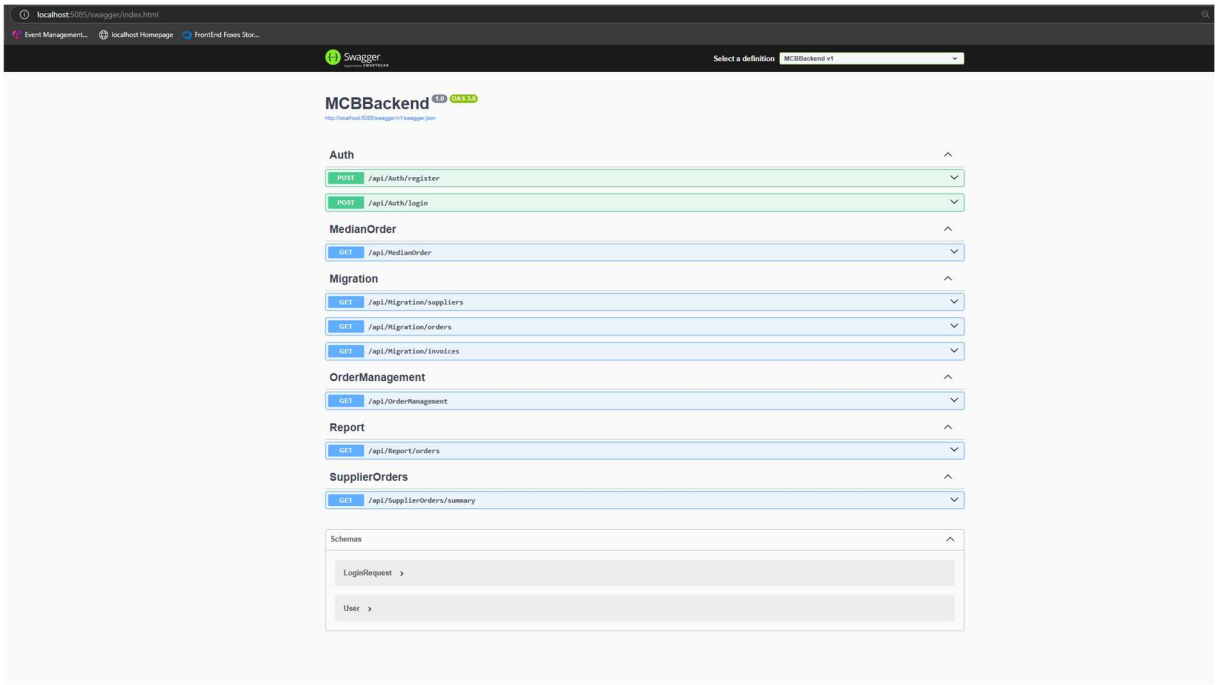
Informative Report on MCB Backend Development

1. Introduction

This report details the development of the **MCBBackend** system, which aims to provide a structured and automated solution for managing **Purchase Orders (POs), Invoices, and Supplier Transactions** for **Moris Ltd**, a financial institution. The backend was designed to migrate data from an **MSSQL database**, normalize it, and expose **RESTful APIs** for data retrieval and reporting.



2. API Endpoints and Authentication

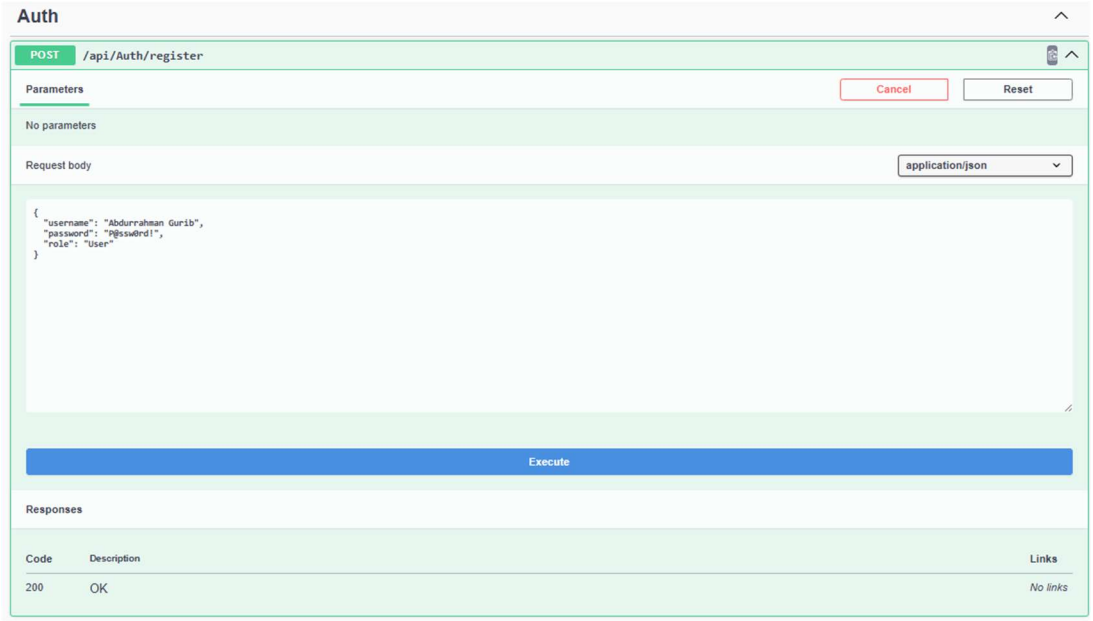
The **MCBBackend** exposes multiple API endpoints, categorized as follows:

2.1 Authentication APIs

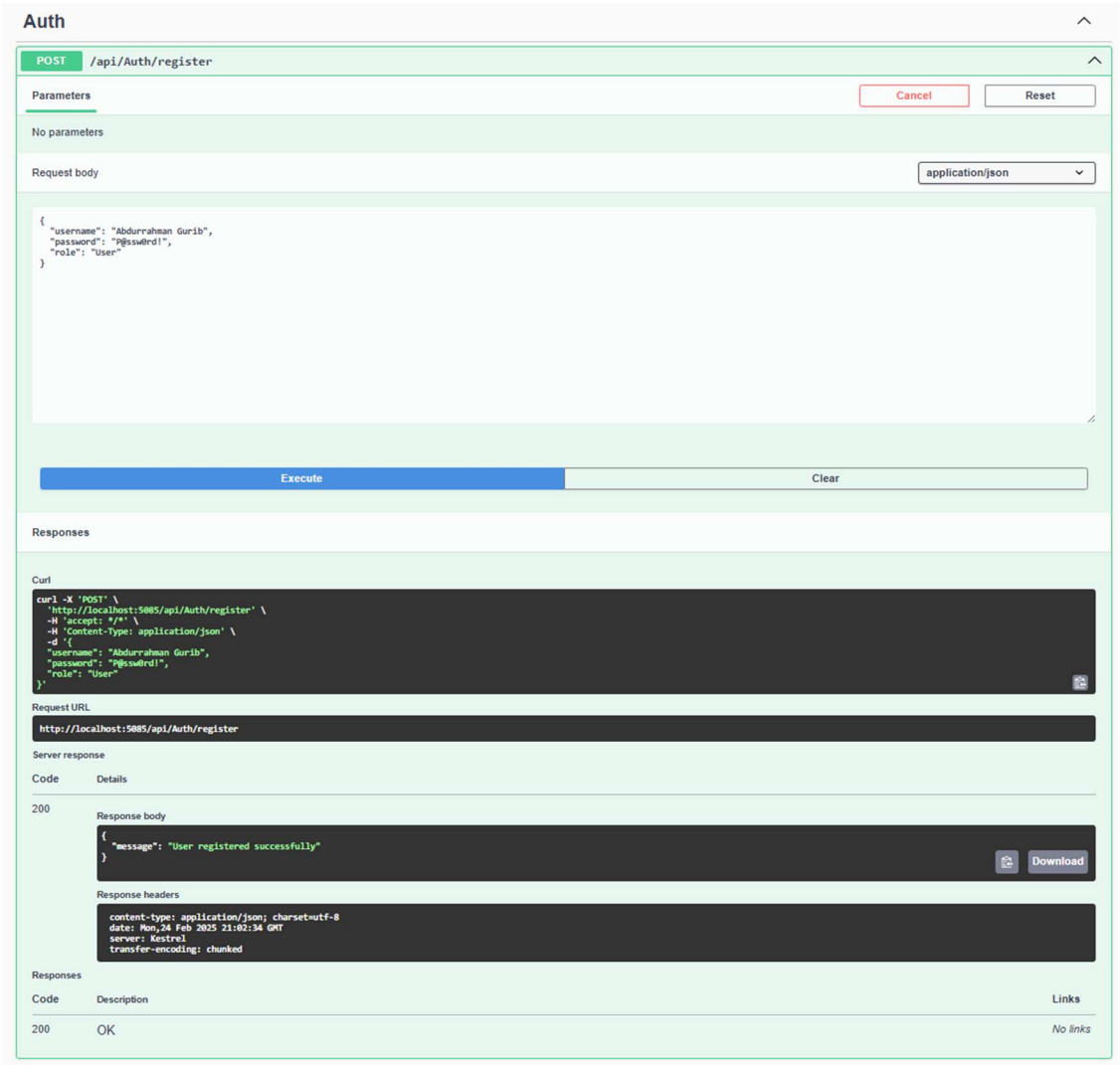
To secure access, the backend provides JWT-based authentication for user registration and login.

- **POST /api/Auth/register** → Registers a new user.

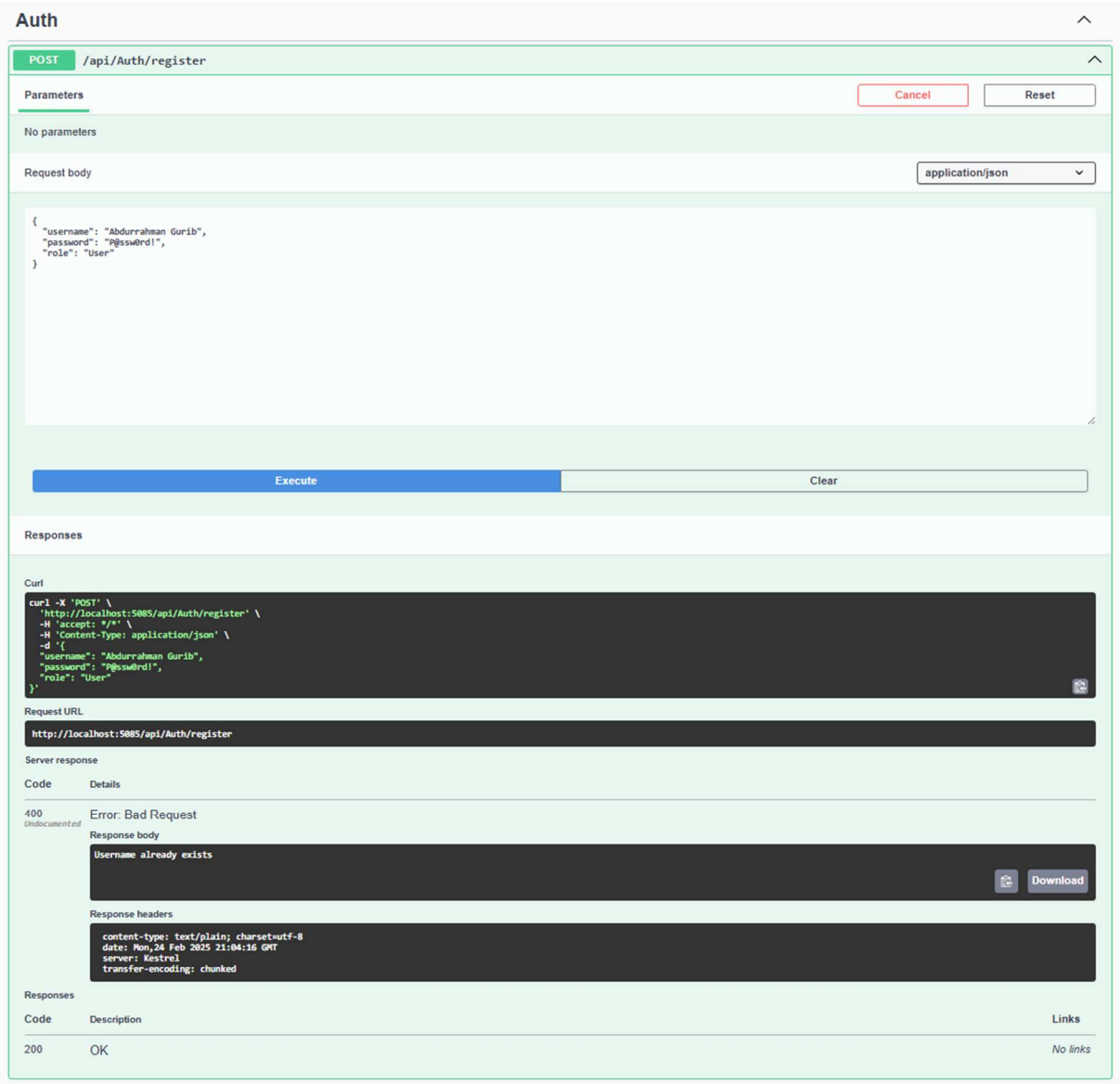
Registering a user



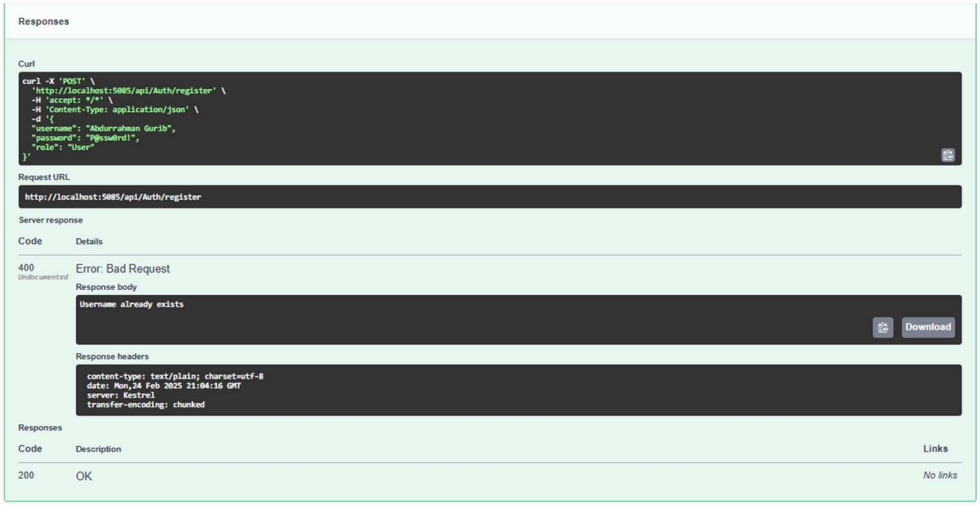
User Registered Successfully



Unit Test: Validation if re-register the same user

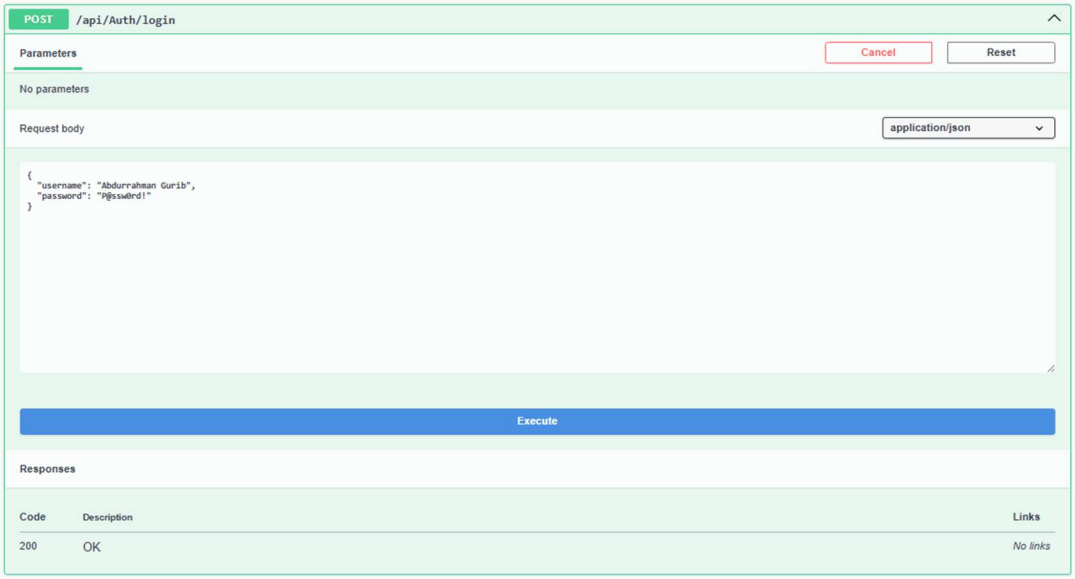


Server response status 400, and indicate Username already exists

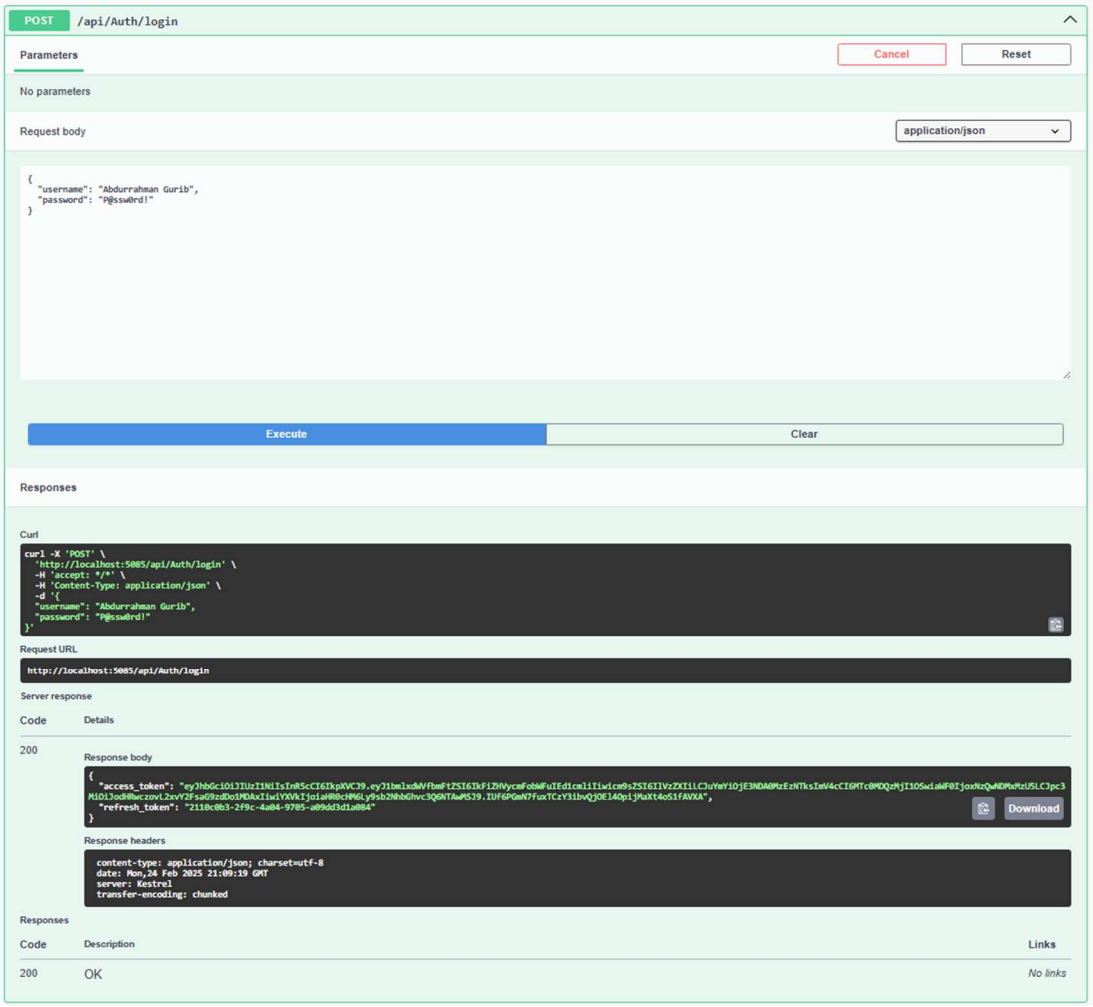


- **POST /api/Auth/login** → Authenticates users and generates a JWT token.

Log in the user:



User Logged Successfully



Security Measures

- **JWT Authentication**: Protects endpoints from unauthorized access.
- **Role-Based Access Control** (RBAC): Ensures restricted access.
- **SQL Injection Prevention**: Parameterized queries used in stored procedures.
- **Data Validation**: Input sanitization for secure data handling.

JWT Token Security:

Signed Token: The JWT is signed using HMAC-SHA256 with a secret key.

Token Expiration: The JWT has a short expiration time (15 minutes) to limit its validity period.

Issuer & Audience Claims: These ensure the token is valid and intended for your API.

Refresh Token Mechanism:

Refresh Token Generation: A GUID is used as a refresh token.

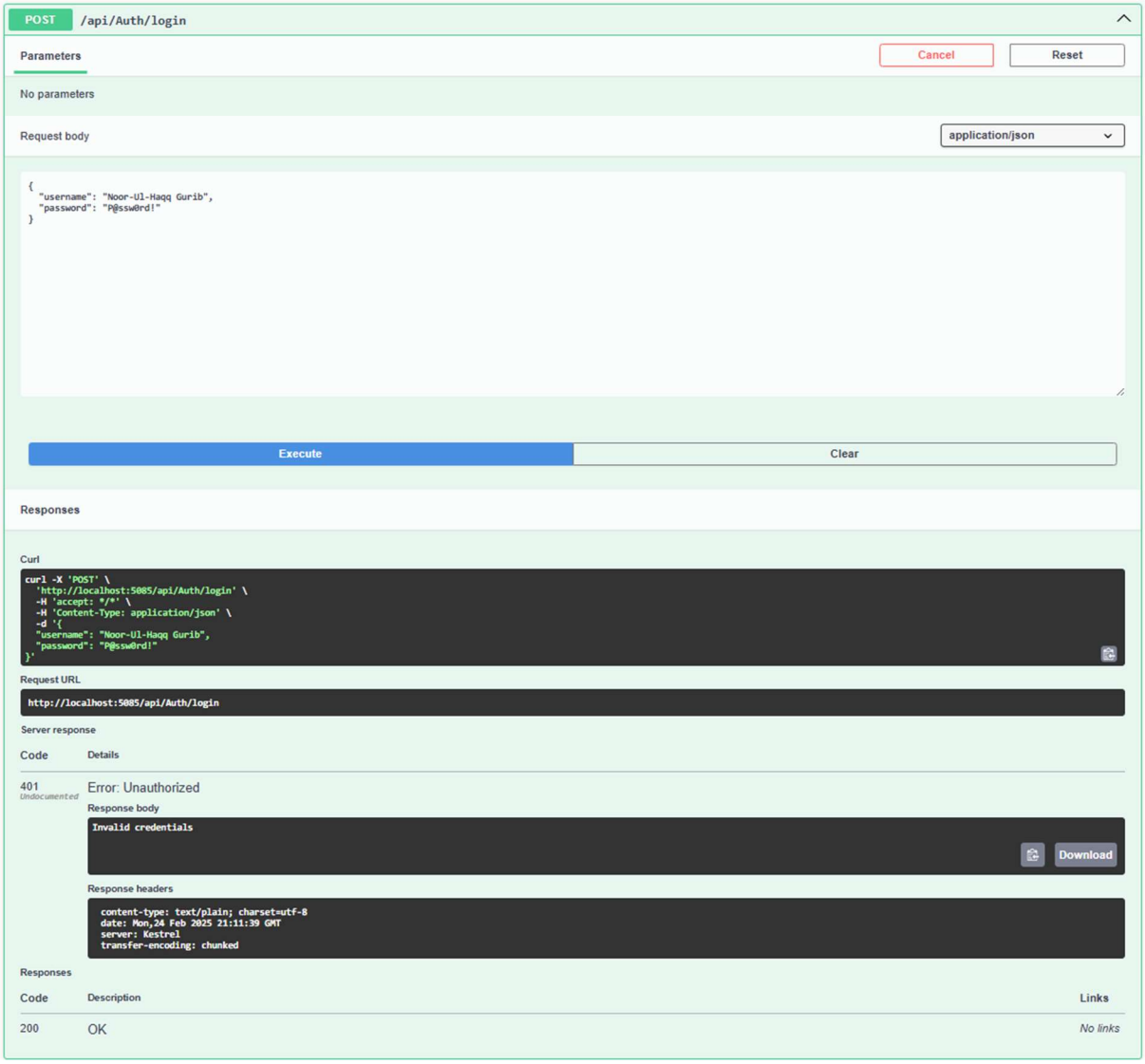
Refresh Token Storage: With the addition of SaveRefreshToken, the refresh token is stored securely.

Token Renewal: The refresh token allows obtaining a new JWT after expiration without re-authenticating.

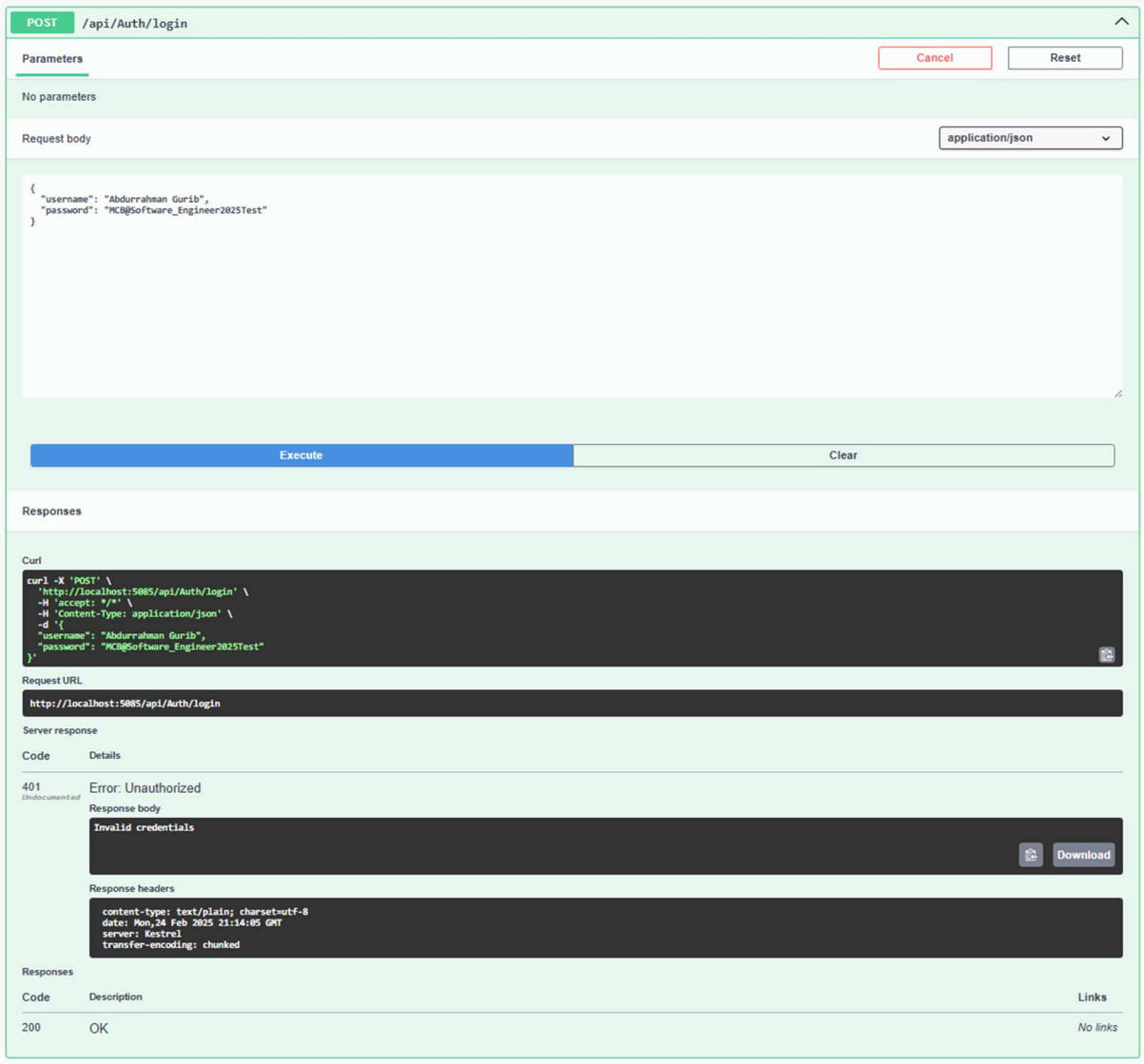
Appsettings.json [JWT, SQL Connection, Tokens]

```
{
  "ConnectionStrings": {
"DefaultConnection": "Server=4JPRJX3\\SQL2022DE;Database=MCB_Assignment_2025;Trusted_Connection=True;TrustServerCertificate=True;"
  },
  "Jwt": {
  "Key": "A0D3C9F1E6B8G5H7I2K4J6L8M9N0P1Q3",
  "Issuer": "https://localhost:5001",
  "Audience": "https://localhost:5001",
  "AccessTokenExpirationMinutes": 15,
  "RefreshTokenExpirationDays": 7
},
  "Logging": {
  "LogLevel": {
    "Default": "Information",
    "Microsoft": "Warning",
    "Microsoft.Hosting.Lifetime": "Information"
  }
},
  "AllowedHosts": "*"
}
```

Unit Test: Validation if Username does not exist

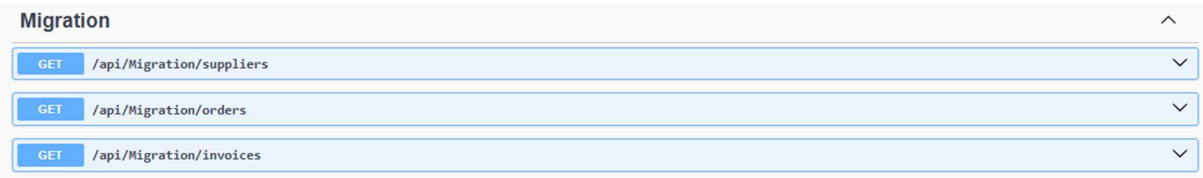


Unit Test: Validation if Username is registered, and password is wrong



2.2 Data Management APIs

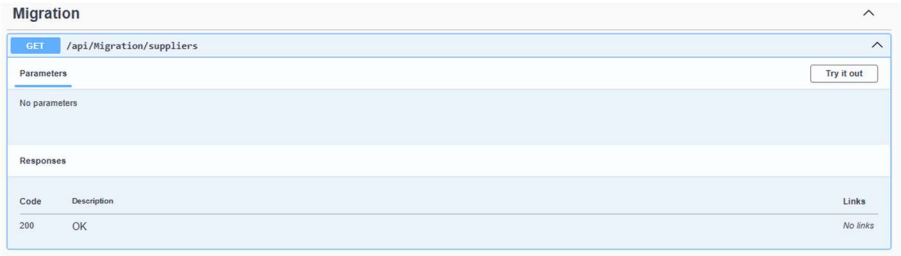
These APIs retrieve key data related to **suppliers, orders, invoices, and reports**.

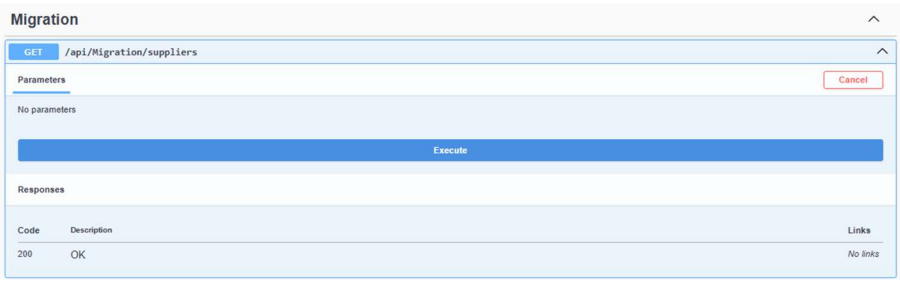


2.2.1 Migration APIs

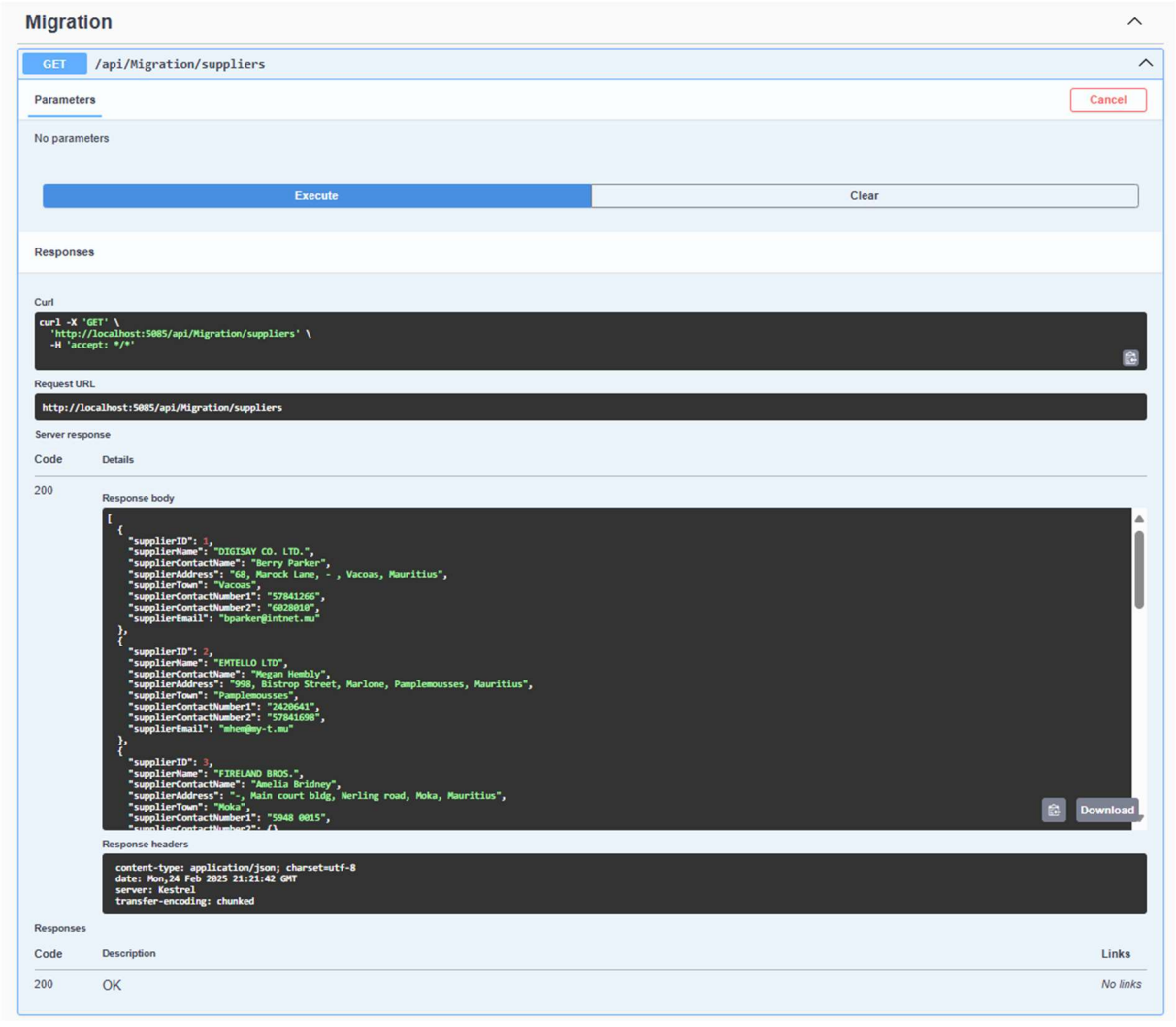
These endpoints fetch data migrated from the **BCM_ORDER_MGT** table:

- **GET /api/Migration/suppliers** → Retrieves supplier details.

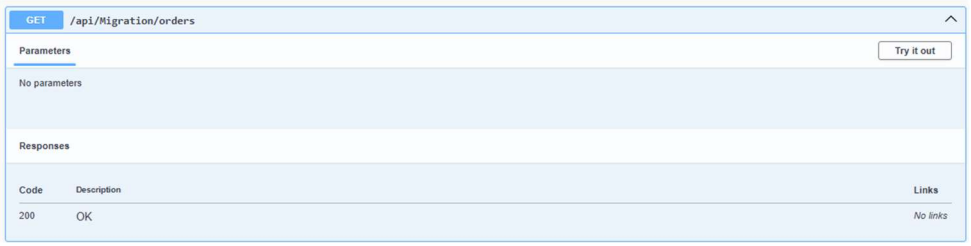




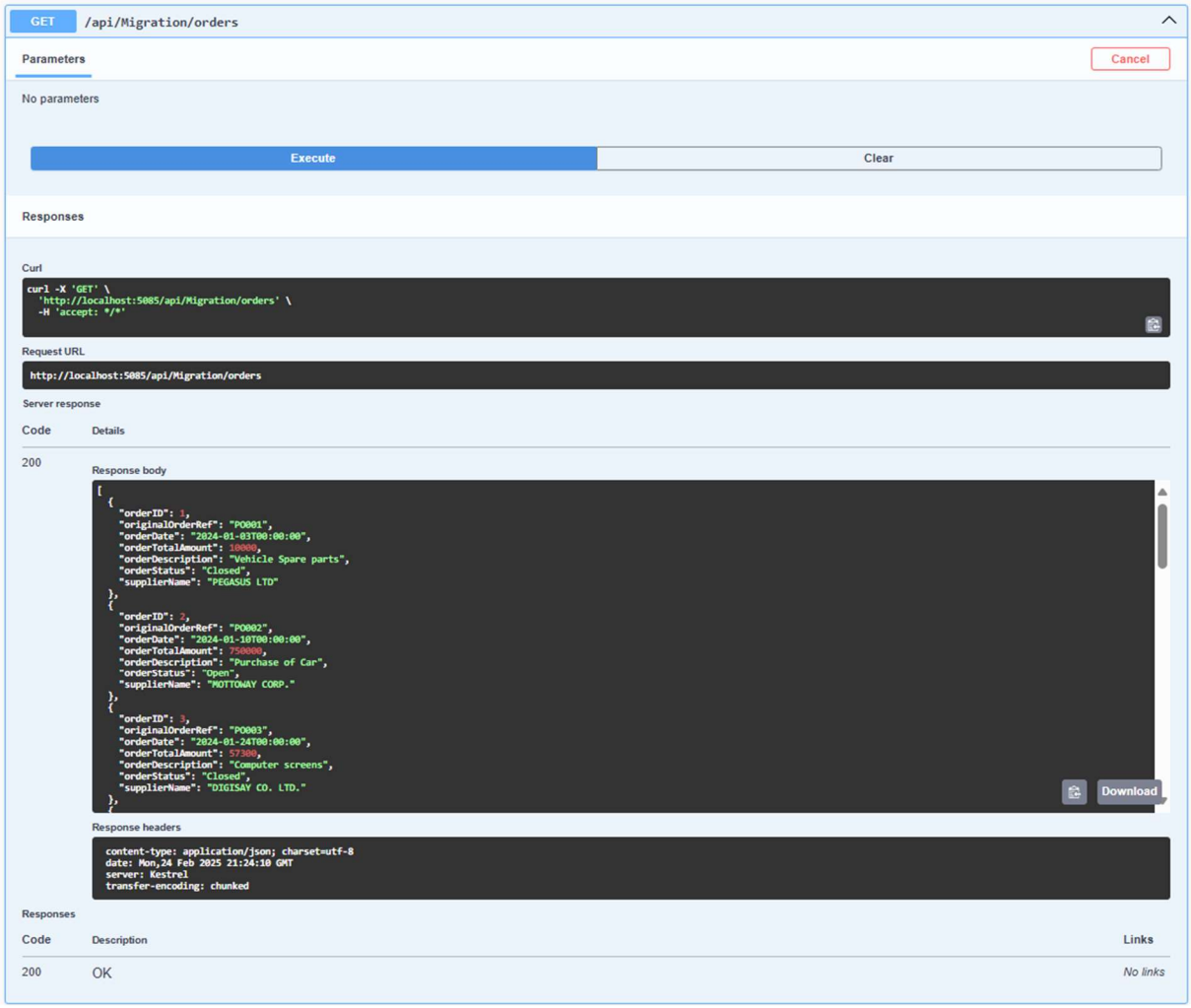
Successfully migrated data to BDO.Suppliers Table



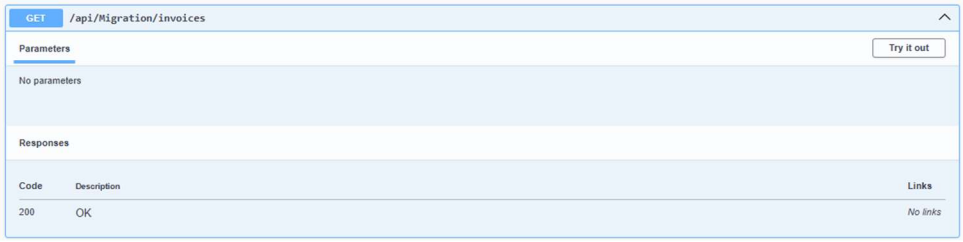
- **GET /api/Migration/orders** → Retrieves orders from suppliers.

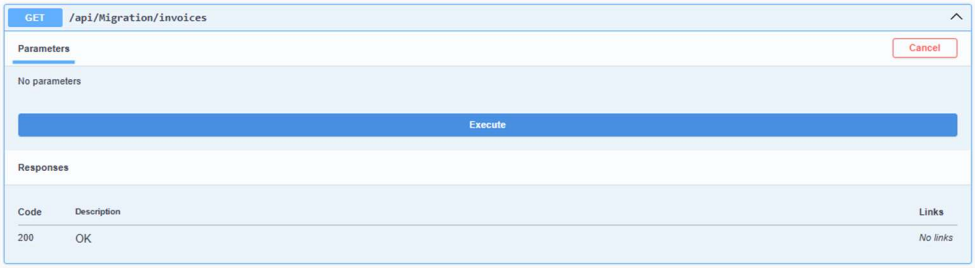


Successfully migrated data to BDO.Orders Table

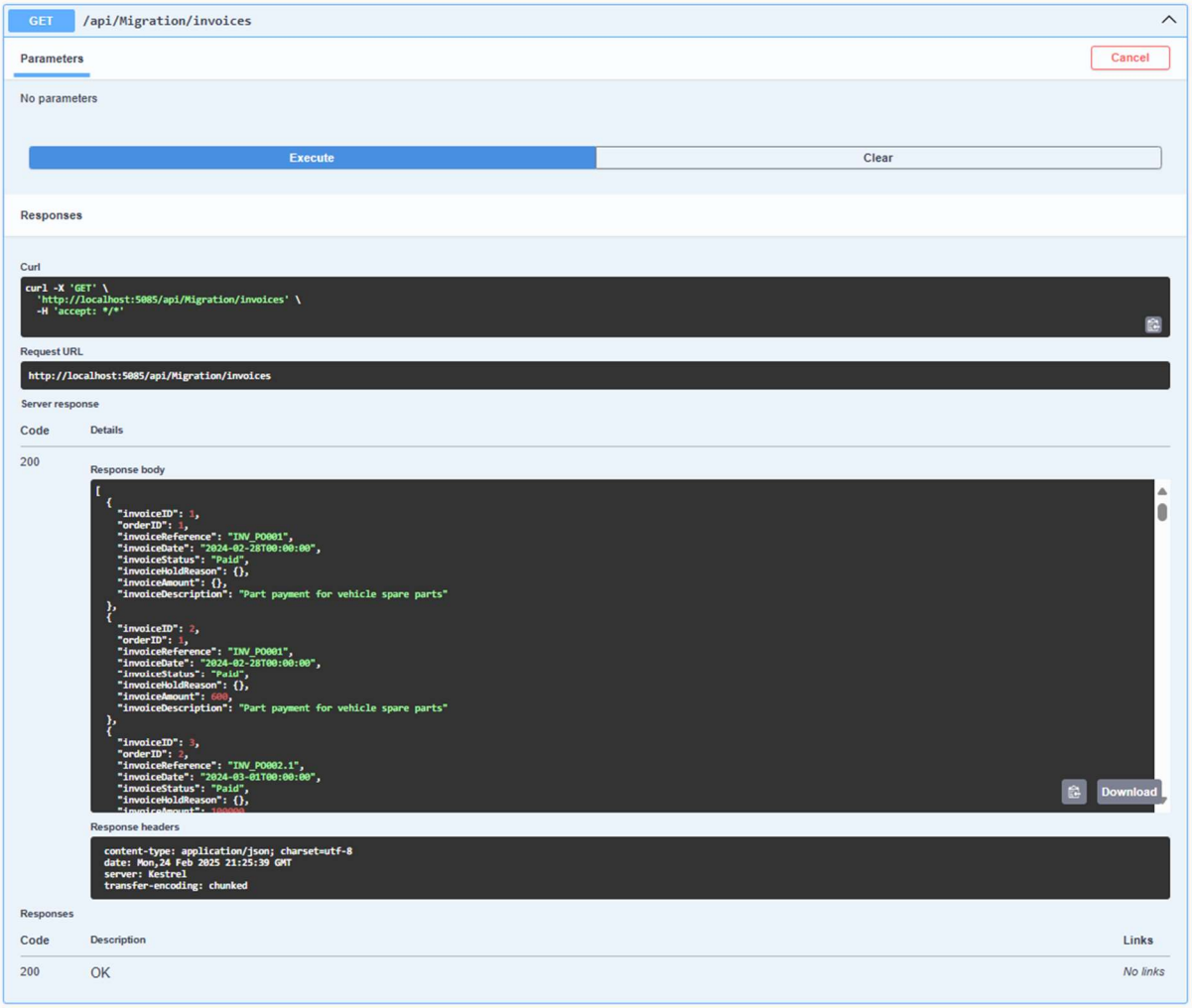


- **GET /api/Migration/invoices** → Retrieves invoices linked to orders.



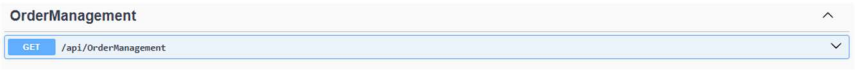


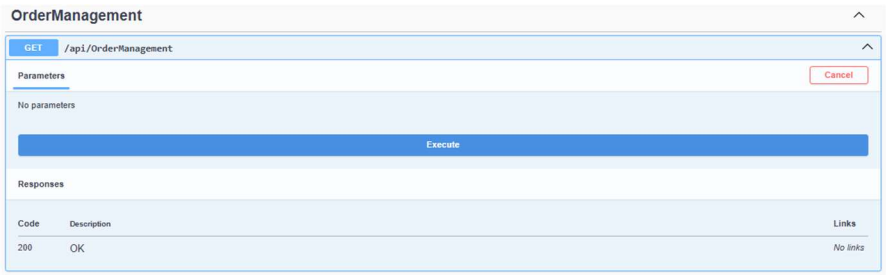
Successfully migrated data to BDO.Invoices Table



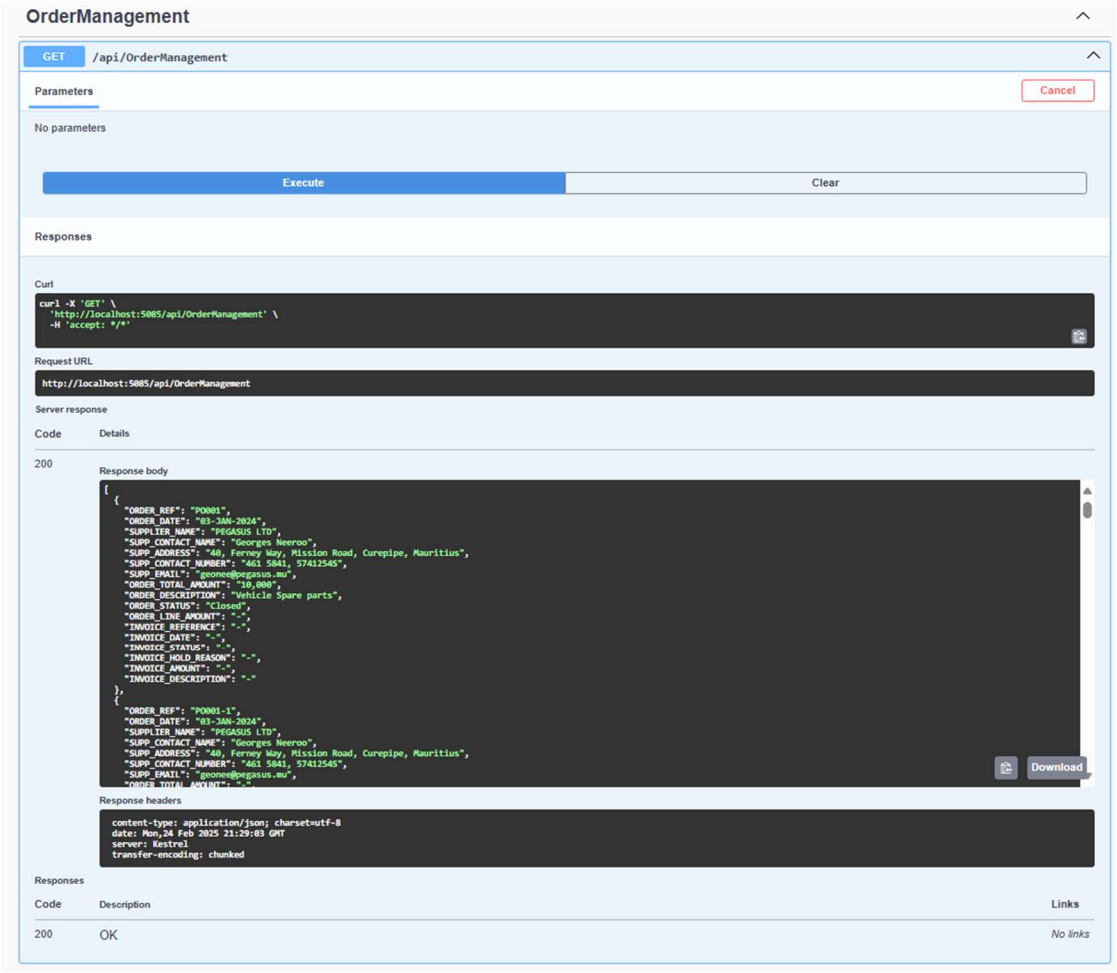
2.2.2 Order Processing APIs

- **GET /api/OrderManagement** → Retrieves structured **purchase order and invoice** details.



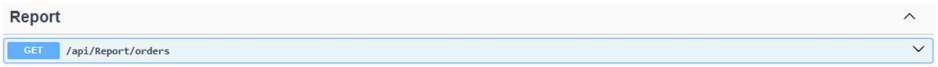


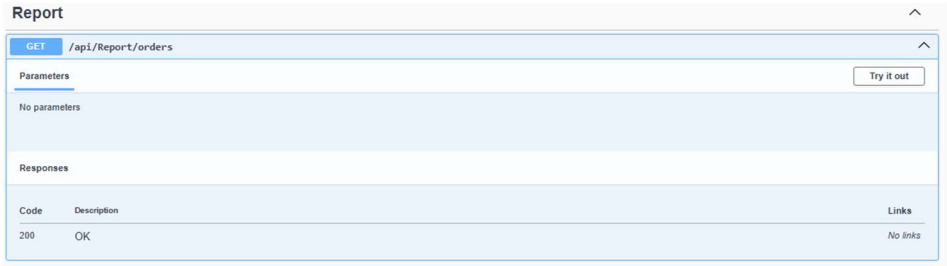
Displaying all raw data from BCM_ORDER_MGT which are not normalized.

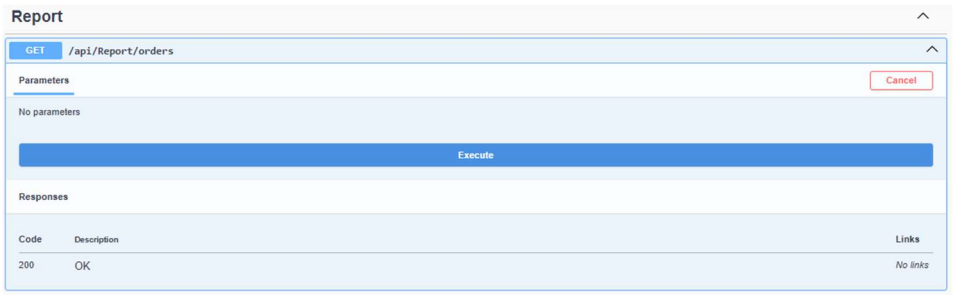


2.2.3 Reporting APIs

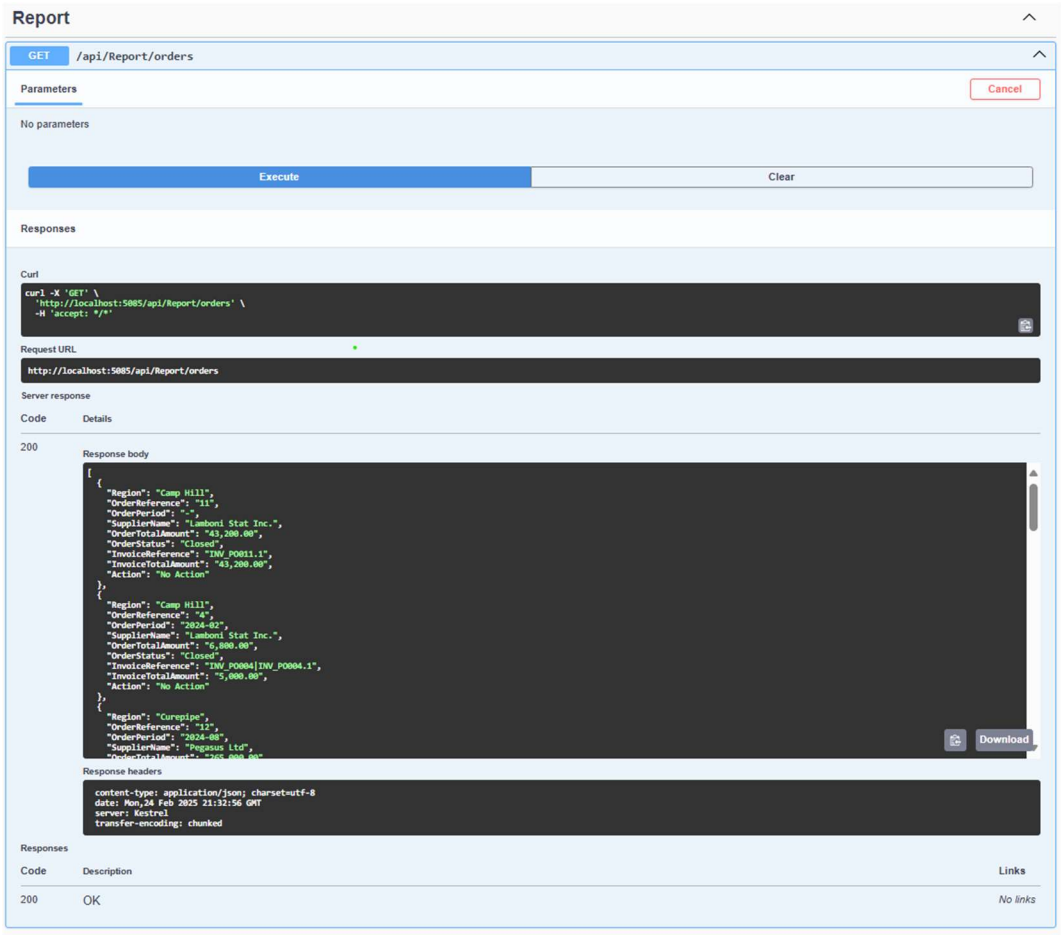
- **GET /api/Report/orders** → Provides a **summary report** of all purchase orders grouped by supplier region.



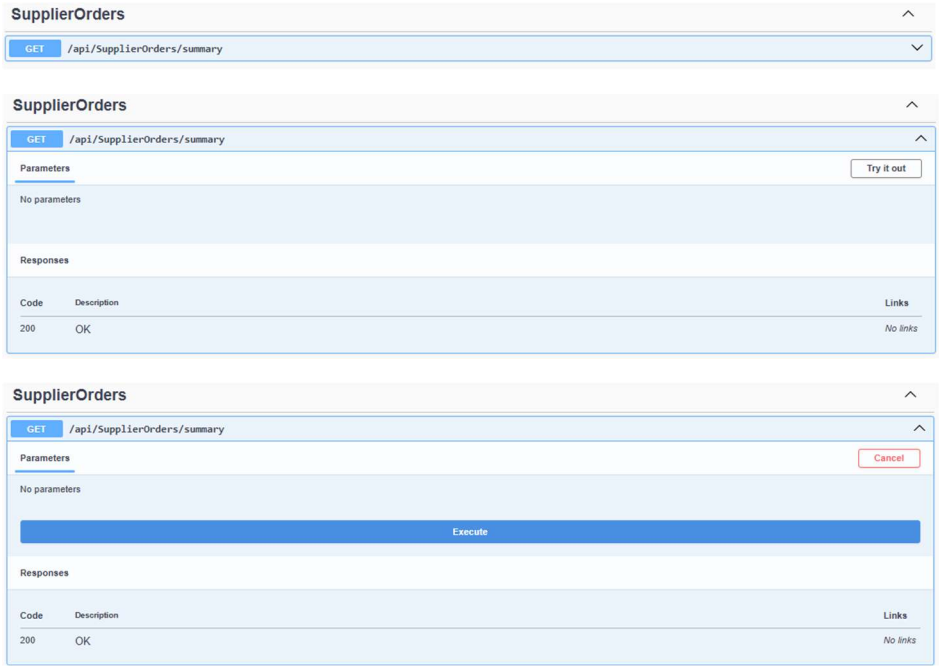




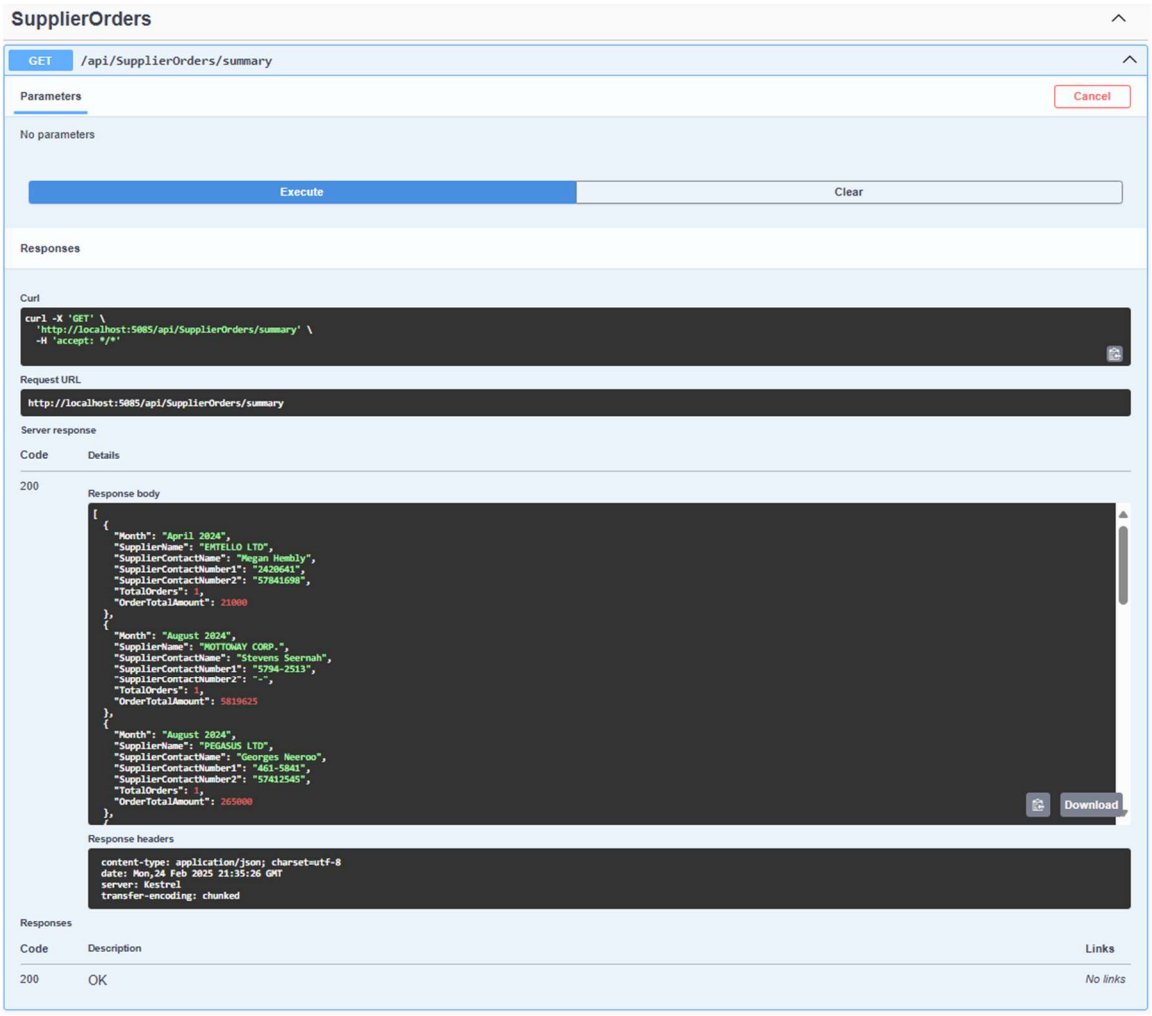
Provides a **summary report** of all purchase orders grouped by supplier region.



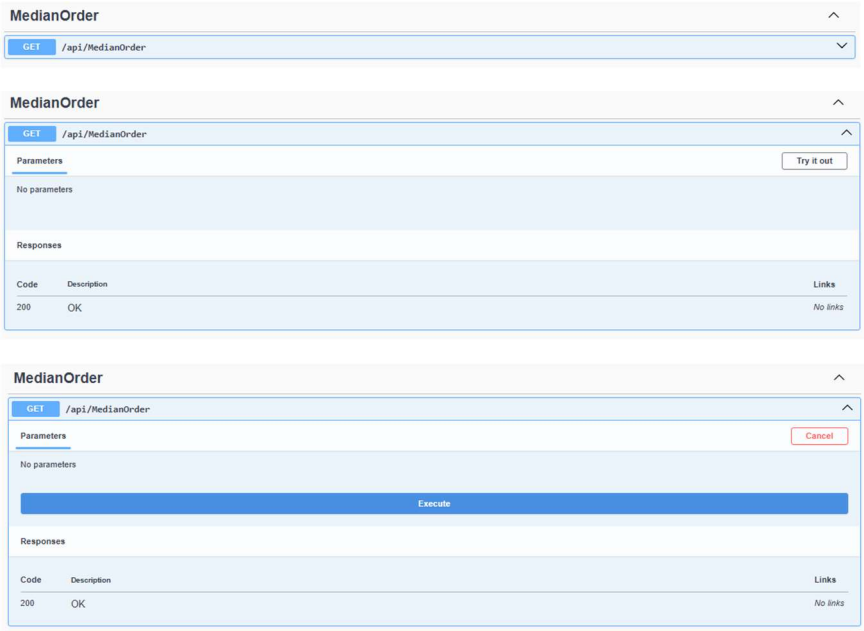
- **GET /api/SupplierOrders/summary** → Lists all suppliers with their total number of orders and amounts per month.



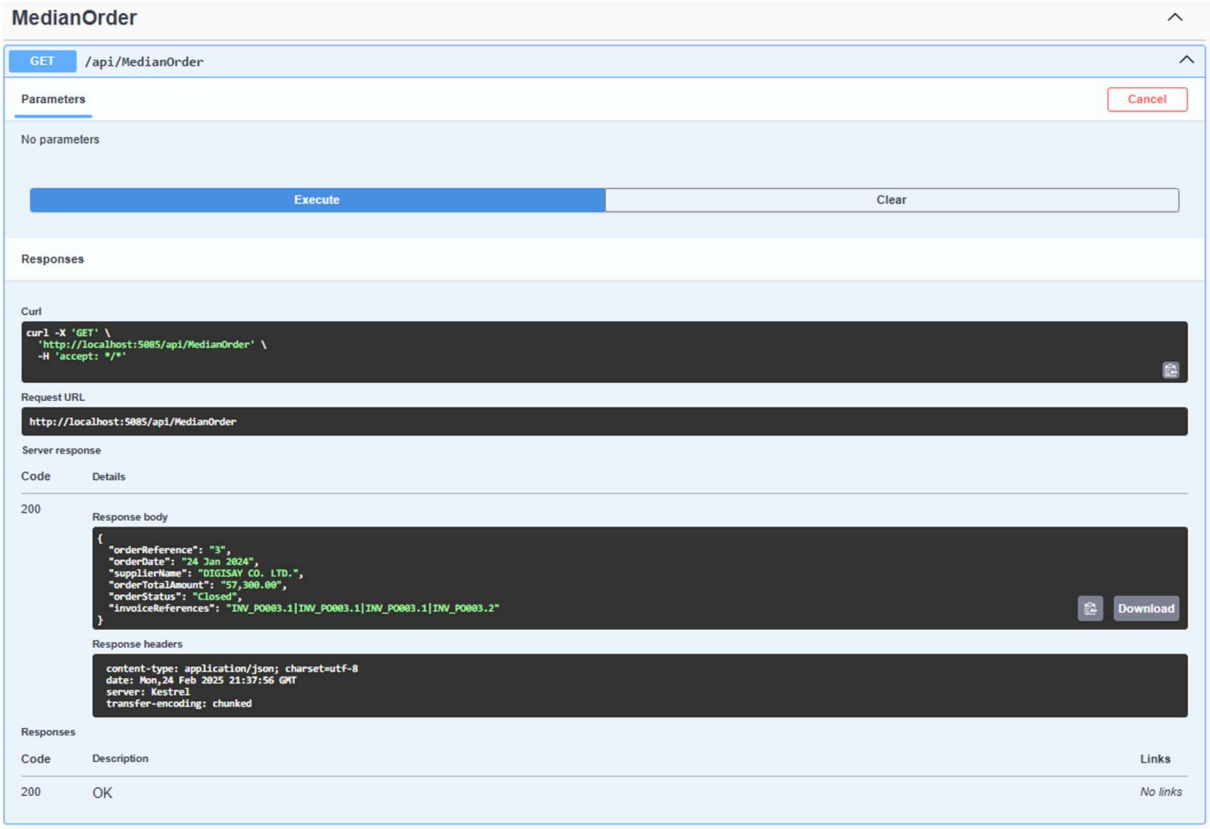
Lists all suppliers with their total number of orders and amounts per month.



- **GET /api/MedianOrder** → Fetches the **median order** based on the order total amount.



Fetches the **median order** based on the order total amount.



3. Database Schema and Migration

The raw **BCM_ORDER_MGT** table was **normalized** into multiple relational tables:

3.1 Table Structure

1. **Suppliers** (Suppliers Table)
    - Stores supplier details, including name, contact information, and location.
2. **Orders** (Orders Table)
    - Stores purchase order details, linked to **Suppliers** via SupplierID.
3. **Invoices** (Invoices Table)
    - Stores invoice details, linked to **Orders** via OrderID.

3.2 Data Migration

A **PL/SQL Procedure** was developed to extract, clean, and store the normalized data.

4. Stored Procedures for Reporting

4.1 Orders Summary Report (sp_ReportOrdersSummary)

This procedure extracts and formats purchase order data per **supplier region** with:
Order Reference (removes PO prefix)
Order Period (YYYY-MM)
Formatted Amounts (99,999,990.00)
Invoice Aggregation (STRING_AGG(InvoiceReference, '|'))
Action Field (To verify, To follow up, No Action)

4.2 Median Order Calculation (sp_ReportMedianOrder)

Fetches **the median purchase order** based on total amount, with:
Uppercased Supplier Name
Date Formatting (DD-MMM-YYYY)
Pipe-delimited Invoice References

4.3 Supplier Orders Summary (sp_ReportSupplierOrdersByMonth)

Extracts monthly order totals for suppliers:
Month (January 2024)
Supplier Contact Name & Numbers (formatted 5999-9999)
Total Orders & Total Order Amount

5. Security and Performance Considerations

5.1 Security Measures

- **JWT Authentication**: Protects endpoints from unauthorized access.
- **Role-Based Access Control** (RBAC): Ensures restricted access.
- **SQL Injection Prevention**: Parameterized queries used in stored procedures.
- **Data Validation**: Input sanitization for secure data handling.

5.2 Performance Optimization

- **Indexes on Foreign Keys (SupplierID, OrderID)**
- **Stored Procedures Instead of Ad-Hoc Queries**
- **Efficient Aggregation Using STRING_AGG()**

6. Tools and Technologies Used

- **Database**: MSSQL SQL
- **Backend Framework**: ASP.NET Core
- **Authentication**: JWT
- **Frontend**: Angular (for dashboard visualization)
- **Version Control**: GitHub (https://github.com/Abdurrahman-gurib/MCB-Assignment-2025)
- **Testing**: Unit Testing for APIs and Database Queries