AZ-204

Developing solutions for Microsoft Azure

Lab 06

Authenticating to and querying Microsoft Graph by using MSAL and .NET SDKs

Table of Contents

1 Pre-requisites 3

1.1 Sign in to the lab virtual machine 3

1.2 Review the installed applications 3

3 Exercise 1: Create an Azure Active Directory (Azure AD) application registration 5

3.1 Task 1: Open the Azure portal 5

3.2 Task 2: Create an application registration 5

3.3 Task 3: Enable the default client type 6

3.4 Task 4: Record unique identifiers 7

3.5 Review 7

3.5.1 Exercise 2: Obtain a token by using the MSAL.NET library 8

3.6 Task 1: Create a .NET project 8

3.7 Task 2: Modify the Program class 8

3.8 Task 3: Obtain a Microsoft Authentication Library (MSAL) token 10

3.9 Task 4: Test the updated application 11

3.10 Review 12

3.10.1 Exercise 3: Query Microsoft Graph by using the .NET SDK 13

3.11 Task 1: Import the Microsoft Graph SDK from NuGet 13

3.12 Task 2: Modify the Program class 13

3.13 Task 3: Use the Microsoft Graph SDK to query user profile information 13

3.14 Task 4: Test the updated application 15

3.15 Review 16

3.15.1 Exercise 4: Clean up your subscription 17

3.16 Task 1: Delete the application registration in Azure AD 17

3.17 Task 2: Close the active applications 17

3.18 Review 17

1 Pre-requisites

1.1 Sign in to the lab virtual machine

Sign in to your Windows 10 virtual machine (VM) by using the following credentials:

- Username: **Admin**
- Password: **Pa55w.rd**

Note: Instructions to connect to the virtual lab environment will be provided by your instructor.

1.2 Review the installed applications

Find the taskbar on your Windows 10 desktop. The taskbar contains the icons for the applications that you'll use in this lab:

- Microsoft Edge
- File Explorer
- Azure CLI
- Windows PowerShell

3 Exercise 1: Create an Azure Active Directory (Azure AD) application registration

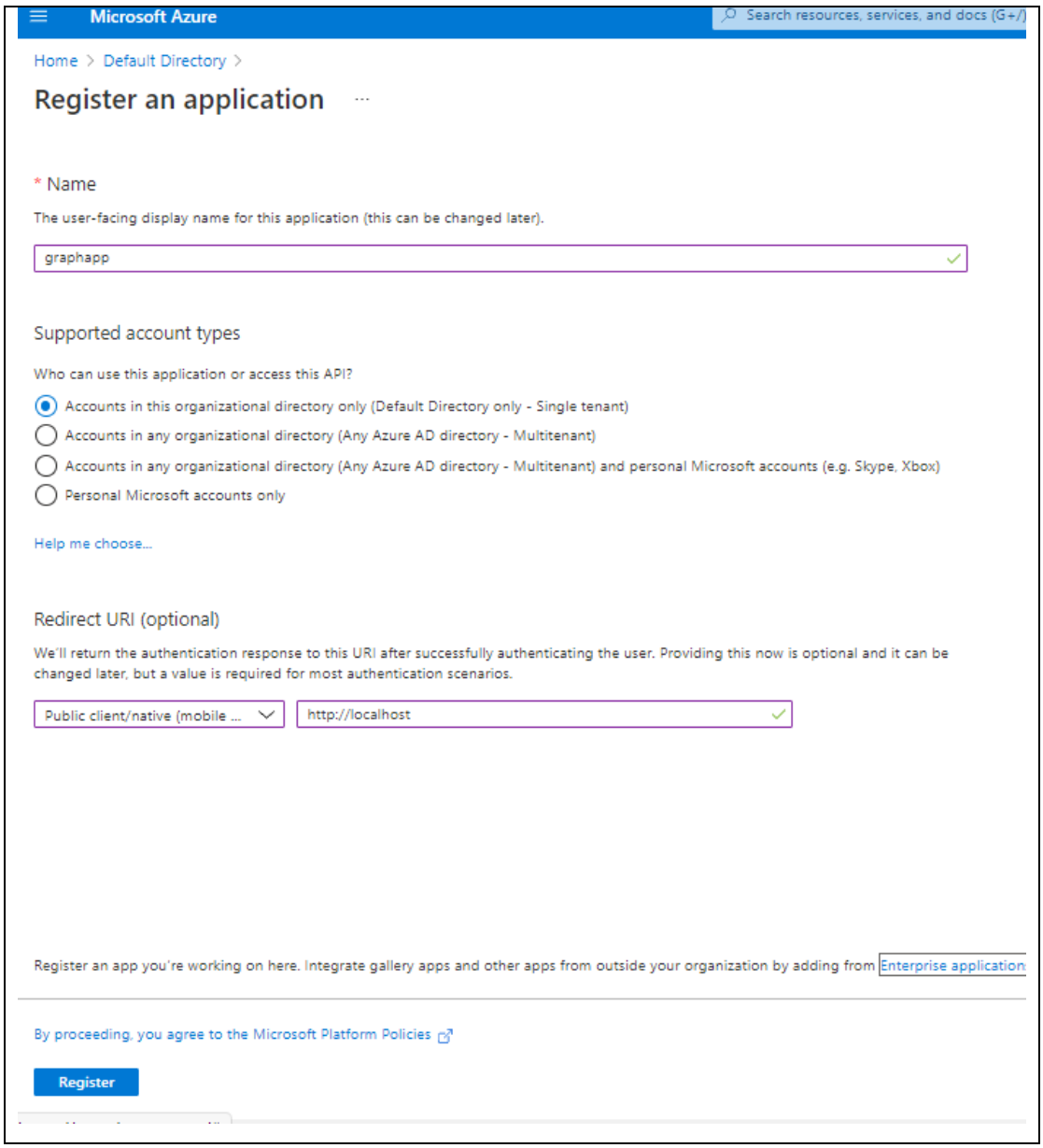
3.1 Task 1: Open the Azure portal

1. On the taskbar, select the **Microsoft Edge** icon.
2. In the open browser window, go to the Azure portal (https://portal.azure.com).
3. Enter the email address for your Microsoft account, and then select **Next**.
4. Enter the password for your Microsoft account, and then select **Sign in**.

   Note: If this is your first time signing in to the Azure portal, you'll be offered a tour of the portal. Select **Get Started** to skip the tour and begin using the portal.

3.2 Task 2: Create an application registration

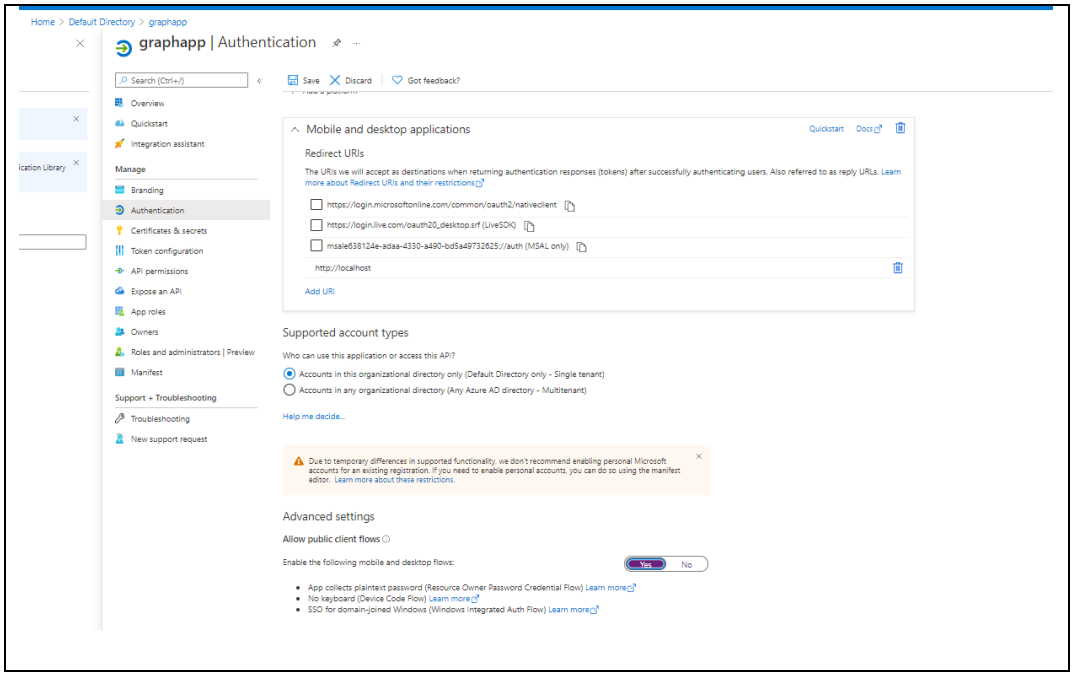
1. In the Azure portal's navigation pane, select **All services**.
2. From the **All services** blade, select **Azure Active Directory**.
3. From the **Azure Active Directory** blade, select **App registrations** in the **Manage** section.
4. In the **App registrations** section, select **New registration**.
5. In the **Register an application** section, perform the following actions:
    1. In the **Name** text box, enter **graphapp**.
    2. In the **Supported account types** list, select the **Accounts in this organizational directory only (Default Directory only - Single tenant)** check box.
    3. In the **Redirect URI** drop-down list, select **Public client/native (mobile & desktop)**.
    4. In the **Redirect URI** text box, enter http://localhost.



5. Select **Register**.

3.3 Task 3: Enable the default client type

1. In the **graphapp** application registration blade, select **Authentication** in the **Manage** section.
2. In the **Authentication** section, perform the following actions:
    1. In the **Advanced settings** - **Allow public client flows** subsection, select **Yes**.



2. Select **Save**.

3.4 Task 4: Record unique identifiers

1. On the **graphapp** application registration blade, select **Overview**.
2. In the **Overview** section, find and record the value of the **Application (client) ID** text box. You'll use this value later in the lab.
3. In the **Overview** section, find and record the value of the **Directory (tenant) ID** text box. You'll use this value later in the lab.

3.5 Review

In this exercise, you created a new application registration and recorded important values that you'll need later in the lab.

3.5.1 Exercise 2: Obtain a token by using the MSAL.NET library

3.6 Task 1: Create a .NET project

1. On the **Start** screen, select the **Visual Studio Code** tile.
2. On the **File** menu, select **Open Folder**.
3. In the **File Explorer** window that opens, browse to **Allfiles (F):\Allfiles\Labs\06\Starter\GraphClient**, and then select **Select Folder**.
4. In the **Visual Studio Code** window, right-click or activate the shortcut menu for the Explorer pane, and then select **Open in Terminal**.
5. At the open command prompt, enter the following command, and then select Enter to create a new .NET project named **GraphClient** in the current folder:

```
dotnet new console --name GraphClient --output .
```

Note: The **dotnet new** command will create a new **console** project in a folder with the same name as the project.

6. At the command prompt, enter the following command, and then select Enter to import version 4.7.1 of **Microsoft.Identity.Client** from NuGet:

```
dotnet add package Microsoft.Identity.Client --version 4.7.1
```

Note: The **dotnet add package** command will add the **Microsoft.Identity.Client** package from NuGet. For more information, go to Microsoft.Identity.Client.

7. At the command prompt, enter the following command, and then select Enter to build the .NET web application:

```
dotnet build
```

7. Select **Kill Terminal** or the **Recycle Bin** icon to close the currently open terminal and any associated processes.

3.7 Task 2: Modify the Program class

1. In the Explorer pane of the **Visual Studio Code** window, open the **Program.cs** file.
2. On the code editor tab for the **Program.cs** file, delete all the code in the existing file.
3. Add the following line of code to import the **Microsoft.Identity.Client** namespace from the **Microsoft.Identity.Client** package imported from NuGet:

```
using Microsoft.Identity.Client;
```

4. Add the following lines of code to add **using** directives for the built-in namespaces that will be used in this file:

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
```

5. Enter the following code to create a new Program class: and create a new asynchronous **Main** method:

```
public class Program
{

    public static async Task Main(string[] args)
    {

    }

}
```

6. In the **Program** class, enter the following line of code to create a new string constant named **_clientId**. Update the **_clientId** string constant by setting its value to the **Application (client) ID** that you recorded earlier in this lab.

```
private const string _clientId = "……..gdfdf….";
```

7. In the **Program** class, enter the following line of code to create a new string constant named **_tenantId**: Update the **_tenantId** string constant by setting its value to the **Directory (tenant) ID** that you recorded earlier in this lab.

```
    private const string _tenantId = "....mmmmmm.......";
```

Observe the **Program.cs** file, which should now include:

```
using Microsoft.Identity.Client;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
public class Program
{

    private const string _clientId = "<app-reg-client-id>";
    private const string _tenantId = "<aad-tenant-id>";

    public static async Task Main(string[] args)
    {
```

```
    }

}
```

3.8 Task 3: Obtain a Microsoft Authentication Library (MSAL) token

1. In the **Main** method, add the following line of code to create a new variable named *app* of type **IPublicClientApplication**:

```
IPublicClientApplication app;
```

2. In the **Main** method, perform the following actions to build a public client application instance by using the static PublicClientApplicationBuilder class, and then store it in the *app* variable:

```
IPublicClientApplication app;
                app = PublicClientApplicationBuilder
                    .Create(_clientId)
                    .WithAuthority(AzureCloudInstance.AzurePublic, _tenantId)
                    .WithRedirectUri("http://localhost")
                    .Build();
3.
```

4. In the **Main** method, add the following line of code to create a new generic string List<> with a single value of **user.read**:

```
List<string> scopes = new List<string>
            {
                "user.read"
            };
```

5. In the **Main** method, add the following line of code to create a new variable named *result* of type AuthenticationResult and add code to use the **AcquireTokenInteractive()** method of the **IPublicClientApplicationBuilder** interface, passing in the *scopes* variable as a parameter and return the result asynchronously

```
AuthenticationResult result;

            result = await app
                        .AcquireTokenInteractive(scopes)
                        .ExecuteAsync();
```

6. In the **Main** method, add the following line of code to use the **Console.WriteLine** method to render the value of the AuthenticationResult.AccessToken member to the console:

```
Console.WriteLine($"Token:\t{result.AccessToken}");
```

7. Observe the **Main** method, which should now include:

```
public static async Task Main(string[] args)
   {
      IPublicClientApplication app;
               app = PublicClientApplicationBuilder
                   .Create(_clientId)
                   .WithAuthority(AzureCloudInstance.AzurePublic, _tenantId)
                   .WithRedirectUri("http://localhost")
                   .Build();
          List<string> scopes = new List<string>
            {
                "user.read"
            };
            AuthenticationResult result;
            result = await app
                        .AcquireTokenInteractive(scopes)
                        .ExecuteAsync();

            Console.WriteLine($"Token:\t{result.AccessToken}");

   }
```

7. Save the **Program.cs** file.

3.9 Task 4: Test the updated application

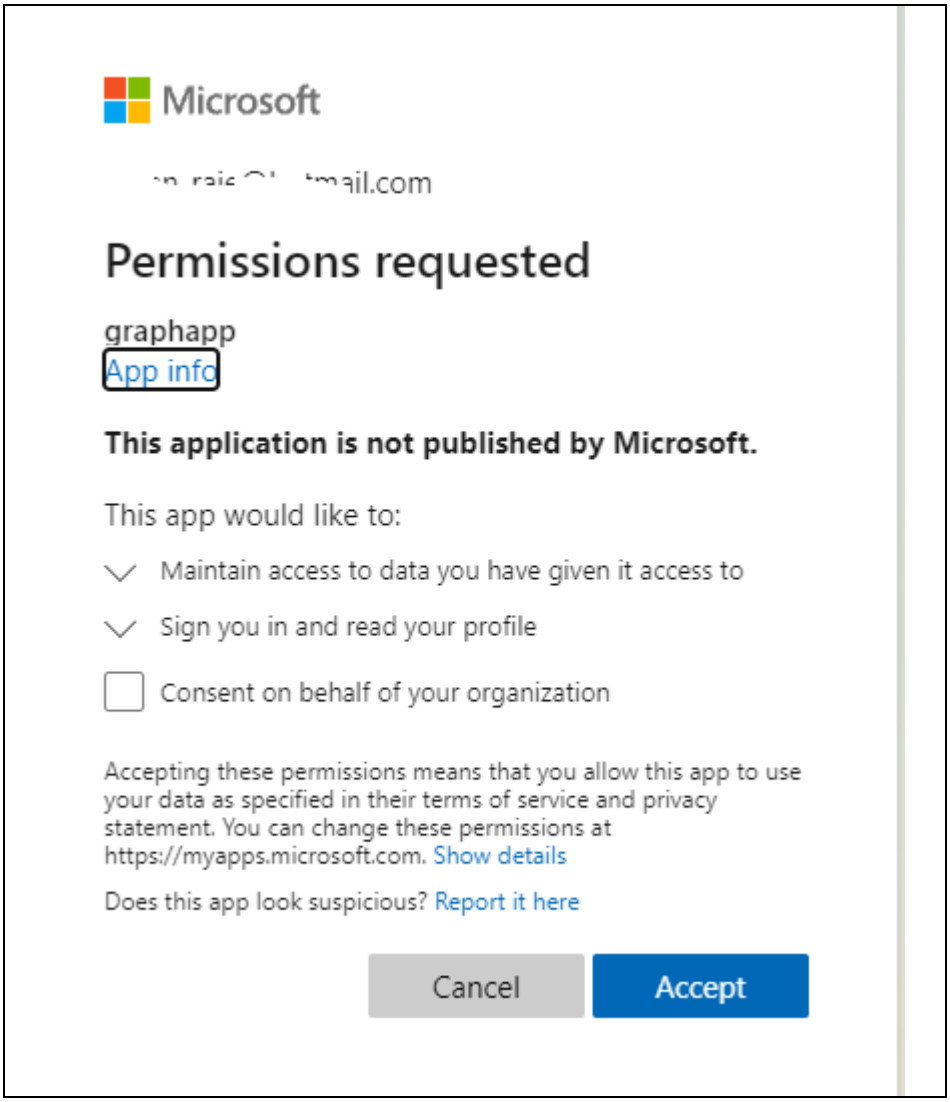
1. In the **Visual Studio Code** window, right-click or activate the shortcut menu for the Explorer pane, and then select **Open in Terminal**.
2. At the open command prompt, enter the following command, and then select Enter to run the .NET web application:

   dotnet run

2. **Note**: If there are any build errors, review the **Program.cs** file in the **Allfiles (F):\Allfiles\Labs\06\Solution\GraphClient** folder.
3. The running console application will automatically open an instance of the default browser.
4. In the open browser window, perform the following actions:
   1. Enter the email address for your Microsoft account, and then select **Next**.
   2. Enter the password for your Microsoft account, and then select **Sign in**.

   Note: You might have the option to select an existing Microsoft account as opposed to signing in again.

5. The browser window will automatically open the **Permissions requested** webpage. On this webpage, perform the following actions:
   1. Review the requested permissions.



2. Select **Accept**.

6. Return to the currently running Visual Studio Code application.
7. Observe the token rendered in the output from the currently running console application.
8. Select **Kill Terminal** or the **Recycle Bin** icon to close the currently open terminal and any associated processes.

3.10 Review

In this exercise, you acquired a token from the Microsoft identity platform by using the MSAL.NET library.

3.10.1 Exercise 3: Query Microsoft Graph by using the .NET SDK

3.11 Task 1: Import the Microsoft Graph SDK from NuGet

1. In the **Visual Studio Code** window, right-click or activate the shortcut menu for the Explorer pane, and then select **Open in Terminal**.
2. At the command prompt, enter the following command, and then select Enter to import version 1.21.0 of **Microsoft.Graph** from NuGet:

```
dotnet add package Microsoft.Graph --version 1.21.0
```

Note: The **dotnet add package** command will add the **Microsoft.Graph** package from NuGet. For more information, go to Microsoft.Graph.

3. At the command prompt, enter the following command, and then select Enter to import version 1.0.0-preview.2 of **Microsoft.Graph.Auth** from NuGet:

```
dotnet add package Microsoft.Graph.Auth --version 1.0.0-preview.2
```

Note: The **dotnet add package** command will add the **Microsoft.Graph.Auth** package from NuGet. For more information, go to Microsoft.Graph.Auth.

4. At the command prompt, enter the following command, and then select Enter to build the .NET web application:

```
dotnet build
```

5. Select **Kill Terminal** or the **Recycle Bin** icon to close the currently open terminal and any associated processes.

3.12 Task 2: Modify the Program class

1. In the Explorer pane of the **Visual Studio Code** window, open the **Program.cs** file.
2. On the code editor tab for the **Program.cs** file, add the following line of code to import the **Microsoft.Graph** and **Microsoft.Graph**.**Auth** namespace from the **Microsoft.Graph** package imported from NuGet and ad

```
using Microsoft.Graph;

using Microsoft.Graph.Auth;
```

3.13 Task 3: Use the Microsoft Graph SDK to query user profile information

1. Within the **Main** method, delete the following block of code:

```
AuthenticationResult result;
            result = await app
                        .AcquireTokenInteractive(scopes)
```

```
                    .ExecuteAsync();

            Console.WriteLine($"Token:\t{result.AccessToken}");
```

2. Observe the **Main** method, which should now include:

```
public static async Task Main(string[] args)
   {
      IPublicClientApplication app;
               app = PublicClientApplicationBuilder
                   .Create(_clientId)
                   .WithAuthority(AzureCloudInstance.AzurePublic, _tenantId)
                   .WithRedirectUri("http://localhost")
                   .Build();
         List<string> scopes = new List<string>
           {
               "user.read"
           };

   }
```

3. In the **Main** method, add the following line of code to create a new variable named *provider* of type **DeviceCodeProvider** that passes in the variables *app* and *scopes* as constructor parameters:

```
DeviceCodeProvider provider = new DeviceCodeProvider(app, scopes);
```

4. In the **Main** method, add the following line of code to create a new variable named *client* of type **GraphServiceClient** that passes in the variable *provider* as a constructor parameter:

```
GraphServiceClient client = new GraphServiceClient(provider);
```

5. In the **Main** method, add code to perform the following actions to use the **GraphServiceClient** instance to asynchronously get the response of issuing an HTTP request to the relative **Me** directory of the REST API:

```
User myProfile = await client.Me
                .Request()
                .GetAsync();
```

6. In the **Main** method, add the following line of code to use the **Console.WriteLine** method to render the value of the **User.DisplayName** member to the console:

```
Console.WriteLine($"Name:\t{myProfile.DisplayName}");
```

7. In the **Main** method, add the following line of code to use the **Console.WriteLine** method to render the value of the **User.Id** member to the console:

```
Console.WriteLine($"AAD Id:\t{myProfile.Id}");
```

8. Observe the **Main** method, which should now include:

```
public static async Task Main(string[] args)
{
   IPublicClientApplication app;
            app = PublicClientApplicationBuilder
                .Create(_clientId)
                .WithAuthority(AzureCloudInstance.AzurePublic, _tenantId)
                .WithRedirectUri("http://localhost")
                .Build();
       List<string> scopes = new List<string>
         {
             "user.read"
         };

    DeviceCodeProvider provider = new DeviceCodeProvider(app, scopes);
    GraphServiceClient client = new GraphServiceClient(provider);
   User myProfile = await client.Me
            .Request()
            .GetAsync();
    Console.WriteLine($"Name:\t{myProfile.DisplayName}");
    Console.WriteLine($"AAD Id:\t{myProfile.Id}");

}
```

8. Save the **Program.cs** file.

3.14 Task 4: Test the updated application

1. In the **Visual Studio Code** window, right-click or activate the shortcut menu for the Explorer pane, and then select **Open in Terminal**.
2. At the open command prompt, enter the following command, and then select Enter to run the .NET web application:

   dotnet run

2. **Note**: If there are any build errors, review the **Program.cs** file in the **Allfiles (F):\Allfiles\Labs\06\Solution\GraphClient** folder.
3. Observe the message in the output from the currently running console application. Record the value of the code in the message. You'll use this value later in the lab.

4. On the taskbar, select the **Microsoft Edge** icon.
5. In the open browser window, go to https://microsoft.com/devicelogin.
6. On the **Enter code** webpage, perform the following actions:
    1. In the **Code** text box, enter the value of the code that you copied earlier in the lab.
    2. Select **Next**.
7. On the login webpage, perform the following actions:
    1. Enter the email address for your Microsoft account, and then select **Next**.
    2. Enter the password for your Microsoft account, and then select **Sign in**.

    Note: You might have the option to select an existing Microsoft account as opposed to signing in again.

8. Return to the currently running Visual Studio Code application.
9. Observe the output from the Microsoft Graph request in the currently running console application.
10. Select **Kill Terminal** or the **Recycle Bin** icon to close the currently open terminal and any associated processes.

3.15 Review

In this exercise, you queried Microsoft Graph by using the SDK and MSAL-based authentication.

3.15.1 Exercise 4: Clean up your subscription

3.16 Task 1: Delete the application registration in Azure AD

1. Return to the browser window with the Azure portal.
2. In the Azure portal's navigation pane, select **All services**.
3. From the **All services** blade, select **Azure Active Directory**.
4. From the **Azure Active Directory** blade, select **App registrations** in the **Manage** section.
5. In the **App registrations** section, select the **graphapp** Azure AD application registration that you created earlier in this lab.
6. In the **graphapp** section, perform the following actions:
    1. Select **Delete**.
    2. In the **Delete app registration** blade, select **I understand the implications of deleting this app registration**, and then select **Delete**.

3.17 Task 2: Close the active applications

1. Close the currently running Microsoft Edge application.
2. Close the currently running Visual Studio Code application.

3.18 Review

In this exercise, you cleaned up your subscription by removing the application registration used in this lab.