# 《药店销售管理平台详细设计文档》

## 目录

1. 引言 1.1 目的 1.2 项目背景 1.3 术语和缩写
2. 系统架构详细设计 2.1 系统架构概述 2.2 技术选型 2.3 模块划分
3. 数据库详细设计 3.1 数据库概述 3.2 数据库表设计
4. 后端详细设计 4.1 项目结构 4.2 核心功能设计 4.3 API接口设计 4.4 业务逻辑设计
5. 前端详细设计 5.1 项目结构 5.2 核心功能设计 5.3 页面设计
6. 安全设计 6.1 身份验证与授权 6.2 数据加密 6.3 日志与监控
7. 部署与运维 7.1 部署环境 7.2 部署流程 7.3 运维管理

### 1. 引言

#### 1.1 目的

本文档详细描述了药店销售管理平台的设计细节，包括系统架构、模块设计、数据库设计、前后端设计、安全设计及部署与运维等内容。

#### 1.2 项目背景

为了提高药店的运营效率和管理水平，开发一套基于Spring Boot和Vue的药店销售管理平台，方便药店进行药品、销售、库存、供应商等业务的管理。

#### 1.3 术语和缩写

* **Spring Boot**：一种基于Java的开源框架，用于简化Spring应用的开发。
* **Vue**：一种渐进式JavaScript框架，用于构建用户界面。
* **POS**：销售点系统（Point of Sale）。
* **JWT**：JSON Web Token，一种用于安全传输信息的紧凑URL安全的令牌。

### 2. 系统架构详细设计

#### 2.1 系统架构概述

系统采用前后端分离的架构，前端使用Vue框架，后端使用Spring Boot框架，前后端通过RESTful API进行数据交互。系统架构图如下：

+----------------+ +----------------+  
| 前端(Vue) | <----> | 后端(Spring) |  
+----------------+ +----------------+  
 ^ |  
 | v  
 +-----------+ +-----------+  
 | 浏览器 | | 数据库 |  
 +-----------+ +-----------+

#### 2.2 技术选型

* **后端**：Spring Boot
* **前端**：Vue
* **数据库**：MySQL
* **API协议**：RESTful
* **安全**：Spring Security, JWT

#### 2.3 模块划分

系统主要分为以下几个模块： - **用户管理模块** - **药品管理模块** - **销售管理模块** - **库存管理模块** - **供应商管理模块** - **报告与分析模块** - **系统设置模块**

### 3. 数据库详细设计

#### 3.1 数据库概述

数据库采用MySQL，设计遵循第三范式，确保数据的一致性和完整性。

#### 3.2 数据库表设计

* **用户表（users）**
  + id: 主键
  + username: 用户名，非空，唯一，长度1-50字符
  + password: 密码，非空，长度8-50字符，必须包含字母和数字
  + role: 角色，非空，取值为”admin”或”user”
  + created\_at: 创建时间
  + updated\_at: 更新时间
* **药品表（medicines）**
  + id: 主键
  + name: 药品名称，非空，长度1-100字符
  + category: 药品类别，长度1-50字符
  + price: 药品价格，非空，精度为两位小数
  + description: 药品描述，长度0-1000字符
  + created\_at: 创建时间
  + updated\_at: 更新时间
* **销售记录表（sales）**
  + id: 主键
  + medicine\_id: 药品ID（外键）
  + quantity: 销售数量，非空，正整数
  + total\_price: 总价，非空，精度为两位小数
  + sale\_date: 销售日期，非空，格式为YYYY-MM-DD
  + created\_at: 创建时间
  + updated\_at: 更新时间
* **库存表（inventory）**
  + id: 主键
  + medicine\_id: 药品ID（外键）
  + quantity: 库存数量，非空，正整数
  + batch\_number: 批次号，非空，长度1-50字符
  + expiration\_date: 有效期，非空，格式为YYYY-MM-DD
  + created\_at: 创建时间
  + updated\_at: 更新时间
* **供应商表（suppliers）**
  + id: 主键
  + name: 供应商名称，非空，长度1-100字符
  + contact\_info: 联系信息，长度0-200字符
  + created\_at: 创建时间
  + updated\_at: 更新时间

### 4. 后端详细设计

#### 4.1 项目结构

后端项目结构如下：

pharmacy-management-platform/  
├── src/  
│ ├── main/  
│ │ ├── java/  
│ │ │ └── com/  
│ │ │ └── example/  
│ │ │ └── pharmacymanagement/  
│ │ │ ├── controller/  
│ │ │ ├── service/  
│ │ │ ├── repository/  
│ │ │ ├── model/  
│ │ │ └── config/  
│ │ └── resources/  
│ │ ├── application.properties  
│ │ └── static/  
│ └── test/  
├── pom.xml  
└── README.md

* **controller**：定义RESTful API控制器
* **service**：实现业务逻辑
* **repository**：定义数据访问接口
* **model**：定义数据模型
* **config**：配置Spring Boot应用

#### 4.2 核心功能设计

* **用户管理**：使用Spring Security进行身份验证，采用JWT进行用户会话管理。
* **药品管理**：实现药品的CRUD操作。
* **销售管理**：记录销售交易，生成销售报表。
* **库存管理**：管理药品库存，提供库存预警功能。
* **供应商管理**：管理供应商信息和采购订单。
* **报告与分析**：生成各种报表，提供数据分析功能。
* **系统设置**：配置系统参数，管理数据备份与恢复。

#### 4.3 API接口设计

**用户管理**

* POST /api/users/login：用户登录
  + 请求体：{ username: string, password: string }
  + 响应体：{ token: string, user: { id: number, username: string, role: string } }
  + 约束规则：
    - 用户名：非空，长度1-50字符。
    - 密码：非空，长度8-50字符。
* POST /api/users/register：用户注册
  + 请求体：{ username: string, password: string, role: string }
  + 响应体：{ id: number, username: string, role: string, created\_at: string, updated\_at: string }
  + 约束规则：
    - 用户名：非空，唯一，长度1-50字符。
    - 密码：非空，长度8-50字符，必须包含字母和数字。
    - 角色：非空，只能为”admin”或”user”。
* GET /api/users/profile：获取用户信息
  + 请求头：Authorization: Bearer <token>
  + 响应体：{ id: number, username: string, role: string, created\_at: string, updated\_at: string }
  + 约束规则：
    - 用户ID：非空，必须存在。

**药品管理**

* GET /api/medicines：获取药品列表
  + 请求头：Authorization: Bearer <token>
  + 响应体：[{ id: number, name: string, category: string, price: number, description: string, created\_at: string, updated\_at: string }]
  + 约束规则：
    - 用户ID：非空，必须存在。

## POST /api/medicines：创建药品

* 请求头：Authorization: Bearer <token>
  + 请求体：{ name: string, category: string, price: number, description: string }
  + 响应体：{ id: number, name: string, category: string, price: number, description: string, created\_at: string, updated\_at: string }
  + 约束规则：
    - 药品名称：非空，长度1-100字符。
    - 药品类别：长度1-50字符。
    - 药品价格：非空，精度为两位小数。
    - 药品描述：长度0-1000字符。
* GET /api/medicines/{id}：获取药品详情
  + 请求头：Authorization: Bearer <token>
  + 响应体：{ id: number, name: string, category: string, price: number, description: string, created\_at: string, updated\_at: string }
  + 约束规则：
    - 药品ID：非空，必须存在。
* PUT /api/medicines/{id}：更新药品
  + 请求头：Authorization: Bearer <token>
  + 请求体：{ name: string, category: string, price: number, description: string }
  + 响应体：{ id: number, name: string, category: string, price: number, description: string, created\_at: string, updated\_at: string }
  + 约束规则：
    - 药品名称：非空，长度1-100字符。
    - 药品类别：长度1-50字符。
    - 药品价格：非空，精度为两位小数。
    - 药品描述：长度0-1000字符。
* DELETE /api/medicines/{id}：删除药品
  + 请求头：Authorization: Bearer <token>
  + 响应体：{ message: string }
  + 约束规则：
    - 药品ID：非空，必须存在。

**销售管理**

* GET /api/sales：获取销售记录
  + 请求头：Authorization: Bearer <token>
  + 响应体：[{ id: number, medicine\_id: number, quantity: number, total\_price: number, sale\_date: string, created\_at: string, updated\_at: string }]
  + 约束规则：
    - 用户ID：非空，必须存在。
* POST /api/sales：记录销售交易
  + 请求头：Authorization: Bearer <token>
  + 请求体：{ medicine\_id: number, quantity: number, total\_price: number, sale\_date: string }
  + 响应体：{ id: number, medicine\_id: number, quantity: number, total\_price: number, sale\_date: string, created\_at: string, updated\_at: string }
  + 约束规则：
    - 药品ID：非空，必须存在。
    - 销售数量：非空，正整数。
    - 总价：非空，精度为两位小数。
    - 销售日期：非空，格式为YYYY-MM-DD。
* GET /api/sales/{id}：获取销售详情
  + 请求头：Authorization: Bearer <token>
  + 响应体：{ id: number, medicine\_id: number, quantity: number, total\_price: number, sale\_date: string, created\_at: string, updated\_at: string }
  + 约束规则：
    - 销售记录ID：非空，必须存在。

**库存管理**

* GET /api/inventory：获取库存信息
  + 请求头：Authorization: Bearer <token>
  + 响应体：[{ id: number, medicine\_id: number, quantity: number, batch\_number: string, expiration\_date: string, created\_at: string, updated\_at: string }]
  + 约束规则：
    - 用户ID：非空，必须存在。
* POST /api/inventory：添加库存
  + 请求头：Authorization: Bearer <token>
  + 请求体：{ medicine\_id: number, quantity: number, batch\_number: string, expiration\_date: string }
  + 响应体：{ id: number, medicine\_id: number, quantity: number, batch\_number: string, expiration\_date: string, created\_at: string, updated\_at: string }
  + 约束规则：
    - 药品ID：非空，必须存在。
    - 库存数量：非空，正整数。
    - 批次号：非空，长度1-50字符。
    - 有效期：非空，格式为YYYY-MM-DD。
* PUT /api/inventory/{id}：更新库存
  + 请求头：Authorization: Bearer <token>
  + 请求体：{ quantity: number, batch\_number: string, expiration\_date: string }
  + 响应体：{ id: number, medicine\_id: number, quantity: number, batch\_number: string, expiration\_date: string, created\_at: string, updated\_at: string }
  + 约束规则：
    - 库存ID：非空，必须存在。
    - 库存数量：非空，正整数。
    - 批次号：非空，长度1-50字符。
    - 有效期：非空，格式为YYYY-MM-DD。
* DELETE /api/inventory/{id}：删除库存
  + 请求头：Authorization: Bearer <token>
  + 响应体：{ message: string }
  + 约束规则：
    - 库存ID：非空，必须存在。

**供应商管理**

* GET /api/suppliers：获取供应商列表
  + 请求头：Authorization: Bearer <token>
  + 响应体：[{ id: number, name: string, contact\_info: string, created\_at: string, updated\_at: string }]
  + 约束规则：
    - 用户ID：非空，必须存在。
* POST /api/suppliers：创建供应商
  + 请求头：Authorization: Bearer <token>
  + 请求体：{ name: string, contact\_info: string }
  + 响应体：{ id: number, name: string, contact\_info: string, created\_at: string, updated\_at: string }
  + 约束规则：
    - 供应商名称：非空，长度1-100字符。
    - 联系信息：长度0-200字符。
* GET /api/suppliers/{id}：获取供应商详情
  + 请求头：Authorization: Bearer <token>
  + 响应体：{ id: number, name: string, contact\_info: string, created\_at: string, updated\_at: string }
  + 约束规则：
    - 供应商ID：非空，必须存在。
* PUT /api/suppliers/{id}：更新供应商
  + 请求头：Authorization: Bearer <token>
  + 请求体：{ name: string, contact\_info: string }
  + 响应体：{ id: number, name: string, contact\_info: string, created\_at: string, updated\_at: string }
  + 约束规则：
    - 供应商名称：非空，长度1-100字符。
    - 联系信息：长度0-200字符。
* DELETE /api/suppliers/{id}：删除供应商
  + 请求头：Authorization: Bearer <token>
  + 响应体：{ message: string }
  + 约束规则：
    - 供应商ID：非空，必须存在。

**报告与分析**

* GET /api/reports/sales：获取销售报表
  + 请求头：Authorization: Bearer <token>
  + 响应体：[{ date: string, total\_sales: number, total\_revenue: number }]
  + 约束规则：
    - 用户ID：非空，必须存在。
* GET /api/reports/inventory：获取库存报表
  + 请求头：Authorization: Bearer <token>
  + 响应体：[{ medicine\_id: number, name: string, category: string, quantity: number }]
  + 约束规则：
    - 用户ID：非空，必须存在。
* GET /api/reports/financial：获取财务报表
  + 请求头：Authorization: Bearer <token>
  + 响应体：[{ date: string, total\_sales: number, total\_revenue: number, total\_cost: number }]
  + 约束规则：
    - 用户ID：非空，必须存在。

#### 4.4 业务逻辑设计

**用户管理**

* **用户注册**：
  + 校验用户名是否唯一，密码是否符合复杂度要求。
  + 约束规则：
    - 用户名：非空，唯一，长度1-50字符。
    - 密码：非空，长度8-50字符，必须包含字母和数字。
* **用户登录**：
  + 验证用户名和密码 是否匹配，生成JWT令牌。
  + 约束规则：
    - 用户名：非空，长度1-50字符。
    - 密码：非空，长度8-50字符。

**药品管理**

* **药品创建**：
  + 校验药品名称是否唯一，价格是否合理。
  + 约束规则：
    - 药品名称：非空，长度1-100字符。
    - 药品类别：长度1-50字符。
    - 药品价格：非空，精度为两位小数。
    - 药品描述：长度0-1000字符。
* **药品更新**：
  + 校验药品ID是否存在，更新药品信息。
  + 约束规则：
    - 药品名称：非空，长度1-100字符。
    - 药品类别：长度1-50字符。
    - 药品价格：非空，精度为两位小数。
    - 药品描述：长度0-1000字符。

**销售管理**

* **记录销售交易**：
  + 校验药品ID和库存数量是否充足，更新库存数量。
  + 约束规则：
    - 药品ID：非空，必须存在。
    - 销售数量：非空，正整数。
    - 总价：非空，精度为两位小数。
    - 销售日期：非空，格式为YYYY-MM-DD。
* **生成销售报表**：
  + 按日、月、年汇总销售数据。
  + 约束规则：
    - 用户ID：非空，必须存在。

**库存管理**

* **添加库存**：
  + 校验药品ID是否存在，批次号和有效期是否正确。
  + 约束规则：
    - 药品ID：非空，必须存在。
    - 库存数量：非空，正整数。
    - 批次号：非空，长度1-50字符。
    - 有效期：非空，格式为YYYY-MM-DD。
* **更新库存**：
  + 校验库存ID是否存在，更新库存数量和批次信息。
  + 约束规则：
    - 库存ID：非空，必须存在。
    - 库存数量：非空，正整数。
    - 批次号：非空，长度1-50字符。
    - 有效期：非空，格式为YYYY-MM-DD。
* **库存预警**：
  + 检查库存数量是否低于预设阈值，发送预警通知。
  + 约束规则：
    - 药品ID：非空，必须存在。

**供应商管理**

* **供应商创建**：
  + 校验供应商名称是否唯一。
  + 约束规则：
    - 供应商名称：非空，长度1-100字符。
    - 联系信息：长度0-200字符。
* **更新供应商信息**：
  + 校验供应商ID是否存在，更新供应商信息。
  + 约束规则：
    - 供应商名称：非空，长度1-100字符。
    - 联系信息：长度0-200字符。

**报告与分析**

* **生成销售报表**：
  + 按指定时间段汇总销售数据。
  + 约束规则：
    - 用户ID：非空，必须存在。
* **生成库存报表**：
  + 统计各药品的库存数量和批次信息。
  + 约束规则：
    - 用户ID：非空，必须存在。
* **生成财务报表**：
  + 按指定时间段汇总销售收入和成本数据。
  + 约束规则：
    - 用户ID：非空，必须存在。

### 5. 前端详细设计

#### 5.1 项目结构

前端项目结构如下：

pharmacy-management-frontend/  
├── public/  
│ ├── index.html  
├── src/  
│ ├── assets/  
│ ├── components/  
│ ├── views/  
│ ├── router/  
│ ├── store/  
│ ├── App.vue  
│ ├── main.js  
├── package.json  
└── README.md

* **assets**：存放静态资源，如图片、样式等。
* **components**：存放可复用的Vue组件。
* **views**：存放各个页面组件。
* **router**：配置路由信息。
* **store**：配置Vuex状态管理。

#### 5.2 核心功能设计

* **用户界面**：提供用户登录、注册及信息管理功能。
* **药品管理界面**：展示药品列表，支持药品的CRUD操作。
* **销售管理界面**：记录销售交易，查看销售记录。
* **库存管理界面**：管理药品库存，查看库存状态。
* **供应商管理界面**：管理供应商信息，查看采购订单。
* **报告与分析界面**：展示销售、库存及财务报表。
* **系统设置界面**：配置系统参数，管理数据备份与恢复。

#### 5.3 页面设计

**登录页面（Login.vue）** - **功能**：用户登录入口。 - 输入用户名和密码进行登录，前端验证输入格式，后端返回JWT令牌。 - 约束规则： - 用户名：非空，长度1-50字符。 - 密码：非空，长度8-50字符。

**注册页面（Register.vue）** - **功能**：用户注册入口。 - 输入用户名、密码和角色进行注册，前端验证输入格式，后端处理注册逻辑。 - 约束规则： - 用户名：非空，唯一，长度1-50字符。 - 密码：非空，长度8-50字符，必须包含字母和数字。 - 角色：非空，只能为”admin”或”user”。

**药品列表页面（MedicineList.vue）** - **功能**：展示药品列表，支持药品的CRUD操作。 - 展示药品名称、类别、价格等信息，提供新增、编辑和删除药品的功能。 - 约束规则： - 药品名称：非空，长度1-100字符。 - 药品类别：长度1-50字符。 - 药品价格：非空，精度为两位小数。 - 药品描述：长度0-1000字符。

**药品详情页面（MedicineDetail.vue）** - **功能**：展示药品详细信息及操作入口。 - 展示药品的详细信息，支持编辑和删除操作。 - 约束规则： - 药品名称：非空，长度1-100字符。 - 药品类别：长度1-50字符。 - 药品价格：非空，精度为两位小数。 - 药品描述：长度0-1000字符。

**销售记录页面（Sales.vue）** - **功能**：记录销售交易，查看销售记录。 - 输入销售药品信息，记录销售交易，展示销售记录。 - 约束规则： - 药品ID：非空，必须存在。 - 销售数量：非空，正整数。 - 总价：非空，精度为两位小数。 - 销售日期：非空，格式为YYYY-MM-DD。

**库存管理页面（Inventory.vue）** - **功能**：管理药品库存，查看库存状态。 - 展示药品库存信息，支持新增、编辑和删除库存记录。 - 约束规则： - 药品ID：非空，必须存在。 - 库存数量：非空，正整数。 - 批次号：非空，长度1-50字符。 - 有效期：非空，格式为YYYY-MM-DD。

**供应商管理页面（Supplier.vue）** - **功能**：管理供应商信息，查看采购订单。 - 展示供应商信息，支持新增、编辑和删除供应商。 - 约束规则： - 供应商名称：非空，长度1-100字符。 - 联系信息：长度0-200字符。

**报告页面（Report.vue）** - **功能**：展示销售、库存及财务报表。 - 生成和展示销售报表、库存报表和财务报表，支持按时间段筛选和导出报表。 - 约束规则： - 用户ID：非空，必须存在。

**系统设置页面（Settings.vue）** - **功能**：配置系统参数，管理数据备份与恢复。 - 配置系统参数，执行数据备份和恢复操作。 - 约束规则： - 用户ID：非空，必须存在。

### 6. 安全设计

#### 6.1 身份验证与授权

* **Spring Security**：用于用户身份验证和授权管理。
* **JWT**：用于用户会话管理。

#### 6.2 数据加密

* **密码加密**：用户密码在存储前使用BCrypt进行加密。
* **数据传输 加密**：重要数据传输使用HTTPS进行加密。

#### 6.3 日志与监控

* **操作日志**：记录用户重要操作行为，包括登录、注册、药品管理、销售记录、库存管理等操作。
* **异常日志**：记录系统错误和异常信息，便于故障排除和问题追踪。
* **性能监控**：使用监控工具（如Prometheus和Grafana）监控系统性能和健康状态，及时发现和处理性能问题。

### 7. 部署与运维

#### 7.1 部署环境

* **操作系统**：Linux
* **应用服务器**：Tomcat
* **数据库服务器**：MySQL
* **前端服务器**：Nginx

#### 7.2 部署流程

1. **克隆项目代码库**
   * 使用Git克隆前后端项目代码库到服务器。
2. **后端部署**
   * 进入后端项目目录，运行mvn clean package命令构建后端项目并打包为WAR文件。
   * 将WAR文件部署到Tomcat服务器的webapps目录下。
   * 启动Tomcat服务器，确保后端服务正常运行。
3. **前端部署**
   * 进入前端项目目录，运行npm install命令安装前端项目依赖。
   * 运行npm run build命令构建前端项目，生成静态文件。
   * 将生成的静态文件部署到Nginx服务器的html目录下。
   * 配置Nginx，将请求代理到后端服务器。
   * 启动Nginx服务器，确保前端页面正常访问。
4. **配置数据库**
   * 创建MySQL数据库，导入数据库脚本初始化数据表结构。
   * 配置后端项目的数据库连接信息，确保后端服务能够正常连接数据库。

#### 7.3 运维管理

* **定期备份数据库**
  + 设置定时任务，定期备份数据库，确保数据安全。
  + 备份文件存储到安全的存储设备，防止数据丢失。
* **监控系统运行状态**
  + 使用监控工具（如Prometheus和Grafana）实时监控系统的性能和健康状态。
  + 设置告警规则，及时发现和处理系统异常和性能问题。
* **定期更新系统**
  + 根据需求和反馈定期更新系统，修复已知问题和漏洞。
  + 在更新前进行充分的测试，确保更新不会影响系统的正常运行。