

FoodController.cs

```
using FoodOrderApi.Services;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using FoodOrderApi.Models;

namespace FoodOrderApi.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class FoodController : ControllerBase
    {
        private readonly DataStore _store;
        public FoodController(DataStore store) {
            _store = store;
        }

        [HttpGet]

        public ActionResult<IEnumerable<FoodItem>> GetAllFoods()
        {
            return Ok(_store.FoodItems);
        }
    }
}
```

OrdersController.cs

```
using FoodOrderApi.Services;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using FoodOrderApi.Models;

namespace FoodOrderApi.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class OrdersController : ControllerBase
    {
        private readonly DataStore _store;
        public OrdersController(DataStore store)
        {
            _store = store;
        }

        [HttpPost]
        public ActionResult<Order> PlaceOrder(Order order)
        {
            order.Id = _store.Orders.Count + 1;
            _store.Orders.Add(order);
            return CreatedAtAction(nameof(GetOrder), new { id = order.Id }, order);
        }

        [HttpGet]
        public ActionResult<IEnumerable<Order>> GetAllOrders()
        {
            return Ok(_store.Orders);
        }

        [HttpGet("{id}")]
        public ActionResult<Order> GetOrder(int id)
        {
            var order = _store.Orders.FirstOrDefault(o => o.Id == id);
            return order == null ? NotFound() : Ok(order);
        }

        [HttpPut("{id}")]
        public IActionResult UpdateOrder(int id, Order updatedOrder)
        {
            var order = _store.Orders.FirstOrDefault(o => o.Id == id);
            if (order == null) return NotFound();

            order.CustomerName = updatedOrder.CustomerName;
            order.FoodItemIds = updatedOrder.FoodItemIds;
            return NoContent();
        }

        [HttpDelete("{id}")]
        public IActionResult DeleteOrder (int id)
```

```
    {
        var order = _store.Orders.FirstOrDefault(o => o.Id == id);
        if(order == null) return NotFound();

        _store.Orders.Remove(order);
        return NoContent();
    }
}
```

FoodItem.cs

```
namespace FoodOrderApi.Models
{
    public class FoodItem
    {
        public int Id { get; set; }
        public string Name { get; set; } = string.Empty;
        public decimal Price { get; set; }
    }
}
```

Order.cs

```
namespace FoodOrderApi.Models
{
    public class Order
    {
        public int Id { get; set; }
        public string CustomerName { get; set; } = string.Empty;
        public List<int> FoodItemIds { get; set; } = new();
        public DateTime OrderTime { get; set; } = DateTime.Now;
    }
}
```

DataStore.cs

```
using FoodOrderApi.Models;

namespace FoodOrderApi.Services
{
    public class DataStore
    {
        public List<FoodItem> FoodItems { get; set; } = new()
        {
            new FoodItem {Id = 1, Name = "Burger", Price = 99},
            new FoodItem {Id = 2, Name = "Pizza", Price = 199},
            new FoodItem {Id = 3, Name = "Fries", Price = 49},
            new FoodItem {Id = 2, Name = "Cola", Price = 30},
            new FoodItem {Id = 2, Name = "Ranch", Price = 20}
        };

        public List<Order> Orders { get; set; } = new();
    }
}
```

Program.cs

```
var builder = WebApplication.CreateBuilder(args);

// Add services to the container.


builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at
https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();
builder.Services.AddSingleton<FoodOrderApi.Services.DataStore>();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseHttpsRedirection();
app.UseAuthorization();
app.MapControllers();
app.Run();
```

OUTPUT

 Swagger
powered by SMARTSCAN

Select a definition **FoodOrderApi v1**

FoodOrderApi

1.0 OAS 3.0

<https://localhost:7008/swagger/v1/swagger.json>

Food

GET /api/Food

Orders

POST /api/Orders

GET /api/Orders

GET /api/Orders/{id}

PUT /api/Orders/{id}

DELETE /api/Orders/{id}

Schemas

FoodItem {
 id > [...]
 name > [...]
 price > [...]
}

Order {
 id > [...]
 customerName > [...]
 foodItemIds > [...]
 orderTime > [...]
}

GET/api/Food

Food

GET /api/Food

Parameters

No parameters

Execute Clear

Responses

Curl

```
curl -X 'GET' \
  'https://localhost:7000/api/Food' \
  -H 'accept: text/plain'
```

Request URL

```
https://localhost:7000/api/Food
```

Server response

Code

Details

200

Response body

```
{
  "id": 1,
  "name": "Burger",
  "price": 99
},
{
  "id": 2,
  "name": "Pizza",
  "price": 199
},
{
  "id": 3,
  "name": "Fries",
  "price": 49
},
{
  "id": 3,
  "name": "Cola",
  "price": 39
},
{
  "id": 3,
  "name": "Ranch",
  "price": 29
}
]
```

Response headers

```
content-type: application/json; charset=utf-8
date: Fri, 08 Aug 2025 15:23:22 GMT
server: Kestrel
```

Responses

Code

Description

Links

POST/api/Orders

Orders

POST /api/Orders

Parameters

No parameters

Request body

application/json

```
{  "id": 0,  "customerName": "Aki",  "foodItemId": {    "1,2"  },  "orderTime": "2025-08-08T15:24:12.252Z"}
```

Execute

Clear

Responses

Curl

```
curl -X 'POST' \  'https://localhost:7000/api/Orders' \  -H 'accept: text/plain' \  -H 'Content-Type: application/json' \  -d '{  "id": 0,  "customerName": "Aki",  "foodItemId": {    "1,2"  },  "orderTime": "2025-08-08T15:24:12.252Z"  }'
```

Request URL

https://localhost:7000/api/Orders

Server response

Code	Details
201	<div>Undocumented</div> <div>Response body</div> <pre>{ "id": 1, "customerName": "Aki", "foodItemId": { "1,2" }, "orderTime": "2025-08-08T15:24:12.252Z"}</pre> <div>Download</div> <div>Response headers</div> <pre>content-type: application/json; charset=utf-8 date: Fri, 08 Aug 2025 15:24:28 GMT location: https://localhost:7000/api/Orders/1 server: Kestrel</pre>

Responses

Media type

text/plain

Controls Accept header.

Example Value | Schema

SECOND POST

The screenshot displays a REST client interface with a light green header and a dark green sidebar on the right. The main area is divided into several sections:

- Request Body:** A large text area containing a JSON object:

```
{  "id": 0,  "customerName": "John",  "foodItemIds": [    1,2,3,4,5  ],  "orderTime": "2025-08-08T15:24:12.252Z"}
```
- Buttons:** Below the request body are two buttons: "Execute" (blue) and "Clear" (white with a grey border).
- Responses:** A section titled "Responses" with a light green background.
- Curl:** A dark grey text area showing the equivalent curl command:

```
curl -X 'POST' \  'https://localhost:7008/api/Orders' \  -H 'accept: text/plain' \  -H 'Content-Type: application/json' \  -d '{  "id": 0,  "customerName": "John",  "foodItemIds": [    1,2,3,4,5  ],  "orderTime": "2025-08-08T15:24:12.252Z"  }'
```
- Request URL:** A dark grey text area containing the URL: `https://localhost:7008/api/Orders`
- Server response:** A section titled "Server response" with a light green background.
- Code:** A tab labeled "Code" is selected, showing the response body in a dark grey text area:

```
{  "id": 2,  "customerName": "John",  "foodItemIds": [    1,    2,    3,    4,    5  ],  "orderTime": "2025-08-08T15:24:12.252Z"}
```
- Response headers:** A section titled "Response headers" is visible at the bottom.

GET/api/Orders

GET

/api/Orders

^

Parameters

Cancel

No parameters

Execute

Clear

Responses

Curl

```
curl -X 'GET' \
'https://localhost:7000/api/Orders' \
-H 'accept: text/plain'
```

Request URL

```
https://localhost:7000/api/Orders
```

Server response

Code

Details

200

Response body

```
{
  "id": 1,
  "customerName": "Ak1",
  "foodItemIds": [
    1,
    2
  ],
  "orderTime": "2025-08-08T15:24:12.252Z"
},
{
  "id": 2,
  "customerName": "John",
  "foodItemIds": [
    1,
    2,
    3,
    4,
    5
  ],
  "orderTime": "2025-08-08T15:24:12.252Z"
}
]
```

Download

Response headers

```
content-type: application/json; charset=utf-8
date: Fri, 08 Aug 2025 15:26:48 GMT
server: Kestrel
```

Responses

GET IDS

GET

/api/Orders/{id}

^

Parameters

Cancel

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

ExecuteClear

Responses

Curl

```
curl -X 'GET' \
'https://localhost:7008/api/Orders/1' \
-H 'accept: text/plain'
```

Request URL

https://localhost:7008/api/Orders/1

Server response

Code	Details
200	<div><div>Response body</div><div><pre>{ "id": 1, "customerName": "Ak1", "foodItemIds": [1, 2], "orderTime": "2025-08-08T15:24:12.252Z" }</pre></div><div>Download</div></div> <div><div>Response headers</div><div><pre>content-type: application/json; charset=utf-8 date: Fri, 08 Aug 2025 15:28:21 GMT server: Kestrel</pre></div></div>

Responses

GET

/api/Orders/{id}

^

Parameters

Cancel

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

ExecuteClear

Responses

Curl

```
curl -X 'GET' \
'https://localhost:7008/api/Orders/2' \
-H 'accept: text/plain'
```

Request URL

https://localhost:7008/api/Orders/2

Server response

Code	Details
200	<div><div>Response body</div><div><pre>{ "id": 2, "customerName": "John", "foodItemIds": [1, 2, 3, 4, 5], "orderTime": "2025-08-08T15:24:12.252Z" }</pre></div><div>Download</div></div> <div><div>Response headers</div><div><pre>content-type: application/json; charset=utf-8 date: Fri, 08 Aug 2025 15:28:33 GMT server: Kestrel</pre></div></div>

Responses

John Rich Nicolas
DAY 9

FRONTEND
API

PUT

PUT

/api/Orders/{id}

Parameters

Cancel

Reset

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

Request body

application/json

```
{
  "id": 0,
  "customerName": "Nicolas, John Rich",
  "foodItemIds": [
    1,2,3
  ],
  "orderTime": "2025-08-08T15:28:56.649Z"
}
```

Execute

Clear

Responses

Curl

```
curl -X 'PUT' \
  'https://localhost:7008/api/Orders/2' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "id": 0,
    "customerName": "Nicolas, John Rich",
    "foodItemIds": [
      1,2,3
    ],
    "orderTime": "2025-08-08T15:28:56.649Z"
  }'
```

Request URL

https://localhost:7008/api/Orders/2

Server response

Code

Details

GET

/api/Orders/{id}

Parameters

Cancel

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

Execute

Clear

Responses

Curl

```
curl -X 'GET' \
  'https://localhost:7008/api/Orders/2' \
  -H 'accept: text/plain'
```

Request URL

https://localhost:7008/api/Orders/2

Server response

Code

Details

200

Response body

```
{
  "id": 2,
  "customerName": "Nicolas, John Rich",
  "foodItemIds": [
    1,
    2,
    3
  ],
  "orderTime": "2025-08-08T15:24:12.252Z"
}
```

Download

Response headers

```
content-type: application/json; charset=utf-8
date: Fri, 08 Aug 2025 15:29:47 GMT
server: Kestrel
```

Responses

Code

Description

Links

DELETE

DELETE

/api/Orders/{id}

Parameters

Cancel

Name	Description
id <small>required</small>	
integer(int32)	2
(path)	

ExecuteClear

Responses

Curl

```
curl -X 'DELETE' \
'https://localhost:7000/api/Orders/2' \
-H 'accept: */*'
```

Request URL

```
https://localhost:7000/api/Orders/2
```

Server response

Code

Details

SHOW ALL AGAIN

GET

/api/Orders

Parameters

Cancel

No parameters

ExecuteClear

Responses

Curl

```
curl -X 'GET' \
'https://localhost:7000/api/Orders' \
-H 'accept: text/plain'
```

Request URL

```
https://localhost:7000/api/Orders
```

Server response

Code

Details

200

Response body

```
{
  "id": 1,
  "customerName": "Aki",
  "foodItemIds": [
    1,
    2
  ],
  "orderTime": "2025-08-08T15:24:12.252Z"
}
```

Response headers

```
content-type: application/json; charset=utf-8
date: Fri, 08 Aug 2025 15:31:02 GMT
server: Kestrel
```

Responses

Code	Description	Links
------	-------------	-------

CHECK ID NG NADELETE

GET

/api/Orders/{id}

⌵

Parameters

Cancel

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

Execute

Clear

Responses

Curl

```
curl -X 'GET' \
'https://localhost:7000/api/Orders/2' \
-H 'accept: text/plain'
```

Request URL

```
https://localhost:7000/api/Orders/2
```

Server response

Code	Details
404	Error: response status is 404

Undocumented

Response body

```
{
  "type": "https://tools.ietf.org/html/rfc9110#section-15.5.5",
  "title": "Not Found",
  "status": 404,
  "traceId": "00-9eda31a3cf376d215e5a6cd79c4beccf-c5c4c9406ad5ae14-00"
}
```

Response headers

```
content-type: application/problem+json; charset=utf-8
date: Fri, 08 Aug 2025 15:31:17 GMT
server: Kestrel
```

Responses

Code

Description

Links

REFLECTION

In my Food Order API project, I followed the instructions given by our professor and used the provided files to build the system. The code was organized into different parts using functions that handled tasks like showing the menu, placing and updating orders, checking if the order is valid, and viewing or deleting existing orders. The FoodController is used to show the list of available food items, while the OrdersController handles everything related to customer orders. This made the code easier to read and understand because each part had its own role.

The API works by sending and receiving data through HTTP requests. I used GET requests to view the food list and orders, and POST requests to add new orders. The data is sent in JSON format and automatically turned into C# objects by the API. I tested the endpoints using Swagger, which made it easier to check if the requests were working and see the results without needing a separate website.

One thing I noticed is that whenever I restart the application, all the orders are gone. That's because the data is stored only in memory, and not saved permanently in a database. So every time the app runs again, it starts fresh.

A challenge I faced while doing this activity was getting build errors because of missing modules. The program showed CS0246 errors saying it couldn't find things like Order, FoodItem, and DataStore. I realized I forgot to add the correct using statements at the top of my files. Once I added lines like using FoodOrderApi.Models; and using FoodOrderApi.Services;, the errors were gone and the project worked properly. This taught me how important it is to double-check all references when working with multiple files.