Product Management System Documentation

1. Project Overview

The **Product Management System** is a Spring Boot application designed for managing product data with basic CRUD (Create, Read, Update, Delete) functionalities. It utilizes a microservices architecture and follows SOLID design principles, allowing for modularity and scalability. The application can also be run as a web application, providing an interactive user interface.

2. Key Components

- **Architecture**:
  - **Microservices**: The application is structured into independent services, each responsible for specific functionalities, enhancing scalability and maintainability.
  - **SOLID Principles**: The design follows SOLID principles, ensuring high cohesion and low coupling between components, making the system easier to manage and extend.
- **Model**: Represents the data structure of the application.
  - **Product**: A class representing a product with attributes such as `id`, `name`, `description`, and `price`.
- **Controller**: Manages the incoming HTTP requests and responses.
  - **ApiProductController**: A REST controller that provides endpoints for product operations.
  - **ProductController**: A controller that handles web requests related to product management.
- **Service**: Contains business logic.
  - A `ProductService` typically manages operations related to products, although it's not included in the provided files.
- **Repository**: Interfaces with the database to perform CRUD operations.
  - Generally, a `ProductRepository` interface would extend `JpaRepository`.

3. Functionality

- **Create Product**:
  - **Endpoint**: `POST /api/products`
  - **Description**: Adds a new product to the database.
  - **Request Body**: JSON representation of the product (name, description, price).
- **Read Products**:
  - **Endpoint**: `GET /api/products`
  - **Description**: Retrieves a list of all products.
  - **Response**: JSON array of products.
- **Read Product by ID**:
  - **Endpoint**: `GET /api/products/{id}`
  - **Description**: Fetches details of a specific product by its ID.
  - **Response**: JSON representation of the product.
- **Update Product**:
  - **Endpoint**: `PUT /api/products/{id}`
  - **Description**: Updates the details of an existing product.
  - **Request Body**: JSON representation of the product with updated fields.

- **Delete Product**:
  - **Endpoint**: `DELETE /api/products/{id}`
  - **Description**: Deletes a product by its ID.

4. Setup Instructions

1. **Clone the Repository**:

   Bash

   ```
   git clone https://github.com/tamzid68/spingboot_product_CRUD-
   ```

2. **Navigate to Project Directory**:

   Bash

   ```
   cd spingboot_product_CRUD-
   ```

3. **Build the Project**:

   Bash

   ```
   mvn clean install
   ```

4. **Run the Application**:

   Bash

   ```
   mvn spring-boot:run
   ```

5. **Access the Application**: Open a web browser and navigate to `http://localhost:8080/api/products`.

5. Deployment

- The application is deployed online and can be accessed at the following link: Product Management System Online

6. Postman Workspace and API Documentation

- **Postman Workspace**: You can access your collection in the public workspace at Postman Workspace.
- **Published API Documentation**: The API documentation can be viewed at API Documentation.

7. API Testing with Postman

- You can use Postman to test the API endpoints.
- Create requests corresponding to the endpoints listed above to perform CRUD operations.

- Make sure to set the request method (GET, POST, PUT, DELETE) correctly and include the necessary headers (e.g., `Content-Type: application/json`) when sending requests.

8. Future Enhancements

- Implement error handling and validation.
- Add user authentication and authorization.
- Introduce unit and integration tests.
- Enhance the UI with a frontend framework like React or Angular.

Name : ASM Tamzid
Phone : 01779078423
Email : mailto:atkthegreat9999@gmail.com
Linkedin : ASM Tamzid
Github : tamzid68