

《软件开发综合实践》实验报告

信息学院 学院 计算机科学与技术 专业 2021
级

实验时间 2023 年 8 月 15 日

姓名 李东骏 学号 20211120141

实验名称 购物管理系统

实验成绩

一、实验目的

培养软件项目管理的意识，使学生掌握规范的软件开发和维护的方法。

二、实验软件

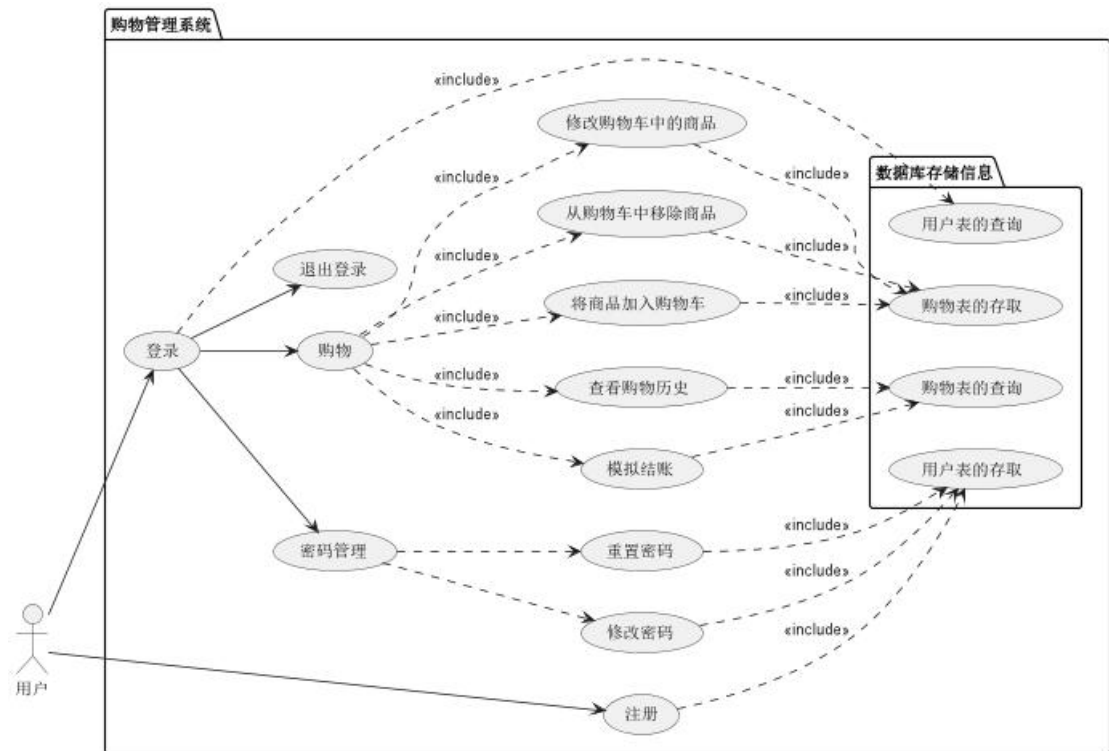
Idea, Springboot, Mybatis, Maven。

三、实验方案

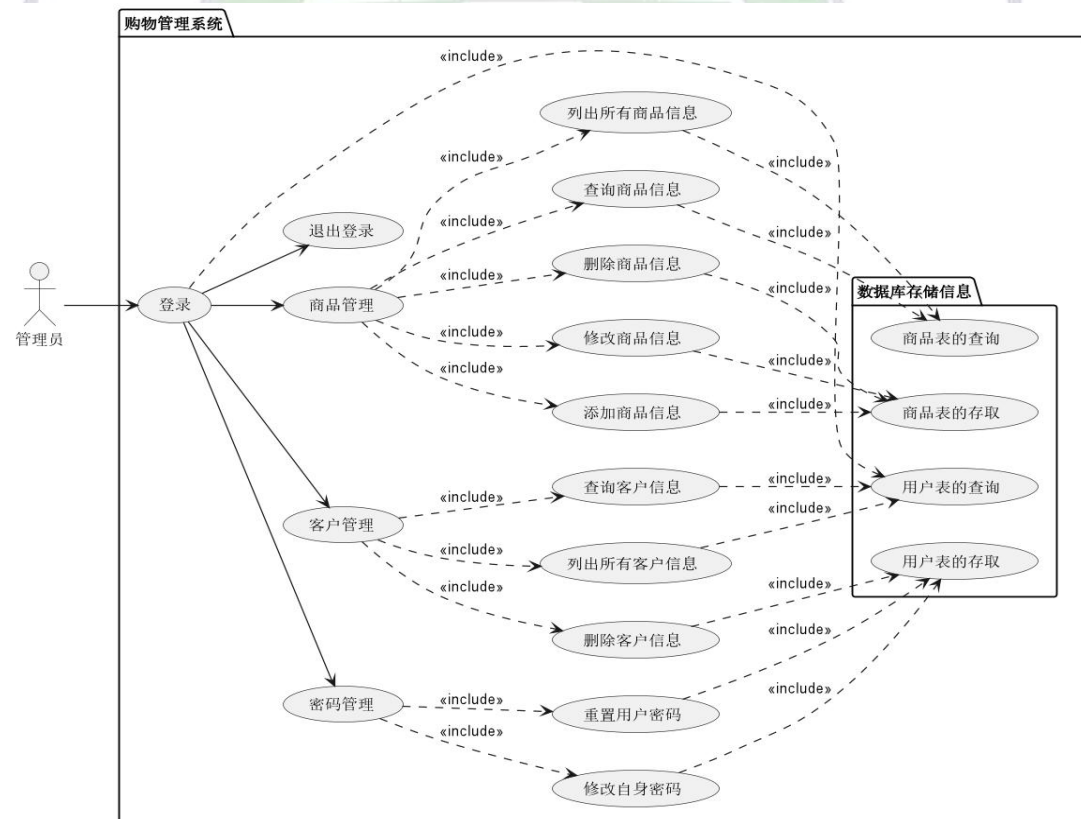
因为本次实验要求的是写一个购物管理系统，分别有管理员和客户两种角色，并且商品以及个人信息需要进行存储以及一系列操作，所以应用到了 mysql 数据库来进行数据的存储，其次就是对数据进行操作，我学习了 mybatis，并且使用 mybatis 来进行对数据库的操作，我使用了 maven 来管理依赖项和构建过程，并且使用了 springboot 来简化程序的开发。

在进行程序的开发之前，我首先根据系统的要求画出了 UML 用例图和顺序图：

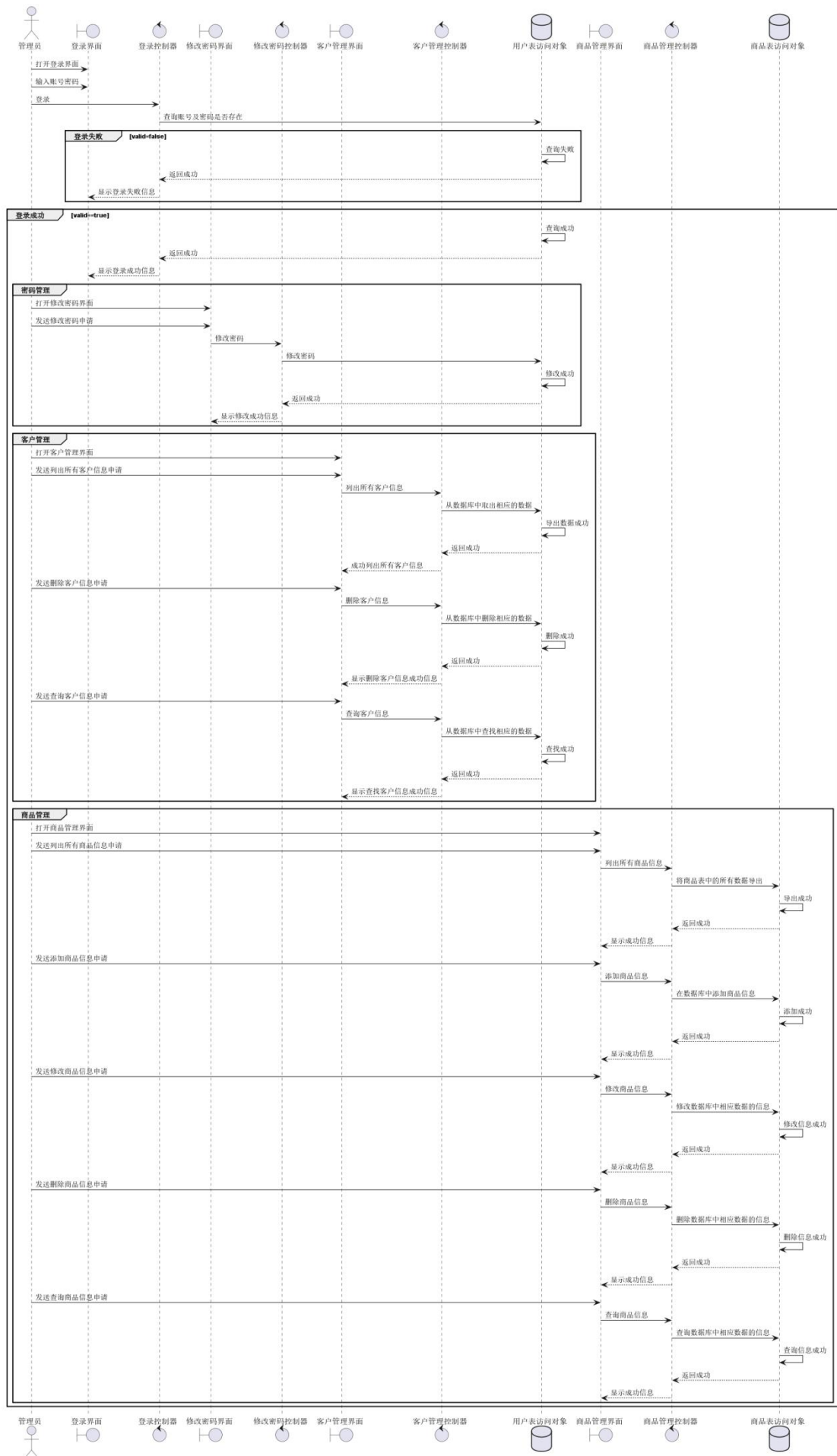
客户的用例图：



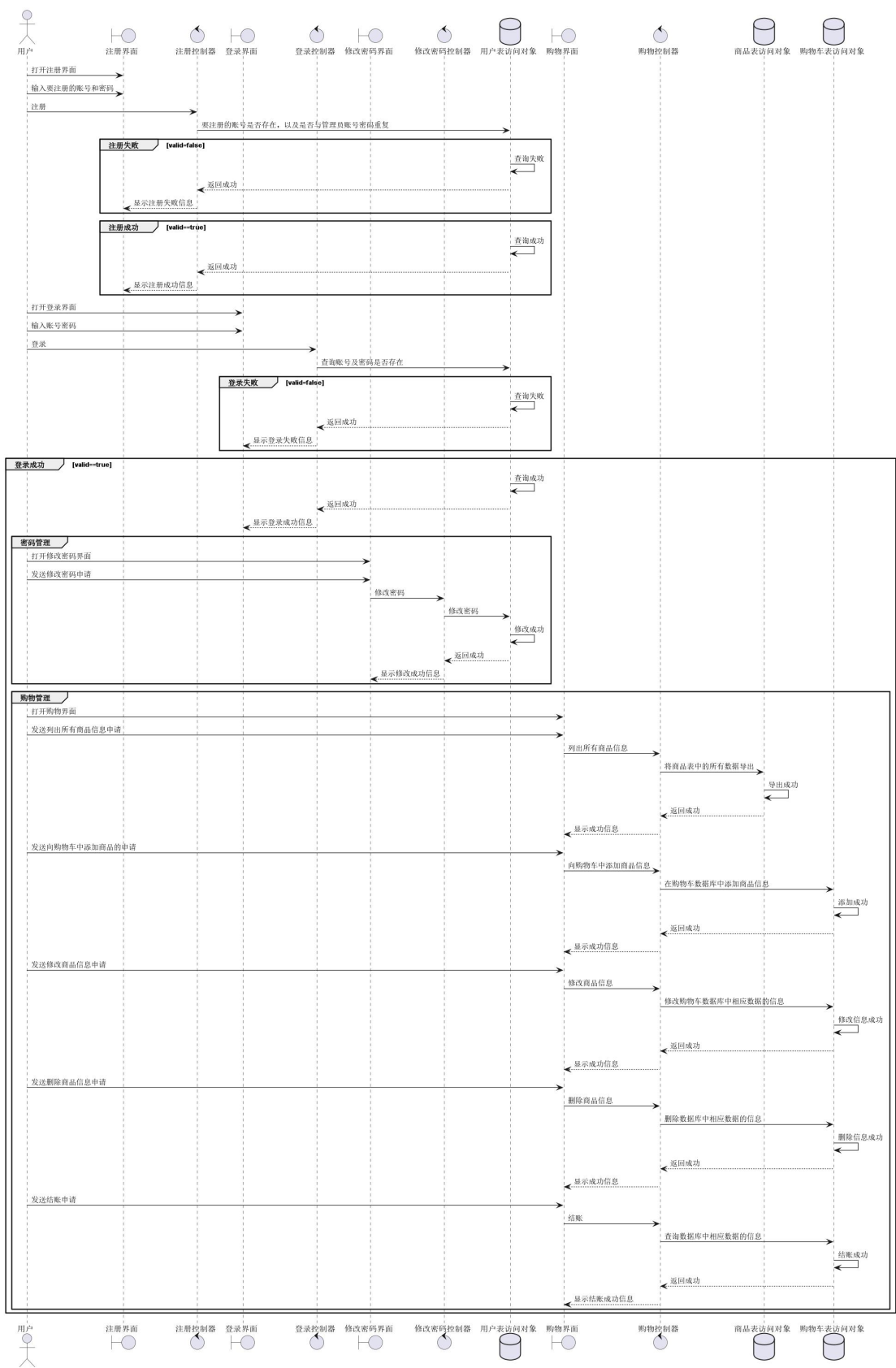
管理员的用例图：



管理员的顺序图：



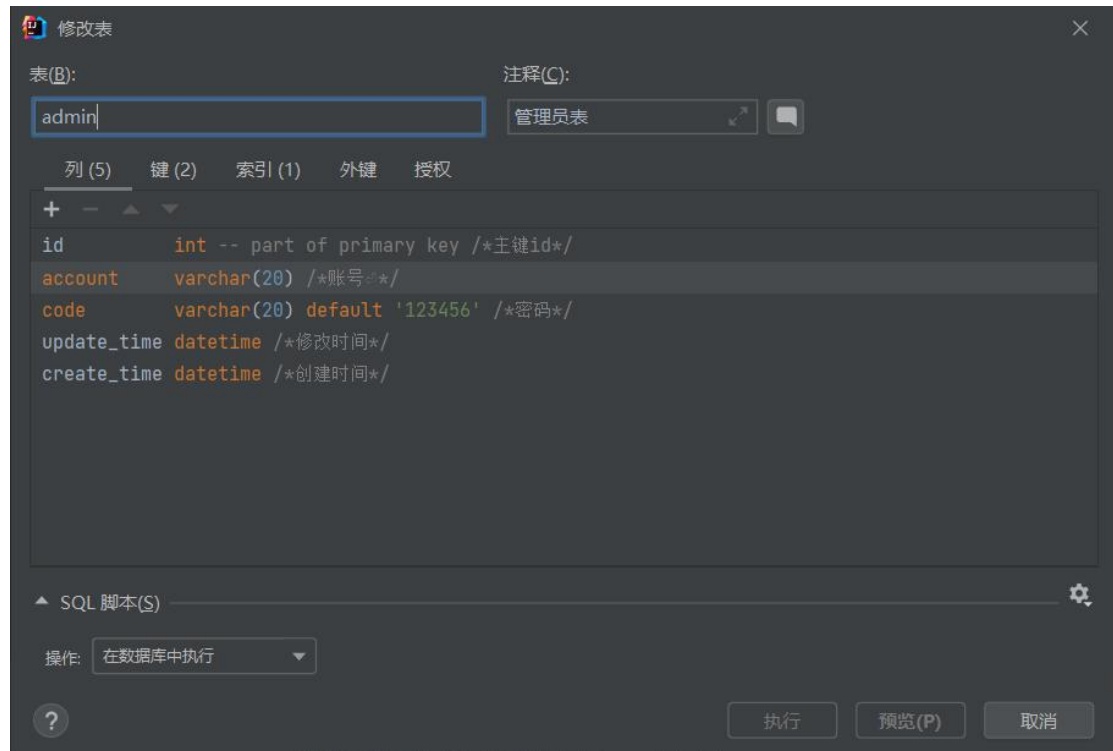
客户的顺序图：



四、实验步骤

在画出上面的用例图以及顺序图之后，我首先对数据里面需要创建的表进行了涉及，因为有客户登录，管理员登录，商品，购物车，四个表进行了设计，下面是通过可视化工具 idea 对表进行创建：

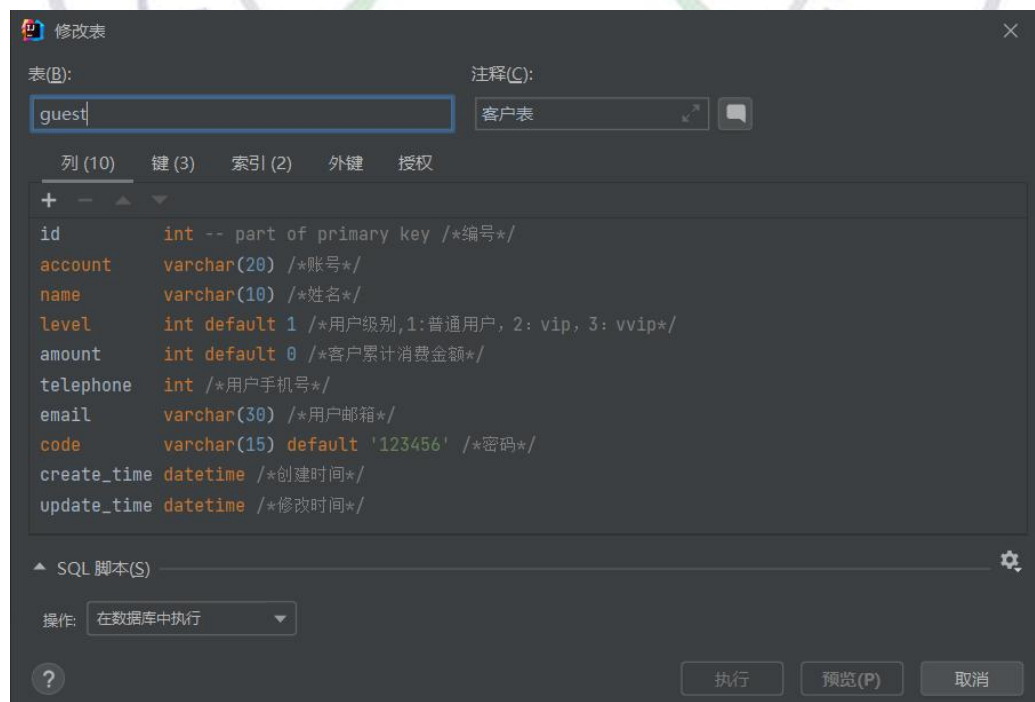
管理员表：



The screenshot shows the 'Modify Table' dialog for a table named 'admin'. The table has 5 columns: 'id' (int, primary key), 'account' (varchar(20)), 'code' (varchar(20) with default '123456'), 'update_time' (datetime), and 'create_time' (datetime). The dialog includes tabs for columns, keys, indexes, foreign keys, and permissions. At the bottom, there are buttons for 'Execute', 'Preview', and 'Cancel'.

列 (5)	键 (2)	索引 (1)	外键	授权
id	int -- part of primary key	/*主键id*/		
account	varchar(20)	/*账号*/		
code	varchar(20)	default '123456'	/*密码*/	
update_time	datetime	/*修改时间*/		
create_time	datetime	/*创建时间*/		

用户表：



The screenshot shows the 'Modify Table' dialog for a table named 'guest'. The table has 10 columns: 'id' (int, primary key), 'account' (varchar(20)), 'name' (varchar(10)), 'level' (int with default 1), 'amount' (int with default 0), 'telephone' (int), 'email' (varchar(30)), 'code' (varchar(15) with default '123456'), 'create_time' (datetime), and 'update_time' (datetime). The dialog includes tabs for columns, keys, indexes, foreign keys, and permissions. At the bottom, there are buttons for 'Execute', 'Preview', and 'Cancel'.

列 (10)	键 (3)	索引 (2)	外键	授权
id	int -- part of primary key	/*编号*/		
account	varchar(20)	/*账号*/		
name	varchar(10)	/*姓名*/		
level	int default 1	/*用户级别, 1: 普通用户, 2: vip, 3: vvip*/		
amount	int default 0	/*客户累计消费金额*/		
telephone	int	/*用户手机号*/		
email	varchar(30)	/*用户邮箱*/		
code	varchar(15)	default '123456'	/*密码*/	
create_time	datetime	/*创建时间*/		
update_time	datetime	/*修改时间*/		

商品表:

修改表

表(B): product 注释(C):

列 (10) 键 (2) 索引 (1) 外键 授权

+	-	▲	▼
id	int	-- part of primary key	/*产品编号*/
name	varchar(10)		/*产品名称*/
manufacturer	varchar(10)		/*生产厂家*/
date	date		/*生产日期*/
model	varchar(20)		/*型号*/
retail_price	int		/*零售价格*/
purchase_price	int		/*进货价*/
quantity	int		/*数量*/
create_time	datetime		/*创建时间*/
update_time	datetime		/*修改时间*/

SQL 脚本(S)

操作: 在数据库中执行

执行 预览(P) 取消

购物车表:

修改表

表(B): shopping_cart 注释(C): 购物车

列 (6) 键 (1) 索引 外键 授权

+	-	▲	▼
id	int	-- part of primary key	/*主键*/
guest_id	int		/*客户主键*/
product_id	int		/*商品主键*/
product_quantity	int		/*商品数量*/
update_time	datetime		/*修改时间*/
create_time	datetime		/*创建时间*/

SQL 脚本(S)

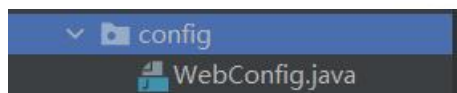
操作: 在数据库中执行

执行 预览(P) 取消

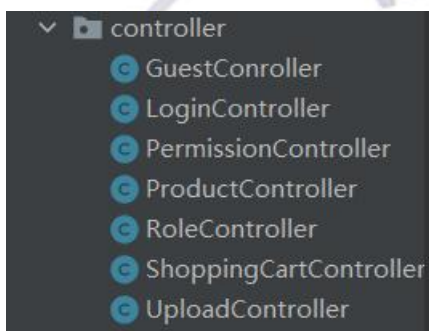
在编写程序的过程中,我采用了前后端分离的程序开发模式,在后端中我主要通过三个层来完成用户的每一项操作,分别为控制层 Controller 和业务逻辑层 Service 以及数据访问层 Dao。

因为需要在 java 程序中要对数据库进行操作，所以要根据数据库表的创建创建一个实体类，然后一个系统首先就是需要登录，所以登录之前就不能对其他的功能进行操作，在这里我采用了 **Interceptor** 拦截器进行身份验证，用到拦截器的时候需要一个配置类来配置和管理拦截器的行为和生命周期，这里同样会用到 **jwt** 令牌，所以需要创建一个工具类用来进行 **JWT** 令牌的生成。在系统操作中难免会有异常出现，我创建了一个异常类来对全局的异常做一个统一的处理。

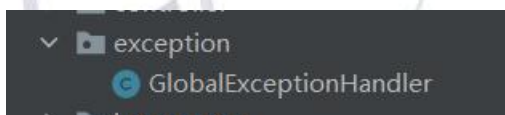
拦截器的配置类：



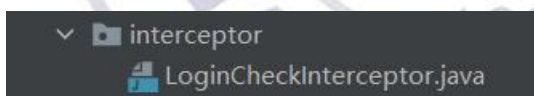
控制器类：



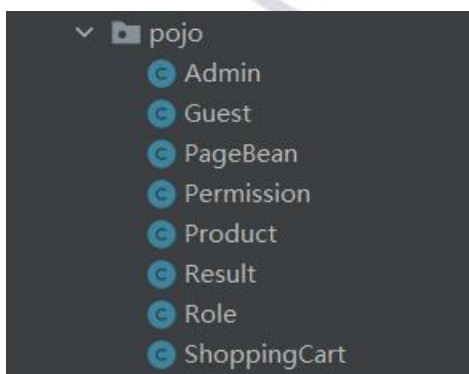
异常处理类：



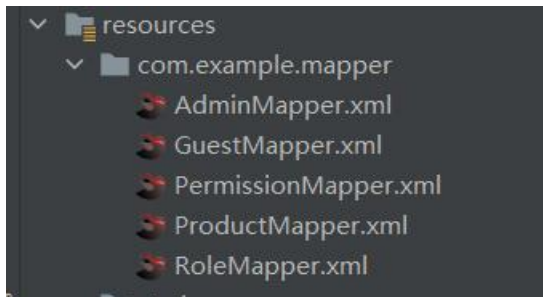
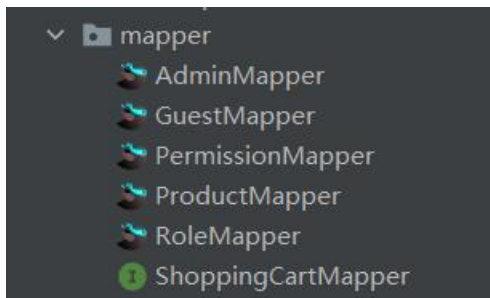
拦截器类：



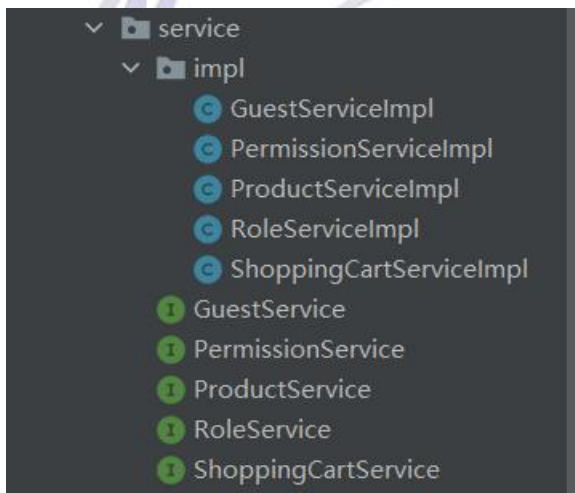
实体类：



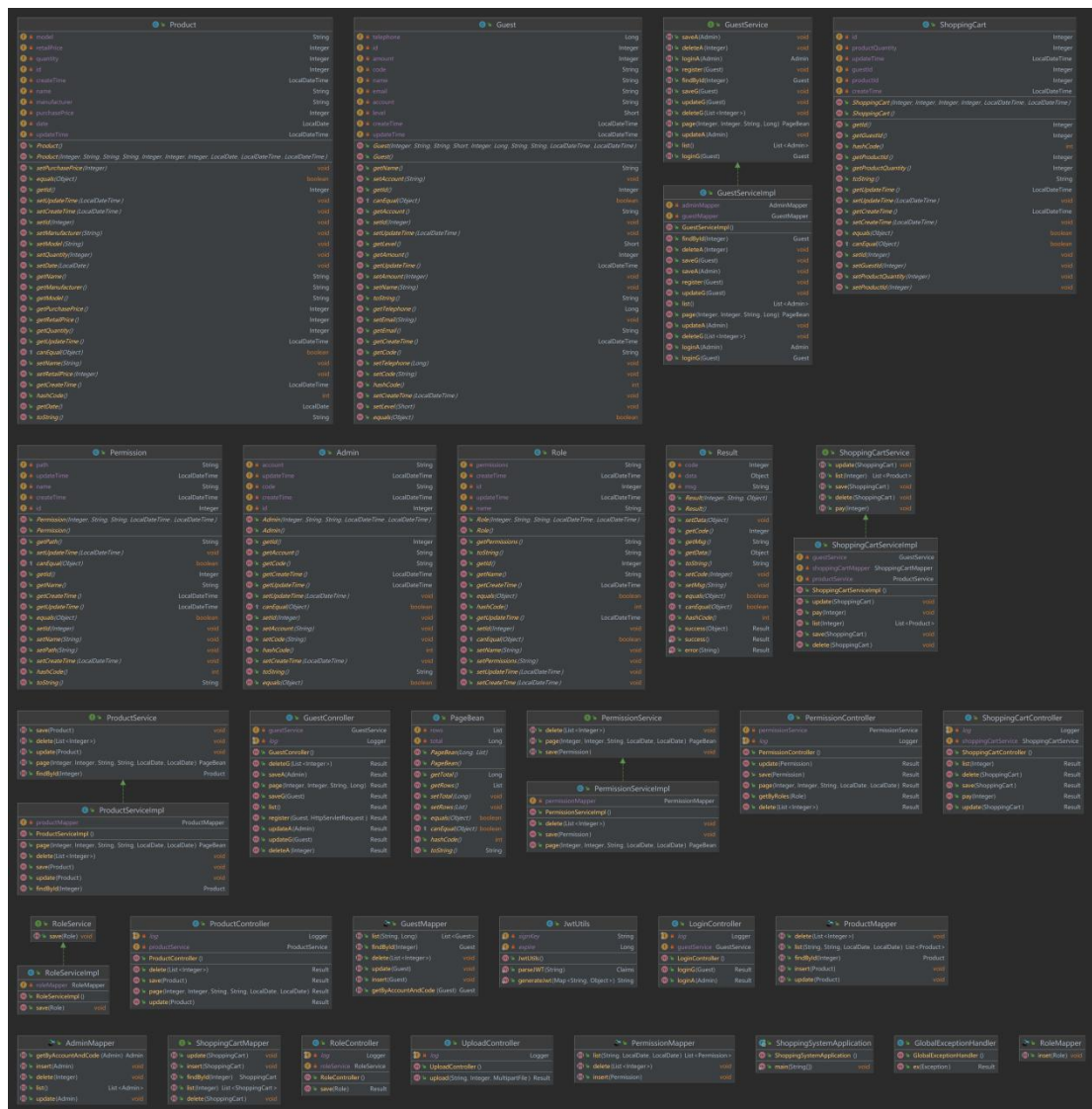
Mapper 类（Dao 层，因为用到了 mybatis，所以这里用 mapper 类）：



业务层:



整体的类图如下所示:



五、实验结果及分析

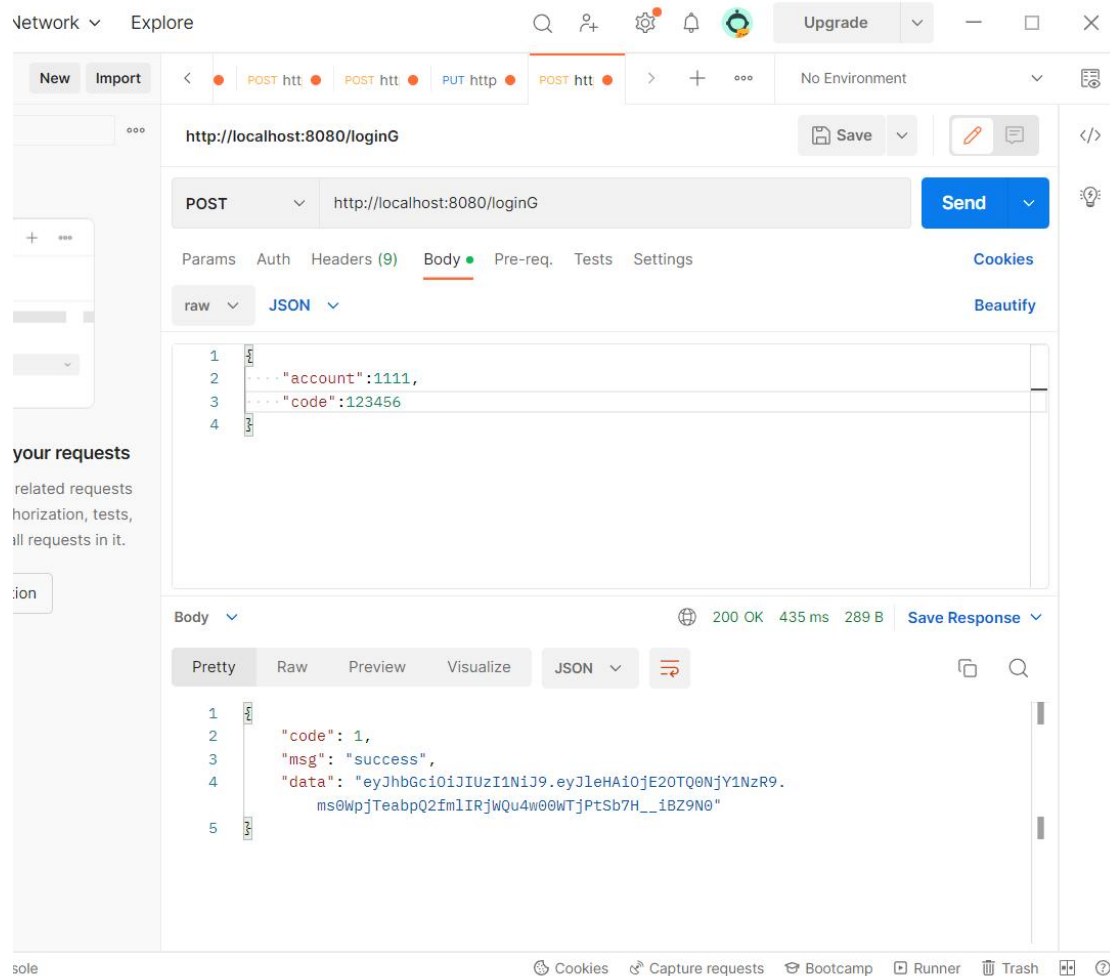
这里我使用了 postman 来对所有的功能进行测试：

1. 管理员与用户的登录功能：

在进行测试之前首先往数据库中添加了几条数据：

	id	account	name	level	amount	telephone	email	code
1	2	admin	admin	1	0	12345678	12345678	123456
2	3	111	商品管理	1	0	11111111	1234.html	123456
3	7	1111	wakee	1	2400	1111111111	1234.html	123456
4	8	11111	wakee	1	0	1111111111	1234.html	123456
5	9	12345	ldj	2	1000	110	234567jfkLfjs	admin
6	10	admi	gao	1	0	123456789	1214136881@qq.com	123456

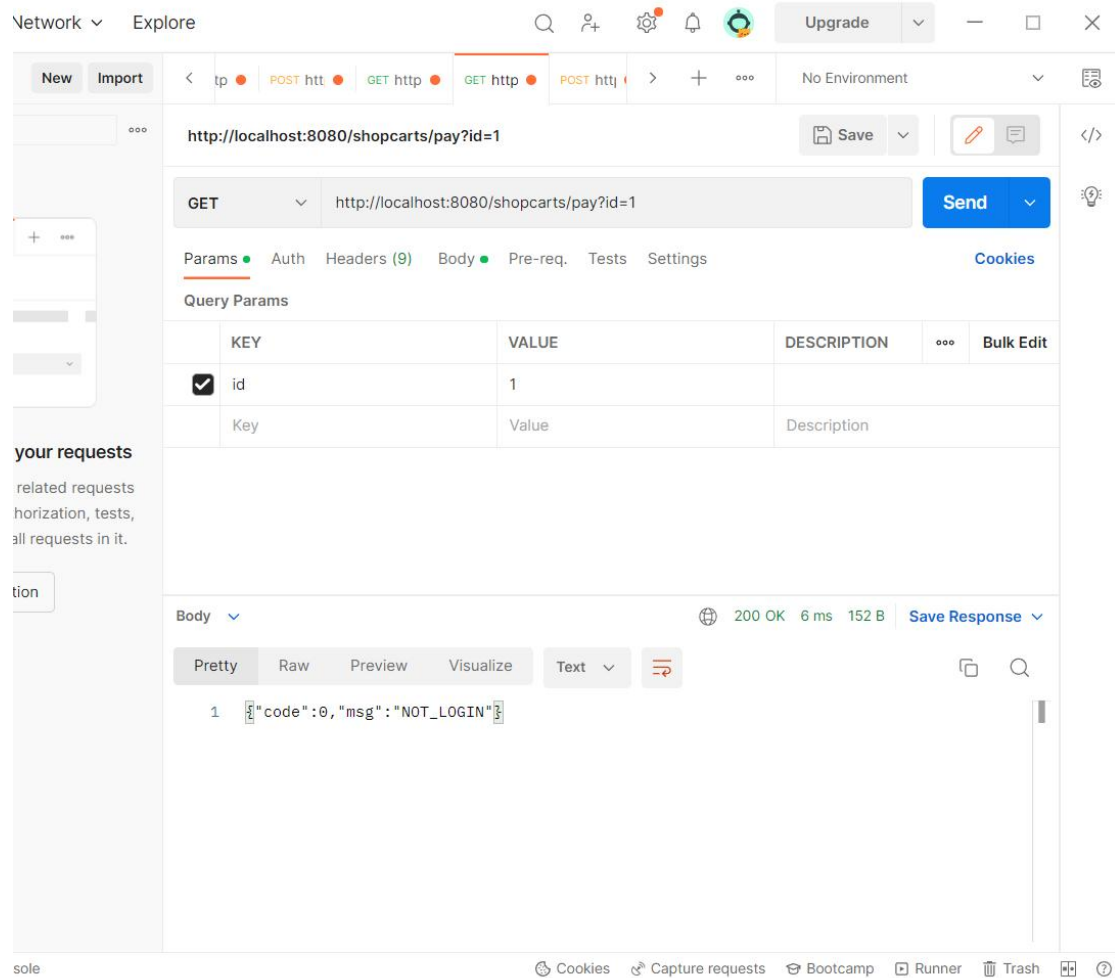
然后通过 postman 进行测试：



可以看到成功登录并且返回了令牌，并且在日志中显示出了有客户正在登录：

```
2023-09-11 17:09:34.603 INFO 27856 --- [nio-8080-exec-1] com.example.controller.LoginController : 顾客登录:Guest(id=null, account=1111, name=null, level=null, amount=null, tel=)
```

在这里我为了后续的测试关闭了拦截器，否则将无法进行其他功能的测试：



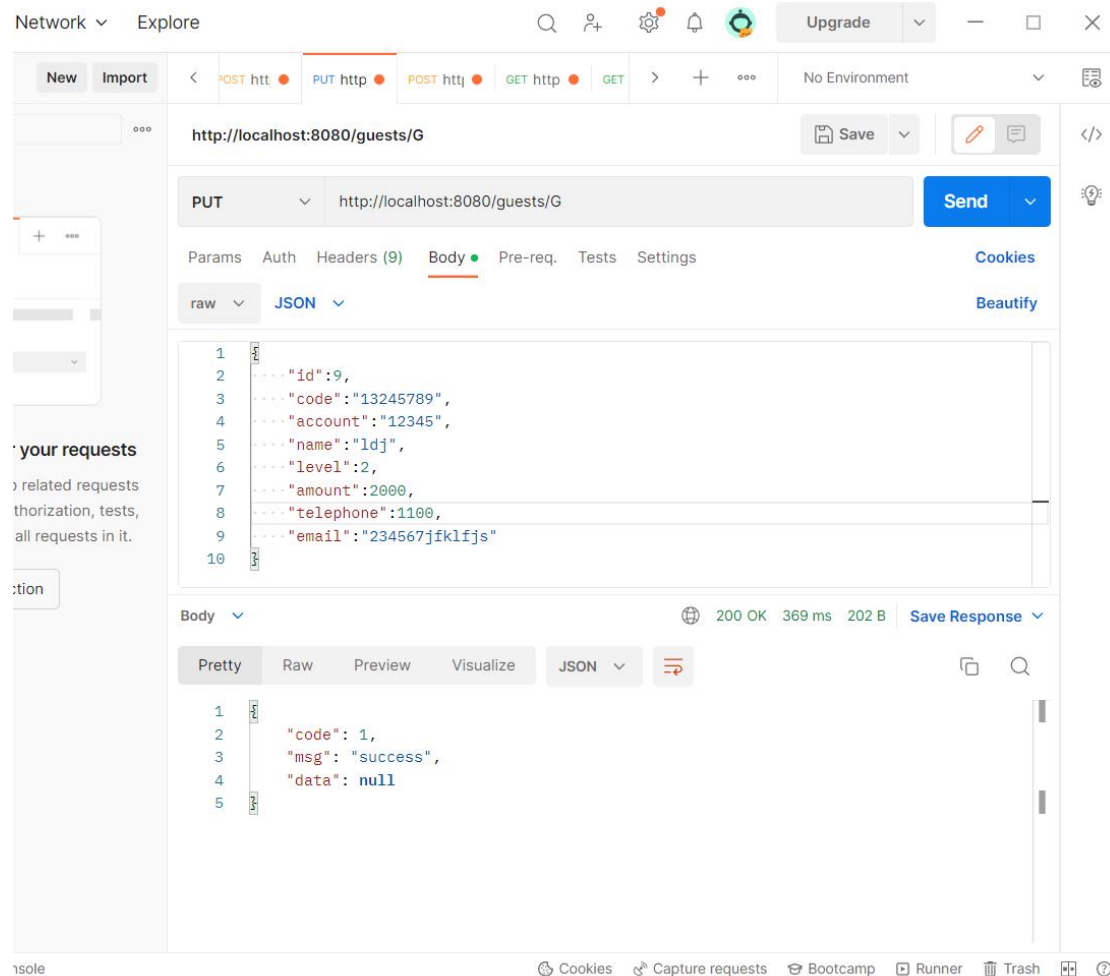
在这里我调用了付款的功能但是显示没有登录，所以不会造成没有登录便可以对其他功能进行操作的错误。

2. 管理员进行客户管理：

首先就是客户信息的修改，首先是修改之前的用户数据：

5	9	12345	ldj	2	1000	110	234567jfkLfjs	admin	2023-08-18 03:15:59	2023-08-18 17:31:44
---	---	-------	-----	---	------	-----	---------------	-------	---------------------	---------------------

运用 postman 进行操作：



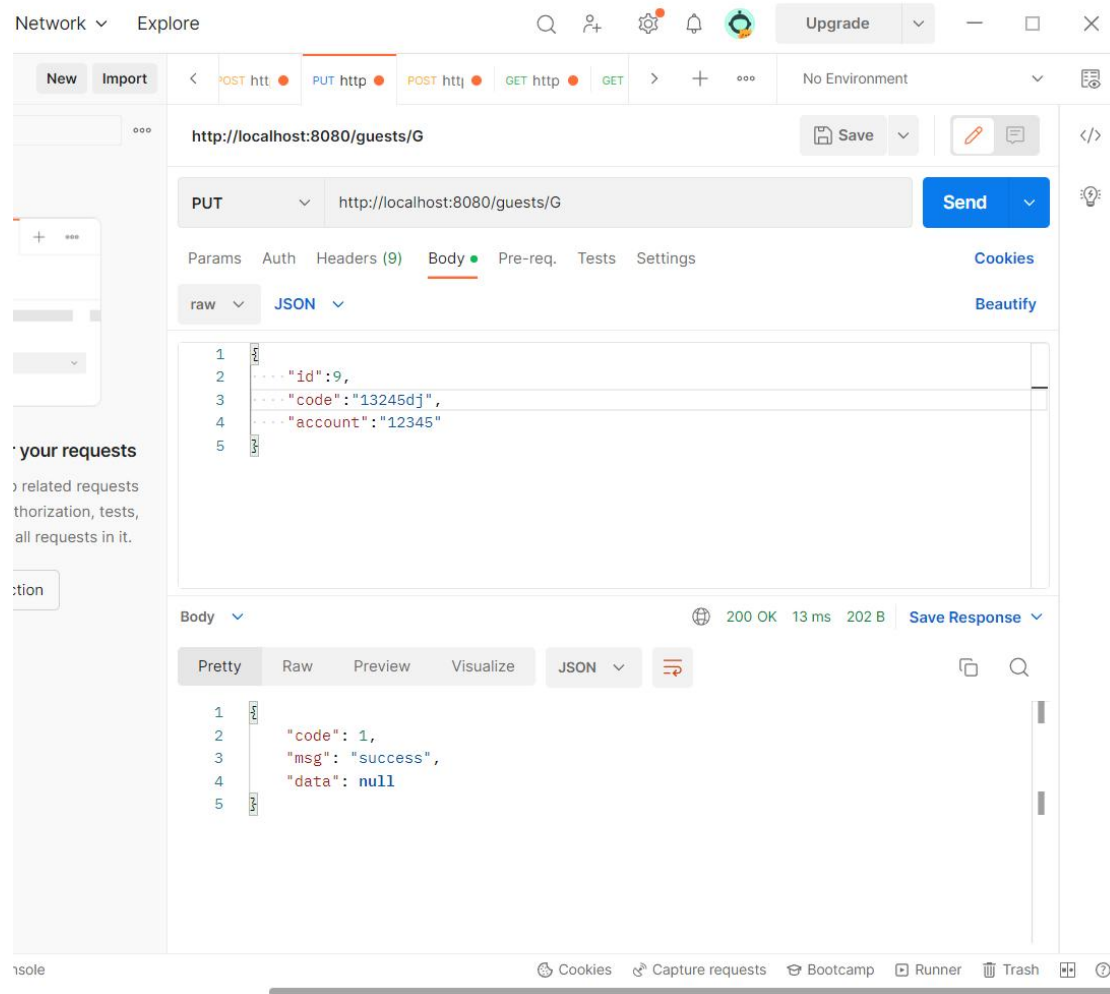
数据库中信息的变化：

5	9	12345	ldj	2	2000	1100	234567jfkLfjs	13245789	2023-08-18 03:15:59	2023-09-11 17:27:32
---	---	-------	-----	---	------	------	---------------	----------	---------------------	---------------------

可以看到用户的数据成功的修改，并且在 `updateTime` 可以看到时间发生了变化，而且因为我在 `mybatis` 中使用了 `if` 来判断数据是否为空：

```
<update id="update">
  update guest
  <set>
    <if test="account != null">account=#{account},</if>
    <if test="name != null">name=#{name},</if>
    <if test="level != null">level=#{level},</if>
    <if test="amount != null"> amount=#{amount},</if>
    <if test="telephone != null"> telephone=#{telephone},</if>
    <if test="email != null">email=#{email},</if>
    <if test="code != null"> code=#{code},</if>
    <if test="updateTime != null"> update_time=#{updateTime}</if>
  </set>
  --      这里的set标签可以自动的将多余的逗号去除掉。
  where id = #{id}</update>
```

所以这个功能还可以专门用来修改用户的密码：



数据库的变化：

5	9	12345	ldj	2	2000	1100	234567jfkLfjs	13245dj	2023-08-18 03:15:59	2023-09-11 17:30:51
---	---	-------	-----	---	------	------	---------------	---------	---------------------	---------------------

我们可以看到除了密码被改变，其他的数据都没有发生变化。

3. 管理员进行商品管理：

因为上面我主要介绍了修改信息的功能，在这里我主要讲解对商品的分页展示，查找以及批量删除功能：

首先是查找功能，当我不传任何参数的时候就是可以展示所有的商品信息：

Network ▾ Explore

New Import < http POST http PUT http POST http GET http > + ... No Environment ▾

http://localhost:8080/products Save ▾

GET ▾ http://localhost:8080/products Send ▾

Params Auth Headers (9) Body ● Pre-req. Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body ▾ 200 OK 77 ms 657 B Save Response ▾

Pretty Raw Preview Visualize JSON ▾

```
7 {
8   "id": 2,
9   "name": "mate60pro",
10  "manufacturer": "华为",
11  "model": "高端型",
12  "purchasePrice": 6999,
13  "retailPrice": 2000,
14  "quantity": 100000,
15  "date": "2023-08-29",
16  "createTime": "2023-09-11T17:43:44",
17  "updateTime": "2023-09-11T17:43:49"
18 },
19 {
20   "id": 1,
21   "name": "旺仔小牛奶",
22   "manufacturer": "wakee",
23   "model": "高端型",
24   "purchasePrice": 110,
25   "retailPrice": 20,
```

然后可以根据单个或多个条件进行查找：

Network Explore

httt POST httt PUT http POST httt GET http > + ... No Environment

http://localhost:8080/products?name=mate60pro

GET http://localhost:8080/products?name=mate60pro Send

Params Auth Headers (9) Body Pre-req. Tests Settings Cookies

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> name	mate60pro			
Key	Value	Description		

Body 200 OK 24 ms 438 B Save Response

Pretty Raw Preview Visualize JSON

```
2  "code": 1,
3  "msg": "success",
4  "data": {
5    "total": 1,
6    "rows": [
7      {
8        "id": 2,
9        "name": "mate60pro",
10       "manufacturer": "华为",
11       "model": "高端型",
12       "purchasePrice": 6999,
13       "retailPrice": 2000,
14       "quantity": 100000,
15       "date": "2023-08-29",
16       "createTime": "2023-09-11T17:43:44",
17       "updateTime": "2023-09-11T17:43:49"
18     }
19   ]
20 }
```

Network Explore

httt POST httt PUT http POST httt GET http > + ... No Environment

http://localhost:8080/products?manufacturer=wakee

GET http://localhost:8080/products?manufacturer=wakee Send

Params Auth Headers (9) Body Pre-req. Tests Settings Cookies

<input type="checkbox"/> name	mate60pro	
<input checked="" type="checkbox"/> manufacturer	wakee	
Key	Value	Description

Body 200 OK 19 ms 437 B Save Response

Pretty Raw Preview Visualize JSON

```
1  {
2    "code": 1,
3    "msg": "success",
4    "data": {
5      "total": 1,
6      "rows": [
7        {
8          "id": 1,
9          "name": "旺仔小牛奶",
10         "manufacturer": "wakee",
11         "model": "高端型",
12         "purchasePrice": 110,
13         "retailPrice": 20,
14         "quantity": 840,
15         "date": "2020-10-01",
16         "createTime": "2023-08-18T17:56:17",
17         "updateTime": "2023-08-18T23:03:28"
18       }
19     ]
20   }
```

因为数据量过少无法展示分页功能，但是我设置了分页展示功能：

```
@GetMapping
public Result page(@RequestParam(defaultValue = "1") Integer page, // @RequestParam的defaultValue属性是专门用来设置默认值的。
                  @RequestParam(defaultValue = "10") Integer pageSize,
                  String name, String manufacturer,
                  @DateTimeFormat(pattern = "yyyy-MM-dd") LocalDate begin,
                  @DateTimeFormat(pattern = "yyyy-MM-dd") LocalDate end){
    log.info("分页查询商品信息,参数:{}, {}, {}, {}, {}, {}, page, pageSize, name, manufacturer, begin, end);
```

然后就是批量删除的功能：

修改之前的数据库数据：

id	name	manufacturer	date	model	retail_price	purchase_price	quantity	create_time
1	旺仔小牛奶	wakee	2020-10-01	高端型	20	110	840	2023-08-18 17:56:17
2	mate60pro	华为	2023-08-29	高端型	2000	6999	100000	2023-09-11 17:43:44
3	mateX5	华为	2023-09-05	高端型	2000	12000	100000	2023-09-11 17:54:37
4	iphone15	apple	2023-09-13	高端型	2000	5999	10000	2023-09-11 17:56:14

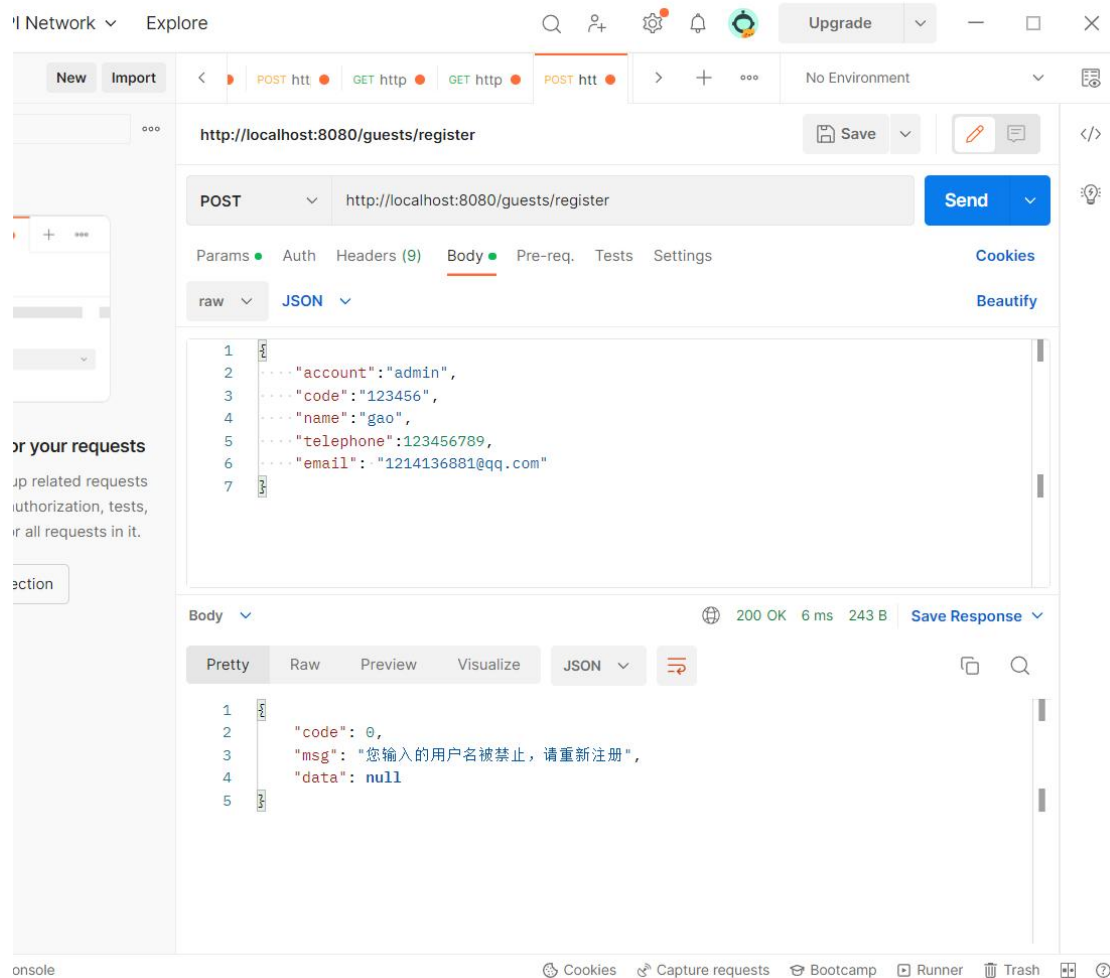
Postman interface showing a DELETE request to `http://localhost:8080/products/2,3,4`. The response is a JSON object: `{\"code\": 1, \"msg\": \"success\", \"data\": null}`. The status is 200 OK.

删除后的数据库数据：

id	name	manufacturer	date	model	retail_price	purchase_price	quantity	create_time
1	旺仔小牛奶	wakee	2020-10-01	高端型	20	110	840	2023-08-18 17:56:17

4. 用户进行注册功能：

注册功能没有什么独特的地方，就是注册的时候不能与管理员的账号发生重复：



如上若是用户自定义的账号等于 `admin` 的话会使得注册过程失败。

5. 用户购物功能:

这部分我主要介绍一下结账功能: 结账功能会涉及到三个表的操作。

在操作之前的数据库信息:

客户表:

id	account	name	level	amount	telephone	email	code	create_time	update_time
2	admin	admin	1	0	12345678	12345678	123456	2023-08-18 02:03:13	2023-08-18 02:03:13
3	111	商品管理	1	0	11111111	1234.html	123456	2023-08-18 03:11:31	2023-08-18 03:11:31
7	1111	wakee	1	2400	11111111	1234.html	123456	2023-08-18 03:14:01	2023-08-18 23:03:28
8	11111	wake	1	0	11111111	1234.html	123456	2023-08-18 03:15:10	2023-08-18 03:15:10
9	12345	ldj	2	2000	1100	234567jfkLfjs	13245dj	2023-08-18 03:15:59	2023-09-11 17:30:51
10	admi	gao	1	0	123456789	1214136881@qq.com	123456	2023-08-20 22:09:39	2023-08-20 22:09:39

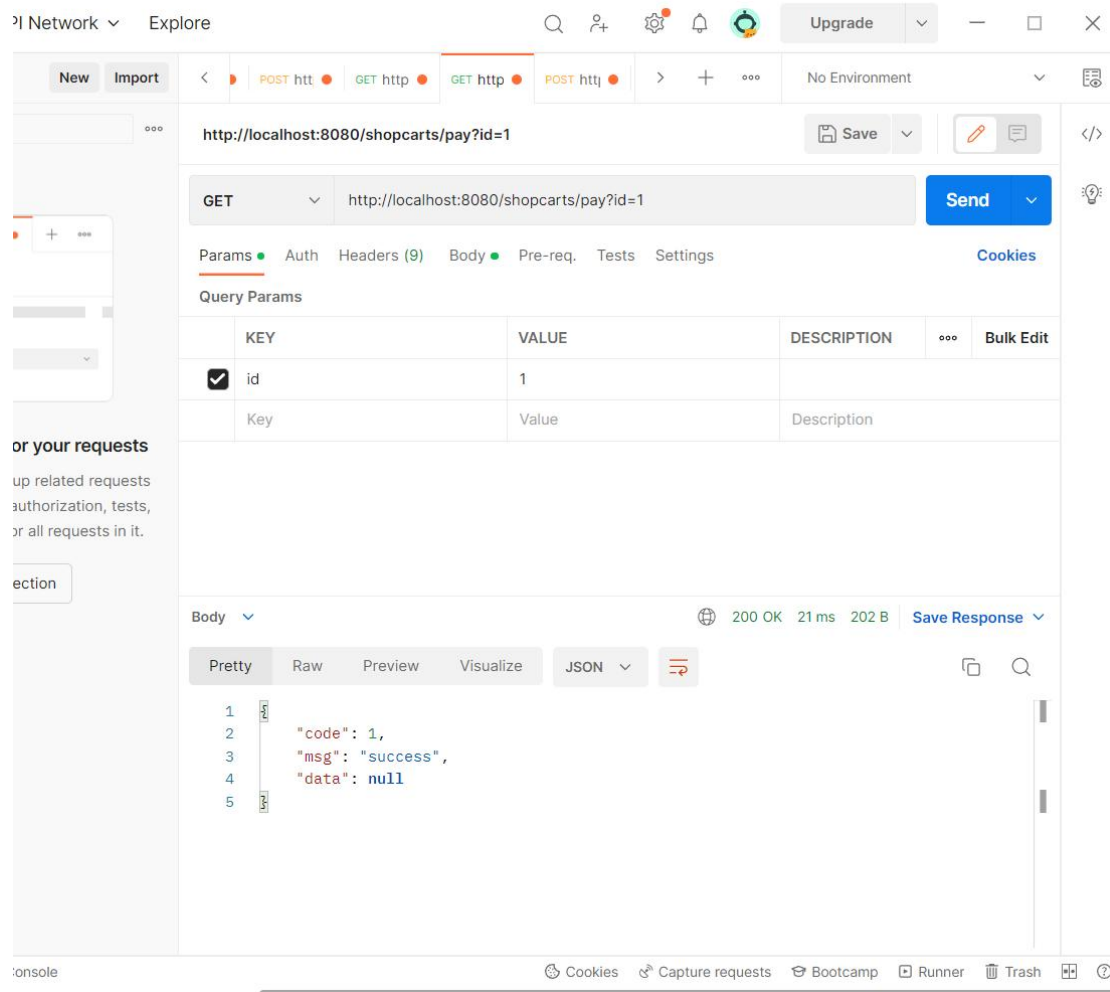
商品表:

id	name	manufacturer	date	model	retail_price	purchase_price	quantity	create_time
1	旺仔小牛奶	wakee	2020-10-01	高端型	20	110	840	2023-08-18 17:56:

购物车表:

id	guest_id	product_id	product_quantity	update_time	create_time
1	1	9	1	100 2023-09-11 18:04:37	2023-09-11 18:04:40

通过 postman 进行付款操作：



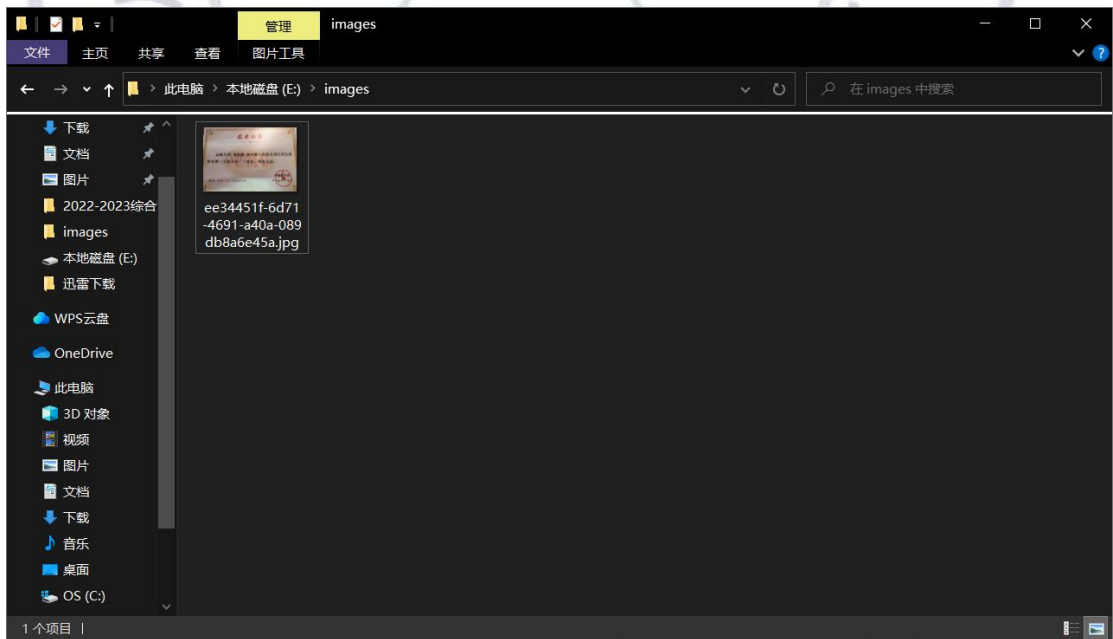
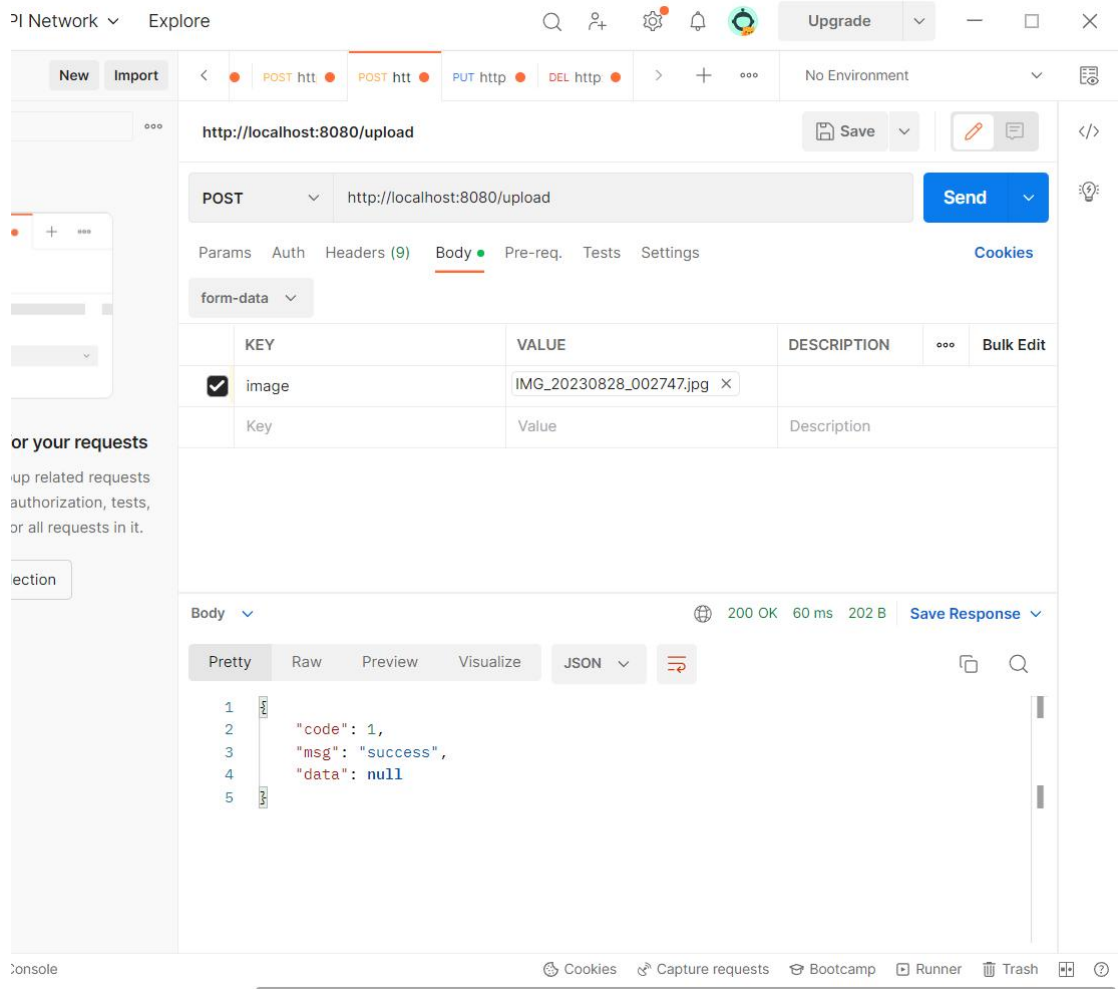
操作之后数据库中数据的变化：

	id	account	name	level	amount	telephone	email	code	create_time	update_time
1	2	admin	admin	1	0	12345678	12345678	123456	2023-08-18 02:03:13	2023-08-18 02:03:13
2	3	111	商品管理	1	0	11111111	1234.html	123456	2023-08-18 03:11:31	2023-08-18 03:11:31
3	7	1111	wakee	1	2400	11111111	1234.html	123456	2023-08-18 03:14:01	2023-08-18 23:03:28
4	8	11111	wake	1	0	11111111	1234.html	123456	2023-08-18 03:15:10	2023-08-18 03:15:10
5	9	12345	ldj	2	4000	1100	234567jfkLfjs	13245dj	2023-08-18 03:15:59	2023-09-11 18:08:22
6	10	admi	gao	1	0	123456789	1214136881@qq.com	123456	2023-08-20 22:09:39	2023-08-20 22:09:39

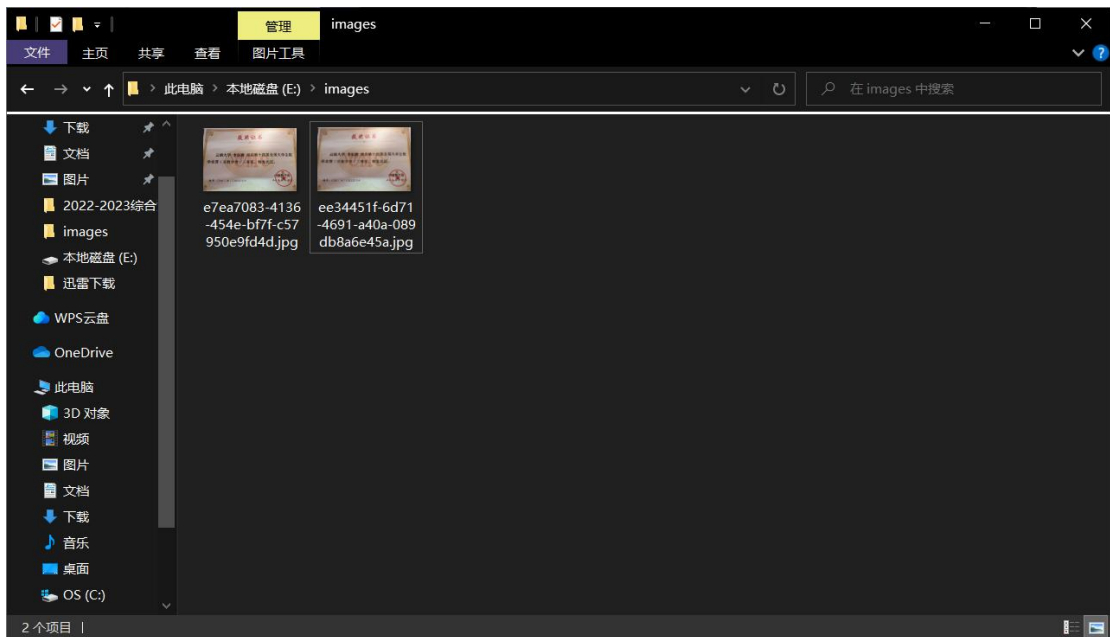
	id	name	manufacturer	date	model	retail_price	purchase_price	quantity	create_time	update
1		旺仔小牛奶	wakee	2020-10-01	高端型	20	110	740	2023-08-18 17:56:17	2023-09-

可以看到用户表中的顾客的消费总金额发生了变化，并且更新时间发生了变化，而且商品表中的商品的数量发生了变化。

最后因为商品的信息中可能会涉及到图片的信息，所以在这里我加了一个上传照片的功能：

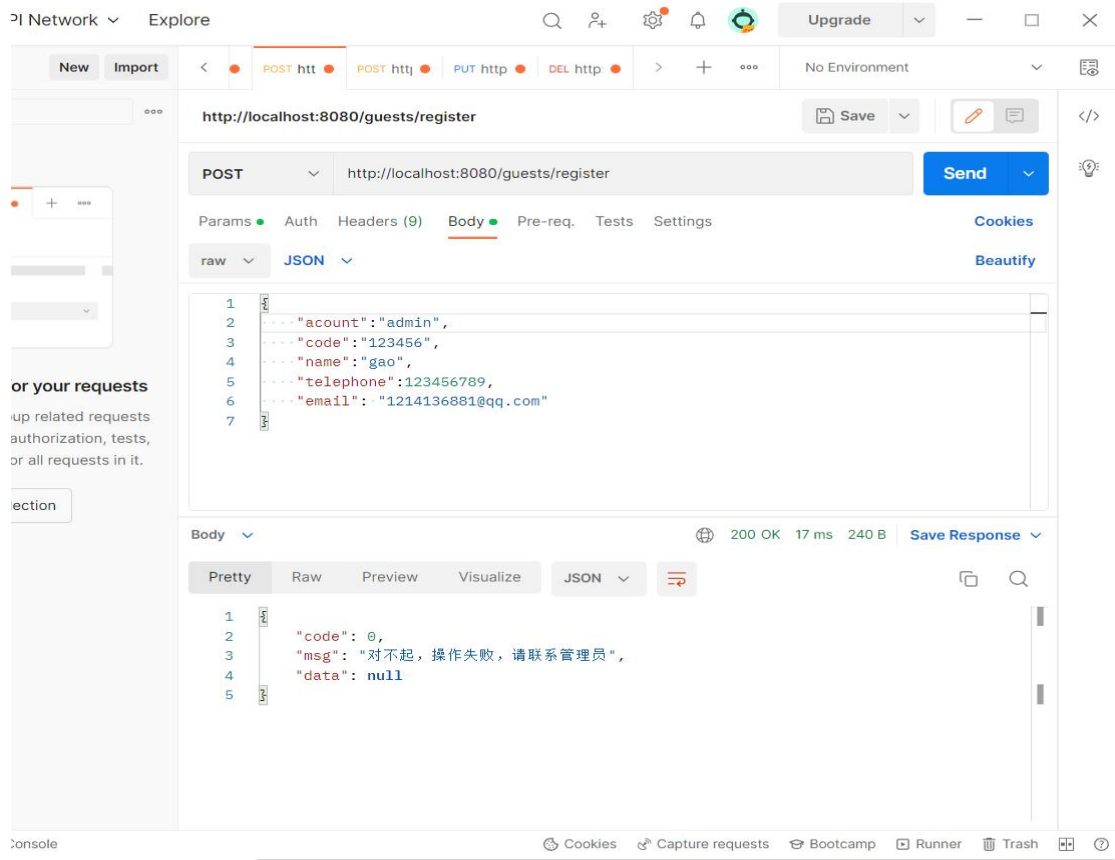


可以看到照片被成功的存到了对应的本地位置，当然这里也可以存到云服务中，并且当我们再次启用相同的功能存储相同的照片时：



可以发现两次的名字会不一样。

最后就是异常处理，在这个简单的购物系统中我没有将异常进行细分，只是简单的对异常进行处理：




```
Closing non transactional SqlSession [org.apache.ibatis.session.defaults.DefaultSqlSession@39a8d845]
java.lang.NullPointerException: Create breakpoint : Cannot invoke "String.equals(Object)" because the return value of "com.example.pojo.Guest.getAccount()" is null
at com.example.controller.GuestController.register(GuestController.java:136) <14 个内部行>
at javax.servlet.http.HttpServlet.service(HttpServlet.java:559) <1 个内部行>
at javax.servlet.http.HttpServlet.service(HttpServlet.java:623) <33 个内部行>
```

这里假如账号 `account` 被错误拼写成了 `acount` 时，会显示操作失败，请联系管理员。

	id	account	name	level	amount	telephone	email	code	create_time	update_time
1	2	admin	admin	1	0	12345678	12345678	123456	2023-08-18 02:03:13	2023-08-18 02:03:13
2	3	111	商品管理	1	0	11111111	1234.html	123456	2023-08-18 03:11:31	2023-08-18 03:11:31
3	7	1111	wakee	1	2400	11111111	1234.html	123456	2023-08-18 03:14:01	2023-08-18 23:03:28
4	8	11111	wakee	1	0	11111111	1234.html	123456	2023-08-18 03:15:10	2023-08-18 03:15:10
5	9	12345	ldj	2	4000	1100	234567jfklfjs	13245dj	2023-08-18 03:15:59	2023-09-11 18:08:22
6	10	admi	gao	1	0	123456789	1214136881@qq.com	123456	2023-08-20 22:09:39	2023-08-20 22:09:39

并且这些错误的操作并不会执行，所以不会对数据库造成改变。

六、实验总结及体会

本次实验中，我在学校进行的几天中，我打算的是像大一写宿舍管理系统一样，用纯 java 代码写出一个在控制台中进行操作的代码，很多类也已经创建好了，但是我觉得这样写起来很麻烦，越写越乱，所以我放弃了一开始写的一切，学习了 SpringBoot，Maven 以及 mybatis。然后重新开始写购物系统，写的十分顺利，而且因为有三个层，所以修改以及思路都十分的清晰，但是由于学的还是不深并且我没有学习过前端，所以没有给这个系统加上前端，这是非常遗憾的一点，其次就是我开始的时候打算的是给系统再加上两个表，一个为角色表，一个为权限表，角色表中我打算的是加入管理员，金牌用户等不同的角色，而权限表中加入不同的权限，给不同的用户提供不同的福利什么的，但是因为这里的角色表中会包含多种权限，但是我不知如何在表中存储这么多权限，所以我尝试了两天之后，我无奈放弃了这个功能。其实这个系统还远远需要进行完善，就比如事务，有些操作必须同时进行，若某些地方出现错误，便需要将之前的所有操作全部撤回。

七、教师评语

(中文字体宋体、英文字体 Times New Roman，小四，1.5 倍行距，首行缩进两个字符，**请删除此行**)

(A4 双面打印，**请删除此行**)