.NET Technology

(503112)

By Mai Van Manh

Lab 09-10: ASP.NET WEB API

ASSIGNMENT DESCRIPTION

Construct a REST API for managing account information, product management, and orders. Detailed information is as follows:

- User accounts have details such as email and password.
- Products include details like product code, product name, price, illustration image, and description.
- Orders encompass information like order code, total selling price, and a list of products. Each element in the product list of an order contains information like product code, quantity, and price.

The REST API provides two endpoints, /products and /orders, supporting various HTTP methods to perform functions like adding, updating, deleting, listing, and retrieving details of products/orders. Additionally, there are /account/register and /account/login APIs to manage account registration and login. Some API endpoints require users to be logged in to access. Input parameters for all endpoints are in JSON format. Detailed descriptions of the API endpoints are as follows:

- http://localhost/api/account/register:
    - **POST**: Register a new account
- http://localhost/api/account/login:
    - **POST**: User logs in
- http://localhost/api/products:
    - **GET**: Retrieve a list of all products in the system
    - **POST**: Add a new product
- http://localhost/api/products/{id}
    - **GET**: Retrieve detailed information of a product based on its ID

    - **PUT**: Update information of a product based on its ID
    - **DELETE**: Delete a product based on its ID
- http://localhost/api/orders:
    - **GET**: Retrieve a list of all orders.
    - **POST**: Add a new order.
- http://localhost/api/orders/{id}:
    - **GET**: Retrieve detailed information of an order based on its ID
    - **PUT**: Update information of an order based on its ID
    - **DELETE**: Delete an order based on its ID

For API endpoint methods with highlighted red, users need to log in to access.

OTHER REQUIREMENTS

- All data related to accounts, products, and orders should be stored in MS SQL Server and accessed from NodeJS through the Entity Framework code first approach.
- Set up Cross-Origin Resource Sharing to allow any web client, even from different domains, to access and interact with the Web API.
- Use bcrypt to hash passwords, and use JWT for user authentication.
- Implement error handling mechanism and return appropriate error messages: for example, missing information, incorrect data format, oversized uploaded files, invalid endpoints, unsupported HTTP methods, etc.