Fontys University of Applied Sciences

API Gateway Configuration

GetawayGo

Anna Kadurina
08/12/2024

Version	Date	Author(s)	Changes	State
1.0	03/11/2024	Anna Kadurina	Creation of the document and filling out initial information	First draft
1.1	04/11/2024	Anna Kadurina	Added Production testing	In progress
2.0	08/12/2024	Anna Kadurina	Added final configuration of API with all deployed service	Final version

Table of Contents

Ocelot 1
Local configuration 1
Production configuration 3

Ocelot

In modern microservices architecture, managing communication between multiple services can be challenging. Ocelot serves as a lightweight API Gateway that simplifies this process by providing a single entry point for clients to access various microservices. By implementing Ocelot, I can centralize routing, load balancing, and security features, thereby enhancing performance and scalability.

Local configuration

I have created 2 ocelot configuration files, one for Development (for local usage) and one for Production. For the local ocelot file, I have configured the urls with localhost and the port that the microservice is running on.

launchSettings.json | Program.cs | ocelot.Development.json

Schema: <No Schema Selected>

```
{
  "Routes": [
    {
      "DownstreamPathTemplate": "/api/user/{everything}",
      "DownstreamScheme": "http",
      "DownstreamHostAndPorts": [
        {
          "Host": "localhost",
          "Port": 5094
        }
      ],
      "UpstreamPathTemplate": "/user/{everything}",
      "UpstreamHttpMethod": [ "Get", "Post", "Put", "Delete" ]
    },
    {
      "DownstreamPathTemplate": "/api/property/{everything}",
      "DownstreamScheme": "http",
      "DownstreamHostAndPorts": [
        {
          "Host": "localhost",
          "Port": 5002
        }
      ],
      "UpstreamPathTemplate": "/property/{everything}",
      "UpstreamHttpMethod": [ "Get", "Post", "Put", "Delete" ]
    }
  ],
  "GlobalConfiguration": {
    "BaseUrl": "http://localhost:5000"
  }
}
```

Figure 1 – Local configuration of the API Gateway

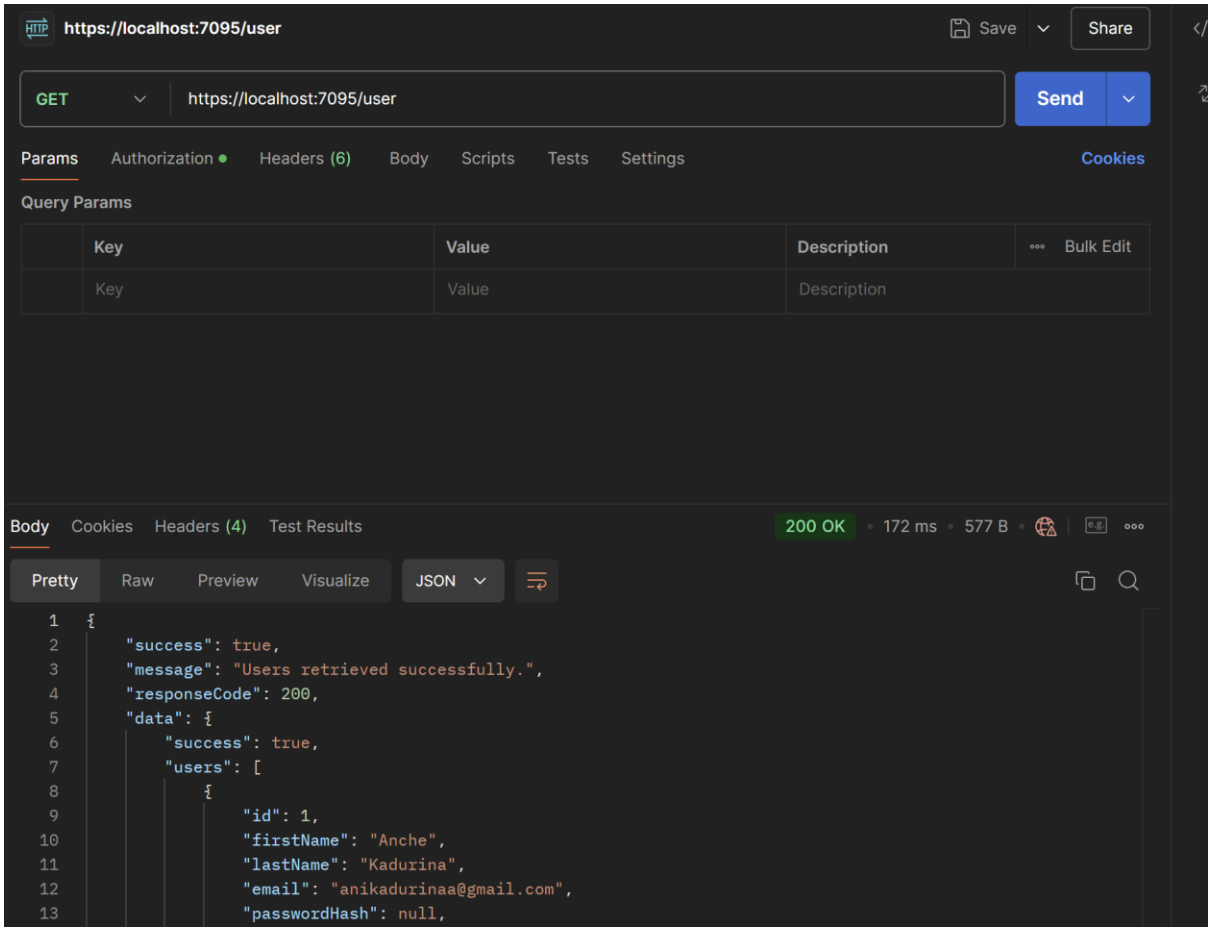


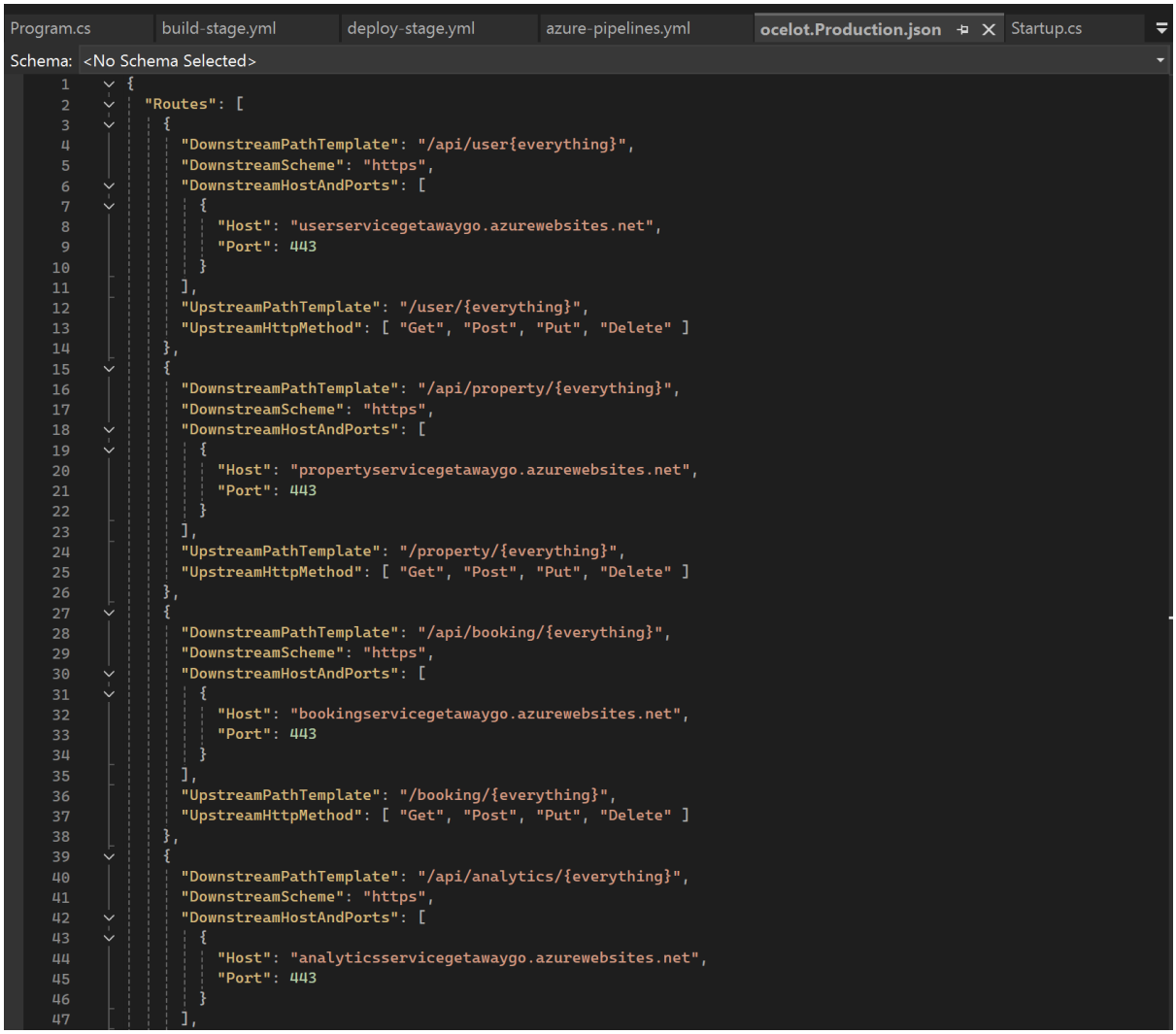
Figure 2 – Local testing of the API Gateway

Production configuration

For Production, I am currently using the same logic, but with the Production hosts, so the ones deployed in Azure. In the near future, I will remove all the urls from the file and put them in a variable group to be picked up in the pipeline.

Program.cs | build-stage.yml | deploy-stage.yml | azure-pipelines.yml | ocelot.Production.json | Startup.cs

Schema: <No Schema Selected>



```
{
  "Routes": [
    {
      "DownstreamPathTemplate": "/api/user{everything}",
      "DownstreamScheme": "https",
      "DownstreamHostAndPorts": [
        {
          "Host": "userservicegetawaygo.azurewebsites.net",
          "Port": 443
        }
      ],
      "UpstreamPathTemplate": "/user/{everything}",
      "UpstreamHttpMethod": [ "Get", "Post", "Put", "Delete" ]
    },
    {
      "DownstreamPathTemplate": "/api/property/{everything}",
      "DownstreamScheme": "https",
      "DownstreamHostAndPorts": [
        {
          "Host": "propertyservicegetawaygo.azurewebsites.net",
          "Port": 443
        }
      ],
      "UpstreamPathTemplate": "/property/{everything}",
      "UpstreamHttpMethod": [ "Get", "Post", "Put", "Delete" ]
    },
    {
      "DownstreamPathTemplate": "/api/booking/{everything}",
      "DownstreamScheme": "https",
      "DownstreamHostAndPorts": [
        {
          "Host": "bookingservicegetawaygo.azurewebsites.net",
          "Port": 443
        }
      ],
      "UpstreamPathTemplate": "/booking/{everything}",
      "UpstreamHttpMethod": [ "Get", "Post", "Put", "Delete" ]
    },
    {
      "DownstreamPathTemplate": "/api/analytics/{everything}",
      "DownstreamScheme": "https",
      "DownstreamHostAndPorts": [
        {
          "Host": "analyticsservicegetawaygo.azurewebsites.net",
          "Port": 443
        }
      ],
```

Figure 3 – Production configuration of the API Gateway – Part 1

Program.cs | build-stage.yml | deploy-stage.yml | azure-pipelines.yml | ocelot.Production.json | Startup.cs

Schema: <No Schema Selected>

```
      }
    ],
    "UpstreamPathTemplate": "/analytics/{everything}",
    "UpstreamHttpMethod": [ "Get", "Post", "Put", "Delete" ]
  },
  {
    "DownstreamPathTemplate": "/api/chat/{everything}",
    "DownstreamScheme": "https",
    "DownstreamHostAndPorts": [
      {
        "Host": "chatservicegetawaygo.azurewebsites.net",
        "Port": 443
      }
    ],
    "UpstreamPathTemplate": "/chat/{everything}",
    "UpstreamHttpMethod": [ "Get", "Post", "Put", "Delete" ]
  },
  {
    "DownstreamPathTemplate": "/api/notification/{everything}",
    "DownstreamScheme": "https",
    "DownstreamHostAndPorts": [
      {
        "Host": "notificationservicegetawaygo.azurewebsites.net",
        "Port": 443
      }
    ],
    "UpstreamPathTemplate": "/notification/{everything}",
    "UpstreamHttpMethod": [ "Get", "Post", "Put", "Delete" ]
  },
  {
    "DownstreamPathTemplate": "/api/review/{everything}",
    "DownstreamScheme": "https",
    "DownstreamHostAndPorts": [
      {
        "Host": "getawaygoreviewservice.azurewebsites.net",
        "Port": 443
      }
    ],
    "UpstreamPathTemplate": "/review/{everything}",
    "UpstreamHttpMethod": [ "Get", "Post", "Put", "Delete" ]
  }
],
"GlobalConfiguration": {
  "BaseUrl": "https://apigatewaygetawaygo.azurewebsites.net"
}
}
```

Figure 4 – Production configuration of the API Gateway – Part 2

The API Gateway is already deployed and running on Azure in its own resource group.

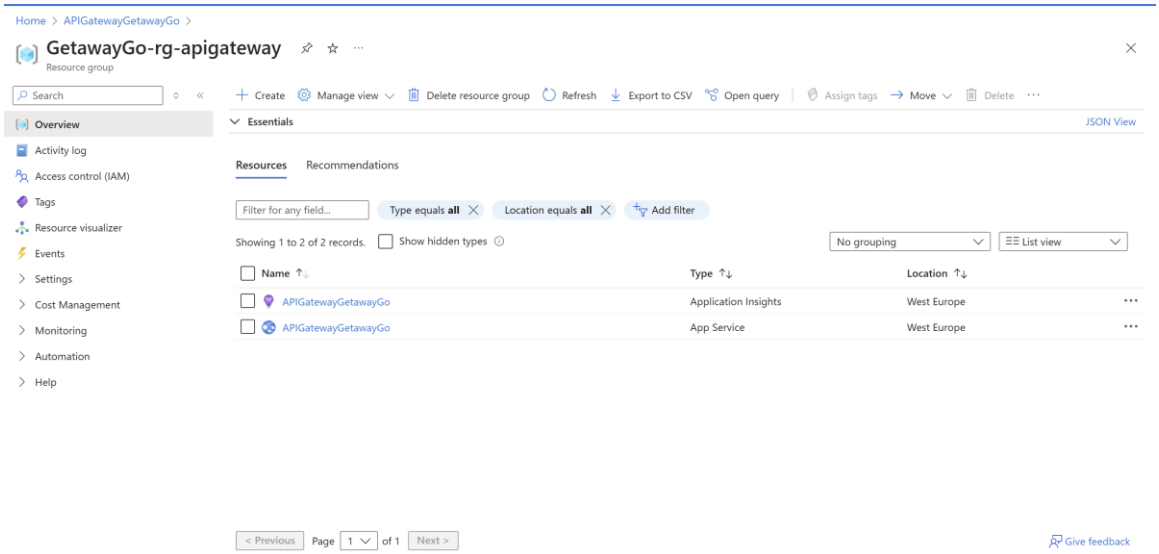


Figure 5 – API Gateway deployed on Azure

Figure 6 – Production testing of the API Gateway - users

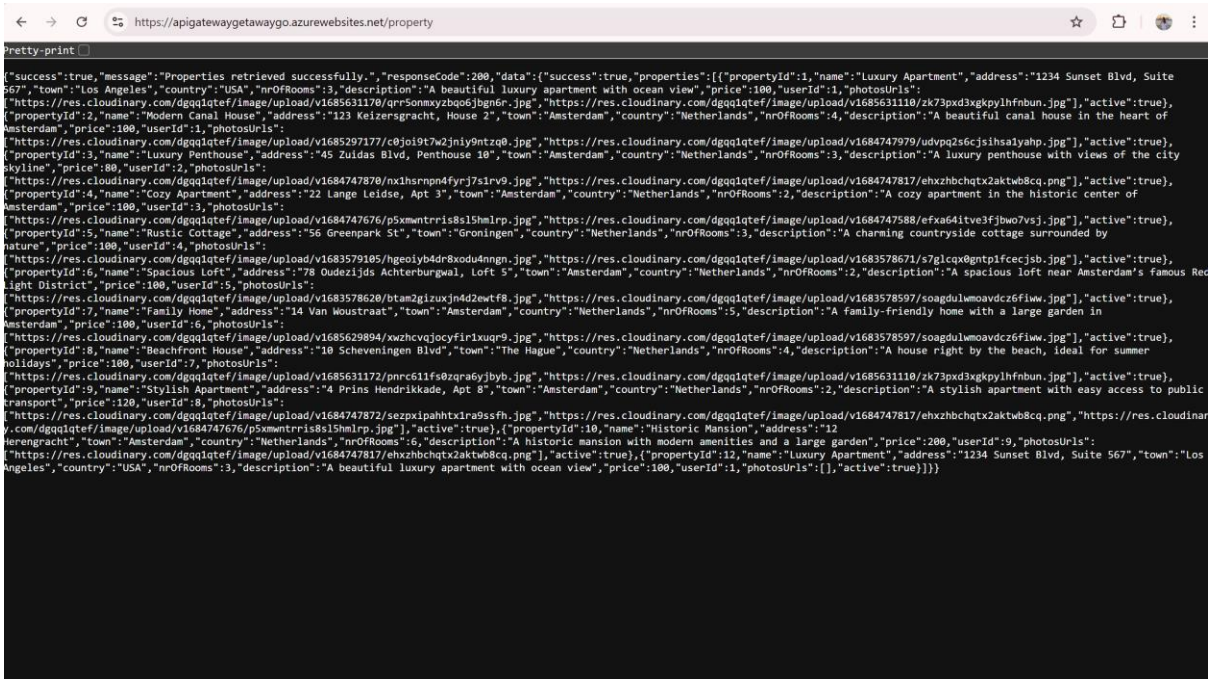


Figure 7 – Production testing of the API Gateway – properties

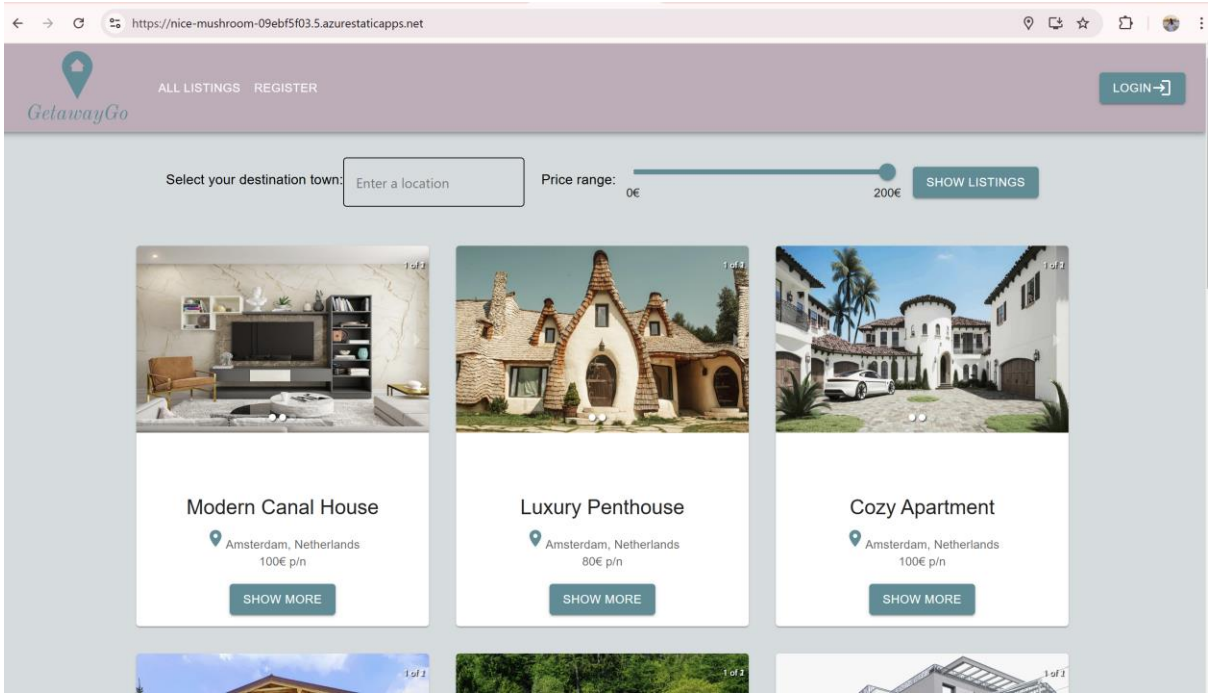


Figure 8 – Frontend connected with API Gateway