Project Technical Specification for “Retail Management Application Myshop”

---

1. Introduction:

This document provides an overview of a study project, focused on backend development. But also the project has frontend part for better back-to-front links and security flow understanding.
This document outlines the technical requirements and specifications for the development of a retail management application aimed at facilitating various processes within a retail store. The application will be built using Java 17, PostgreSQL, and Angular, following a monolithic architecture.

2. System Overview:

The application will provide functionality for inventory management, sales tracking, and authentication. It will consist of a backend, developed using Spring Framework, and a frontend, developed using Angular. The system will support SSL, OAuth2 for secure communication and RESTful architecture principles.

3. Functional Requirements:

3.1 Inventory Management:

- Track inventory levels, including product quantities, stock entries, and stock withdrawals.
- Support product categorization and grouping. Categories have only one level
- Implement stock entries (product supplies) and withdrawals (sales).
- Track suppliers
- Each supply invoice can be linked to corresponding supplier

3.2 Sales Management:

- Manage sales transactions through sales receipts.
- Track Customers: CRUD-operations
- Each sale receipt can be linked to a customer

- Generate sales reports and summaries.

3.3 User Authentication:

- Implement user authentication and authorization using Spring Security and Keycloak integration (OAuth2 standard).

4. Architecture, Stack:

The application will follow a monolithic architecture.

- Backend: Java 17 with Spring Framework, including Spring Boot and Spring Data JPA for data access, Spring Web for RESTful API
- Documentation: Swagger for API documentation, Javadoc for method description
- DataBase: PostgreSQL, Liquibase for data DDL and Test data and db-versions tracking
- Security: third-party authorization server Keycloak 18.0.0. Security design pattern “Backend for frontend” with server-side cookies. SSL.
- Mupstruct for DTO-management.
- Tests: JUnit 5 and Mockito.
- Frontend: Angular.

5. Data Persistence:

- Use PostgreSQL 15 as the relational database management system.
- Utilize Spring Data JPA for data access and manipulation.

6. Validation and Data Integrity:

- Implement data validation and integrity checks.
- Use custom validators on the backend for specific data validation requirements.

7. Data Seeding:

- Use Liquibase for database schema management.
- Seed initial data, including products, users, and other relevant information.

8. Security:

- Implement SSL for secure communication.
- Integrate Spring Security and Keycloak for user authentication and authorization.
- Keycloak security roles are converted to Spring Security roles by custom converter

- OAuth2 Security specification support.
- Backend for frontend pattern: middle layer application BFF proxies requests to backend, works with server-side cookies.

9. DTOs and Mappers:

Utilize MapStruct to handle data transfer objects (DTOs) and mapping between entities and DTOs.

10. Testing:

- Develop unit tests and integration tests for the backend.
- Use Junit 5 and Mockito for module testing.
- Implement testing strategies to ensure code quality and robustness.
- I don’t test controllers because of authentication issues.
- I make tests for services and repositories, validation.

11. Entity Descriptions:

- Product: Represents individual products, categorized by groups.
- ProductCategory: Represents product categories, one level.
- Customer: Represents store customers.
- Supplier: Represents product suppliers.
- User: Represents application users.
- Sale: Represents sales transactions.
- SaleItem: items (products) in sale document.
- Supply: Represents product supplies from suppliers.
- SupplyItem: items (products) in supply document.

12. User Interface:

- Implement a responsive and user-friendly UI using Angular.
- The frontend design was developed independently.

13. Deployment:

Deploy the application on a chosen server environment using Docker.

14. Documentation:

- Try to provide comprehensive documentation for installation, usage, and maintenance.

15. Project Management System:

Project tasks, kanban board and documentation will be made by Obsidian and corresponding plugins