Twitter Sentiment Analysis Workshop

# Pre-requisitos:

1. **Internet**
2. **Suscripción de Azure**
3. **Cuenta de Desarrollo de Twitter**
4. **Postman**
5. **Visual Studio Code**
6. **.NET 6 SDK**

# Pasos de desarrollo de este Workshop:

## **Crear una cuenta de desarrollador en Twitter**

Para habilitar las funciones de desarrollador en una cuenta de Twitter existente se deben seguir los siguientes pasos:

**Paso 1:** Acceder a la siguiente URL: https://developer.twitter.com/en/apply-for-access

**Paso 2:** Iniciar sesión en la cuenta de Twitter que será usada para el acceso de desarrollador.

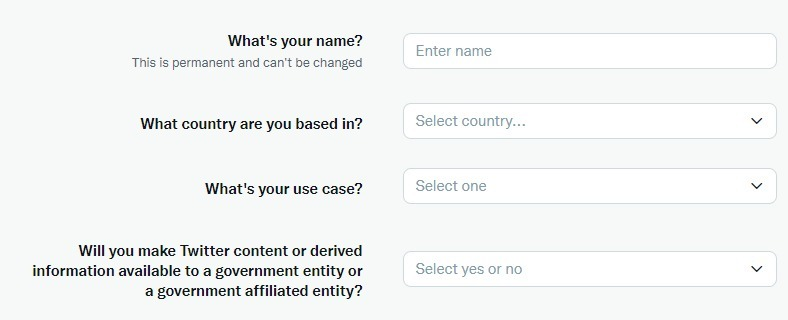
**Paso 3:** Diligenciar la siguiente información solicitada por el formulario:

- Nombre

- País

- Caso de uso (Para este taller, seleccionar Estudiante)

- Seleccionar **No** en la opción “Will you make Twitter content or derived information available to a government entity or a government affiliated entity?”



Al hacer envío de este formulario, habrá que esperar la aprobación de Twitter para este acceso. Por esto es importante realizar este proceso lo más pronto posible antes del taller.

## Instalar el SDK de .Net 6.0 y VS Code

Esta es una guía paso a paso para programadores principiantes, intermedios y experimentados para configurar el entorno y crear aplicaciones .net con Visual Studio Code + C# 10 y .net 6.0

Comencemos con la instalación, primero necesitamos instalar .net 6.0 SDK primero.

* **PASO 1** - Descargue .net 6 SDK desde el sitio oficial de Microsoft [aquí](https://dotnet.microsoft.com/en-us/download).

Graphical user interface, application, Teams

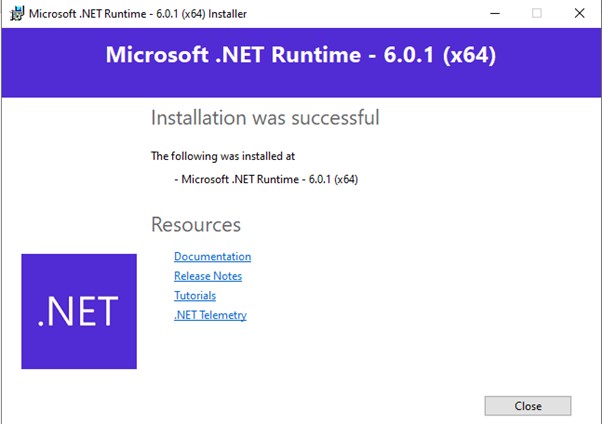
Description automatically generated

* **PASO 2** – Seleccione el sistema operativo en el que se realizara la instalación y de clic la opción de acorde con la arquitectura de su procesador.

Graphical user interface, application, Teams

Description automatically generated

* **PASO 3** – abra el archivo de descarga y complete la instalación.



* **PASO 4** - Vamos a comprobar la versión instalada en su sistema utilizando el siguiente comando.

1. >\_ dotnet --info

Text

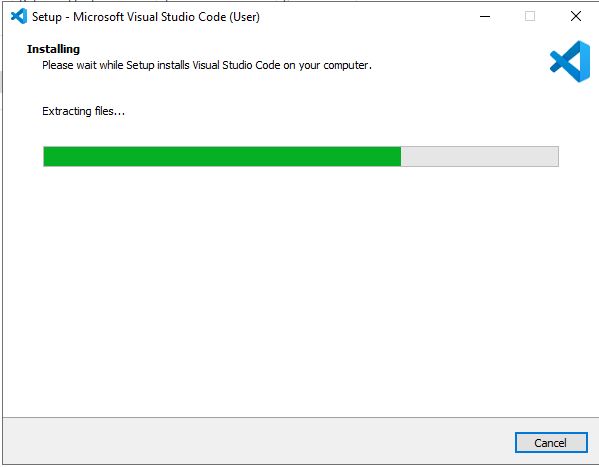
Description automatically generated

* **PASO 5 - Instalar código de Visual Studio desde** [aquí](https://code.visualstudio.com/download)

**Graphical user interface, website

Description automatically generated**

* **PASO 6** - Instalación completa basada en su sistema operativo.



Una vez completado, haga clic en el acabado en la pantalla de abajo.



* **PASO 7** - Instalar C# desde [aquí](https://marketplace.visualstudio.com/items?itemName=ms-dotnettools.csharp)

A picture containing graphical user interface

Description automatically generated

* **PASO 9** - Otra forma de instalar C# es buscar en Visual Studio Code -> Extensión e instalar.

A screenshot of a phone

Description automatically generated with medium confidence

Una vez instalado C#, nuestro entorno está listo para usar. Ahora podemos crear un nuevo proyecto dotnet usando Visual Studio Code + .net 6 y C# 10.

## Pasos para crear el motor de text analytics en Azure

Para crear un servicio de análisis de texto en Azure, es necesario lo siguiente

* Una cuenta de Azure
* Preferiblemente qué esta cuenta tenga créditos (funciones anticipadas o muchas operaciones por segundo)

Pasos para crear:

1. Vaya a la página de inicio en <https://portal.azure.com>, en la barra de búsqueda, escriba cognitive services y seleccione el resultado con este nombre.

Graphical user interface, text, application, email

Description automatically generated

1. Graphical user interface, application

   Description automatically generatedLuego diríjase a la sección de idioma, seleccione la opción crear en la tarjeta Language Service, donde debe seguir las instrucciones de configuración del servicio.
2. En la siguiente sección puede elegir características adicionales como preguntas y respuestas, clasificación y extracción de texto personalizada, una vez seleccionado pasamos a la siguiente sección haciendo clic en el botón continuar para crear su recurso.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

1. En el formulario de creación siga las instrucciones, seleccione la suscripción, el grupo de recursos (si no tiene uno, cree uno).
2. Seleccione la región (No todas las regiones tienen todos los servicios y el precio cambia para cada uno).
3. Ingrese el nombre del servicio y seleccione el plan de tarifas, si el plan gratuito está disponible (F0) selecciónelo (esto se aplica solo para el primer servicio), si tiene más de uno, se aplica el plan estándar, si seleccionó alguna de las características adicionales, debe seleccionar la región del servicio de búsqueda y seleccionar el plan gratuito F.
4. Hacer clic en revisar y crear y esperar a que se cree el recurso.

## Pasos para crear el proyecto en VS Code

Antes de comenzar con la creación de nuestra aplicación debemos crear la carpeta que almacenará nuestro código, luego de hacer esto seleccionamos nuestra nueva carpeta y damos clic derecho y seleccionamos la “opción abrir con code”, en donde se nos abrirá una nueva instancia de Visual Studio Code.

Paso siguiente para crear nuestro proyecto vamos a utilizar la consola de comandos integrada de Visual Studio Code la cual ubicamos en la parte inferior izquierda de la ventana de la aplicación.

Graphical user interface, text

Description automatically generated

en la pestaña terminal ingresamos el siguiente comando:

1. >\_ dotnet new web

Este comando creara los archivos mínimos que debe tener nuestra aplicación la cual ya es funcional, y podremos probar con el comando

1. >\_ dotnet run
2. Building...
3. info: Microsoft.Hosting.Lifetime[14]
4. Now listening on: **https://localhost:7265**
5. info: Microsoft.Hosting.Lifetime[14]
6. Now listening on: **http://localhost:5059**
7. info: Microsoft.Hosting.Lifetime[0]
8. Application started. Press Ctrl+C to shut down.
9. info: Microsoft.Hosting.Lifetime[0]
10. Hosting environment: Development
11. info: Microsoft.Hosting.Lifetime[0]
12. Content root path: C:\Users\david.rodriguezp\OneDrive - Perficient, Inc\Projects\DemoWorkshop\

Una vez ejecutado este comando nuestra aplicación ya estará en funcionamiento y que la aplicación esta trabajando con 2 URL, las cuales están resaltadas en el texto anterior, una para http y otra para https.

**Nota:** Cabe resaltar que estas URL son aleatorias por lo que serán diferentes en cada caso.

Ahora que comprobamos que nuestra aplicación funciona, vamos a utilizar en nuestro proyecto los siguientes paquetes que nos permitirán interactuar con los diferentes componentes de la aplicación:

* **Azure.AI.TextAnalytics v.5.2.0-beta.2**: Nos permitirá la interacción con nuestro servicio de análisis de texto.
* **Swashbuckle.AspNetCore v. 6.2.3**: Nos proveerá de una interfaz de prueba para nuestra aplicación.
* **TweetinviAPI v.5.0.4**: Nos permitirá la interacción las API de desarrollo de Twitter.

Para utilizarlas en nuestro proyecto debemos ejecutar los siguientes comandos en la terminal de VS Code para instalarlas.

1. >\_ dotnet add package Azure.AI.TextAnalytics -v 5.2.0-beta.2
2. >\_ dotnet add package Swashbuckle.AspNetCore -v 6.2.3
3. >\_ dotnet add package TweetinviAPI -v 5.0.4

Ahora en el archivo Program.cs de nuestro proyecto vamos a agregar las siguientes líneas para dar soporte de swagger que será el encargado de ayudarnos a probar los endpoint que vayamos construyendo, dejare el fragmento de codigo completo para reemplazar el contenido completo del archivo:

1. using Microsoft.OpenApi.Models;
3. var builder = WebApplication.CreateBuilder(args); //crea
4. builder.Services.AddEndpointsApiExplorer();
5. builder.Services.AddSwaggerGen(options =>
6. {
7. options.SwaggerDoc("v1", new OpenApiInfo { Title = "Demo TextAnalytics Api", Version = "v1" });
8. });
10. var app = builder.Build();
11. app.UseSwagger();
12. app.UseSwaggerUI(c => c.SwaggerEndpoint("/swagger/v1/swagger.json", "Api v1"));
13. app.MapGet("/", () => "Hello World!");
14. app.Run();

Si reemplazamos el contenido de nuestro archive Program.cs con el texto anterior ejecutamos la aplicación con el comando dotnet run en donde si visitamos en el navegador la URL <https://localhost:xxxx/swagger> veremos que ya esta implementada la funcionalidad para poder probar y documentar nuestros endpoint.

Graphical user interface, application

Description automatically generated