GatewayService

GatewayService is an API gateway implemented using Ocelot in ASP.NET Core. It acts as a reverse proxy to route incoming requests to various downstream microservices in the application, simplifying and centralizing API management.

Features

- Reverse proxy routing using Ocelot.
- Centralized gateway for managing multiple services.
- Configurable upstream and downstream routes.
- Supports multiple HTTP methods and routes.
- Secure communication using HTTPS.

Prerequisites

- .NET 8 SDK
- Ocelot NuGet Package
- **IdentityService** or other microservices running on different ports.

Getting Started

1. Clone the Repository

```
cd backend/GatewayService
```

2. Install Dependencies

Make sure you have installed all the required NuGet packages for Ocelot:

```
dotnet restore
```

3. Configure Ocelot Routes

The `ocelot.json` file contains the routing configuration for the gateway. Here's an example configuration for routing login requests to the **IdentityService** :

```
{
  "Routes": [
    {
      "DownstreamPathTemplate": "/api/Auth/login/{everything}",
      "DownstreamScheme": "https",
      "DownstreamHostAndPorts": [
        {
          "Host": "localhost",
          "Port": 7069
        }
      ],
      "UpstreamPathTemplate": "/api/login/{everything}",
      "UpstreamHttpMethod": [ "POST" ]
    }
  ],
  "GlobalConfiguration": {
    "BaseUrl": "https://localhost:5000"
  }
}
```

- **Upstream Path** : `/api/login/{everything}` is exposed to clients and accepts `POST` requests.
- **Downstream Path** : Requests are forwarded to `https://localhost:7069/api/Auth/login/{everything}`.
- **Port 7069** : This is the port where **IdentityService** is running.

4. Configure Kestrel for Gateway

Ensure that the Gateway listens on the correct port by modifying the `appsettings.json`:

```
{
  "Kestrel": {
    "Endpoints": {
      "Http": {
        "Url": "https://localhost:5000"
      }
    }
  }
}
```

5. Run the Gateway

To run the gateway, use the following command:

```
dotnet run
```

The **GatewayService** will now be running at `https://localhost:5000`.

6. Test the Gateway

You can test the Gateway by sending a request to the upstream URL. For example, to test login requests:

- **Request** :

```
POST https://localhost:5000/api/login
Content-Type: application/json

{
  "username": "mehran",
  "password": "password"
}
```

- This request will be routed to **IdentityService** at `https://localhost:7069/api/Auth/login`.

API Gateway Routes

`/api/login/{everything}` (POST)

- **Description** : This route handles login requests by forwarding them to the downstream **IdentityService** .
- **Downstream Service** : IdentityService running on `https://localhost:7069`.

Adding More Routes

To add more routes, update the `ocelot.json` file with the upstream and downstream configurations for each microservice.

Technology Stack

- **.NET 8**
- **Ocelot** - API Gateway for managing microservice routing.
- **ASP.NET Core** for building the Gateway API.
- **IdentityService** (or other microservices) as the downstream services.