**WebAPI application using C# and ASP.NET that securely calls the Microsoft Graph API to retrieve all AAD users of a given O365 tenant.**

To achieve this requirement:

* Azure AD app registration with API permissions for Microsoft graph.
* Create a ASP.NET core minimal API project
* Install below packages (nuget)

Microsoft.Identity.Client

Microsoft.Graph

Swashbuckle.AspNetCore

* Add configuration for Azure AD in appsettings.json

"AzureAd": {

"Instance": "https://login.microsoftonline.com/",

"TenantId": "TENANT\_ID",

"ClientId": "CLIENT\_ID",

"ClientSecret": "CLIENT\_SECRET",

"Scopes": "User.Read.All",

"Instance": "https://login.microsoftonline.com/",

}

* Configure Authentication and Authorization in Program.cs
* Configure Swagger for API documentation
* Call Microsoft Graph API in controller in an HttpGet call .

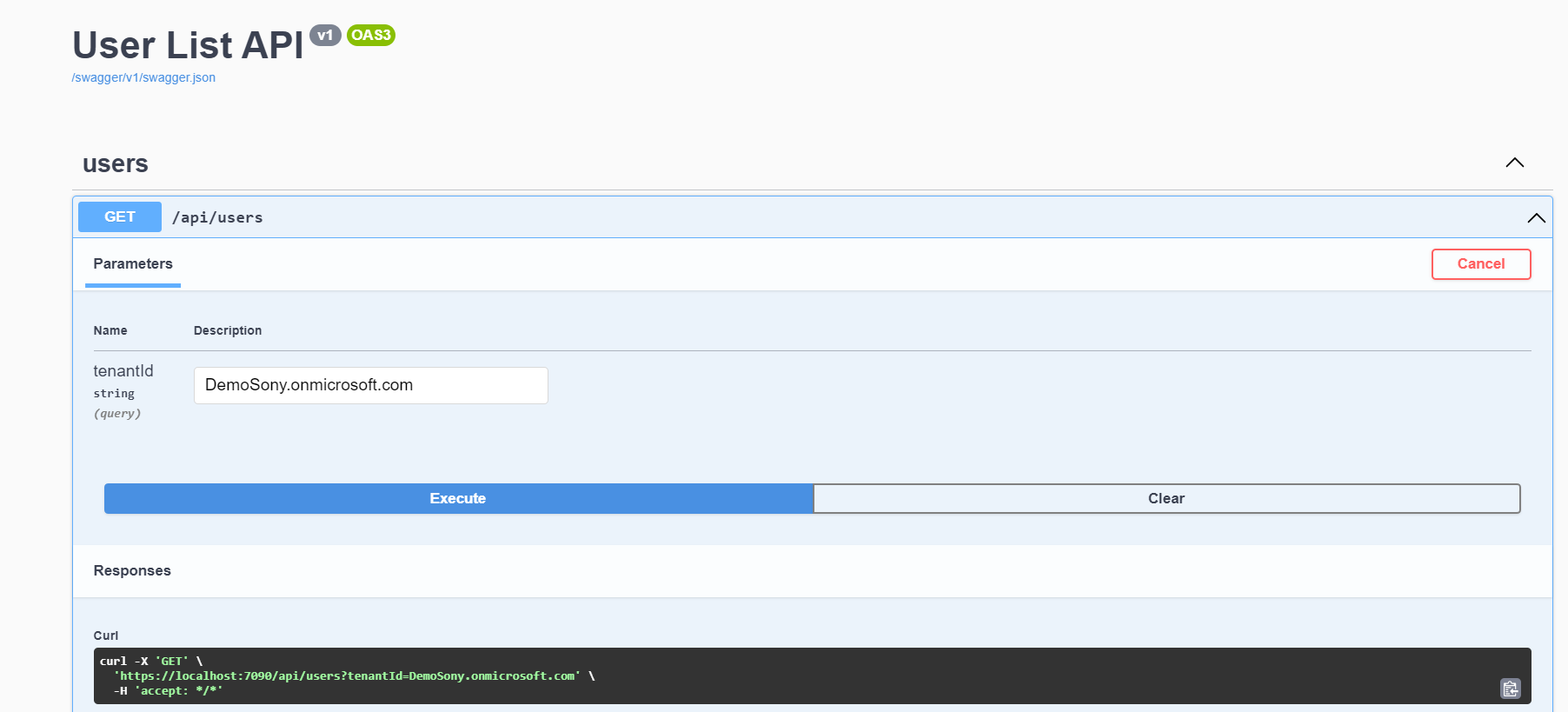
To achive this in this application , created a class called GraphServiceClientFactory, there the method GetGraphServiceClientAsync accept the tenantID as parameter , and an authenticated instance of the GraphServiceClient for making requests to the Microsoft graph API.

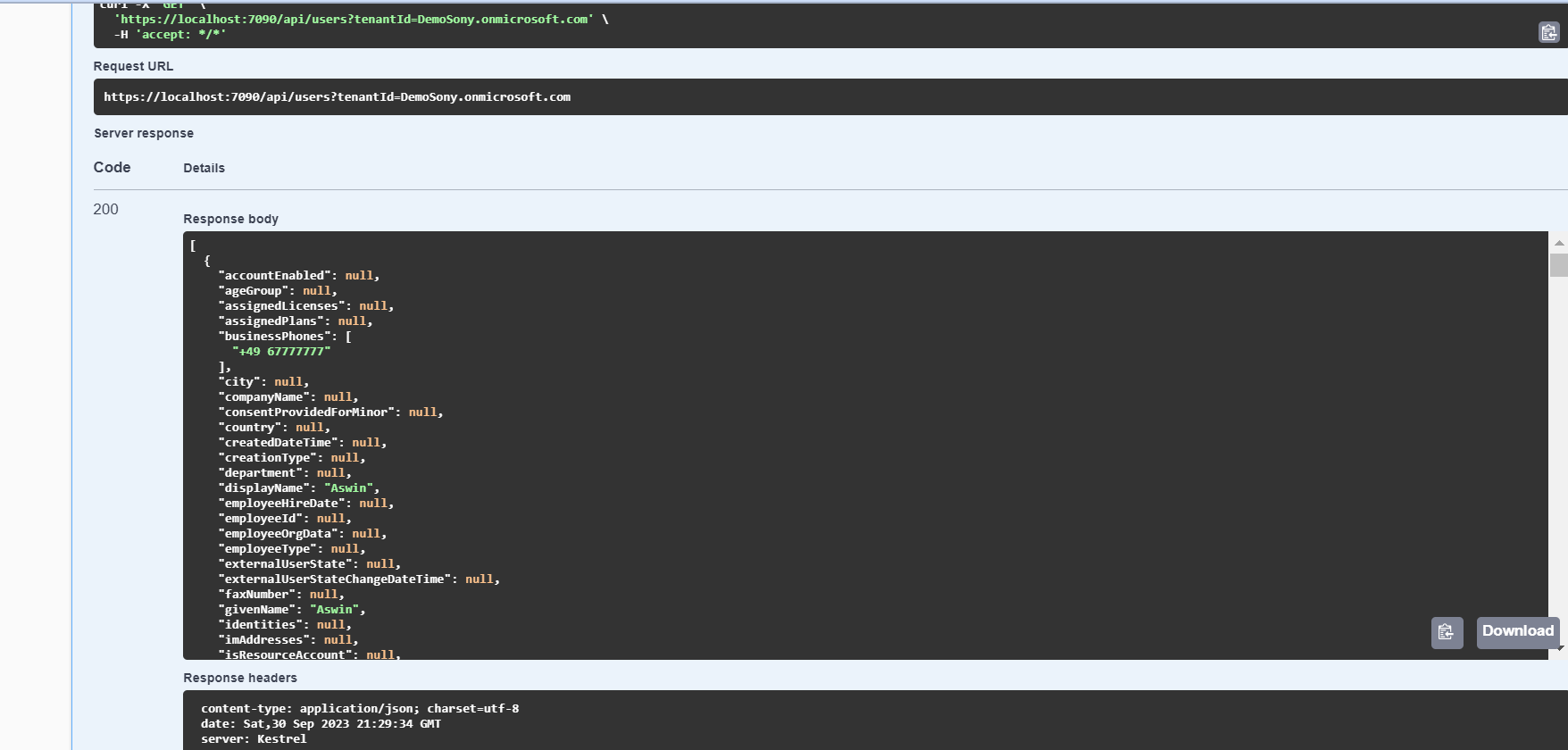
Use the IConfiguration object to retrieve the Azure AD configuration values, such as ClientId, TenantId, ClientSecret, and Instance.

When calling GetGraphServiceClientAsync, it acquire an access token for the specified tenant.

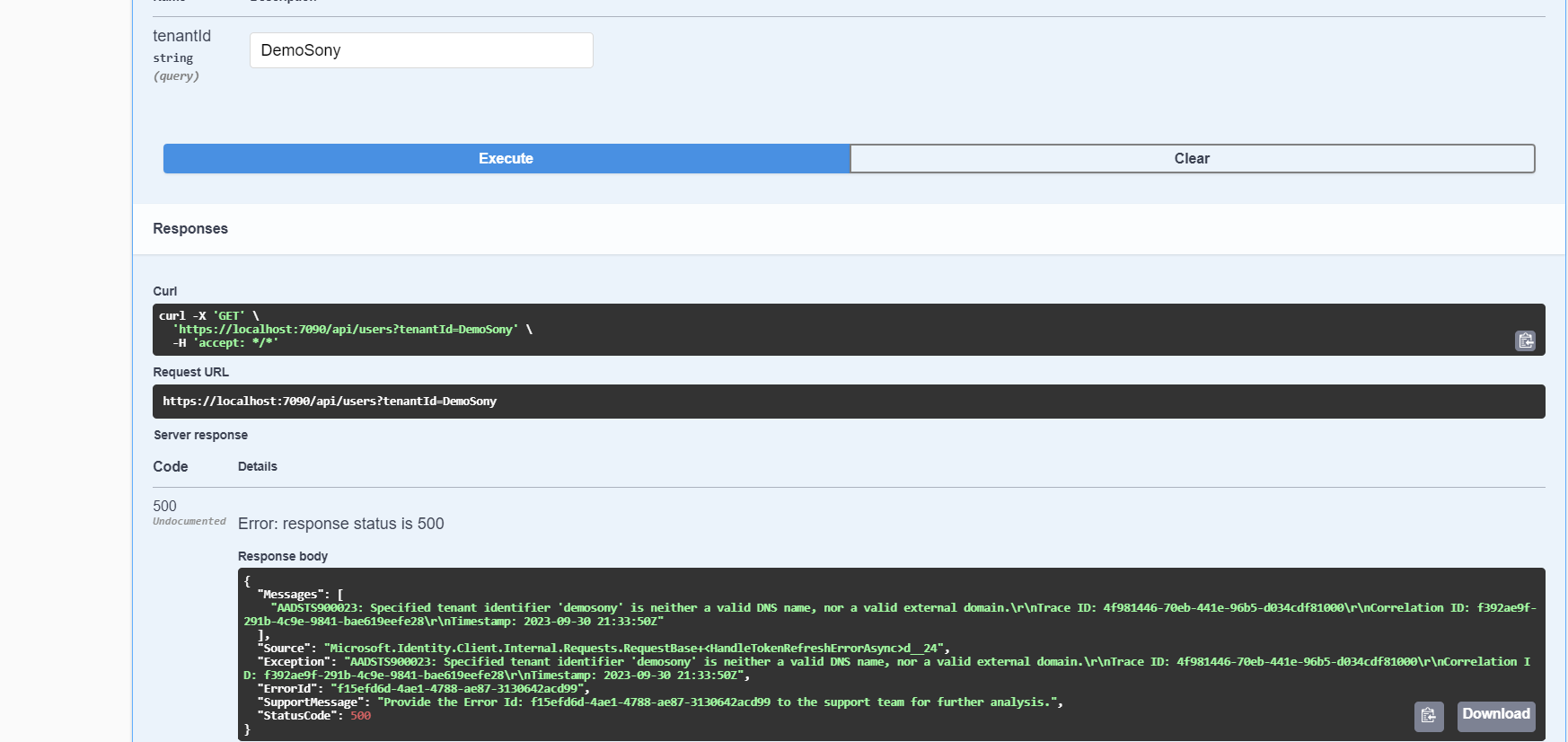
GetUsers method in the controller make use of GraphServiceClientFactory to get the information of all users belong to a particulat tenantID.

**Response from the API request:**





*For an invalid tenant id:*



**Global Exception Handling:**

To handle exception globally, created a folder called *Exceptions* at the root of the project, add the below exception classes.

* CustomException
* NotFoundException
* InternalServerException
* ConflictException

Custom exception is the base calss inherit from System.Exception, each exception inherit from CustomException . Populate the error with list of error messages and status code.

ErrorResult class inside Models folder , hold the error result.

To work the global exception, created a ExceptionMiddleware class inside a Middleware folder, and this class inherit from Imiddleware, and implement InvokeAsync() method to run the middleware on the request delegate.

For serialize the response, installed the package Newtonsoft.Json and installed another package named Serilog to get the logs from the middleware.

To register the global exception-handling middleware in the project, created Startup.cs inside Middleware. This class is static and is an extension method for ApplicationBuilder. This allows to inject the middleware in Program.cs.

Challenges:

To build this application need to have better analysis on how all the required configuration works, and how to incorporate those configurations with respect to authentication and authorization for accessing Microsoft Graph data and making the application confidential. Setting up the consent, get the token, validate token are crucial in this scenario. Any wrong configuration or permission-set leads to failure in to getting expected result. This takes good amount of time to understand and achieve proper result, requires n number of testing.