Sri Lanka Institute of Information Technology

Assignment 03 : Implement microservice backend for the minimal viable product (MVP)

IT4020 - Modern Topics in IT

Year 4, Semester 1

(2023) – Weekend Batch

Group Details

Group ID - MTIT-011

Project Name - Implement microservice backend for the minimal viable product (MVP)

Our business domain is Dialog Axiata company

Batch - 2023 -Y4S1

Registration Number	Student Name
IT20147396	Peiris B M G
IT20178154	Dilshan P A D S D
IT20081416	Ahamed M M Z
IT20122782	Amani M P N

Student Contribution

REGISTRATION NUMBER	STUDENT CONTRIBUTION
IT20147396	• Shop Manager can ADD communication items. • Shop Manager can VIEW communication items. • Shop Manager can UPDATE communication items. • Shop Manager can DELETE communication items.
IT20178154	• Supplier can ADD supplier menu. • Supplier can VIEW the supplier menu. • Supplier can UPDATE supplier menu. • Supplier can DELETE supplier menu.
IT20122782	• Storekeeper can ADD storekeeper menu. • Storekeeper can VIEW the storekeeper menu. • Storekeeper can UPDATE storekeeper menu. • Storekeeper can DELETE storekeeper menu
IT20081416	• Delivery person can ADD delivery items. • Delivery person can VIEW delivery items. • Delivery person can UPDATE delivery items . • Delivery person can DELETE delivery items .

The Scenario

Our business domain is Dialog Axiata company

Dialog Axiata is a Sri Lankan telecommunications company that provides mobile, fixed-line, and broadband services to customers in Sri Lanka. The company has embraced microservices and cloud computing as a way to deliver services more efficiently and effectively.Dialog Axiata has implemented a number of microservices, including Manager API, Storekeeper API, Supplier API, and Delivery Person API. Each microservice has a specific role and set of responsibilities:

Storekeeper API microservice: This microservice provides a set of RESTful endpoints to manage inventory-related resources, such as stock levels, product information, and order fulfillment. The Storekeeper API microservice helps to optimize inventory management processes, reduce costs, and improve customer satisfaction.

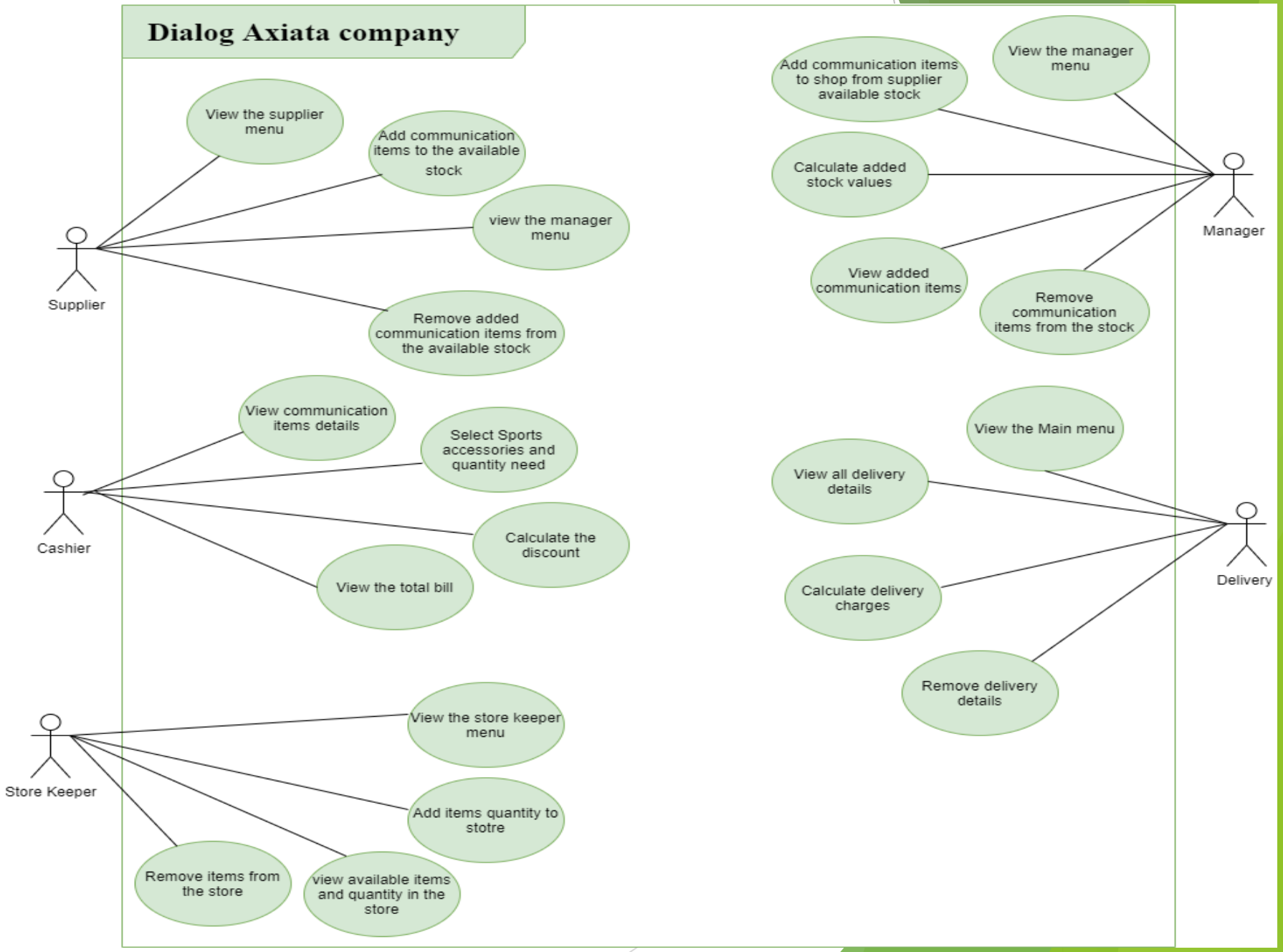
Supplier API microservice: This microservice provides a set of RESTful endpoints to manage supplier-related resources, such as supplier information, purchase orders, and payment processing. The Supplier API microservice helps to streamline supplier management processes and improve the efficiency of the procurement process.

Manager API microservice: This microservice provides a set of RESTful endpoints to manage employee-related resources, such as employee information, work schedules, and performance reviews. The Manager API microservice helps managers to efficiently manage their teams and improve employee productivity.

Delivery Person API microservice: This microservice provides a set of RESTful endpoints to manage delivery-related resources, such as delivery orders, delivery schedules, and delivery routes. The Delivery Person API microservice helps to optimize delivery management processes, improve delivery performance, and enhance the overall customer experience.

To implement these microservices, Dialog Axiata has also adopted cloud computing technologies, such as containerization, Kubernetes, and cloud-based databases. These technologies enable Dialog Axiata to deploy and manage microservices more efficiently and cost-effectively, while also providing scalability, reliability, and high availability.

System Diagram



Define and elaborate the microservices

Sports Delivery service

1.Define the scope and functionality of the Delivery Person API microservice:

Identify the delivery-related resources that the microservice will manage.Define the CRUD operations that will be supported by the microservice for each resource.Specify the input and output parameters for each operation.

2.Design the architecture of the Delivery Person API microservice:

Choose a programming language and framework to build the microservice.Define the data models for the delivery-related resources that will be managed by the microservice.Decide on the data storage mechanism for the microservice (e.g., relational database, NoSQL database, in-memory cache).

3.Implement the Delivery Person API microservice:

Write the code for the RESTful endpoints that will handle the requests from clients.Implement the business logic for each endpoint.

Integrate the microservice with any necessary external services (e.g., delivery management systems, authentication and authorization services).

4.Test and deploy the Delivery Person API microservice:

Write test cases for each endpoint to ensure that the microservice functions as expected.Deploy the microservice to a production environment.Monitor the performance of the microservice and make any necessary improvements.

5.Document the Delivery Person API microservice:

Create API documentation that describes the endpoints, input/output parameters, and error codes.Provide examples of how to use the API.Publish the documentation to a centralized location for developers to access.

Sports Supplier service

1.Define the scope and functionality of the Supplier API microservice:

Identify the supplier-related resources that the microservice will manage.Define the CRUD operations that will be supported by the microservice for each resource.Specify the input and output parameters for each operation.

2.Design the architecture of the Supplier API microservice:

Choose a programming language and framework to build the microservice.Define the data models for the supplier-related resources that will be managed by the microservice.Decide on the data storage mechanism for the microservice (e.g., relational database, NoSQL database, in-memory cache).

3.Implement the Supplier API microservice:

Write the code for the RESTful endpoints that will handle the requests from clients.Implement the business logic for each endpoint.Integrate the microservice with any necessary external services (e.g., supplier databases, authentication and authorization services).

4.Test and deploy the Supplier API microservice:

Write test cases for each endpoint to ensure that the microservice functions as expected.Deploy the microservice to a production environment.Monitor the performance of the microservice and make any necessary improvements.

5.Document the Supplier API microservice:

Create API documentation that describes the endpoints, input/output parameters, and error codes.Provide examples of how to use the API.Publish the documentation to a centralized location for developers to access.

Sports Manager service

1.Define the scope and functionality of the Manager API microservice:

Identify the resources that the microservice will manage.Define the CRUD operations that will be supported by the microservice for each resource.

Specify the input and output parameters for each operation.

2.Design the architecture of the Manager API microservice:

Choose a programming language and framework to build the microservice.Define the data models for the resources that will be managed by the microservice.Decide on the data storage mechanism for the microservice (e.g., relational database, NoSQL database, in-memory cache).

3.Implement the Manager API microservice:

Write the code for the RESTful endpoints that will handle the requests from clients.Implement the business logic for each endpoint.Integrate the microservice with any necessary external services.

4.Test and deploy the Manager API microservice:

Write test cases for each endpoint to ensure that the microservice functions as expected.Deploy the microservice to a production environment.

Monitor the performance of the microservice and make any necessary improvements.

5.Document the Manager API microservice:

Create API documentation that describes the endpoints, input/output parameters, and error codes.Provide examples of how to use the API.

Publish the documentation to a centralized location for developers to access.

Sports Storekeeper service

1.Define the scope and functionality of the Storekeeper API microservice:

Identify the inventory-related resources that the microservice will manage.Define the CRUD operations that will be supported by the microservice for each resource.Specify the input and output parameters for each operation.

2.Design the architecture of the Storekeeper API microservice:

Choose a programming language and framework to build the microservice.Define the data models for the inventory-related resources that will be managed by the microservice.Decide on the data storage mechanism for the microservice (e.g., relational database, NoSQL database, in-memory cache).

3.Implement the Storekeeper API microservice:

Write the code for the RESTful endpoints that will handle the requests from clients.Implement the business logic for each endpoint.Integrate the microservice with any necessary external services (e.g., inventory management systems, authentication and authorization services).

4.Test and deploy the Storekeeper API microservice:

Write test cases for each endpoint to ensure that the microservice functions as expected.Deploy the microservice to a production environment.Monitor the performance of the microservice and make any necessary improvements.

5.Document the Storekeeper API microservice:

Create API documentation that describes the endpoints, input/output parameters, and error codes.Provide examples of how to use the API.Publish the documentation to a centralized location for developers to access.

Explain the usage of API gateway

- Centralized API management - An API Gateway serves as a centralized point for managing all the APIs in a system. It can handle requests from multiple clients and route them to the appropriate backend services.

- Security - An API Gateway can provide an additional layer of security to APIs by enforcing authentication and authorization rules. It can also protect against common attacks like DDoS, SQL injection, and cross-site scripting.

- Traffic management- An API Gateway can distribute incoming requests across multiple instances of a backend service to ensure optimal performance and availability. It can also perform load balancing, caching, and rate limiting to prevent service overload.

- Monitoring and analytics-An API Gateway can collect data on the usage of APIs and provide insights into performance, errors, and usage patterns. This information can be used to optimize the API design and improve the user experience.

- Protocol translation-An API Gateway can translate between different protocols and data formats to enable communication between different systems. For example, it can translate RESTful API requests into SOAP requests or vice versa.

- Simplified client development- An API Gateway can provide a unified interface for multiple backend services, making it easier for clients to interact with them. It can also hide the complexity of the underlying services by providing a simpler API.

IT20178154

Dilshan P A D S D

Supplier Service

Supplier Service

Elaborate how you can avoid having multiple ports with the gateway you implemented and Solution should have proper folder structure with minimum two microservices and an Ocelot API Gateway.

Ocelot.json in Supplier Service

Process: [12228] MTIT.Manager.exe | Lifecycle Events | Thread: | Stack Frame:

ocelot.json | Program.cs | MTIT.Ocelote.csproj:2 | Program.cs | launchSettings.json | Program.cs | launchSettings.json | launchSettings.json

Schema: https://json.schemastore.org/ocelot.json

```
1  {
2
3    "GlobalConfiguration": {
4
5      "BaseUrl": "http://localhost:5071"
6
7    },
8
9    "Routes": [
10
11     {
12
13       "UpstreamPathTemplate": "/gateway/mobile",
14
15       "UpstreamHttpMethod": [ "Get", "Post", "Put" ],
16
17       "DownstreamPathTemplate": "/api/mobile",
18
19       "DownstreamScheme": "http",
20
21       "DownstreamHostAndPorts": [
22
23         {
24
25           "Host": "localhost",
26
27           "Port": 5099
28
29         }
30
31       ]
```

82 % | No issues found | Ln: 17 | Ch: 34 | SPC | CRLF

Ocelot.json in Supplier Service

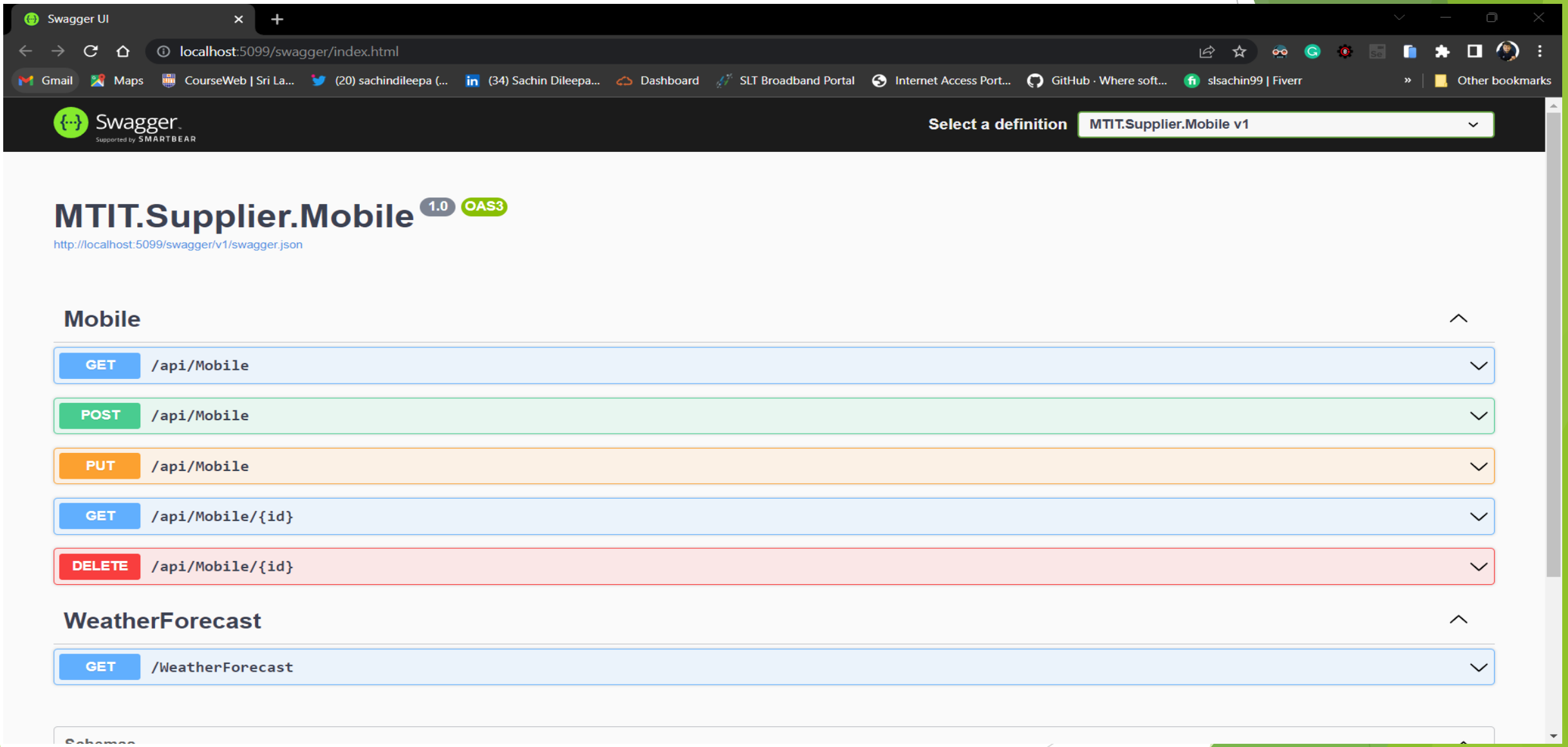
Process: [12228] MTIT.Manager.exe | Lifecycle Events | Thread: | Stack Frame:

ocelot.json | Program.cs | MTIT.Ocelote.csproj:2 | Program.cs | launchSettings.json | Program.cs | launchSettings.json | launchSettings.json

Schema: https://json.schemastore.org/ocelot.json

```
33      },
34
35      {
36
37        "UpstreamPathTemplate": "/gateway/mobile/{Id}",
38
39        "UpstreamHttpMethod": [ "Get", "Delete" ],
40
41        "DownstreamPathTemplate": "/api/mobile/{Id}",
42
43        "DownstreamScheme": "http",
44
45        "DownstreamHostAndPorts": [
46
47          {
48
49            "Host": "localhost",
50
51            "Port": 5099
52
53          }
54
55        ]
56
57      }
58
59
60    ]
61
62  }
```

Screen captures OF Supplier API



GET Method

Swagger UI

localhost:5099/swagger/index.html

Gmail | Maps | CourseWeb | Sri La... | (20) sachindileepa (... | (34) Sachin Dileepa... | Dashboard | SLT Broadband Portal | Internet Access Port... | GitHub · Where soft... | slsachin99 | Fiverr | Other bookmarks

GET /api/Mobile

Parameters

Cancel

No parameters

Execute | Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:5099/api/Mobile' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:5099/api/Mobile
```

Server response

Code | Details

200

Response body


```
[
  {
    "id": 1,
    "brand": "Apple",
    "country": "USA",
    "price": 200000
  },
  {
    "id": 2,
    "brand": "Huawei",
    "country": "China",
    "price": 22000
  },
  {
    "id": 3,
    "brand": "Samsung",
    "country": "Korea",
    "price": 100000
  },
  {
    "id": 4,
    "brand": "Oppo",
    "country": "India",
    "price": 23000
  },
  {
    "id": 5,
    "brand": "Nokia",
```


Download

POST Method

POST /api/Mobile

Parameters

Cancel Reset

No parameters

Request body

application/json

```
{
  "id": 6,
  "brand": "Sony",
  "country": "England",
  "price": 23000
}
```

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:5099/api/Mobile' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
  "id": 6,
  "brand": "Sony",
  "country": "England",
  "price": 23000
}'
```

Request URL

http://localhost:5099/api/Mobile

Server response

Code	Details
200	Response body

```
{
  "id": 6,
  "brand": "Sony",
  "country": "England",
  "price": 23000
}
```

Download

Response headers

```
content-type: application/json; charset=utf-8
date: Thu,13 Apr 2023 06:25:13 GMT
server: Kestrel
transfer-encoding: chunked
```

Responses

Code	Description	Links
200	Success	*No links*

PUT Method

PUT /api/Mobile

Parameters

Cancel Reset

No parameters

Request body

application/json

```
{
  "id": 1,
  "brand": "Apple 13",
  "country": "string",
  "price": 0
}
```

Responses

Curl

```
curl -X 'PUT' \
  'http://localhost:5099/api/Mobile' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
  "id": 1,
  "brand": "Apple 13",
  "country": "string",
  "price": 0
}'
```

Request URL

```
http://localhost:5099/api/Mobile
```

Server response

Code	Details

200	Response body
	`{ "id": 1, "brand": "Apple 13", "country": "string", "price": 0 }`
	Download
	Response headers
	`content-type: application/json; charset=utf-8`
	`date: Thu,13 Apr 2023 06:29:20 GMT`
	`server: Kestrel`
	`transfer-encoding: chunked`

Responses

Code	Description	Links
200	Success	*No links*

SEARCH

GET /api/Mobile/{id}

Parameters

Cancel

Name	Description
id * required integer($int32) (path)	4

Execute | Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:5099/api/Mobile/4' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:5099/api/Mobile/4
```

Server response

Code	Details
200	Response body

```
{
  "id": 4,
  "brand": "Oppo",
  "country": "India",
  "price": 23000
}
```

Download

Response headers

```
content-type: application/json; charset=utf-8
date: Thu,13 Apr 2023 06:34:35 GMT
server: Kestrel
transfer-encoding: chunked
```

Responses

Code	Description	Links
200	Success	No links

DELETE Method

DELETE /api/Mobile/{id}

Parameters

Cancel

Name	Description
id * required integer($int32) (path)	6

Execute | Clear

Responses

Curl

```
curl -X 'DELETE' \
  'http://localhost:5099/api/Mobile/6' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:5099/api/Mobile/6
```

Server response

Code	Details
200	Response body

```
mobile with ID:6 got deleted successfully.
```

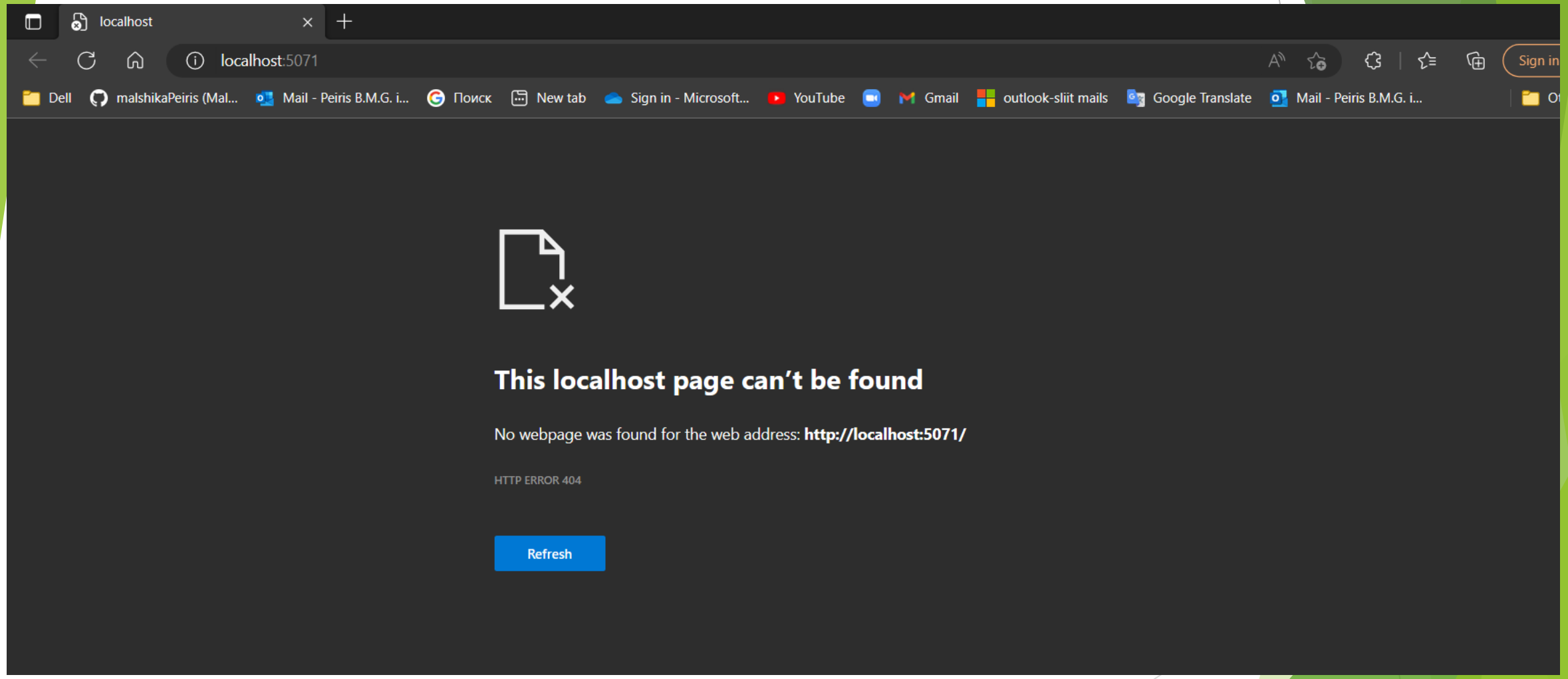
Download

Response headers

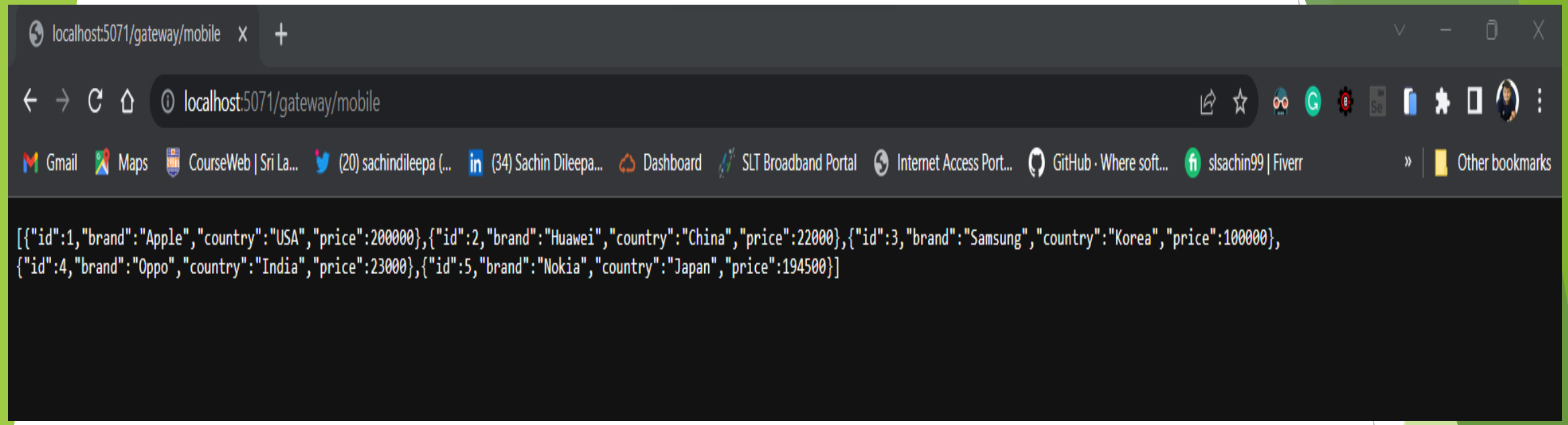
```
content-type: text/plain; charset=utf-8
date: Thu,13 Apr 2023 06:37:03 GMT
server: Kestrel
transfer-encoding: chunked
```

Gateway project and the API project will both run and open up in two separate browsers.

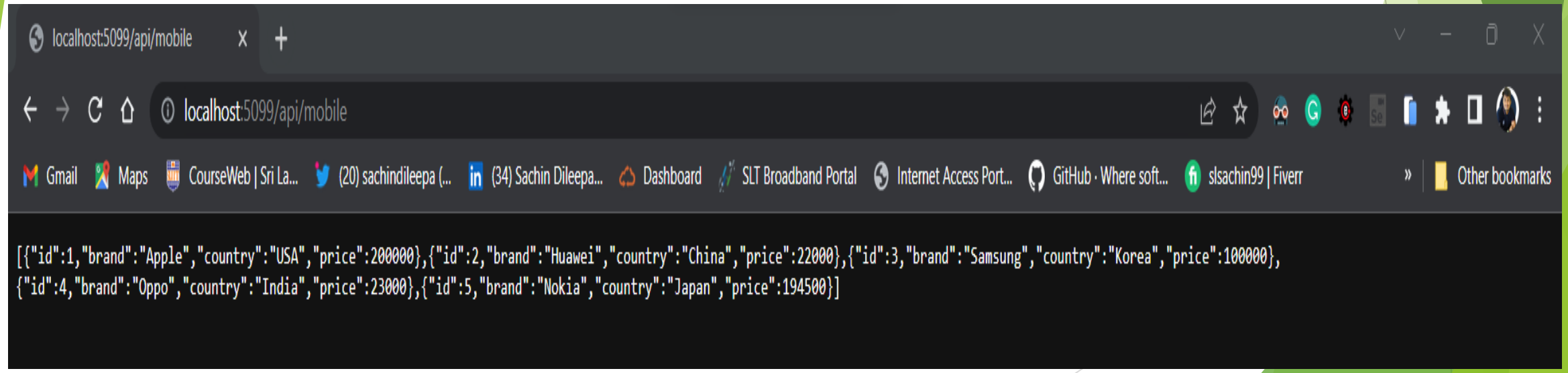
Gateway:



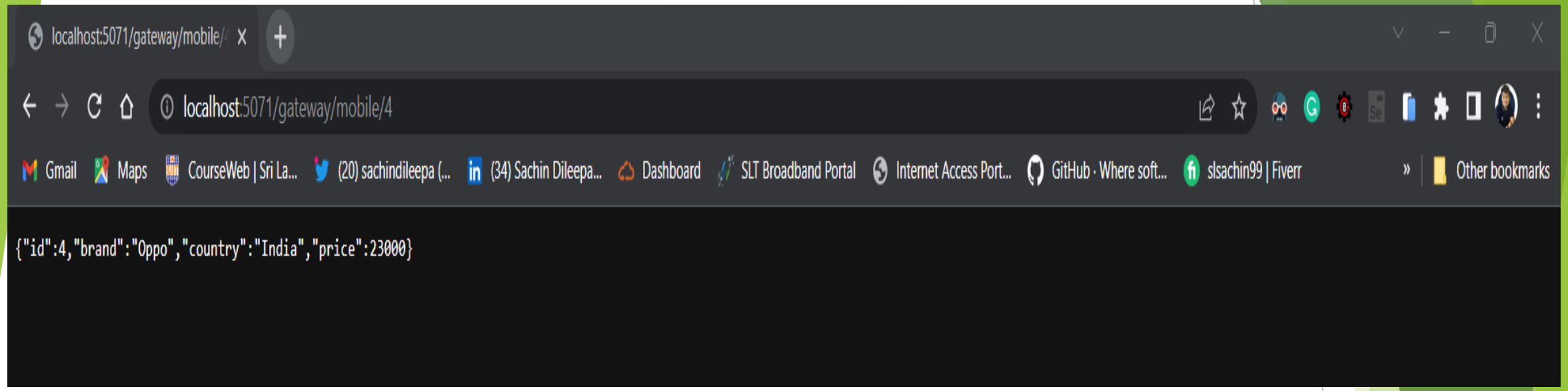
Test the upstream URL in browser.



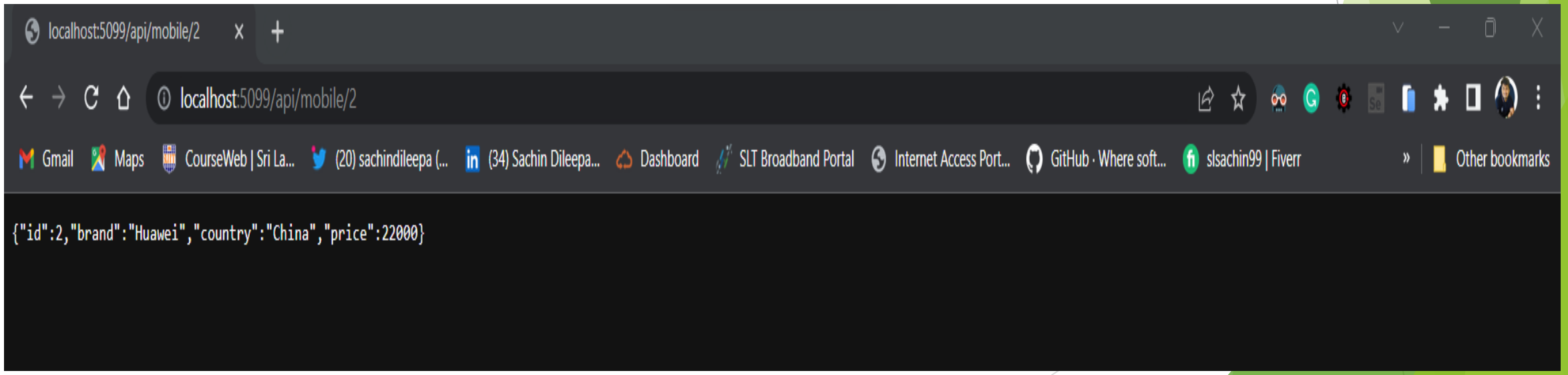
Test the downstream URL in browser.



Test the upstream URL in browser.



Test the downstream URL in browser.



IT20147396

Peiris B M G

Manager Service

Manager Service

Elaborate how you can avoid having multiple ports with the gateway you implemented and Solution should have proper folder structure with minimum two microservices and an Ocelot API Gateway.

launchsettings.json in Manager service

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+Q) Manager.MTIT

Debug Any CPU <Multiple Startup Projects> Continue

Process: [5396] MTIT.Manager.exe Lifecycle Events Thread: Stack Frame:

launchSettings.json ocelot.json Program.cs

Schema: https://json.schemastore.org/launchsettings.json

```
1  {
2    "$schema": "https://json.schemastore.org/launchsettings.json",
3    "iisSettings": {
4      "windowsAuthentication": false,
5      "anonymousAuthentication": true,
6      "iisExpress": {
7        "applicationUrl": "http://localhost:56706",
8        "sslPort": 0
9      }
10   },
11   "profiles": {
12     "MTIT.Manager": {
13       "commandName": "Project",
14       "dotnetRunMessages": true,
15       "launchBrowser": true,
16       "launchUrl": "swagger",
17       "applicationUrl": "http://localhost:5109",
18       "environmentVariables": {
19         "ASPNETCORE_ENVIRONMENT": "Development"
20       }
21     },
22     "IIS Express": {
23       "commandName": "IISExpress",
24       "launchBrowser": true,
25       "launchUrl": "swagger",
26       "environmentVariables": {
27         "ASPNETCORE_ENVIRONMENT": "Development"
28       }
```

Ocelot.json in Manager service

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+Q) Manager.MTIT

Debug Any CPU <Multiple Startup Projects> Continue

Process: [5396] MTIT.Manager.exe Lifecycle Events Thread: Stack Frame:

launchSettings.json | ocelot.json | Program.cs

Schema: https://json.schemastore.org/ocelot.json

```
1  {
2
3    "GlobalConfiguration": {
4
5      "BaseUrl": "http://localhost:5071"
6
7    },
8
9    "Routes": [
10
11     {
12
13       "UpstreamPathTemplate": "/gateway/manager",
14
15       "UpstreamHttpMethod": [ "Get" , "Post" , "Put" ],
16
17       "DownstreamPathTemplate": "/api/manager",
18
19       "DownstreamScheme": "http",
20
21       "DownstreamHostAndPorts": [
22
23         {
24
25           "Host": "localhost",
26
27           "Port": 5109
28
```

100 % No issues found

Autos | Immediate Window

Search (Ctrl+E) Search Depth:

Ocelot.json in Manager service

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+Q) Manager.MTIT

Debug Any CPU <Multiple Startup Projects> Continue

Process: [5396] MTIT.Manager.exe Lifecycle Events Thread: Stack Frame:

launchSettings.json | ocelot.json | Program.cs

Schema: https://json.schemastore.org/ocelot.json

```
31        ]
32
33      },
34
35      {
36
37        "UpstreamPathTemplate": "/gateway/manager/{Id}",
38
39        "UpstreamHttpMethod": [ "Get" , "Delete" ],
40
41        "DownstreamPathTemplate": "/api/manager/{Id}",
42
43        "DownstreamScheme": "http",
44
45        "DownstreamHostAndPorts": [
46
47          {
48
49            "Host": "localhost",
50
51            "Port": 5109
52
53          }
54
55        ]
56
57      }
58
```

100 % No issues found Ln: 13

Autos Search (Ctrl+E) Search Depth:

Immediate Window

Program.cs in Manager service

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+Q) Manager.MTIT

Debug Any CPU <Multiple Startup Projects> Continue

Process: [5396] MTIT.Manager.exe Lifecycle Events Thread: Stack Frame:

launchSettings.json ocelot.json Program.cs

MTIT.Manager.Ocelote

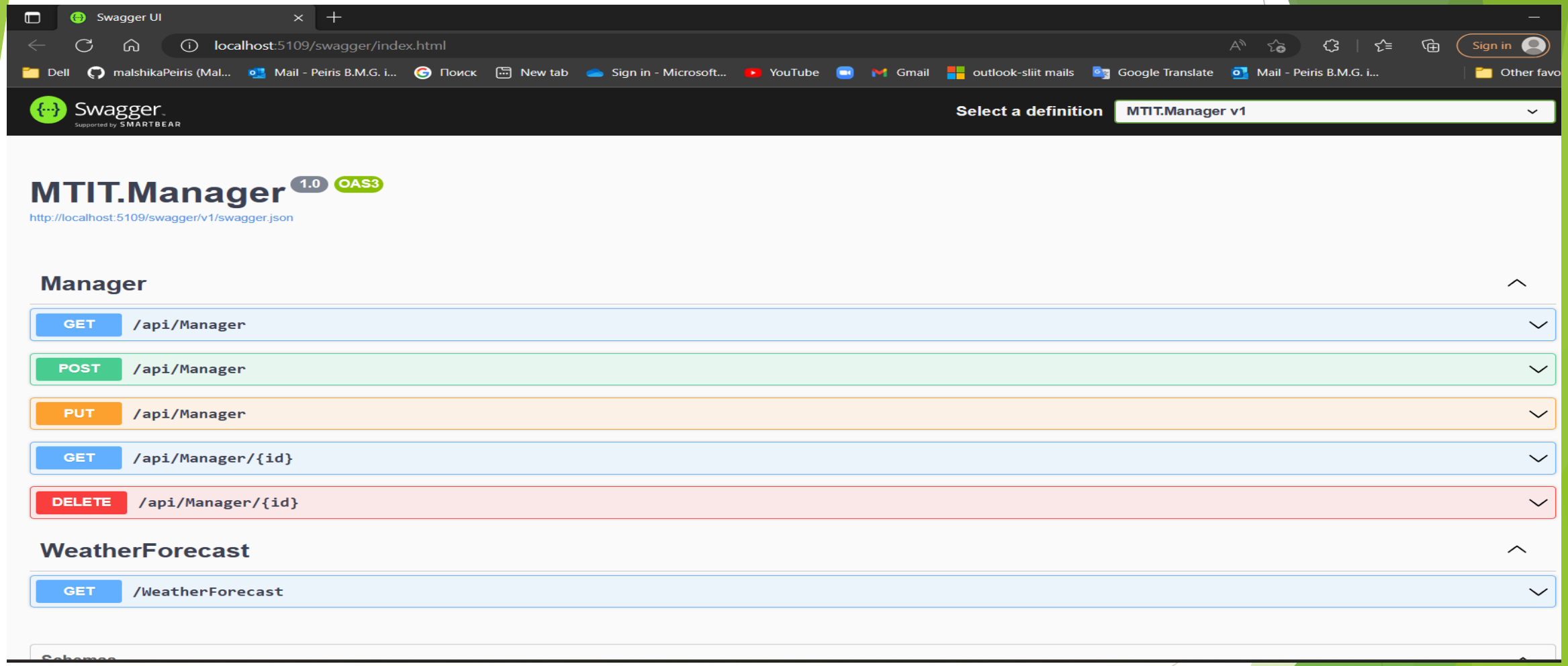
```
using Ocelot.DependencyInjection;
using Ocelot.Middleware;

var builder = WebApplication.CreateBuilder(args);
builder.Configuration.AddJsonFile("ocelot.json");
builder.Services.AddOcelot();
var app = builder.Build();

app.MapGet("/", () => "Welcome to Dialog Axiata Company Manager Service!");
app.UseOcelot();

app.Run();
```

100 % No issues found Ln:

Autos Immediate Window

Search (Ctrl+E) Search Depth:

Screen captures OF Manager API



GET Method

Manager

GET /api/Manager

Parameters

Cancel

No parameters

Execute | Clear

Responses

Response body


```
[
  {
    "id": 1,
    "itemName": "Dialog Broadband",
    "itemQuantity": "15",
    "itemPrice": 200000
  },
  {
    "id": 2,
    "itemName": "Dialog TV",
    "itemQuantity": "40",
    "itemPrice": 22000
  },
  {
    "id": 3,
    "itemName": "Dialog Phone",
    "itemQuantity": "78",
    "itemPrice": 21000
  },
  {
    "id": 4,
    "itemName": "Dialog Satellite ",
    "itemQuantity": "32",
    "itemPrice": 23000
  },
  {
    "id": 5,
```


Download

POST Method

POST /api/Manager

Parameters

Cancel Reset

No parameters

Request body

application/json

```
{
  "id": 6,
  "itemName": "SIM",
  "itemQuantity": "30",
  "itemPrice": 700
}
```

Request URL

http://localhost:5109/api/Manager

Server response

Code	Details
200	Response body

```
{
  "id": 6,
  "itemName": "SIM",
  "itemQuantity": "30",
  "itemPrice": 700
}
```

Download

Response headers

```
content-type: application/json; charset=utf-8
date: Wed,12 Apr 2023 14:30:25 GMT
server: Kestrel
transfer-encoding: chunked
```

PUT Method

PUT /api/Manager

Parameters

Cancel Reset

No parameters

Request body

application/json

```
{
  "id": 6,
  "itemName": "SIM",
  "itemQuantity": "44",
  "itemPrice": 890
}
```

Request URL

```
http://localhost:5109/api/Manager
```

Server response

Code	Details
200	Response body

```
{
  "id": 6,
  "itemName": "SIM",
  "itemQuantity": "44",
  "itemPrice": 890
}
```

Download

SEARCH

GET /api/Manager/{id}

Parameters

Cancel

Name	Description
id * required integer($int32) (path)	6

Execute | Clear

Curl

```
curl -X 'GET' \
  'http://localhost:5109/api/Manager/6' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:5109/api/Manager/6
```

Server response

Code	Details
200	Response body

```
{
  "id": 6,
  "itemName": "SIM",
  "itemQuantity": "44",
  "itemPrice": 890
}
```

Download

DELETE Method

DELETE /api/Manager/{id}

Parameters

Cancel

Name	Description
id * required integer($int32) (path)	6

Execute | Clear

Curl

```
curl -X 'DELETE' \
  'http://localhost:5109/api/Manager/6' \
  -H 'accept: */*'
```

Request URL

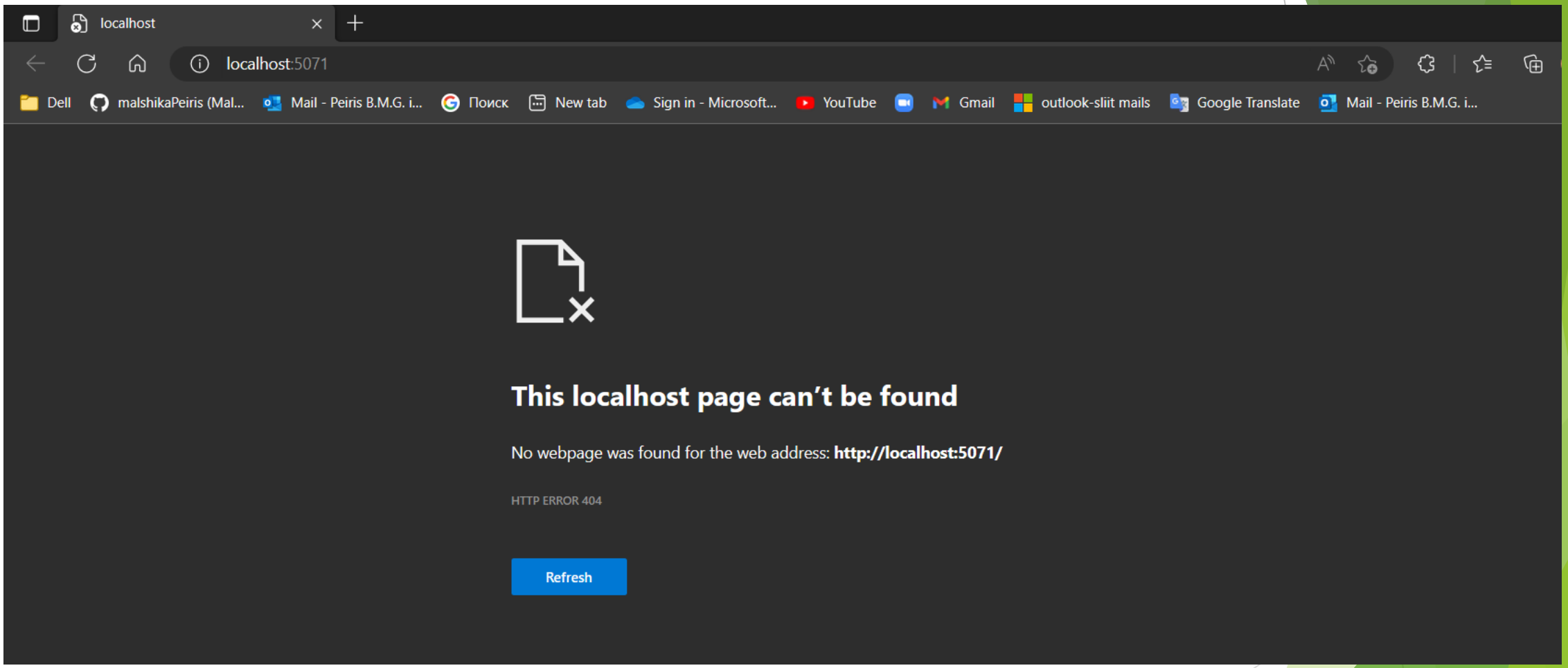
```
http://localhost:5109/api/Manager/6
```

Server response

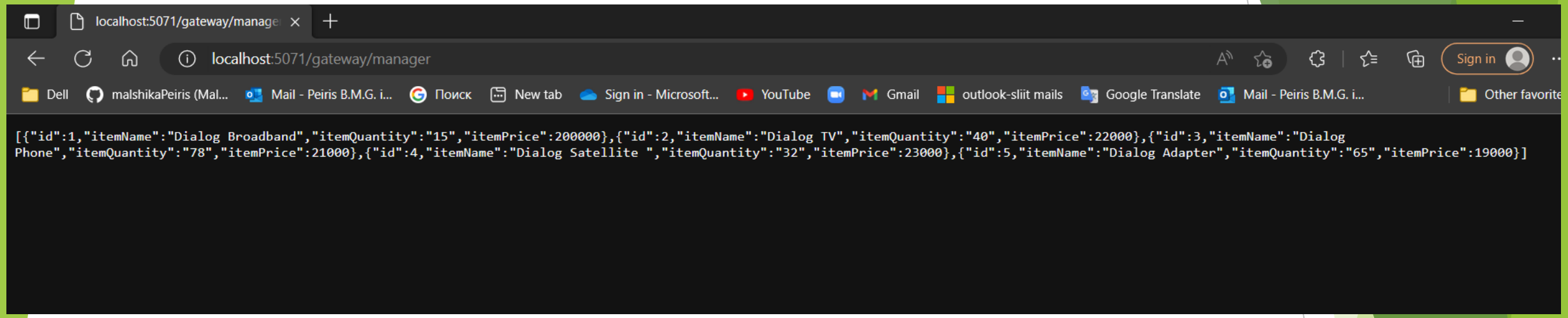
Code	Details
200	Response body `manager with ID:6 got deleted successfully.` Download

Gateway project and the API project will both run and open up in two separate browsers.

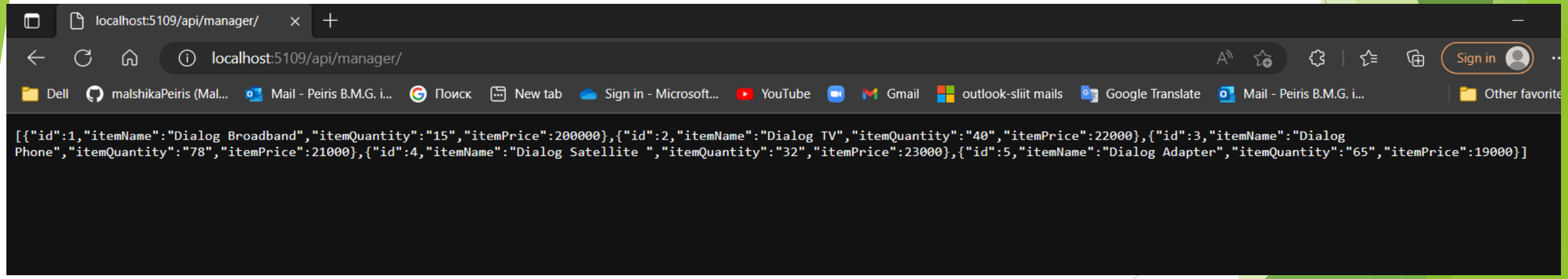
Gateway:



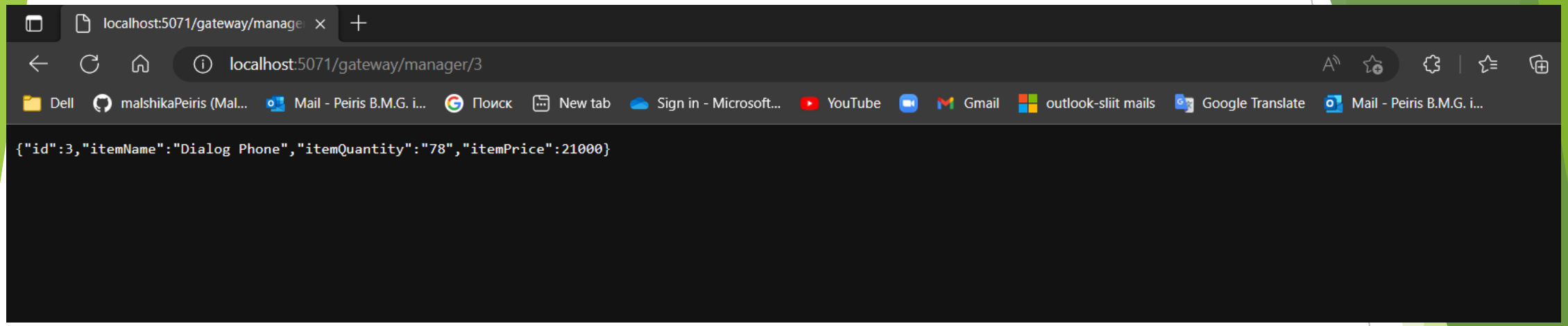
Test the upstream URL in browser.



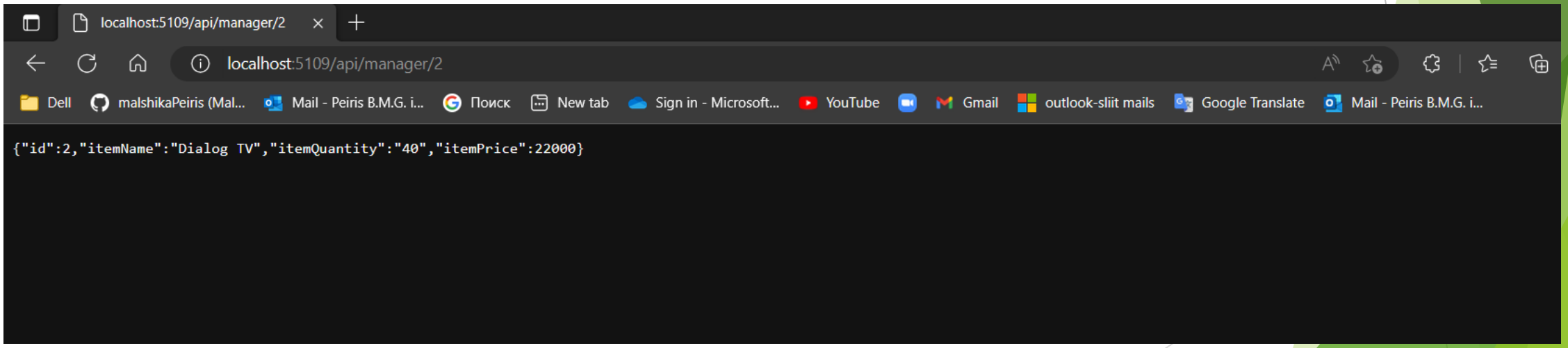
Test the downstream URL in browser.



Test the upstream URL in browser.



Test the downstream URL in browser.



IT20122782

Amani M P N

Store Keeper Service

StoreKeeper Service

Elaborate how you can avoid having multiple ports with the gateway you implemented and Solution should have proper folder structure with minimum two microservices and an Ocelot API Gateway.

launchsettings.json in StoreKeeper service

ocelot.json | launchSettings.json | **launchSettings.json** | MTIT.Ocelote | launchSettings.json

Schema: https://json.schemastore.org/launchsettings.json

```
1  {
2    "$schema": "https://json.schemastore.org/launchsettings.json",
3    "iisSettings": {
4      "windowsAuthentication": false,
5      "anonymousAuthentication": true,
6      "iisExpress": {
7        "applicationUrl": "http://localhost:48409",
8        "sslPort": 0
9      }
10   },
11   "profiles": {
12     "MTIT.Storekeeper": {
13       "commandName": "Project",
14       "dotnetRunMessages": true,
15       "launchBrowser": true,
16       "launchUrl": "swagger",
17       "applicationUrl": "http://localhost:5177",
18       "environmentVariables": {
19         "ASPNETCORE_ENVIRONMENT": "Development"
20       }
21     },
22     "IIS Express": {
23       "commandName": "IISExpress",
24       "launchBrowser": true,
25       "launchUrl": "swagger",
26       "environmentVariables": {
27         "ASPNETCORE_ENVIRONMENT": "Development"
28       }
29     }
30   }
31 }
32
```

Solution Explorer

Search Solution Explorer (Ctrl+;)

- **Solution 'MTIT.Supplier' (5 of 5 projects)**
  - Gateway
    - MTIT.Ocelote
      - Connected Services
      - Dependencies
      - Properties
      - appsettings.json
      - ocelot.json
      - Program.cs
  - Microservices
    - MTIT.Manager
    - MTIT.Storekeeper
      - Connected Services
      - Dependencies
      - Properties
        - launchSettings.json
      - Controllers
      - Data
      - Models
      - Services
      - appsettings.json
      - Program.cs
    - MTIT.Supplier.Mobile
    - MTIT.Transport
      - Connected Services
      - Dependencies
      - Properties

Ocelot.json in StoreKeeper service

launchSettings.json | ocelot.json | launchSettings.json | launchSettings.json | MTIT.Ocelote

Schema: https://json.schemastore.org/ocelot.json

```
132            "UpstreamPathTemplate": "/gateway/storekeeper",
133
134            "UpstreamHttpMethod": [ "Get", "Post", "Put" ],
135
136            "DownstreamPathTemplate": "/api/storekeeper",
137
138            "DownstreamScheme": "http",
139
140            "DownstreamHostAndPorts": [
141
142              {
143
144                "Host": "localhost",
145
146                "Port": 5177
147
148              }
149
150            ]
151
152          },
153
154          {
155
156            "UpstreamPathTemplate": "/gateway/storekeeper/{Id}",
157
158            "UpstreamHttpMethod": [ "Get", "Delete" ],
159
160            "DownstreamPathTemplate": "/api/storekeeper/{Id}",
161
162            "DownstreamScheme": "http",
163
164            "DownstreamHostAndPorts": [
165
166              {
167
168                "Host": "localhost",
169
170                "Port": 5177
171
172              }
```

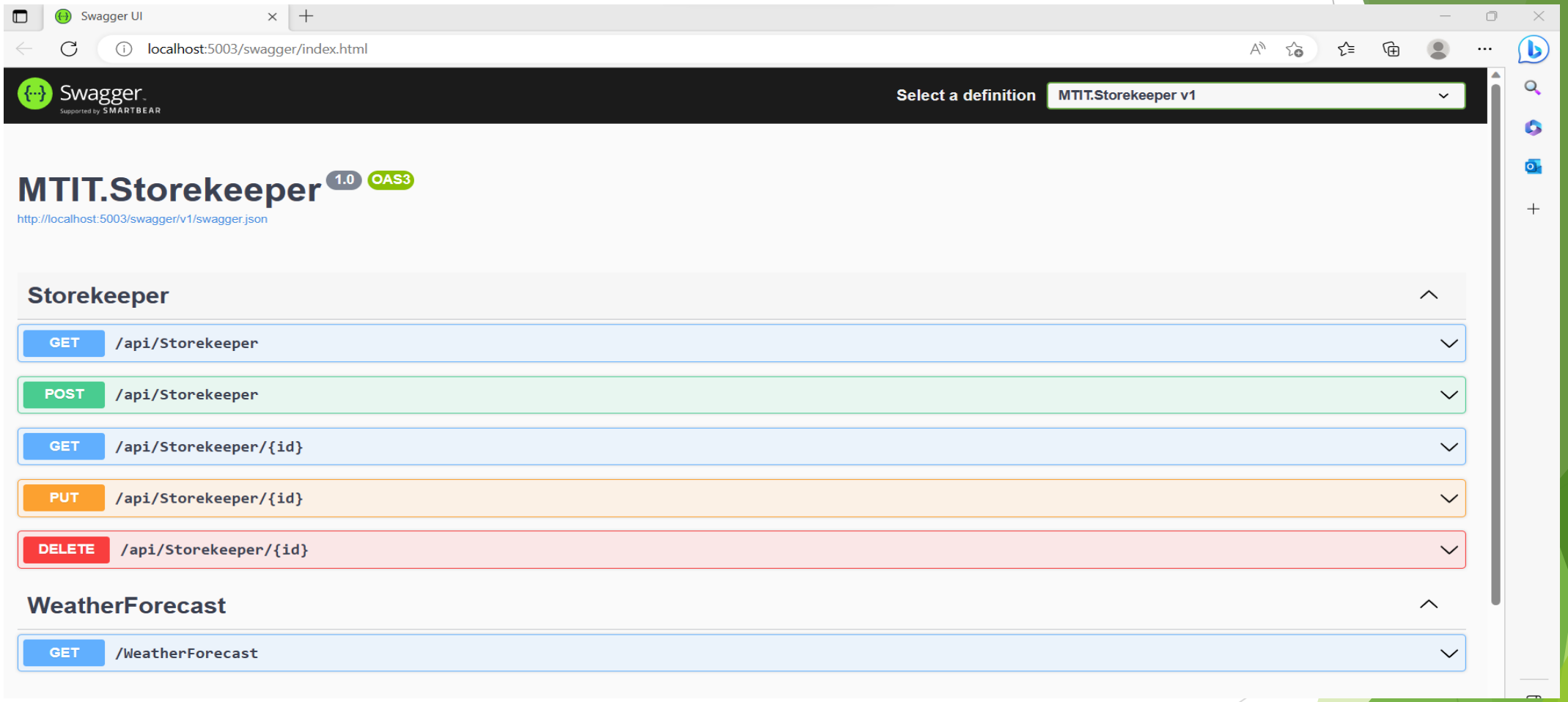
100 % | No issues found | Ln: 100 | Ch: 13 | SPC | CRLF

Solution Explorer

Search Solution Explorer (Ctrl+;)

- **Solution 'MTIT.Supplier' (5 of 5 projects)**
  - Gateway
    - MTIT.Ocelote
      - Connected Services
      - Dependencies
      - Properties
      - appsettings.json
      - ocelot.json
      - Program.cs
  - Microservices
    - MTIT.Manager
    - MTIT.Storekeeper
    - MTIT.Supplier.Mobile
    - MTIT.Transport

Screen captures OF Storekeeper API



GET Method

Swagger UI

localhost:5003/swagger/index.html

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:5003/api/Storekeeper' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:5003/api/Storekeeper
```

Server response

Code	Details
200	Response body


```
[
  {
    "id": 1,
    "itemName": "Dialog Adapter",
    "itemQuantity": "25",
    "itemPrice": 25000
  },
  {
    "id": 2,
    "itemName": "Dialog Phone",
    "itemQuantity": "40",
    "itemPrice": 20000
  },
  {
    "id": 3,
    "itemName": "Dialog Satellite",
    "itemQuantity": "30",
    "itemPrice": 24000
  },
  {
    "id": 4,
    "itemName": "Dialog Broadband ",
    "itemQuantity": "100",
    "itemPrice": 18000
  },
```

POST Method

Swagger UI

localhost:5003/swagger/index.html

POST /api/Storekeeper

Parameters

Cancel | Reset

No parameters

Request body

application/json

```
{
  "id": 6,
  "itemName": "Dialog TV",
  "itemQuantity": "100",
  "itemPrice": 150000
}
```

Swagger UI

localhost:5003/swagger/index.html

Curl

```
curl -X 'POST' \
  'http://localhost:5003/api/Storekeeper' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
  "id": 6,
  "itemName": "Dialog TV",
  "itemQuantity": "100",
  "itemPrice": 150000
}'
```

Request URL

```
http://localhost:5003/api/Storekeeper
```

Server response

Code	Details
200	Response body

```
{
  "id": 6,
  "itemName": "Dialog TV",
  "itemQuantity": "100",
  "itemPrice": 150000
}
```

Download

PUT Method

Swagger UI

localhost:5003/swagger/index.html

PUT /api/Storekeeper/{id}

Parameters

Cancel | Reset

Name	Description
id * required string (path)	2

Request body

application/json

```
{
  "id": 2,
  "itemName": "Dialog Phone",
  "itemQuantity": "500",
  "itemPrice": 200000
}
```

Swagger UI

localhost:5003/swagger/index.html

Responses

Curl

```
curl -X 'PUT' \
  'http://localhost:5003/api/Storekeeper/2' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
  "id": 2,
  "itemName": "Dialog Phone",
  "itemQuantity": "500",
  "itemPrice": 200000
}'
```

Request URL

```
http://localhost:5003/api/Storekeeper/2
```

Server response

Code	Details
200	Response body

```
{
  "id": 2,
  "itemName": "Dialog Phone",
  "itemQuantity": "500",
  "itemPrice": 200000
}
```

Download

SEARCH

Swagger UI

localhost:5003/swagger/index.html

GET /api/Storekeeper/{id}

Parameters

Cancel

Name	Description
id * required integer($int32) (path)	6

Swagger UI

localhost:5003/swagger/index.html

Curl

```
curl -X 'GET' \
  'http://localhost:5003/api/Storekeeper/6' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:5003/api/Storekeeper/6
```

Server response

Code	Details
200	Response body

```
{
  "id": 6,
  "itemName": "Dialog TV",
  "itemQuantity": "100",
  "itemPrice": 150000
}
```

Download

DELETE Method

Swagger UI

localhost:5003/swagger/index.html

DELETE /api/Storekeeper/{id}

Parameters

Cancel

Name	Description
id * required integer($int32) (path)	6

Swagger UI

localhost:5003/swagger/index.html

Responses

Curl

```
curl -X 'DELETE' \
  'http://localhost:5003/api/Storekeeper/6' \
  -H 'accept: */*'
```

Request URL

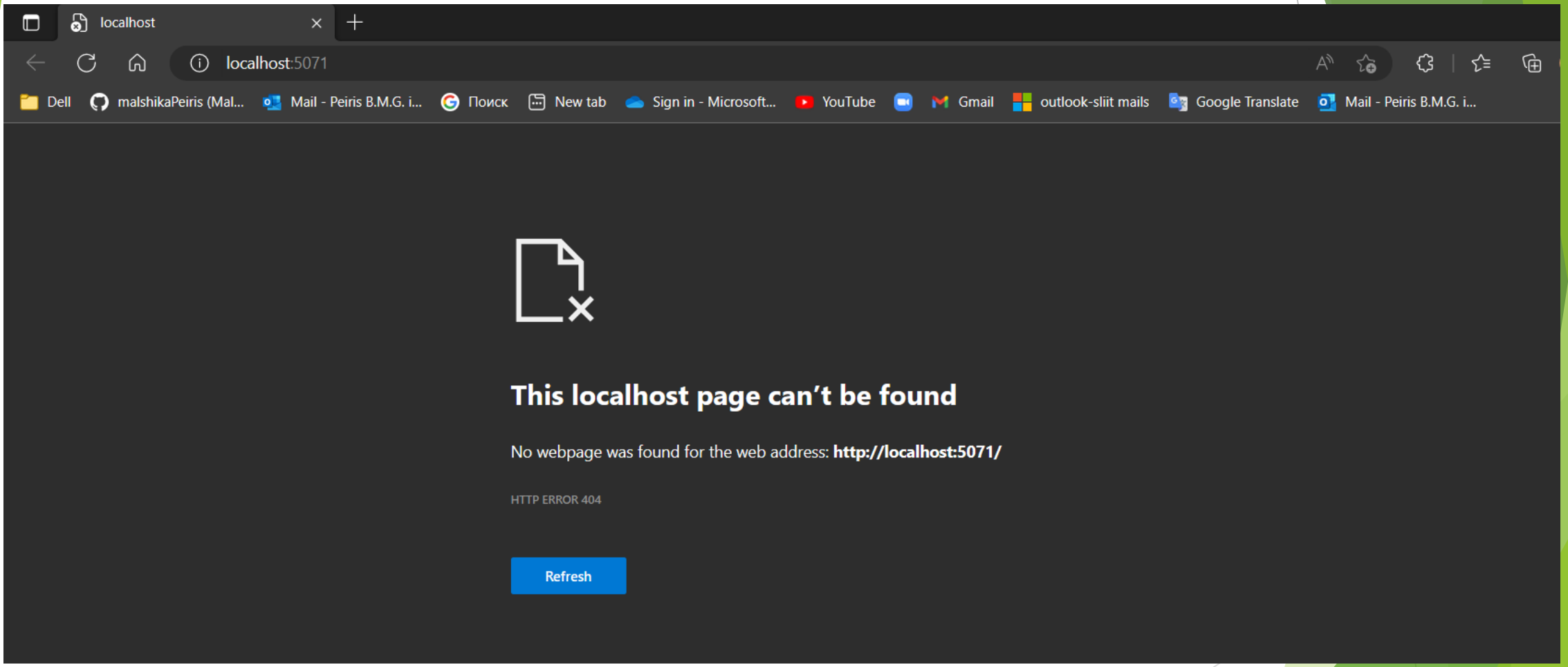
```
http://localhost:5003/api/Storekeeper/6
```

Server response

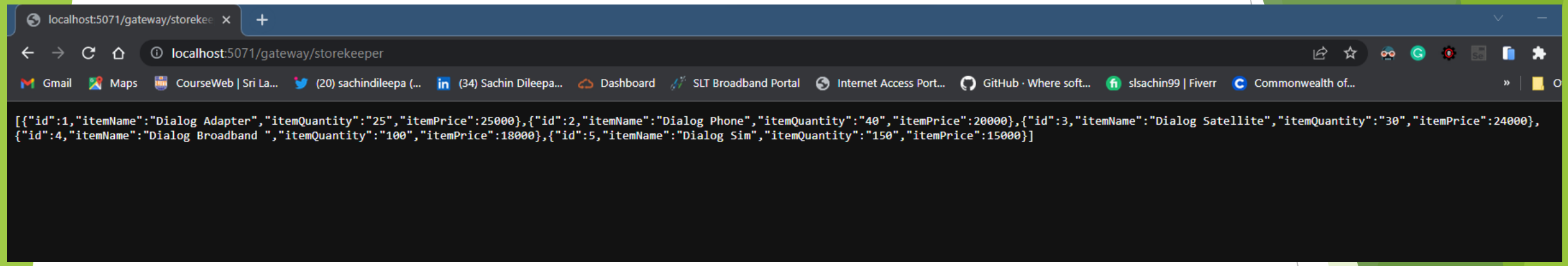
Code	Details
200	Response body `storekeeper with ID:6 got deleted successfully.` Download

Gateway project and the API project will both run and open up in two separate browsers.

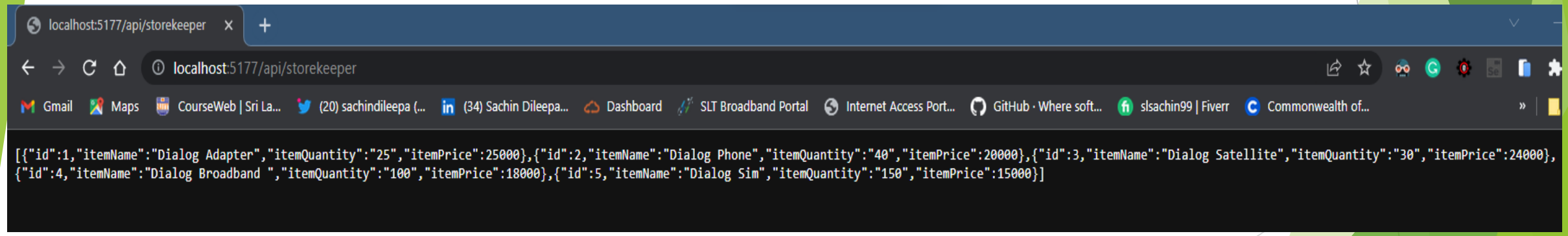
Gateway:



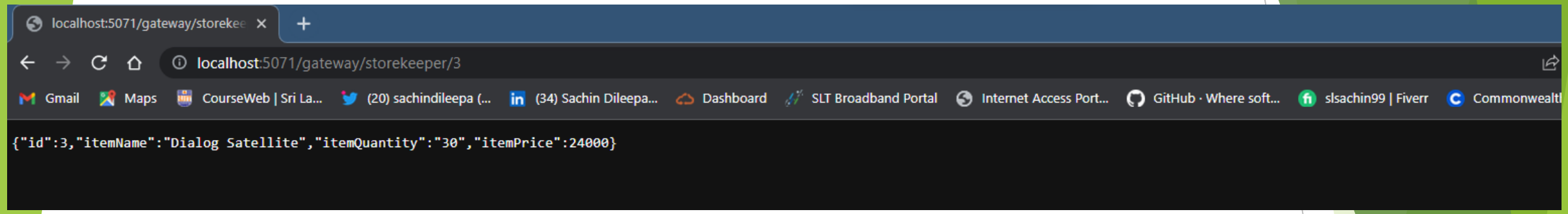
Test the upstream URL in browser.



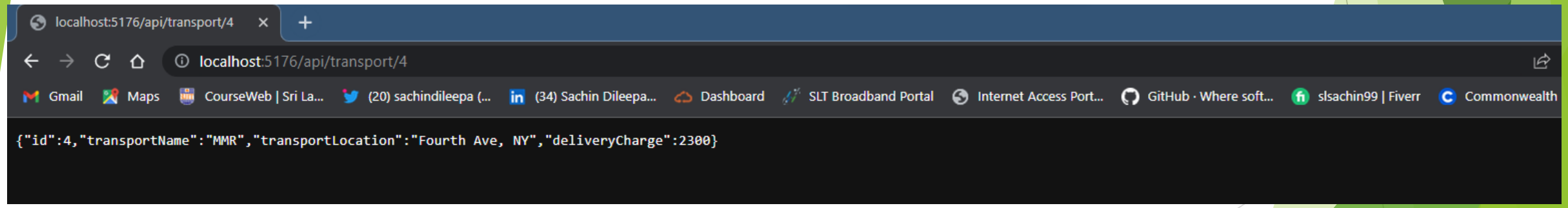
Test the downstream URL in browser.



Test the upstream URL in browser.



Test the downstream URL in browser.



IT20081416

Ahamed M.M.Z

Transport Service

Transpoart Service

Elaborate how you can avoid having multiple ports with the gateway you implemented and Solution should have proper folder structure with minimum two microservices and an Ocelot API Gateway.

launchsettings.json in Transpoart service

ocelot.json | launchSettings.json | launchSettings.json | MTIT.Ocelote | launchSettings.json

Schema: https://json.schemastore.org/launchsettings.json

```
1  {
2    "$schema": "https://json.schemastore.org/launchsettings.json",
3    "iisSettings": {
4      "windowsAuthentication": false,
5      "anonymousAuthentication": true,
6      "iisExpress": {
7        "applicationUrl": "http://localhost:40573",
8        "sslPort": 0
9      }
10   },
11   "profiles": {
12     "MTIT.Transport": {
13       "commandName": "Project",
14       "dotnetRunMessages": true,
15       "launchBrowser": true,
16       "launchUrl": "swagger",
17       "applicationUrl": "http://localhost:5176",
18       "environmentVariables": {
19         "ASPNETCORE_ENVIRONMENT": "Development"
20       }
21     },
22     "IIS Express": {
23       "commandName": "IISExpress",
24       "launchBrowser": true,
25       "launchUrl": "swagger",
26       "environmentVariables": {
27         "ASPNETCORE_ENVIRONMENT": "Development"
28       }
29     }
30   }
31 }
32
```

Solution Explorer

Search Solution Explorer (Ctrl+;)

- Solution 'MTIT.Supplier' (5 of 5 projects)
  - Gateway
    - MTIT.Ocelote
      - Connected Services
      - Dependencies
      - Properties
      - appsettings.json
      - ocelot.json
      - Program.cs
  - Microservices
    - MTIT.Manager
    - MTIT.Storekeeper
    - MTIT.Supplier.Mobile
    - MTIT.Transport
      - Connected Services
      - Dependencies
      - Properties
        - launchSettings.json
      - Controllers
      - Data
      - Models
      - Services
      - appsettings.json
      - Program.cs

Ocelot.json in Transpoart service

launchSettings.json | ocelot.json | launchSettings.json | launchSettings.json | MTIT.Ocelote

Schema: https://json.schemastore.org/ocelot.json

```
98                "Port": 5109
99
100             }
101
102           ]
103
104         },
105         {
106           "UpstreamPathTemplate": "/gateway/transport",
107           "UpstreamHttpMethod": [ "Get", "Post", "Put" ],
108           "DownstreamPathTemplate": "/api/transport",
109           "DownstreamScheme": "http",
110           "DownstreamHostAndPorts": [
111             {
112               "Host": "localhost",
113               "Port": 5176
114             }
115           ]
116         },
117         {
118           "UpstreamPathTemplate": "/gateway/transport/{Id}",
119           "UpstreamHttpMethod": [ "Get", "Delete" ],
120           "DownstreamPathTemplate": "/api/transport/{Id}",
121           "DownstreamScheme": "http",
122           "DownstreamHostAndPorts": [
123             {
124               "Host": "localhost",
125               "Port": 5176
126             }
127           ]
128         },
129         {
130         {
131
```

100 % | No issues found | Ln: 158 | Ch: 45 | SPC | CRLF

Solution Explorer

Search Solution Explorer (Ctrl+;)

- Solution 'MTIT.Supplier' (5 of 5 projects)
  - Gateway
    - MTIT.Ocelote
      - Connected Services
      - Dependencies
      - Properties
      - appsettings.json
      - ocelot.json
      - Program.cs
  - Microservices
    - MTIT.Manager
    - MTIT.Storekeeper
    - MTIT.Supplier.Mobile
    - MTIT.Transport

Screen captures OF Transpoart API

Swagger UI

localhost:5176/swagger/index.html

Swagger
Supported by SMARTBEAR

Select a definition MTIT.Assignment3.Transport v1

MTIT.Assignment3.Transport 1.0 OAS3

http://localhost:5176/swagger/v1/swagger.json

Transport

GET /api/Transport

POST /api/Transport

PUT /api/Transport

GET /api/Transport/{id}

DELETE /api/Transport/{id}

Schemas

Transport ›

GET Method

Swagger UI

localhost:5176/swagger/index.html 80%

GET /api/Transport

Parameters

Cancel

No parameters

Execute Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:5176/api/Transport' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:5176/api/Transport
```

Server response

Code	Details
200	Response body


```
[
  {
    "id": 1,
    "transportName": "LLC",
    "transportLocation": "Main St, California",
    "deliveryCharge": 2000
  },
  {
    "id": 2,
    "transportName": "JBC",
    "transportLocation": "Second Ave, LA",
    "deliveryCharge": 2200
  },
  {
    "id": 3,
    "transportName": "HCB",
    "transportLocation": "Third St, Vegas",
    "deliveryCharge": 2100
  },
  {
    "id": 4,
    "transportName": "MMR",
    "transportLocation": "Fourth Ave, NY",
    "deliveryCharge": 2300
```

POST Method

Swagger UI

localhost:5176/swagger/index.html

POST /api/Transport

Parameters

Cancel Reset

No parameters

Request body

application/json

```
{
  "id": 6,
  "transportName": "New Transport",
  "transportLocation": "Sixth lane, NJ",
  "deliveryCharge": 5600
}
```

Execute Clear

Responses

Curl

PUT Method

Swagger UI

localhost:5176/swagger/index.html

PUT /api/Transport

Parameters

Cancel

Reset

No parameters

Request body

application/json

```
{
  "id": 1,
  "transportName": "MSC",
  "transportLocation": "New lane, LA",
  "deliveryCharge": 620
}
```

Execute

Clear

Responses

Search

GET /api/Transport/{id}

Parameters

Cancel

Name	Description
id * required integer($int32) (path)	2

Execute | Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:5176/api/Transport/2' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:5176/api/Transport/2
```

Server response

Code	Details
200	Response body

```
{
  "id": 2,
  "transportName": "JBC",
  "transportLocation": "Second Ave, LA",
  "deliveryCharge": 2200
}
```

Download

DELETE Method

Swagger UI

localhost:5176/swagger/index.html

DELETE /api/Transport/{id}

Parameters

Cancel

Name	Description
id * required integer($int32) (path)	1

Execute | Clear

Swagger UI

localhost:5176/swagger/index.html

Responses

Curl

```
curl -X 'DELETE' \
  'http://localhost:5176/api/Transport/1' \
  -H 'accept: */*'
```

Request URL

http://localhost:5176/api/Transport/1

Server response

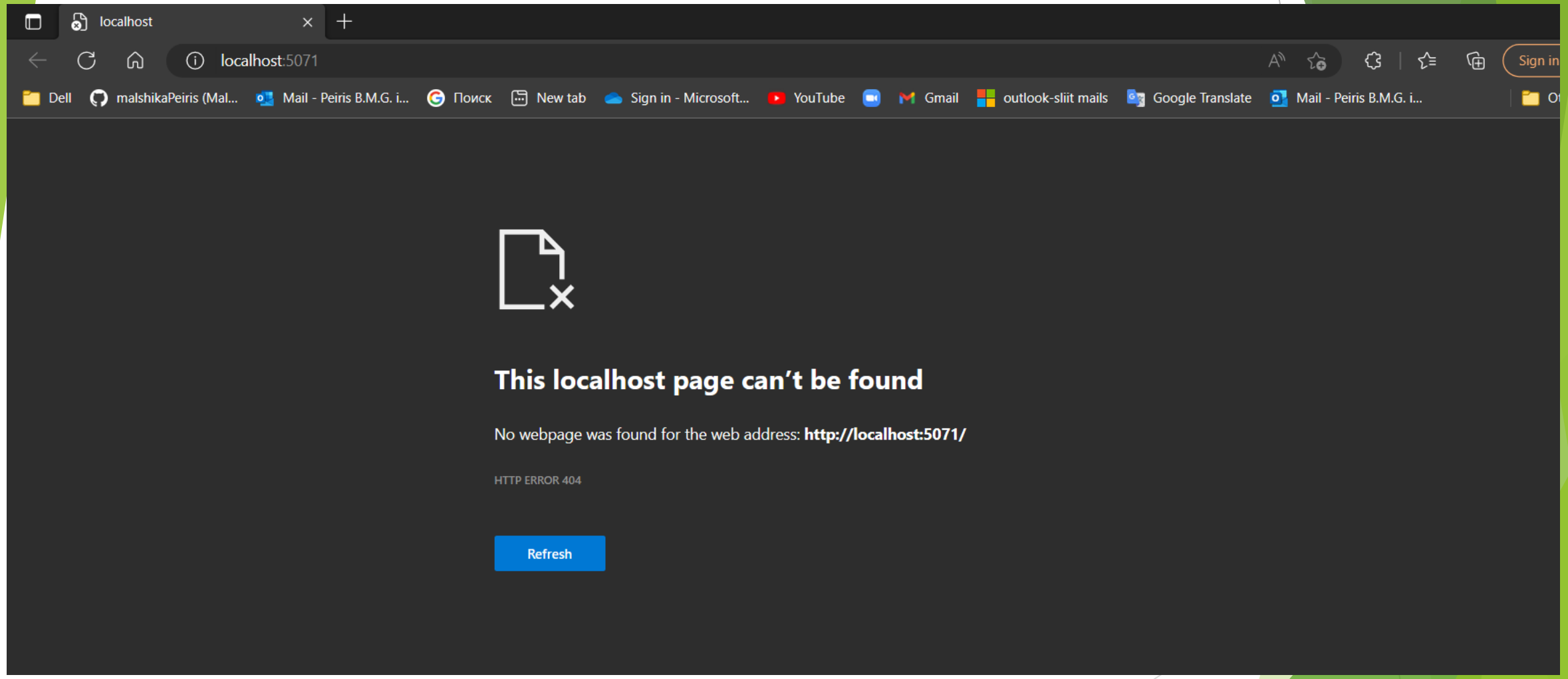
Code	Details
200	Response body Transport ID: 1 got deleted successfully. Download Response headers content-type: text/plain; charset=utf-8 date: Sat,15 Apr 2023 09:48:43 GMT server: Kestrel transfer-encoding: chunked

Responses

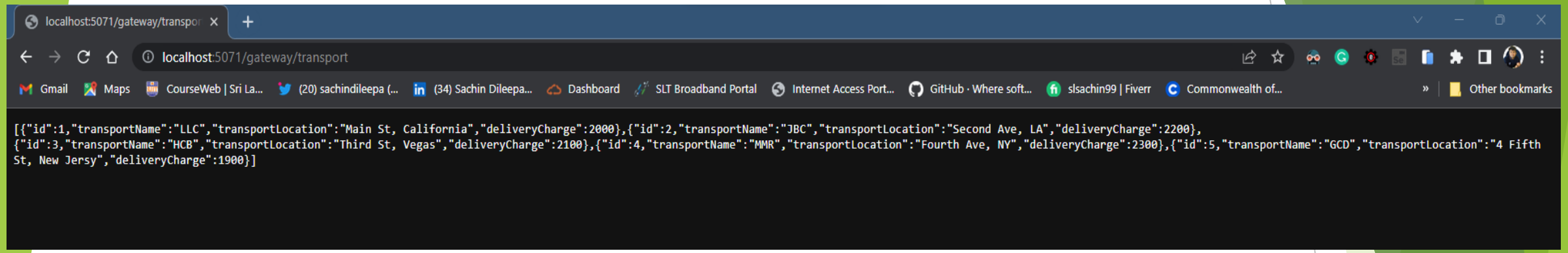
Code	Description	Links
200	Success	No links

Gateway project and the API project will both run and open up in two separate browsers.

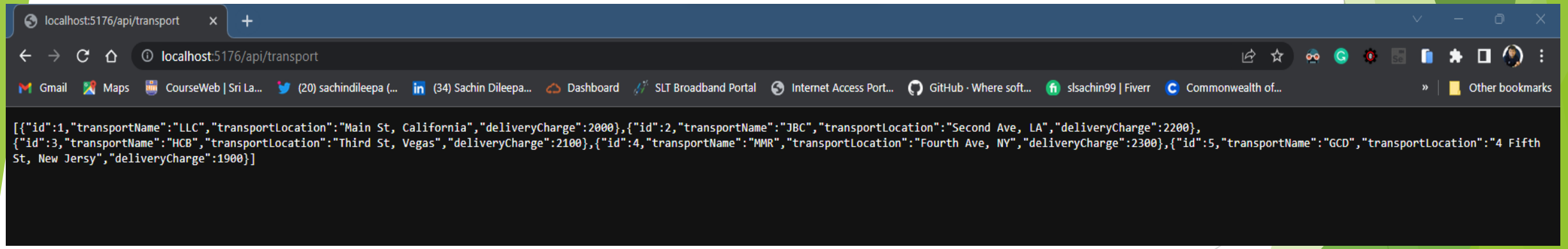
Gateway:



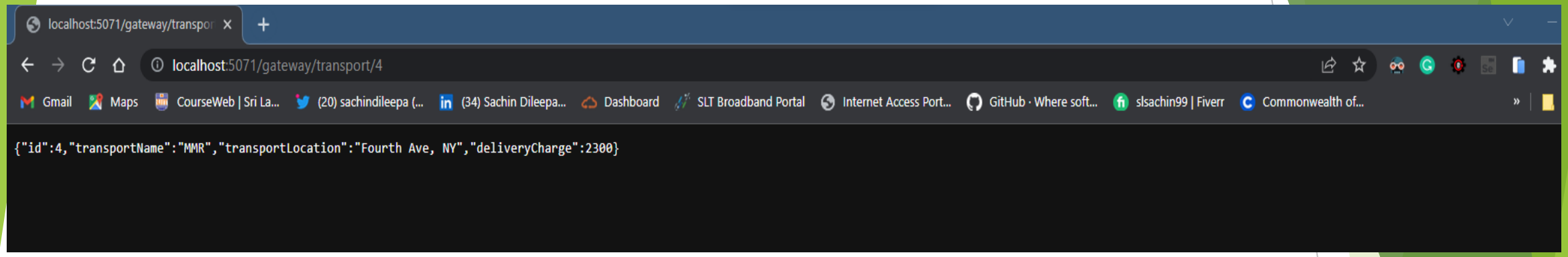
Test the upstream URL in browser.



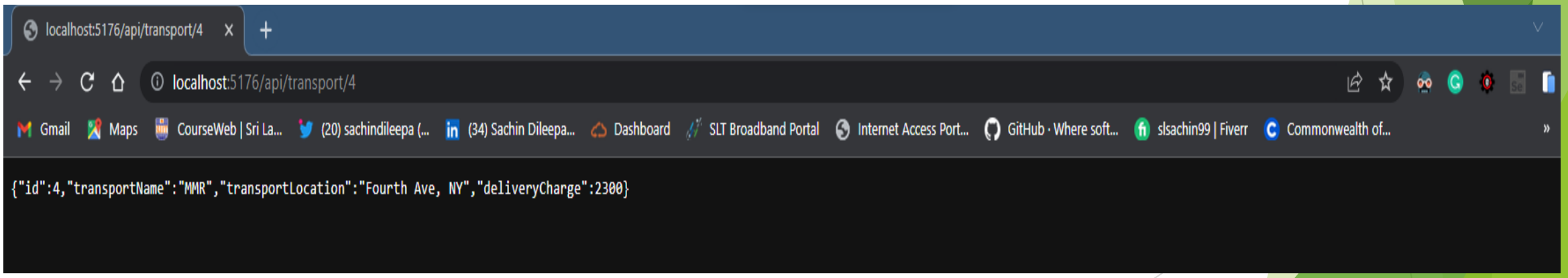
Test the downstream URL in browser.



Test the upstream URL in browser.



Test the downstream URL in browser.



REFERENCES

ASP.NET Core Web API CRUD With Entity Framework - Full Course = https://youtu.be/3NWT9k-6xGg

ASP.NET Core Web API + Entity Framework Core : Database First - EP01= https://youtu.be/CLVJVA9cTuU

ASP.NET Core Web API + Entity Framework Core : REST API Methods - EP03 = https://youtu.be/uPJ2-6YPRjM

ASP.NET CORE Crud operation using Entity framework using swagger= https://youtu.be/RsaUVrg9aNo