**TABLE OF CONTENT**

Contents

[**Introduction** 3](#_Toc69811558)

[**Application Architecture** 3](#_Toc69811559)

[**Development Environment** 3](#_Toc69811560)

[**Technologies** 3](#_Toc69811561)

[**Open source Tools Used** 3](#_Toc69811562)

[**WebApi Endpoints** 4](#_Toc69811563)

[**End-points configured and accessible through API Gateway** 4](#_Toc69811564)

[**End-points implemented at the Microservice level** 4](#_Toc69811565)

[**Solution Structure** 4](#_Toc69811566)

[**Postman Collection** 5](#_Toc69811567)

[**How to run the application** 6](#_Toc69811568)

[**Login data to test** 6](#_Toc69811569)

[**Policy Details** 6](#_Toc69811570)

[**Console App - Gateway Client** 7](#_Toc69811571)

[7](#_Toc69811572)

**Introduction**

This is a .Net Core sample application and an example of how to build and implement a microservices based back-end system for a simple showing Product & Customer details in ASP.NET Core Web API with C#.Net.

**Application Architecture**

The sample application is build based on the microservices architecture. There are several advantages in building an application using Microservices architecture like Services can be developed, deployed and scaled independently. The below diagram shows the high level design of Back-end architecture.

* **Identity Microservice** - Authenticates user based on username, password and issues a JWT Bearer token which contains Claims-based identity information in it.
* **Product Microservice** – Perform Product related data operations.
* **Customer Microservice** - Perform Customer related data operations.
* **API Gateway** - Acts as a center point of entry to the back-end application, Provides data aggregation and communication path to microservices.

**Development Environment**

* [.Net Core 2.2 SDK](https://dotnet.microsoft.com/download)
* [Visual Studio .Net 2017](https://visualstudio.microsoft.com/downloads/)

**Technologies**

* C#.NET
* ASP.NET WEB API Core

**Open source Tools Used**

* Ocelot (For API Gateway Aggregation)
* Swashbucke (For API Documentation)

**WebApi Endpoints**

The application has four API endpoints configured in the API Gateway to demonstrate the features with token based security options enabled. These routes are exposed to the client app to consume the back-end services.

**End-points configured and accessible through API Gateway**

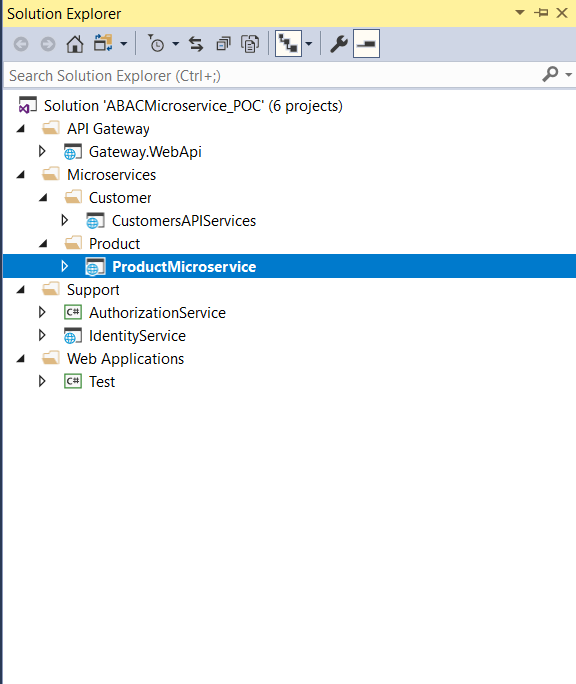
1. Route: **"/user/authenticate"** [HttpPost] - To authenticate user and issue a token
2. Route: **"/Products"**[HttpGet] - To retrieve products information.
3. Route: **"/Customers"** [HttpGet] - To retrieve customers information.

**End-points implemented at the Microservice level**

1. Route: **"/api/user/authenticate"** [HttpPost]- To authenticate user and issue a token
2. Route: **"/api/Products"** [HttpGet] - To retrieve products information.
3. Route: **"/api/Customers"** [HttpGet] - To retrieve customers information.

**Solution Structure**

* **Identity.WebApi**
  + Handles the authentication part using username, password as input parameter and issues a JWT Bearer token with Claims-Identity info in it.
* **ProductService.WebApi**
  + Supports http Get methods. Receives http request for these methods.
  + Reads Identity information from the Authorization Header which contains the Bearer token
  + Returns the response result back to the client
* **CustomerService.WebApi**
  + Supports http Get methods. Receives http request for these methods.
  + Reads Identity information from the Authorization Header which contains the Bearer token
  + Returns the response result back to the client
* **AuthorizationService.ClassLibrary**
  + Class library is used to contain all policy definition & verification logic
* **Gateway.WebApi**
  + Validates the incoming Http request by checking for authorized JWT token in it.
  + Reroute the Http request to a downstream service.
* **Test.ConsoleApp**
  + A console client app that connects to API Gateway, can be used to login with username, password and retrieve products & customers information.



**Postman Collection**

Download the postman collection from [here](https://1drv.ms/u/s!AshsJMsiN-avgQCgr5NIFpZCoM5r) to run the api endpoints through gateway

**How to run the application**

1. Open the solution (.sln) in Visual Studio 2017 or later version
2. Check the Identity.WebApi -> UserService.cs file for Identity info. User details are hard coded for few accounts in Identity service which can be used to run the app. same details shown in the below table.
3. Run “**ExecuteAllProject.bat**” file for executing all projects solutions.

**Login data to test**

| **Username** | **Password** |
| --- | --- |
| swapnil | test@123 |
| chetan | test@123 |
| rehan | test@123 |
| parul | test@123 |

**Policy Details**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Policy Name** | **Policy Description** |
| **1** | AgeUnder25 | Policy satisfied when user age is “**greater then equal to 25 years**” |
| **2** | AgeAbove25RoleAdmin | Policy satisfied when user age is “**greater than equal to 25 years**” and User Role is “**Admin**” |
| **3** | DesignationSrLead | Policy satisfied when user designation is **“SrLead”** |
| **4** | AgeAbove27Consultant | Policy satisfied when user age is “**greater than 27 years**” and User designation is “**Consultant**” |

**Console App - Gateway Client**

