Taller

Empezando con Webhooks sobre Sharepoint Online

Índice

[1 Requerimientos 4](#_Toc497496053)

[2 Empezando con Webhooks 4](#_Toc497496054)

[3 Registrando el PostMan en Microsoft Azure Active Directory 4](#_Toc497496055)

[4 Gestión de suscripciones desde Postman 8](#_Toc497496056)

[5 Creando nuestro primer cliente 9](#_Toc497496057)

[6 Debug en local 14](#_Toc497496058)

[7 Obteniendo trazas del evento 17](#_Toc497496059)

# Requerimientos

Para poder cubrir este taller vamos a necesitar tener lo siguiente:

* Acceso a un tenant de 0365
* Acceso al portal de Azure y en concreto al Azure Active Directory asociado al tenant de 0365
* Tener instaladas las aplicaciones PostMan y Ngrok
* Visual studio 2017

# Empezando con Webhooks

El taller va a consistir en poder registrar desde PostMan suscripciones a una lista de Sharepoint Online, y poder capturar todos los eventos que en ella se produzcan.

Para conseguir esto construiremos un cliente Webhooks que reciba todos los eventos y una clase Tracer que deje traza de todo lo que ocurre en nuestro Sharepoint sobre una lista.

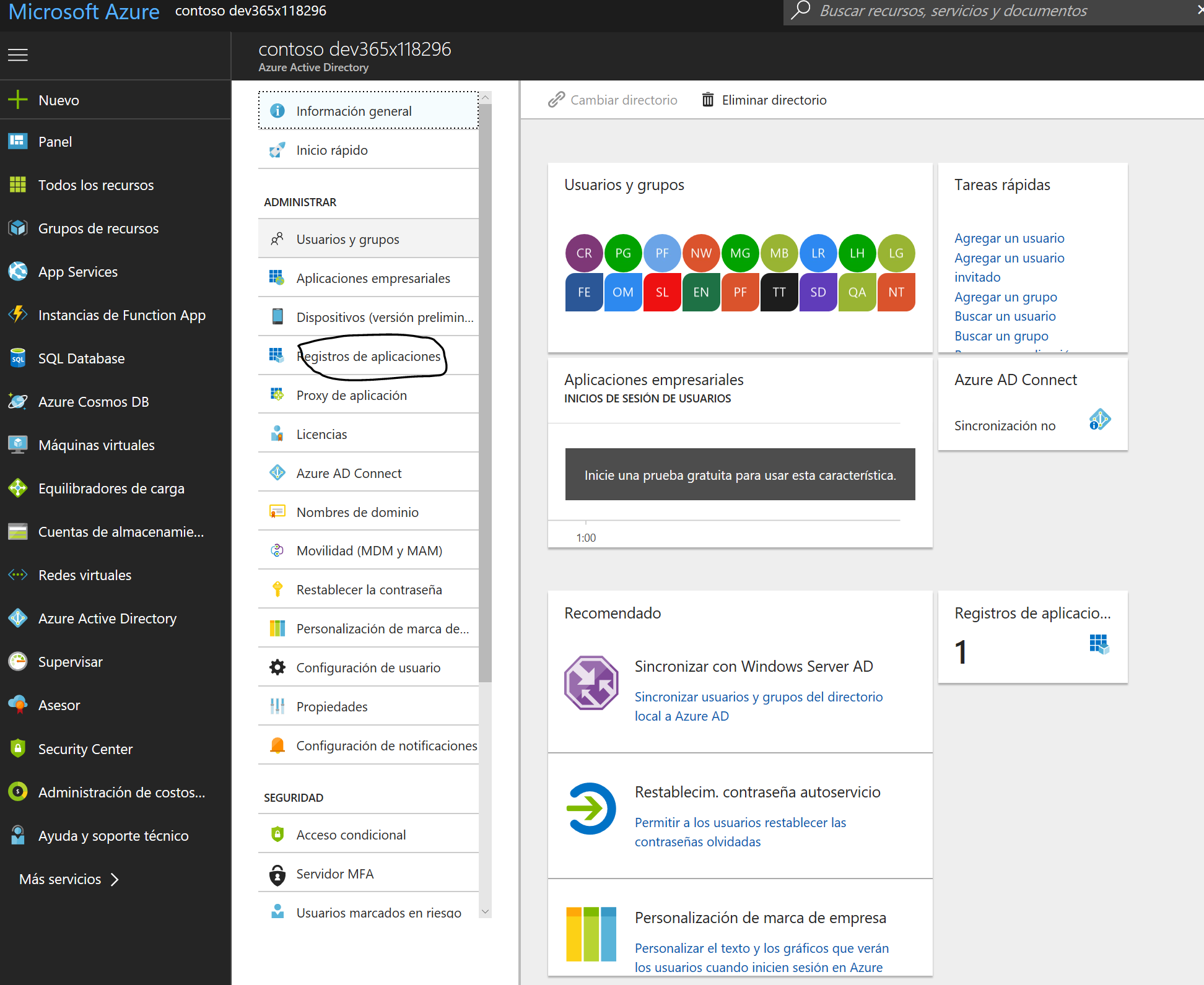
Durante el taller veremos cómo vía API de Sharepoint podemos controlar nuestras suscripciones webhooks, y como debugar en local mediante el uso de Ngrok

# Registrando el PostMan en Microsoft Azure Active Directory

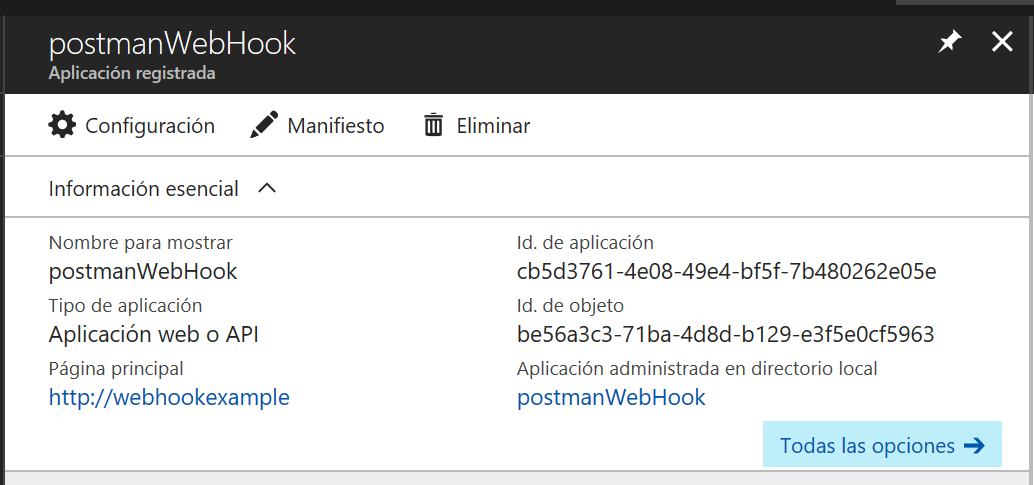
El primer paso dado que usamos la API de Sharepoint, es tener contexto de autenticación sobre nuestro tenant desde Postman, registrando una nueva aplicación en el directorio activo y dándole permisos de consulta sobre Sharepoint Online.

Para conseguirlo necesitamos seguir los siguientes pasos:

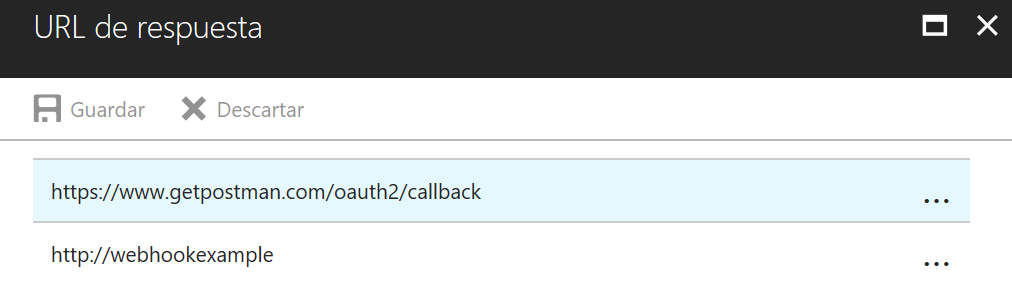
1. Accedemos al portal de Azure (<http://portal.azure.com>) con las credenciales de nuestro tenant de 0365
2. Accedemos en el panel de servicios del portal de Azure y seleccionamos Azure Active Directory, y en concreto accedemos a registro de aplicaciones.



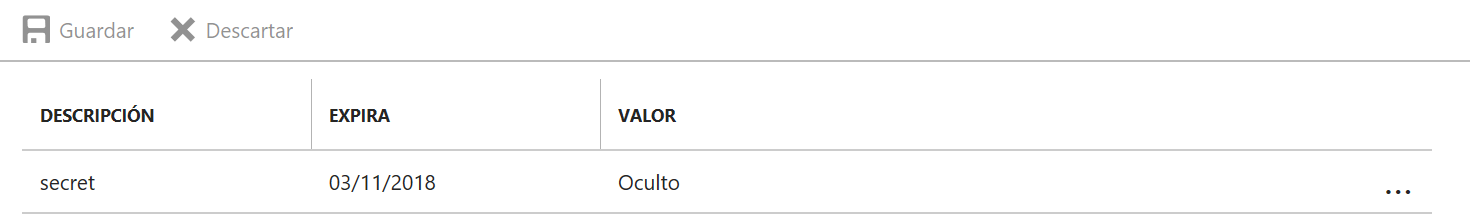
1. Añadimos una nueva aplicación dándole un nombre como por ejemplo “postManWebhook” y una url que puede ser del tipo <http://webhooksample>



1. Nos debemos anotar el “Id. De aplicación” que será nuestro “Client ID”.
2. Añadimos una nueva url de respuesta con el valor: <https://www.getpostman.com/oauth2/callback>

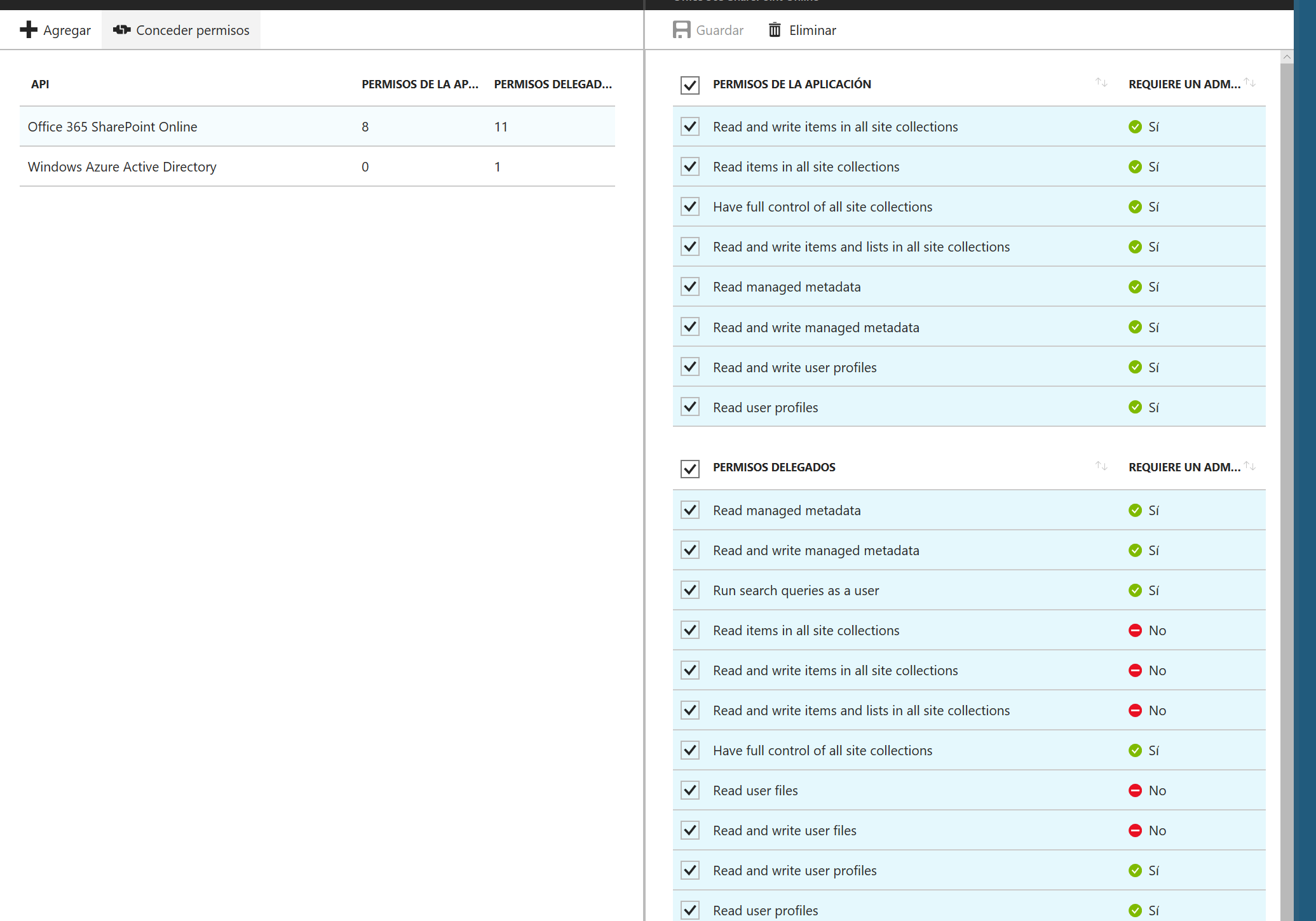


1. Creamos una nueva “key” privada en el apartado claves, con un nombre cualquiera, un 1 año de duración y una vez guardamos nos anotamos la clave que se autogenera.



1. Una vez tengamos el “Client ID y el Client Secret” guardados y anotados, vamos a dar permisos sobre Sharepoint Online. Para ello accedemos al panel de “Permisos necesarios”, añadimos nuevos permisos y seleccionamos Office 365 Sharepoint Online.

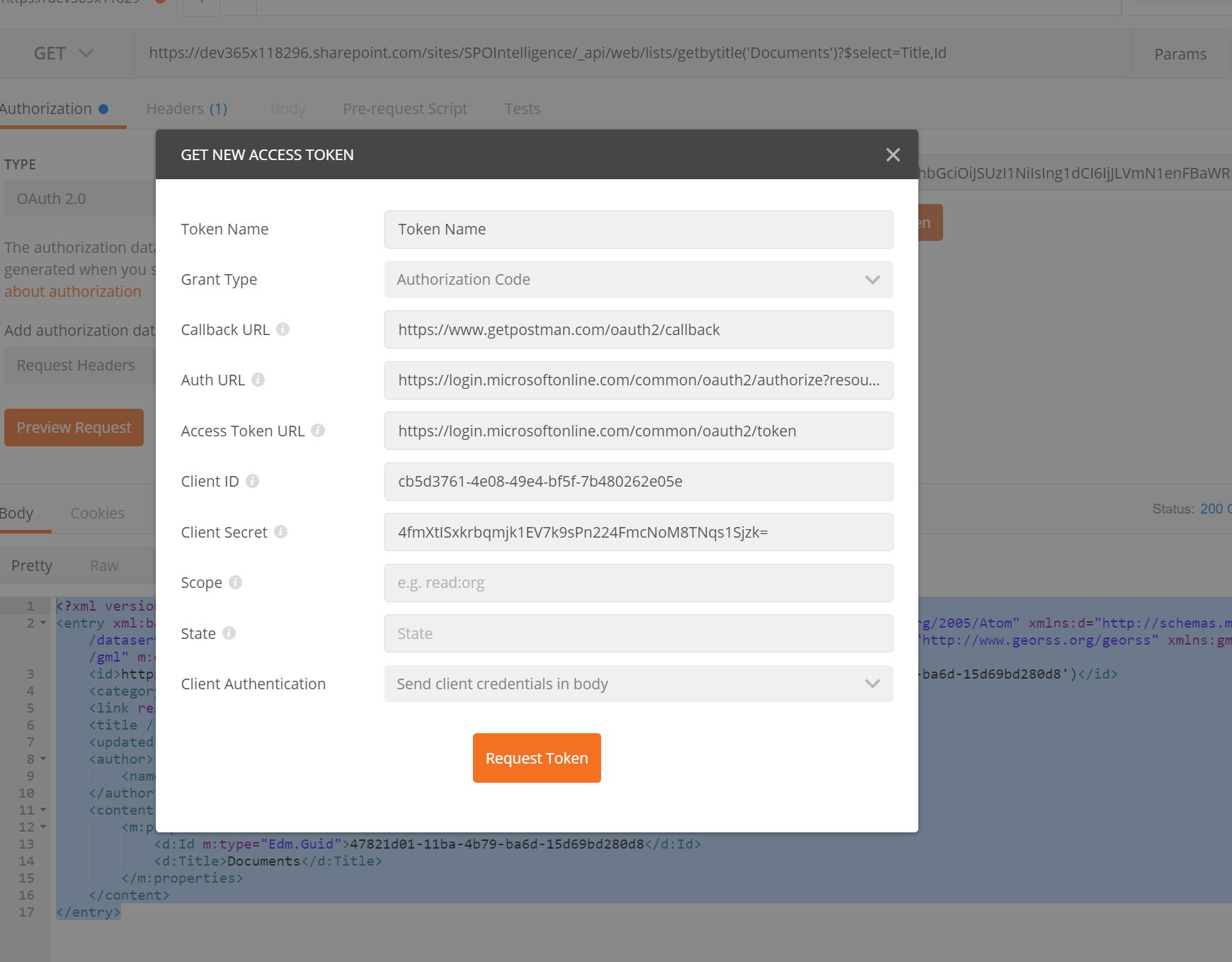
Vamos a dar permisos a todo, por lo que seleccionamos todas las opciones tal cual se ve en la imagen.



1. Una vez realizado los pasos anteriores, nos quedaría irnos a nuestro Postman (Native App o desde Google Chrome), y creamos una nueva llamada.

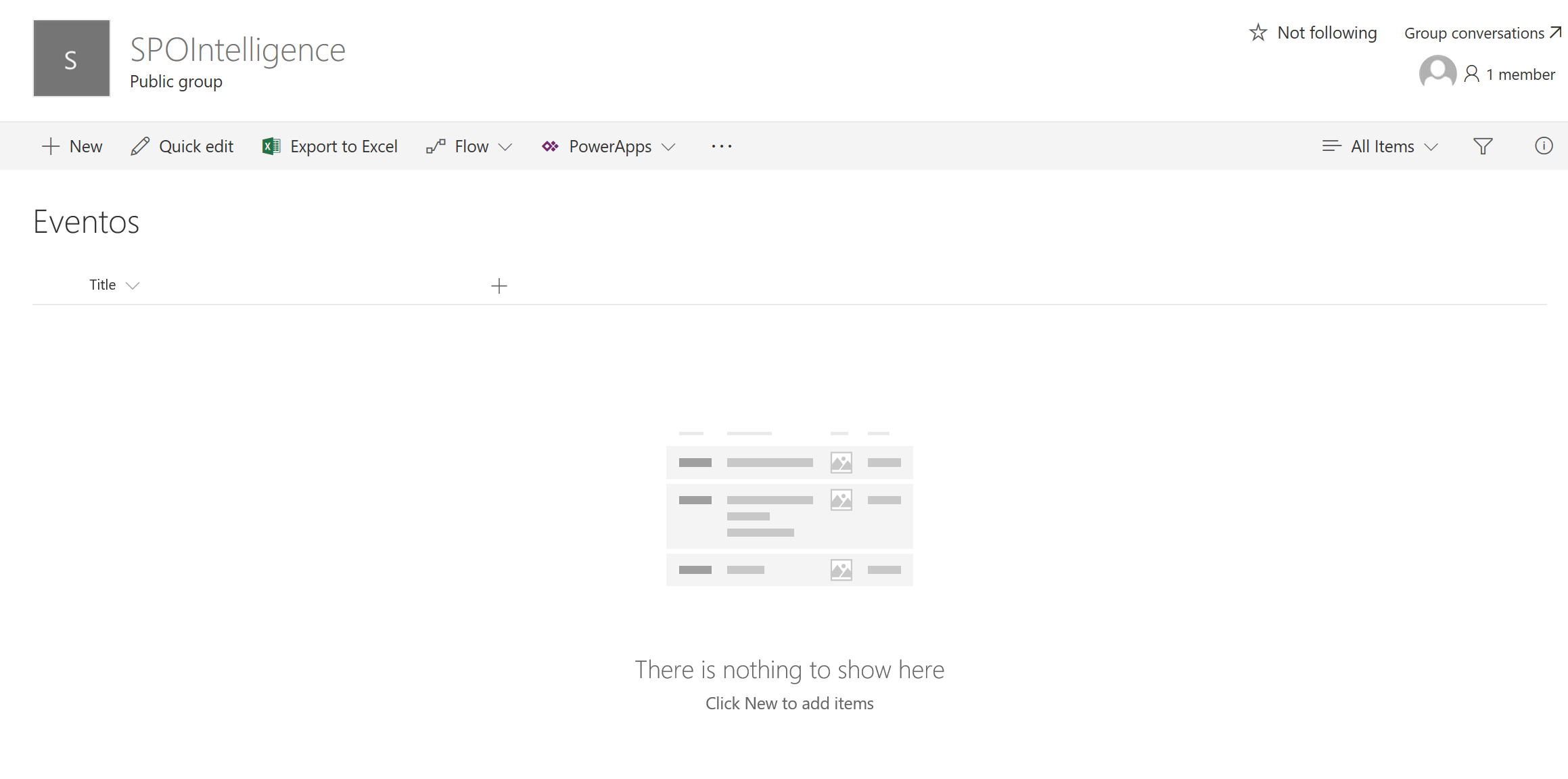
En el apartado Authorization seleccionamos “Get New Access Toke” y rellenamos el formulario que se nos presenta:

* **Callbak url**: <https://www.getpostman.com/oauth2/callback>
* **Auth URL**: [https://login.microsoftonline.com/common/oauth2/authorize?resource=https%3A%2F%2F<your-sharepoint-root-site-collection-url-without-https](https://login.microsoftonline.com/common/oauth2/authorize?resource=https%3A%2F%2F%3cyour-sharepoint-root-site-collection-url-without-https)**>**
* **Token URL**:
* **Client ID:** Id. Aplicación anotada en el paso 4
* **Client Secret:** Clave secreta generada en el portal en el paso 6
* **Token Name**: “Webhook sample”
* **Client Authentication**: Sent client credentials in body (Solo en la versión App Native)
* **Grant Type**: Authorization code

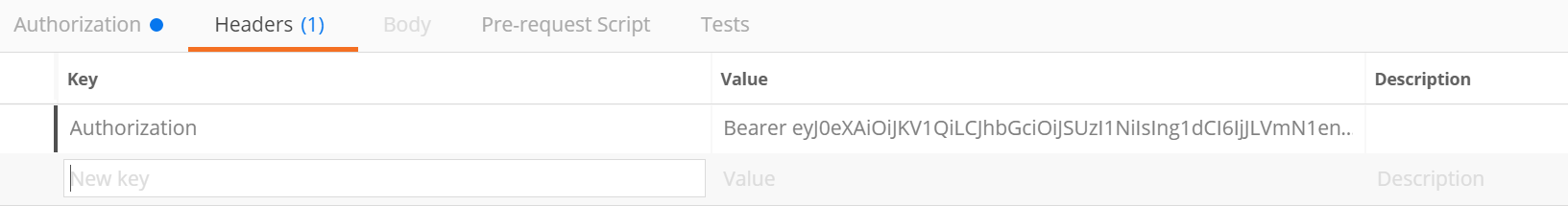


Una vez seleccionemos “Request Token” nos pedirá credenciales de O365, ahí introducimos las del tenant proporcionado para el taller.

1. Creamos un sitio de Sharepoint en nuestro tenant con el nombre “SPOIntelligence”.
2. Creamos una lista con el nombre “Eventos” en el nuevo site creado, que tenga los campos de serie, nos vale con que tenga un campo “Title.

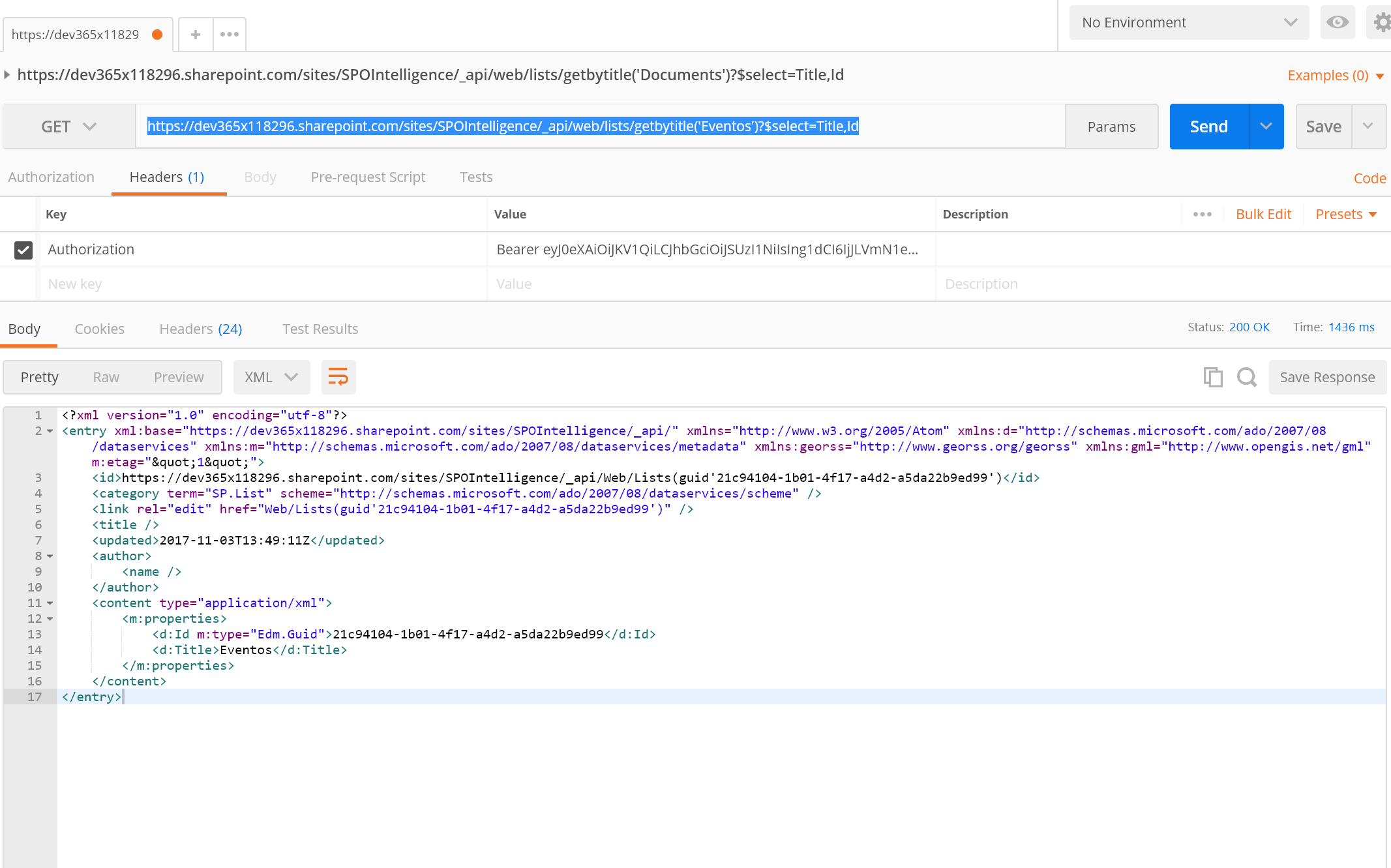


1. Probamos el acceso desde postMan a esta nueva lista ejecutando la siguiente consulta:
   1. Usamos el token generado anteriomente seleccionado en App Native “Preview Request” con Request Headers o bien en la aplicación de Google Chrome seleccionamos “Use Token”. En ambos casos nos aparecerá una cabecera del siguiente estilo:



* 1. Construimos una petición **GET** a la Url **<urlTenant>/**sites/SPOIntelligence/\_api/web/lists/getbytitle('Eventos')?$select=Title,Id

Esto nos devolverá un XML del cual nos anotaremos el campo **GUID** de la lista



# Gestión de suscripciones desde Postman

Una vez tenemos creada la app en Azure Active Directory y tenemos el GUID de la lista eventos, podemos utilizar los siguientes comandos desde postMan y haciendo uso del mismo Token para crear,modificar o cancelar una suscripción.

Añadir una nueva suscripción

* Petición POST
* Url: <https://site-collection/_api/web/lists('list-id')/subscriptions>
* Headers:
  + Authorization: Bearer “Token”
  + Accept: application/json
  + Content-Type: application/json
* Body: Seleccionamos RAW y añadimos el siguiente JSON:

{

"resource": "https://site-collection/\_api/web/lists('list-id')",

"notificationUrl": "https://ngrok-forwarding-address/api/webhook/handlerequest",

"expirationDateTime": "2018-10-27T16:17:57+00:00",

"clientState": "A0A354EC-97D4-4D83-9DDB-144077ADB449"

}

Listar suscripciones existentes

* Petición GET
* Url: <https://site-collection/_api/web/lists('list-id')/subscriptions>
* Headers:
  + Authorization: Bearer “Token”

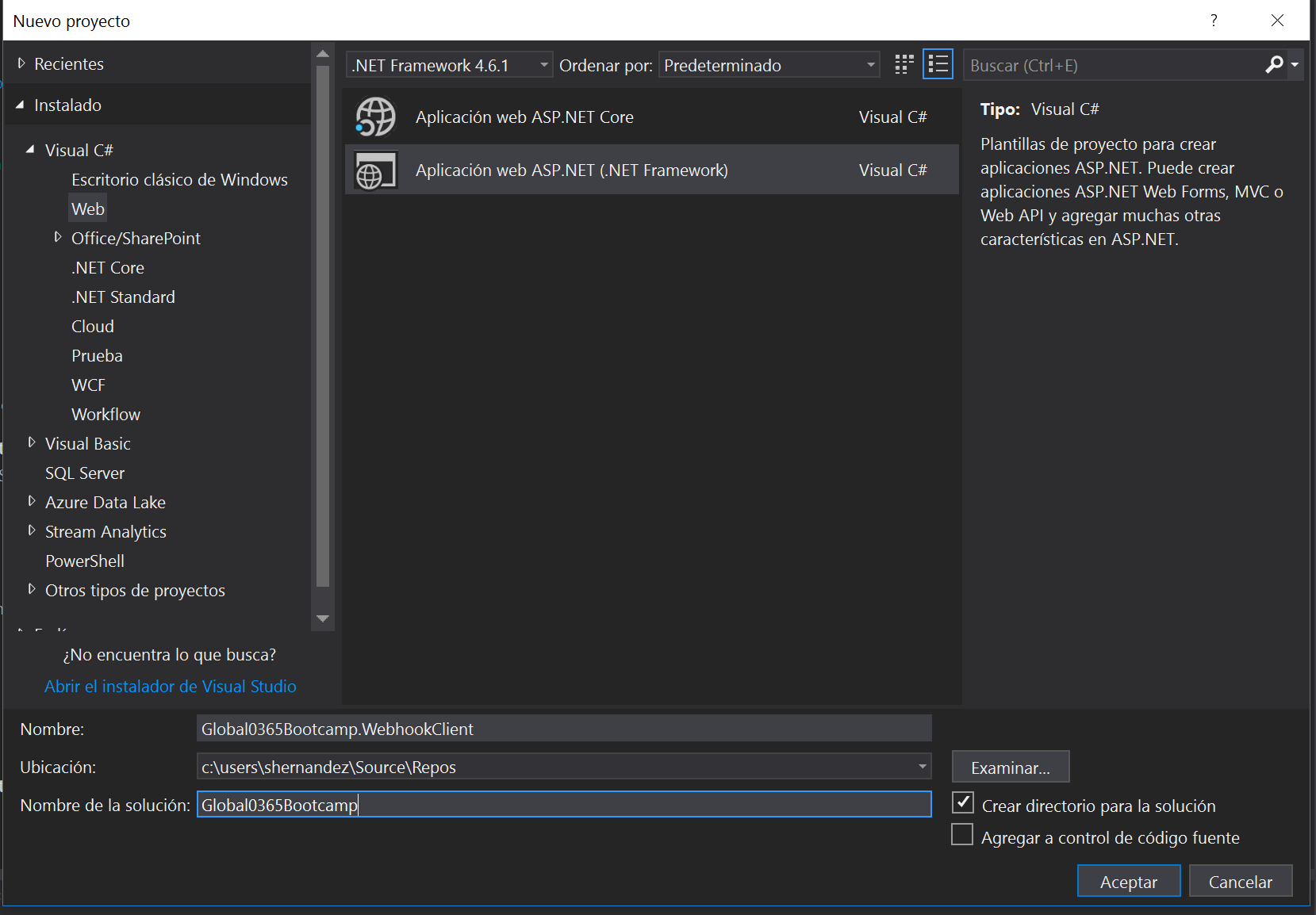
Borrar una suscripción

* Petición DELETE
* Url: <https://site-collection/_api/web/lists('list-id')/subscriptions>**(‘idSuscripcion’)**
* Headers:
  + Authorization: Bearer “Token”

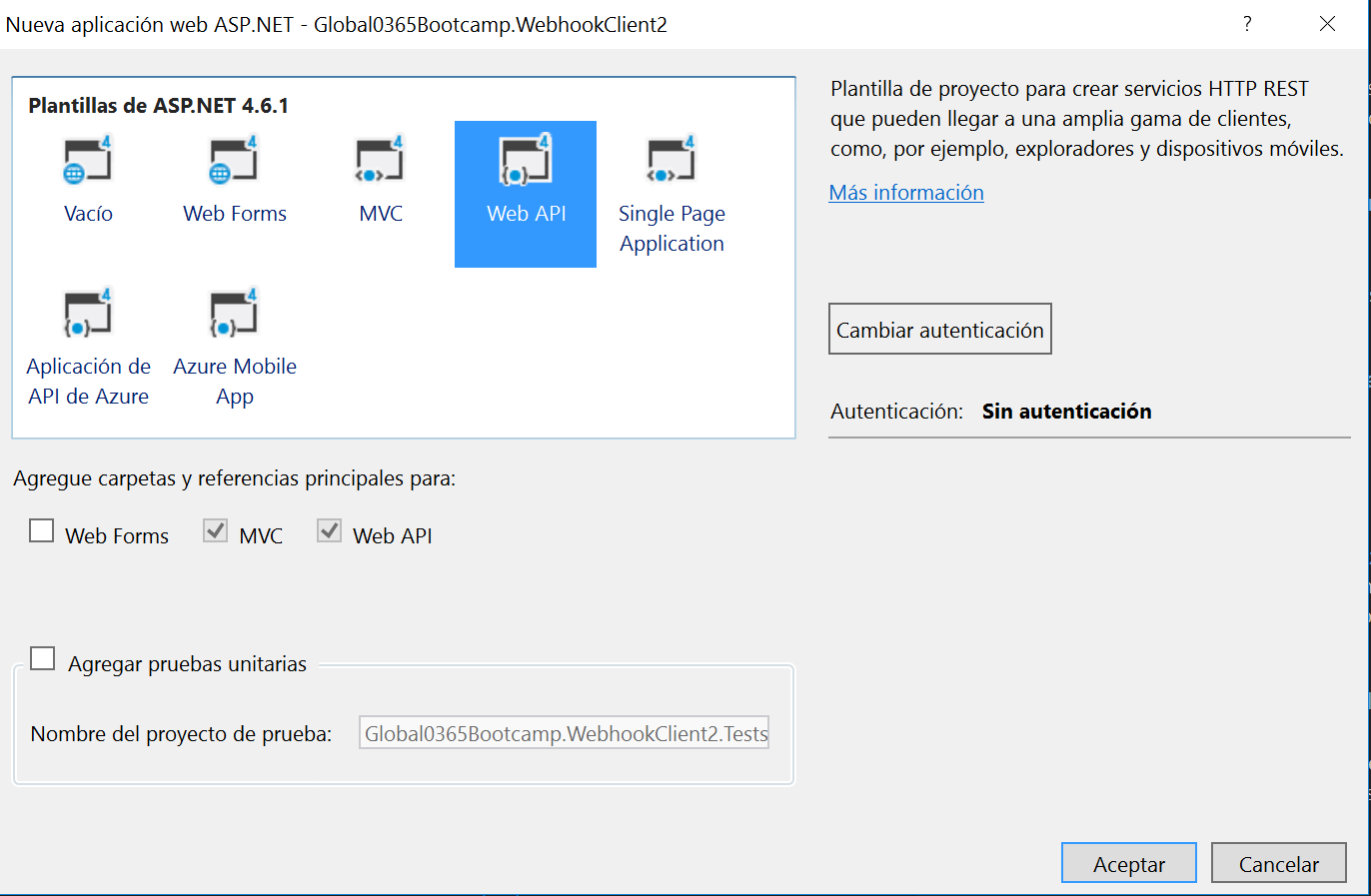
# Creando nuestro primer cliente

Para crear el cliente webhook que va a recibir los eventos necesitaremos ejecutar los siguiente pasos:

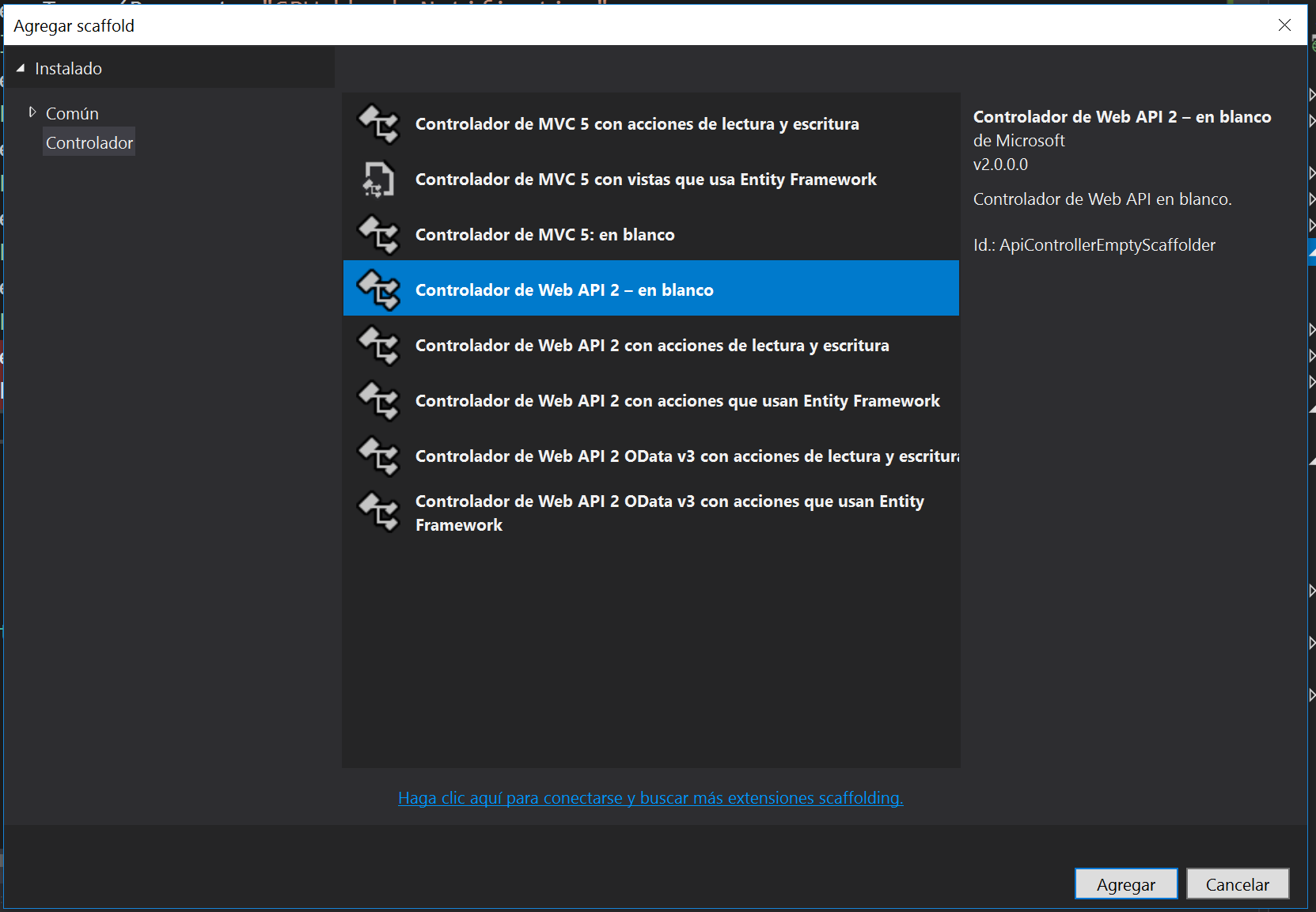
1. Crear un proyecto en VisualStudio del tipo Web->ASP.NET Web Application



1. Escogemos una aplicación WebApi sin autenticación



1. Creamos un nuevo controller que llamaremos **WebhookController**



1. Creamos en la carpeta Model los siguiente dos modelos:
   * WebhookNotifications

public class WebhookNotification

{

[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; }

}

* + WebhookContents

public class WebhookContent<T>

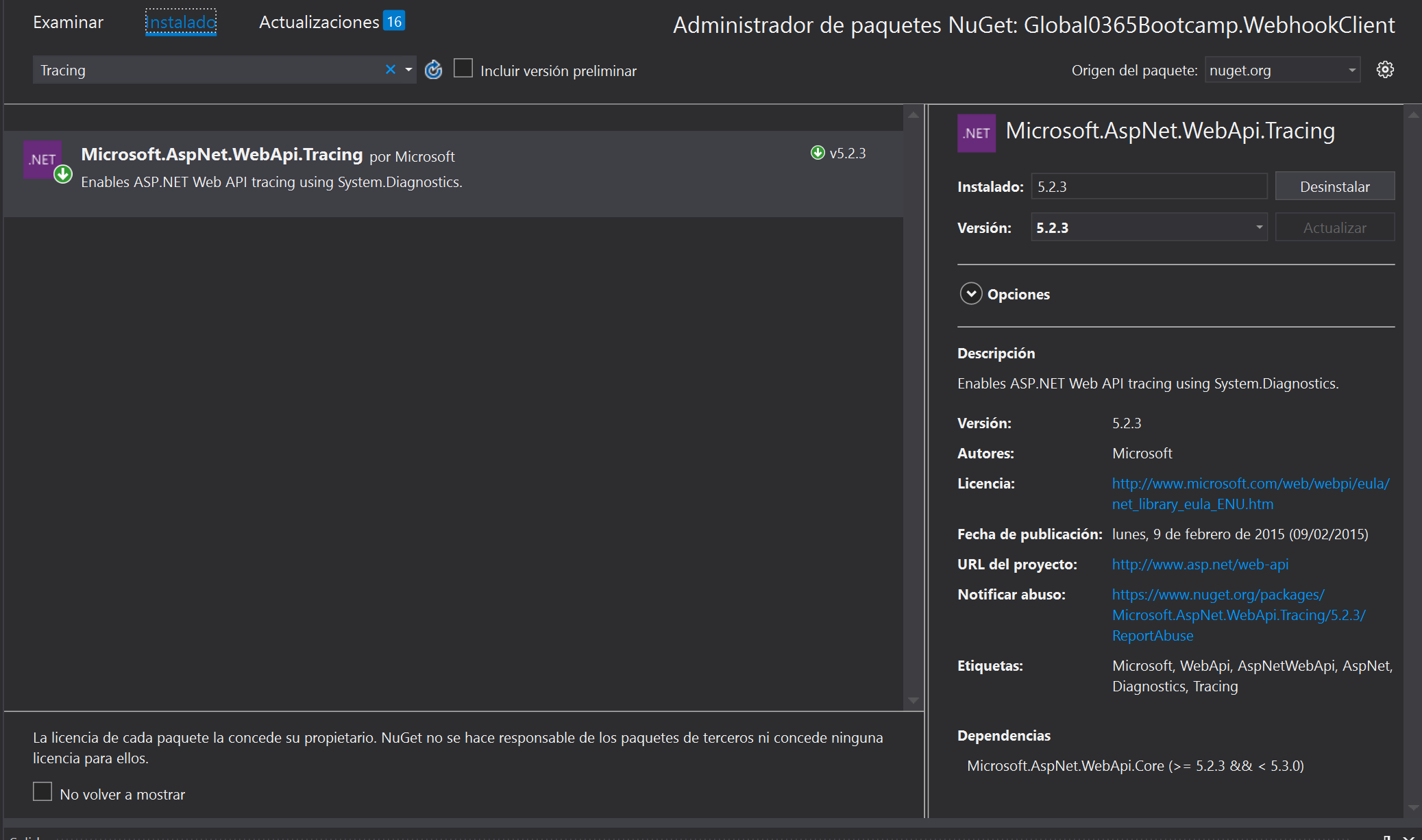
{

[JsonProperty(PropertyName = "value")]

public List<T> Value { get; set; }

}

1. Añadimos el siguiente paquete Nuget para poder tracear las ejecuciones de nuestro cliente



1. Añadimos al web config las siguientes entradas:
   * **En AppSettings**

<add key="webhookclientstate" value="A0A354EC-97D4-4D83-9DDB-144077ADB449" />

* + **En System.web**

<trace enabled="true"></trace>

1. Añadimos a **webhookController** el siguiente código:

using Global0365Bootcamp.WebhookClient.Models;

using Newtonsoft.Json;

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Web;

using System.Web.Http;

using System.Web.Http.Tracing;

public class WebhookController : ApiController

{

[HttpPost]

public HttpResponseMessage HandleRequest()

{

var traceWriter = Configuration.Services.GetTraceWriter();

HttpResponseMessage httpResponse = new HttpResponseMessage(HttpStatusCode.BadRequest);

string validationToken = string.Empty;

IEnumerable<string> clientStateHeader = new List<string>();

string webhookClientState = ConfigurationManager.AppSettings["webhookclientstate"].ToString();

if (Request.Headers.TryGetValues("ClientState", out clientStateHeader))

{

string clientStateHeaderValue = clientStateHeader.FirstOrDefault() ?? string.Empty;

if (!string.IsNullOrEmpty(clientStateHeaderValue) && clientStateHeaderValue.Equals(webhookClientState))

{

var queryStringParams = HttpUtility.ParseQueryString(Request.RequestUri.Query);

if (queryStringParams.AllKeys.Contains("validationtoken"))

{

httpResponse = new HttpResponseMessage(HttpStatusCode.OK);

validationToken = queryStringParams.GetValues("validationtoken")[0].ToString();

httpResponse.Content = new StringContent(validationToken);

return httpResponse;

}

else

{

var requestContent = Request.Content.ReadAsStringAsync().Result;

if (!string.IsNullOrEmpty(requestContent))

{

WebhookNotification notification = null;

try

{

var objNotification = JsonConvert.DeserializeObject<WebhookContent<WebhookNotification>>(requestContent);

notification = objNotification.Value[0];

}

catch (Exception e)

{

return httpResponse;

}

if (notification != null)

{

httpResponse = new HttpResponseMessage(HttpStatusCode.OK);

traceWriter.Trace(Request, "SPWebhook Notification",

TraceLevel.Info, string.Format("Resource: {0}", notification.Resource));

traceWriter.Trace(Request, "SPWebhook Notification",

TraceLevel.Info, string.Format("SubscriptionId: {0}", notification.SubscriptionId));

traceWriter.Trace(Request, "SPWebhook Notification",

TraceLevel.Info, string.Format("TenantId: {0}", notification.TenantId));

traceWriter.Trace(Request, "SPWebhook Notification",

TraceLevel.Info, string.Format("SiteUrl: {0}", notification.SiteUrl));

traceWriter.Trace(Request, "SPWebhook Notification",

TraceLevel.Info, string.Format("WebId: {0}", notification.WebId));

traceWriter.Trace(Request, "SPWebhook Notification",

TraceLevel.Info, string.Format("ExpirationDateTime: {0}", notification.ExpirationDateTime));

}

}

}

}

else

{

httpResponse = new HttpResponseMessage(HttpStatusCode.Forbidden);

}

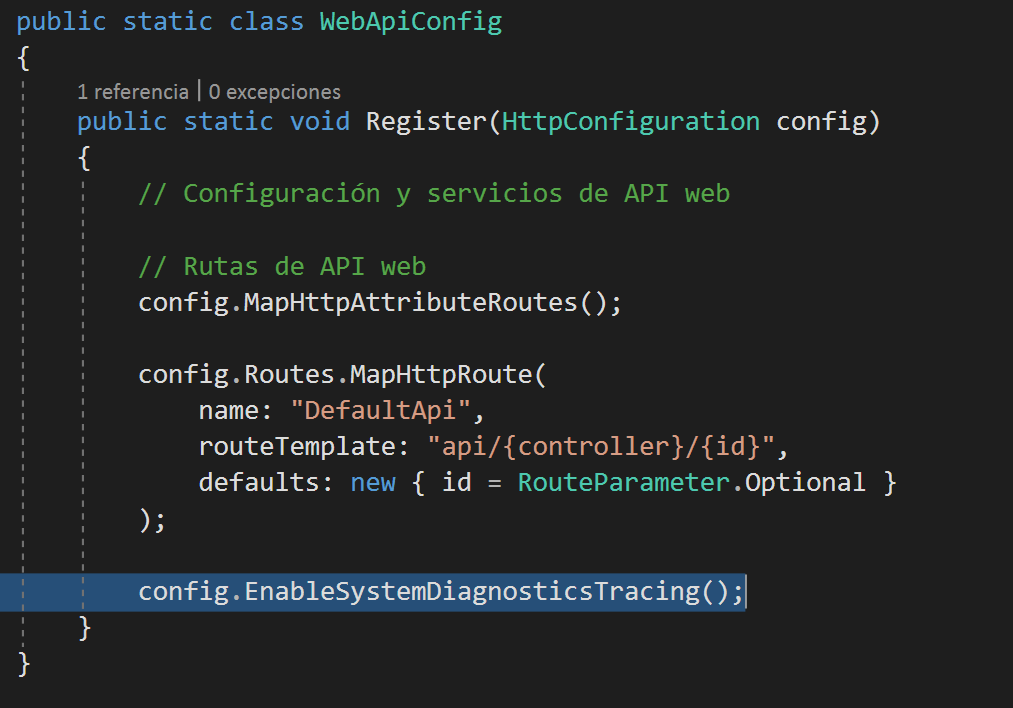
}

return httpResponse;

}

}

1. Añadimos al fichero AppStart->WebApiConfig la siguiente línea:



# Debug en local

Para debugar nuestro cliente webhook en local debemos lo primero obtener la aplicación “ngrok” que encontrareis en la solución descargada desde GitHub.

Una vez tenemos el cliente debemos seguir los siguiente pasos:

1. Abrimos una consola de Windows
2. Nos posicionamos en el directorio donde hemos descargado ngrok:

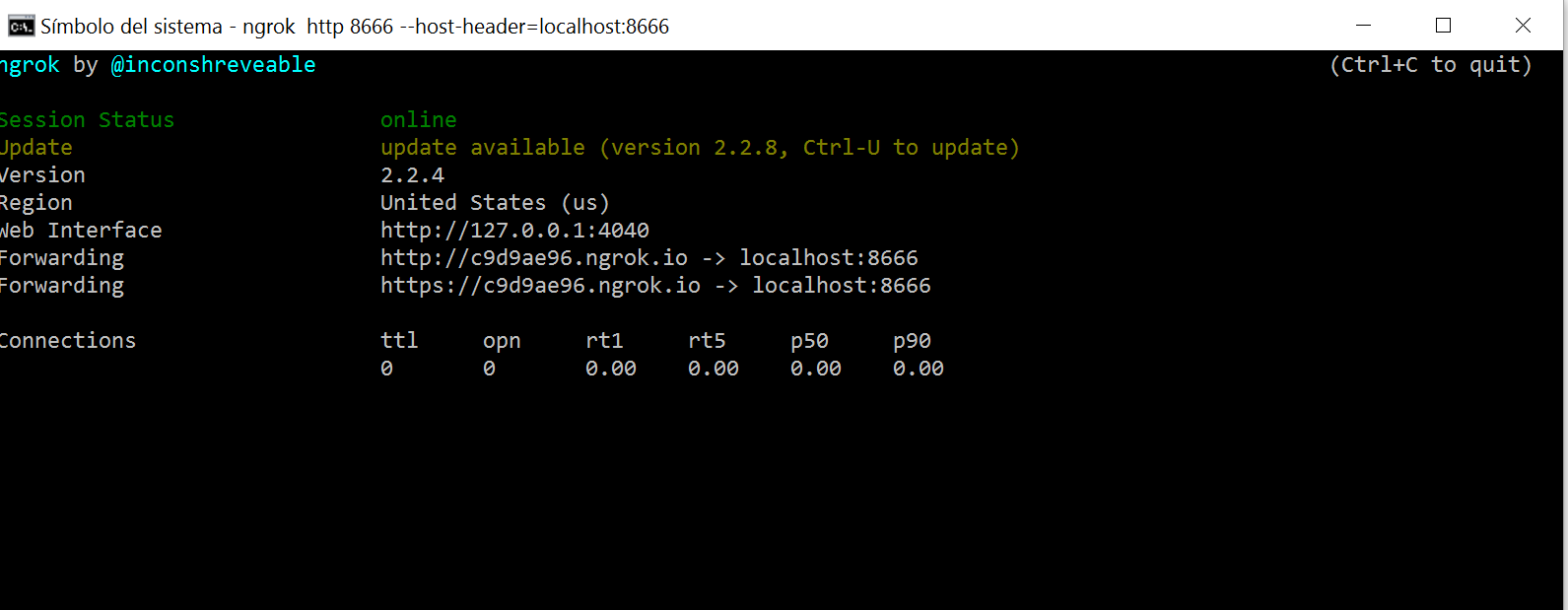
**“cd c:\<carpeta de la app >”**

1. Lanzamos nuestra cliente webhook en “debug” desde el visual studio y copiamos el puerto por el que se está ejecutando la api.
2. Ejecutamos el siguiente comando:

**ngrok http <port-number> --host-header=localhost:<port-number>**

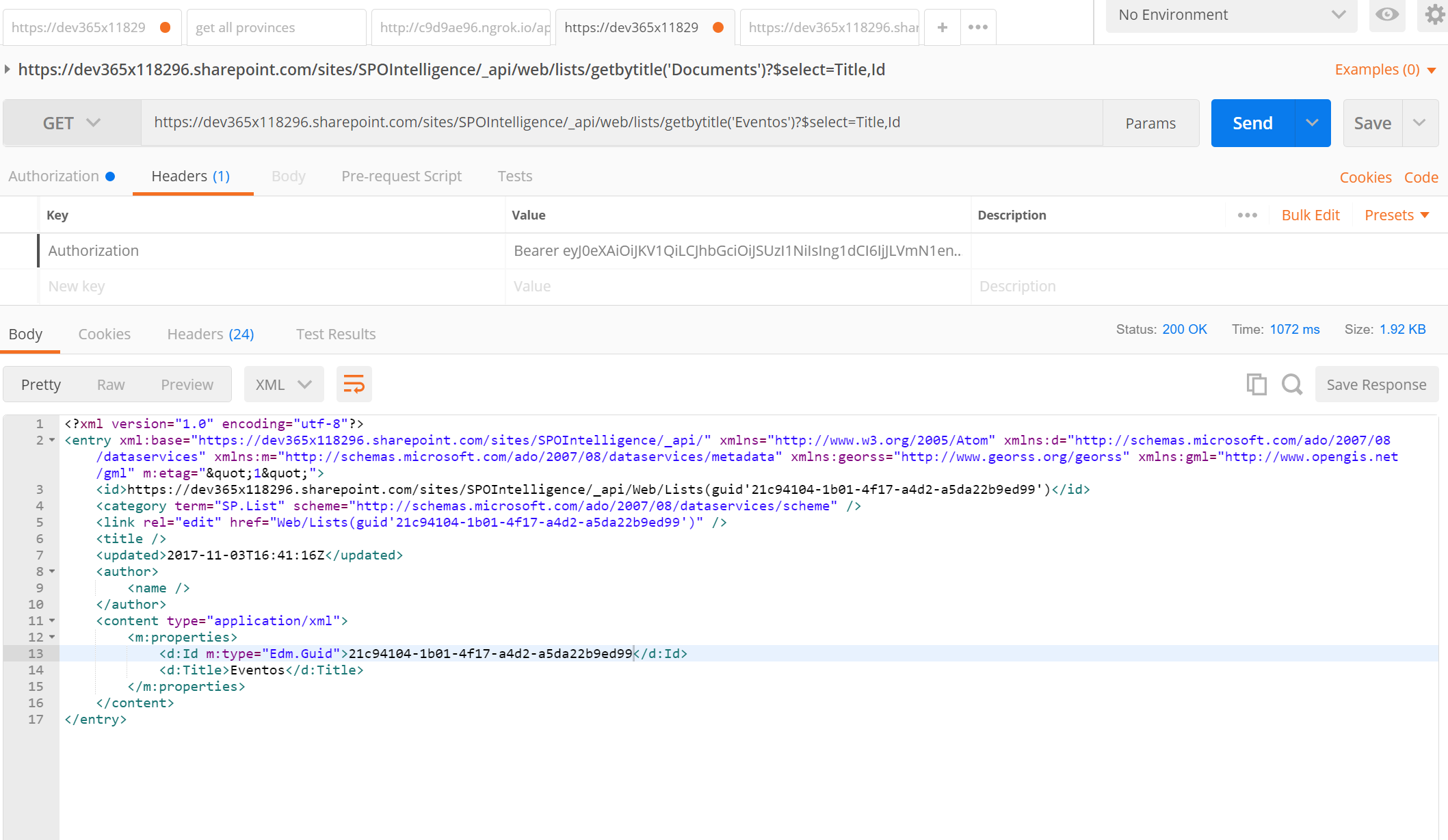
Donde “port-number” es el puerto de nuestra api.

1. Copiamos de la consola la url de Forwarding que vamos a pasarle en el cuerpo de la petición desde el postman para crear una suscripción contra nuestra lista en el campo “notificationUrl “.



1. Ejecutamos en este orden las siguientes peticiones con el PostMan:
   1. Refrescamos el TOKEN contra nuestro Sharepoint Online
   2. Obtenemos el “GUID” de nuestra lista EVENTOS para crear la suscripción:

**https://myTenant.sharepoint.com/sites/SPOIntelligence/\_api/web/lists/getbytitle('Eventos')?$select=Title,Id**

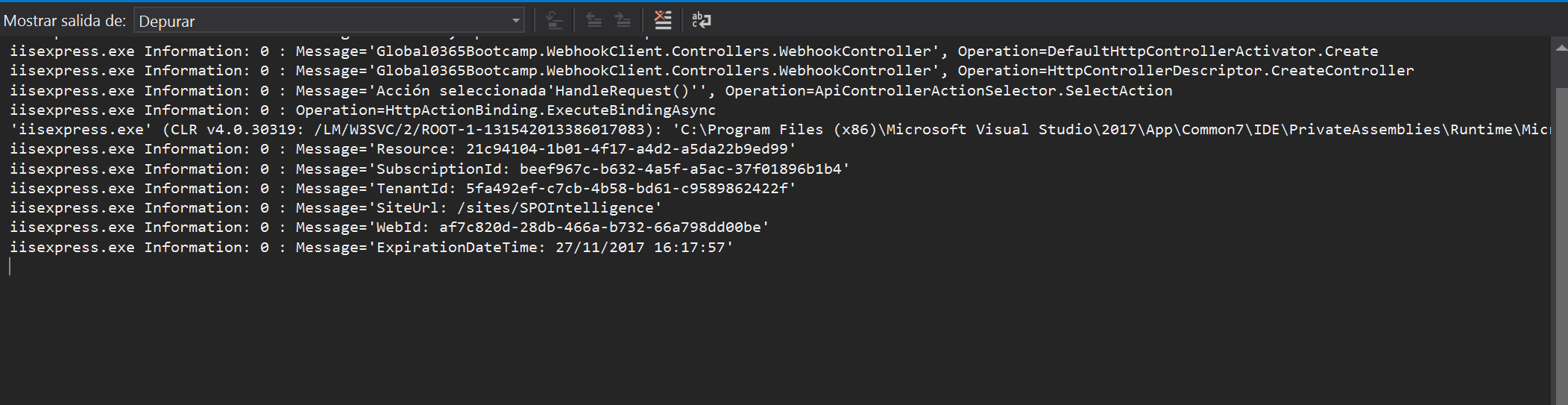


* 1. Creamos una suscripción pasando como URL de respuesta la generada por ngRok y concatenando el route de nuestra api, que será del estilo “http://**guid**.ngrok.io/api/webhook/HandleRequest





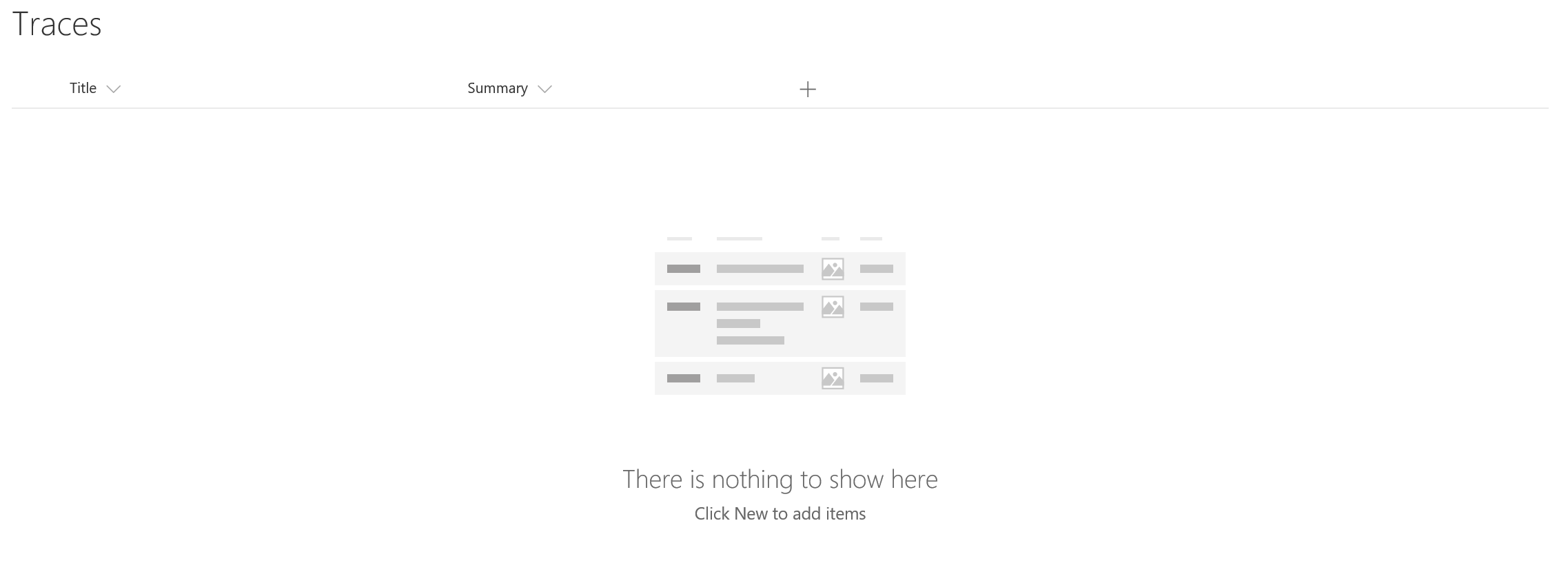
* 1. Al asociar la suscripción se lanzará el debug en nuestro visual studio, le dejamos terminar
  2. Una vez creada la suscripción añadimos un elemento a la lista y vemos como se ejecuta en visualstudio nuestro cliente webhook y por la consola aparece la traza del evento producido.



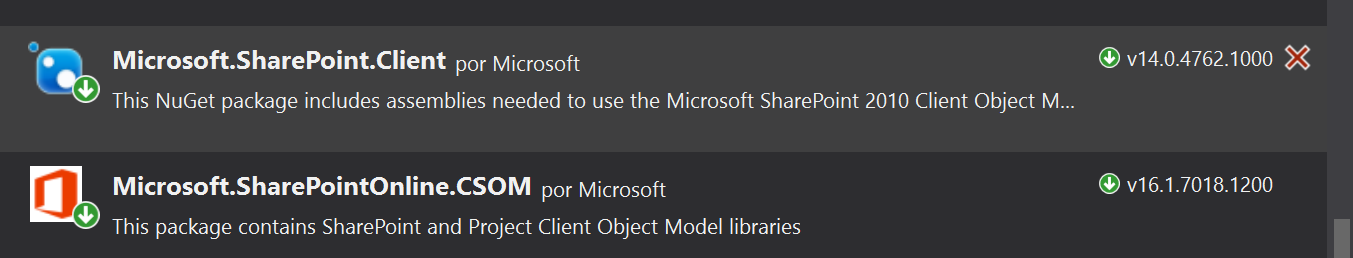
# Obteniendo trazas del evento

Ahora mismo tenemos un cliente que capta el “evento”, pero no tenemos datos de que ha sucedido realmente, simplemente sabemos que ha pasado algo.

1. Vamos añadir una segunda lista de Sharepoint que llamaremos Traces, y que tenga un campo de varias líneas de texto llamado “Summary”.



1. Añadimos los siguientes paquetes Nuget para poder consultar nuestro Sharepoint Online.



1. Para añadir la traza creamos una carpeta Shared en nuestro API, y añadimos una clase Tracer con el siguiente código:

using Microsoft.SharePoint.Client;

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Linq;

using System.Security;

using System.Web;

namespace Global0365Bootcamp.WebhookClient.Shared

{

public class Tracer

{

private readonly string app = ConfigurationManager.AppSettings["app"];

private readonly string userName = ConfigurationManager.AppSettings["userName"];

private readonly string passWord = ConfigurationManager.AppSettings["passWord"];

private readonly string traceLibrary = ConfigurationManager.AppSettings["traceLibrary"];

public bool AddTrace(string listId, string web)

{

bool success = false;

var url = string.Format(app, web);

try

{

using (ClientContext client = new ClientContext(url))

{

client.Credentials = new SharePointOnlineCredentials(userName, GetPassword(passWord));

var originList = client.Web.Lists.GetById(new Guid(listId));

client.Load(originList);

client.ExecuteQuery();

ChangeQuery changes = new ChangeQuery(true, true);

changes.Item = true;

changes.RecursiveAll = true;

changes.Add = true;

changes.User = true;

changes.Update = true;

changes.List = true;

changes.Field = true;

if (originList != null)

{

var listChages = originList.GetChanges(changes);

client.Load(listChages);

client.ExecuteQuery();

var result = listChages.LastOrDefault();

if (result != null)

{

var itemChange = (ChangeItem)result;

var type = result.ChangeType;

if (type != ChangeType.NoChange)

{

var traceList = client.Web.Lists.GetByTitle(traceLibrary);

client.Load(traceList);

client.ExecuteQuery();

ListItemCreationInformation createInfo = new ListItemCreationInformation();

var item = traceList.AddItem(createInfo);

item["Title"] = string.Format("Trace {0}\_{1} {2}",originList.Title, itemChange.ItemId, DateTime.Now.ToShortDateString());

item["Summary"] = string.Format("Se ha producido un evento del tipo {0}", itemChange.ChangeType);

item.Update();

client.ExecuteQuery();

}

else

success = true;

}

}

}

}

catch (Exception e)

{

//logger

}

return success;

}

public static SecureString GetPassword(string pass)

{

SecureString result = new SecureString();

pass.ToCharArray().ToList().ForEach(p => result.AppendChar(p));

return result;

}

}

}

1. Añadimos al web.config las siguientes entradas en AppSettings, para poder conectanos con CSOM a Sharepoint por client credentials:

<add key="app" value="https://dev365x118296.sharepoint.com/{0}" />

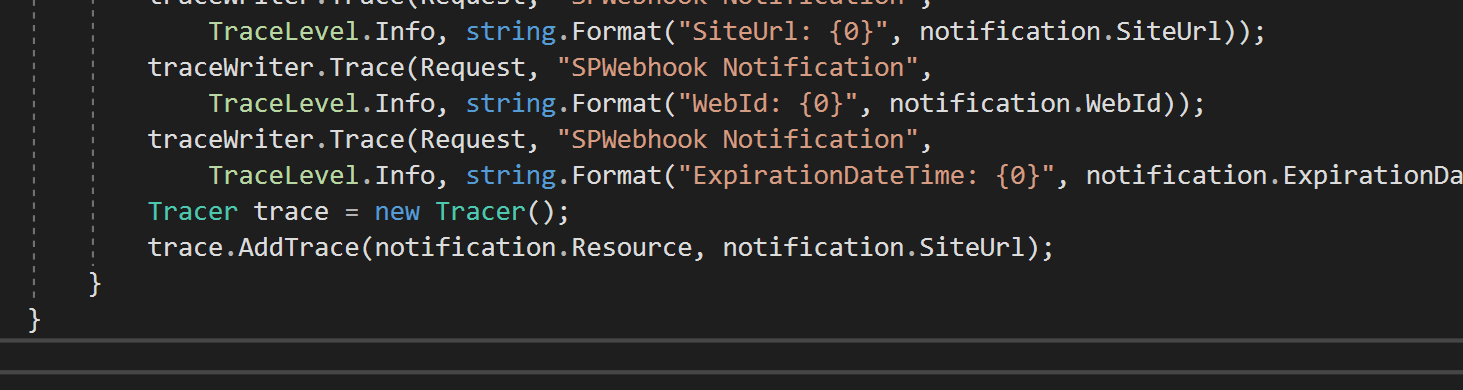
<add key="site" value="https://dev365x118296.sharepoint.com/sites/SPOIntelligence" />

<add key="userName" value="admin@DEV365x118296.onmicrosoft.com" />

<add key="passWord" value="bootcamp1104!" />

<add key="traceLibrary" value="Traces" />

1. Añadimos a nuestro cliente el siguiente código, justo después del bloque de TraceWriters que teníamos inicialmente:



1. Si lanzamos el cliente de nuevo, sin haber cerrado el ngrok (si no es así debemos crear una nueva suscripción con la nueva url que nos devuelve ngrok) y añadimos un elemento a Eventos, podemos comprobar si la traza del evento se ha añadido a nuestra lista Traces.

