Inicios de sesión desde IPs con actividades sospechosas.

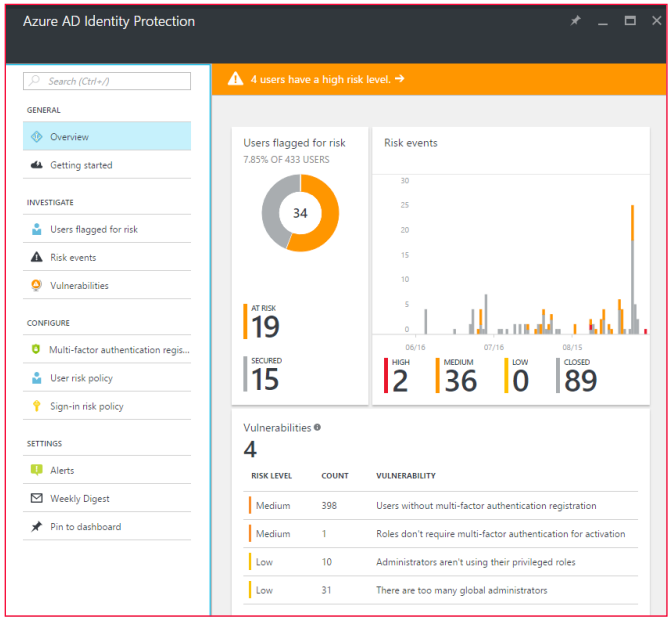


Imagen 8.- Panel de Azure AD Identity Protection.

Al igual que Advanced Threat Analytics (ATA) o Azure Advanced Threat Protection (AATP), Azure AD Identity Protection utiliza Machine Learning y análisis de comportamiento para detector actividades sospechosas en relación a identidades de usuarios. Por cada incidente detectado se crea un evento de riesgo.

En cada directiva se definen acciones para resolver los eventos de riesgo cuando se alcanza un cierto nivel. El nivel por defecto es Medio.

Azure AD Identity protection permite seleccionar a qué usuarios se va a aplicar la directiva, de tal modo que se pueden crear diferentes directivas para diferentes grupos de riesgo.

https://docs.microsoft.com/es-es/azure/active-directory/active-directory-identityprotection

Administración de cuentas con privilegios

Azure AD Privileged Identity Management es otra característica de Azure AD Premium P2 que ayuda a gestionar y proteger las credenciales de administrador, monitorizando las actividades de los administradores y restringiendo el acceso a los recursos.

Las configuraciones disponibles son:

- Ver los usuarios con rol de administrador.
- Habilitar acceso de administrador puntual (“just in time”).
- Obtener reportes históricos de acceso de administrador.
- Obtener reportes de cambios de permisos de administrador.
- Alertas sobre accesos de administrador.

Azure AD PIM introduce el concepto de “administrador apto” (según la traducción de “elegible administrator” del portal de Azure). Cuando un usuario se define como “administrador apto” recibe permisos de administrador puntuales para determinada aplicación, por un período de tiempo limitado.

“la autenticación multifactor es un método que requiere más de una comprobación y agrega otro nivel de seguridad”

El rol de ”administrador apto” está inactivo hasta que el usuario necesita acceder, momento en el cual debe completar un proceso de activación para convertirse en administrador por tiempo limitado.

En el panel de control podemos ver la siguiente información:

- Alertas sobre mejoras de seguridad.
- Número de usuarios en cada rol con privilegios.
- Número de usuarios aptos y permanentes.
- Revisiones de acceso en curso.

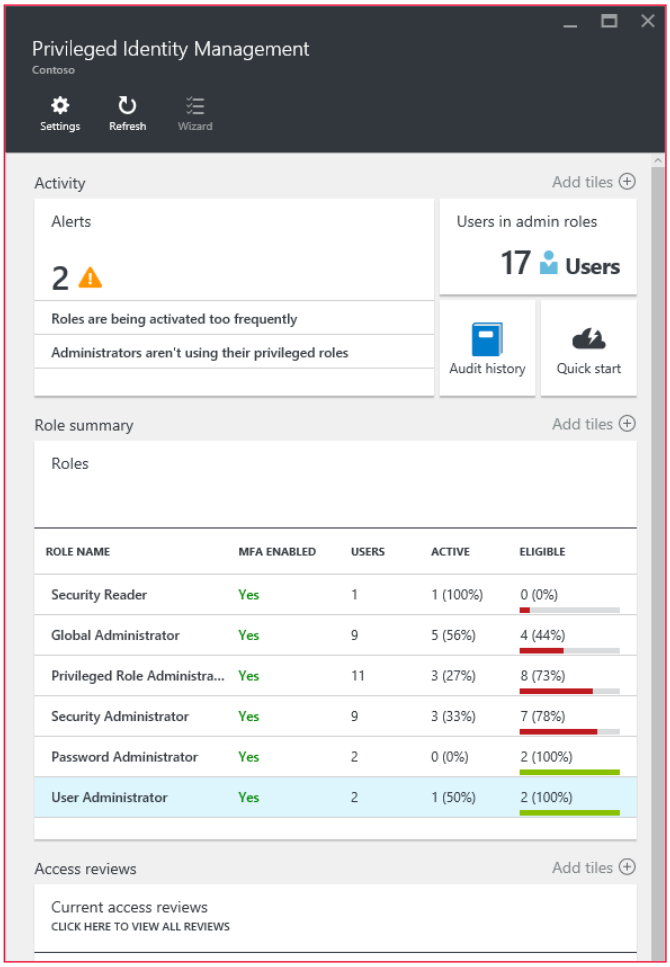


Imagen 9.- Azure AD Privileged Identity Management.

Administración de roles con privilegios

Con Azure AD Privileged Identity Management, podemos agregar o quitar administradores permanentes o aptos en cada rol de directorio de Azure AD. Mediante la configuración de roles se establecen las propiedades de activación de rol apto para las aplicaciones de Azure AD como:

- La duración del período de activación del rol.
- La notificación de activación del rol.
- La información que un usuario debe proporcionar durante el proceso de activación del rol.
- Número de incidente.
- Approval workflow requirements (Requisitos de flujo de trabajo de aprobación) (versión preliminar).

Activación de rol

Para activar un rol, un administrador apto solicita una "activación" puntual para ese rol. Se puede solicitar la activación mediante la opción Activar mi rol en Administración de Azure AD PIM. La activación del rol es personalizable. En la configuración de PIM, se puede determinar la duración de la activación, así como la información que el administrador debe proporcionar para activar el rol.

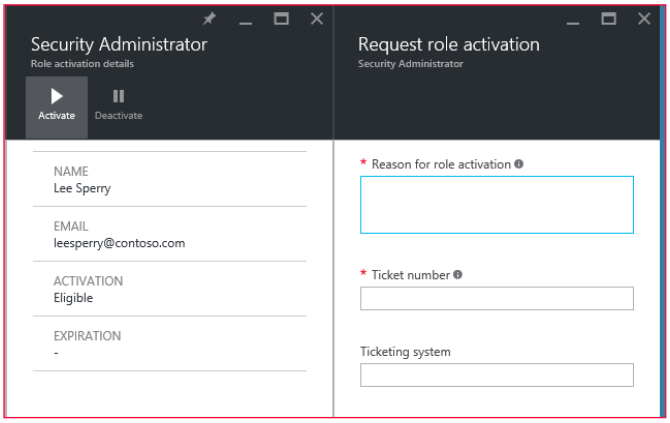


Imagen 10.- Solicitud de activación de rol de administrador.

https://docs.microsoft.com/es-es/azure/active-directory/active-directory-privileged-identity-management-getting-started

Acceso Condicional

El acceso condicional es una funcionalidad de Azure AD Premium que permite aplicar controles de acceso a las aplicaciones según condiciones específicas. El cumplimiento de estas condiciones permitirá o bloqueará el acceso. La implementación del acceso condicional se basa en directivas. Estas directivas definen los requisitos de acceso mediante declaraciones basadas en el siguiente patrón:

Si ocurre esto	Se debe hacer esto

Al reemplazar las declaraciones por información real, obtenemos un ejemplo de una declaración de directiva:

Cuando los usuarios intentan acceder a las aplicaciones desde IPs anónimas, se bloquea el acceso.

En el contexto del acceso condicional de Azure AD,

- "Si ocurre esto" se denomina declaración de condición
- "Se debe hacer esto" se denomina controles.

La combinación de una condición con los controles representa una directiva de acceso condicional.

Directiva de Acceso Condicional	
Condición	Control

Controles

En una directiva de acceso condicional, los controles definen qué es lo que debe suceder cuando se cumple una condición. Con estos controles se bloquea o permite el acceso con requisitos adicionales. Cuando la directiva permite el acceso, debemos seleccionar al menos un requisito.

Existen dos tipos de controles:

- Controles de concesión: determinan los requisitos para que un usuario pueda autenticarse e iniciar sesión, por ejemplo MFA, dispositivo unido a dominio, o dispositivo administrado:

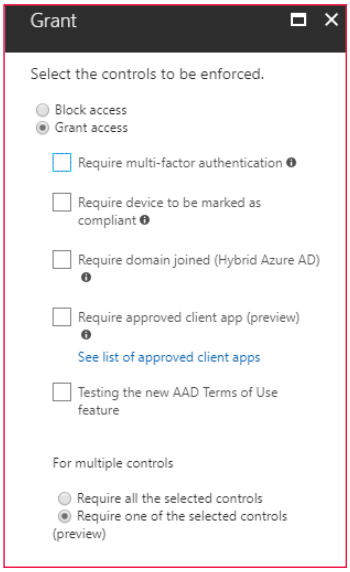


Figura 11.- Controles de Acceso Condicional.

- Controles de sesión: Azure AD envía información del dispositivo a la aplicación, y es ésta la que limita la experiencia del usuario. De momento sólo disponible para SharePoint online.

Para más información sobre cómo utilizar el Acceso Condicional de Azure AD para controlar el acceso a SharePoint online:

https://support.office.com/es-es/article/Controlar-el-acceso-desde-dispositivos-no-administrados-5ae550c4-bd20-4257-847b-5c20fb053622?ui=es-ES&rs=es-ES&ad=ES

Declaración de condición

En las directivas de acceso condicional, se definen los criterios que es necesario satisfacer para que los controles se apliquen.

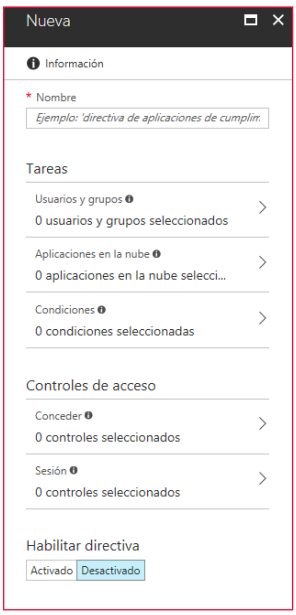


Imagen 12.- Condiciones (Tareas) y Controles.

Usuarios y grupos

Aquí se seleccionan los usuarios y grupos a los que aplica la directiva. También podemos seleccionar usuarios o grupos a excluir de la aplicación de la directiva.

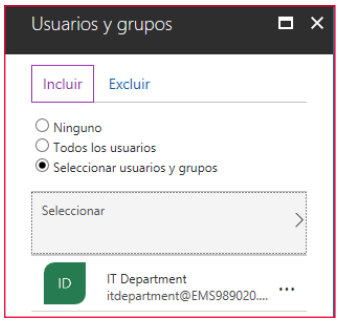


Imagen 13.- Usuarios y grupos.

Aplicaciones en la nube

Seleccionamos las aplicaciones a las que aplica la directiva. Aquí podremos seleccionar cualquier aplicación registrada en Azure AD, desde todas las apps de Office 365 y Microsoft online, hasta Azure Information Protection, administración de Azure, o aplicaciones propias.

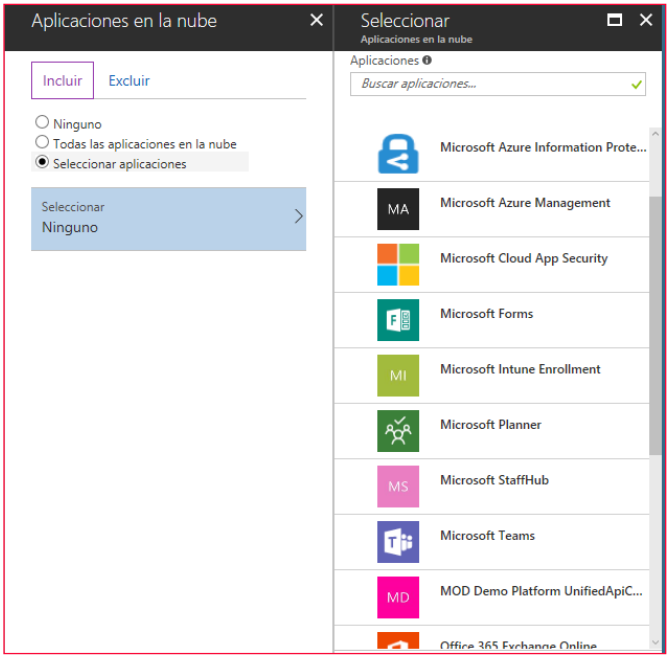


Imagen 14.- Aplicaciones afectadas por la directiva.

Ejemplo de Acceso Condicional aplicado a SharePoint online: exigir MFA a usuarios externos, usando Grupos Dinámicos.

https://blogs.technet.microsoft.com/kekirkpa/2017/10/13/enforcing-multi-factor-authentication-for-external-users-on-sharepoint-online/

Ejemplo de Acceso Condicional aplicado a Azure Information Protection: exigir MFA para el acceso móvil a documentos confidenciales

http://blogs.encamina.com/por-una-nube-sostenible/2017/10/27/azure-information-protection-acceso-condicional-a-documentos/

Condiciones extra

Además de usuarios y aplicaciones, hay otras condiciones que podemos configurar, como inicio de sesión en riesgo (Identity Protection), localizaciones (IPs de confianza de MFA), plataformas de dispositivos móviles, o aplicaciones cliente:

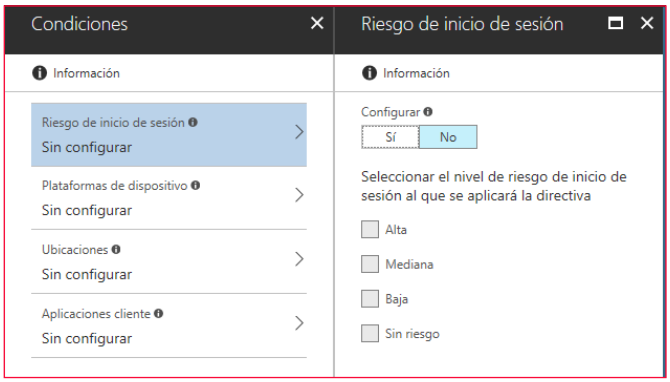


Imagen 15.- Condiciones extra.

¡Y listo! Habilitamos la directiva y ya comenzará a aplicarse.

Casos de uso

- Exigir MFA para las aplicaciones que tienen acceso a datos confidenciales.
- Exigir MFA para el acceso desde redes que no son de confianza.
- Acceso a Office 365 sólo desde dispositivos unidos a dominio, utilizando Intune.

https://www.microsoft.com/en-us/cloud-platform/conditional-access

Conclusión

Dentro de su firme apuesta por la seguridad, Microsoft está llevando la protección a todos los rincones de la nube. También a las identidades. En este artículo he querido dar un paseo por algunas de las complejas características de seguridad de Azure AD. Pero hay más. Mucho más.

PABLO ORTIZ BAIARDO
Cloud Solutions Specialist
ortiz.pablo@gmail.com
@portiz2017
http://blogs.encamina.com/por-una-nube-sostenible/
https://www.linkedin.com/in/portiz

16

Integrando un sistema IA de ayuda con el servicio QnA Maker en MS Access

Los usuarios de una aplicación suelen tener dudas sobre el manejo de la misma. La respuesta típica a esta necesidad es crear una guía de ayuda que explica todos los pormenores del programa. Ya tenemos pues nuestro manual, ya sea impreso o digital, con un fabuloso índice y unos elaborados marcadores (si es una edición digital) que intentan facilitar la búsqueda de información que el usuario requiere.

¿Y si fuera posible que dicho usuario hiciera una pregunta directamente, en lenguaje natural, en nuestra aplicación MS Access, y obtuviera la respuesta conveniente? Teniendo en cuenta que, con toda probabilidad, no habrá dos usuarios que formulen la misma pregunta de la misma manera. Y, yendo un poco más allá, ¿y si los propios usuarios fueran los que nos ayudaran a "depurar" los mensajes de ayuda mediante un proceso de retroalimentación?

La primera pregunta es lo que podemos conseguir implementando en nuestra aplicación el Cognitive Service llamado QnA Maker, y es lo que intentaré explicar en este artículo. Y en el conjunto de dicha explicación intentaré esbozar un sistema para dar respuesta a esa segunda pregunta, lo que me permitirá introducir uno de los elementos que intervienen en la configuración del servicio.

QnA Make: Visión Previa y Creación de nuestra BD de conocimiento

Lo primero que debemos hacer es disponer de una cuenta, ya sea personal o corporativa, de Microsoft. Nos dirigiremos a la página https://qnamaker.ai/ y nos pedirá el login. Entramos con los datos de nuestra cuenta y ya tenemos acceso al sistema de forma gratuita, aunque con limitaciones de cuota y uso: nuestra base de datos "de conocimiento" no puede superar los 20MB. Asimismo, cada clave de suscripción (hablaremos de ello más adelante) admite un máximo de 10.000 transacciones al mes y 10 transacciones por minuto.

Una vez hayamos entrado en la interface de QnA Maker nos encontraremos con diferentes apartados: (1) Visión preliminar del servicio / (2) Relación de servicios que hayamos creado (podemos crear más de uno) / (3) Creación de un nuevo servicio / (4) Documentación y ayuda / (5) Envío de nuestros comentarios a Microsoft.

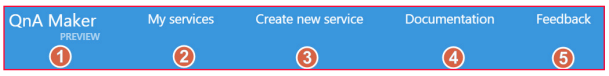


Imagen 1 - Opciones de QnA Maker

Dado que el ejemplo a través del cual desarrollaremos esta explicación va a ser un sistema de ayuda en Access, lo que haremos será, como primer paso, crear un nuevo servicio (3) , al cual he puesto de nombre AyudaAccess.

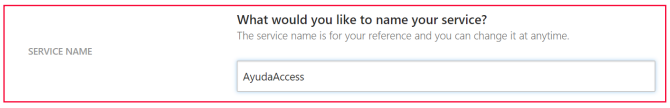


Imagen 2 .- Dando nombre a nuestro servicio.

Como segundo paso deberíamos crear nuestros pares de pregunta-respuesta. Eso lo podemos hacer de tres maneras: 1-Proporcionando una URL donde tengamos un sistema tipo FAQ / 2-Subiendo un archivo (cuidado, pues el peso del archivo no puede superar los 2MB) / 3-Creando los pares de manera manual. Eso es lo que nos pide la página web en a la que acabamos de acceder.

Como vamos a aprovechar las características de Access, en este punto simplemente lo que hacemos será crear el servicio a través del botón "Create", a pie de dicha página web.

Hecho esto nos encontraremos con una pantalla a la que tendremos que recurrir varias veces. Sus elementos son los que podéis ver en la ilustración

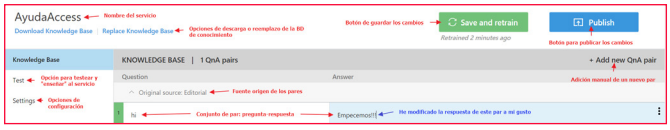


Imagen 3.- Elementos de control de la BD de conocimiento de nuestro servicio.

Si hacemos algún cambio debemos, primero, hacer clic sobre el botón para guardar los cambios, y a continuación en el botón para publicarlos. Eso nos llevará a una pantalla donde se nos informará de las diferencias derivadas de nuestras manipulaciones, tal y como se muestra en la ilustración siguiente.

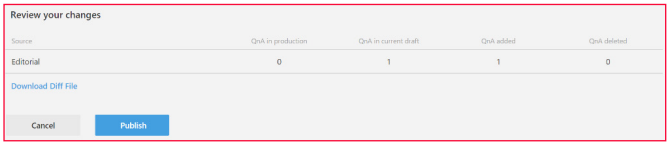


Imagen 4.- Resultados comparativos de nuestros cambios en la BD de conocimiento.

Es importante no olvidarse de volver a clicar sobre el botón "Publish" para que esos cambios pasen a definitivos.

Ya tenemos nuestro servicio creado. Ahora lo que necesitamos son las claves de suscripción y el identificador de

nuestra base de datos de conocimiento. Y esta información la conseguiremos en la siguiente ventana que nos aparece tras publicar los cambios, donde se nos da una muestra de los elementos que necesitamos para poder acceder al servicio a través de REST (importante guardar esa información para, después, saber cómo configurar el código en VBA Access).

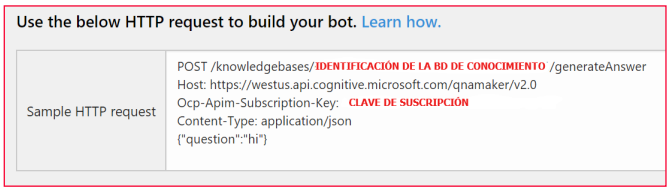


Imagen 5.- Ejemplo de llamada al servicio a través de REST.

Hay que tener en cuenta que si creamos más de un servicio lo que nos cambiará será el identificador de la BD de conocimiento; la clave de suscripción será siempre la misma. Y permitidme hacer un inciso, dado que, realmente, disponemos de dos claves de suscripción: una primaria y una secundaria. Para poder obtener estas claves basta con irnos a la parte superior derecha y clicar sobre nuestro nombre de usuario. Se nos desplegará un menú con la opción que nos interesa.

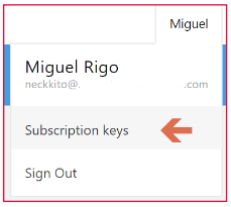


Imagen 6.- Obtención de las claves de suscripción primaria y secundaria.

Si seleccionamos la opción Mis Servicios (My Services) veremos todos los servicios que hayamos creado

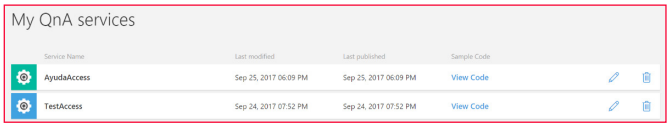


Imagen 7.- Lista de los servicios creados.

"Microsoft está actualmente desarrollando sistemas de Inteligencia Artificial bajo la denominación de Cognitive Services"

Estructurando nuestro Access

Antes de entrar en la explicación comentaros que, al final del artículo, os dejaré un enlace donde podréis bajaros la base de datos de ejemplo para que podáis ver estructura y código a vuestro gusto (aunque sin los datos identificativos del servicio. Ahí vosotros deberíais poner los vuestros).

Dado que la idea es integrar al máximo todo el proceso en nuestro Access os indico a continuación las tablas que he creado, su estructura y su finalidad:

TABLA	ESTRUCTURA	FINALIDAD
TPares	TPares Nombre del campo / Tipo de datos IdPar / Autonumeración Par_Pregunta / Texto corto Par_Respuesta / Texto largo	Construirnos los pares y llevar un control de los mismos. Importante: indexar el campo [Par-Respuesta]
TServicios	TServicios Nombre del campo / Tipo de datos IdServicio / Autonumeración NomServicio / Texto corto BaseDatos / Texto corto AutentKey / Texto corto	Llevar un control de los datos clave para acceder al servicio. En este ejemplo solo utilizaré un servicio para evitar complicar en exceso el ejemplo
TErrores	TErrores Nombre del campo / Tipo de datos IdError / Autonumeración NotifError / Texto corto FechaError / Fecha/Hora DescripError / Texto corto ProcError / Texto corto / Procedencia del error	Gestionar tanto los errores de Access como los errores devueltos por la respuesta del servicio
TNotificaciones	TNotificaciones Nombre del campo / Tipo de datos IdNotif / Autonumeración NotifQuien / Texto corto FechaNotif / Fecha/Hora RespuestaNotif / Número PreguntaNotif / Texto corto ValidezNotif / Texto corto RevisadaNotif / Sí/No	Nos permitirá recibir el feedback de los usuarios y nos servirá de base para "entrenar" nuestro servicio
TCodificacion	TCodificacion Nombre del campo / Tipo de CaractOriginal / Texto corto CaractCorrecto / Texto corto	Necesaria para soslayar el problema de la codificación en español.

¿Cuál va a ser la interfaz mínima que necesitaremos? Pues lo más básico será un cuadro de texto para que el usuario pueda introducir su pregunta, otro cuadro de texto para que el usuario pueda recibir la respuesta y un botón de comando para que el usuario pueda notificarnos que no está conforme con la respuesta dada.

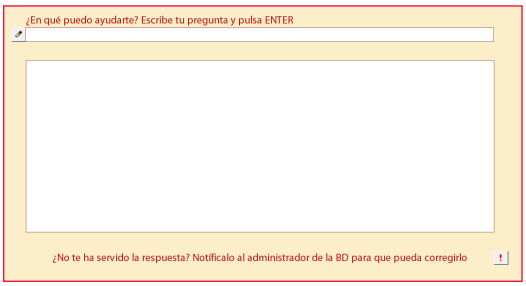


Imagen 8.- Estructura básica de formulario para el sistema.

Permitidme insistir: si utilizamos la BD de ejemplo, no debemos olvidarnos de introducir, en TServicios, los datos identificativos de nuestra BD de conocimiento y clave de suscripción.

Preparando nuestros pares y subiéndolos a nuestra BD de conocimiento

La preparación de los pares es tarea muy mecánica, pues simplemente basta ir rellenando los datos en nuestra tabla TPares. El único punto que debemos tener muy en cuenta es que, en las respuestas, debemos evitar el uso de los saltos de línea (en caso contrario se pierde la información tras el salto de línea en la respuesta).

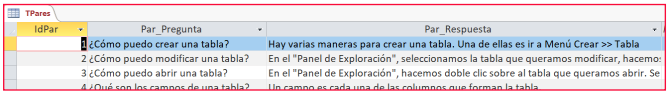


Imagen 9.- Ejemplo de pares pregunta-respuesta.

Una vez rellenada nuestra tabla de pares, y dado que el campo principal [IdPar] no nos interesa en estos momen-

tos, nos crearemos una consulta que nos recoja simplemente los campos [Par_Pregunta] y [Par_Respuesta]. Si vamos a realizar varias subidas podemos filtrar por [IdPar] (>UltimoSubido), por ejemplo, o cualquier otro sistema que se nos ocurra.

El siguiente paso es exportar esa consulta a un archivo de texto. En las características de la exportación debemos indicar que se trata de un archivo delimitado, que el delimitador es la tabulación y que el cualificador de texto es "ninguno".

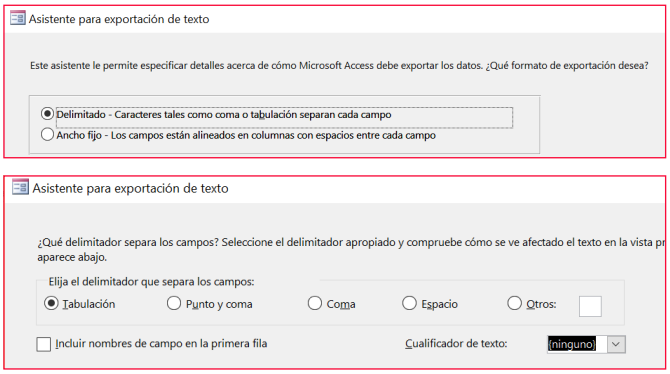


Imagen 10.- Exportación e la consulta.

Obtenido nuestro archivo de texto lo abrimos con el bloc de notas y lo volvemos a guardar, sobrescribiéndolo, pero indicándole que queremos la codificación UTF-8.

Imagen 11.- Guardando el archivo como UTF-8.

¿Qué problema tendremos? El servicio está pensado para inglés y, en la propia información del mismo, se indica que para otros idiomas se puede utilizar la codificación UTF-16. Sin embargo, en las pruebas que he realizado no hay manera de conseguir que la devolución de la respuesta (que veremos más adelante) retorne con caracteres reconocibles para los acentos, la eñe, las comillas, etc. Quizá en versiones posteriores de QnA Maker este aspecto se "internacionalice" un poco más. De cualquier manera, a lo largo del texto os indicaré cómo solventar el problema de la codificación.

Ya estamos listos pues para volver a nuestra cuenta de QnA Maker y añadir nuestros pares. El proceso para realizar lo anterior es:

- En mis servicios, editamos el servicio a través del icono del lápiz, a la derecha.
- Seleccionamos la opción de configuración: Settings.
- Elegimos la opción que nos permite cargar un archivo. Seleccionamos nuestro archivo de texto.
- Guardamos y publicamos >> Si estamos conformes volvemos a publicar. Nuestros pares deberían haberse incorporado a nuestra BD de conocimiento.

Y ya que hemos hablado de codificación lo que haremos será introducir, en TCodificacion, los valores "extraños" devueltos y su equivalencia. Utilizo este sistema de tabla porque si nos aparece, en diferentes pruebas, algún carácter no codificado, la adición del mismo y su correspondencia en dicha tabla soluciona el problema, sin necesidad de tocar el código en absoluto.

La ilustración siguiente muestra la correspondencia para un archivo subido con codificación UTF-8.

TCodificacion

Caráter original	Carácter correcto
&#193;	Á
&#201;	É
&#205;	Í
&#209;	Ñ
&#211;	Ó

Imagen 12.- Ejemplo de datos en TCodificacion

Programando nuestro código VBA

El código se halla en un módulo estándar, que he llamado mdlQ_A(). Los resultados se recogen en tres variables públicas, a saber: una variable que recoge la respuesta, una variable que recoge el porcentaje de validez de la respuesta (en formato texto) y una booleana que recoge la existencia de algún error durante el proceso.

Dado que los errores que pueden producirse son de dos tipos, y dado que en el sistema que os propongo recogemos dichos errores en una tabla, he creído conveniente estructurar el código en una especie de "cascada", en términos genéricos. Me explico: la primera tipología de error viene dada por los errores propios de VBA. Hasta aquí ningún secreto. Sin embargo, la segunda tipología viene dada por la respuesta del servicio. En este segundo caso, como obligatoriamente me veía forzado al análisis de la respuesta, he aprovechado el código para determinar la existencia de una respuesta correcta o de un error. Esquemáticamente sería algo así:

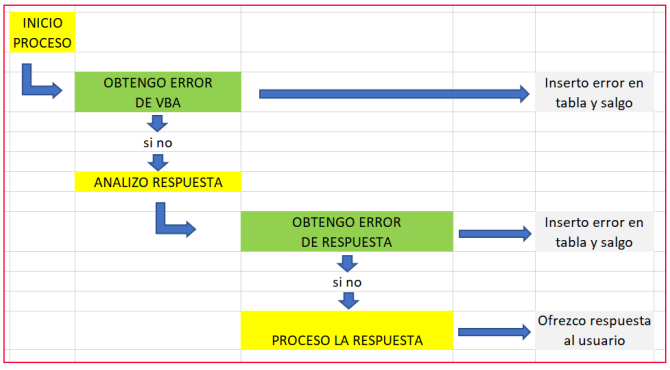


Imagen 13 - Esquema básico del proceso.

El inicio del proceso está situado en el evento "Después de actualizar" del cuadro de texto en el que el usuario escribe su pregunta, cuyo código sería:

```
Private Sub txtPregunta_AfterUpdate()
    Me.txtRespuesta = Null
    p_hayError = False

    If IsNull(Me.txtPregunta) Then Exit Sub

    Call subQ_A(Me.txtPregunta)

    If p_hayError = False Then
        Me.txtRespuesta = p_msgFinal
    End If
End Sub
```

Lo que nos lleva a ver que ya utilizamos dos de las variables públicas que comentaba unas líneas más arriba: p_hayError (booleana), y p_msgFinal (texto).

El código hace una llamada al procedimiento subQ_A(), pasándole como argumento la pregunta realizada por el usuario. Ese código es el siguiente:

```
Public p_msgFinal As String
Public p_porcValidez As String
Public p_hayError As Boolean

Public Sub subQ_A(laPregunta As String)
On Error GoTo sol_err

    Dim KnowBase As String
    Dim SuscrKey As String
    Dim miUrl As String
    Dim miXmlHttp As Object

        'Obtenemos los valores de la base y de la clave
    KnowBase = DLookup("BaseDatos", "TServicios")
    SuscrKey = DLookup("AutentKey", "TServicios")

        'Creamos el objeto
    Set miXmlHttp = CreateObject("MSXML2.serverXMLHTTP")

        'Construimos la url de conexión
    miUrl = "https://westus.api.cognitive.microsoft.com/qnamaker/v2.0/knowledgebases/" _
        & KnowBase & "/generateAnswer/"

        'Iniciamos el servicio REST
    With miXmlHttp
        .Open "POST", miUrl, False
        .setRequestHeader "Content-Type", "application/json"
        .setRequestHeader "Ocp-Apim-Subscription-Key", SuscrKey
        .send "{" & """ & "question" & """ & ":" & """ & laPregunta & """ & "}"
            'Analizamos la respuesta por si se generan errores
        If fncGestionoError(.responseText) = True Then
            MsgBox "Se ha producido un error. Contacte con el administrador de la aplicación", vbCritical, "ERROR"
            p_hayError = True
            Exit Sub
        Else
            If p_msgFinal = "No good match found in the KB" Then
                p_msgFinal = "No se ha podido encontrar una respuesta adecuada a su pregunta. Por favor," _
                    & " notifíquelo al administrador con el botón correspodiente" & vbCrLf & vbCrLf _
                    & "Gracias por su colaboración"
            Else
                p_msgFinal = fncProcesoCodificacion(p_msgFinal)
            End If
        End If
    End With
```

```
        'Eliminamos la instancia
    Set miXmlHttp = Nothing

    Exit Sub

sol_err:
    If Err.Number = -2147012889 Then 'No se pudo establecer conexión
        Call subNotificoError(Err.Description, "Mod: mdlQ_A; P/F: subQ_A")
    Else 'Cualquier otro error
        Call subNotificoError(Err.Description, "Mod: mdlQ_A; P/F: subQ_A")
    End If
    MsgBox "Se ha producido un error de proceso. Contacte con el administrador de la aplicación", vbCritical, "ERROR"
End Sub
```

Fijémonos que lo que hago es buscar la información del identificador de la base de conocimiento y de la clave de suscripción en la tabla correspondiente, y con el primero me creo la cadena de conexión para el servicio REST. Y, si recordamos la ilustración 5 con el ejemplo que nos proporcionó QnA Maker, vamos configurando el objeto HTTP en sus cabeceras (importante la clave de suscripción), ejecutando un Send() que, como veis, debe tener muy en cuenta la existencia de comillas dobles en la construcción de su estructura.

"los usuarios de una aplicación suelen tener dudas sobre el manejo de la misma"

En todo este proceso ya hemos gestionado la inserción de los errores de VBA a través de la llamada al procedimiento subNotificoError(), cuyo código es el siguiente:

```
Private Sub subNotificoError(elError As String, laProc As String)
        CurrentDb.Execute ("INSERT INTO TErrores(NotifError, FechaError, DescripError, ProcError) VALUES ('" _
        & Environ("username") & "','" & Date & "','" & Replace(elError, "'", "''") & "','" & laProc & "')")
End Sub
```

Como veis, no es más que una SQL de inserción, donde me he tomado la libertad de informar "por detrás" a quién le ha saltado el error a través del uso de Environ().

Otra característica a destacar es que la respuesta del servicio, en caso de error, puede contener comillas simples. Es por ello por lo que utilizo la función Replace() para evitar un error de SQL.

Siguiendo con el código de subQ_A(), el bloque If, a través de la función fncGestionoError(), es quien continúa con el proceso para analizar el mensaje recibido por el servicio. Si la función devuelve True se gestiona el error, esta vez del servicio, y, si no, se gestiona la respuesta, teniendo en cuenta el caso particular en el que no se encuentre una equivalencia, dado que el mensaje retornaría "No good match found in the KB", en inglés.

La función fncGestionoError() es, en realidad, un "puente" que me permite detectar el error devuelto por el servicio, si es el caso, y saber cómo debo procesar el mensaje; esto es, qué palabra clave debo buscar dentro del mensaje para obtener una respuesta legible para el usuario. Su código es, simplemente:

```
Private Function fncGestionoError(laRespuesta As String) As Boolean
    If InStr(laRespuesta, "error" & """ & ":") > 0 Then
        Call subProcesoJSon(laRespuesta, True)
        fncGestionoError = True
    Else
        Call subProcesoJSon(laRespuesta, False)
    End If
End Function
```

Y esta a su vez (de ahí lo que os comentaba del sistema "en cascada"), hace una llamada a un procedimiento denominado subProcesoJSon(). Como habéis intuido, la devolución del servicio es un JSON. Podríamos insertar un parseador en nuestro código para procesar ese JSON, pero dado que no requerimos grandes extracciones de datos me he decantado por realizar un sistema más manual.

Veamos: si el JSON devuelve error se recibe un mensaje del tipo:

Inmediato

```
{
  "error": {
    "code": "NotFound",
    "message": "The knowledge base does not exist."
  }
}
```

Imagen 14 - Respuesta del servicio con error

Es decir, que en mensaje aparece el conjunto de caracteres (error":), que es el que utilizo para determinar si hay error en el mensaje, y la palabra "message", que es la que me da la descripción del error (y ese es el texto que introduciré en la tabla TErrores).

Analizo esos caracteres (error":) así porque si en la respuesta hubiéramos incluido la palabra "error" el código pensaría que hay un error, cuando eso no sería correcto. En consecuencia, en las respuestas, debemos evitar incluir el constructo (error":).

Por otra parte, si el mensaje llega sin errores obtendremos una respuesta del tipo:

Inmediato

```
{
  "answers": [
    {
      "answer": "Hay varias maneras para crear una tabla. Una de ellas es ir a Men&#250; Crear &gt;&gt; Tabla",
      "questions": [
        "&#191;C&#243;mo puedo crear una tabla?"
      ],
      "score": 84.0520027577877
    }
  ]
}
```

Imagen 15 - Respuesta correcta del servicio.

De la anterior ilustración podemos extraer varias conclusiones:

- La palabra clave no puede ser “answer”, dado que previamente a esa palabra completa tenemos “answers”. Debo pensar en construir otra palabra clave que vaya directa al grano. Eso lo consigo construyendo como palabra clave answer”:
- Aparece otra palabra clave, que es “score”, que nos da el porcentaje de validez que el servicio “piensa” que ha alcanzado. En otras palabras, el tanto por cierto que cree que ha acertado en la respuesta.
- La codificación nos juega una mala pasada, dado que no nos devuelve los caracteres para nosotros correctos (por ejemplo, Men&#250 en lugar de Menú).

Con todos esos puntos en mente veamos qué hace el procedimiento subProcesoJSon():

```
Private Sub subProcesoJSon(laRespuesta As String, hayError As Boolean)
    Dim laPos As Integer
    Dim laResp As String
    Dim elPorc As String
    Dim codeProcess As String

    If hayError = True Then
        codeProcess = "message"
    Else
        codeProcess = "answer" & """" & ":"
    End If

        'Determinamos el inicio del mensaje que nos interesa
    laPos = InStr(laRespuesta, codeProcess)
    laPos = laPos + 8 'Ajustamos la posición hasta el texto del mensaje
    laResp = Right(laRespuesta, Len(laRespuesta) - laPos)
    laPos = InStr(laResp, """") 'El mensaje aparece tras las primeras comillas dobles
    laResp = Right(laResp, Len(laResp) - laPos)
        'Determinamos el final del mensaje de error
    laPos = InStr(laResp, """") 'El mensaje acaba con comillas dobles
    laResp = Left(laResp, laPos - 1)

    If codeProcess <> "message" Then
        laPos = InStr(laRespuesta, "score")
        laPos = laPos + 7
        elPorc = Mid(laRespuesta, laPos, 8)
    End If

    If hayError = True Then
            'Informamos del error en la tabla TErrores
        Call subNotificoError(laResp, "mdlQ_A; P/F: fncProcesoJSon")
    Else
        p_porcValidez = elPorc
        p_msgFinal = laResp
    End If
End Sub
```

El primer bloque If me discrimina por qué palabra clave debo buscar, y a partir de ahí ajusta el texto de la respuesta, sabiendo que los mensajes que debo capturar empiezan y acaban con comillas dobles.

“dejaré un enlace donde podréis bajaros la base de datos de ejemplo para que podáis ver estructura”

A continuación, y si no se ha producido error, aprovecho para capturar el porcentaje de validez a través de una variable local.

Finalmente, y en el último bloque If, si hay error lo introduzco en la tabla TErrores. No necesito procesar su codificación dado que capturo el mensaje de error literal en inglés. Si no, asigno los valores del porcentaje y de la respuesta final a las variables públicas.

No obstante, la respuesta final podría ser no entendible por los problemas comentados de codificación. Sin embargo, y si recordamos el procedimiento subQ_A(), determinamos que si todo ha ido bien y el mensaje devuelto no ha sido “No good match found in the KB”, reasigno el valor de la variable pública p_msgFinal a un nuevo valor “tratado” por la función fncProcesoCodificacion(), cuyo código es el siguiente:

```
Private Function fncProcesoCodificacion(elMsg As String) As String
    Dim rst As DAO.Recordset

        'Cambiamos los caracteres originales por los correctos. Si no se devuelve el mismo mensaje de entrada.
    If DCount("*", "TCodificacion") > 0 Then
        Set rst = CurrentDb.OpenRecordset("SELECT CaractOriginal, CaractCorrecto FROM TCodificacion")
        With rst
            .MoveFirst
            Do Until .EOF
                If InStr(elMsg, .Fields(0)) > 0 Then
                    elMsg = Replace(elMsg, .Fields(0), .Fields(1))
                End If
                .MoveNext
            Loop
        End With
    End If

    fncProcesoCodificacion = elMsg

    If Not rst Is Nothing Then
        rst.Close
        Set rst = Nothing
    End If
End Function
```

Los resultados son espectaculares. A modo de ejemplo os muestro dos ilustraciones con los resultados obtenidos tras dos preguntas sobre lo mismo, pero formuladas de manera distinta (y diferente de la pregunta original del par):

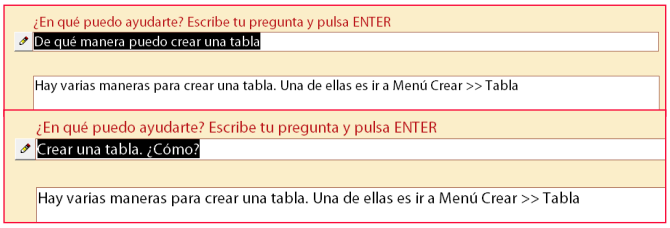


Imagen 16 – Resultados obtenidos.

Aprovechar el feedback del usuario y entrenando a nuestro servicio

Hemos comentado que le íbamos a dar al usuario la posibilidad de mostrar su disconformidad con la respuesta recibida, permitiéndole notificarnos que la ayuda recibida no ha sido útil. Esto es lo que hace el botón de comando de nuestro formulario, cuyo código es:

```
Private Sub cmdNotifica_Click()
    Dim elId As Long

    If IsNull(Me.txtPregunta) Then
        MsgBox "No hay pregunta que notificar", vbExclamation, "SIN DATOS DE PREGUNTA"
        Exit Sub
    End If

    elId = Nz(DLookup("IdPar", "TPares", "Par_Respuesta='" & Me.txtRespuesta & "'"), 0)

    CurrentDb.Execute ("INSERT INTO TNotificaciones(NotifQuien, FechaNotif, RespuestaNotif, PreguntaNotif, ValidezNotif) VALUES ('" _
        & Environ("username") & "','" & Date & "','" & elId & "','" & Me.txtPregunta & "','" & Trim(p_porcValidez) & "')")

    MsgBox "Se ha notificado al administrador su opinión. Gracias por su colaboración", vbInformation, "NOTIFICACIÓN CORRECTA"
End Sub
```

Dado que lo que guardamos es el identificador de la respuesta (para no duplicar el contenido de un texto largo), la obtención de la información para comprobar las notificaciones como administrador pasa por crearnos una consulta con, en mi caso, un right join para ver también aquellos mensajes huérfanos de respuesta en TPares.

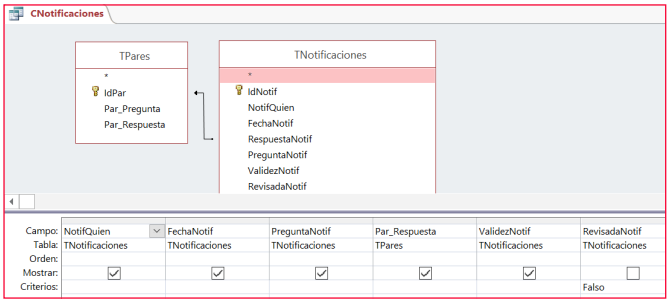


Imagen 17 - Estructura de consulta para comprobar las notificaciones de los usuarios.

Lo anterior no solo me permite ver la relación pregunta-respuesta_obtenida que no ha gustado al usuario (junto con el dato del porcentaje de validez), sino que con ello puedo volver a mi servicio y "entrenar" su respuesta. ¿Cómo? Supongamos que un usuario ha realizado una pregunta y ha notificado que no está conforme con el resultado. La consulta anterior nos mostrará la siguiente información:

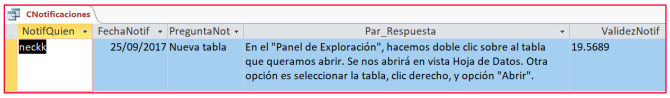


Imagen 18 - Ejemplo de registro de notificación.

Automáticamente ya podemos comprobar que, por una pregunta tan escueta, el servicio solo le otorga una validez de respuesta de poco más del 19%. Y, obviamente, la respuesta no es acorde con lo pedido.

Lo que haremos a continuación es dirigirnos a nuestro servicio en el navegador y acceder a la opción "Test" (ver ilustración 3). Planteamos ahí la misma pregunta, con lo que obtenemos

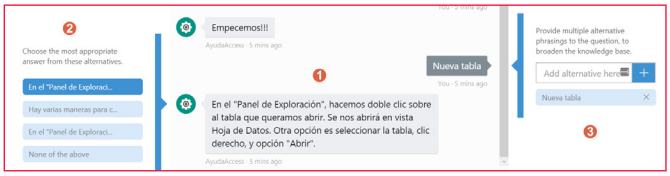


Imagen 19 - Panel de testeo de nuestro servicio.

En (1) podemos ver la pregunta y la respuesta obtenida. Y, dado que no la consideramos correcta, en (2) tenemos la posibilidad de elegir la respuesta que más se ajusta a dicha pregunta. Si la seleccionamos (en nuestro caso, "Hay varias maneras para c…") el servicio "aprenderá" cuál es la respuesta más adecuada para esta pregunta o variaciones similares.

Por otra parte, en (3), podemos añadir diferentes alternativas para esa pregunta en concreto.

Una vez "enseñada" la respuesta correcta (o, en su defecto, preguntas alternativas) no debemos olvidarnos de hacer clic sobre la opción de guardar ("Save and retrain") para salvar las modificaciones, y a continuación seguir todo el proceso de publicar dichas variaciones.

Ni que decir tiene que, hecho lo anterior, el usuario, ante la pregunta "Nueva tabla", obtendrá ahora la respuesta correcta.

Consideraciones finales

Ni que decir tiene que, por no alargar más este ya extenso artículo, no he profundizado en el uso de las API's del servicio. Para obtener la respuesta he recurrido a la API V1.0. Mas si ojeamos la documentación en su apartado "Api Reference" veremos que con la API V2.0 podemos manipular, a través de código, las distintas acciones que aquí os he enseñado a hacer manualmente. Para más información a dicha ayuda os remito.

Y como lo prometido es deuda aquí os dejo el enlace desde donde os podréis descargar la base de datos Access que he utilizado para esta explicación: http://bit.ly/CompartiMossQA

Conclusión

Creo que el uso de este servicio representa una alternativa muy atractiva a un sistema de ayuda convencional. Si la preparación de los pares nos ha parecido "fatigosa" parémonos a pensar un momento en el tiempo que nos consumiría crear un pdf completo de ayuda, con sus ilustraciones y demás elementos, además de lo que es el texto de ayuda en sí (aunque en ningún caso descarto la complementariedad de ambos sistemas). La ventaja añadida es que, formados los usuarios en el nuevo sistema, la retroalimentación que les solicitamos les involucra en el proceso, lo que, en términos generales, podría considerarse como un factor positivo.

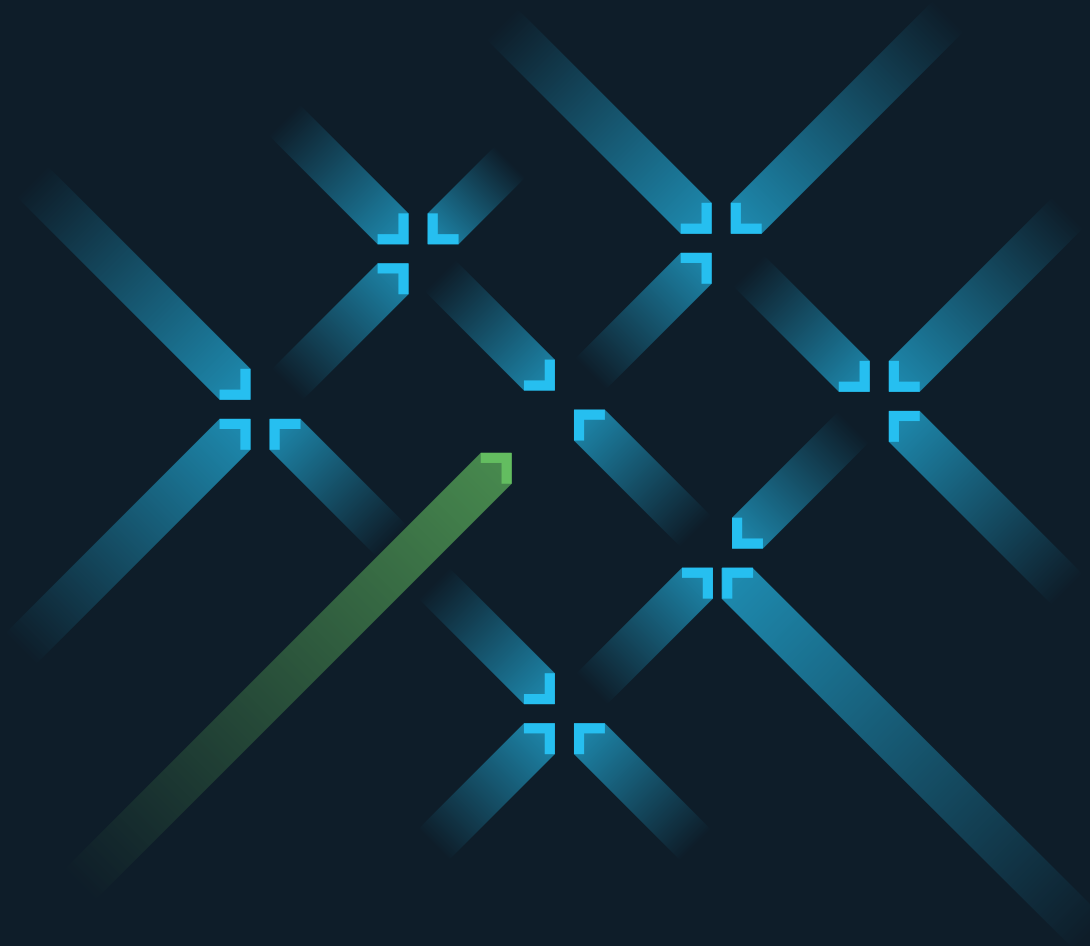
Y, por qué no decirlo (aunque quizás no sea políticamente correcto), el sistema nos permitiría sonreírnos ante la expresión asombrada del usuario que dice: "¿y con Access se puede hacer ESTO?".

Deseo que la explicación, si no útil, os haya resultado interesante.

MIQUEL "NECKKITO"
MVP Access
neckkito@outlook.com
@neckkito
http://bit.ly/neckkito

Your Data.
Your Office 365.

Smarter Migration, Reporting,
Security and Management.



Quadrotech

www.quadrotech-it.com

23

Haciendo uso de los Cognitive Services desde spfx

En este artículo, vamos a ver que ventajas nos aporta los MS Cognitive Services, y como podemos hacer uso de ellos desde soluciones basadas en el SharePoint Framework (SPFx). Durante el artículo, veremos cómo podemos registrar el servicio de Vision API, para posteriormente consumirlo desde una acción personalizada añadida a la Command Bar de una lista, haciendo uso de las capacidades de extensión de SPFx, en este caso con un List View Command Bar customiser. El resultado final podemos verlo en la imagen siguiente:

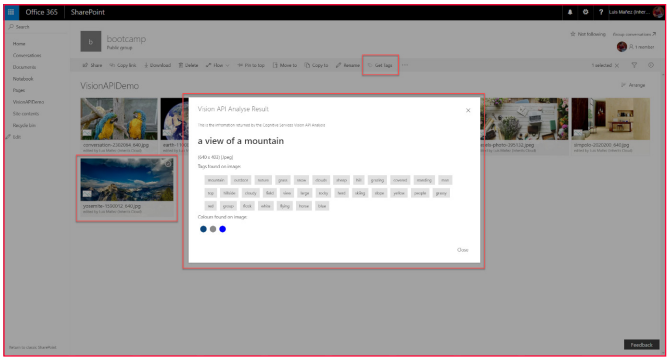


Imagen 1.- Resultado final obtenido al desplegar el List View Command Bar customizer.

Rápida introducción a los MS Cognitive Services

Los Cognitive Services son un servicio más dentro de Azure, que nos ofrece capacidades de Inteligencia Artificial, pero abstrayéndonos de gran parte de la dificultad de este tipo de sistemas de AI y Machine Learning. Son servicios ya entrenados por Microsoft, y expuestos a través de APIs sencillas. Los servicios están organizados en diferentes categorías principales:

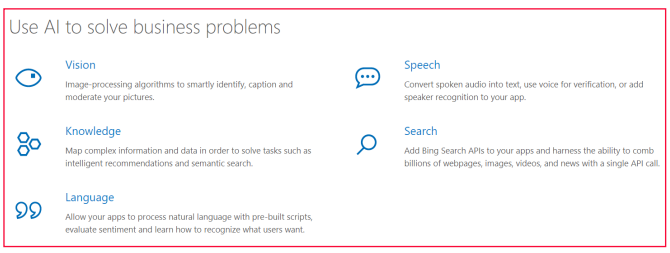


Imagen 2.- Categorías principales de Cognitive Services.

Podéis encontrar el detalle de estas categorías con sus diferentes servicios en el siguiente link:

https://azure.microsoft.com/en-us/services/cognitive-services/?v=17.42n

Algunos ejemplos son:

- Vision API: Este servicio ofrece capacidades de reconocimiento de imágenes y OCR. Este servicio será el utilizado para el artículo, y nos permitirá analizar una imagen almacenada en una biblioteca de SharePoint, y obtener información, como una descripción de la imagen, sus dimensiones, posibles tags, e incluso los colores dominantes que aparecen en la imagen.
- Text Analytics API: Permite analizar un texto y obtener posibles palabras clave, y también evalúa el sentimiento del mismo, por lo que puede distinguir si el texto está indicando un sentimiento positivo o negativo, algo que es muy útil para analizar feedback de usuarios o comentarios en redes sociales.
- L.U.I.S: Reconocimiento de lenguaje natural. Es la base de Cortana.
- Translator Text API: Ofrece servicios de traducción automática.

Registrar servicio Cognitive Services Vision API

Para poder utilizar la Vision API de Cognitive Services, primero tenemos que registrar el servicio desde el portal de Azure, para así obtener la Key para invocar la API.

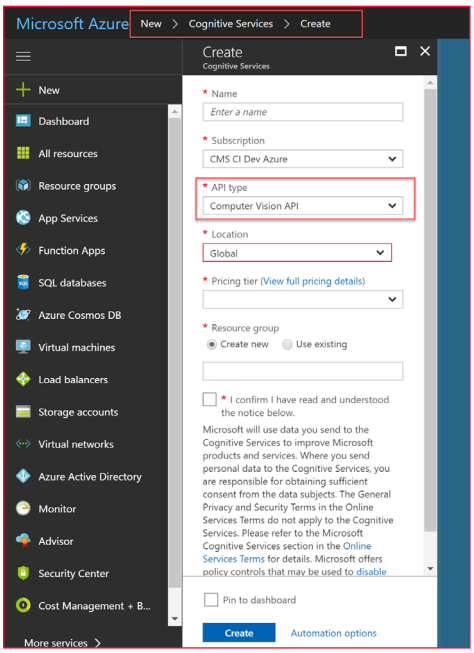


Imagen 3.- Registro de la API de Visio de Cognitive Services en el portal de Azure.

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

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

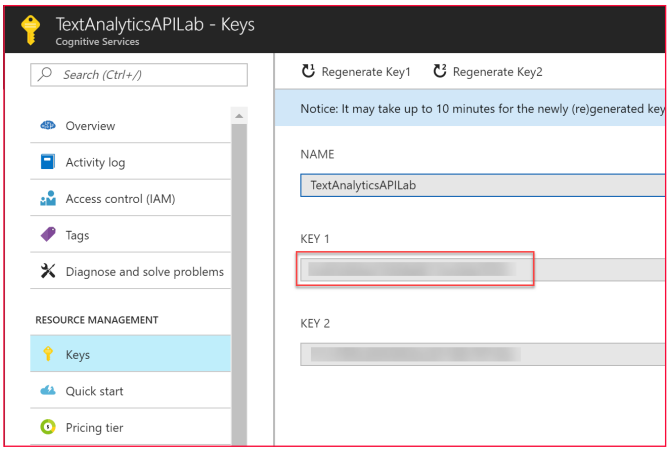


Imagen 4.- Obtención de la Key.

Introducción y uso de Tenant Properties en SharePoint Online

Desde hace unas pocas semanas, tenemos una nueva funcionalidad disponible en SharePoint Online, llamada Tenant Properties. Esto no es más que algo muy similar a las famosas y veteranas: Property Bags, así que nos va a permitir almacenar pares de clave/valor a nivel de Tenant, y que luego podremos recuperar utilizando una API disponible. De esta manera, nos evitaremos hard-codear el valor de la Key en el mismo JavaScript.

Nota: En el momento de escritura del artículo, esta nueva funcionalidad está sólo disponible para Tenants en modo "First Release".

Para almacenar una Tenant Property, debemos hacer uso de los comandos de PowerShell para SharePoint Online (en su última versión).

Con los comandos instalados, abrimos la consola de comandos de SharePoint Online:

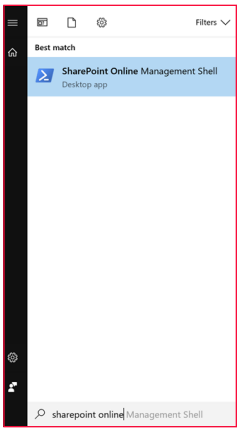


Imagen 5.- SharePoint Online Management Shell.

Primero, nos conectamos a nuestra Tenant (nos preguntara usuario y password). Nota: es necesario conectar a la URL de Admin:

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

Para almacenar la Key (Las Tenant Properties se almacenan a nivel de AppCatalog, por lo que el parametro site debe especificar la URL del AppCatalog)

```
Set-SPOStorageEntity -Site "https://YOUR_TENANT.sharepoint.com/sites/appcatalog" -Key "TextAPIKey" -value YOUR_KEY_HERE -Description "Key to use Text API" -Comments "O365 Dev Bootcamp 2017"
```

Podemos comprobar que la propiedad se ha creado correctamente con el siguiente comando:

```
Get-SPOStorageEntity -Site "https://YOUR_TENANT.sharepoint.com/sites/appcatalog" -Key "TextAPIKey"
```

Invocando Vision API desde spfx

Para el artículo, estamos utilizando una ListView Command Set extension, pero sería similar si lo hacemos desde un webpart.

"podemos hacer uso de ellos desde soluciones basadas en el SharePoint Framework (SPFx)"

Primero de todo, necesitamos recuperar la API Key, que en el paso anterior hemos almacenado en una Tenant Property:

```
// Getting Vision API Key from Tenant Properties (First Release only: https://docs.microsoft.com/en-us/sharepoint/dev/spfx/tenant-properties)
if (this.cognitiveServicesKey === '') {
  this.context.spHttpClient.get( `${this.context.pageContext.web.absoluteUrl}/_api/web/GetStorageEntity('VisionAPIKey')`, SPHttpClient.configurations.v1)
  .then((response: SPHttpClientResponse) => {
    response.json().then((responseJson: any) => {
      //console.log(LOG_SOURCE, responseJson);
      this.cognitiveServicesKey = responseJson.Value;
    });
  });
}
```

El siguiente paso consiste en obtener la URL de la imagen almacenada en SharePoint. Esto sería muy sencillo, si no fuera porque dicha URL está securizada, así que no podemos enviársela tal cual a la Vision API. Una opción sería obtener la imagen como un array de bytes, pero desde TypeScript no es tarea sencilla, así que utilizaremos un pequeño truco para poder trabajar con una URL. Existe una API que nos devuelve cierta información sobre un ítem de una lista. La URL tiene este formato:

https://tenant.sharepoint.com:443/_api/v2.0/drives/b!IgNz_osK_E6YZuP28vyURyJvxwztrcpNuqPkLRSqlOaVA-TUy1EqMT7tfP4UIuLhU/items/01HBKUZTUQKQB7NFK4H-VHLU354TEWATCOP?version=Published&$select=@content.downloadUrl

Lo bueno es que esa URL, viene en un campo del ítem seleccionado, por lo que podemos sacarla fácilmente desde una SPFx extensión:

```
const imageInfoUrl = event.selectedRows[0].getValueByName('.spItemUrl') + '&$select=@content.downloadUrl';
```

Entre la diferente información proporcionada por esa API,

tenemos una campo “@content.downloadUrl”, que nos ofrece la URL de la imagen, incluyendo un Token de Autenticación. Esto significa que, a todos los efectos, esa URL es pública, y si la cargamos en cualquier navegador, nos va a descargar la imagen de SharePoint, sin solicitarnos credenciales. Un ejemplo de esta URL “pública” sería:

https://inheritscloud.sharepoint.com/sites/bootcamp/_layouts/15/download.aspx?UniqueId=f6035490-5c95-4e3d-ba6f-bc992c0989cf&Translate=false&tempauth=eyJ0eXAiOiJKV1QiLCJhbGciOiJub25lIn0.eyJhdWQiOiIwMDAwMDAwMy0wMDAwLTBmZjEtY2UwMC0wMDAwMDAwMDAwMDAvaW5oZXJpdHNjbG91ZC5zaGFyZXBvaW50LmNvbUBjZDhiMzdkOS01M2RlLTQ2MzEtYTE5Ni0yZDAxZGM4MTQyNDMiLCJpc3MiOiIwMDAwMDAwMy0wMDAwLTBmZjEtY2UwMC0wMDAwMDAwMDAwMDAiLCJuYmYiOiIxNTEwMTY3OTY1IiwiZXhwIjoiMTUxMDE3MTU2NSIsImVuZHBvaW50aXAiOiJsIjoiZzhJM0w5UGhMMjR2VGk4bXM1K29Iem9mS1ZqOG9iVjk-wTHAraFhkazBXVT0iLCJlbmRwb2ludHVybExlbmd0aCI6IjEzOSI-sImlzbG9vcGJhY2siOiJUcnVlIiwiY2lkIjoiTXpSbU16SmhPV1V0TWp-BME1TMDBNREF3TFdNMlpEa3RNamM0TXpBeFlqZzNNelF6Ii-widmVyIjoiaGFzaGVkcHJvb2Z0b2tlbiIsInNpdGVpZCI6IlptVTN-NekF6TWpJdE1HRTRZaTAwWldaakxUazROall0WlROObU5tWX-labU01TkRRMyIsIm5hbWVpZCI6IjAjLmZ8bWVtYmVyc2hpcHxs-dWlzQGluaGVyaXRzY2xvdWQuY29tIiwibmlpIjoibWljcm9zb2Z0Ln-NoYXJlcG9pbnQiLCJpc3VzZXIiOiJ0cnVlIiwiY2FjaGVrZXkiOiIwaC-5mfG1lbWJlcnNoaXB8MTAwMzdmZmVhMDhiOTg2NEBsaXZlL-mNvbSIsInR0IjoiMCIsInVzZVBlcnNpc3RlbnRDb29raWUiOiIyIn0.SkNYbmlKenJ2a1InNnF3RzFkcHlaenVOSGxBVUR1cS9RR-WpNYmt3Tkt1ND0&ApiVersion=2.0

Como veis, el parámetro “tempauth” es un token temporal de autenticación. Es un token oAuth2 a todos los efectos.

Una vez obtenida dicha URL, ya la podemos enviar al servicio de Vision API.

Primero preparamos las cabeceras de la petición, donde metemos la Key de la API:

```
private _prepareHeadersForVisionApi(): Headers {
  const requestHeaders: Headers = new Headers();
  requestHeaders.append('Content-type', 'application/json');
  requestHeaders.append('Cache-Control', 'no-cache');
  requestHeaders.append('Ocp-Apim-Subscription-Key', this.cognitiveServicesKey);

  return requestHeaders;
}
```

Luego preparamos el body, donde incluimos la URL anterior:

```
private _prepareHttpOptionsForVisionApi(imageDownloadUrl: string): IHttpClientOptions {
  const body: string = JSON.stringify({
    'Url': imageDownloadUrl
  });

  const httpOptions: IHttpClientOptions = {
    body: body,
    headers: this._prepareHeadersForVisionApi()
  };

  return httpOptions;
}
```

Finalmente, enviamos la petición al servicio, y mapeamos el JSON obtenido, a un objeto de nuestro modelo:

```
const httpOptions: IHttpClientOptions = this._prepareHttpOptionsForVisionApi(downloadUrl);

const cognitiveResponse: HttpClientResponse = await this.context.httpClient.post(this.cognitiveServicesVisionUrl, HttpClient.configurations.v1, httpOptions);
const cognitiveResponseJSON: any = await cognitiveResponse.json();

return this._toCognitiveServicesImage(cognitiveResponseJSON);
```

> ***“sumando la potencia de Cognitive Services y spfx, podemos dar una capa más de inteligencia a nuestro SharePoint.”***

Finalmente, con nuestro modelo, hacemos uso de varios componentes de ReactJS para mostrar el resultado en un Dialog:

```
public render(): React.ReactElement<ICognitiveServicesImageDialogContentProps> {

    return (<div className={styles.cognitiveServicesImageDialogContent}>
        <DialogContent
            title={"Vision API Analyse Result"}
            subText={"This is the infromation returned by the Cognitive Services Vision API Analysis"}
            onDismiss={this.props.close}
            showCloseButton={true}
            type={DialogType.close} >

            <h1>{this.props.cognitiveServicesImage.description.captions[0].text}</h1>

            <Label>({this.props.cognitiveServicesImage.metadata.width} x {this.props.cognitiveServicesImag

            <Label className={styles.clear}>Tags found on image:</Label>
            <DocumentCardTags tags={this.props.cognitiveServicesImage.description.tags}></DocumentCardTags

            <Label className={styles.clear}>Colours found on image:</Label>
            <SwatchColorPicker
                columnCount={3}
                cellShape={'circle'}
                colorCells={
                    [
                        { id: 'a', label: 'accentColor', color: `#${this.props.cognitiveServicesImage.colo
                        { id: 'c', label: 'dominantColorBackground', color: this.props.cognitiveServicesIm
                        { id: 'd', label: 'dominantColorForeground', color:  this.props.cognitiveServicesI
                    ]
                } />

            <DialogFooter>
                <CommandButton text='Close' title='Close' onClick={this.props.close} />
            </DialogFooter>
```

Tenéis todo el proyecto en mi cuenta de GitHub:

https://github.com/luismanez/sp-dev-fx-extensions/tree/react-command-vision-api-v2/samples/react-command-vision-api

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

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

Resumen

Como veis, sumando la potencia de Cognitive Services y spfx, podemos dar una capa más de inteligencia a nuestro SharePoint.

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

Entrevista Vladimir Medina

Mi nombre es Vladimir Medina, nací en México y desde 2005 estoy enfocado en tecnologías Microsoft, específicamente con SharePoint y Office 365.

Trabajé para Microsoft como Premier Field Engineer por 5 años y actualmente me desempeño como Consultor independiente apoyando a diferentes empresas, Partners y a Microsoft en proyectos de SharePoint y Office 365, capacitación especializada y eventos de comunidad.

Cuento con diversas acreditaciones de Microsoft como Master Trainer, Trainer of Trainers, Technical Mentor, SharePoint Subject Matter Expert (SME) y Performance monitoring & debugging Engineer; también he sido certificado como MCP, MCTS, MCITP y MOF 4.0. Finalmente he sido reconocido por Microsoft como Most Valuable Professional (MVP) en dos periodos diferentes, del 2009 al 2010 y del 2016 a la fecha.

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

Comencé en este mundo de la tecnología y específicamente con SharePoint debido a que hice mi servicio social en la Organización Mundial de la Salud (OMS), como parte de mi titulación en la carrera de Ingeniería en Computación en la UNAM. En la oficina de la OMS en México tenían SharePoint Portal Server 2001 como plataforma para la intranet local y me toco administrarla durante el servicio social, por esas mismas fechas surgió en México la Comunidad de SharePoint y me envolví en sus actividades en línea y presenciales, lo que me ayudo a desarrollar mis habilidades y experiencia profesional.

Ya concluida mi formación académica seguí en el mismo camino de SharePoint trabajando para diferentes Partners de Microsoft en el área de Colaboración, realizando proyectos de SharePoint y creciendo más como profesional.

A principios de 2010 acepté una oferta de trabajo en Microsoft Latinoamérica para desempeñarme como Premier Field Engineer en el área de Premier Support, un área especializada que se encarga de clientes con contrato de soporte premier, durante esta etapa estuve siempre dedicado a actividades de soporte proactivo y reactivo de SharePoint on-premise y SharePoint Online, que surgió en 2008. Concluí mi faceta en Microsoft a finales de 2014 y entonces comencé mi etapa actual como consultor independiente.

¿Cuáles son tus principales actividades tecnológicas hoy en día?

Debido a mi formación profesional en Microsoft, actualmente mi principal labor como consultor independiente se basa en mentoring y capacitación especializada, trabajo con diferentes empresas, Partners y con Microsoft mismo para dar apoyo especializado en proyectos, cursos, eventos, coaching y mentoring.

Además, como una de mis más grandes pasiones es ayudar a otros, participo como Profesor en una Universidad dando clases, donde, aprovechando la libertad de catedra, apoyo en la formación académica de nuevas generaciones de ingenieros en sistemas con temas que rara vez se consigue aprender en ese ámbito; yo en lo personal fomento los conocimientos de tecnologías como SharePoint, Office 365, PowerApps, Flow, Visual Studio, Xamarin, Windows Server, System Center, etc., así como metodologías como SCRUM, ITIL, MOF.

Como parte de esa misma pasión, soy Community Manager de la comunidad SharePoint en México y en Latinoamérica, donde principalmente me dedico a organizar y ejecutar eventos y charlas en toda la región, con la co-

munidad organizamos SharePoint Saturdays, Meetups, Simposios, entre otros; además de apoyar a Microsoft en eventos corporativos como los Tech Summits, Bootcamps, Ignite, etc.

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

Mi principal actividad, incluso por encima de las relacionadas con la tecnología, está en el ámbito espiritual, como Testigo de Jehová dedico la mayor parte de mi tiempo predicando y enseñando a otros acerca de las verdades que Dios transmite mediante la Biblia y que, al ponerlas en práctica, nos ayudan a todos a ser mejores seres humanos; estoy convencido de que esa es la mejor forma de aprovechar mi tiempo ahora y es por ello que es a lo que más me dedico.

¿Cuáles son tus hobbies?

Leer y estudiar.

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

A estas alturas de mi vida, he tenido la oportunidad de estar en varias facetas: como cliente, como Partner, como Microsoft y todas ellas me han ayudado a ser más consciente de cómo se ven las cosas desde cada lado. Recuerdo que como cliente se ve a los Partners como aves de rapiña que solo quieren sacarte lo que puedan y no tener verdadero conocimiento de lo que estaban haciendo, me toco en la OMS buscar candidatos para el proyecto de migración de la intranet a SharePoint 2003 y los Partners entrevistados demostraban que solo querían "ganar" el proyecto sin tener realmente experiencia o conocimiento de SharePoint; como Partner se ve a los clientes como astutos mentirosos que quieren aprovecharse de ti para que les hagas las cosas "gratis" o "extender" el alcance del proyecto para exprimirte más, por ejemplo una vez trabajamos en un proyecto que estaba "a cargo" de dos diferentes instituciones, las cuales no se ponían de acuerdo entre ellas sobre lo que realmente querían y nada más nos hacían trabajar y retrabajar sin satisfacer a nadie, proyecto que se alargó a dos años y nunca se completó, cada lunes llegaba uno de los "encargados" y nos decía "por qué no hay avances?, qué hicieron todo el fin de semana?". Finalmente, como Microsoft te das cuenta de la jungla en la que se vive en este mundo y aprendes a sobrellevarla.

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

Ya con todo el camino recorrido que me ha tocado experimentar, y lo que falta, he aprendido que con los seres humanos no hay nada escrito, lo que hoy día es popular mañana deja de serlo, nada de lo que podamos prever como seres humanos hacia futuro es totalmente seguro, la gente cambia, las circunstancias cambian, las ideas cambian… en SharePoint lo vivimos de primera mano, lo que hoy se dice que será el futuro, mañana deja de serlo; así que mi "visión" del futuro de la tecnología es que seguirá cambiando y adaptándose.

VLADIMIR MEDINA
@vladpoint
https://www.linkedin.com/in/vladpoint/
https://mvp.microsoft.com/es-es/PublicProfile/5001983
https://www.facebook.com/groups/56850858767/

28

SharePoint y Azure: El Azure Text Analytics API

Que es el Azure Text Analytics API

El servicio de Análisis de Texto de Azure permite el procesamiento de texto en tres formas diferentes: detección del lenguaje utilizado, extracción de palabras clave y análisis de sentimiento. El servicio forma parte del conjunto de Cognitive Services de Azure, que es una colección de algoritmos de Inteligencia Artificial que se pueden utilizar en la nube de Microsoft.

La parte de detección de lenguaje permite encontrar que lenguaje es utilizado en un texto. Hasta este momento es posible detectar 120 lenguajes con el Text Anaytics API. El resultado de una consulta devuelve el nombre del lenguaje, su código y un indicador de la probabilidad de acierto de la detección (un integer entre 0 y 1).

La extracción de palabras clave identifica las palabras o frases mas importantes en una oración. Esta parte del API se puede utilizar para determinar rápidamente el tema de una oración, crear resúmenes y para utilizar con sistemas de búsqueda.

El análisis de sentimiento produce un resultado sobre que tan positivo o negativo es un texto. El resultado es un integer entre 0 (sentimiento negativo) y 1 (sentimiento positivo). El análisis funciona para 120 idiomas y se puede utilizar para analizar comentarios, revisiones de productos, emociones en social media e Emails, etc. El análisis de sentimiento utiliza técnicas avanzadas de procesamiento de texto en algoritmos de Inteligencia Artificial, no simple "diccionarios" de palabras "buenas" y "malas", por lo que los resultados se mejoran constantemente con cada nueva consulta.

Las tres partes de Text Analytics API aceptan texto Unicode codificado como UTF-8 o UTF-16 y con un máximo de 5000 caracteres por documento; una consulta puede contener hasta 1000 documentos, pero no debe exceder 1 MB de tamaño.

"El servicio de Análisis de Texto de Azure permite el procesamiento de texto"

Microsoft proporciona un simulador del API (https://azure.microsoft.com/en-us/services/cognitive-services/text-analytics/) que permite introducir un texto cualquiera y encontrar su idioma, palabras clave y sentimiento:

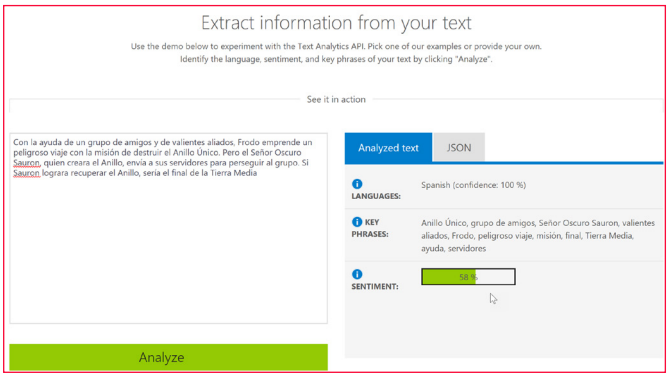


Imagen 1.- Simulador de Text API.

"Las tres partes de Text Anaytics API aceptan texto Unicode codificado como UTF-8 o UTF-16 "

Configuración del Azure Text API

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

1.– Entre al portal de manejo de Azure (https://portal.azure.com) utilizando sus credenciales.
2.– Vaya a la sección de "Resource Groups" y cree un nuevo Grupo de Recursos (también es posible reutilizar un grupo ya existente).
3.– Cree una cuenta de Cognitive Services:
   - En el Resource Group, utilice el botón de "+Add" para crear un recurso, busque por "cognitive services" en la casilla de búsqueda y seleccione "Cognitive Services" en los resultados.
   - Asígnele un nombre al servicio y utilice el Grupo de Recursos deseado. En "API type" seleccione "Text Anaytics API". En la casilla de "Pricing tier" seleccione el número máximo de transacciones que utilizara por mes, lo que determina el precio del servicio (gratis si se utilizan menos de 25000 transacciones por mes). Confirme el anuncio de privacidad que aparece en la configuración (Microsoft utilizara los datos enviados para mejorar automáticamente los algoritmos de reconocimiento).

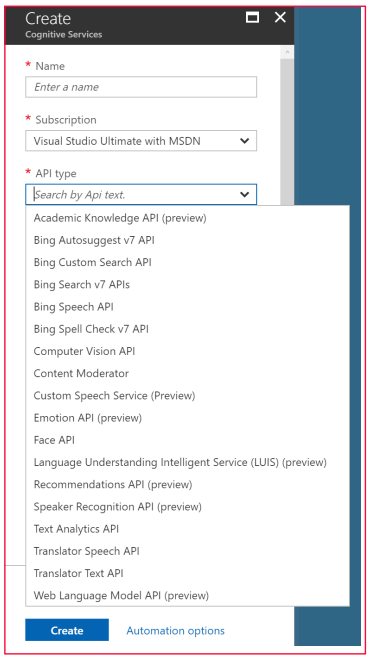


Imagen 2.- Creación del servicio de Text API.

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 Text API con SharePoint

En el siguiente ejemplo se va a utilizar una Lista de SharePoint con un campo de "Comentarios". Cuando se introduce un nuevo elemento en la Lista con un comentario, un WebHook hace que una Función de Azure comience a funcionar, utilice el texto del comentario para hacer una llamada al Azure Text Analytics API y determina su lenguaje, palabras clave y sentimiento. Finalmente, estos valores son introducidos de regreso en el elemento de la Lista.

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

5.– Cree una cuenta de Funciones básica en el Grupo de Recursos, asignándole un nombre, plan de servicios y cuenta de storage.

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

7.– Agréguele a la solución los paquetes NuGet "Microsoft.Azure.CognitiveServices.Language" (versión "1.0.0-preview"), "AppForSharePointOnlineWebToolkit" y "Newtonsoft.Json".

8.– Reemplace todo el código de la rutina con el siguiente (reemplace los valores de "baseUrl", "myUserName" y "myPassword" por los valores correctos de su sistema):":

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Security;
using Newtonsoft.Json;
using Microsoft.SharePoint.Client;
using Microsoft.Azure.CognitiveServices.Language.TextAnalytics;
using Microsoft.Azure.CognitiveServices.Language.TextAnalytics.Models;

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

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

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

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

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

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

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

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

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

```
static void GetChanges(ClientContext SPClientContext, string
ListId, TraceWriter log)
    {
        // Get the List
        Web spWeb = SPClientContext.Site.RootWeb;
        List changedList = spWeb.Lists.GetById(new Guid(ListId));
        SPClientContext.Load(changedList);
        SPClientContext.ExecuteQuery();

        // Create the ChangeToken and Change Query
        ChangeToken lastChangeToken = new ChangeToken();
        lastChangeToken.StringValue = string.For-
mat("1;3;{0};{1};-1", ListId, DateTime.Now.AddMinutes(-1).ToUni-
versalTime().Ticks.ToString());
        ChangeToken newChangeToken = new ChangeToken();
        newChangeToken.StringValue = string.For-
mat("1;3;{0};{1};-1", ListId, DateTime.Now.ToUniversalTime().
Ticks.ToString());
        ChangeQuery myChangeQuery = new ChangeQuery(false,
false);
        myChangeQuery.Item = true;  // Get only Item changes
        myChangeQuery.Add = true;  // Get only the new Items
        myChangeQuery.ChangeTokenStart = lastChangeToken;
        myChangeQuery.ChangeTokenEnd = newChangeToken;

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

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

            // Create a Text Analytics client
            ITextAnalyticsAPI AnalyticsClient = new TextAnalyt-
icsAPI();
            AnalyticsClient.AzureRegion = AzureRegions.Westeu-
rope;
            AnalyticsClient.SubscriptionKey = "c52262ef44d14e4c-
be59....";

            string textToExaminate = changedListItem["Comentar-
io"].ToString();
            CognitiveText myText = getCognitiveProps(textToEx-
aminate, AnalyticsClient);

            // Insert the values back in the Item
            changedListItem["NombreLanguage"] = myText.Lan-
guageName;
            changedListItem["Sentimiento"] = myText.Sentiment-
Score;
            changedListItem.Update();
            SPClientContext.ExecuteQuery();
          }
        }
    }
```

Primero se crea un objeto que contiene la Lista utilizando el contexto de SharePoint. Luego se crea una consulta de cambio (variable "myChangeQuery") que especifica que se requieren los cambios ocurridos en el último minuto, que ocurren en elementos de la Lista y que sean del tipo "Add", es decir, elementos nuevos. Luego de ejecutar la consulta, se examina cada uno de los cambios y se obtiene un objeto con el elemento agregado (el que contiene, a su vez, el texto del comentario). Note que en un sistema de producción se debe guardar el valor del ultimo ChangeToken para hacer el sistema más dinámico, no utilizando valores fijos como en este ejemplo.

En la misma rutina se crea un cliente del Analytics Text API utilizando la clave de suscripción (punto 04 en este articulo). La rutina "getCognitiveProps" se encarga de hacer la consulta en Azure utilizando el Text Anaytics API. Cuando ya se han obtenido los valores del texto, se introducen en los campos "NombreLenguage" y "Sentimiento" del elemento de la Lista.

10.– La rutina "getCognitiveProps" recibe como parámetros de entrada el texto del comentario y una referencia al cliente del Text Anaytics API:

```
static CognitiveText getCognitiveProps(string TextToExaminate,
ITextAnalyticsAPI AnalyticsClient)
    {
        CognitiveText myText = new CognitiveText();
        myText.Text = TextToExaminate;
        myText.TextId = Guid.NewGuid().ToString();

        // Extracting language
        LanguageBatchResult resultLanguage = AnalyticsClient.
DetectLanguage(
              new BatchInput(
                new List<Input>()
                {
                 new Input(myText.TextId, myText.Text)
                }));
        myText.LanguageCode = resultLanguage.Documents[0].
DetectedLanguages[0].Iso6391Name;
        myText.LanguageName = resultLanguage.Documents[0].
DetectedLanguages[0].Name;

        // Getting key-phrases
        KeyPhraseBatchResult resultPhrases = AnalyticsClient.
KeyPhrases(
              new MultiLanguageBatchInput(
                new List<MultiLanguageInput>()
                {
                 new MultiLanguageInput(myText.LanguageCode,
myText.TextId, myText.Text)
                }));
        myText.KeyPhrases = new List<string>(resultPhrases.
Documents[0].KeyPhrases);

        // Extracting sentiment
        SentimentBatchResult resultSentiment = AnalyticsClient.
Sentiment(
              new MultiLanguageBatchInput(
                new List<MultiLanguageInput>()
                {
                 new MultiLanguageInput(myText.LanguageCode,
myText.TextId, myText.Text)
                }));
        myText.SentimentScore = resultSentiment.Documents[0].
Score;

        return myText;
    }
```

"un WebHook hace que una Función de Azure comience a funcionar"

Cada consulta al Text Anaytics API puede contener múltiples documentos, como se indicó en la introducción del artículo. Cada documento tiene que tener un identificador, que es lo que inicialmente realiza la rutina: asignarle un GUID al texto a analizar. En este caso se esta enviando siempre un solo texto (un solo documento), por lo que solo es necesario crear un GUID. Inicialmente se extrae el lenguaje utilizado por medio del método "DetectLanguage", el que produce dos valores: el nombre del lenguaje

y su abreviatura según ISO6391. Luego se extraen las palabras clave, y finalmente el valor del sentimiento. Todos los valores se almacenan en un objeto del tipo "CognitiveText" que está definido en la clase correspondiente:

```
public class CognitiveText
    {
      public string Text { get; set; }
      public string TextId { get; set; }
      public string LanguageName { get; set; }
      public string LanguageCode { get; set; }
      public double? SentimentScore { get; set; }
      public List<string> KeyPhrases { get; set; }
    }
```

Otras tres clases definen objetos utilizados por el WebHook:

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

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

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

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

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

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

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

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

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

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

11.– Registre el WebHook (utilizando Postman, por ejemplo, como se indica en el artículo mencionado al principio del artículo) y cree un elemento en la Lista. El WebHook hará que la Función realice su trabajo, analice el texto, y cambie los valores en el elemento:

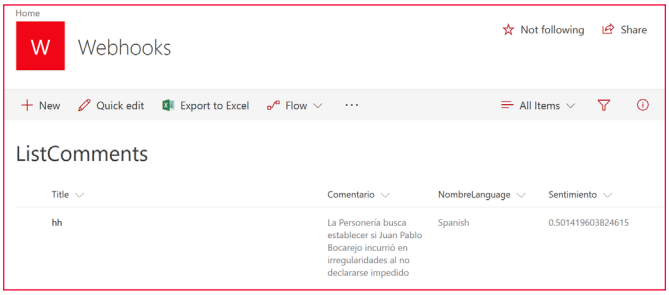


Imagen 3.- Elemento en la Lista con los resultados del servicio de Text API.

> ***"El servicio de Análisis de Texto de Azure puede llegar a ser una herramienta muy valiosa en SharePoint"***

Conclusiones

El servicio de Análisis de Texto de Azure puede llegar a ser una herramienta muy valiosa en SharePoint, especialmente en lo que se refiere a su parte de análisis de sentimiento. El Azure Text Anaytics API es fácil de utilizar desde cualquiera lenguaje de programación, y produce resultados confiables rápida y seguramente. El API utiliza algoritmos de Inteligencia Artificial que se mejoran con el uso, por lo no es necesario crear ni entrenar algoritmos propios.

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

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

Instalación de SCOM 2016 para la monitorización de nuestra infraestructura (Parte 1)

¿Qué es SCOM?

System Center 2016 Operations Manager (yo prefiero hablar de System Center Operations Manager 2016 aunque no sea correcto), también conocido como SCOM 2016, o de manera coloquial como OM (Operations Manager) es la aplicación desarrollada por Microsoft para poder monitorizar y controlar la infraestructura de sistemas y comunicaciones de nuestra explotación.

Simplificando el funcionamiento de SCOM, podemos decir que el administrador del sistema, instala el agente de SCOM, que es un pequeño programa que está siempre en ejecución, después registra el cliente en el servidor y configura en el mismo que cosas tiene que monitorizar.

Tras esto, el agente, tal como se ha dicho queda continuamente en ejecución y monitoriza aquellos recursos que consideremos importantes y tomando por ejemplo información del registro de eventos para buscar eventos que serán importantes para el correcto funcionamiento de la infraestructura.

> ***"es un producto que está incluido en la suite Microsoft System Center que está compuesta por utilidades"***

Si se produce un evento que se cataloga como importante y sobre el que se debe actuar, el agente se pone en contacto con el servidor de SCOM para informarle de lo sucedido. SCOM actuará en función de la criticidad de la alarma y de la configuración que tenga preestablecida para este tipo de eventos.

Este software ha evolucionado mucho desde su lanzamiento en el año 1998, cuando fue lanzado al mercado, no por la propia Microsoft si no por una empresa británica llamada Serverware Group plc. Esta empresa vendió a Microsoft los derechos de explotación y desarrollo en el año 2000 y desde entonces ha ido evolucionando.

Esta evolución, llevó al equipo de producto a rediseñar y recodificar el programa entero en 2007 para lanzar System Center Operations Manager 2007, que ya se parecía más a lo que hoy conocemos como SCOM.

Posteriormente, se volvió a hacer un gran salto con la versión de System Center Operations Manager 2012, que ha tenido cierto grado de continuidad en las versiones 2012R2 y en la versión actual, que es System Center Operations Manager 2016.

Cuando hablo de cierto grado de continuidad, me refiero a que cambian cosas, pero no hay un gran salto en lo que es la arquitectura interna. De hecho, se mejoran muchas; se amplía el soporte a sistemas, se agiliza la gestión de consultas y de DDBB, etc. Pero a nivel apariencia y funcionamiento todo continúa bastante parecido.

Lo primero que tenemos que tener claro antes de empezar a plantearnos siquiera el adquirir la licencia de Microsoft System Center Operations Manager 2016, debemos saber que SCOM no es un producto aislado, sino que es un producto que está incluido en la suite Microsoft System Center que está compuesta por utilidades que nos permiten gestionar de manera integral y centralizada toda nuestra infraestructura.

Tal como ya se apuntará en el número 31 de esta publicación, publicado en marzo de este mismo año, System Center contiene varios productos, entre los que la estrella incontestable es System Center Configuration Manager, seguido en importancia por System Center Operations Manager, que es el aplicativo sobre el que vamos a hablar en esta serie de artículos. Pero no hay que olvidarse de otros elementos de la suite como Data Protection Manager, Service Manager, Orchestator y Virtual Machine Manager.

Imagen 1.- Logotipo de System Center Operations Manager.

Aunque no es el propósito del presente artículo, vamos a

repasar brevemente cual es la funcionalidad de cada uno de estos elementos:

- System Center Configuration Manager: Herramienta destinada al despliegue de sistemas operativos, programas y aplicaciones, sin olvidar los parches de seguridad y las actualizaciones.
- System Center Operations Manager: Herramienta de monitorización centralizada de los sistemas de red, equipos, servidores y aplicativos.
- System Center Data Protection Manager: Herramienta profesional para realizar las copias de seguridad de Microsoft.
- System Center Service Manager: Herramienta dedicada a la creación, seguimiento y cierre de las incidencias y problemas, así como a la elaboración de inventario del parque de la empresa. Este elemento desarrolla todo su potencial en colaboración con el resto de elementos de la suite.
- System Center Orchestator: Herramienta de orquestación, dedicada a la automatización de respuestas a distintos eventos.
- System Center Virtual Machine Manager: Herramienta dedicada a la gestión de máquinas virtuales a lo largo de todo su ciclo de vida.

Estas explicaciones son muy simplistas, pero bastan para establecer los pilares sobre los que empezar a trabajar con SCOM.

Ahora que sabemos que no compramos la licencia de SCOM sola si no que se vende junto con el resto de los productos de la suite Microsoft System Center y sabemos también que puede sacar un mayor rendimiento a los productos de la misma mediante la instalación e integración de dichos productos, debemos pensar en si nos beneficia o no instalar otros productos de dicha suite.

Desde mi punto de vista sí, es profundamente enriquecedor ya que al pasarse la información de unos a otros e interactuar en función de lo que obtienen, se genera una retroalimentación que hace más funcional la detección y respuesta a incidentes que es lo que buscamos al fin y al cabo cuando desplegamos un sistema de monitorización.

Pongamos un ejemplo:

Si tenemos SCCM instalado que nos da el conocimiento de todos los elementos de nuestra infraestructura, además está conectado con SCOM y se ha desplegado Orchestator, puede darse la siguiente situación: SCOM recibe una alerta de un elemento del inventario de SCCM y que ha sido provisionado con el agente de SCOM, acto seguido, consulta las reglas establecidas en la orquestación y decide que ante tal alarma hay que proceder a realizar un reinicio.

Si nos fijamos en este ejemplo, el sistema ha sido capaz de corregir automáticamente un error, que es un error bien conocido, pero ¿Qué sucede si no queremos este nivel de automatización? O ¿Si no se encuentra con un error conocido?

Pongamos otro ejemplo:

Si tenemos la misma infraestructura, con SCCM, SCOM y Orchestator, Y además tenemos SCSM, puede generarse un escenario en el que cuando SCOM recibe una alerta de un elemento del inventario de SCCM y que ha sido provisionado con el agente de SCOM, acto seguido, consulta las reglas establecidas en la orquestación y en función de estas puede generar una incidencia que dirigirá por ejemplo al departamento de comunicaciones vía mail o si así está descrito en sus reglas realizará el reinicio del que hablábamos antes.

Como podemos ver, SCOM, junto al resto de la suite nos ofrece un gran potencial, que estará solamente limitado por nuestra imaginación, ya que el mundo de la orquestación, de la cual hablaremos en el futuro abre un abanico de posibilidades casi ilimitado.

Componentes de SCOM:

Cuando hablamos de SCOM, no hablamos de una aplicación "monolítica" como por ejemplo un Adobe Photoshop, si no que se trata de un pequeño grupo de aplicaciones que interactúan entre sí. Una vez más, es posible que peque de simplismo al hablar de esta manera, pero trato de ser lo más próximo y evitar argot complicado que pueda entorpecer el entendimiento del lector.

Tenemos varios componentes dentro del ecosistema de System Center Operation Manager que pasamos a enumerar a continuación:

- Agente.
- Servidor de administración.
- Consola del operador.
- Consola web.
- Servidor de base de datos.
- Servidor de informes.

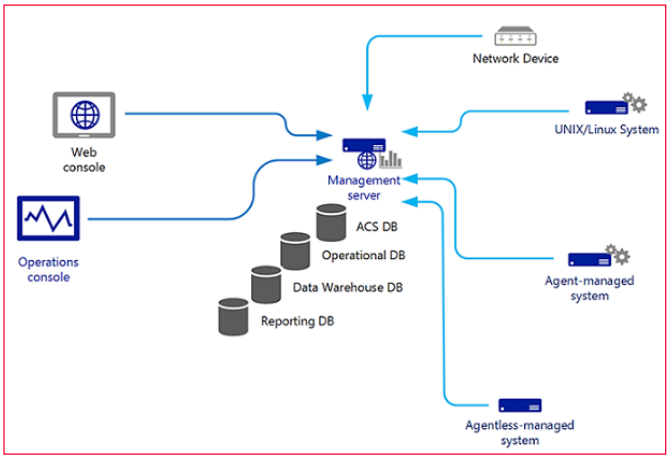


Imagen 2.- Ejemplo de esquema de System Center Operations Manager.

Para que podamos proseguir debemos saber qué hace cada uno de ellos, por lo que pasamos a explicar brevemente su función:

- Agente: Es un pequeño programa de software que se

instala en la máquina cliente y que se encarga de recopilar datos del mismo para enviárselos al servidor de SCOM. Envía datos sobre el estado de los distintos sistemas, aplicaciones y sobre el propio sistema operativo.

- Servidor de administración: Es propiamente dicho el corazón de SCOM. Contiene todas las herramientas de administración y gestión y se puede extender con paquetes adicionales para adquirir nuevas funcionalidades.
- Consola del operador: Aplicación que se puede instalar en cualquier equipo, servidor o cliente, siempre que cumpla unas especificaciones dadas y que permite realizar tareas de administración.
- Consola web: Una pequeña aplicación que permite realizar ciertas tareas administrativas de las que se pueden hacer desde el servidor de administración o la consola del operador. Permite realizar estas tareas atacando una URL y sin necesidad de instalar ningún paquete.
- Servidor de base de datos: Es un servidor destinado a hospedar las distintas instancias de System Center, como por ejemplo la base de datos operacional de SCOM.
- Servidor de informes: Es un aplicativo que corre sobre la base de datos. Almacena todos los eventos de la explotación sobre un periodo de tiempo predeterminado durante la configuración. Utiliza Bussines Inteligence para poder analizar y predecir comportamientos futuros.

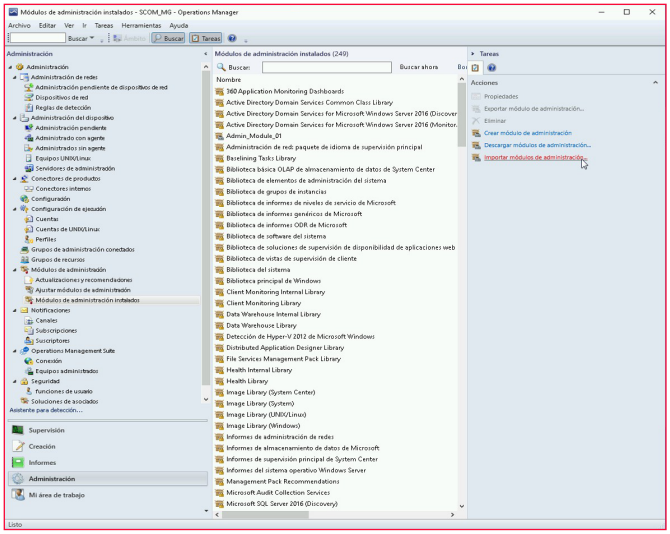


Imagen 3.- Consola de administración de System Center Operations Manager.

Orden de instalación de los componentes de SCOM:

Puede parecernos lógico empezar a instalar el servidor de administración, pero en principio, esto no es del todo correcto. Para bien, deberíamos haber preparado previamente una máquina con un servidor de bases de datos Microsoft SQL Server que hospedará las distintas instancias de los componentes de System Center y una vez hecho eso podremos empezar a instalar el servidor de administración en un servidor dedicado únicamente para tal propósito.

Pero ojo, aquí estamos hablando de SCOM, pero ¿Qué sucede si queremos instalar otros productos de System Center? O ¿Qué sucede si estamos actualizando desde una versión anterior? Esto también lo tenemos que tener en cuenta.

Si vamos a realizar la actualización desde una instalación previa de SCOM 2012R2 integrada con otros componentes de la suite de System Center, debemos seguir el siguiente orden de manera obligatoria y por razones de seguridad, ya que en caso de no hacerlo así podemos encontrarnos con problemas durante el despliegue o a posteriori.

- Orchestrator.
- Service Manager.
- Data Protection Manager.
- Operations Manager.
- Virtual Machine Manager.

El resto de los productos se podrán instalar posteriormente sin problemas, pero os recomiendo leeros la documentación de cada producto para evitar sustos. También debemos tener en cuenta la versión de los distintos componentes de Microsoft System Center 2012 para poder ser actualizados a Microsoft System Center 2016.

- Para actualizar Data Protection Manager debemos tener una versión de System Center 2012 R2 con UR10 o posterior.
- Para actualizar Operations Manager necesitamos System Center 2012 R2 con UR9 o posterior.
- Para Orchestrator tendremos System Center 2012 R2 con UR8 o posterior.
- La Automatización de administración de servicios necesita una versión anterior de System Center 2012 R2 con UR7 o posterior.
- Para actualizar Service Manager requiere System Center 2012 R2 con UR9 o posterior.
- Para actualizar Virtual Machine Manager requiere System Center 2012 R2 con UR9 o posterior.

"Tenemos varios componentes dentro del ecosistema de System Center Operation Manager "

Requisitos de SCOM

Lo siguiente que tenemos que definir son los requisitos mínimos que tiene que cumplir cada aplicativo que vamos a instalar. El primero, una vez instalada la base de datos (o no) es instalar el servidor de administración. Por ello vamos a empezar a mirar qué necesidades tiene, pero poniendo una pequeña nota previa sobre la base de datos. La base de datos, para no dar problema debe tener la intercalación configurada como "SQL_Latin1_General_CP1_CI_AS".

Esto es lo más importante en lo que respecta a la base de datos. Aun así, le dedicaremos un hueco en algún artículo posterior.

Los requisitos a nivel hardware del servidor de administración son:

- Procesador mínimo: CPU de 4 núcleos a 2,66 GHz.
- Memoria RAM mínima: 8 GB.
- Espacio en disco mínimo recomendado: 10 GB.

El servidor podrá ejecutar los siguientes sistemas operativos para poder soportar la posterior instalación de SCOM 2016:

- Windows Server 2012 R2 Standard.
- Windows Server 2012 R2 Datacenter.
- Windows Server 2016 Standard.
- Windows Server 2016 Datacenter.
- Windows Server Core 2016.

Otras necesidades software del servidor de administración:

- Windows PowerShell versión 2.0 o Windows PowerShell versión 3.0, así como las posteriores, que en el caso de Windows Server 2016 es la 5.1.
- Debemos habilitar la administración remota de Windows.
- Debemos instalar .NET Framework 4 o .NET Framework 4.5.

Mucho ojo también con la versión de .NET ya que puede causarnos grandes dolores de cabeza, al igual que la versión de PowerShell que deberemos instalar a pesar de tener una versión superior, ya que algunos de los scripts que usa System Center utilizan estas versiones. Tras haber hablado de estos requisitos, dejaremos en stand-by la instalación para dedicarle el siguiente artículo de esta misma serie.

Conclusión

Lo primero que tenemos que hacer antes de ponernos a desplegar Microsoft System Center Operations Manager 2016 en nuestra infraestructura es conocer las herramientas que lo integran, así como el entramado de aplicaciones que conforman la suite en la que se integra SCOM. Lo siguiente es conocer el orden de instalación y los requisitos necesarios para el despliegue. Básicamente tenemos que darle una buena pensada, planificar todo bien y tener muy claro que necesitamos para poder crear la infraestructura apropiada, ya que no será lo mismo poner SCOM para una empresa con una sede y 20 servidores homogéneos que para una con 100 sedes y servidores de todo tipo. De todas maneras, debemos recordar que éste es el primero de una serie de artículos que nos guiarán para poder desplegar sin problemas System Center Operations Manager 2016 en nuestras respectivas empresas.

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

Usando Cosmos DB Gremlin Graph API

Una vez que Microsoft anunció en el Build 2017 su nuevo servicio Cosmos DB (https://docs.microsoft.com/en-us/azure/cosmos-db/), en el Connect 2017 del 15 al 17 de Noviembre (https://www.microsoft.com/en-us/connectevent/default.aspx) anunciaron que los servicios de Azure Table Storage y Gremlin (grafos) dejaban de estar en preview.

En este articulo vamos a conocer la API para Gremlin de Cosmos DB.

¿Qué es Gremlin Graph?

Gremlin es un producto de Apache TinkerPop, que nos permite tener base de datos de grafos y realizar consultas sobre estos grafos mediante rutas transversales definida en pasos y que se ejecutan de forma secuencial.

Cada paso se ejecuta de forma automática y pasa el resultado al siguiente paso y las operaciones que se pueden ejecutar en cada paso son tres:

- Map -> Transformación de los objetos de la secuencia.
- Filter -> Eliminar objetos de la secuencia.
- Side Effect -> Estadísticas de cálculo de la secuencia.

¿Cuándo utilizaremos una BBDD de grafos?

Una BBDD de grafos nos puede ser útil en los siguientes escenarios:

- Redes sociales
- Geolocalización
- Detección fraudes y anomalías.

¿Por qué utilizar Gremlin con Cosmos DB y no directamente?

No existe diferencia apreciable entre utilizar Gremlin directamente o mediante Cosmos DB, el hecho de utilizarla mediante Cosmos DB lo que nos permitirá es beneficiarnos de las características que nos ofrece este servicio:

- Distribución global.
- Escalabilidad de throughpout y storage.
- 99,99% de disponibilidad
- Auto indexación y partición
- Latencias de lectura < 10 ms y de escritura indexada de < 15 ms.

> ***"Gremlin nos permite tener base de datos de grafos y realizar consultas sobre estos"***

Cosmos DB Gremlin Graph en Azure

Lo primero que vamos hacemos es crear el servicio Cosmos DB seleccionando Gremlin (graph).

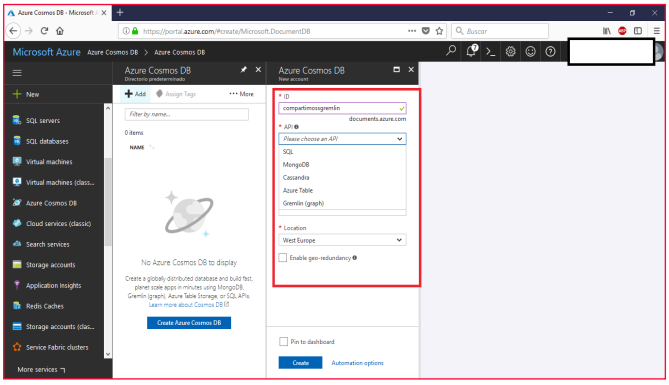


Imagen 1.- Creando Cosmos DB Gremlin(graph).

Una vez creada verás que se ha creado el servicio sin ningún grafo asociado:

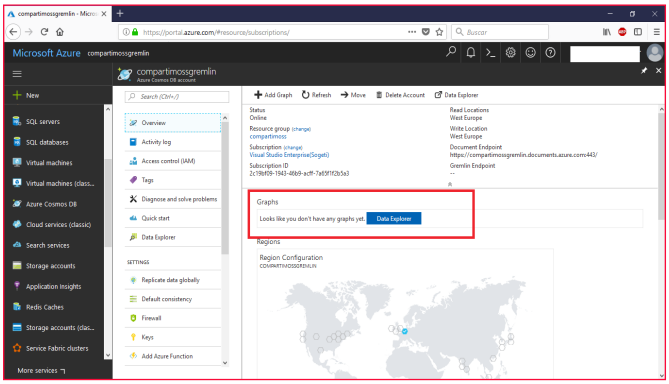


Imagen 2.- Cosmos DB Gremlin graph.

Una vez tenemos creado el servicio vamos a ver cómo crear grafos y consumirlos mediante .Net.

El grafo que vamos a construir es el siguiente:

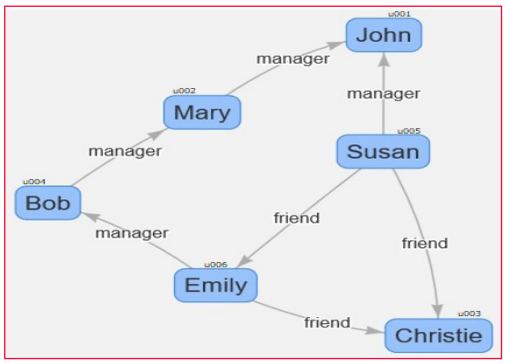


Imagen 3.- Grafo de ejemplo.

Crearemos un proyecto de consola y añadimos el paquete NuGet:

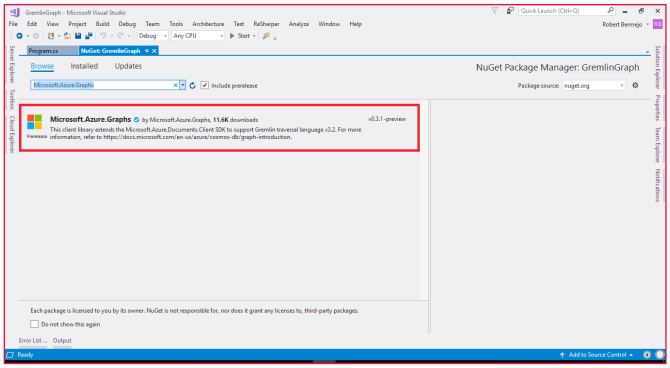


Imagen 4.- Paquete NuGet para usar Gremlin.

Lo primero que haremos es crear la conexión con el servicio:

```
string endpoint = ConfigurationManager.AppSettings["EndPoint"];
string authKey = ConfigurationManager.AppSettings["AuthKey"];

using (DocumentClient client = new DocumentClient(
        new Uri(endpoint),
        authKey,
        new ConnectionPolicy { ConnectionMode = ConnectionMode.Direct, ConnectionProtocol = Protocol.Tcp }))
{
        Program p = new Program();
        p.RunAsync(client).Wait();
}
```

Creamos la conexión informando el endopoint y la key para realizar la conexión:

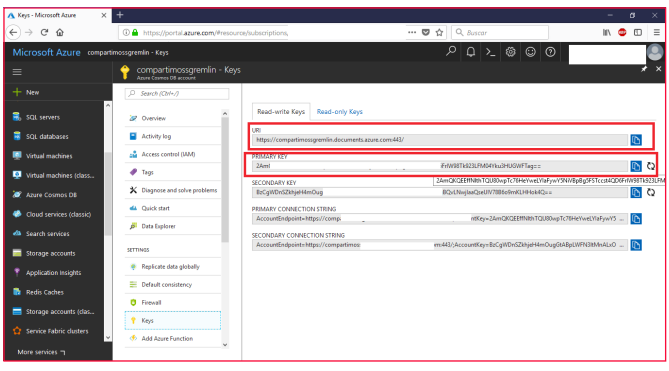


Imagen 5.- Settings del EndPoint.

Acto seguido creamos un databse y la collection que contendrá el grafo.

```
Database database = await client.CreateDatabaseIfNotExistsAsync(
                                        new Database { Id = "graphdb" });

DocumentCollection graph = await client.CreateDocumentCollectionIfNotExistsAsync(
        UriFactory.CreateDatabaseUri("graphdb"),
        new DocumentCollection { Id = "Persons" },
        new RequestOptions { OfferThroughput = 1000 });
```

Definiendo las operaciones

Las operaciones de Gremlin las tenemos que definir con su sintaxis, por lo que las tenemos que escribir en él código como string. En nuestro ejemplo:

```
Dictionary<string, string> gremlinQueries = new Dictionary<string, string>
{
  { "Cleanup",     "g.V().drop()" },
  { "AddVertex 1",  "g.addV('employee').property('id', 'u001').property('firstName', 'John').property('age', 44)" },
  { "AddVertex 2",  g.addV('employee').property('id', 'u002').property('firstName', 'Mary').property('age', 37)" },
  { "AddVertex 3",  g.addV('employee').property('id', 'u003').property('firstName', 'Christie').property('age', 30)" },
  { "AddVertex 4",  g.addV('employee').property('id', 'u004').property('firstName', 'Bob').property('age', 35)" },
  { "AddVertex 5",  g.addV('employee').property('id', 'u005').property('firstName', 'Susan').property('age', 31)" },
  { "AddVertex 6",  "g.addV('employee').property('id', 'u006').property('firstName', 'Emily').property('age', 29)" },
  { "AddEdge 1",    "g.V('u002').addE('manager').to(g.V('u001'))" },
  { "AddEdge 2",    "g.V('u005').addE('manager').to(g.V('u001'))" },
  { "AddEdge 3",    "g.V('u004').addE('manager').to(g.V('u002'))" },
  { "AddEdge 4",    "g.V('u005').addE('friend').to(g.V('u006'))" },
  { "AddEdge 5",    "g.V('u005').addE('friend').to(g.V('u003'))" },
  { "AddEdge 6",    "g.V('u006').addE('friend').to(g.V('u003'))" },
  { "AddEdge 7",    "g.V('u006').addE('manager').to(g.V('u004'))" },
  { "ReturnVertex", "g.V().hasLabel('employee').has('age', gt(40))" },
  { "AndOr",        "g.V().hasLabel('employee').and(has('age', gt(35)), has('age', lt(40)))"},
  { "Transversal",  "g.V('u002').out('manager').hasLabel('employee')" },
  { "outE/inV",     "g.V('u002').outE('manager').inV().hasLabel('employee')" },
  { "CountVertices","g.V().count()" },
  { "Filter Range", "g.V().hasLabel('employee').and(has('age', gt(35)), has('age', lt(40)))"},
};
```

Mediante un diccionario definimos las operaciones que vamos a ejecutar, y mediante el siguiente código las ejecutaríamos todas

```
foreach (KeyValuePair<string, string> gremlinQuery in gremlinQueries)
{
 Console.WriteLine($"Running {gremlinQuery.Key}: {gremlinQuery.Value}");

 IDocumentQuery<dynamic> query = client.CreateGremlinQuery<dynamic>(graph, gremlinQuery.Value);
 while (query.HasMoreResults)
 {
   foreach (dynamic result in await query.ExecuteNextAsync())
   {
     Console.WriteLine($"\t {JsonConvert.SerializeObject(result)}");
   }
 }

 Console.WriteLine();
}
```

Como vemos mediante CreateGremlinQuey creamos la query a ejecutar, y mediante el comando ExecuteNextAsync las ejecutamos.

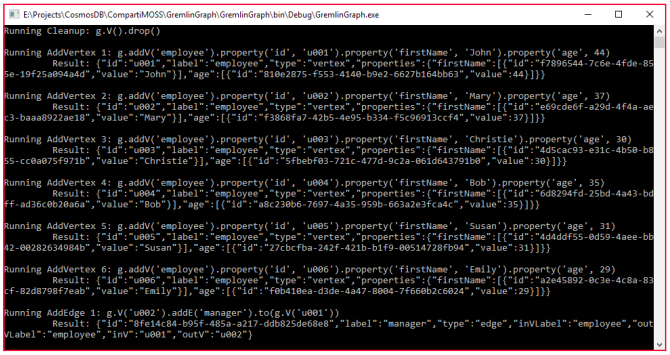


Imagen 6.- Ejemplo de resultados.

> ***"Las operaciones de Gremlin las tenemos que definir con su sintaxis"***

Ahora si vamos al portal vemos que se ha creado la colección:

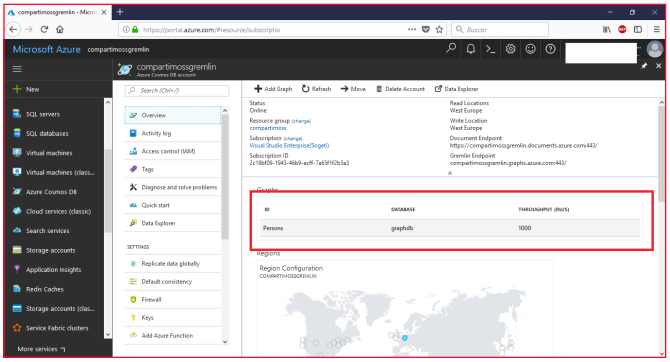


Imagen 7.- Ejemplo de resultados.

Ahora podemos ir a ver los datos al data explorer:

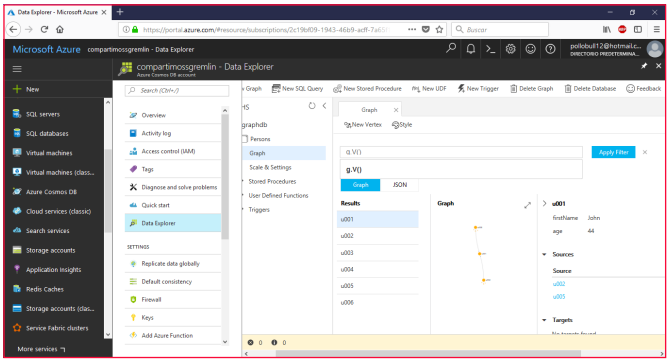


Imagen 8.- Data Explorer.

Aquí podríamos ejecutar cualquiera de las operaciones que hemos ejecutado anteriormente.

¿Qué operaciones podemos ejecutar?

En la siguiente tabla se enumeran todas las operaciones que soporta Cosmos DB para Gremlin Graph.

> ***"la forma de usar Gremlin es muy fácil y sencilla"***

STEP	DESCRIPCIÓN
addE	Agrega una arista entre dos vértices.
addV	Agrega un vértice al grafo.
and	Garantiza que todos los recorridos devuelven un valor.
as	Modulador de pasos para asignar una variable a la salida de un paso.
by	Modulador de pasos que se usa con group y order.
coalesce	Devuelve el primer recorrido que devuelve un resultado.
constant	Devuelve un valor constante. Se usa con coalesce.
count	Devuelve el número del recorrido.
dedup	Devuelve los valores sin duplicados.
drop	Quita los valores (vértice/arista).
fold	Actúa como una barrera que calcula el agregado de los resultados.
group	Agrupa los valores en función de las etiquetas especificadas.
has	Se utiliza para filtrar las propiedades, los vértices y las aristas. Admite las variantes hasLabel, hasId, hasNot y has.
inject	Inserta valores en un flujo.
is	Se usa para filtrar mediante una expresión booleana.
limit	Se usa para limitar el número de elementos en el recorrido.
local	Encapsula localmente una sección de un recorrido, de forma similar a una subconsulta.
not	Se usa para generar la negación de un filtro.
optional	Devuelve el resultado del recorrido especificado si produce un resultado; en caso contrario, devuelve el elemento que realiza la llamada.
or	Garantiza que al menos uno de los recorridos devuelve un valor.
order	Muestra los resultados con el criterio de ordenación especificado.
path	Devuelve la ruta de acceso completa del recorrido.
project	Proyecta las propiedades como un mapa.
properties	Devuelve las propiedades de las etiquetas especificadas.
range	Filtra el intervalo de valores especificado.
repeat	Repite el paso el número de veces especificado. Se usa para crear bucles.
sample	Se usa para tomar muestras de datos del recorrido.
select	Se usa para proyectar los resultados del recorrido.
store	Se utiliza para realizar agregados sin bloqueo del recorrido.
tree	Agrega las rutas de acceso desde un vértice en un árbol.
unfold	Extrae un iterador como un paso.
union	Combina los resultados de varios recorridos.
V	Incluye los pasos necesarios para los recorridos entre los vértices y las aristas V, E, out, in, both, outE, inE, bothE, outV, inV, bothV y otherV
where	Se usa para filtrar los resultados del recorrido. Admite los operadores eq, neq, lt, lte, gt, gte y between

Tabla 1.- Operaciones CosmosDB Gremlin Graph.

Conclusiones

La forma de usar Gremlin es muy fácil y sencilla como se puede ver, además puedes seguir ejecutando las operaciones por la cli de Gremlin apuntando simplemente al servicio de Cosmos DB.

El código completo de los ejemplos los podéis ver en:

- https://github.com/bermejoblasco/CosmosDBGremlinExample
- Referencias:

- https://docs.microsoft.com/en-us/azure/cosmos-db/gremlin-support
- https://tinkerpop.apache.org/gremlin.html
- https://tsmatz.wordpress.com/2017/07/04/azure-cosmos-db-gremlin-graph-tutorial/

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

41

Que listo es mi SharePoint!!!

Azure para dotar de inteligencia a tus aplicaciones

Que Azure, y más en concreto Azure PaaS se ha convertido en nuestro pozo de los deseos es más que evidente. Para todo aquel que haya saboreado de este catálogo de servicios, sabrá que no estoy loco, ¿Cómo si no podríamos tener arquitecturas con escalado vertical y horizontal en un solo click, servicios de búsquedas que indexen millones de elementos con pocas líneas de código, o procesos inteligentes en Serverless que se disparen de forma mágica sin preocuparnos de ningún tipo de infraestructura?

Azure algo más que músculo y arquitecturas ambiciosas

No voy a decir que no tenga mérito hacer una buena arquitectura en Azure, pero es de ser honestos admitir que nos lo han puesto bastante más fácil que cuando estábamos en sistemas Onpremises

Me gustaría que con este artículo nos posicionemos en una visión más ambiciosa, y lleguemos a ver que con Azure PaaS podemos incluso dotar de inteligencia a nuestras aplicaciones, mejorarlas al punto que un usuario pueda tener una experiencia de usuario mejorada, y todo con el objetivo de retener usuarios en aplicaciones que durante años nos han dado tanto trabajo y horas de desarrollo profesional.

¿Cómo mejorar SharePoint? – Seamos osados

Vamos a empezar por un gran conocido nuestro, y no es otro que SharePoint. Yo he tenido la suerte de trabajar con SharePoint desde las versiones 2003 hasta la actual version 2016 y Online. Y la verdad los que conocemos este producto, y hemos visto su evolución, podemos determinar que es la joya de la corona, que posiblemente hemos maltratado hasta el punto de que poco a poco lo hemos ido olvidando e incluso relegando a una posición más secundaria.

Uno de los motivos de "su no evolución", en las últimas versiones, yo creo que es la gestión o la mala gestión del dato que todos como usuarios hacíamos de esta plataforma. Es decir, SharePoint nos permite tener y definir arquitecturas de la información muy completas basadas en una estructura de sitios compleja, tipos de contenido a medida del caso de uso planteado por el usuario, taxonomías…; y una serie de artefactos como listas y bibliotecas en las que podemos almacenar casi cualquier tipo de información.

¿Dónde está el problema entonces?, en mi humilde opinión que si bien definir esto es un proceso de mucha ingeniería y el resultado es inmejorable si se hace bien, luego el usuario o bien no quiere o no sabe cómo rellenar esto, dejando al final la información mínima en la plataforma. Esto la verdad es fatal ya que muchos servicios de SharePoint como el buscador necesitan alimentarse de los datos del usuario.

¿Os imagináis que esto lo pudiéramos mejorar y conseguir que, con la mínima información por parte del usuario, nosotros pudiéramos autocompletar estos datos?

Buscador de noticias inteligente

La idea planteada es hacer un buscador de noticias, que permita al usuario subir los datos de la noticia como puede ser el título, la imagen o el cuerpo de la noticia; pero por otro lado los tags para rastrear esta noticia en el buscador se van a generar de forma dinámica.

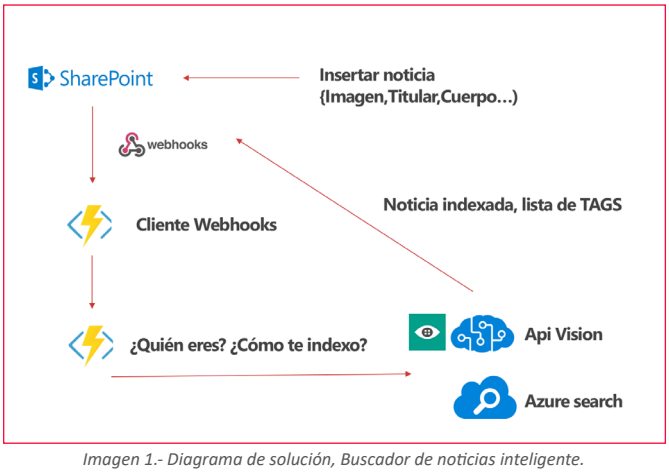


Imagen 1.- Diagrama de solución, Buscador de noticias inteligente.

Necesitamos un disparador, Webhook siempre contigo

Todo necesita de un inicio, y en este caso vamos a implementar un cliente webhook que reciba la notificación de que se ha creado una nueva noticia. Como vimos en artículos anteriores, si registramos un webhook de forma correc-

ta sobre la lista, nos va a permitir un mediante un cliente webhook que en este caso hemos implementado sobre un Azure Functions, recibir el evento y procesarlo de forma que sepamos que ha sucedido en esa biblioteca.

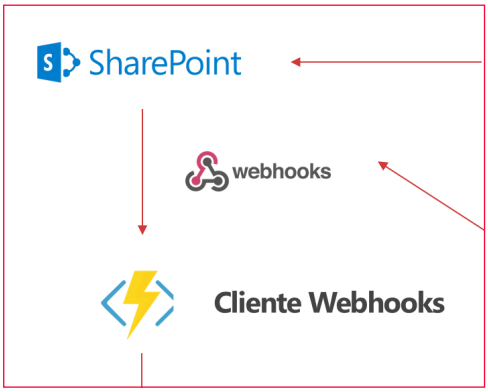


Imagen 2.- Disparador con Webhooks.

Para crear un Azure Function con nuestro cliente lo primero que tenemos que hacer es instalarnos las Tools de Azure Functions para Visual Studio 2017. Una vez instaladas las Tools, creamos un proyecto del tipo Azure Functions.

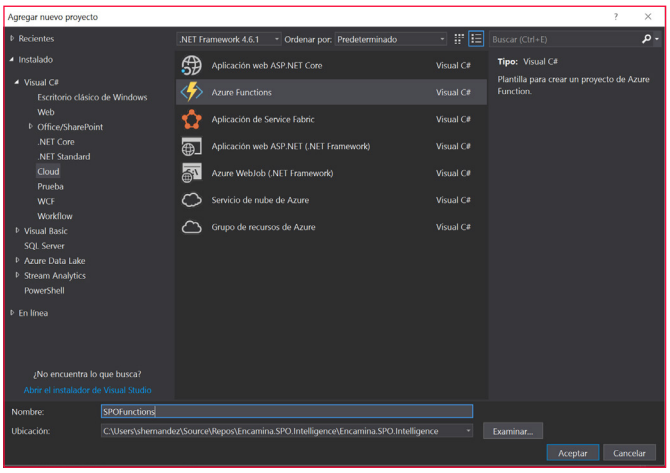


Imagen 3.- Creando un proyecto Azure Functions.

Sobre el proyecto creado, deberemos agregar una nueva función y como disparador seleccionaremos un HttpTrigger.

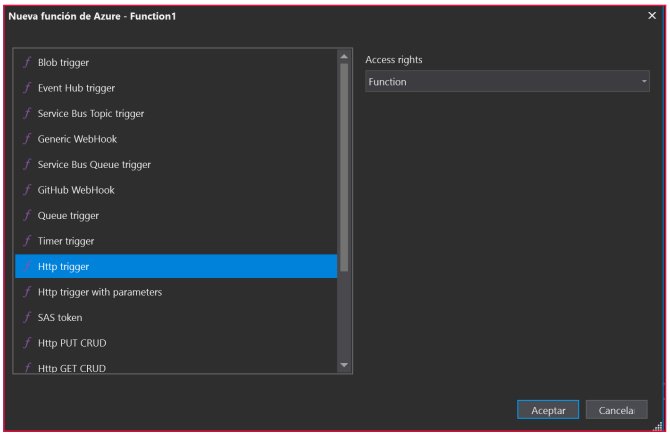


Imagen 4.- Azure Functions con Http Trigger.

Con esto ya tenemos nuestra function creada, ya solo nos quedaría añadir nuestro código del cliente Webhook que podría ser algo parecido a lo siguiente:

```
public static class WebhooksClient
  {
    [FunctionName("WebhooksClient")]
 public static async Task<HttpResponseMessage>
Run([HttpTrigger(AuthorizationLevel.Function, "get", "post",
Route =  null)]HttpRequestMessage req, TraceWriter log)
    {
      string validationToken = req.GetQueryNameValuePairs()
    .FirstOrDefault(q => string.Compare(q.Key, "validationto-
ken", true) == 0)
    .Value;

      if (validationToken != null)
      {
        log.Info($"Validation token {validationToken} received");
        var response = req.CreateResponse(HttpStatusCode.
OK);
        response.Content = new StringContent(validationToken);
        return response;
      }

      var notifications = JsonConvert.DeserializeObject<Respon-
seModel<NotificationModel>>(content).Value;

      if (notifications.Count > 0)
      {

      }
    }
  }
```

Si revisamos el código, es el código mínimo del que debemos de partir para crear un cliente Webhook. Por un lado, si recordamos del artículo anterior, cuando registramos un webhook sobre una lista, la primera vez se manda una petición al cliente para ver que este existe y es correcto, por eso necesitamos el primer bloque de código de "Validation Token".

Luego el resto es obtener la notificación y deserializarla en un objeto NotificationModel (es un modelo estándar que podéis coger del artículo anterior), y por último dejamos hueco a lanzar la lógica que en diagrama aparece como "Indexar y obtener Tags".

Imagen 5.- El paso inteligente: Tags con Azure Search y Api Vision

Contenido dinámico con Search y Cognitives Services

Este es el paso más interesante, porque hasta ahora simplemente estamos obteniendo el cambio vía webhook de un ítem en una biblioteca. Si hemos superado este punto, lo que vamos hacer es crear otra function que podemos denominar News o NewManager y que lo que va hacer es obtener la foto de la biblioteca, descargarla y procesarla.

Vamos a ver el siguiente código para que lo veamos más claro:

"podemos determinar que es la joya de la corona, que posiblemente hemos maltratado"

```
// Obtenemos la foto
SPOService spoM = new SPOService(site);
var image = spoM.GetPhotoInfo(lista, id);

// Analizamos quien aparece en la foto
CelebrityService cService = new CelebrityService();
var content = cService.MakeAnalysisCelebrity(image);
var celebrityName = cService.GetCelebrity(content.celebrity);

log.Info("Obteniendo el celebrity");

//Indexamos los tags
var azureSearch = new AzureSearchService();
var indexCreate = azureSearch.CreateIndexAsync<AzureSearch-
Model>("newsindex", false, null).Result;
var contentIndex = new AzureSearchModel() { IdSharepoint = id,
Name = celebrityName, Tags = content.tags, Id = id };
var uploadDocument = azureSearch.UploadDocuments<Azure-
SearchModel>("newsindex", new List<AzureSearchModel>() {
contentIndex }.ToArray());
spoM.SetResultNews(lista, id, JsonConvert.SerializeObject(con-
tent.tags));
return req.CreateResponse(HttpStatusCode.OK, "Noticia cate-
gorizada");
```

Si lo analizamos poco a poco podemos dividir este código en 3 bloques:

- Obtener la imagen desde SharePoint.
- Enviarla a procesar: Analizar la foto, ¿Quién es?
- Indexar los tags relacionados con la foto en Azure Search:
  - Crear un index en el servicio.
  - Generar el contenido del index.
  - Subir el contenido al servicio.

Lo interesante es entender esta functions que es la más importante del proceso, ya que el código está compartido de forma pública en este repositorio Git.

¿Quién eres? – API VISION

Azure nos proporciona una gran variedad de servicios para dotar de inteligencia a nuestras aplicaciones que denomina servicios Cognitivos. Dentro de este catálogo podemos encontrar la API VISION, que nos permite obtener información práctica de las imágenes, analizar el sentimiento de las personas que aparecen ellas o detectar información más específica de caras o textos, entre otras funcionalidades.

Imagen 6.- Servicios cognitivos en Azure.

En este caso para el ejemplo se ha utilizado la parte de "Computer Vision API", y más en concreto hemos utilizado la información relativa a Celebrities, para poder obtener la información necesaria de un famoso. Por ejemplo si a la biblioteca de SharePoint subiéramos la foto de Satya Nadella, podríamos obtener datos interesantes sobre él, con lo que construirnos nuestro listado de Tags.

Imagen 7.- Datos de celebridades con Computer Api Vision.

Volviendo al código de nuestra function de News, necesitamos implementar un método MakeAnalysisCelebrity que nos obtenga desde el servicio de Vision de Cognitive services los datos de la foto:

```
CelebrityService cService = new CelebrityService();
var content = cService.MakeAnalysisCelebrity(image);
```

El contenido que vamos a devolver es algo parecido al siguiente modelo, y no es otro que los datos de un Celebrity y su listado de Tags

```
public class ContentResult
{
  public Celebrity celebrity { get; set; }
  public List<string> tags { get; set; }
}
```

Para empezar a trabajar con Computer Vision API, debemos crearnos un servicio de este tipo en Azure, y obtener el suscription Key dentro del apartado de claves del servicio, ya que es necesario pasárselo en la cabecera de la petición al servicio. Para consumir el servicio, lo podemos hacer mediante el SDK de Azure, o bien haciendo peticiones a como a cualquier API, como vemos en el siguiente código:

```
public ContentResult MakeAnalysisCelebrity(Stream image)
{
var client = new HttpClient();
client.DefaultRequestHeaders.Add("Ocp-Apim-Subscrip-
tion-Key", "*********");

string uri = "https://northeurope.api.cognitive.microsoft.com/
vision/v1.0/analyze?details=celebrities&visualFeatures=Catego-
ries,Description,Color";

HttpResponseMessage response;

var buffer = GetImageAsByteArray(image);

using (var content = new ByteArrayContent(buffer))
{
// This example uses content type "application/octet-stream".
// The other content types you can use are "application/json"
and "multipart/form-data".
content.Headers.ContentType = new MediaTypeHeaderVal-
ue("application/octet-stream");
response = client.PostAsync(uri, content).Result;

string contentString = response.Content.ReadAsStringAsync().
Result;
var result = JsonConvert.DeserializeObject<RootCelebrityOb-
ject>(contentString);
return new ContentResult { celebrity = result.categories.Fir-
stOrDefault().detail.celebrities.FirstOrDefault(), tags = result.
description.tags };
}
}
```

A tener en cuenta la construcción de la URL del servicio, ya que debemos especificarle que queremos consultar, en este caso necesitamos obtener "celebrities", y dentro de las que denomina "VisualFeatures", necesitamos los campos categorías, descriptions y color, ya que con ellas construimos los tags. Como se puede observar es una petición post, con los parámetros y las cabeceras que necesitamos, para poder serializar el objeto en un RootCelebrityObjet que es del tipo:

```
public class RootCelebrityObject
{
public List<Category> categories { get; set; }
public Description description { get; set; }
public string requestId { get; set; }
public Metadata metadata { get; set; }
}
```

El resto de clases las obtenéis desde el ejemplo, pero en este caso es un mapeo directo desde lo que devuelve el servicio, analizando el json de salida se puede sacar sin problemas.

Indexando los Tags obtenidos

Una vez obtenemos el listado de Tags deberemos implementar haciendo uso de Azure Search un índice de búsquedas para nuestras noticias. Lo primero claro está es tener creado en el portal de Azure el Servicio de búsqueda, y después implementaremos un proyecto que maneje el servicio de Search.

En el ejemplo encontrareis un proyecto que se llama Encamina.SPO.Intelligence.Search, y con el cual podemos cubrir las tres necesidades básicas para interactuar con nuestro servicio:

- 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 generados desde la obtención de los Tags al servicio de búsqueda.

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

- Run query

  Lanza las consultas contra un índice en concreto, y mediante un texto pasado por parámetro

```
public async Task<Result<T>> RunQuery<T>(string indexName,
SearchParameters searchParameters,
string key,
SearchContinuationToken continuationToken,
bool highScoring = false)
```

Además de implementar el servicio es muy importante implementar de forma correcta el modelo que va a formar nuestro índice de búsqueda. En este caso si analizáis un poco los métodos del servicio, necesitamos implementar un modelo que se llame AzureSearchModel, y que puede ser de la siguiente forma:

```
public class AzureSearchModel
{
[System.ComponentModel.DataAnnotations.Key]
[IsFilterable]
public string Id { get; set; }

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

[IsRetrievable(true)]
public string IdSharepoint { get; set; }

[IsRetrievable(true), IsFacetable, IsFilterable, IsSearchable]
public List<string> Tags { get; set; }
}
```

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:

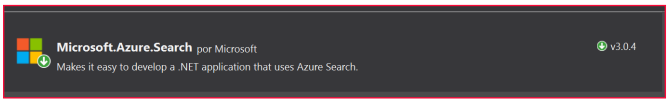


Imagen 8.- Paquetes necesarios para Azure Search.

Solo queda probarlo, interfaz libre

Una de las ventajas de utilizar Azure Functions con entrada por Http, es que somos independientes para visualizar estos datos donde queramos, es decir podemos por ejemplo utilizar SPFX para hacer un webpart en nuestro SharePoint o bien optar por algo más nativo y desarrollar una PowerApps.

En cualquier caso, por ver que esto funciona, vamos a ponerlo un poco más fácil aun así a los desarrolladores de Front-end y vamos a unificar los datos de nuestra lista de SharePoint con los datos obtenido desde nuestro Azure Search.

Para ello vamos a implementar una functions más que vamos a llamar Search, y que va a cruzar los datos del índice con los datos de la lista, y que tendrá una pinta parecida a lo siguiente:

```
public static class Search
{
[FunctionName("Search")]
public static async Task<HttpResponseMessage> Run([HttpTrigger(AuthorizationLevel.Function, "get", "post", Route = null)]
HttpRequestMessage req, TraceWriter log)
{
log.Info("C# HTTP trigger function processed a request.");

// parse query parameter
string text = req.GetQueryNameValuePairs()
.FirstOrDefault(q => string.Compare(q.Key, "text", true) == 0)
.Value;

string site = req.GetQueryNameValuePairs()
.FirstOrDefault(q => string.Compare(q.Key, "site", true) == 0)
.Value;
```

```
string library = req.GetQueryNameValuePairs()
.FirstOrDefault(q => string.Compare(q.Key, "library", true) == 0)
.Value;

// Get request body
dynamic data = await req.Content.ReadAsAsync<object>();

// Set name to query string or body data
text = text ?? data?.text;
site = site ?? data?.site;
library = library ?? data.library;



var azureSearch = new AzureSearchService();
var result = azureSearch.RunQuery<AzureSearchMod-
el>("newsindex", new Microsoft.Azure.Search.Models.Search-
Parameters(), text, null).Result;
var searched = JsonConvert.SerializeObject(result);

var spoS = new SPOManager.SPOService(site);
var resultSPO = spoS.GetData(library);


return text == null
? req.CreateResponse(HttpStatusCode.BadRequest, "Inserte
texto de busqueda")
: req.CreateResponse(HttpStatusCode.OK,new FullResult() {
SearchResult = searched, SharepointResult = resultSPO });
}
}
```

Si observamos el código, veremos que nuestra función necesita revisar los datos del site, la biblioteca de Sharepoint, y por otro lado el "texto de búsqueda", sobre el cual haremos la consulta.

"la verdad es fatal ya que muchos servicios de SharePoint como el buscador necesitan alimentarse de los datos del usuario"

Después de esto necesitamos invocar al RunQuery de AzureSearchService que vimos en puntos anteriores, y tendremos un servicio que vía CSOM nos obtenga los datos de la biblioteca. Lo único necesario ahora es juntar ambos contenidos (search y SharePoint) y devolverlo en un objeto FullResult, que no es más que una clase de la siguiente forma:

```
public class FullResult
{
public string SearchResult { get; set; }
public string SharepointResult { get; set; }
}
```

En resumen, esta functions nos va a permitir devolver dos JSON de resultado, para que luego el desarrollador de front pueda cruzar los datos en cliente, ya que el índice de Azure tiene todos los identificadores de SharePoint. Este proceso se puede refinar tanto como sea necesario, filtrando por consultas paginadas, o reduciendo el número de resultados, pero para el ejemplo esto nos sobra.

Ya en función de las necesidades, podríamos englobar esta function en una UI personalizada, como podría ser la del siguiente ejemplo.

Imagen 9.- Ejemplo de interfaz sencilla desde nuestro Functions.

Con muy poco obtenemos mucho….

Está claro que este ejemplo es muy tonto, y que posiblemente no se dé este caso de uso en la realidad, pero podemos sacar conclusiones muy positivas. Desde una simple foto, podemos auto-completar los datos de nuestro SharePoint, y esto ya es un paso, un paso pequeñito que con un poco de imaginación podemos ir haciendo crecer.

Si pensamos que estamos en la era "social", en la que los Facebook, Instagram…, dominan el mercado de lo social, y en la que tenemos miles de fotos sin explotar, ya que no obtenemos casi información de esas imágenes para un uso empresarial. Si esto lo llevamos a las intranets, o las soluciones sociales de empresa como Yammer, podríamos empezar a recoger muchísima información útil e integrarla en nuestros desarrollos. Nada que no hagan ya aplicaciones como Delve por ejemplo.

La idea es que ya tenemos herramientas para mejorar la inteligencia y los procesos de nuestras aplicaciones, sin necesidad de insertar miles de registros en una base de datos, y mucho más aun tener un registro de datos vivo y dinámico.

Eso si no os paséis dando inteligencia, la humanidad y la privacidad de la gente os lo agradecerá, no queremos acabar como en Terminator o Blade Runner entre otros ejemplos :)

SERGIO HERNANDEZ MANCEBO
Principal Team Leader en Encamina
@shmancebo

46

Application Lifecycle Management (ALM) API's

Como hemos visto en anteriores artículos, el SharePoint Framework ha venido para quedarse. El equipo de producto ha realizado muchos esfuerzos en unificar el modelo de desarrollo, por un lado, para poder personalizar SharePoint Online de una forma asumible y durable en el tiempo. Y por otro lado intentar unificar el modelo de desarrollo tanto en SharePoint Online como OnPremise. Sin embargo, todos estos esfuerzos, tenían un freno y es que todo este modelo de desarrollo no tenia un ciclo de vida acorde para que las empresas que lo utilizan lo puedan adoptar de una forma natural sin tener que recurrir a utilizar herramientas manuales para poder utilizarlo y desplegarlo en entornos productivos.

Ahora bien, el equipo de producto lleva tiempo hablando que estaban trabajando en una API para poder incorporarla en el ciclo de desarrollo. Pues bien, durante la pasada European SharePoint Conference se anunció que esta API iba a estar disponible.

¿Qué tiene esta API?

La API de ALM proporciona una serie de API simples para administrar la implementación de las soluciones del SharePoint FrameWork. Esta API admite las siguientes capacidades:

- Añadir soluciones de SPFX y/o SharePoint Add-in al catálogo de aplicaciones de nuestro tenant
- Activar/Desactivar soluciones de SPFX y/o SharePoint Add-in que están instaladas en nuestro catálogo de aplicaciones
- Instalar/Desinstalar soluciones de SPFX y/o SharePoint Add-in que están disponibles en nuestro catalogo de aplicaciones en nuestro sitio
- Actualizar la solución de SPFX y/o SharePoint Add-in en nuestro sitio con la versión más reciente que hay en nuestro catálogo de aplicaciones.

Esta API naturalmente la podemos consultar mediante servicios REST, utilizando CSOM o PnP.

"el nuevo modelo de desarrollo para SharePoint, era todo lo relativo con el proceso del ciclo de vida"

Métodos disponibles

- Añadir solución al Catálogo de aplicación:

```
/_api/web/tenantappcatalog/Add(overwrite=true, url='test.
txt')";
method: POST
binaryStringRequestBody: true
body: 'byte array of the file'
```

- Desplegar la solución en el cátalo de Aplicaciones:

```
/_api/web/tenantappcatalog/AvailableApps/GetById('xxxxxxxx-
xxxx-xxxx-xxxx-xxxxxxxxxx')/Deploy";
```

- Eliminar la solución del catálogo de aplicaciones:

```
/_api/web/tenantappcatalog/AvailableApps/GetById('xxxxxxxx-
xxxx-xxxx-xxxx-xxxxxxxxxx')/Retract";
```

- Mostrar las soluciones disponibles en el catálogo de aplicaciones:

```
url: /_api/web/tenantappcatalog/AvailableApps
```

- Mostrar los detalles de una solución que está en el catálogo de aplicaciones:

```
url: /_api/web/tenantappcatalog/AvailableApps/GetBy-
Id('xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxx')
```

- Instalar el paquete de solución en el Sitio de SharePoint:

```
url: /_api/web/tenantappcatalog/AvailableApps/GetBy-
Id('xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxx')/Install";
method: POST
```

- Actualizar el paquete de solución en el sitio de SharePoint:

```
url: /_api/web/tenantappcatalog/AvailableApps/GetBy-
Id('xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxx')/Upgrade";
```

- Desinstalar la solución en el sitio de SharePoint:

```
/_api/web/tenantappcatalog/AvailableApps/GetById('xxxxxxxx-
xxxx-xxxx-xxxx-xxxxxxxxxx')/uninstall";
method: POST
```

"ha realizado muchos esfuerzos en unificar el modelo de desarrollo, por un lado, para poder personalizar SharePoint Online"

Métodos en PnP PowerShell

Junto con estos servicios REST, el equipo de PnP ha añadido dentro de su repositorio de PowerShell una serie de cmldets para poder integrarlo con el resto de procesos de automatización de los que dispone la herramienta. Estos cmdlets son los siguientes:

- Para agregar una solución al catálogo y tenerla activa:

```
Add-PnPApp -Path ./myapp.sppkg"
Publish-PnPApp -Identity <app id> -SkipFeatureDeployment
```

- Para eliminar una app del catálogo:

```
Remove-PnPApp -Identity <app id>
```

- Utilizar la App en tu sitio de SharePoint Online:

```
Install-PnPApp -Identity <app id>
```

- Modificar la App en tu sitio de SharePoint Online:

```
Update-PnPApp -Identity <app id>
```

- Eliminar la App en tu sitio de SharePoint Online:

```
Uninstall-PnPApp -Identity <app id>
```

¿Qué aspectos le faltan a esta API ALM?

Lo único que le quedaría a esta API es el despliegue de las Extensions, es decir tener unos métodos para poder activar/desplegar las extensions dentro de los artefactos en los que le hemos implementado. Pero si tenemos en cuenta que las Extensión aún no se pueden utilizar en entornos productivos podemos esperar que estos comandos salgan a la luz una vez se puedan utilizar las Extensions en entornos de producción.

"aparición de esta API es una gran noticia para los Desarrolladores de Office 365/ SharePoint Online que por fin van a poder automatizar"

Resumen

La aparición de esta API es una gran noticia para los Desarrolladores de Office 365/SharePoint Online que por fin van a poder automatizar la gran parte de su desarrollo sin necesidades de intervención manual. Es de esperar que esta API evolución al compás de los nuevos artefactos que vayan apareciendo en la plataforma.

En futuros artículos veremos cómo incluir esta API dentro de nuestro proceso de desarrollo.

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

48

4 formas de añadir un administrador secundario a ODFB

Añadir un administrador secundario a OneDrive for Business (ODFB) es una tarea que, como administradores de Office 365, es posible que tengamos que realizar en más de una ocasión como por ejemplo cuando un empleado abandona la compañía y se nos solicita que tomemos el control de los documentos almacenados en su ODFC. Para realizar esta tarea tenemos al menos 4 posibilidades como demostraré en este artículo:

- Haciendo uso del Centro de Administración de Office 365.
- Haciendo uso de las opciones de configuración de los Perfiles de Usuario en SharePoint Online (SPO) disponible en la administración de SPO.
- Por medio de cmdlets estándar de SPO.
- Por medio del Modelo de Objetos en Cliente (CSOM) para SPO.

Añadiendo un administrador secundario a ODFB a través del Centro de Administración de Office 365

Como administradores de Office 365 es muy sencillo tomar el control del ODFB de un usuario haciendo uso de la opción “Acceso a archivos” (“Access files”) disponible al seleccionar un usuario en el listado de usuarios activos del tenant:

- Para probar esta opción, navegamos a la Administración de Office 365 y en la sección Usuarios hacemos clic en Usuarios activos. A continuación, seleccionamos un usuario existente en el listado de forma que se muestra el panel de detalles del usuario. Si nos desplazamos por el panel, encontraremos una sección “Configuración de OneDrive” desde la que podremos tomar el control del ODFB del usuario haciendo clic en el enlace “Acceso a archivos”

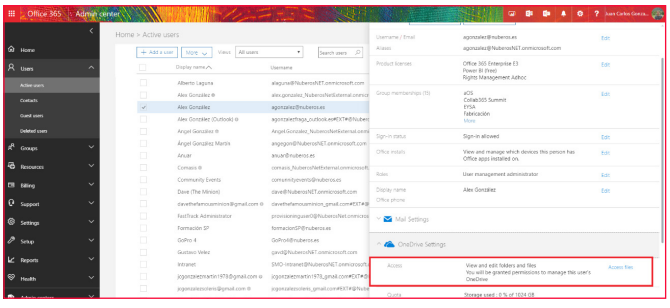


Imagen 1.- Sección de configuración de OneDrive en el panel de detalle del usuario.

- Una vez que se ha hecho clic en “Acceso a archivos”, se mostrará el enlace al ODFB del usuario.

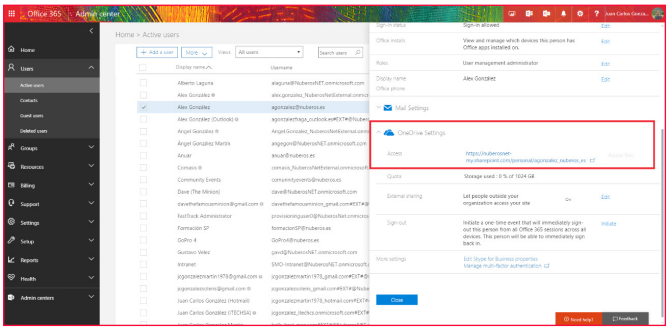


Imagen 2.- Enlace de acceso al ODFB del Usuario.

- Si hacemos clic en el enlace del ODFB del Usuario, tendremos acceso al ODFB del Usuario con permisos de administrador.

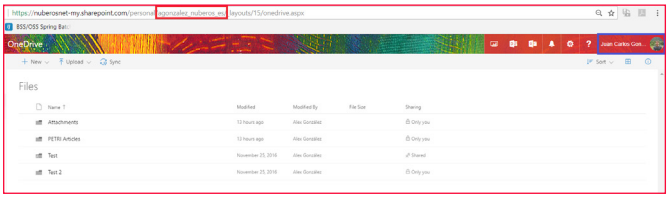


Imagen 3.- Acceso al ODFB del usuario como administrador secundario.

“seleccionamos un usuario existente en el listado de forma que se muestra el panel de detalles del usuario”

Añadiendo un administrador secundario a ODFB a través del Centro de Administración de SPO

Como seguramente sabéis, ODFB está construido sobre el servicio de SPO en Office 365 lo que significa que muchas opciones de configuración están disponibles a través de la Administración de SPO. En concreto, podemos administrar algunas de las opciones de configuración de SPO a través del servicio de los perfiles de usuario de SPO. Por ejemplo, desde este servicio podemos añadir un administrador secundario al ODFB de un usuario:

- En primer lugar, accedemos al centro de Administración de SPO y hacemos clic en el enlace de “perfiles de usuario” (“User profiles”) disponible en el menú ver-

tical. En la página de administración de los perfiles de usuario, hacemos clic en “Administrar perfiles de usuario” (“Manage user profiles”).

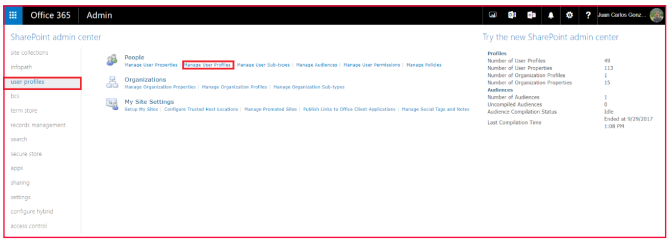


Imagen 4.- Accediendo a la administración de los perfiles de usuario en SPO.

- En la página de “Administrar perfiles de usuario”, buscamos un usuario concreto y desplegamos las opciones de administración disponibles para su perfil. Hacemos clic en la opción “Administrar propietarios de la colección de sitios” (“Manage site collection owners option”).

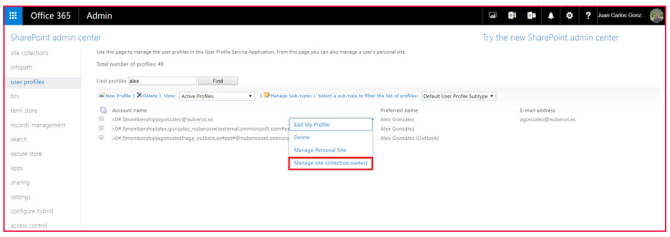


Imagen 5.- Opción de “Administrar propietarios de la colección de sitios”.

- En la Ventana de “Administrar propietarios de la colección de sitios”, simplemente añadimos / eliminamos un administrador secundario del ODFB del usuario. En mi caso, simplemente he editado el mismo usuario en el que ya había configurado un administrador secundario de ODFB a través del panel de detalle del usuario en el Centro de Administración de Office 365.

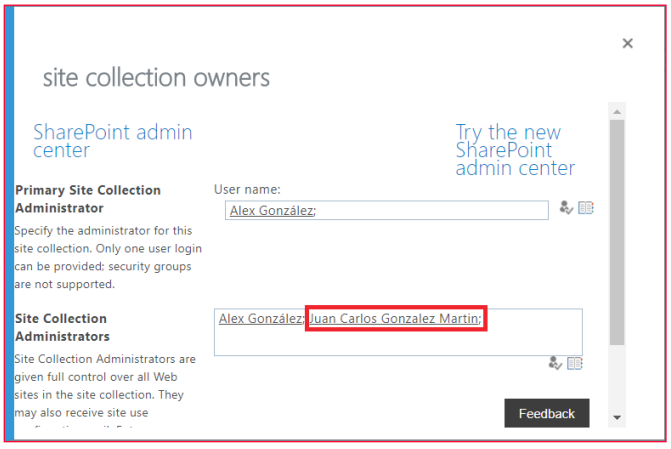


Imagen 6.- Añadiendo un administrador secundario a ODFB a través de la administración de perfiles de usuario de SPO.

Añadiendo un administrador secundario a ODFB usando cmdlets estándar de SPO

Los comandos PowerShell para SPO proporciona una ruta alternativa para realizar tareas de administración de ODFB y SPO de manera que con unas pocas líneas de código PowerShell se puede añadir un administrador secundario a ODFB. En concreto, esta operación se pude realizar por medio del comando estándar Set-SPOSite como se muestra a continuación:

```
$ssODFBSite=”<ODFB_Site_Url>”
$sSecondaryODFBAdmin =”<O365_User_To_Add>”
Set-SPOUser -Site $sODFBSite -LoginName $sSecondaryODF-
BAdmin -IsSiteCollectionAdmin $true
```

Se puede descargar un script Powershell más completo que permite añadir un administrador secundario al ODFB de un usuario desde el siguiente enlace: How to add a secondary administrator to a user’s ODFB

Añadiendo un administrador secundario a ODFB por medio de CSOM

Finalmente, la cuarta forma de añadir un administrador secundario a ODFB se pude lograr a través de la API CSOM para SPO que se utilice en una aplicación .NET (en una simple aplicación de consola) o en un script PowerShell. Por ejemplo, a continuación, se muestra el código PowerShell necesario para añadir un administrador secundario a ODFB usando CSOM:

```
$ssODFBSite=”<ODFB_Site_Url>”
$sSecondaryODFBAdmin =”<O365_User_To_Add>”
$spoCtx = New-Object Microsoft.SharePoint.Client.ClientCon-
text($sODFBSite)
$spoCredentials = New-Object Microsoft.SharePoint.Client.
SharePointOnlineCredentials($sUserName, $sPassword)
$spoCtx.Credentials = $spoCredentials
$spoUser=$spoCtx.Web.EnsureUser($sSecondaryODFBAdmin)
$spoUser.IsSiteAdmin=$true
$spoUser.Update()
$spoCtx.Load($spoUser)
$spoCtx.ExecuteQuery()
```

Se puede descargar un script Powershell más completo que permite añadir un administrador secundario al ODFB de un usuario haciendo uso de la API de CSOM de SPO desde el siguiente enlace: How to add a secondary administrator to a user’s ODFB (CSOM Edition)

Conclusiones

Añadir un administrador secundario al ODFB de un usuario se puede conseguir al menos de 4 formas diferentes aprovechando las capacidades de los centros de administración de Office 365 y SPO por un lado y de toda la potencia que los cmdlets estándar de SPO y la API CSOM proporcionan.

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

50

Skype for Business Ethical Wall

Cuantas veces un cliente les pide restringir la comunicación entre OU's , o que nadie pueda enviar IM al CIO. O en las compañías bancarias y financieras solamente permitir que los cajeros o usuarios reciban IM únicamente, que no hagan otra acción.

La solución a esto es una herramienta de 3ros llamada ETHICAL WALL. Con una solución de este tipo, lo que se pretende es evitar fuga de información basándonos en políticas para limitar la comunicación entre nuestros usuarios. Así que nosotros como administradores seremos capaces de restringir las características y funcionalidades que los usuarios pueden utilizar entre ellos. Desde bloquear quien puede ver presencia y enviar mensajes, sino hasta quien puede enviar archivos, Audio y Video o compartir el escritorio.

Les mostrare como un Ethical Wall puede ayudarnos a cumplir con el objetivo.

Lo primero que vemos es que tenemos dos diferentes endpoints, y las diferentes funcionalidades que podemos habilitar\deshabilitar entre ellos. Y no son restrictivas en ambos sentidos, sino que podemos seleccionar que únicamente el punto A no pueda ver la presencia del punto B. Pero el punto B si podrá ver la presencia del A. Es decir, no son mutuamente excluyentes.

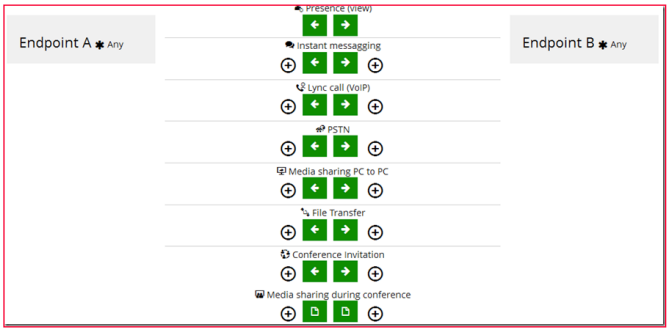


Imagen 1.- EndPoints the Ethical Wall.

"seleccionamos un usuario existente en el listado de forma que se muestra el panel de detalles del usuario"

Un Endpoint no solamente es un usuario, un endpoint puede ser una OU, un dominio Federado, un número de teléfono, etc. Aquí está la lista con todas las posibilidades de Endpoint.

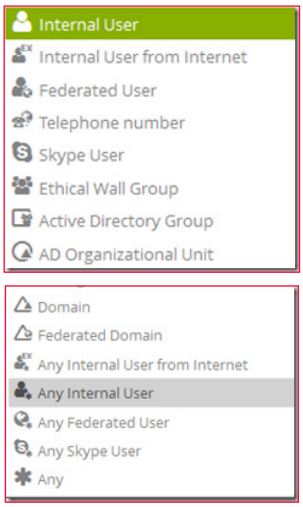


Imagen 2.- Posibilidades de EndPoint.

Ahora les mostraré como Ethical Wall trabaja.

Ejemplo1

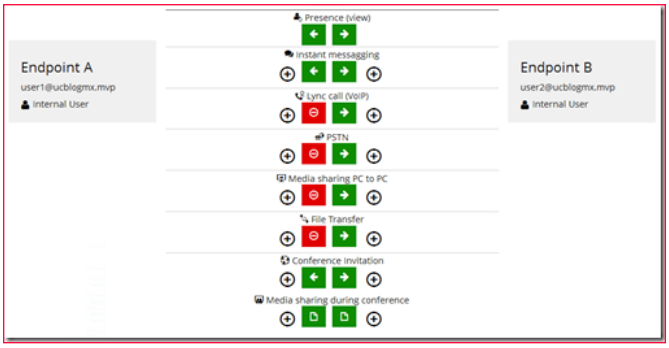


Imagen 3.- Ejemplo de configuración de Ethical Wall.

En la imagen previa pueden ver que tenemos dos usuarios internos, Endpoint A (user1@ucblogmx.mvp) y Endpoint B (user2@ucblogmx.mvp) . Se bloqueo al usuario 2 la posibilidad de llamar, compartir escritorio y transferencia de archivos al usuario 1.

Si el usuario 2 trate de enviar un archivo al usuario 1, recibe el siguiente mensaje de error.

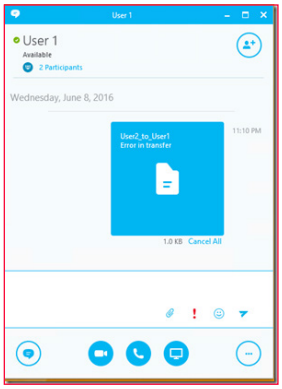


Imagen 4.- Error al transferir archivos por parte del usuario 2.

Pero el usuario 1 puede enviar archivos sin problema.

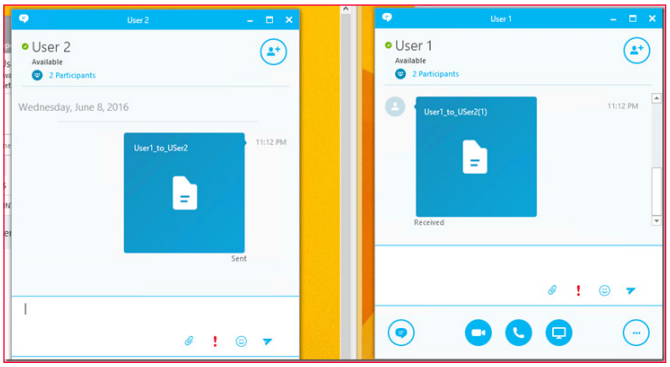


Imagen 5.- Experiencia del primer usuario compartiendo un archivo.

Ejemplo 2

Ahora les mostraré como bloquear comunicaciones de usuarios de bajo nivel a los Gerentes/Administradores/CIO:

Tengo dos Ou's Managers y Mortals

El usuario 1 es miembro de la OU Managers

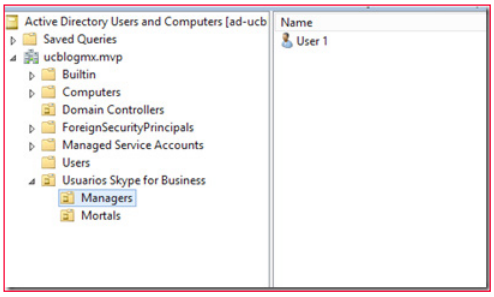


Imagen 6.- Usuario "User 1" miembro de la OU Managers.

El usuario 2 es miembro de la OU Mortals

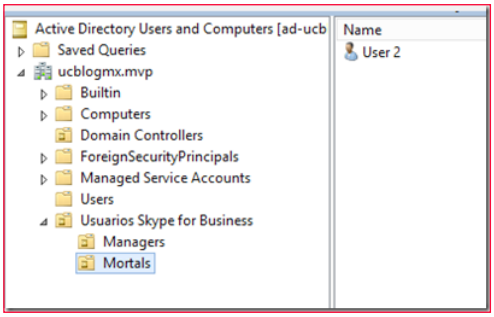


Imagen 7.- Usuario "User 2" miembro de la OU Mortals.

Ahora, la regla del Ethical wall quedo de la siguiente manera.

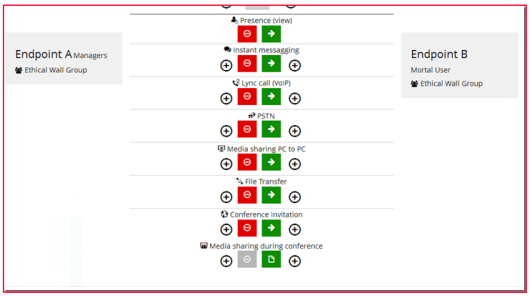


Imagen 8.- Regla del Ethical Wall.

Donde solamente los usuarios de la OU Managers, pueden comunicarse con los usuarios de la OU Mortals. A continuación, se muestra como un Gerente puede comunicarse con un usuario normal.

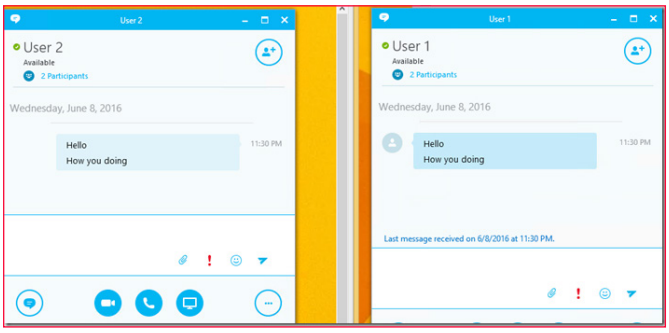


Imagen 9.- Comunicación entre User 1 y User 2.

Pero cuando un normal intenta buscar a un Gerente, se ve lo siguiente.

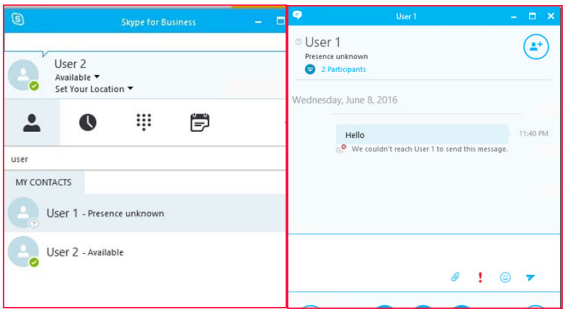


Imagen 10.- Experiencia cuando se busca un gerente.

La presencia del usuario1 no se muestra al usuario2, y cuando intenta enviar IM, el ethical wall rechaza la comunicación. Esto fue solamente un ejemplo de como funciona, pero pueden definir políticas a niveles mas avanzados no solamente de usuario a usuario.

NOTA: Agradezco a fabbricadigitale por la oportunidad y apoyo para probar Ethical Wall.

Si deseas tener mayor información sobre Ethical Wall o cualquier tema relacionado con Skype for Business o Microsoft Teams, no dudes en contactarme.

RODOLFO CASTRO AGUILAR
MVP Office Server and Services
SFB User Group: https://www.facebook.com/groups/S4BUGIberoamerica/
Twitter : @ucblogmx
http://ucblogmx.com

52

Análisis del libro "Windows Server 2016: Arquitectura y Administración de los servicios de dominio Active Directory (AD DS)"

Este libro está escrito por Jean-François Aprea, un MVP francés con amplia experiencia en sistemas Microsoft ya que trabaja como arquitecto en una gran empresa del país galo. Además, ha sido distinguido durante 10 años consecutivos como MVP de Cloud and Datacenter Management, el mismo grupo de expertise en el que estoy galardonado yo desde 2011.

Bueno, vamos a hablar del libro en sí.

Como siempre empezaré por hablar de los aspectos físicos del libro.

El libro está editado, al menos en nuestro país, como tapa blanda, por lo que os recomiendo poner algún refuerzo en las puntas de la cubierta o forrarlo si le vais a dar mucha caña como he hecho yo. Y especialmente si lo vais a llevar de aquí para allá. A pesar de ello, es tapa blanda con un encolado impecable por lo que no creo que tengáis problemas con las tapas más allá de que se os doblen y por supuesto no creo que perdáis hojas como pasa en otras editoriales.

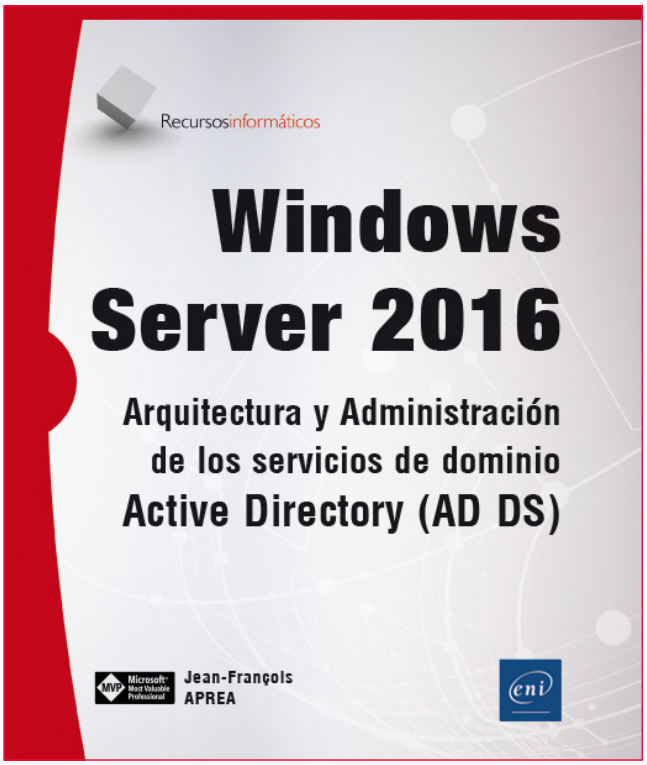


Imagen 1.- Portada del libro en su edición española.

El libro se compone de 719 páginas por lo que el precio en el que se vende me parece bastante interesante, y aunque hablaré de ello más adelante, además se puede acceder a los contenidos on-line siguiendo un proceso que básicamente es ponerse en contacto con la editorial y hacerles llegar la factura de compra.

Este acceso online tiene una gran ventaja, permite acceder a estos contenidos sin necesidad de llevarlos siempre encima, pero tiene la desventaja (a mi modo de ver) de permitir imprimir sólo pequeñas secciones y no capítulos enteros. Aun así, es un auténtico lujo al que pocas editoriales dan acceso.

"El libro está editado en nuestro país por ENI Ediciones en tapa blanda"

Al igual que el resto de libros de esta colección, el gramaje de las páginas es más que aceptable y acepta perfectamente el usar marcadores por ambas caras de la misma hoja.

Ahora vamos a revisar el contenido del libro en sí.

El libro se compone de diez capítulos que analizaremos a lo largo del resto de este artículo.

"Capítulo 1: Introducción a los servicios AD DS". Con un título tan explícito hay poco que explicar. Se trata de un capítulo de diez páginas en el que se explica para qué sirve el servicio de directorio, qué cambios revolucionarios nos trae Windows Server 2016, las novedades de todo tipo, incluyendo la seguridad, y una explicación de la evolución de AD desde Windows Server 2008 R2 hasta Windows Server 2016. También se habla de la integración de la innovación en Windows Server 2016, cerrando el capítulo con los servicios básicos y protocolos estándar. Vamos, un repaso en toda regla a lo que sería Active Directory, su evolución y sus novedades.

"Capítulo 2: DNS: Conceptos, arquitectura y administración". Éste es un capítulo bastante extenso, con ciento veinte páginas llenas de contenido. Comienza hablando sobre los orígenes y la historia de los servicios de resolución de nombres, y con una explicación de qué son y para qué sirven. Además, dedica una sección de nada más y nada

menos que seis páginas a definir la terminología de DNS de manera exhaustiva para que el lector no pase apuros posteriormente al intentar entender los fundamentos y funcionamiento de dicho servicio.

A lo largo del capítulo nos habla de la base de datos distribuida, la estructura del espacio DNS y la jerarquía de dominios, así como de registros de recursos o zonas y servidores. En este capítulo no sólo se detalla cómo desplegar y administrar el servicio de resolución de nombres, sino que poco a poco la cosa va poniéndose más complicada y por o tanto más ininteresante. Se van explicando opciones del servidor de DNS como la delegación de zonas, los reenvíos, las rutas internas, gestión de nombres multihost, limpieza de registros y caducidad de los mismos, configuración de las opciones de arranque del servidor DNS. También trata temas de seguridad como por ejemplo la protección de servidores, bloqueo de los ataques spoofing internos y externos, uso de NetBIOS, supervisión, troubleshoting y restauración del servicio, configuración de AD, etc.

No quiero extenderme en demasía por lo que no voy a entrar en mucho detalle, pero se tratan temas tan importantes como la gestión del servicio desde la línea de comandos, cosa que nos es de gran interés. En cada apartado que se ha ido viendo se ha hablado no sólo de la tecnología, sino que además nos explican buenas prácticas para su gestión.

“Capítulo 3: Integración de las zonas DNS en Active Directory”. Este capítulo parte de lo establecido en el capítulo previo, y con esto en mente empieza a construir una “desconexión” ya que comienza a trabajar con el servicio de DNS frente a el directorio activo.

¿Por qué comenzar por el capítulo anterior si este libro es de AD DS? Pues bueno, porque hay que tener una base para comenzar a trabajar. En este capítulo de unas cincuenta páginas, se habla de la integración de directorio activo con DNS de Windows Server 2000, integración con directorio activo y servidores de resolución de nombres de 2016, despliegue y replicación y por supuesto hay una extensa sección de casi veinte páginas dedicada a la seguridad de las actualizaciones dinámicas.

“El libro se compone de diez capítulos repletos de información útil y guías de gran interés”

“Capítulo 4: Servicios de ubicación AD DS y servicios DNS”. Este capítulo dedica treinta páginas al estudio del servicio de resolución para búsqueda de máquinas en un sentido o en otro. Se dedica una sección al estudio de la estructura y la integración en el directorio de AD, el estudio de los registros de ubicación de servicio, incluyendo la explicación de los registros de mantenimiento y los recursos SRV. Posteriormente se tratan los servidores DNS de tipo dinámico y estáticos o no dinámicos (que es más apropiado). A su vez tratan sobre aquellos problemas que podemos encontrarnos y de las limitaciones que se nos presenta. Por último, al igual que en el caso anterior se habla sobre el control de los registros de recursos, test que podemos hacer, borrado y reescritura, etc.

“Capítulo 5: Componentes de la estructura lógica”. Este capítulo de ochenta páginas, se dedica a hacer un estudio completo de la estructura de Active Directory. Parte de un estudio en profundidad de los dominios, tratando el funcionamiento de los containers dentro de un bosque de AD, los niveles funcionales de los dominios y las estrategias con las que se pueden gestionar, la administración y delegación de los mismos, las unidades organizativas, los árboles y los bosques. Esto es sólo un pequeño resumen de este capítulo que es imprescindible tanto para los que llegan nuevos a la administración de AD como para aquellos que ya peinan canas en estas tecnologías. En este capítulo no sólo tienes esto, si no que tienes opciones de seguridad, los pasos a seguir para actualizar desde otras versiones, etc. Otro capítulo ineludible y al que se sacará miga sin dudarlo.

“El acceso online permite acceder a todos los contenidos del libro”

“Capítulo 6: Grupos, unidades organizativas y delegación”. Este capítulo, es de gran importancia, y para mí se queda un poco corto en su alcance ya que tiene poco más de veinte páginas y como ya he dicho es un tema de gran importancia para cualquier empresa. Aun así, a pesar de ser corto, el espacio es rico en contenido ya que da un repaso bastante bueno a todo lo que cuenta de esta sección. Se tratan los diferentes tipos de grupos, el alcance de los mismos, la aplicación de las reglas, etc. Además, define cómo se aplican las estructuras de las UO y cómo se trabaja con la delegación de autoridad en estas unidades organizativas. Por supuesto no elude el entrar en el estudio de las mejores prácticas para UO.

“Capítulo 7: Fundamentos de las directivas de grupo”. Este capítulo de ciento diez páginas es de gran importancia para el correcto funcionamiento de toda infraestructura de directorio activo. Se trata de las políticas de grupo, directivas de grupo, también conocidas como GPO.

El capítulo comienza con la clásica introducción para entender que son las GPO, pasando a explicar qué aporta a la gestión de la empresa y un poco de historia sobre las políticas y su evolución. A continuación, el capítulo explora las novedades de las GPO en Windows Server 2016 y cómo afectan los clientes. Trata muchos temas, de hecho, tantos que no vamos a pararnos a explicar todos, pero sí que merece la pena hacer referencia al estudio del despliegue de las directivas, a la creación de objetos y su posterior configuración, la forma en que se ejecutan y las herramientas como la consola de GPMC para su administración. Una vez estudiado todo esto se habla de los parámetros que podemos configurar para la actualización, terminando el capítu-

lo con la delegación del control de las directivas de grupo y con recomendaciones y mejores prácticas. Este capítulo para mí es otra gran joya.

"Capítulo 8: Gestión de software con directivas de grupo". Este capítulo de aproximadamente treinta páginas que continúan tratando las directivas de grupo. Estudia la forma de desplegar aplicaciones de manera automatizada y usando distintos tipos de paquetes, trata la forma en la que se puede configurar este despliegue, cerrando el capítulo con el mantenimiento del software desplegado. Este capítulo es de gran interés en caso de no tener desplegado System Center Configuration Manager, en caso de tenerlo puede ser complementario, aunque la potencia de SCCM es superior.

"El libro trata entre otras cosas la configuración de los distintos roles de Active Directory y de las GPO"

"Capítulo 9: Configuración de los roles Active Directory". Este capítulo de casi doscientas páginas es el más importante de todos los que se tratan en el presente libro y por supuesto el de mayor contenido aportado. Parte de explicar qué son los servicios de directorio y todos los servicios asociados al mismo. A partir de aquí comienza a dividirse en secciones, tratando primero los servicios de directorio (AD DS), incluyendo la configuración y mantenimiento en el modo de Windows Server Core. La siguiente parte que trata es el Servicio de Certificados (AD CS) dándole un repaso completo, desde la explicación general de su propósito hasta la configuración, el mantenimiento y las mejoras de esta nueva versión. Este apartado es de gran importancia ya que la gestión de certificados es uno de los puntos más escabrosos dentro de la administración de las infraestructuras de Windows Server. Los siguientes puntos tratados son el servicio de federación de Active Directory (AD FS), los servicios ligeros de directorio activo (AD LDS), los servicios de derecho digitales del directorio activo (AD RMS). O sea, un capítulo repleto de contenidos útiles.

"Capítulo 10: Introducción a Azure Active Directoy". Este capítulo de treinta páginas que sirve para cerrar el libro trata de la gestión de AD dentro de Azure. Trata la amortización del gasto en infraestructura, la gestión de identidades en entornos híbridos, la migración de usuarios usando EMS y Azure AD, la dirección de identidades, el uso completo del frontal de Azure AD, las distintas versiones de Azure AD y los distintos escenarios en los que nos podemos mover. Con lo que tenemos más que terminado el repaso de todo lo que es Active Directory.

Conclusión

Este libro es muy completo, realmente completo, y de gran utilidad para cualquiera que quiera acercarse a la administración de Windows Server 2016 y su directorio activo. Si que le pongo un "pero" y es que no dediquen un capítulo a PowerShell ya que cada día tiene más influencia en toda la administración de las infraestructuras de Windows, pero no es tan problemático ya que hay un libro de la misma colección dedicado íntegramente a la programación con Windows PowerShell, libro que también recomiendo.

"Este libro goza de una gran relación calidad-precio y excelentes contenidos"

¡Y ojo! No nos olvidemos de que este libro dispone de versión digital para todos los que lo adquieran y puedan justificarlo mediante ticket de compra.

Para mí, este libro es tan redondo como el resto de los que estoy adquiriendo hasta el momento.

Gran relación calidad-precio y excelentes contenidos.

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

52

Nosotros

Alberto Diaz

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

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

Fabián Imaz

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

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

Gustavo Velez

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

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

Juan Carlos González Martín

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

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

Coordinadores de sección

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

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

GASTÓN CRUZ
Coordinador de PowerBi
gastoncruz@gmail.com

XAVIER MORERA
Coordinador de .Net
xavier@familiamorera.com

PABLO PERALTA
Coordinador de Dynamics CRM
pablop2006@gmail.com

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

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

ESTEBAN SOLANO GRANADOS
Coordinador de Xamarin
stvansolano@outlook.com

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

¿Desea colaborar con CompartiMOSS?

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

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

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

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

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

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


Comparti
MOSS