**System Overview**

The GAC WMS Integration System provides real-time and file-based integration between ERP systems and GAC's Warehouse Management System It features as below:

- RESTful API endpoints for real-time integration

- File based processing for legacy systems

- Resilient architecture with retry mechanisms

- Comprehensive validation and error handling

**Project Structure:**

**GAC.WMS.Integration**

1. Application/ # Application layer services and DTOs

* Extensions/ # Extension methods
* Filters/ # Action filters
* Interfaces/ # Service contracts
* Middleware/ # Custom middleware
* Models/ # Request Dto’s
* Services/ # Application service
* Validators/ # Fluent Validation

1. Domain/ # Core domain models and business logic

* Entities/ # Domain entities
* Interfaces/ # Repository interfaces

1. Infrastructure/ # Infrastructure implementations

* Database context
* Repository/ # Repository implementations

1. FileProcessor/ # Background file processing service

* Services/ # File processing logic
* SchemaValidator

1. Tests/ # Unit and integration tests

* UnitTests/ # Unit tests

**Development:**

* Visual Studio 2022
* SSMS 20.

**Technical:**

* **ASP.NET Core**
* EntityFrameWorkCore
* **BackgroundService**
* **Dependency Injection**

**.Net Featues**

* **Polly Resilience-** Retry policies for file processing
* **XML Serialization – for xdocument** processing
* **FileSystemWatcher – for** File monitoring

**Architecture:**

The project follows a **Clean Architecture** with **Domain-Driven Design (DDD)** principles:

1. **Application Layer** (Business logic/services)
2. **Domain Layer** (Core models)
3. **Infrastructure Layer** (Persistence/implementations)
4. **FileProcessor** (BackGroundProcessor)

**Implementation Notes**

**Real-time Integration**:

* RESTful API endpoints for Customer, Product, Purchase Order, and Sales Order management
* Comprehensive validation and error handling
* Database persistence with Entity Framework Core
* Each request will get validated before it hit end point.

**File-based Integration**:

* Background service for polling configured directories
* XML file parsing and transformation
* Configurable via appsettings.json
* Automatic retry and error handling

**Resilience**:

* Retry policies with exponential backoff
* Circuit breaker pattern
* Timeout policies
* Comprehensive exception handling

**Extensibility**:

* Clean architecture with separation of concerns
* Dependency injection throughout
* Easy to add new file formats or integration points

**Testing**:

* Comprehensive unit tests for all services
* Mocked dependencies for isolated testing
* Test coverage for success and failure scenarios

**Local Testing:**

Step-1:

Before starting the application ensure the **Application** and **FileProcessor** projects are multiple startup projects configured or not if not then setup both the projects to start.

Check the **Application** and **FileProcessor** appsettings ConnectionStrings

Step-2 :

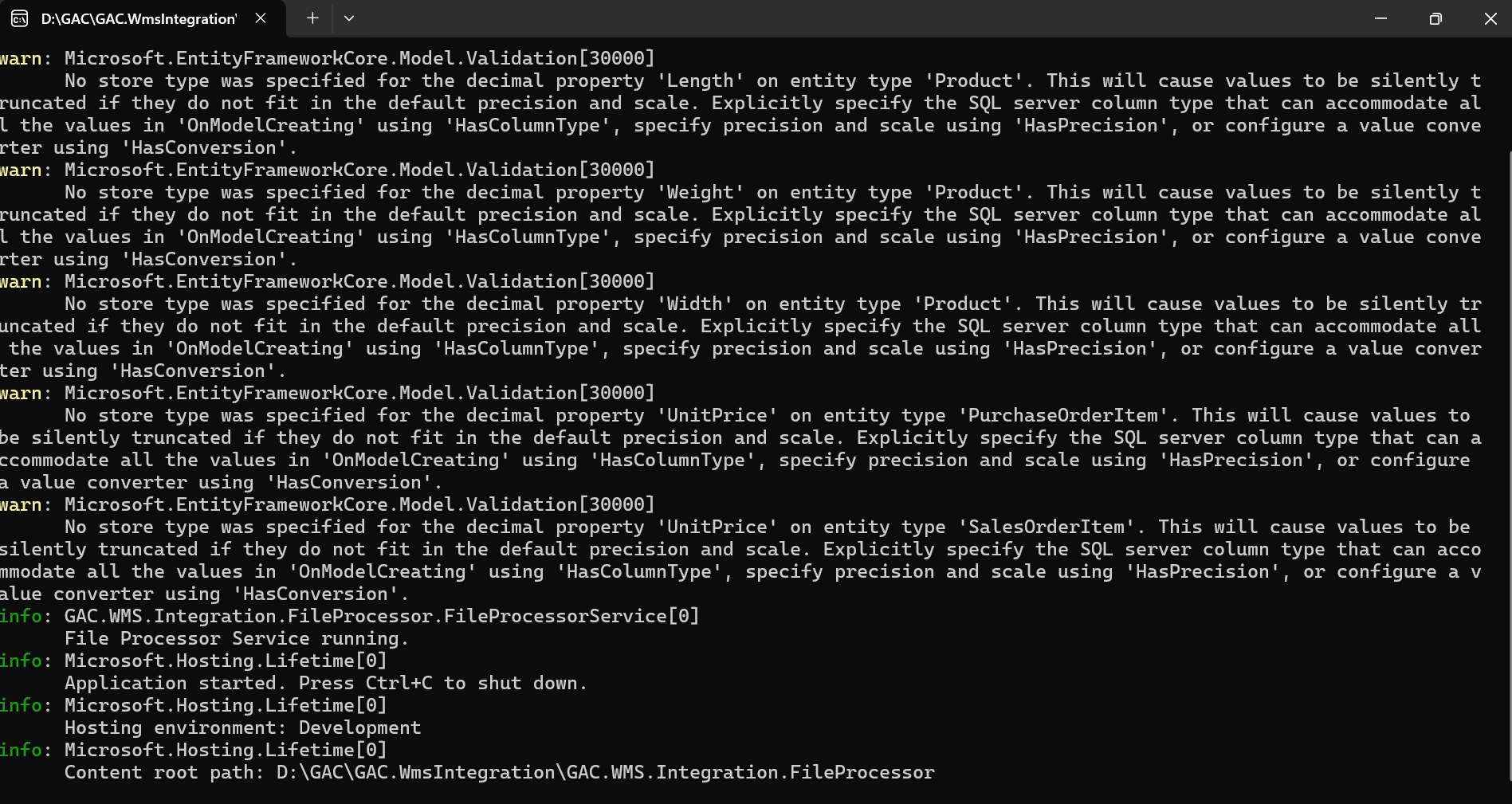
Run the project and check the swagger end points in the browser, and a console for the FileProcessor

**Note**: Currently I did not include any authentication so the end points can call directly without any authentication.

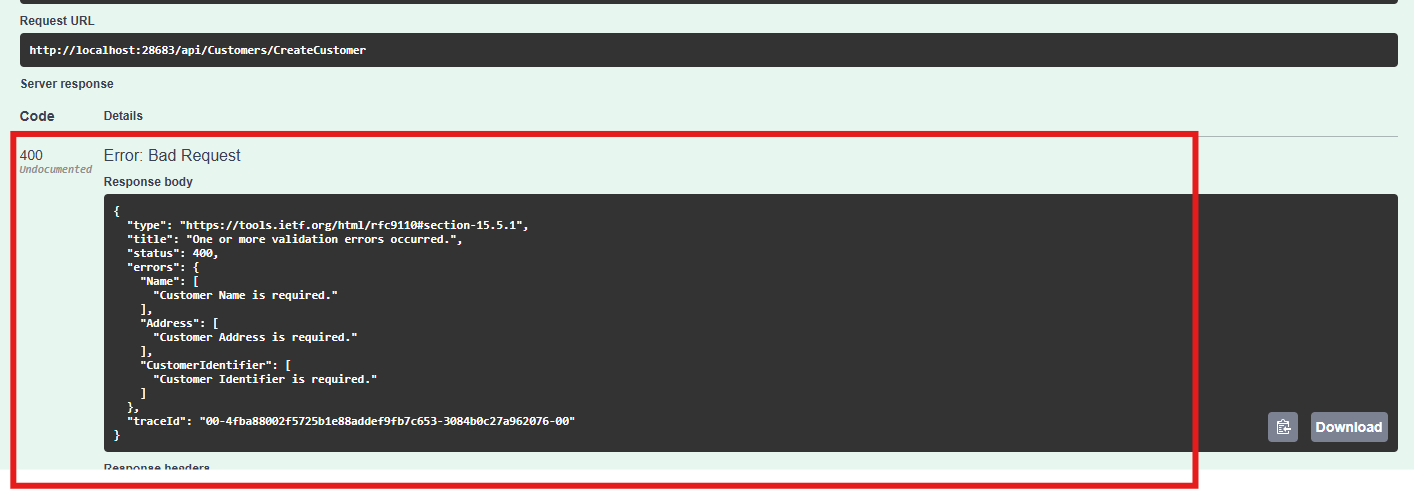
Application Sawgger :

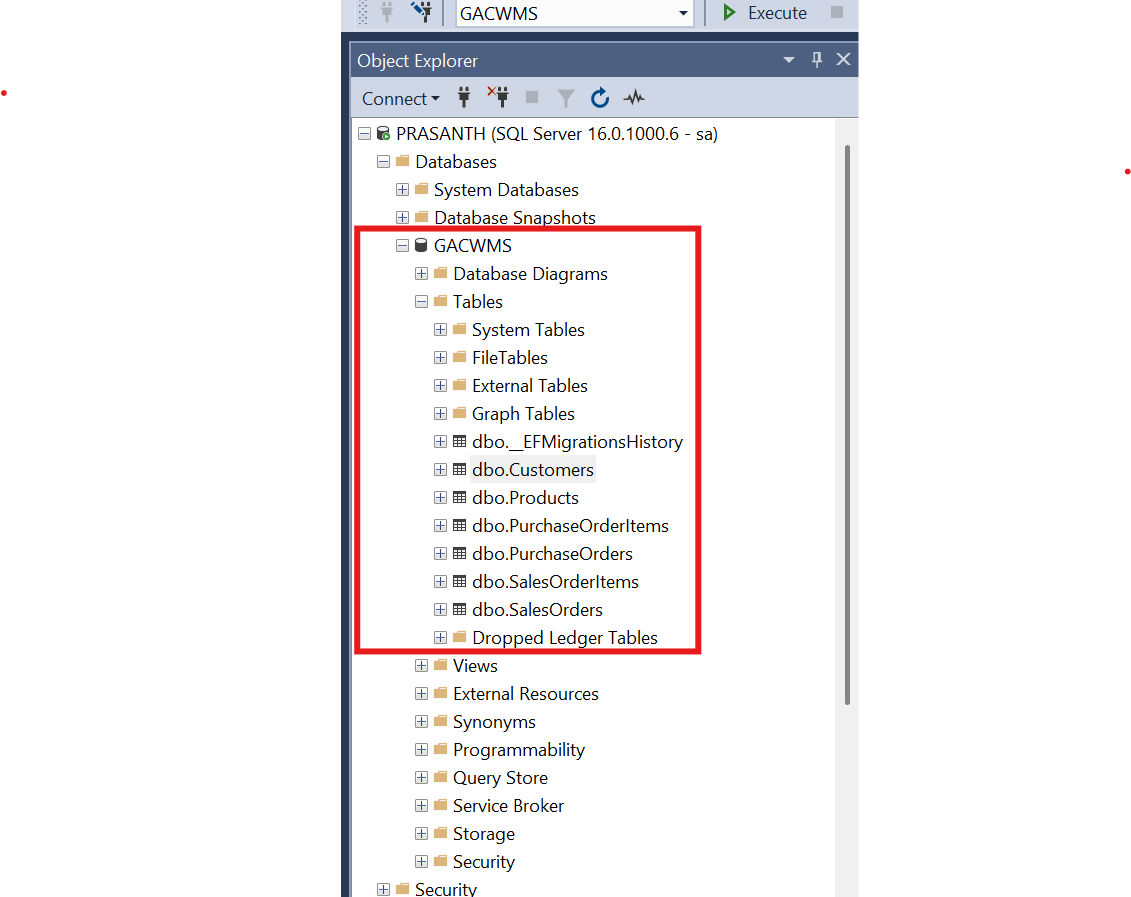


FileProcessor Console:



Added Fluent validations :



SQL DB:  
  


Test the Customer API end points

Create Customer :

Input:

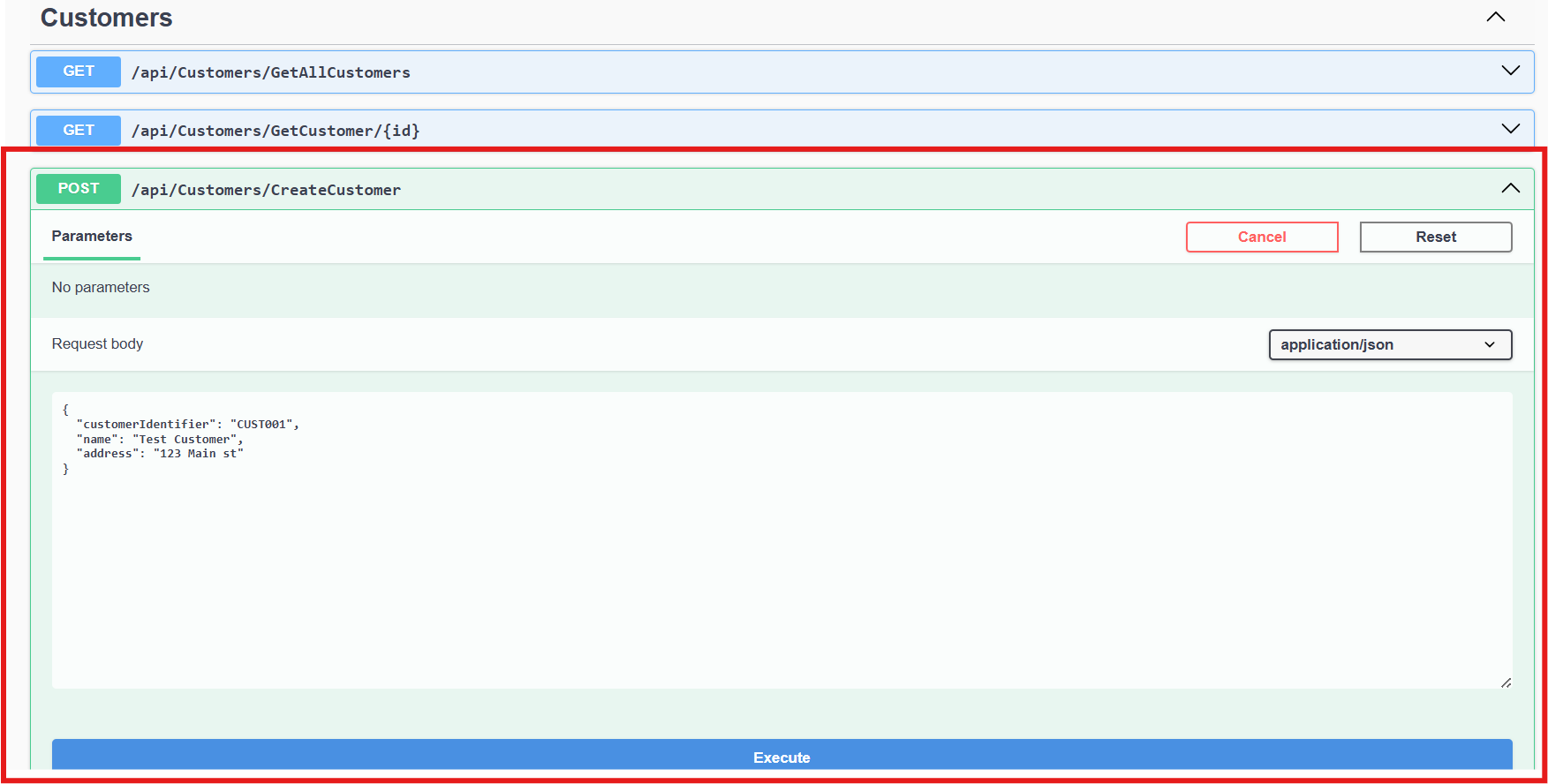
{

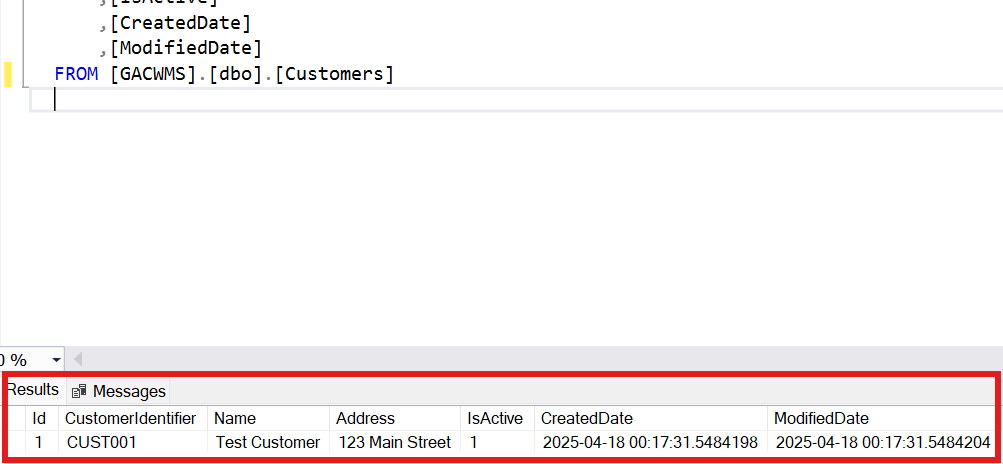
"customerIdentifier": "CUST001",

"name": "Test Customer",

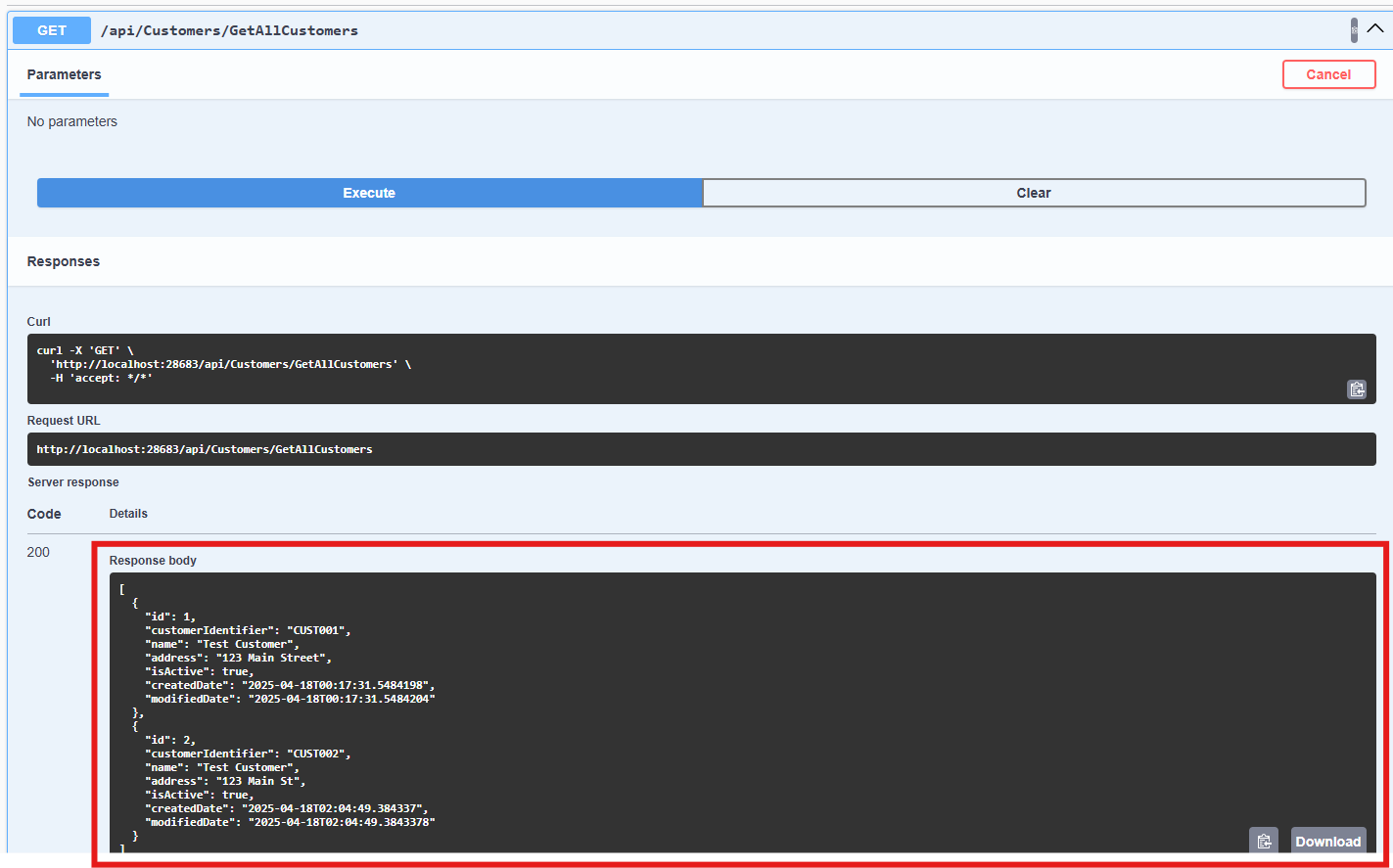
"address": "123 Main st"

}



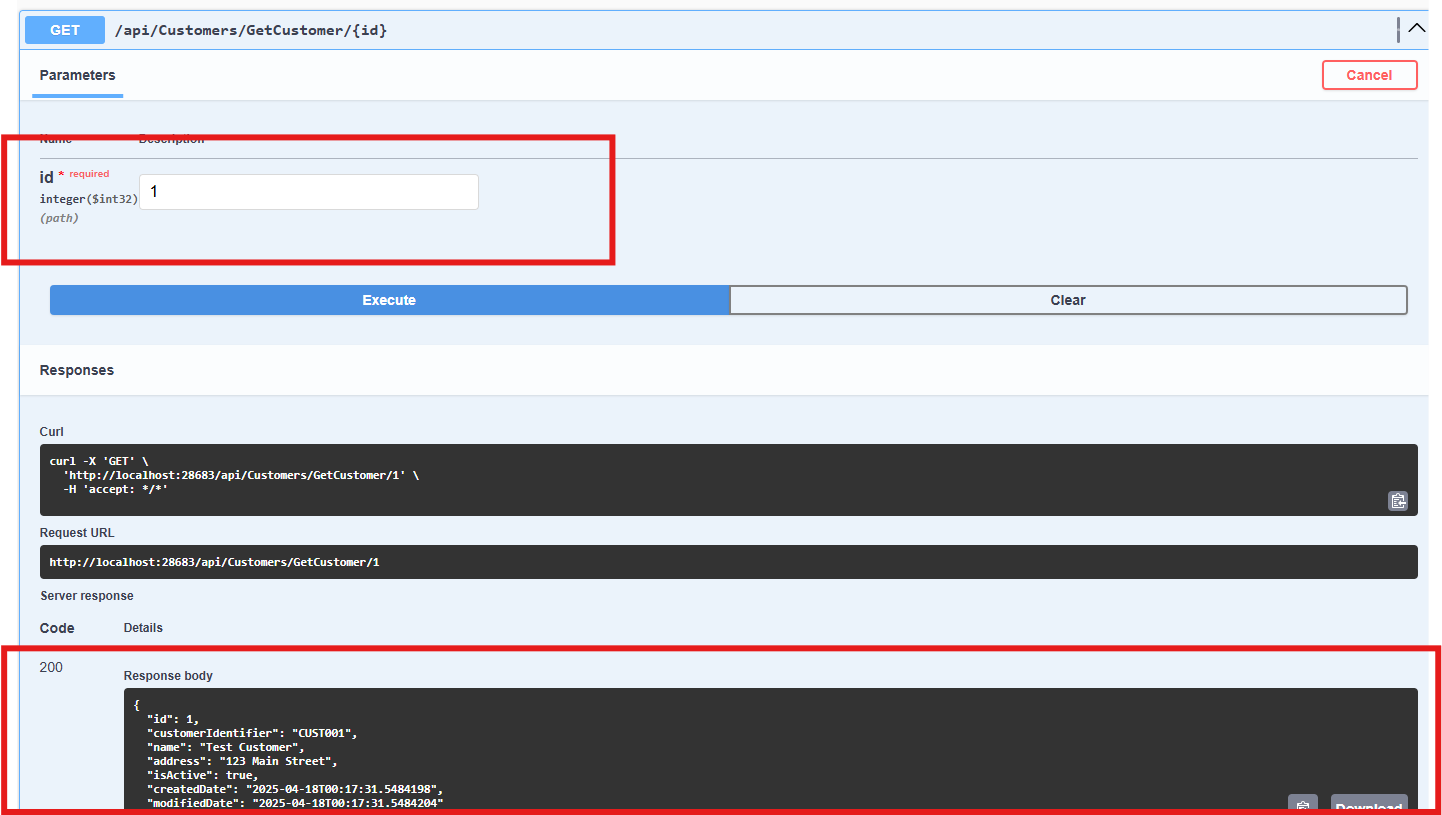


Get All Customers :



Get Customer By Id:

Input : 1



Update Customer by id :  
=> Updating customer name from Test Customer to Prasanth Kesana  
Input : Id : 1

{

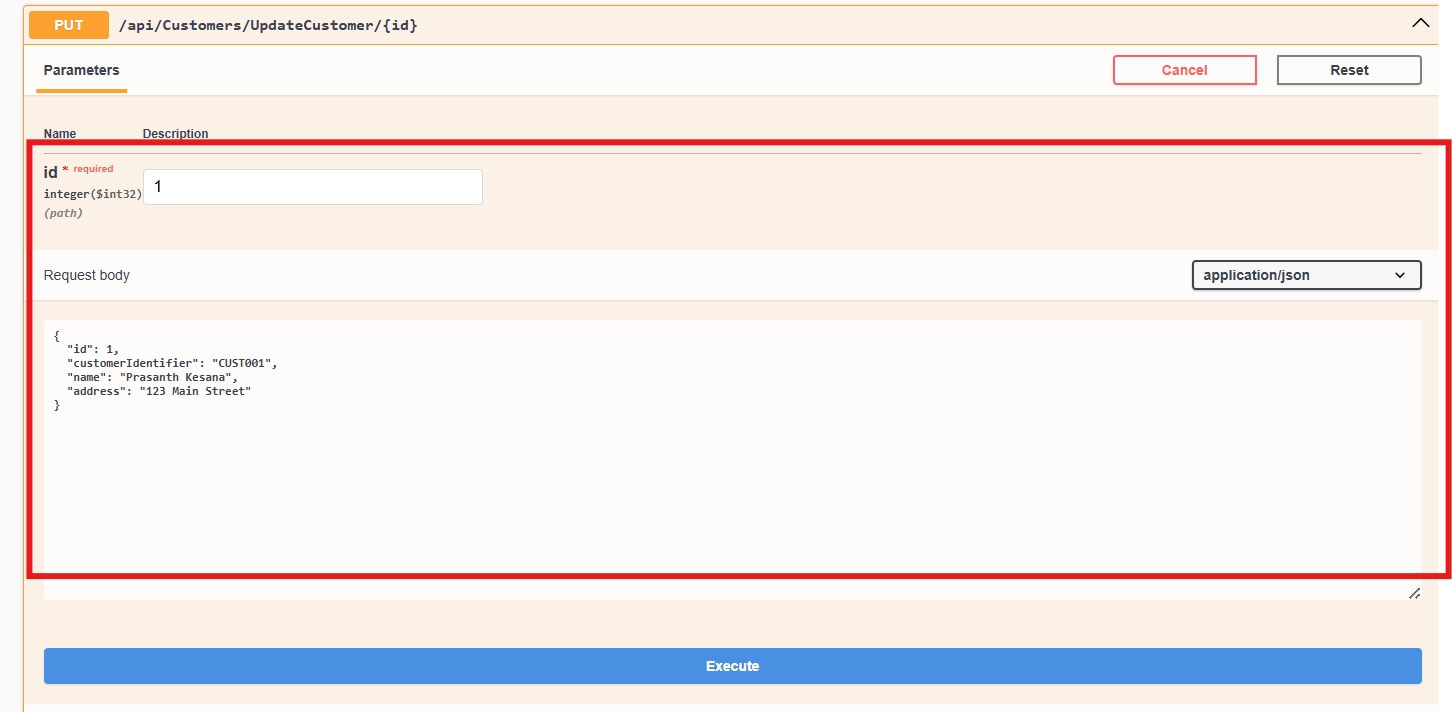
"id": 1,

"customerIdentifier": "CUST001",

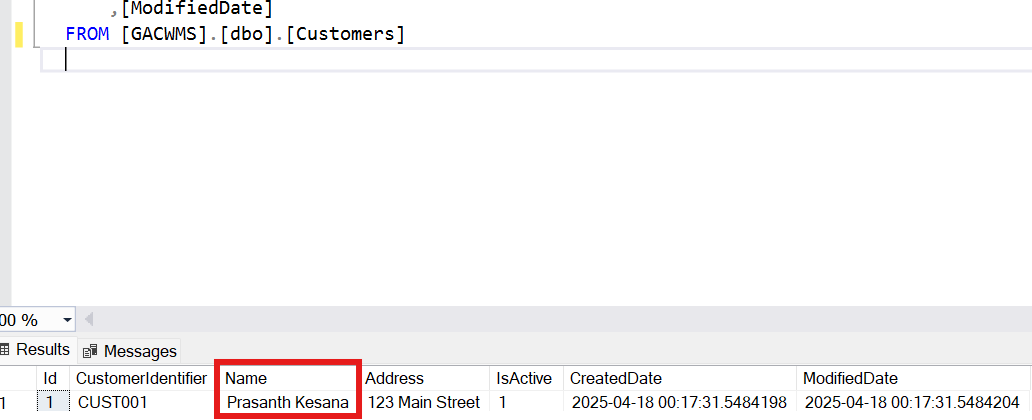
"name": "Prasanth Kesana",

"address": "123 Main Street"

}

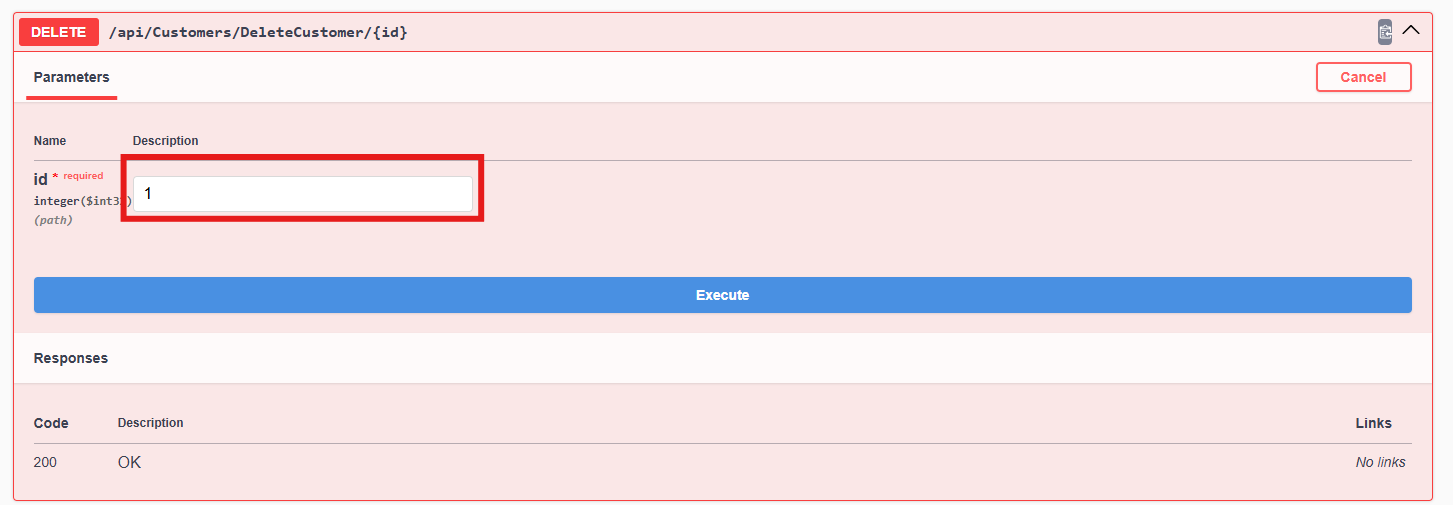


Output:



Delete Customer:

Input Id = 1 and execute



Output: record deleted from db table

Test the Products API end points:

Create Product:

Input:

{

"productCode": "P001",

"title": "Mobile",

"description": "test desc",

"length": 6.7,

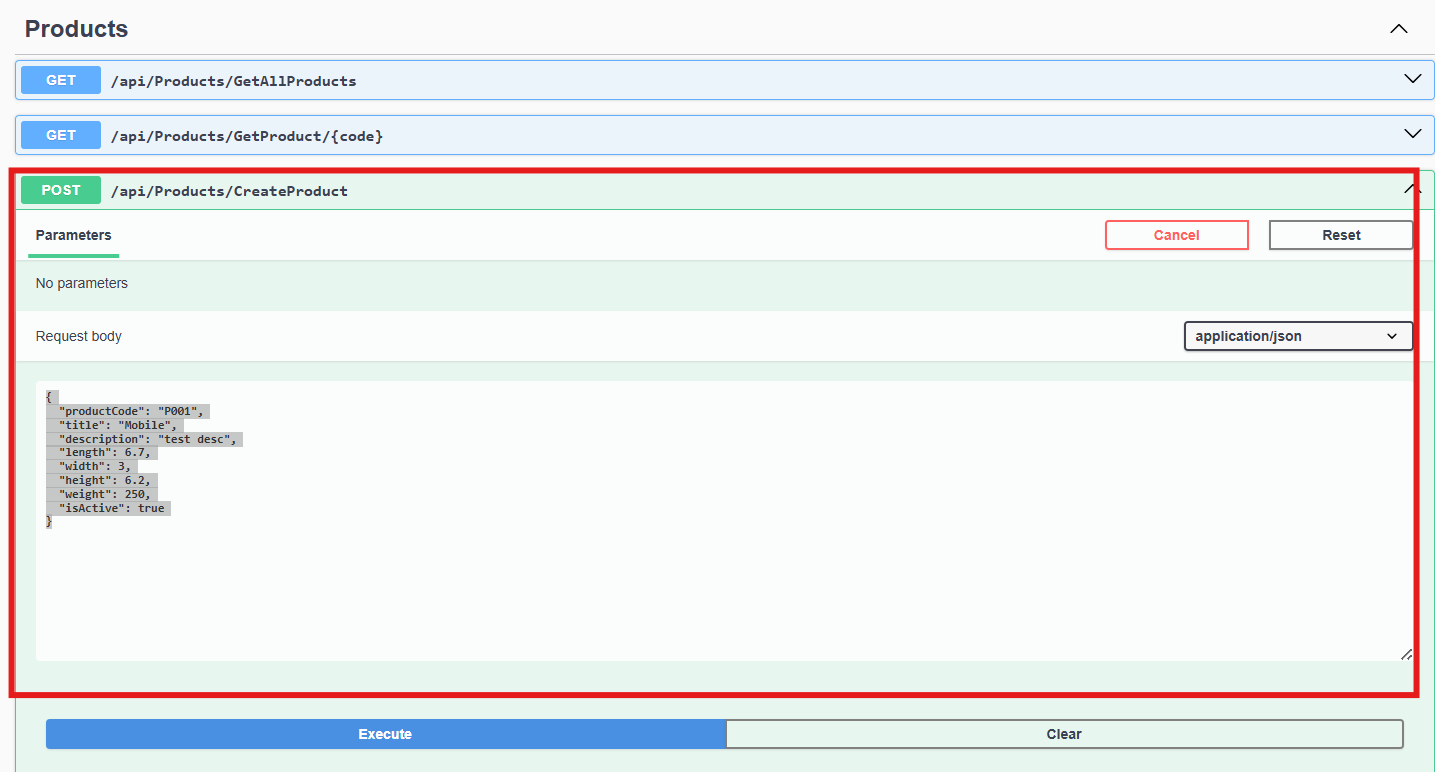
"width": 3,

"height": 6.2,

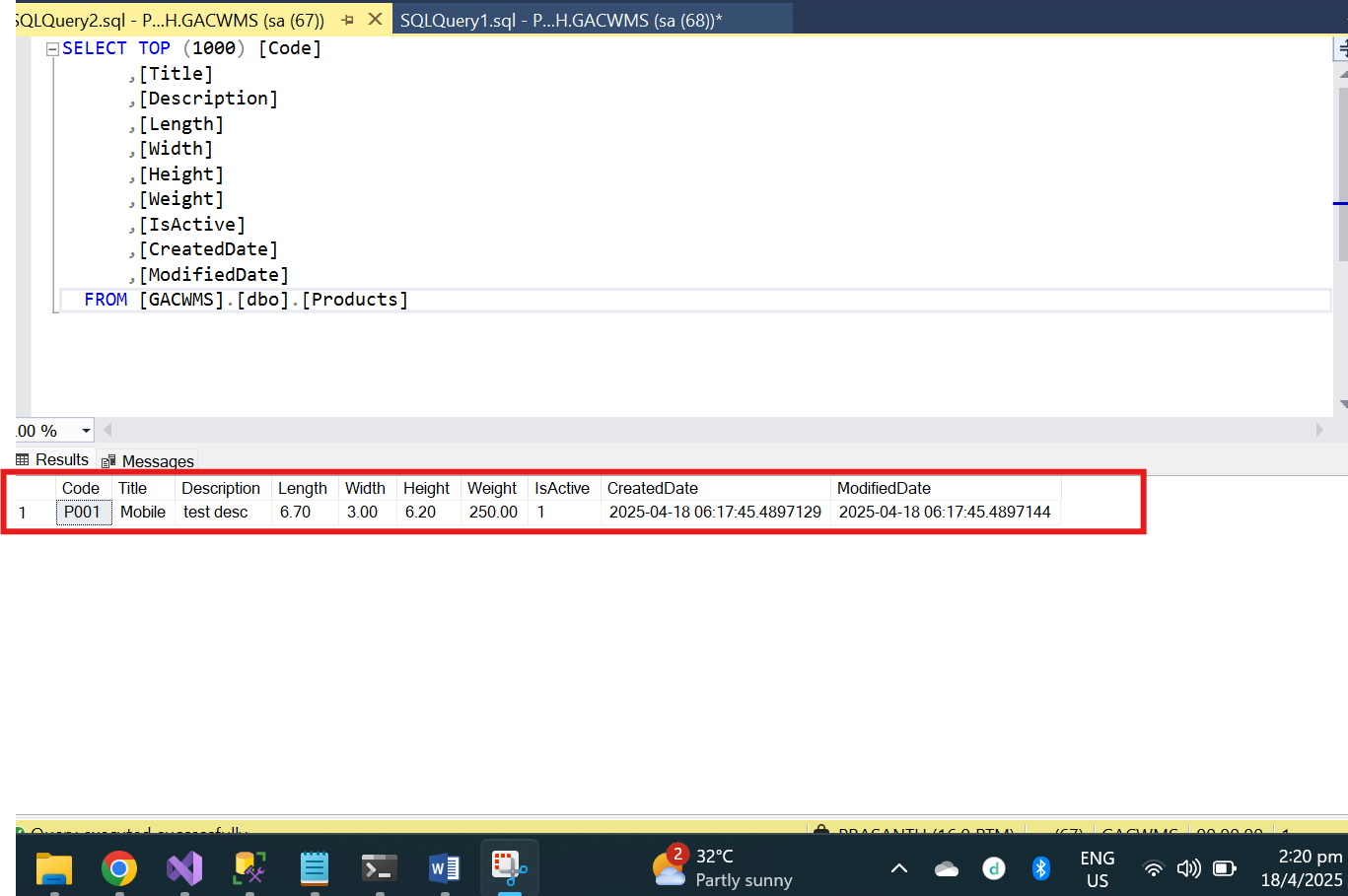
"weight": 250,

"isActive": true

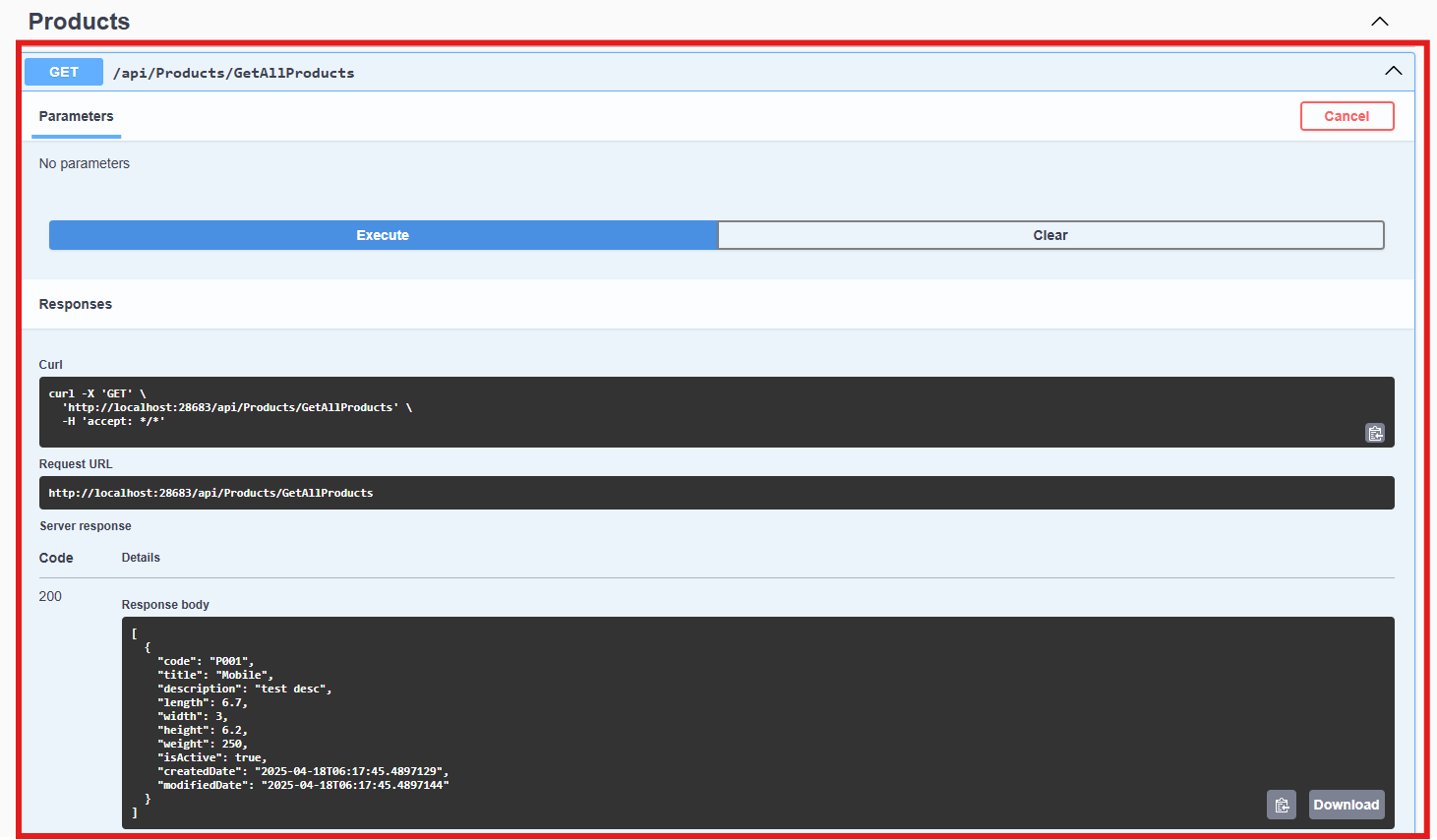
}



Output :

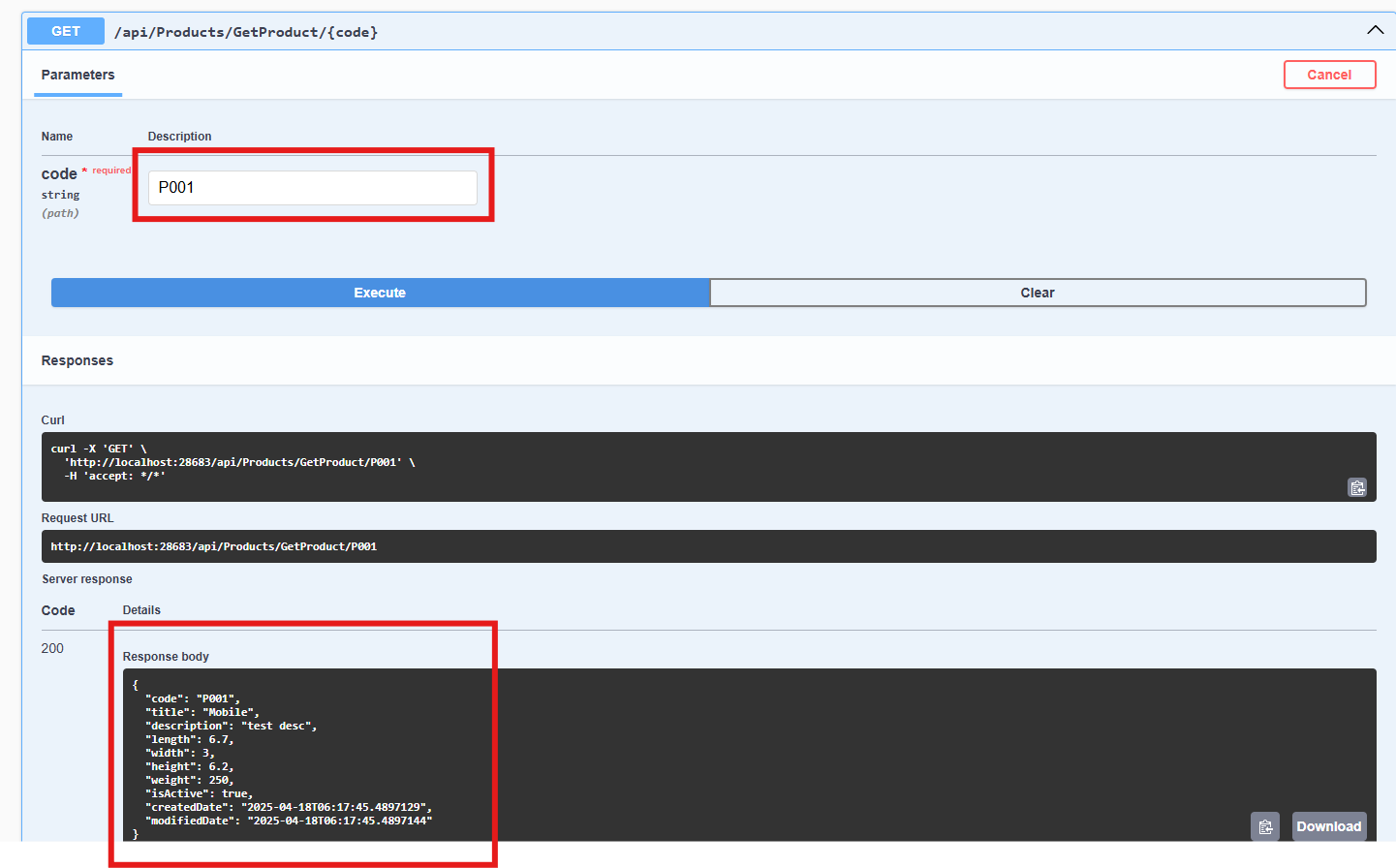


Get All Products   
I have created only one product, the response as below



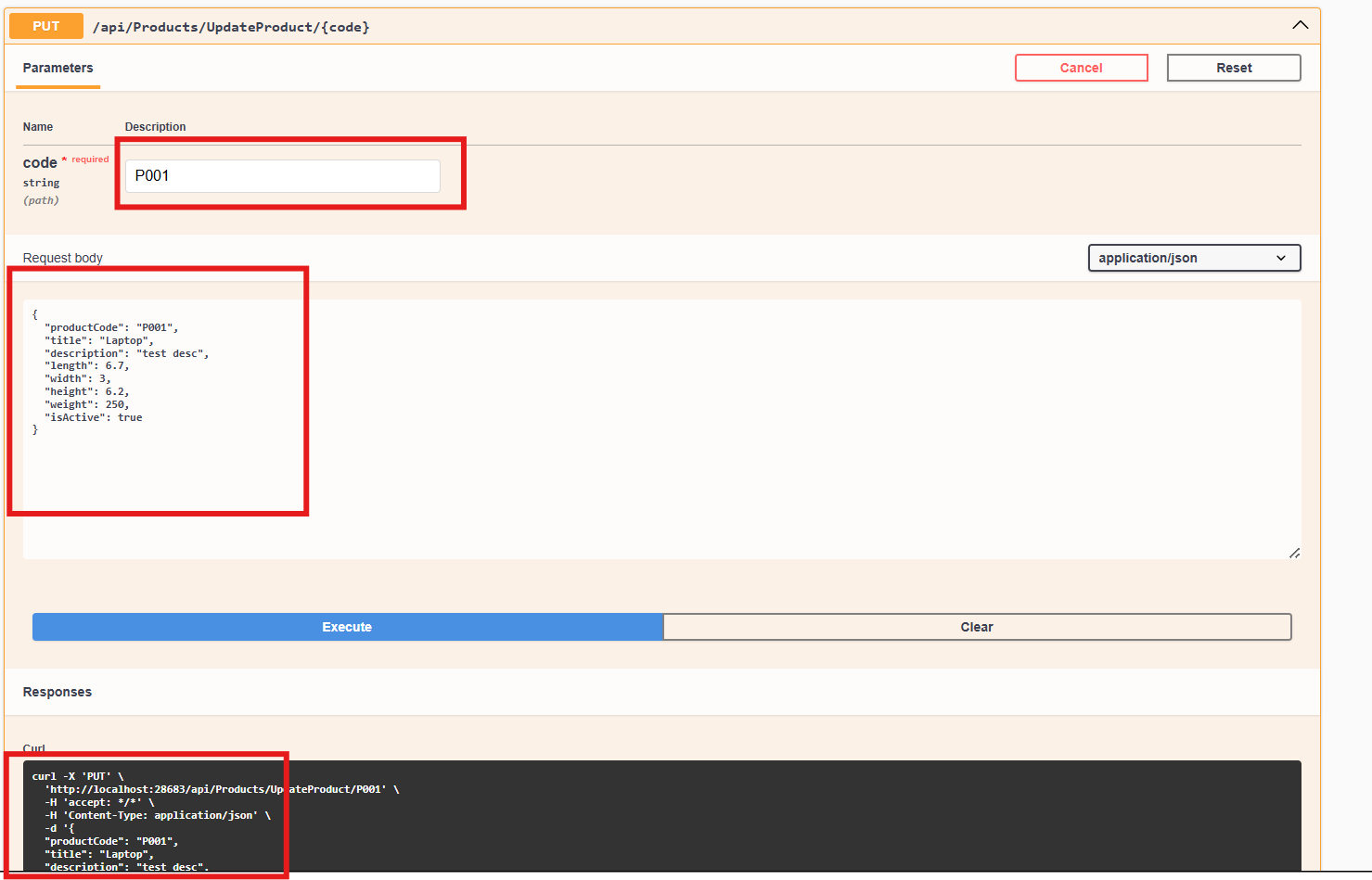
Get Product By Code :

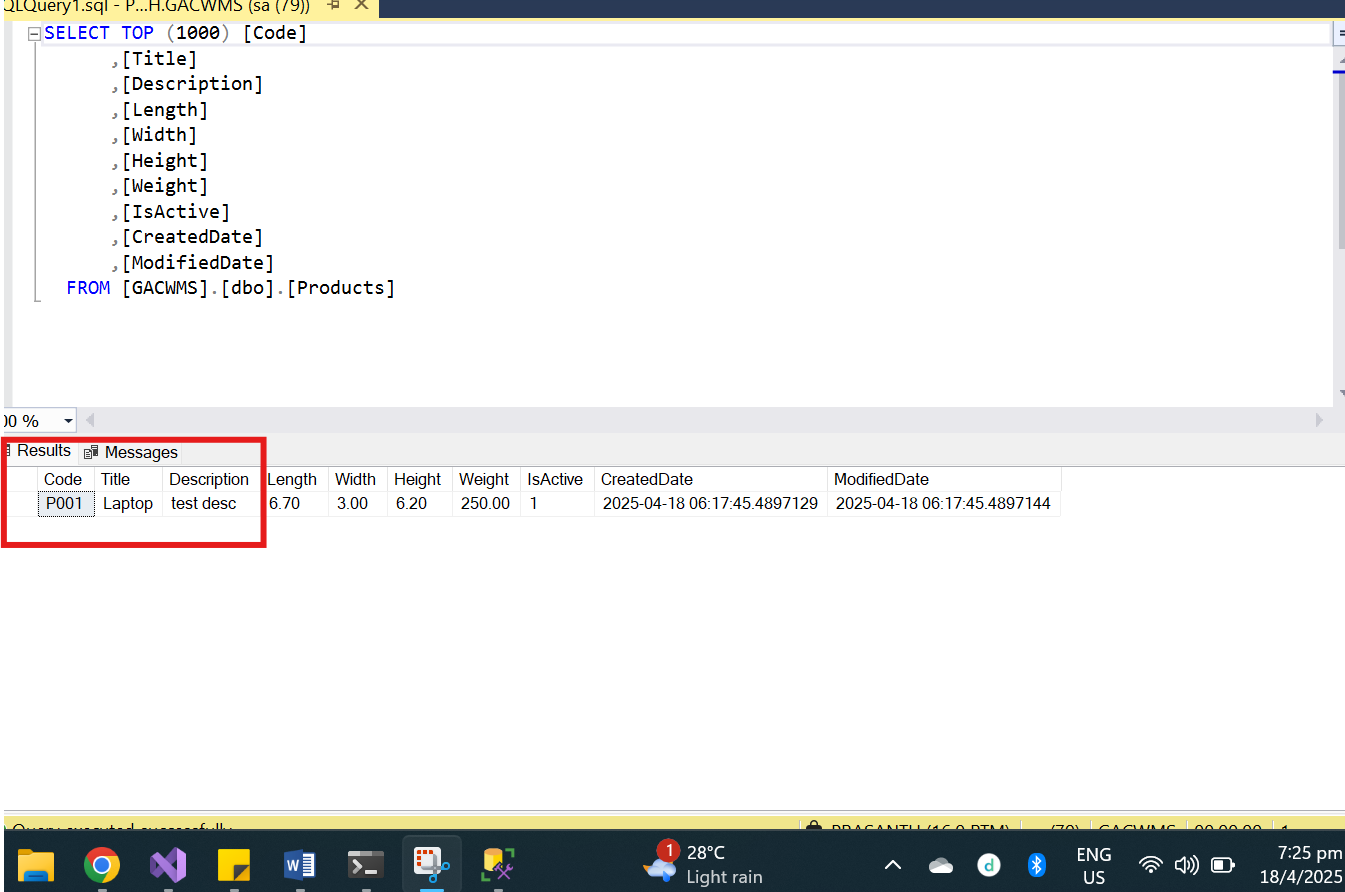
Input: Product Code : P001 and execute api, the response a below



Update Product By Code :

Input P001 as product code and updating the product title as “Laptop”.





Create Purchase Order :

Input

{

"orderId": "PO001",

"processingDate": "2025-04-18T11:28:26.713Z",

"customerId": 2,

"items": [

{

"productCode": "P001",

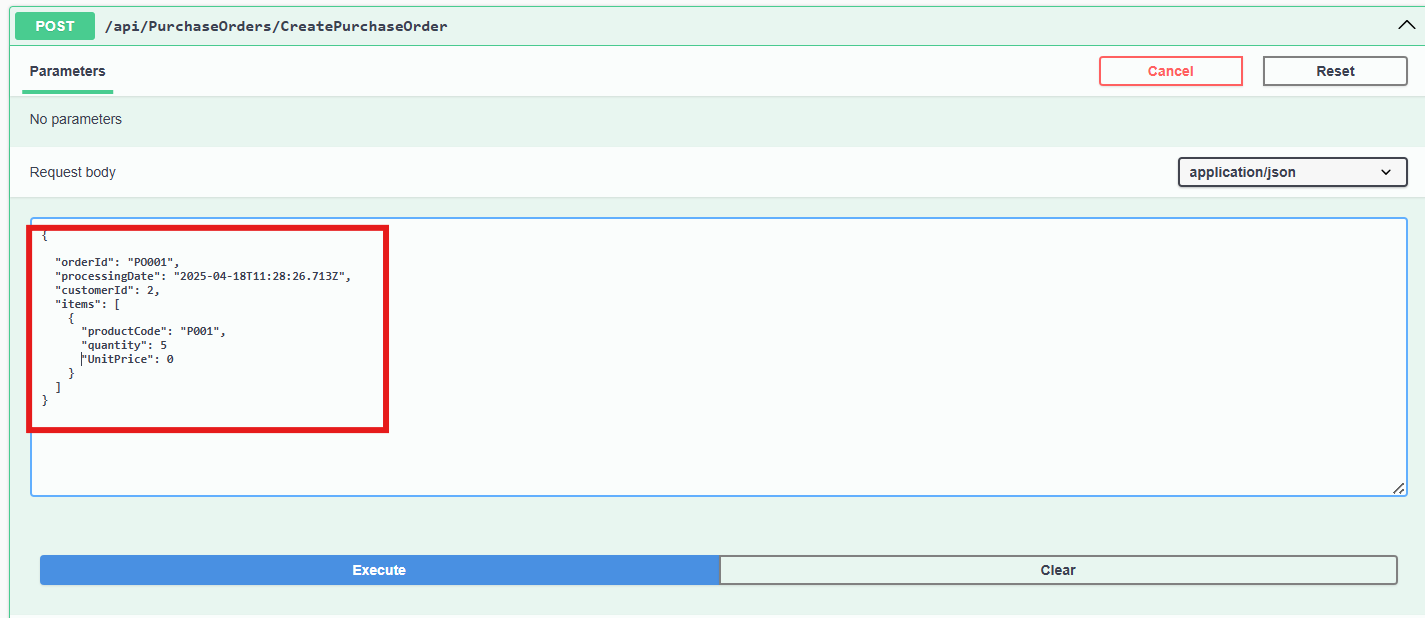
"quantity": 5

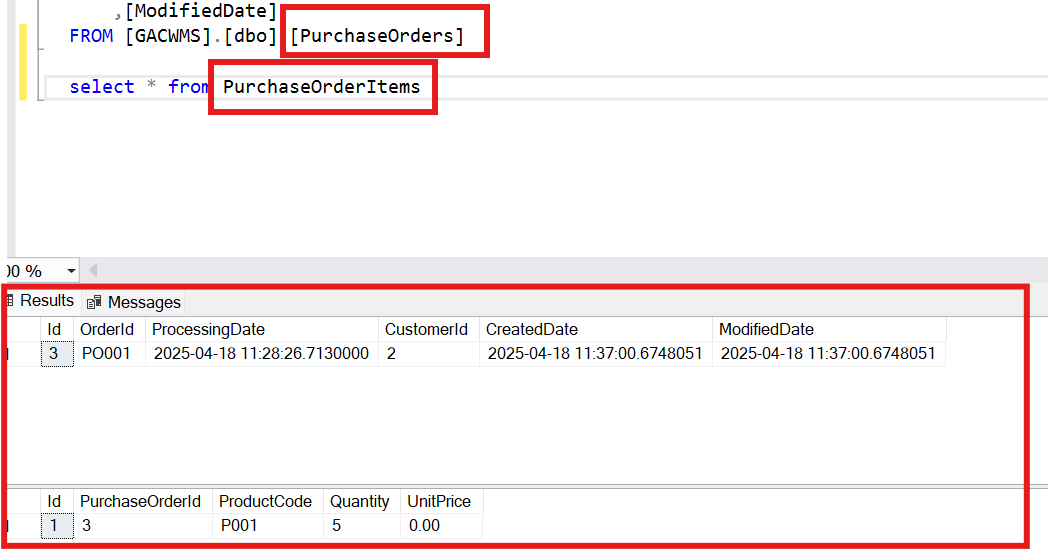
"UnitPrice": 0

}

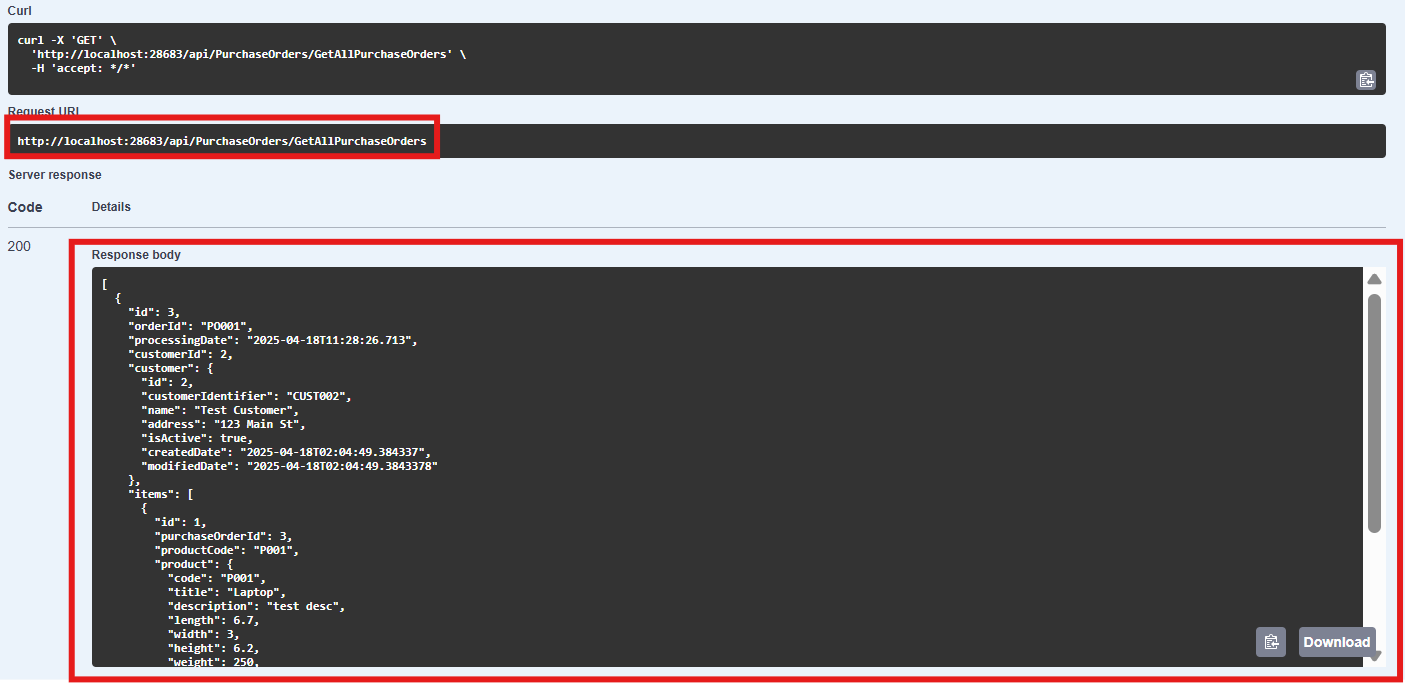
]

}





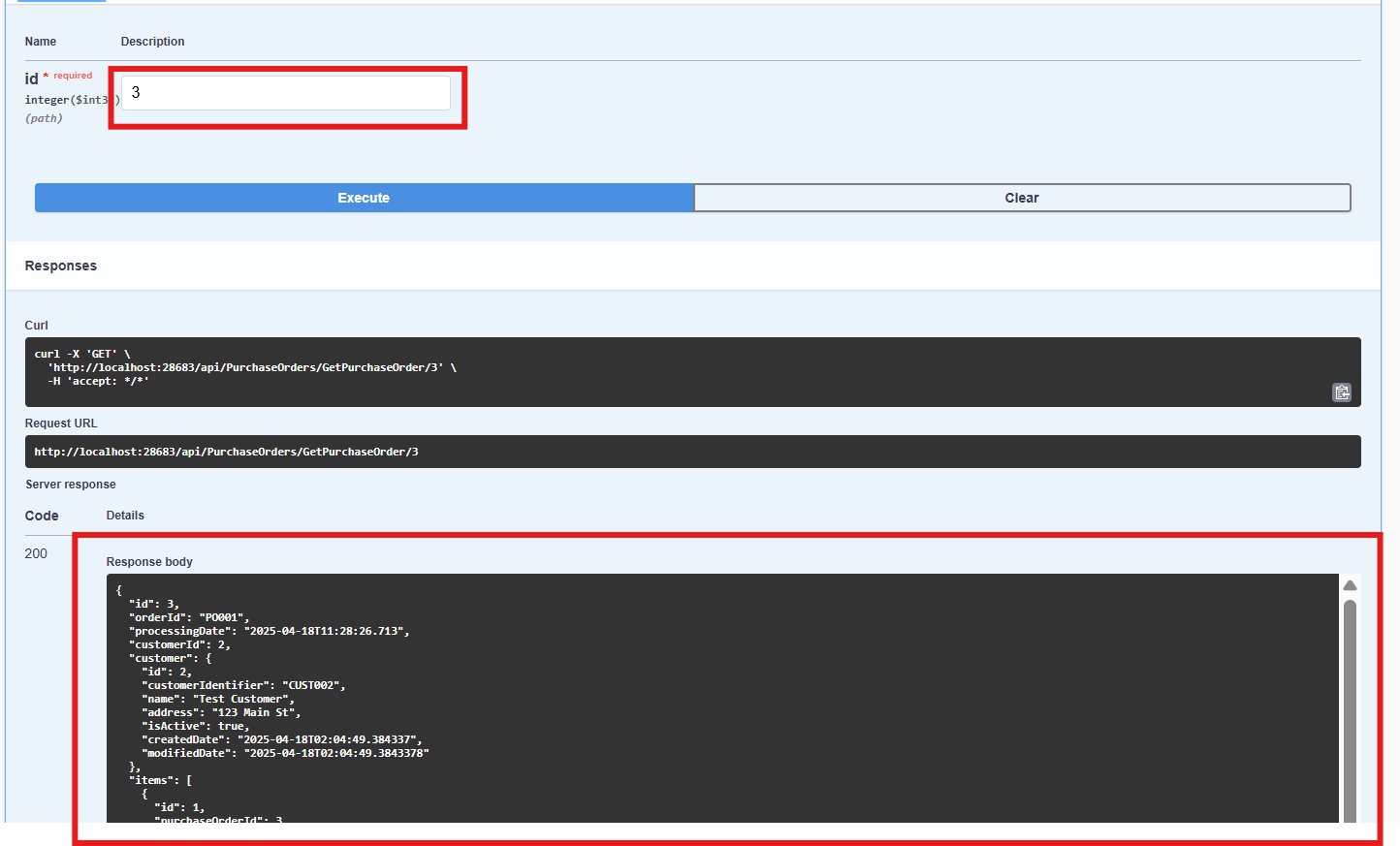
Get All Purchase Orders:



Get Purchase Order ById

Input Purchase Order Id = 3

Output:



Update Purchase Order by id   
Input : updating Quantity = 10 and unitprice = 500;

{

"id": 3,

"orderId": "PO001",

"processingDate": "2025-04-18T12:06:31.581Z",

"customerId": 2,

"items": [

{

"productCode": "P001",

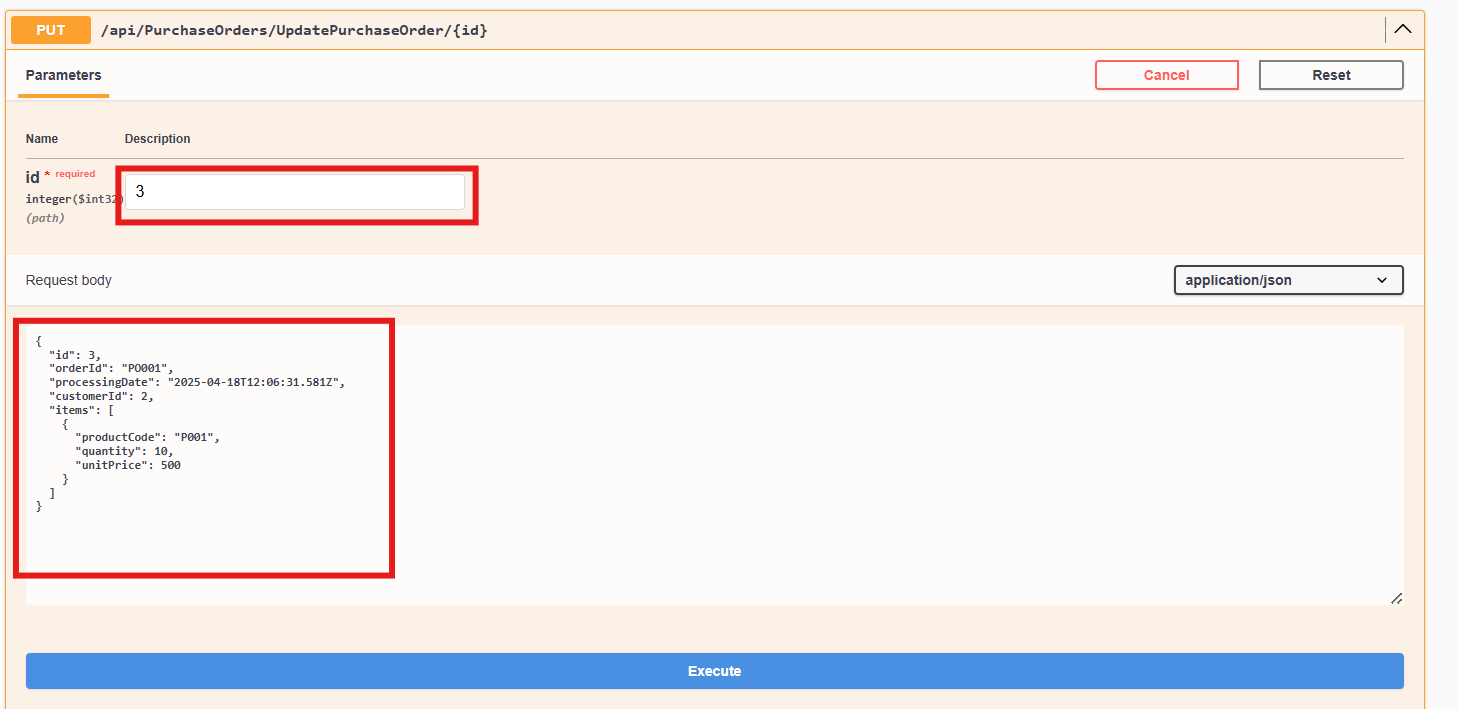
"quantity": 10,

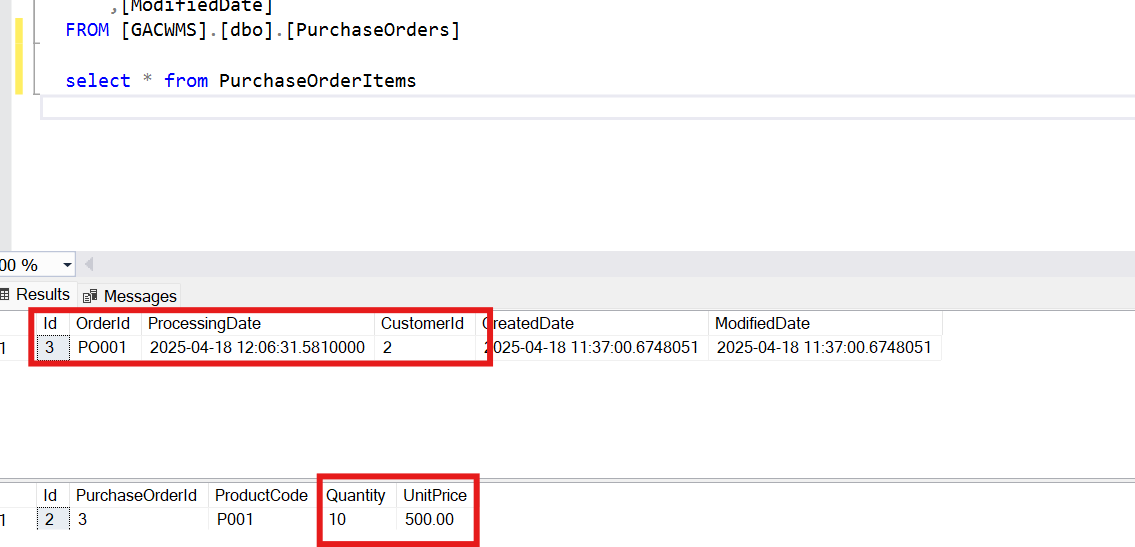
"unitPrice": 500

}

]

}





Create Sales Order :

Input :

{

"orderId": "SO001",

"processingDate": "2025-04-18T12:14:26.547Z",

"customerId": 2,

"shipmentAddress": "Singapore",

"items": [

{

"productCode": "P001",

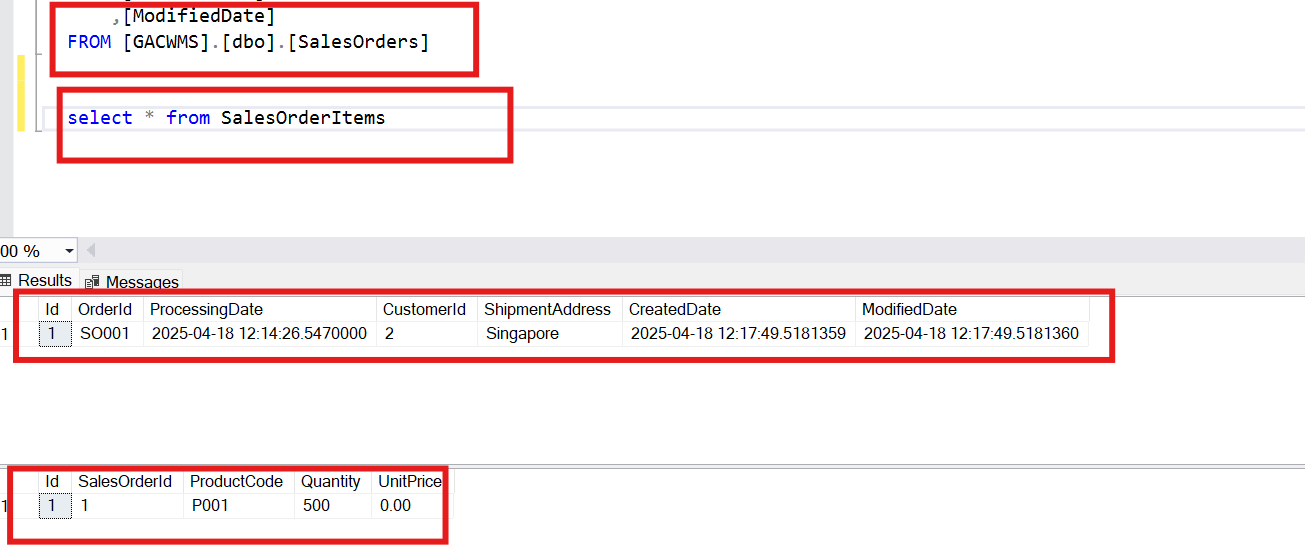
"quantity": 500

"untiPrice": 1000

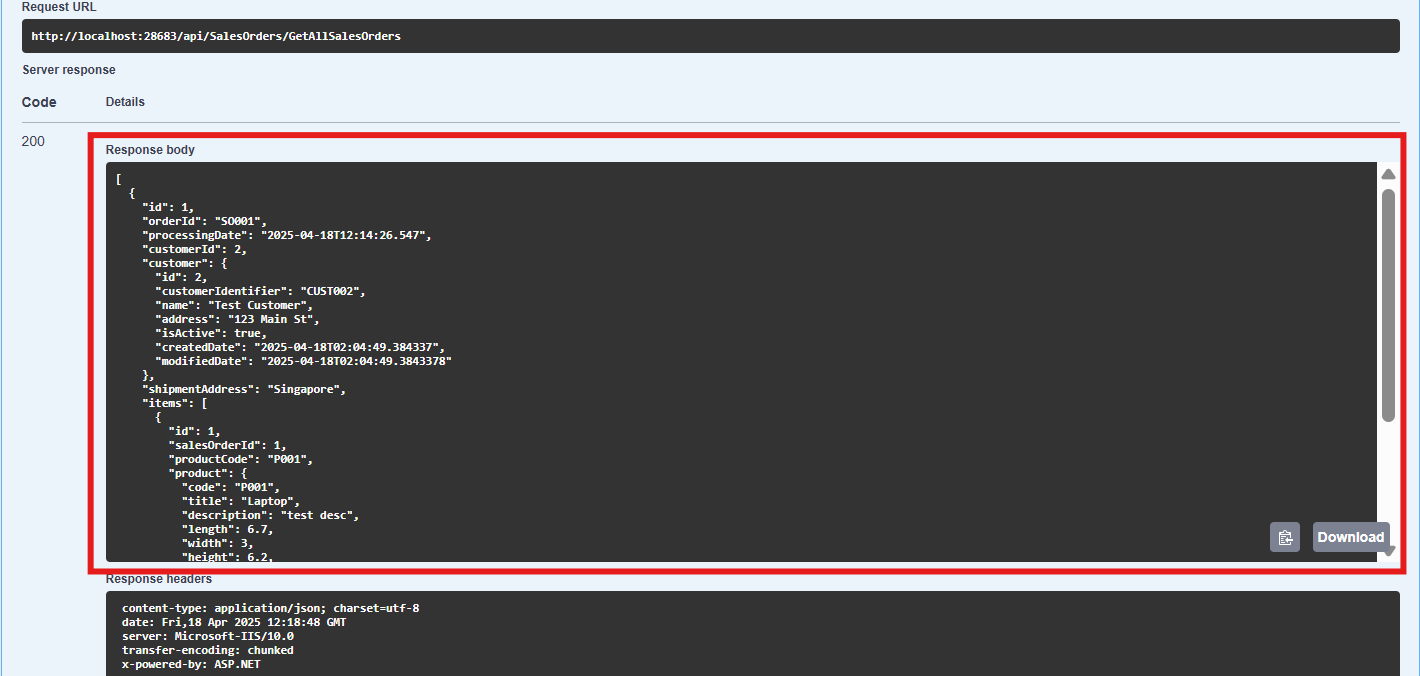
}

]

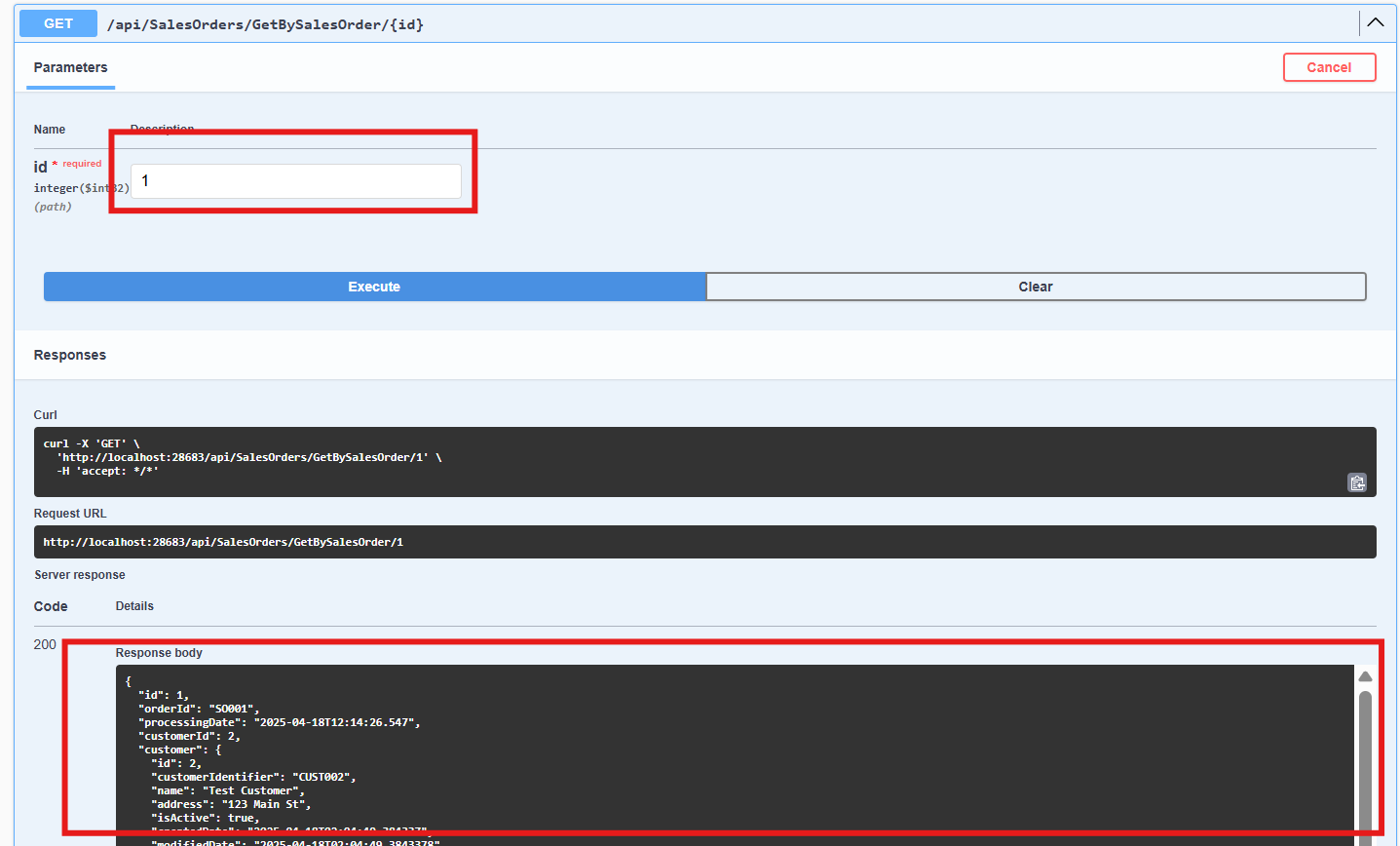
}



Get All Sales Orders



Get Sales Order by Id: input SalesOrderId = 1



Update SalesOrder by Id

Input Id =1

{

"id":1,

"orderId": "SO001",

"processingDate": "2025-04-18T12:14:26.547Z",

"customerId": 2,

"shipmentAddress": "Dubai",

"items": [

{

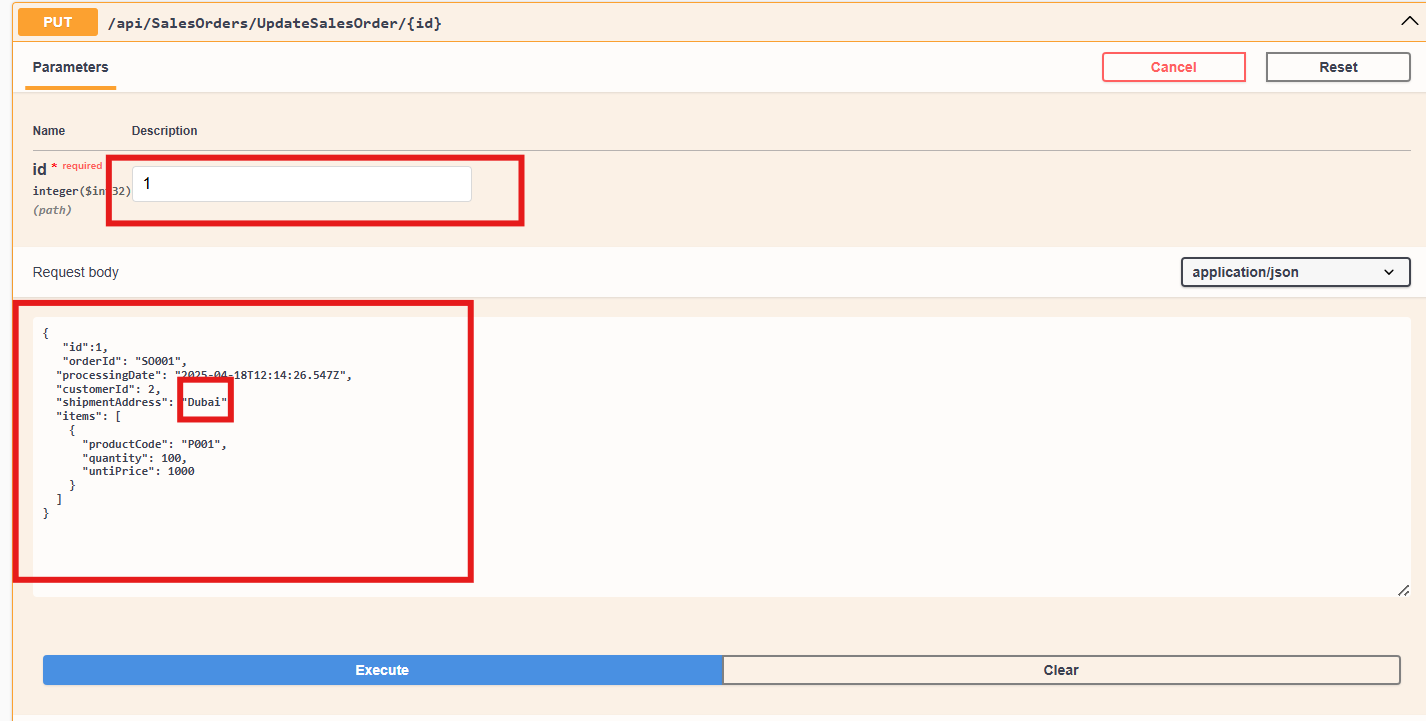
"productCode": "P001",

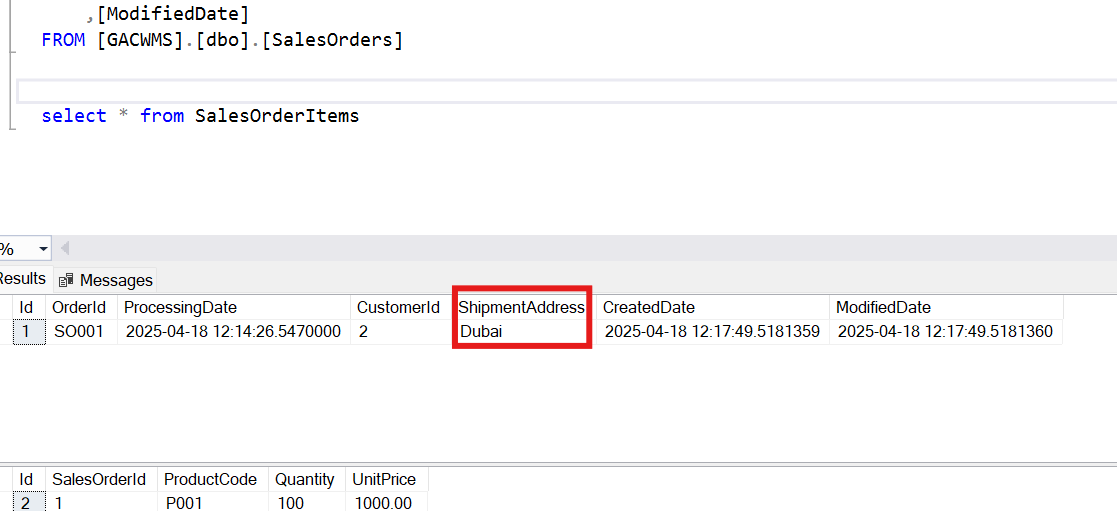
"quantity": 100

"untiPrice": 1000

}

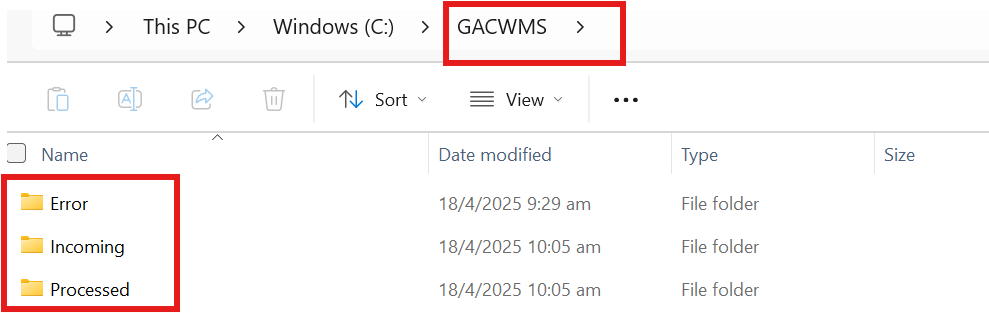
]

}  




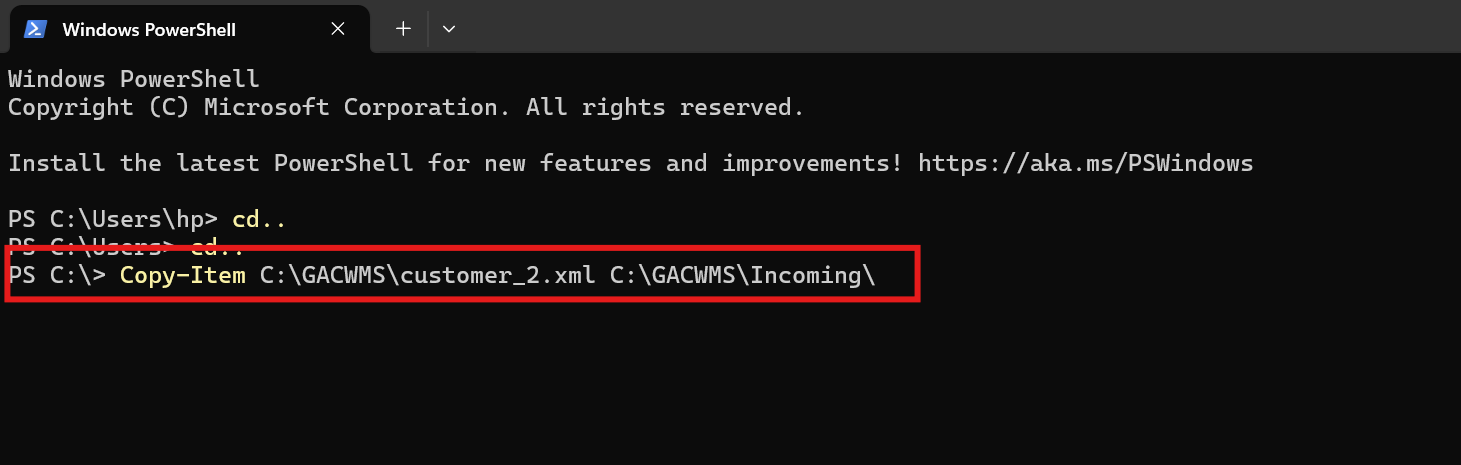
File Processor Service :

For local testing this service will create folders as mentioned in the appsettings



Using the customer\_2.xml file to trigger the file processing, when the file got created in the Incoming folder the system will auto process the file.

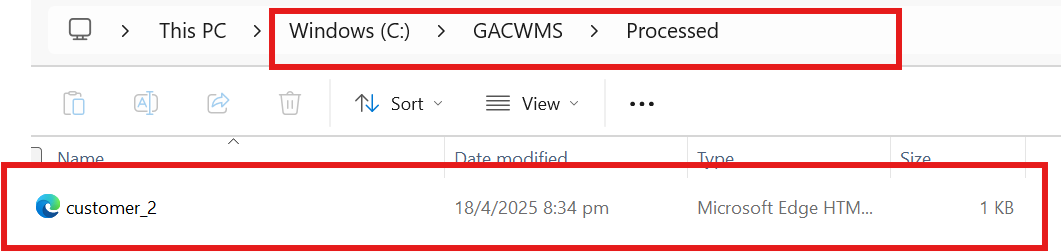
To create the file inside the Incoming folder,Using power shell to copy the file into incoming folder.

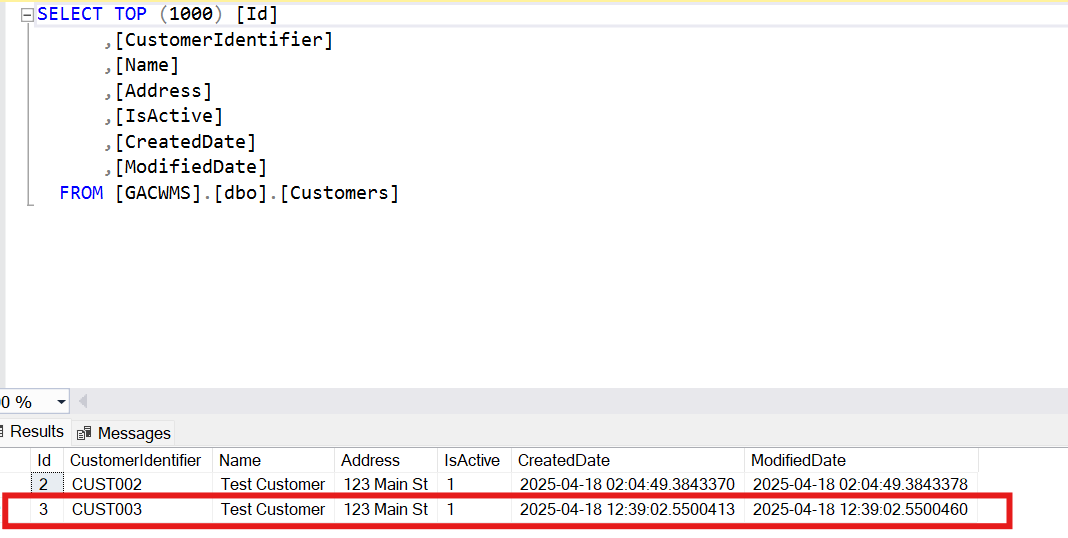


Sample Xml:   


Creating the new customer “CUST003”

After copying the file into Incoming folder the system triggered to process it and after it got created it will place the file in ProcessedFile folder.





Docker Setup:

Actually I don’t have knowledge on yamal code to write. We have separate team to handle the deployment and environment configuration, As I know how to utilize it.

I used to compose the docker file to up the docker images in local. I know how to setup the schema registration for the Topics used in the project and terraform creation for the topics to register in env likes Dev, SIT,UAT, ORT.

Thank You.