

# **数据库系统原理课程设计报告**

## **茶馆销售仓库管理系统**

## 目 录

1 概述 .....	4
1.1 项目背景 .....	4
1.1 问题定义 .....	4
2 需求分析 .....	4
2.1 功能需求 .....	4
2.2 平台需求 .....	4
2.3 界面需求 .....	4
2.4 维护需求 .....	4
2.5 数据字典 .....	5
2.5.1 employee (员工) .....	5
2.5.2 customer (顾客) .....	5
2.5.3 card (会员卡) .....	5
2.5.4 service (服务) .....	6
2.5.5 product (产品) .....	6
2.5.6 purchase (进货) .....	6
2.5.7 shipment (出货) .....	6
2.5.8 expense_record (消费记录) .....	6
2.5.9 service_record (服务记录) .....	7
2.5.10 product_record (销货记录) .....	7
2.6 数据流图 .....	7
2.6.1 顶层数据流图 .....	7
2.6.2 增改查员工数据流图 .....	8
2.6.3 增改查顾客数据流图 .....	8
2.6.4 增删改查会员卡数据流图 .....	8
2.6.5 增删改查服务数据流图 .....	9
2.6.6 增删改查产品数据流图 .....	9
2.6.7 增删改查进货数据流图 .....	9
2.6.8 增删改查出货数据流图 .....	9
2.6.9 增删改查消费记录数据流图 .....	10
2.6.10 增删改查服务记录数据流图 .....	10
2.6.11 增删改查销货记录数据流图 .....	11
3 可行性分析 .....	11
3.1 市场可行性 .....	11
3.2 技术可行性 .....	11
3.3 经济可行性 .....	11
3.4 社会可行性 .....	11
4 数据库概念设计 .....	12
4.1 实体 .....	12
4.1.1 员工(employee)实体及局部 E-R 图 .....	12
4.1.2 顾客(customer)实体及局部 E-R 图 .....	12
4.1.3 会员卡(card)实体及局部 E-R 图 .....	13
4.1.4 服务(service)实体及局部 E-R 图 .....	13
4.1.5 产品(product)实体及局部 E-R 图 .....	13
4.1.6 进货(purchase)实体及局部 E-R 图 .....	14
4.1.7 出货(shipment)实体及局部 E-R 图 .....	14
4.1.8 消费记录(expense_record)实体及局部 E-R 图 .....	14
4.1.9 服务记录(service_record)实体及局部 E-R 图 .....	15
4.1.10 销货记录(product_record)实体及局部 E-R 图 .....	15

装  
订  
线

4.2 实体局部联系 E-R 图.....	16
4.2.1 顾客与会员卡 .....	16
4.2.2 产品与出货、产品与进货 .....	16
4.2.3 员工与出货、员工与进货 .....	16
4.2.4 员工与消费记录、员工与销货记录、员工与服务记录.....	16
4.2.5 顾客与销货记录、顾客与服务记录 .....	17
4.2.6 消费记录与销货记录、消费记录与服务记录.....	17
4.2.7 消费记录与会员卡 .....	17
4.2.8 服务记录与服务 .....	18
4.2.9 销货记录与服务 .....	18
4.3 全局 E-R 图.....	19
5 数据库逻辑设计 .....	19
5.1 关系模型 .....	19
5.1.1 E-R 图向关系模式的转换 .....	19
5.1.2 数据模型的优化及规范化设计验证 .....	20
5.2 表设计 .....	21
6 数据库物理设计 .....	21
7 应用系统设计 .....	21
7.1 技术路线 .....	21
7.2 模块划分 .....	22
7.3 详细设计 .....	24
8 实现过程 .....	29
8.1 数据库建立 .....	29
8.2 后端搭建 .....	32
8.2 前端搭建 .....	37
9 应用系统展示 .....	41
10 性能测试与改进 .....	49
11 系统维护 .....	51

装  
订  
线

装  
订  
线

## 1 概述

### 1.1 项目背景

茶馆自宋代开始兴起，是我国的传统行业之一。随着现代化的发展和茶馆行业连锁扩张的需求，茶馆需要更加科学化的销售、库存管理系统来帮助其进一步发展。茶馆的经营模式往往具有地方特色，比如基于“熟人”生意、产品价格波动频繁，常规的管理系统并不能很好的适应其管理销售模式。许多从业人员希望使用现代化的管理方式，帮助这一传统行业继续发展。

### 1.1 问题定义

基于茶馆行业特点提供方便的现代化销售仓库管理系统。用户可根据权限的不同，基于网页端进行数据库的操作与常见信息的查询。

## 2 需求分析

### 2.1 功能需求

系统要实现的主要功能有：

1. 提供系统操作员账户，保证用户可根据账号密码登录，并以依其身份，给与不同级别的权限。
2. 销售管理。拥有销售权限的用户可以对销售商品进行维护，增删改查消费记录、服务记录、销货记录、会员卡信息、顾客信息。
3. 库存管理。拥有仓库管理权限的用户可以管理进货、出货信息，对其进行增删改查。
4. 人员管理。拥有人员管理权限的用户可对顾客信息、员工信息进行增删改查；可以根据员工姓名查找与其相关的销货记录、服务记录、进货出货记录的部分信息；可以根据顾客姓名查找与之相关的消费记录会员卡等部分信息。
5. 定价管理。拥有定价权限的用户可以对产品信息、服务信息进行增删改查。

### 2.2 平台需求

在本地设备上运行，在 Web 中显示。

### 2.3 界面需求

1. 界面设计清晰简洁，用户可快速找到可以进行的操作。
2. 不同权限的用户可操作查看的内容不同，但整体上的界面布局风格保持一致。
3. 减少界面设计时造成的歧义，对用户的非法操作予以提示。

### 2.4 维护需求

1. 从数据角度，根据权限不同，用户可在界面内对数据库进行删改。
2. 从代码角度，要求程序最大限度上保留增添后续新功能添加的可行性；要求可读性强，

模块可分割性强，即在模块设计时遵循“高内聚低耦合”的原则。

## 2.5 数据字典

主码部分用下划线直线标出，外码用下划波浪线标出。

### 2.5.1 employee (员工)

属性名	字段	类型	长度	约束
编号	no	varchar	5	not null
姓名	name	varchar	20	not null
性别	sex	varchar	1	'f' 'm' <sup>[1]</sup> not null
是否在职	is_on_job	varchar	1	'y' 'n' <sup>[2]</sup> not null
电话	tel	varchar	11	
类型	type	varchar	20	
权限	permission	integer		not null, in range(0,5) <sup>[3]</sup>
密码	password	varchar	20	not null

[1] f:女, m:男

[2] y:是, n:否

[3] 0:系统管理员, 1:拥有全部权限(销售管理、库存管理、人员管理、定价管理), 2:拥有销售管理权限, 3:拥有库存管理权限 4:拥有人员管理权限 5:拥有定价管理权限

### 2.5.2 customer (顾客)

属性名	字段	类型	长度	约束
编号	no	varchar	6	not null
姓名	name	varchar	20	not null
性别	sex	varchar	1	'f' 'm' <sup>[1]</sup> not null
类型	type	varchar	1	'P' 'G' 'D' <sup>[4]</sup>

[4]P: 普通会员 (Priority Member), G: 金卡会员 (Gold Card Member), D: 钻石会员 (Diamond Card Member)

注: 此处的顾客仅指会员

### 2.5.3 card (会员卡)

属性名	字段	类型	长度	约束
卡号	card_no	varchar	10	not null
持卡人编号	customer_no	varchar	6	not null
余额	balance	float	(10,2)	>=0,not null

装  
订  
线

## 2.5.4 service (服务)

属性名	字段	类型	长度	约束
编号	no	varchar	10	not null
名称	name	varchar	30	not null
指导价	price	float	(10,2)	$\geq 0$ ,not null
备注	remarks	varchar	50	

## 2.5.5 product (产品)

属性名	字段	类型	长度	约束
编号	no	varchar	10	not null
名称	name	varchar	30	not null
店内现存量	number_in_shop	integer		$\geq 0$ ,not null
仓库剩余存货量	number_in_storehouse			$\geq 0$ ,not null
指导价	price	float	(10,2)	$\geq 0$ ,not null
备注	remarks	varchar	50	

## 2.5.6 purchase (进货)

属性名	字段	类型	长度	约束
产品编号	product_no	varchar	10	not null
批次	batch	varchar	5	not null
数量	all_number	integer		$\geq 0$ ,not null
负责员工编号	employee_no	varchar	5	not null
进货日期	purchase_date	date		
进价(单价)	price	float	(10,2)	$\geq 0$ ,not null
备注	remarks	varchar	50	

## 2.5.7 shipment (出货)

属性名	字段	类型	长度	约束
产品编号	product_no	varchar	10	not null
批次	batch	varchar	5	not null
数量	all_number	integer		$\geq 0$ ,not null
负责员工编号	employee_no	varchar	5	not null
出货日期	shipment_date	date		
备注	remarks	varchar	50	

## 2.5.8 expense\_record (消费记录)

属性名	字段	类型	长度	约束
编号	no	varchar	20	not null
操作员编号	employee_no	varchar	5	not null
购货记录编号	product_record_no	varchar	20	

装  
订  
线

服务记录编号	service_record_no	varchar	20	
应付金额	should_pay_amount	float	(10,2)	>=0,not null
实付金额	actual_pay_amount	float	(10,2)	>=0,not null
结算方式	pay_way	integer		range in (1,3) <sup>[4]</sup>
会员卡号	card_no	varchar	10	

[4] 1: 现金, 2: 信用卡, 3: 店内会员卡

购货记录编号和服务记录编号二者之一为空, 另一为非空。

### 2.5.9 service\_record (服务记录)

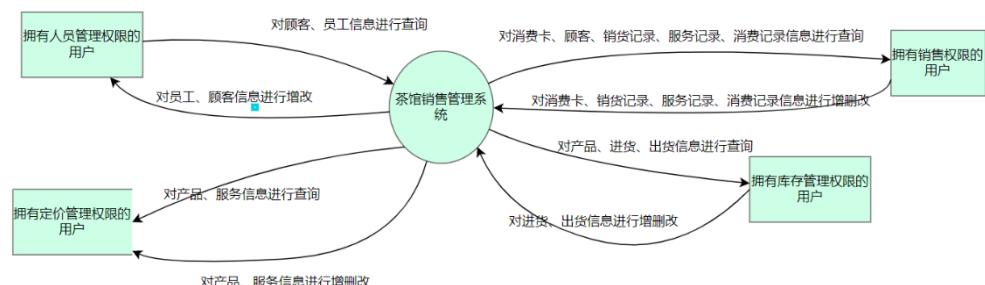
属性名	字段	类型	长度	约束
编号	no	varchar	20	not null
茶艺师编号	employee_no	varchar	5	not null
服务编号	service_no	varchar	10	not null
顾客编号	customer_no	varchar	6	

### 2.5.10 product\_record (销货记录)

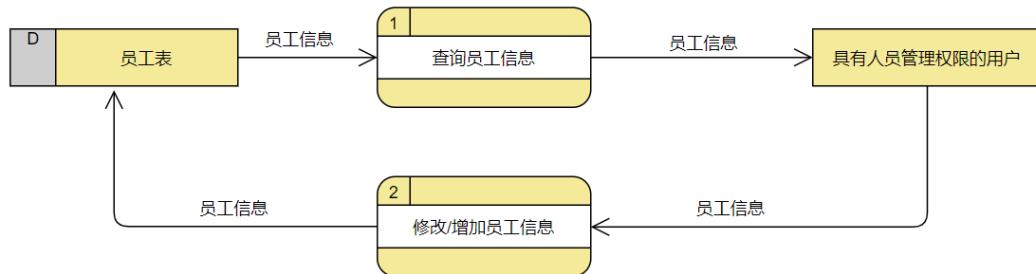
属性名	字段	类型	长度	约束
编号	no	varchar	20	not null
销售员编号	employee_no	varchar	5	not null
产品编号	product_no	varchar	10	not null
顾客编号	customer_no	varchar	6	

## 2.6 数据流图

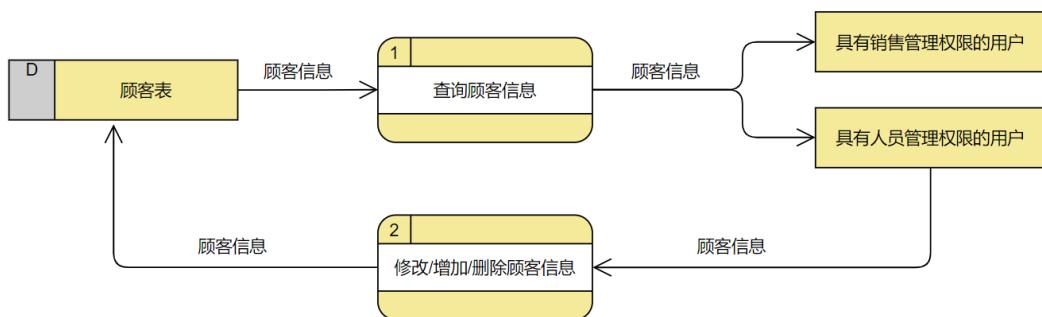
### 2.6.1 顶层数据流图



### 2.6.2 增改查员工数据流图

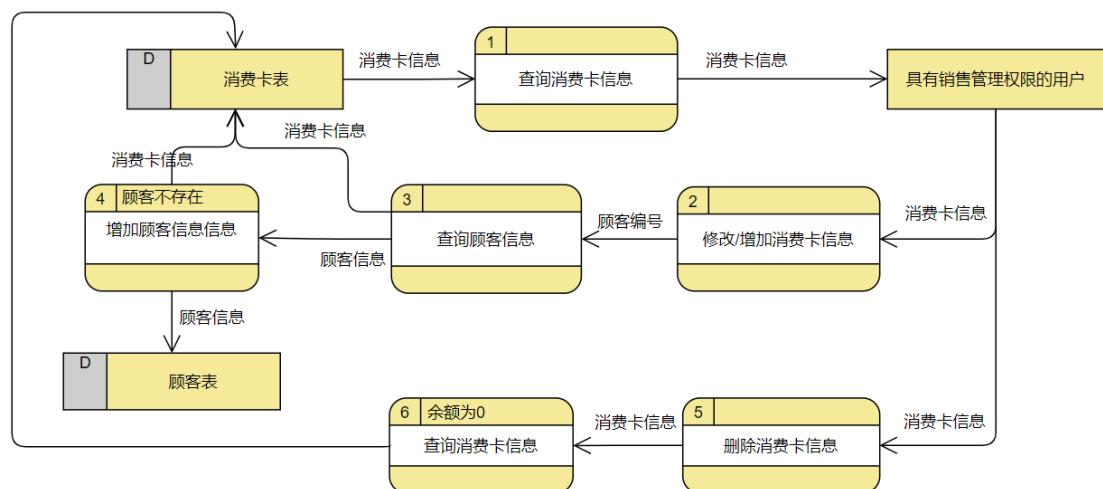


### 2.6.3 增改查顾客数据流图

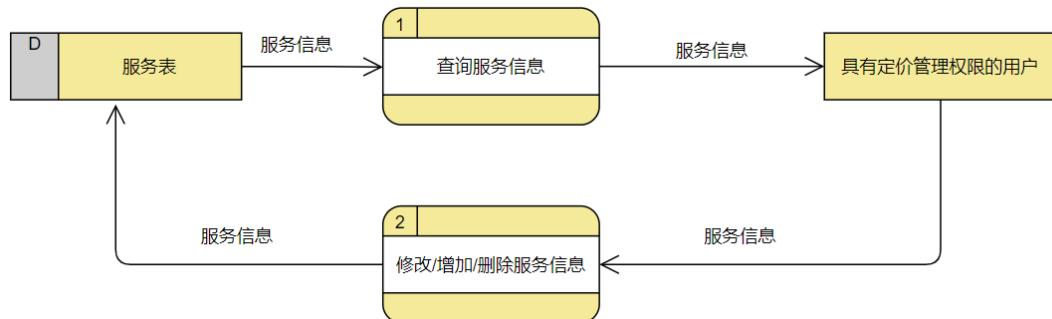


装  
订  
线

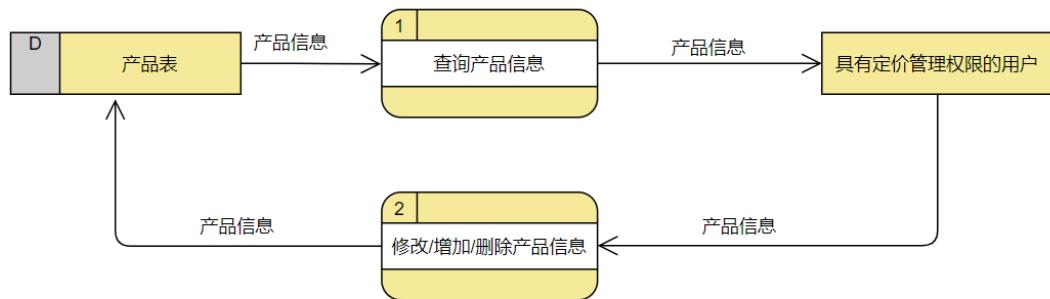
### 2.6.4 增删改查会员卡数据流图



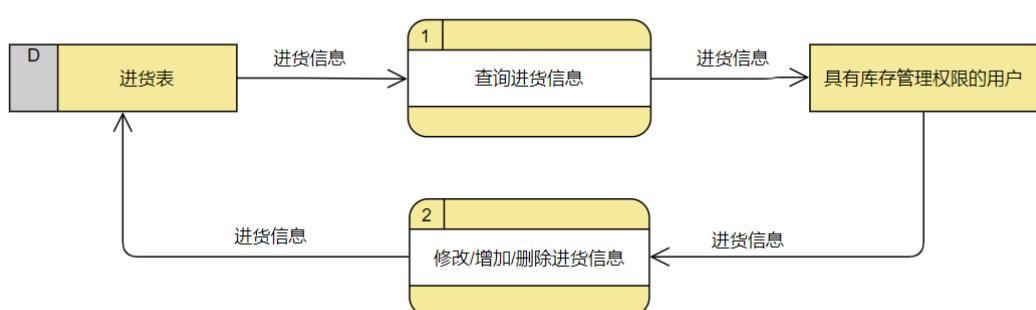
### 2.6.5 增删改查服务数据流图



### 2.6.6 增删改查产品数据流图

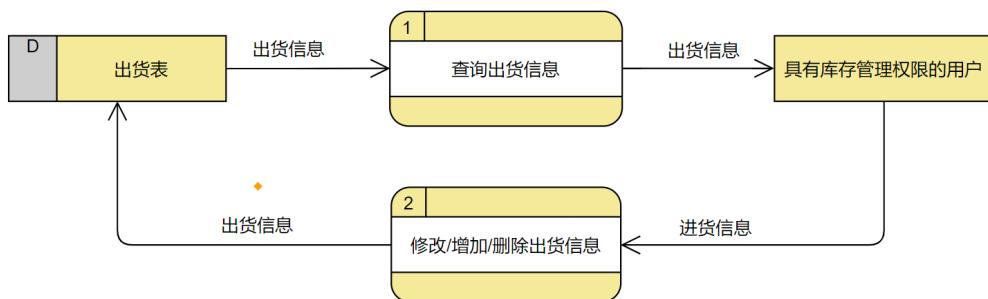


### 2.6.7 增删改查进货数据流图

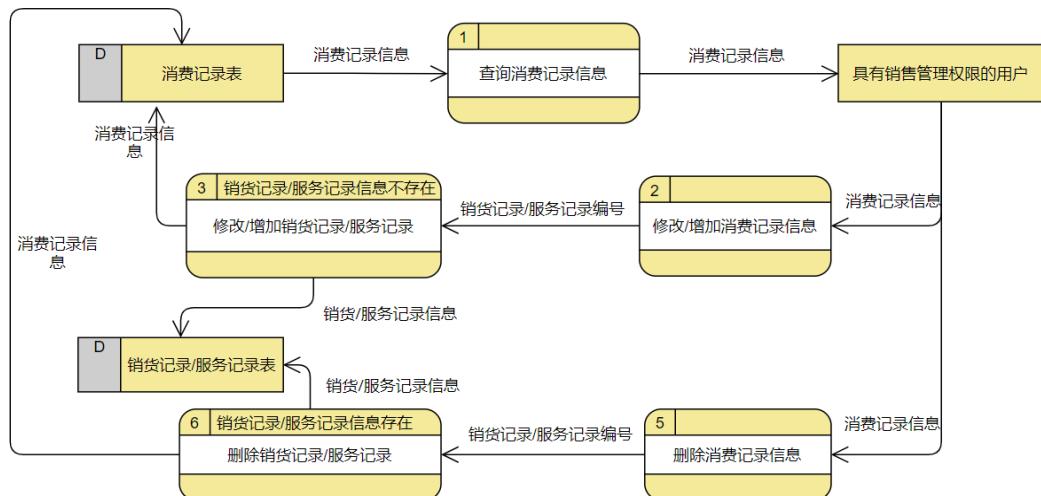


### 2.6.8 增删改查出货数据流图

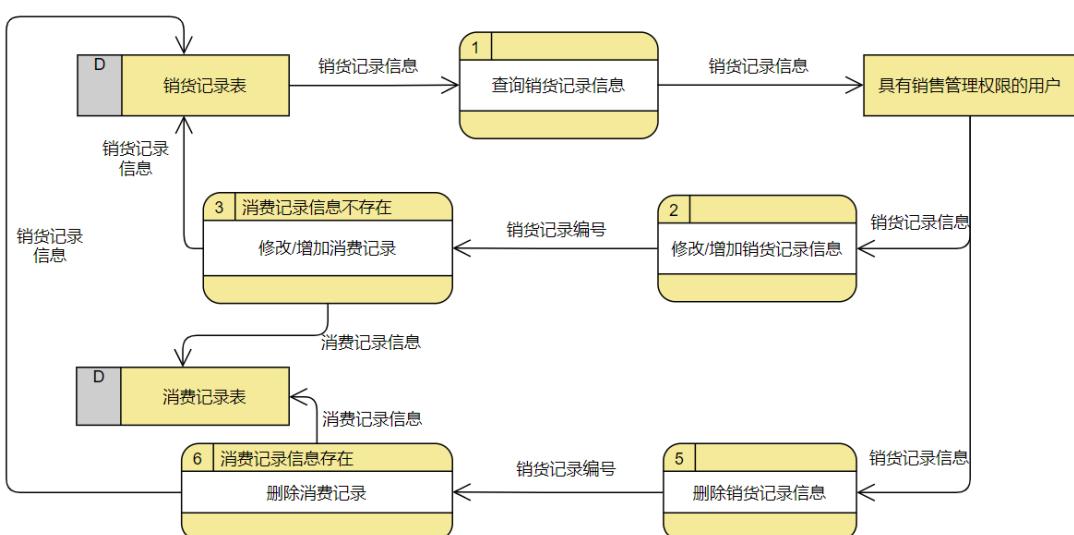
装  
订  
线



2.6.9 增删改查消费记录数据流图

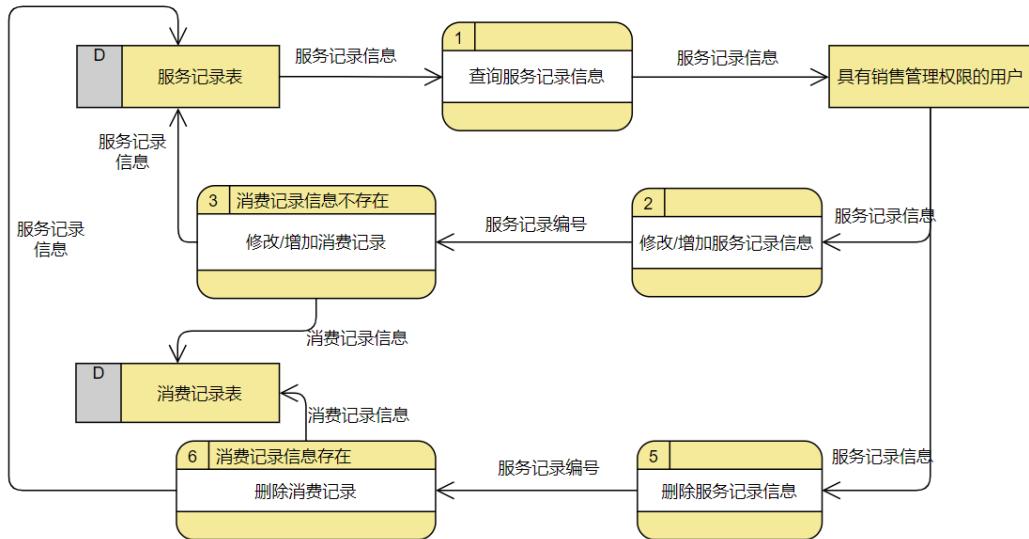


2.6.10 增删改查服务记录数据流图



装  
订  
线

### 2.6.11 增删改查销货记录数据流图



## 3 可行性分析

### 3.1 市场可行性

传统的进出货与销售管理采用手工记账的方式，费时费力。当前市场针对茶馆行业软件产品几乎没有，基本都是基于通用平台的直接移植。而常见的软件系统销售与仓储平台是分离的，不利于统一化的管理。市场上需要一款基于茶馆行业的销售仓储系统。

开发基于 Web 端，可兼容于不同操作系统，具有巨大的市场和发展潜力。由于各茶馆之间的数据不共享，将数据库部署在本地，保证数据的安全性。

### 3.2 技术可行性

采用主流的 React、SpringBoot 框架和 MySQL 数据库，相关的资料丰富，我也比较熟悉，有相关的开发经验。前后端分离开发，实现效率更高。

### 3.3 经济可行性

个人独立完成，投资少。

### 3.4 社会可行性

#### 3.4.1 法律可行性

该产品没有侵权或者抄袭等违法情况，也没有被申请过专利，故可行。

#### 3.4.2 政策可行性

无国家政策限制，也无地方政府（或其它机构）的限制

#### 3.4.3 使用可行性

由于茶馆行业从业者中很大部分是中年人，基于 Web 的客户端和简单的 UI 操作更便于使用。使用者无需学习繁杂的操作方法，且系统设计的查询常用，速度较快，具有良好

装

订

线

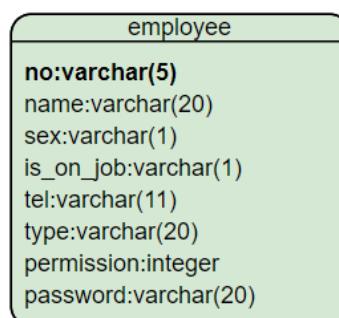
的交互性。设计完成后，使用者无需对数据库进行直接维护。

## 4 数据库概念设计

### 4.1 实体

#### 4.1.1 员工(employee)实体及局部 E-R 图

属性	字段名	补充说明
编号	no	员工的编号，长度必须为 5
姓名	name	
性别	sex	
是否在职	is_on_job	员工不可删除，若员工离职，标记为'n'，在职标记为'y'
电话	tel	
类型	type	如'saler'销售，'manager'经理，'warehouse staff'库房管理
权限	permission	每个员工所具有的对数据库的操作权限，不同类型可进行的操作权限不同
密码	password	登录数据库时所使用的密码

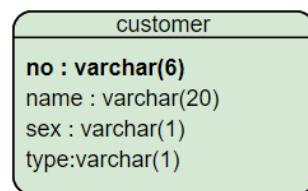


#### 4.1.2 顾客(customer)实体及局部 E-R 图

属性	字段名	补充说明
编号	no	顾客的编号，长度必须为 6，顾客不可删除
姓名	name	
性别	sex	
类型	type	顾客类型。P: 普通会员 (Priority Member), G: 金卡会员 (Gold Card Member), D: 钻石会员 (Diamond Card Member)

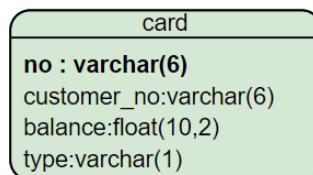
注：此处的顾客仅指会员

装  
订  
线

装  
订  
线

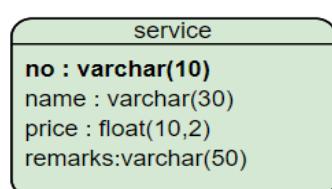
## 4.1.3 会员卡(card)实体及局部 E-R 图

属性	字段名	补充说明
卡号	card_no	长度必须为 10
持卡人编号	customer_no	
余额	balance	卡内剩余金额，一定大于等于 0



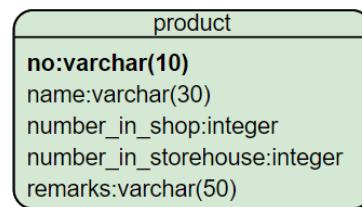
## 4.1.4 服务(service)实体及局部 E-R 图

属性	字段名	补充说明
编号	no	服务编号，长度必须为 10
名称	name	
指导价	price	即定价
备注	remarks	其他服务信息



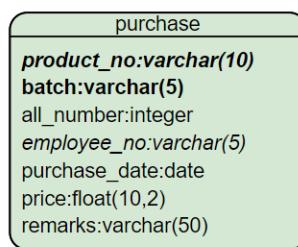
## 4.1.5 产品(product)实体及局部 E-R 图

属性	字段名	补充说明
编号	no	产品标号，长度必须为 10
名称	name	
店内现存量	number_in_shop	>=0,店内现有数量，注意与仓库剩余存货量区分
仓库剩余存货量	number_in_storehouse	>=0,库房现有数量，注意与店内现存量区分
指导价	price	即定价
备注	remarks	其他产品信息

装  
订  
线

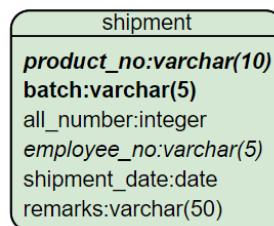
## 4.1.6 进货(purchase)实体及局部 E-R 图

属性	字段名	补充说明
<u>产品编号</u>	product_no	
<u>批次</u>	batch	
<u>数量</u>	all_number	>=0
<u>负责员工编号</u>	employee_no	
<u>进货日期</u>	purchase_date	yyyy-mm-dd 格式
<u>进价(单价)</u>	price	同一产品编号不同批次的进价可能有所不同
<u>备注</u>	remarks	



## 4.1.7 出货(shipment)实体及局部 E-R 图

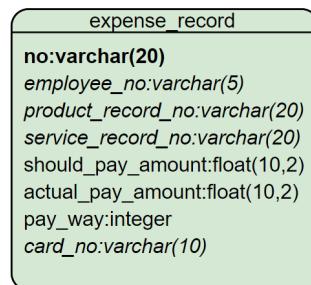
属性	字段名	补充说明
<u>产品编号</u>	product_no	
<u>批次</u>	batch	
<u>数量</u>	all_number	>=0
<u>负责员工编号</u>	employee_no	
<u>出货日期</u>	shipment_date	yyyy-mm-dd 格式
<u>备注</u>	remarks	



## 4.1.8 消费记录(expense\_record)实体及局部 E-R 图

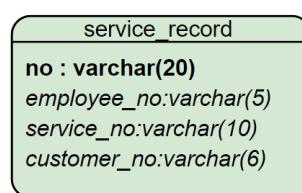
属性	字段名	补充说明

<u>编号</u>	no
<u>操作员编号</u>	employee_no
<u>购货记录编号</u>	product_record_no
<u>服务记录编号</u>	service_record_no
<u>应付金额</u>	should_pay_amount >=0, 单价*总量
<u>实付金额</u>	actual_pay_amount >=0, 实际付款, 因折扣等原因可与应付金额不同
<u>结算方式</u>	pay_way 结算方式为会员卡时, 必须记录会员卡号
<u>会员卡号</u>	card_no



4.1.9 服务记录(service\_record)实体及局部 E-R 图

属性	字段名	补充说明
<u>编号</u>	no	
<u>茶艺师编号</u>	employee_no	
<u>服务编号</u>	service_no	
<u>顾客编号</u>	customer_no	

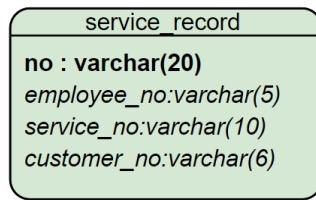


4.1.10 销货记录(product\_record)实体及局部 E-R 图

属性	字段名	补充说明
<u>编号</u>	no	
<u>销售员编号</u>	employee_no	
<u>产品编号</u>	product_no	
<u>顾客编号</u>	customer_no	

装订线

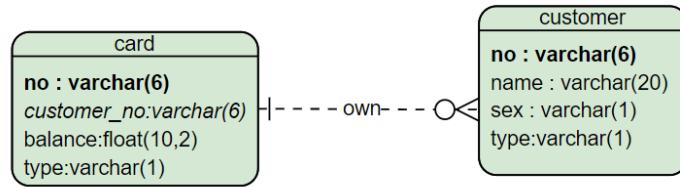
装  
订  
线



## 4.2 实体局部联系 E-R 图

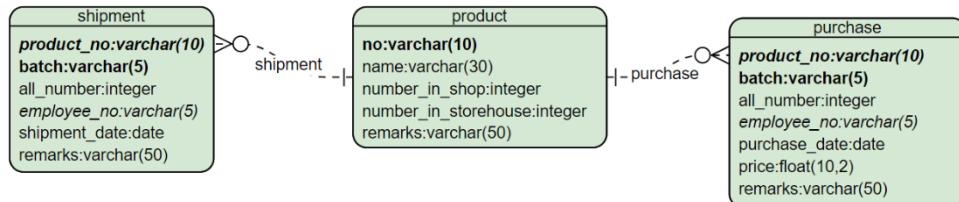
### 4.2.1 顾客与会员卡

一个顾客可以有多张会员卡，是一对多的关系



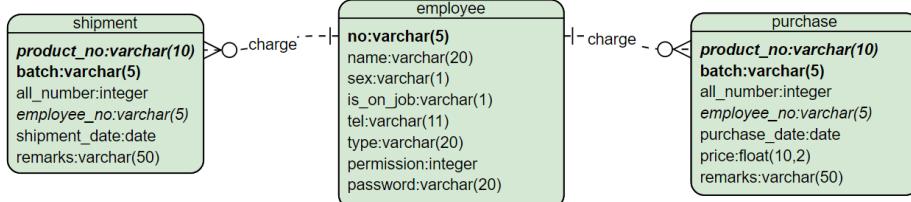
### 4.2.2 产品与出货、产品与进货

一个产品对应多个进货和出货，是一对多的关系



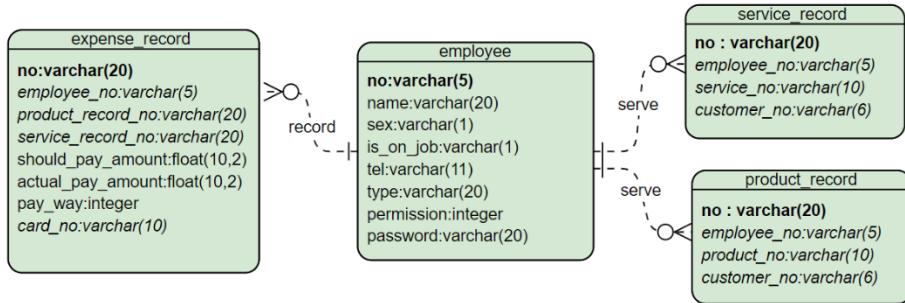
### 4.2.3 员工与出货、员工与进货

一个员工负责多次出货/进货，是一对多的关系



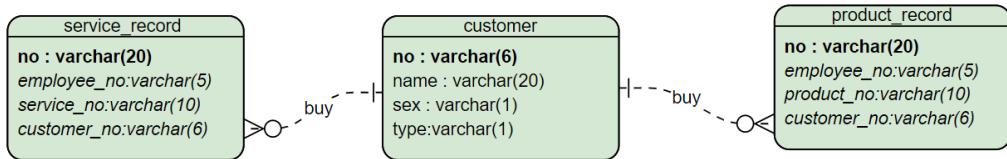
### 4.2.4 员工与消费记录、员工与销货记录、员工与服务记录

一个员工负责多条消费记录的记载，一个员工负责多次销售产品，一个员工负责多次服务，都是一对多的关系



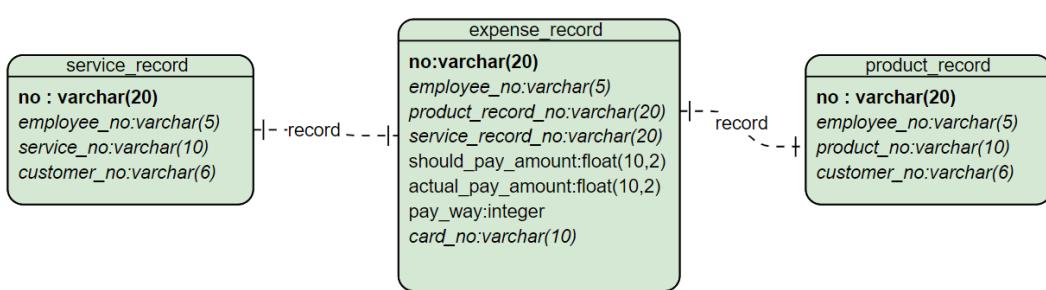
#### 4.2.5 顾客与销货记录、顾客与服务记录

一个顾客有多次消费，对应着多条销货记录和服务记录，是一对多的关系



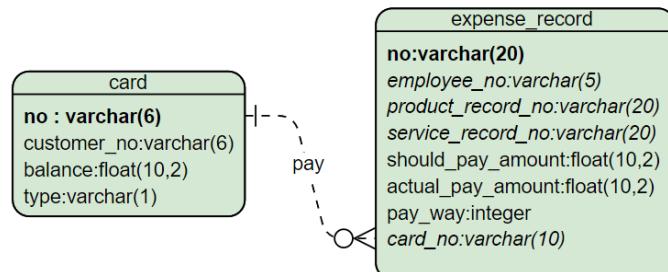
#### 4.2.6 消费记录与销货记录、消费记录与服务记录

一条消费记录对应一条销货记录或一条服务记录，是一对一的关系



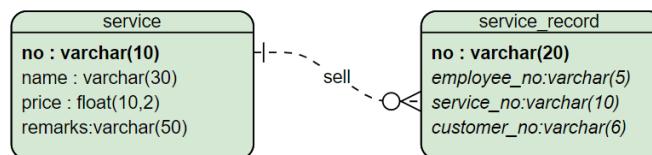
#### 4.2.7 消费记录与会员卡

一张会员卡对应多条消费记录，是一对多的关系



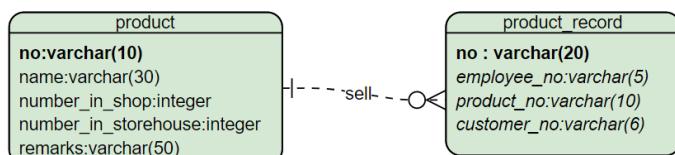
#### 4.2.8 服务记录与服务

一种服务对应多条服务记录，是一对多的关系



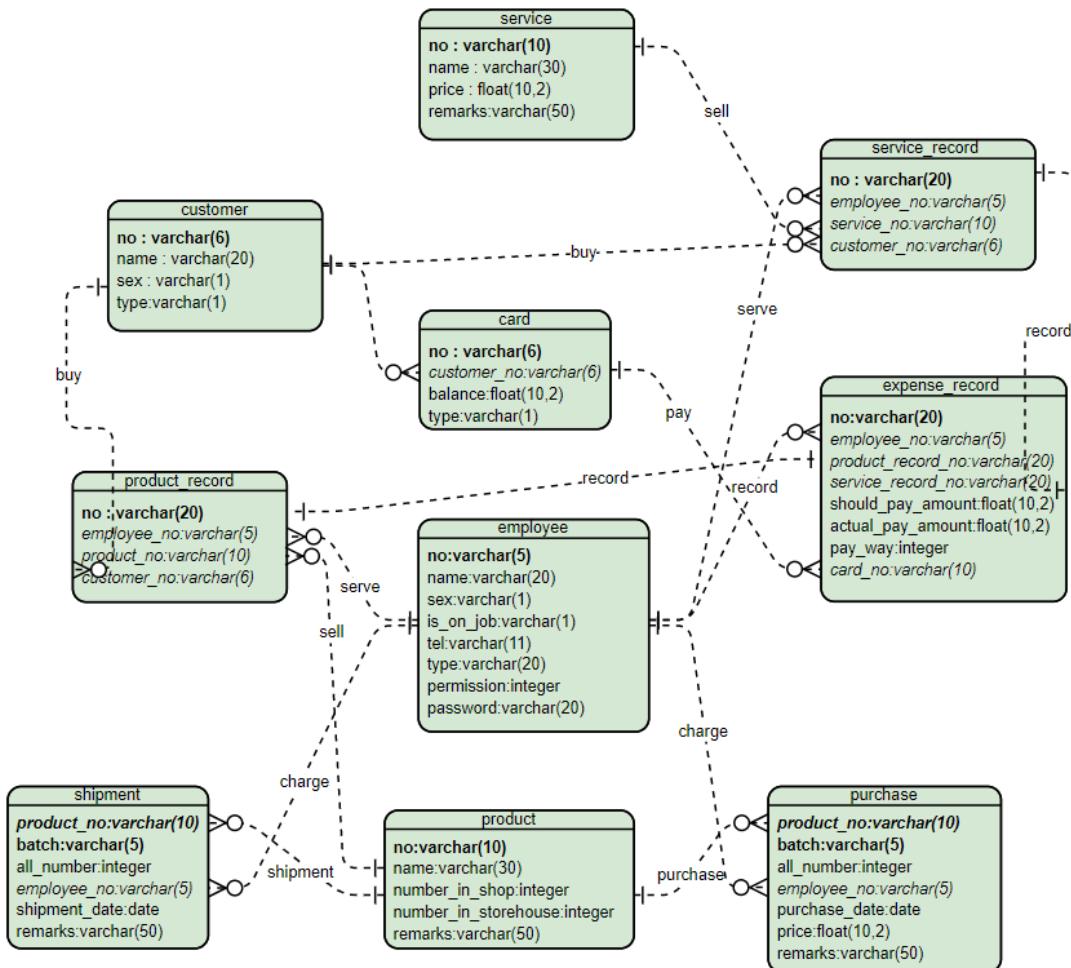
#### 4.2.9 销货记录与服务

一种产品对应多条销货记录，是一对多的关系



装  
订  
线

### 4.3 全局 E-R 图



装  
订  
线

## 5 数据库逻辑设计

### 5.1 关系模型

#### 5.1.1 E-R 图向关系模式的转换

转换得到的表结果如下：

1. employee (no, name, sex, is\_on\_job, tel, type, permission, password);
2. customer (no, name, sex, type);
3. card (card\_no, customer\_no, balance);
4. service (no, name, price, remarks);
5. product (no, name, number\_in\_shop, number\_in\_storehouse, price, remarks);
6. purchase (product\_no, batch, all\_number, employee\_no, purchase\_date, price, remarks);
7. shipment (product\_no, batch, all\_number, employee\_no, shipment\_date, remarks);
8. expense\_record (no, employee\_no, product\_record\_no, service\_record\_no, should\_pay\_amount, actual\_pay\_amount, pay\_way, card\_no);

9. service\_record (no, employee\_no, service\_no, customer\_no);
10. product\_record(no, employee\_no, product\_no, customer\_no);

### 5.1.2 数据模型的优化及规范化设计验证

设计要满足 BCNF。由于 5.1.1 中的设计不含多值属性(表中表嵌套)，满足第一范式；非主码属性都和数据表中的主码属性有完全依赖关系，满足第二范式；表中的数据元素之间相互独立，不存在除与主码间的其他的函数关系，满足第三范式。下验证 5.1.1 中设计的表满足 BCNF。

1. employee (no, name, sex, is\_on\_job, tel, type, permission, password);  
 $\text{no} \rightarrow \text{name}$ ;  $\text{no} \rightarrow \text{sex}$ ;  $\text{no} \rightarrow \text{is\_on\_job}$ ;  $\text{no} \rightarrow \text{tel}$ ;  $\text{no} \rightarrow \text{type}$ ;  $\text{no} \rightarrow \text{permission}$ ;  $\text{no} \rightarrow \text{password}$   
 $\text{no}$  为超码，关系属于 BCNF，无需再分解
2. customer (no, name, sex, type);  
 $\text{no} \rightarrow \text{name}$ ;  $\text{no} \rightarrow \text{sex}$ ;  $\text{no} \rightarrow \text{type}$   
 $\text{no}$  为超码，关系属于 BCNF，无需再分解
3. card (card\_no, customer\_no, balance);  
 $\text{card\_no} \rightarrow \text{customer\_no}$ ;  $\text{card\_no} \rightarrow \text{balance}$   
 $\text{card\_no}$  为超码，关系属于 BCNF，无需再分解
4. sevice (no, name, price, remarks);  
 $\text{no} \rightarrow \text{name}$ ;  $\text{no} \rightarrow \text{price}$ ;  $\text{no} \rightarrow \text{remarks}$   
 $\text{no}$  为超码，关系属于 BCNF，无需再分解
5. product (no, name, number\_in\_shop, number\_in\_storehouse, price, remarks);  
 $\text{no} \rightarrow \text{name}$ ;  $\text{no} \rightarrow \text{number\_in\_shop}$ ;  $\text{no} \rightarrow \text{number\_in\_storehouse}$ ;  $\text{no} \rightarrow \text{price}$ ;  $\text{no} \rightarrow \text{remarks}$   
 $\text{no}$  为超码，关系属于 BCNF，无需再分解
6. purchase (product\_no, batch, all\_number, employee\_no, purchase\_date, price, remarks);  
 $\text{product\_no}$ ,  $\text{batch} \rightarrow \text{all\_number}$ ;  $\text{product\_no}$ ,  $\text{batch} \rightarrow \text{employee\_no}$   
 $\text{product\_no}$ ,  $\text{batch} \rightarrow \text{purchase\_date}$ ;  $\text{product\_no}$ ,  $\text{batch} \rightarrow \text{price}$   
 $\text{product\_no}$ ,  $\text{batch} \rightarrow \text{remarks}$   
 $\text{product\_no}$ ,  $\text{batch}$  为超码，关系属于 BCNF，无需再分解
7. shipment (product\_no, batch, all\_number, employee\_no, shipment\_date, remarks);  
 $\text{product\_no}$ ,  $\text{batch} \rightarrow \text{all\_number}$ ;  $\text{product\_no}$ ,  $\text{batch} \rightarrow \text{employee\_no}$   
 $\text{product\_no}$ ,  $\text{batch} \rightarrow \text{shipment\_date}$ ;  $\text{product\_no}$ ,  $\text{batch} \rightarrow \text{remarks}$   
 $\text{product\_no}$ ,  $\text{batch}$  为超码，关系属于 BCNF，无需再分解
8. expense\_record (no, employee\_no, product\_record\_no, service\_record\_no,  
                        should\_pay\_amount, actual\_pay\_amount, pay\_way, card\_no);  
 $\text{no} \rightarrow \text{employee\_no}$ ;  $\text{no} \rightarrow \text{product\_record\_no}$ ;  $\text{no} \rightarrow \text{service\_record\_no}$   
 $\text{o} \rightarrow \text{should\_pay\_amount}$ ;  $\text{no} \rightarrow \text{actual\_pay\_amount}$ ;  $\text{no} \rightarrow \text{pay\_way}$ ;  $\text{no} \rightarrow \text{card\_no}$   
 $\text{no}$  为超码，关系属于 BCNF，无需再分解
9. service\_record (no, employee\_no, service\_no, customer\_no);  
 $\text{no} \rightarrow \text{employee\_no}$ ;  $\text{no} \rightarrow \text{service\_no}$ ;  $\text{no} \rightarrow \text{customer\_no}$   
 $\text{no}$  为超码，关系属于 BCNF，无需再分解
10. product\_record(no, employee\_no, product\_no, customer\_no);

$\text{no} \rightarrow \text{employee\_no}$ ;  $\text{no} \rightarrow \text{product\_no}$ ;  $\text{no} \rightarrow \text{customer\_no}$ ;  
 $\text{no}$  为超码, 关系属于 BCNF, 无需再分解

## 5.2 表设计

因 5.1.1 中设计表满足 BCNF, 所以最终设计与 3.1 中设计表相同。

## 6 数据库物理设计

索引设计:

从现实情况考虑, 员工表和服务表的数据量很小, 在“逻辑上”是可数的, 无需建立索引就可较快的增删改查。

对销货记录表、服务记录表、消费记录表、进货表、购货表这样数据量很大的表, 其查询需建立索引。除上述不建立索引的两表外, 其他表的查询往往是单行查询(如 `select * from card where customer_no='000001';`)、排序查询(`order by, group by`)混合的。对单行查询来说, 哈希索引增删改查的平均时间复杂度都是  $O(1)$ , 树的都是  $O(\lg(n))$ ; 但对于排序查询来说, 哈希型的索引, 时间复杂度会退化为  $O(n)$ , 而树型的“有序”特性, 依然能够保持  $O(\log(n))$  的高效率。基于此, 使用单行查询较为频繁, 使用哈希索引的方法; 使用排序查询更频繁, 使用 B+树作为索引。对一些查询不需要的属性, 如 `remark` 则不建立索引。

表名	字段名	索引类型
<b>employee</b>		无
<b>customer</b>	no, name, sex, type	哈希索引
<b>card</b>	card_no, customer_no balance	哈希索引 B+树索引
<b>service</b>		无
<b>product</b>	no, name, number_in_shop, number_in_storehouse, price	哈希索引 B+树索引
<b>purchase</b>	product_no, batch, employee_no, purchase_date, price, all_number,	哈希索引 B+树索引
<b>shipment</b>	product_no, batch, employee_no, all_number, shipment_date	哈希索引 B+树索引
<b>expense_record</b>	no, employee_no, product_record_no, service_record_no, pay_way, card_no should_pay_amount, actual_pay_amount,	哈希索引 B+树索引
<b>service_record</b>	no, employee_no, service_no, customer_no	哈希索引
<b>product_record</b>	no, employee_no, product_no, customer_no	哈希索引

## 7 应用系统设计

### 7.1 技术路线

采用前后端分离的开发。

前端基于 Web 展示, 使用 React 框架, MUI 开源模板, 用 JavaScript 语言编写

后端基于 SpringBoot 框架, 用 Java 语言编写

数据库选择 MySQL

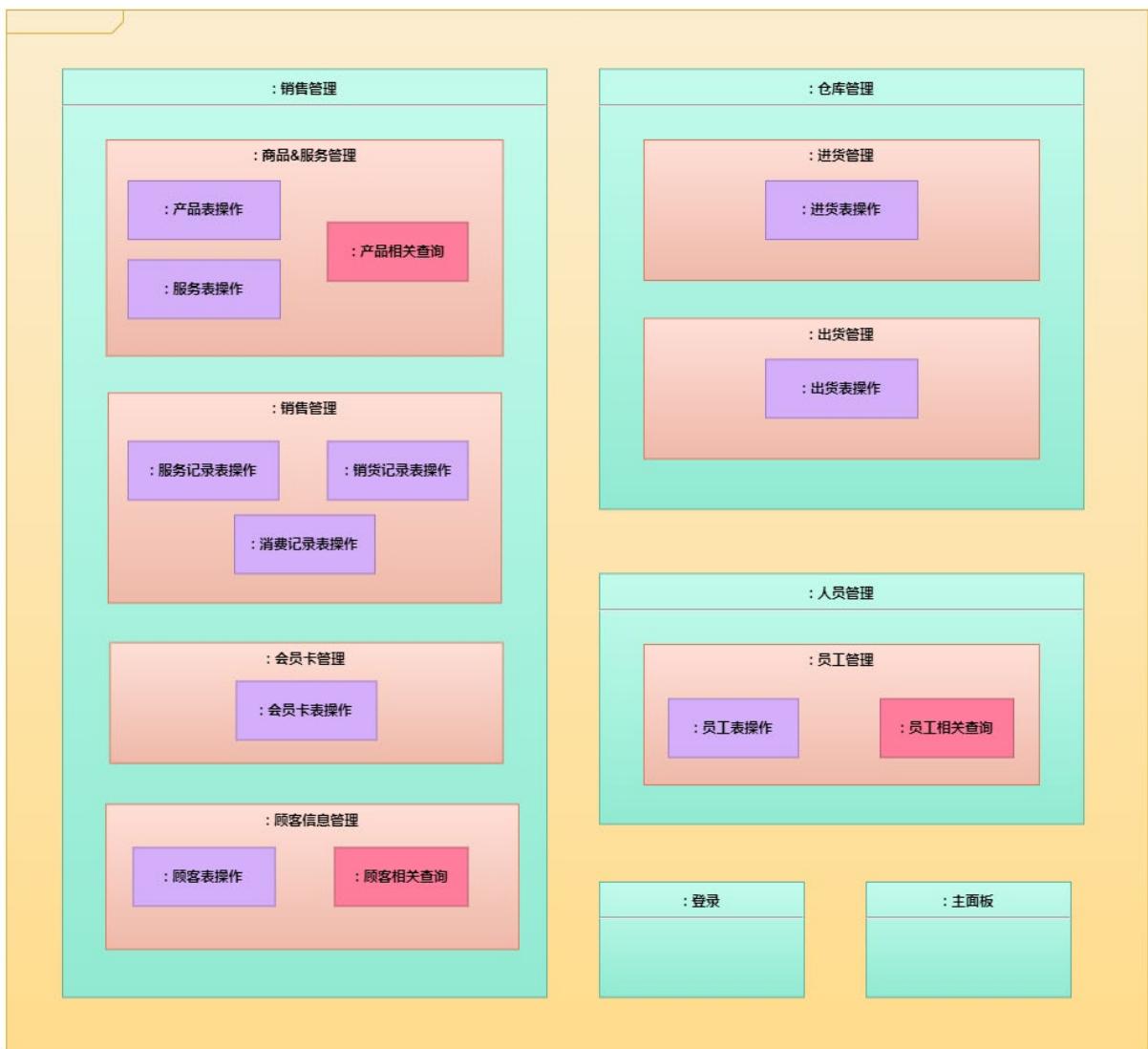
前后端之间使用 axios 进行通信

使用 IntelliJ IDEA 进行开发, Postman 进行串口调试

## 7.2 模块划分

### 7.2.1 整体模块划分

装  
订  
线



### 7.2.2 各模块功能

- 主面板：欢迎页，显示系统的基本信息和操作指引
- 登录：用户输入账号密码登录，以获得数据库查看操作权限。
- 销售管理

销售管理一级模块下又细分四个二级模块，分别是商品&服务管理、销售管理、会员卡管理、顾客管理。

✓ 商品&服务管理

➤ 产品表和服务表的增删改查

对产品表的添加，需要输入合法形式所有的非空属性（除了店内现存量、仓库剩余存货量），否则系统会弹出对应错误提醒。添加时店内现存量、仓库剩余存货量均为 0。对产品表的修改，编号、店内现存量、仓库剩余存货量不可更改。增删改成功/失败会有相应提示。对产品表单表的查找，对于编号、名称、备注是部分匹配，店内现存量、仓库剩余存货量、指导价是完全匹配。一页显示最多 10 条记录。

对服务表的添加，需要输入合法形式所有的非空属性，否则系统会弹出对应错误提醒。对服务表的修改，编号不可更改。增删改成功/失败会有相应提示。对服务表单表的查找，对于编号、名称、备注是部分匹配，指导价是完全匹配。一页显示最多 10 条记录。

➤ 产品相关进出货查询

根据产品名查找相关的进货与出货记录，该查询是针对产品名的部分匹配，其中进货记录包括产品名、批次、数量、进货日期、进价（单价）、备注，出货记录包括产品名、出批次、数量、出货日期、备注。进货记录与出货记录分别以一个表的形式展示，一张表一页显示最多 10 条记录。

✓ 销售管理

➤ 服务记录表、销货记录表、消费记录表增删改查

对服务记录表的添加，需要输入合法形式所有的非空属性，否则系统会弹出对应错误提醒。对服务记录表的修改，编号不可更改。增删改成功/失败会有相应提示。对服务记录表单表的查找，所有属性均是部分匹配。一页显示最多 10 条记录。

对销货记录表的添加，需要输入合法形式所有的非空属性，否则系统会弹出对应错误提醒。对销货记录表的修改，编号不可更改。增删改成功/失败会有相应提示。对销货记录表单表的查找，所有属性均是部分匹配。一页显示最多 10 条记录。

对消费记录表的添加，需要输入合法形式所有的非空属性，否则系统会弹出对应错误提醒。对消费记录表的修改，编号不可更改。增删改成功/失败会有相应提示。对消费记录表单表的查找，编号、操作员编号、购货记录编号、服务记录编号、会员卡号是部分匹配，结算方式、应付金额、实付金额是完全匹配。一页显示最多 10 条记录。

✓ 会员卡管理

➤ 会员卡表增删改查

对会员卡表的添加，需要输入合法形式所有的非空属性，否则系统会弹出对应错误提醒。对会员卡表的修改，编号不可更改。增删改成功/失败会有相应提示。对会员卡表单表的查找，卡号、持卡人 ID 是部分匹配，余额是完全匹配。一页显示最多 10 条记录。

✓ 顾客信息管理

➤ 顾客信息表增删改查

对顾客信息表的添加，需要输入合法形式所有的非空属性，否则系统会弹出对应错误提醒。对顾客信息的修改，编号不可更改。增删改成功/失败会有相应提示。对顾客信息单表的查找，编号、姓名、类型是部

分匹配，性别是完全匹配。一页显示最多 10 条记录。

➤ 顾客相关记录查找

根据顾客姓名查找相关的产品购买记录、服务记录、会员卡信息，该查询是针对顾客姓名的完全匹配，其中产品购买记录包括顾客姓名、产品编号、数量，服务记录包括顾客姓名、服务编号、数量，会员卡信息包括持卡人、卡号、余额。购买记录、服务记录与会员卡信息分别以一个表的形式展示，一张表一页显示最多 10 条记录。

• 仓库管理

✓ 进货管理

➤ 进货表增删改查

对进货表的添加，需要输入合法形式所有的非空属性，否则系统会弹出对应错误提醒。对进货表的修改，产品编号和批次不可更改。增删改成功/失败会有相应提示。对进货表单表的查找，产品编号、批次、负责员工编号、进货日期、备注是部分匹配，数量、进价（单价）是完全匹配。一页显示最多 10 条记录。

✓ 出货管理

➤ 出货表增删改查

对出货表的添加，需要输入合法形式所有的非空属性，否则系统会弹出对应错误提醒。对出货表的修改，产品编号和批次不可更改。增删改成功/失败会有相应提示。对出货表的查找，产品编号、批次、负责员工编号、出货日期、备注是部分匹配，数量是完全匹配。一页显示最多 10 条记录。

• 人员管理

✓ 员工管理

➤ 员工信息表增删改查

对员工信息表的添加，需要输入合法形式所有的非空属性，否则系统会弹出对应错误提醒。对员工信息的修改，编号不可更改。增删改成功/失败会有相应提示。对员工信息单表的查找，所有属性均是部分匹配。一页显示最多 10 条记录。

➤ 员工相关记录查找

根据员工姓名查找相关的员工进货/出货记录、负责的销货记录、服务记录，该查询是针对员工姓名的完全匹配，其中员工进货记录包括负责员工、产品编号、批次、数量、进货日期、进价（单价）、备注，员工出货记录包括负责员工、产品编号、批次、数量、出货日期、备注，员工销货记录包括销售员、产品编号、数量，服务记录包括服务员、员工服务编号、数量。进货/出货记录、负责的销货记录、服务记录分别以一个表的形式展示，一张表一页显示最多 10 条记录。

## 7.3 详细设计

### 7.3.1 类设计

后端参考数据库表设计进行实体类设计，由于进货表与出货表使用联合主键，故为其主键单独设定一个类，每个实体类对应一个 repository 接口和 Controller 类。Controller 另外编写相关 url 以供前端调用。

涉及到的实体类如下：

- Employee

```
@Entity  
@Data  
public class Employee {  
    @Id //是指定当前实体的主键是哪一个属性，因为实体和数据库表绑定到一起了，  
    所以说必须指定  
    private String id; /  
    private String name;  
    private String sex;  
    private String is_on_job;  
    private String tel;  
    private String type;  
    private Integer permission;  
    private String password;  
}
```

- Customer

```
@Entity  
@Data  
public class Customer {  
    @Id  
    private String id;  
    private String name;  
    private String sex;  
    private String type;  
}
```

- Card

```
@Entity  
@Data  
public class Card {  
    @Id  
    private String card_id;  
    private String customer_id;  
    private float balance;  
}
```

- Service

```
@Entity  
@Data
```

装  
订  
线

```
public class Service {  
    @Id  
    private String id;  
    private String name;  
    private float price;  
    private String remarks;  
}  
• Product  
@Entity  
@Data  
public class Product {  
    @Id  
    private String id;  
    private String name;  
    private int number_in_shop;  
    private int number_in_storehouse;  
    private float price;  
    private String remarks;  
}  
• PurchaseKey  
@Embeddable  
public class PurchaseKey implements Serializable {  
    private String product_no;  
    private String batch;  
}  
• Purchase  
@Entity  
@Data  
public class Purchase {  
    @EmbeddedId  
    private PurchaseKey purchaseKey;  
    private int all_number;  
    private String employee_no;  
    private Date purchase_date;  
    private float price;  
    private String remarks;  
}  
• ShipmentKey  
@Data  
@Embeddable  
public class ShipmentKey implements Serializable {  
    private String product_no;
```

```
    private String batch;
}

• Shipment
@Entity
@Data
public class Shipment {

    @EmbeddedId
    private ShipmentKey shipmentKey;
    private int all_number;
    private String employee_no;
    private Date shipment_date;
    private String remarks;
}

• Expense_record
@Entity
@Data
public class Expense_record {

    @Id
    private String id;
    private String employee_no;
    private String product_record_no;
    private String service_record_no;
    private float should_pay_amount;
    private float actual_pay_amount;
    private int pay_way;
    private String card_no;
}

• Service_record
@Entity
@Data
public class Service_record {

    @Id
    private String id;
    private String employee_no;
    private String service_no;
    private String customer_no;
    private int num;
}

• Product_record
@Entity
@Data
```

装  
订  
线

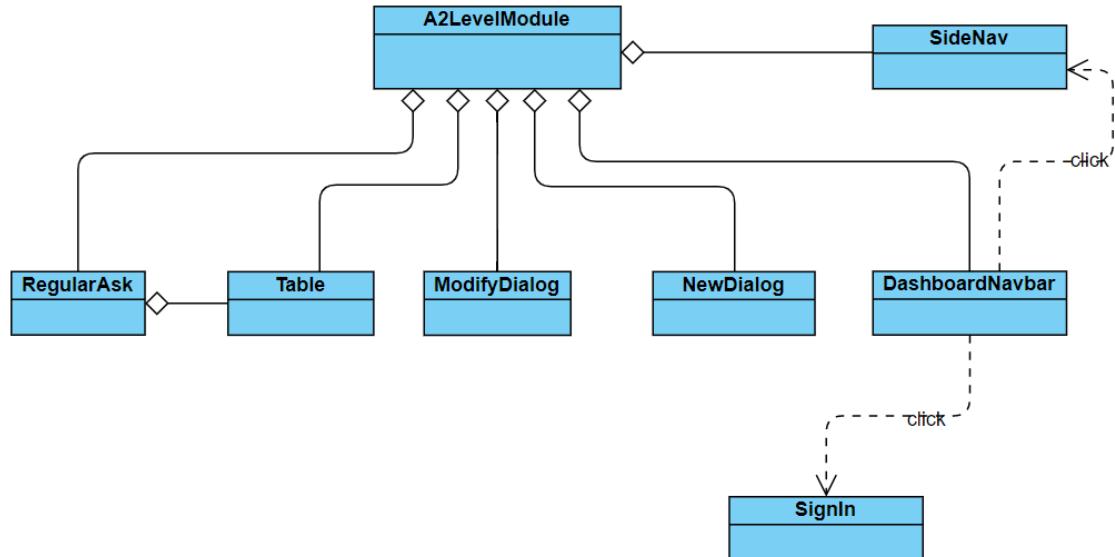
```

public class Product_record {
    @Id
    private String id;
    private String employee_no;
    private String product_no;
    private String customer_no;
    private int num;
}

```

### 7.3.2 组件设计

前端 React 框架以组件为单元进行操作。除主面板模块与登录模块外，其余一级模块下的二级模块的单表查询你都是由 Table(可以操作的动态数据表)、ModifyDialog(修改对话框)、NewDialog(新建对话框)、DashboardNavbar(顶部导航栏)、SideNav(侧边导航栏)、Search(单表搜索)几个组件组成的。某些模块还具有多表查询功能,使用 RegularAsk(常用查询)组件实现，RegularAsk 组件下包括 Table(可以操作的动态数据表)组件。一个二级模块下的组件封装在一个模块中，在模块内相应组件之间进行通信。二级组件之间不通信。



- **Table 动态表格**  
针对不同的表格数据，调用 Url 获取相应数据转换为合理的输出形式后展示，具有翻页功能
- **ModifyDialog 修改对话框**  
点击对应表项的修改按钮弹出该对话框，针对当前模块，使用 Url 获取原信息并填写在对应的输入表单中，用户可直接在原信息上进行修改，用户可选择提交或取消按钮，若提交，对不合法的输入弹出对应提示，若提交成功则直接关闭修改对话框并弹出成功提示。

- **NewDialog** 新建对话框  
点击表格底部的添加按钮弹出对话框，初始时信息表单项均为空，用户填写好后，可选择提交或取消按钮，若提交，对不合法的输入弹出对应提示，若提交成功则直接关闭添加对话框并弹出成功提示。
- **RegularAsk** 常用查询  
用户点击搜索按钮后，根据用户输入的查询表单和关键词进行搜索查询，用 Table 组件展示相应的查询结果。
- **DashboardNavbar** 顶部导航栏  
用户可点击最右侧按钮进行侧栏的显示隐藏的切换，点击次右侧按钮可进入登录界面，左侧字体展示当前模块名字，点击最左侧按钮可回到主面板
- **SideNav** 侧边栏  
根据用户权限的不同，显示不同的侧边信息，用户可点击跳转至对应模块
- **SignIn** 登录页面  
用户输入账号密码进行登录，登录成功后跳转至主页面，登录失败则弹出对应提示仍停留在该页面

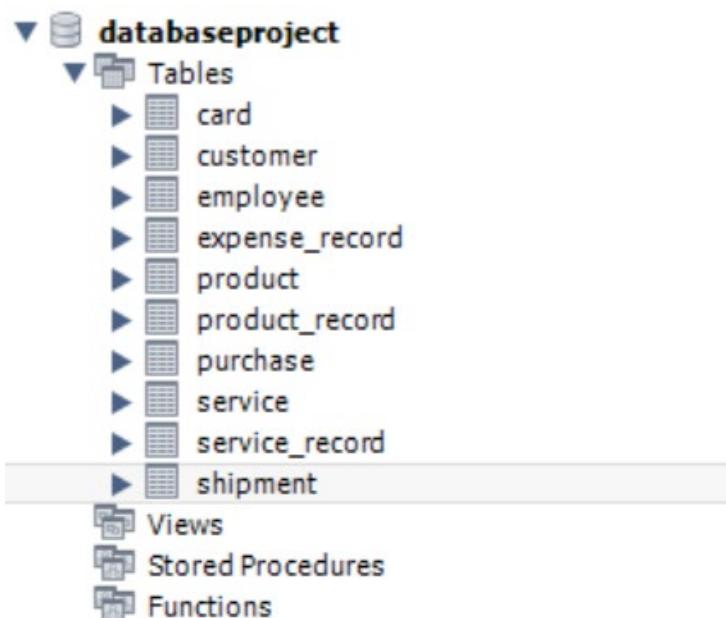
装

订  
线

## 8 实现过程

### 8.1 数据库建立

数据库总览



注意以下非先后顺序，存在交替操作。

- 建表  
使用 MySQL Workbench 进行直接操作  
以 shipment 建表过程为例

### Review the SQL Script to be Applied on the Database

Online DDL

Algorithm:	Default	Lock Type:	Default
------------	---------	------------	---------

```

1  CREATE TABLE `databaseproject`.`shipment` (
2      `product_no` VARCHAR(10) NOT NULL,
3      `batch` VARCHAR(5) NOT NULL,
4      `all_number` INT NOT NULL,
5      `employee_no` VARCHAR(5) NOT NULL,
6      `shipment_date` DATE NULL,
7      `remarks` VARCHAR(50) NULL,
8      PRIMARY KEY (`product_no`, `batch`),
9      INDEX `employee_no_idx` (`employee_no` ASC) VISIBLE,
10     CONSTRAINT `employee_no`
11         FOREIGN KEY (`employee_no`)
12             REFERENCES `databaseproject`.`employee` (`id`)
13             ON DELETE NO ACTION
14             ON UPDATE NO ACTION);
15

```

- 添加约束

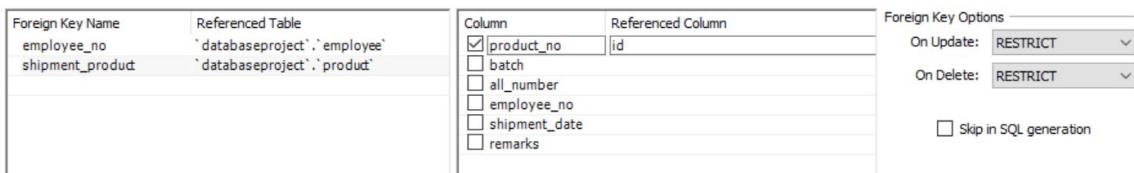
以 shipment 表为例

一般约束：

```
alter table shipment add constraint sp_an check (all_number>=0);
```

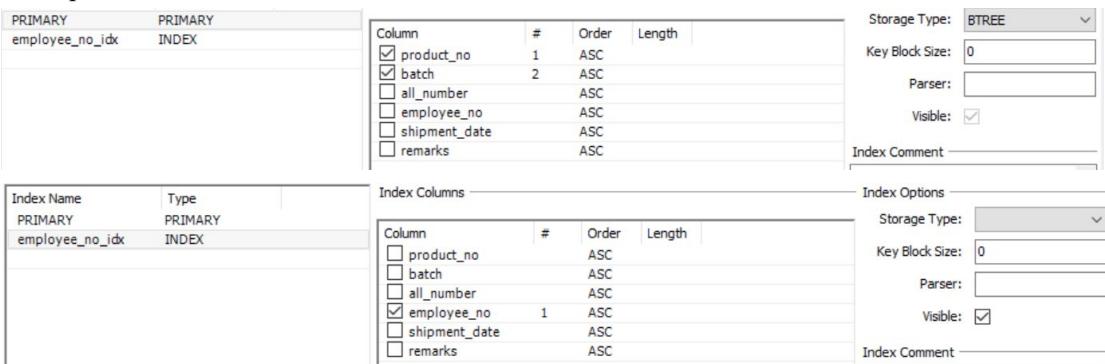
外码约束：

Foreign Key Name	Referenced Table	Column	Referenced Column	Foreign Key Options
employee_no	`databaseproject`.`employee`	<input checked="" type="checkbox"/> employee_no	<input type="checkbox"/> id	On Update: RESTRICT On Delete: RESTRICT
shipment_product	`databaseproject`.`product`	<input type="checkbox"/> product_no <input type="checkbox"/> batch <input type="checkbox"/> all_number <input type="checkbox"/> shipment_date <input type="checkbox"/> remarks		<input type="checkbox"/> Skip in SQL generation



- 添加索引

以 shipment 表为例（部分）



- 添加触发器

项目中一个重要的约束是 product 表中店内数量和仓库内数量的维护。当有进货时，仓库内数量会相应增加；当有出货时，仓库内数量对应减小，店内数量对应增加。当有销货记录产生时，对应的店内数量减小。

下展示与之相关的部分触发器

```

delimiter //
create trigger purchase_update_product
after update on purchase
for each row
begin
    update product set number_in_storehouse = number_in_storehouse - old.all_number + new.all_number
    where product.id = old.product_no;
end //

delimiter //
create trigger purchase_insert_product
after insert on purchase
for each row
begin
    update product set number_in_storehouse = number_in_storehouse + new.all_number
    where product.id = new.product_no;
end //

drop trigger if exists purchase_delete_product;

delimiter //
create trigger purchase_delete_product
after delete on purchase
for each row
begin
    update product set number_in_storehouse = number_in_storehouse - old.all_number
    where product.id = old.product_no;
end //

```

```
delimiter //
create trigger pr_update_product
after update on product_record
for each row
begin
update product set number_in_shop = number_in_shop - new.num + old.num
where product.id = new.product_no;
end //

delimiter //
create trigger pr_insert_product
after insert on product_record
for each row
begin
update product set number_in_shop = number_in_shop - new.num
where product.id = new.product_no;
end //

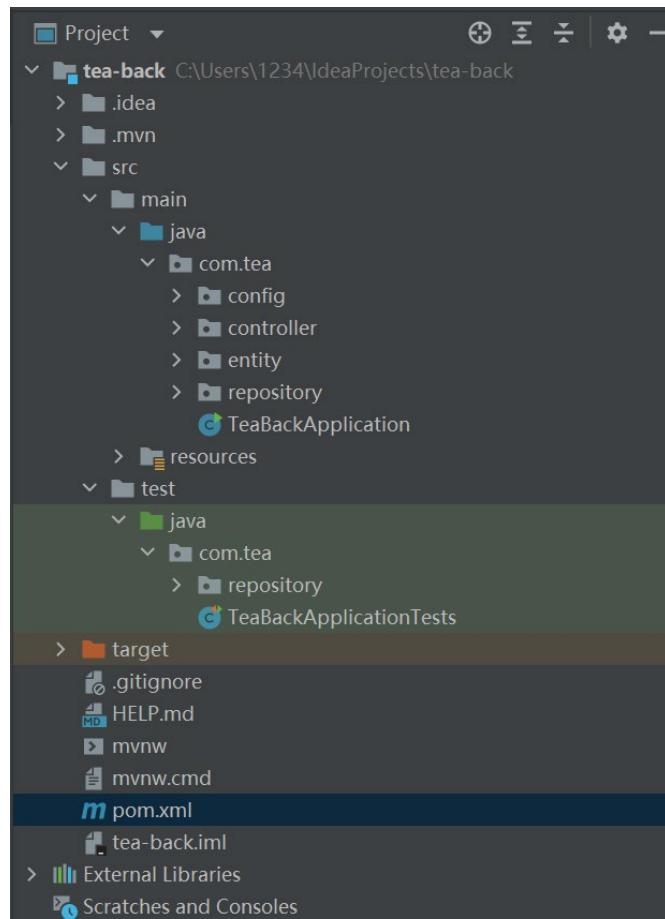
drop trigger if exists pr_delete_product;

delimiter //
create trigger pr_delete_product
after delete on product_record
for each row
begin
update product set number_in_shop = number_in_shop + old.num
where product.id = old.product_no;
end //
```

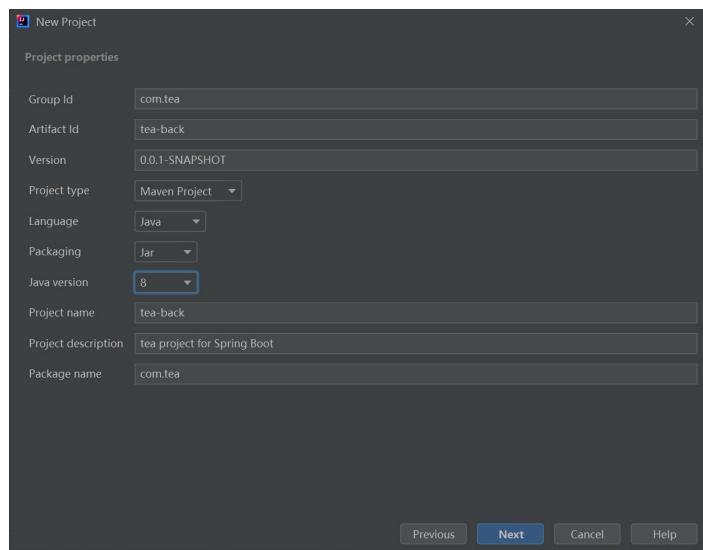
装  
订  
线

## 8.2 后端搭建

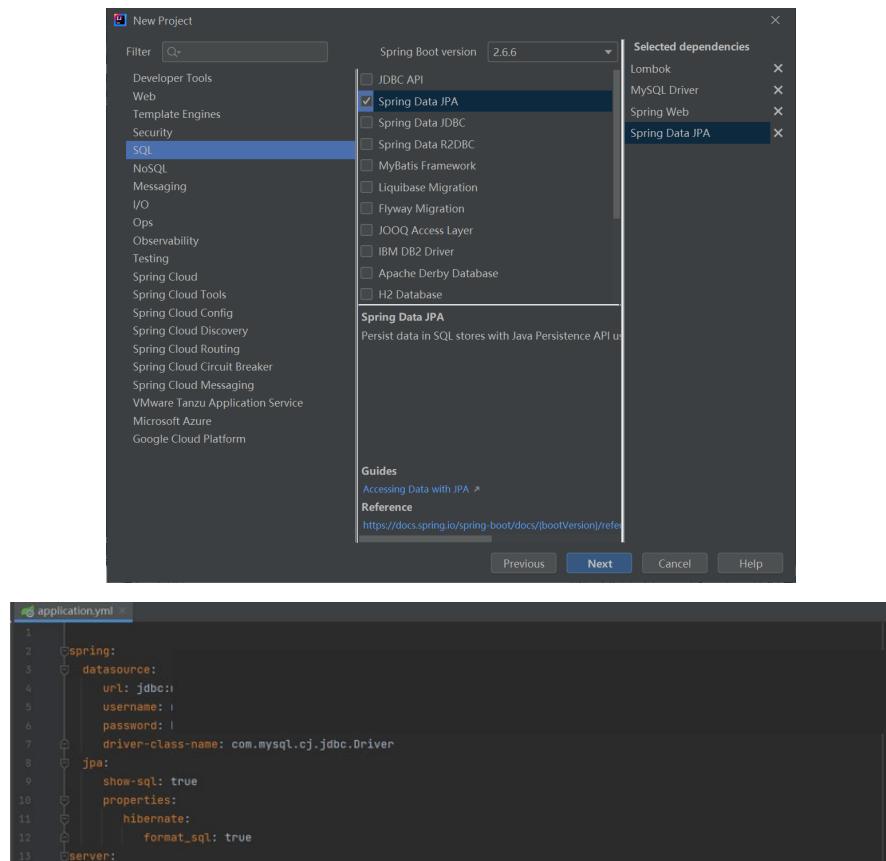
后端整体文件结构图

装  
订  
线

- 环境配置



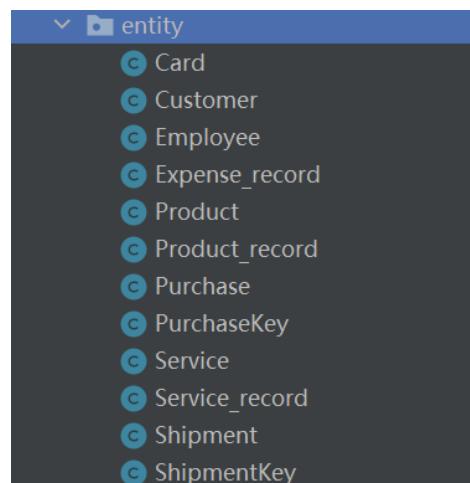
装  
订  
线



- 实体类实现

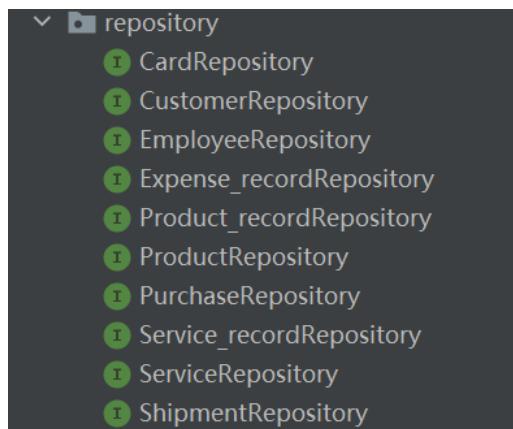
实体类实现代码见 7.3.1 类设计

实体类概览，对应 10 张表有 10 个实体类，另有两个类分别是 purchase 和 shipment 的主键



- 接口实现

接口概览，对应 10 张表有 10 个接口



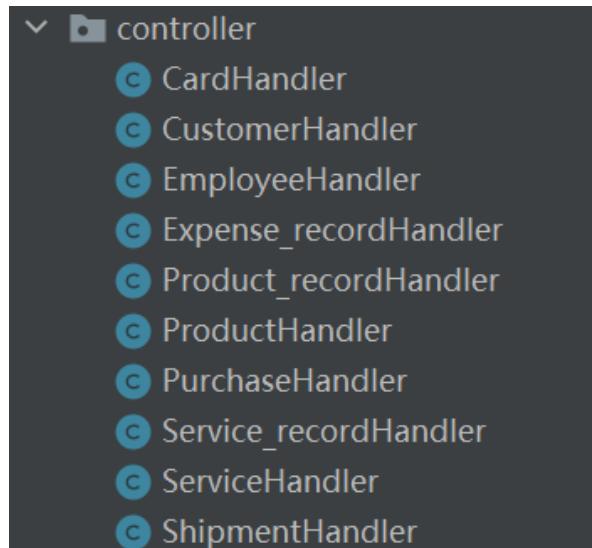
接口实现，以 Product\_recordRepository 为例

```
public interface Product_recordRepository extends
JpaRepository<Product_record, String> {
    @Query("select product_record from Product_record
product_record where product_record.employee_no=?1")
    List<Product_record> findByEmployeeNo(String employee_no);

    @Query("select product_record from Product_record
product_record where product_record.customer_no=?1")
    List<Product_record> findByCustomerNo(String employee_no);
}
```

- Controller 类实现

Controller 类概览，对应 10 张表有 10 个 Controller 类



Controller 类实现，以 Product\_recordHandler 为例

```
@RestController
@RequestMapping("/product_record")
public class Product_recordHandler {
```

装  
订  
线

```
@Autowired
private Product_recordRepository product_recordRespository;

@GetMapping("/findAll")
public List<Product_record> findAll(){return
product_recordRespository.findAll();}

@PostMapping("/save")
public String save(@RequestBody Product_record
product_record){
    Product_record result =
product_recordRespository.save(product_record);
    if(result!=null){
        return "success";
    }else{
        return "error";
    }
}

@GetMapping("/findById/{id}")
public Product_record findById(@PathVariable("id") String id){
    return product_recordRespository.findById(id).get();
}

@GetMapping("/findByEmployeeNo/{employee_no}")
public List<Product_record>
findByEmployeeNo(@PathVariable("employee_no") String
employee_no){
    return
product_recordRespository.findByEmployeeNo(employee_no);
}

@GetMapping("/findByCustomerNo/{customer_no}")
public List<Product_record>
findByCustomerNo(@PathVariable("customer_no") String
customer_no){
    return
product_recordRespository.findByCustomerNo(customer_no);
}

@PutMapping("/update")
```

```
public String update(@RequestBody Product_record product_record) {
    Product_record result =
product_recordRespository.save(product_record);
    if(result!=null){
        return "success";
    }else{
        return "error";
    }
}

@DeleteMapping("/deleteById/{id}")
public void deleteById(@PathVariable("id") String id) {
    product_recordRespository.deleteById(id);
}
```

装

订  
线

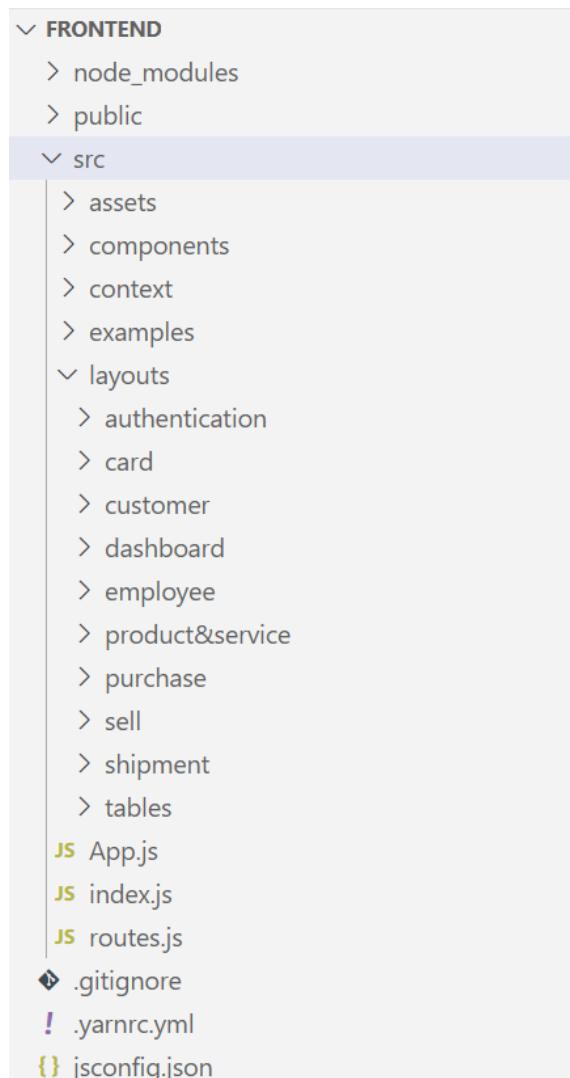
- 跨域问题解决

在 CrosConfig 文件中解决

```
@Configuration
public class CrosConfig implements WebMvcConfigurer {
    @Override
    public void addCorsMappings(CorsRegistry registry) {
        registry.addMapping("/**")
            .allowedOriginPatterns(" * ")
            .allowedMethods("GET", "HEAD", "POST", "PUT", "DELETE", "OPTIONS")
            .allowCredentials(true)
            .allowedHeaders(" * ")
            .maxAge(3600);
    }
}
```

## 8.2 前端搭建

前端整体文件结构图

装  
订  
线

- 环境搭建  
使用模板，`npm install` 指令进行安装。
- 组件编写  
在模板基础上加以改造，编写好各个组件，尽量保持其通用性
- 模块编写  
使用编写好的组件，组装各个模块，对组件进行适当调整，以适应各模块之间差异
- 路由添加  
将模块与路由绑定，实现页面的正常跳转

#### 8.4 功能调试

##### 8.4.1 数据库约束调试

针对 `product` 中店内数量和仓库内数量的触发约束进行调试  
初始 `product` 表

	id	name	number_in_shop	number_in_storehouse	price	remarks
▶	0000000000	力开天地	25	10	1000	可随赠小礼品
	0000000001	护犊情深	0	15	2300	
	0000000002	08年7542	0	3	230	
	0000000003	09年7572	0	0	10000	NULL

进货 5 件 09 年 7572

```
INSERT INTO `databaseproject`.`purchase` (`product_no`, `batch`, `all_number`, `employee_no`, `price`) VALUES ('0000000003', '00001', '5', '10004', '4000');
```

product 表变为

	id	name	number_in_shop	number_in_storehouse	price	remarks
▶	0000000000	力开天地	25	10	1000	可随赠小礼品
	0000000001	护犊情深	0	15	2300	
	0000000002	08年7542	0	3	230	
	0000000003	09年7572	0	5	10000	NULL

出货 2 件 09 年 7572

```
INSERT INTO `databaseproject`.`shipment` (`product_no`, `batch`, `all_number`, `employee_no`) VALUES ('0000000003', '00001', '2', '10004');
```

product 表变为

	id	name	number_in_shop	number_in_storehouse	price	remarks
▶	0000000000	力开天地	25	10	1000	可随赠小礼品
	0000000001	护犊情深	0	15	2300	
	0000000002	08年7542	0	3	230	
	0000000003	09年7572	2	3	10000	NULL

销售 1 件 09 年 7572

```
INSERT INTO `databaseproject`.`product_record` (`id`, `employee_no`, `product_no`, `num`) VALUES ('00000000000000000000000000000002', '10002', '000000000003', '1');
```

product 表变为

	id	name	number_in_shop	number_in_storehouse	price	remarks
▶	0000000000	力开天地	25	10	1000	可随赠小礼品
	0000000001	护犊情深	0	15	2300	
	0000000002	08年7542	0	3	230	
	0000000003	09年7572	1	3	10000	NULL

将销货记录修改为 2 件 09 年 7572

```
UPDATE `databaseproject`.`product_record` SET `num` = '2' WHERE (`id` = '00000000000000000000000000000002');
```

product 表变为

	id	name	number_in_shop	number_in_storehouse	price	remarks
▶	0000000000	力开天地	25	10	1000	可随赠小礼品
	0000000001	护犊情深	0	15	2300	
	0000000002	08年7542	0	3	230	
	0000000003	09年7572	0	3	10000	NULL

#### 8.4.2 后端调试

编写 test 代码进行测试

```

7  @SpringBootTest
8  class ShipmentRepositoryTest {
9      @Autowired
10     private ShipmentRepository shipmentRepository;
11
12     @Test
13     void findAll() { System.out.println(shipmentRepository.findAll()); }
14
15     @Test
16     void findByProductNo() { System.out.println(shipmentRepository.findByProductNo("0000000000")); }
17 }
18
19
20
21

```

Run: ShipmentRepositoryTest.findByProductNo

Test Results: 172 ms

- ShipmentRepositoryTest: 172 ms
  - findByProductNo: 172 ms

```

shipment0_.shipment_date as shipment6_9_
from
shipment shipment0_
where
shipment0_.product_no=?
[Shipment(shipmentKey=ShipmentKey(product_no=0000000000, batch=00001), all_number=2, employee
2022-05-05 12:58:23.287 INFO 24044 --- [ionShutdownHook] j.LocalContainerEntityManagerFacto
2022-05-05 12:58:23.289 INFO 24044 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource
2022-05-05 12:58:23.291 INFO 24044 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource

Process finished with exit code 0

```

test示例

### 8.4.3 串口调试

使用 postman 对后端编写好的串口进行调试

GET http://localhost:8082/card/findByCustomerId/10023

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Query Params

	KEY	VALUE	DESCRIPTION
	Key	Value	Description

Body Cookies Headers (8) Test Results

Status: 200 OK Time: 295 ms

Pretty Raw Preview Visualize JSON

```

1 [
2   {
3     "card_id": "0000000002",
4     "customer_id": "10023",
5     "balance": 100.0
6   },
7   {
8     "card_id": "1000010002",
9     "customer_id": "10023",
10    "balance": 13.0
11  }
12 ]

```

串口调试示例

### 8.4.3 前端调试

在开发者工具中，使用 console.log 进行输出，结合断点进行调试

装

订  
线

```

▶ :≡ 7 messages
▶ ⏪ 6 user me...
▶ ✎ 2 errors
⚠ No warn...
▶ 🌐 4 info
▶ 🌈 1 verbose

js/bundle.js:61377:70)
  at ThemeProvider (http://localhost:3000/static/js/bundle.js:61236:5)
    at ThemeProvider (http://localhost:3000/static/js/bundle.js:61397:5)
      at App (http://localhost:3000/static/js/bundle.js:309:94)
        at
SoftUIControllerProvider (http://localhost:3000/static/js/bundle.js:9494:5)
          at Router (http://localhost:3000/static/js/bundle.js:128670:15)
            at BrowserRouter (http://localhost:3000/static/js/bundle.js:127480:5)

index.js:37
▼ (3) [{...}, {...}, {...}] ⓘ
▶ 0: {card_id: '0000000002'}
▶ 1: {card_id: '1000000000'}
▶ 2: {card_id: '1000010002'}
  length: 3
  [[Prototype]]: Array(0)

index.js:39
▶ {columns: Array(4), rows: Array(3)}

```

前端调试截图

## 9 应用系统展示

进入系统，自动转至主面板，此时是未登录状态

侧边栏按钮

登录按钮

Dashboard

适应销售仓储管理需求  
茶馆仓储管理系统  
权责分明，简单易用

登录使用 →

销售记录  
(+3% 与去年同期相比)

仓库进出货量总览  
↑ 4% more in 2021

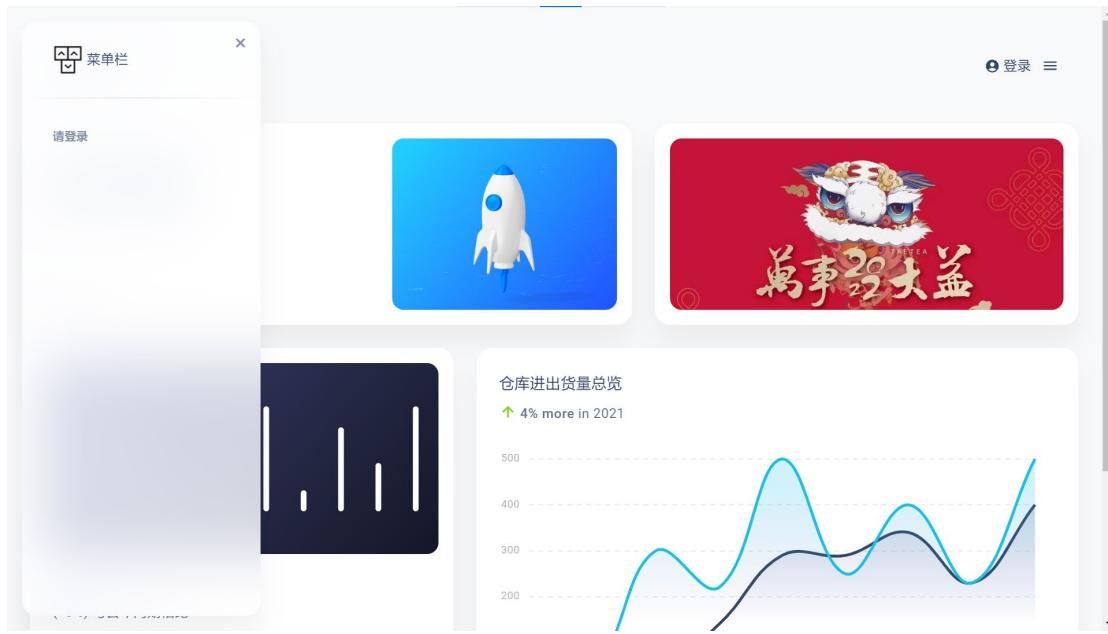
主面板

点击侧边栏按钮，此时侧栏（菜单栏）无内容展示

装

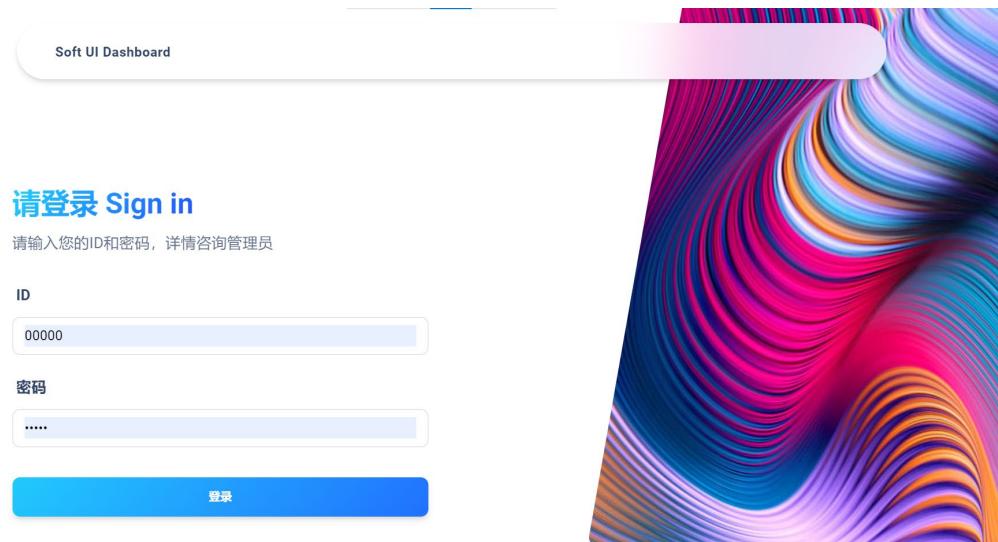
订

线



未登录状态下打开侧边状态的主面板

点击登录按钮，跳转至登录页面



登录页面

输入账号密码后点击登录，若输入正确弹出相应提示，点击确定后，跳转至登录成功后的页面

Soft UI Dashboard

localhost:3000 显示  
登录成功，准备跳转

确定

### 请登录 Sign in

请输入您的ID和密码，详情咨询管理员

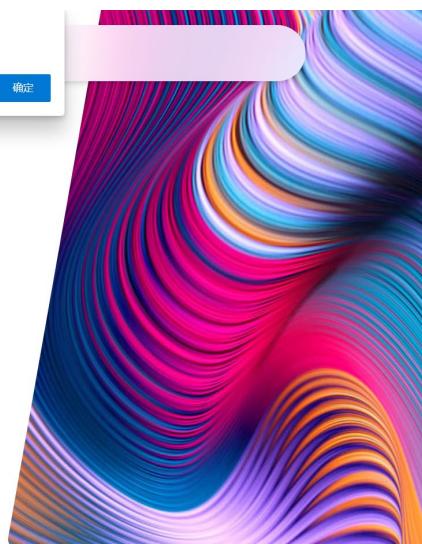
ID

00000

密码

.....

登录



登录成功后提示

装  
订  
线

用户名 → 管理员

登录成功后的主面板，以管理员为例

进入二级模块商品&服务管理，滑动滚动条可以看到如下界面，包括产品信息表、服务信息表、和产品进出货查找

装  
订  
线

**Product & Service**

编号	名称	店内现存量	仓库剩余存货量	指导价	备注	操作
0000000000	力开天地	25	10	1000	可随赠小礼品	<button>修改</button> <button>删除</button>
0000000001	护犊情深	0	15	2300		<button>修改</button> <button>删除</button>
0000000002	08年7542	0	3	230		<button>修改</button> <button>删除</button>
0000000003	09年7572	0	3	10000		<button>修改</button> <button>删除</button>
0000000004	10年8582	0	0	23000		<button>修改</button> <button>删除</button>

**Service Information Table**

编号	名称	指导价	备注	操作	
0000000008	玫瑰袋泡	0	0	49 盒	<button>修改</button> <button>删除</button>
0000000009	荷叶袋泡	0	0	49	<button>修改</button> <button>删除</button>

1-10 of 12 |< < > >|

**添加新产品**

**Product Outbound Record Search**

进货记录  出货记录

产品名

搜索

### 二级模块商品&服务管理

管理员进行产品信息表的查看，表单每页展示 10 项，可上下翻页查看

产品信息表

编号	名称	店内现存量	仓库剩余存货量	指导价	备注	操作
0000000010	11年陈皮	0	0	1000	每斤	<button>修改</button> <button>删除</button>
0000000011	15年陈皮	0	0	1000	斤	<button>修改</button> <button>删除</button>

11-12 of 12 |< < > >|

### 产品信息表翻页效果

管理员对产品进行添加，单击下方的添加新产品按钮，弹出如下新建对话框

添加新产品

编号  
如: 1234567890

名称  
名称

指导价  
价格 (元)

备注  
备注

取消 提交

对对话框进行填写，比如有必填项忘记填写，会弹出对应报错提示

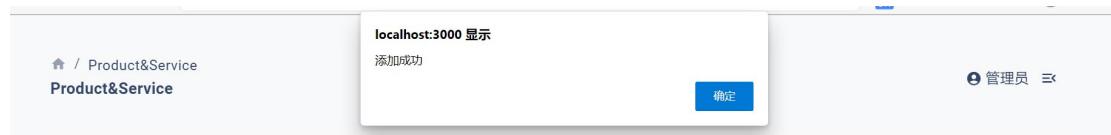
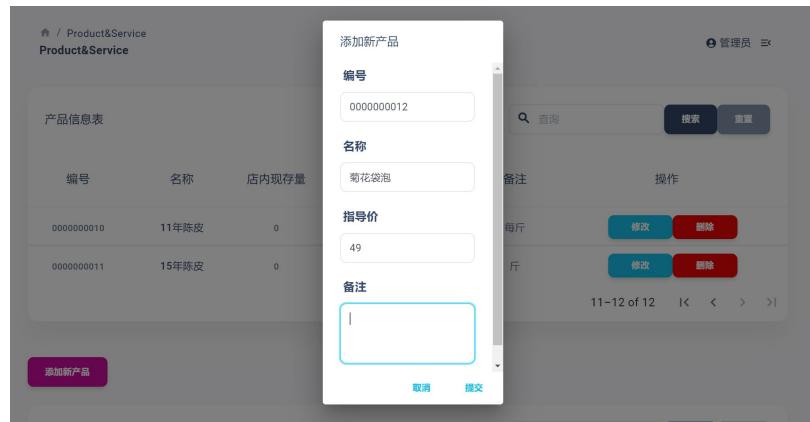
localhost:3000 显示  
编号不能为空

确定

添加新产品

添加编号后，再次提交，添加成功

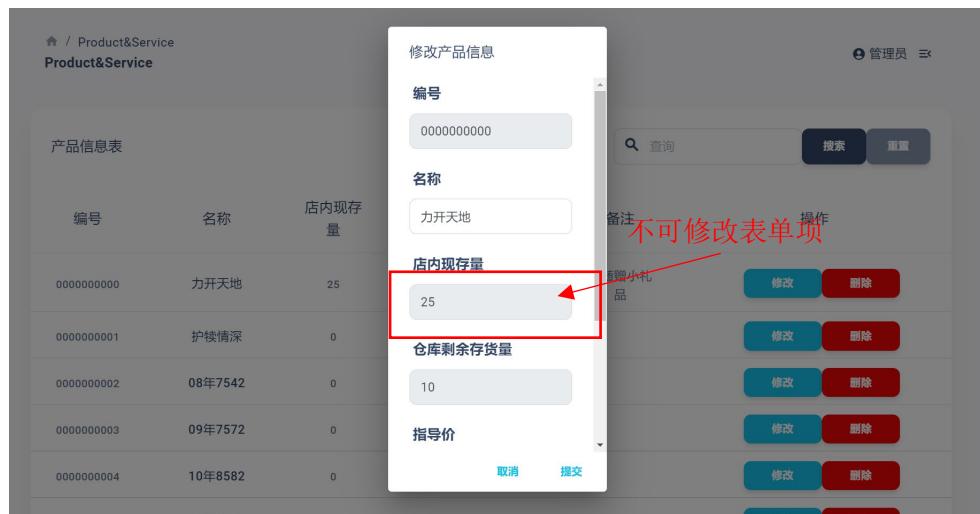
装  
订  
线



编号	名称	店内现存量	仓库剩余存货量	指导价	备注	操作
0000000010	11年陈皮	0	0	1000	每斤	<button>修改</button> <button>删除</button>
0000000011	15年陈皮	0	0	1000	斤	<button>修改</button> <button>删除</button>
0000000012	菊花袋泡	0	0	49		<button>修改</button> <button>删除</button>

### 添加后的的产品信息表

管理员对产品进行修改，单击要从修改的表单项操作栏中的修改按钮，弹出修改对话框，可以看到部分选项是不可修改的



修改后提交，可发现修改前后指导价改变

编号	名称	店内现存量	仓库剩余存货量	指导价	备注	操作
0000000000	力开天地	25	10	1000	可随赠小礼品	<button>修改</button> <button>删除</button>
0000000000	力开天地	25	10	1020	可随赠小礼品	<button>修改</button> <button>删除</button>

修改前（上） 修改后（下）

管理员对 11 年陈皮进行删除，点击对应表单项的删除按钮，删除成功则弹出对应提示

编号	名称	店内现存量	仓库剩余存货量	指导价	备注	操作
0000000010	11年陈皮	0	0	1000	每斤	<button>修改</button> <button>删除</button>
0000000011	15年陈皮	0	0	1000	斤	<button>修改</button> <button>删除</button>
0000000012	菊花袋泡	0	0	49		<button>修改</button> <button>删除</button>

编号	名称	店内现存量	仓库剩余存货量	指导价	备注	操作
0000000011	15年陈皮	0	0	1000	斤	<button>修改</button> <button>删除</button>
0000000012	菊花袋泡	0	0	49		<button>修改</button> <button>删除</button>

删除后结果

管理员对“袋泡”进行查找，在搜索框中输入袋泡，点击搜索按钮进行查找

装  
订  
线

编号	名称	店内现存量	仓库剩余存货量	指导价	备注	操作
0000000000	力开天地	25	10	1020	可随赠小礼品	<button>修改</button> <button>删除</button>
0000000001	护犊情深	0	15	2300		<button>修改</button> <button>删除</button>
0000000002	08年7542	0	3	230		<button>修改</button> <button>删除</button>
0000000003	09年7572	0	3	10000		<button>修改</button> <button>删除</button>
0000000004	10年8582	0	0	23000		<button>修改</button> <button>删除</button>

编号	名称	店内现存量	仓库剩余存货量	指导价	备注	操作
0000000008	玫瑰袋泡	0	0	49	盒	<button>修改</button> <button>删除</button>
0000000009	荷叶袋泡	0	0	49		<button>修改</button> <button>删除</button>
0000000012	菊花袋泡	0	0	49		<button>修改</button> <button>删除</button>

### 搜索结果

管理员想查找与“力开天地”相关的所有进货出货记录，在产品进货出货信息查找中勾选进货记录、出货记录，输入产品名“力开天地”，点击搜索按钮进行搜索。

得到产品进货信息表和出货信息表与之有关的记录

### 产品进出货信息查找

进货记录  出货记录

#### 产品名

力开天地

**搜索**

产品进货表

产品名	批次	数量	进货日期	进价 (单价)	备注
力开天地	00001	23	2022-05-06	1200	已验货
力开天地	00002	10	2022-05-26	1200	
力开天地	00003	2	2022-05-04	2300	

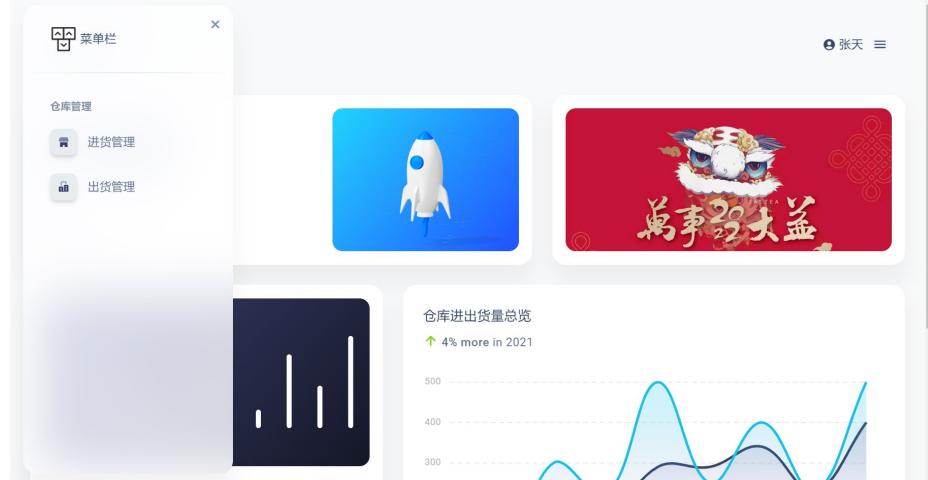
1-3 of 3 |&lt; &lt; &gt; &gt;|

产品出货表

产品名	批次	数量	出货日期	备注
力开天地	00001	2	2022-04-12	
力开天地	00002	23	2022-04-13	

1-2 of 2 |&lt; &lt; &gt; &gt;|

现在切换用户为张天，该用户只具有仓库管理权限，可以看到，其只能进货出货管理



## 10 性能测试与改进

### 10.1 性能测试

#### 10.1.1 QPS(每秒钟查询数量)

```
1 show global status like 'Question%'
```

Variable_name	Value
Questions	47537

## 10.1.2 TPS(每秒钟事务数量)

```
1 show global status like 'Com_commit'
```

	Variable_name	Value
▶	Com_commit	605

```
1 show global status like 'Com_rollback'
```

	Variable_name	Value
▶	Com_rollback	14

$$\text{TPS} = (\text{Com_commit} + \text{Com_rollback}) / \text{seconds} = 619$$

## 10.1.3 线程连接数

```
show global status like 'Threads%'
```

	Variable_name	Value
▶	Threads_cached	1
	Threads_connected	12
	Threads_created	22
	Threads_running	2

## 10.1.3 锁定状态

```
1 show global status like '%lock%'
```

	Variable_name	Value
▶	Com_lock_instance	0
	Com_lock_tables	0
	Com_unlock_instance	0
	Com_unlock_tables	0
	Handler_external_lock	100639
	Innodb_row_lock_current_waits	0
	Innodb_row_lock_time	0
	Innodb_row_lock_time_avg	0
	Innodb_row_lock_time_max	0
	Innodb_row_lock_waits	0
	Key_blocks_not_flushed	0
	Key_blocks_unused	6698
	Key_blocks_used	0
	Locked_connects	0
	Performance_schema_locker_lost	0
	Performance_schema_metadata_lock_time	0
	Performance_schema_rwlock_contention	0
	Performance_schema_rwlock_iowait_time	0
	Performance_schema_table_handles_scanned	0

## 10.2 性能改进

- 使用 InnoDB 引擎。因为其支持事务，支持外键，支持行级锁。
- 创建 B+树索引
- 修改查询语句，如查找与名字为张天的员工相关的销货记录，使用

```
select * from product_record where product_record.no in  
    select id from employee where employee.name = '张天'
```

而不是

```
select * from product_record join employee  
    where employee.name = '张天'
```

## 11 系统运行维护

### 11.1 运行部署

安装 MySQL，导入数据库.sql 文件，见附件

定位至前端文件夹 frontend，在控制台中输入 npm install 指令安装前端，

装

订

线

连接 MySQL 数据库，定位至前端文件夹，在控制台中输入 npm start 运行前端，定位至后端文件夹在控制台中输入指令 java -jar tea-back.jar 运行后端代码 打包好的 jar 包见 backend 文件夹

```
C:\Users\1234>cd C:\Users\1234\IdeaProjects\tea-back\out\artifacts\tea_back_jar  
C:\Users\1234\IdeaProjects\tea-back\out\artifacts\tea_back_jar>java -jar tea-back.jar
```

### 11.2 备份

使用如下 sql 语句进行备份

```
mysqldump -u username -P --all-databases>filename.sql
```

### 11.2 故障处理

暂时人工进行故障处理