Inventory Management System

API-Models

Brief overview of the database models used in the **Inventory Management System**. Each model represents a table within the database and defines the structure of the data being managed.

1. **AppUser.cs**
   Manages user information within the system.
2. **AvailableProductInstance.cs**
   Represents individual instances of available products in the inventory.
3. **Customer.cs**
   Stores information about customers, including contact details and order history.
4. **InternalAuthentication.cs**
   Handles internal authentication processes for system users.
5. **Inventory.cs**
   Tracks the inventories available for each user.
6. **Product.cs**
   Defines the general properties of products, name and description.
7. **ProductType.cs**
   Categorizes products into specific types for organizational purposes.
8. **RestockingTransaction.cs**
   Logs transactions related to the restocking of inventory from suppliers.
9. **SalesTransaction.cs**
   Records sales transactions, including details of products sold and customer information.
10. **SoldProductInstance.cs**
    Represents individual instances of products that have been sold.
11. **Supplier.cs**
    Stores supplier details, tracking the sources of products for restocking.

API-DTOs

In addition to models, the project includes **DTOs (Data Transfer Objects)**, which are similar to models but specifically used to transfer data between layers or systems based on the needs of each transaction.

API-DATA DBContext

The **DbContext** serves as the bridge between the application's models and the database, allowing for querying and saving data. It manages the entity sets corresponding to the models and orchestrates the CRUD operations with the underlying database.

Functionality:

- Manages interactions between the application and the database, mapping models to database tables.
- Provides APIs to retrieve, update, and delete records from the database.
- Tracks changes to entities and ensures data consistency.

Key Components:

- **DbSet<TEntity>**: Represents a collection of entities mapped to a database table for each model, enabling LINQ queries and CRUD operations.
- **OnModelCreating()**: Configures relationships, constraints, and other database behaviors during the model creation phase.

API-Controllers

Introduction

Controllers in the **Inventory Management System** handle HTTP requests, coordinate interactions between the front-end and back-end, and manage the communication with the database through services or repositories. Each controller corresponds to a specific model or functionality, managing CRUD operations and business logic for different areas of the system.

Controllers

1. **AvailableProductsController.cs**
   Manages the retrieval and display of available products in the inventory.

2. **CustomerController.cs**
   Handles operations related to customer management, including creating, deleting, and retrieving customer data.

3. **InventoryController.cs**
   Manages inventory data, such as name and inventory details for each user.

4. **ProductController.cs**
   Handles CRUD operations for products, managing product creation, updates, and retrieval.

5. **ProductTypeController.cs**
   Manages product categories, allowing for the organization and retrieval of product types.

6. **RestockingTransactionController.cs**
   Handles restocking transactions, including logging and managing restock records from suppliers.

7. **SalesTransactionController.cs**
   Manages sales transactions, handling the creation and retrieval of sales data.

8. **SupplierController.cs**
   Manages supplier information and tracks suppliers involved in restocking the inventory.

9. **UsersController.cs**
   Handles user management, including authentication, and user creation.

Web Models

Introduction

The web models in the **Inventory Management System** are essentially the same as the models in the API. They are used to properly structure the data for making HTTP requests and interacting with the API. These models ensure that the client-side (Blazor Web) application can effectively communicate with the back-end API, manage the UI state, and perform data operations smoothly.

Functionality:

- Used to represent the same data structure as the API models, allowing for seamless data transfer.
- Facilitate the creation and management of HTTP requests (GET, POST, PUT, DELETE) to interact with the API.
- Ensure that the front-end consistently aligns with the data structures defined in the back-end.

Web Services

Introduction

The services in the **Inventory Management System** handle HTTP requests to interact with the API. Each service is responsible for managing data retrieval, updates, and other operations related to a specific model, ensuring communication between the front-end and back-end through API calls.

Services

1. **AvailableProductService.cs**
   Handles API requests for retrieving and managing available product data in the inventory.

2. **CustomerService.cs**
   Manages HTTP requests related to customer data, such as fetching and creating customer information.

3. **InventoryService.cs**
   Facilitates API communication for managing inventories of each user.

4. **ProductService.cs**
   Handles HTTP requests for managing product details, including creation, and retrieval.

5. **ProductTypeService.cs**
   Manages API interactions related to product categories, ensuring proper product classification.

6. **RestockingTransactionService.cs**
   Handles API requests for restocking transactions, allowing for the management of inventory restocks.

7. **SalesTransactionService.cs**
   Facilitates communication with the API for handling sales transactions, including creation and retrieval of sales records.

8. **SupplierService.cs**
   Manages supplier-related API requests, enabling the system to track suppliers and their transactions.

9. **UserService.cs**
   Handles user-related operations, including authentication, user creation through API requests.

Razor Components (Pages)

Introduction

The **Razor Components** in the **Inventory Management System** are Blazor components that utilize HTML, CSS, and C# code blocks (`@code`). These components are responsible for rendering the user interface, managing user input, and interacting with the back-end services via API calls.

Pages

1. **AddInventoryForm.razor**
   Displays a form to add new inventory records.

2. **AddProductInfo.razor**
   Provides a form for adding product information.

3. **AddRestockForm.razor**
   Handles input for restocking products, allowing users to submit restock transactions.

4. **AddSaleForm.razor**
   Presents a form to log sales transactions, capturing details of the sold products.

5. **AddSupplierForm.razor**
   Displays a form for adding supplier information.

6. **ConsumerView.razor**
   Renders a view for managing customer data, displaying customer-related operations. Also allows creation and deletion of sales transactions and displays all added transactions.

7. **CreateUser.razor**
   Handles user registration and allows new users to be created within the system.

8. **Dashboard.razor**
   Acts as the main dashboard, providing an overview of system and quick access to core features.

9. **Header.razor**
   Renders the header for the web app.

10. **Login.razor**
    Provides the login form, handling user authentication.

11. **ProductView.razor**
    Displays a list of products and their details, including options to add product name and product type,

12. **SupplyView.razor**
    Displays supplier information and allows for managing supplier-related data. Also displays restocking transactions and allows deletion and addition of restocking transactions