

**数据库系统原理实践报告**

|  |  |
| --- | --- |
| 项目名称： | 网上销售系统 |
|  |  |
| 姓 名： | 聂鸿勇 |
| 专 业： | 计算机科学与技术 |
| 班 级： | CS1707 |
| 学 号： | U201714785 |
| 指导教师： | 吴海 |

|  |  |
| --- | --- |
| 分数 |  |
| 教师签名 |  |

2020年6月24日

**教师评分页**

|  |  |
| --- | --- |
| 子目标 | 子目标评分 |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
|  |  |

目 录

[1 课程任务概述 1](#_Toc43883321)

[2 软件功能学习 1](#_Toc43883322)

[2.1任务要求 1](#_Toc43883323)

[2.2完成过程 1](#_Toc43883324)

[2.3任务总结 4](#_Toc43883325)

[3 SQL语句 5](#_Toc43883326)

[3.1任务要求 5](#_Toc43883327)

[3.2 完成过程 7](#_Toc43883328)

[3.3任务总结 23](#_Toc43883329)

[4 综合实践任务 24](#_Toc43883330)

[4.1系统设计目标 24](#_Toc43883331)

[4.2 需求分析 24](#_Toc43883332)

[4.3 总体设计 26](#_Toc43883333)

[4.4 数据库设计 27](#_Toc43883334)

[4.5 详细设计与实现 30](#_Toc43883335)

[4.6 系统测试 32](#_Toc43883336)

[4.7 系统设计与实现总结 40](#_Toc43883337)

[4 课程总结 40](#_Toc43883338)

[附录 41](#_Toc43883339)

# 课程任务概述

·熟悉数据库软件的使用，练习备份和创建用户和配置用户权限等基本功能的使用。

·利用SQL语言实现对数据库的增删改查等操作。

·进行一个数据库应用系统的设计，完成一个小型数据库应用系统的设计与实现工作。

# 软件功能学习

## 2.1任务要求

完成下列1-2题，并在实验报告中叙述过程，可适当辅以插图

1）练习sqlserver的两种完全备份方式：数据和日志文件的脱机备份。

2）练习在新增的数据库上增加用户并配置权限的操作，通过用创建的用户登录数据库并且执行未经授权的SQL语句验证自己的权限配置是否成功。

## 2.2完成过程

### 2.2.1练习两种完全备份方式

方法一：

在根目录下选择备份，然后选择需要备份的数据库lab2，新建一个备份备注为lab1任务的备份记录，图 2‑1。

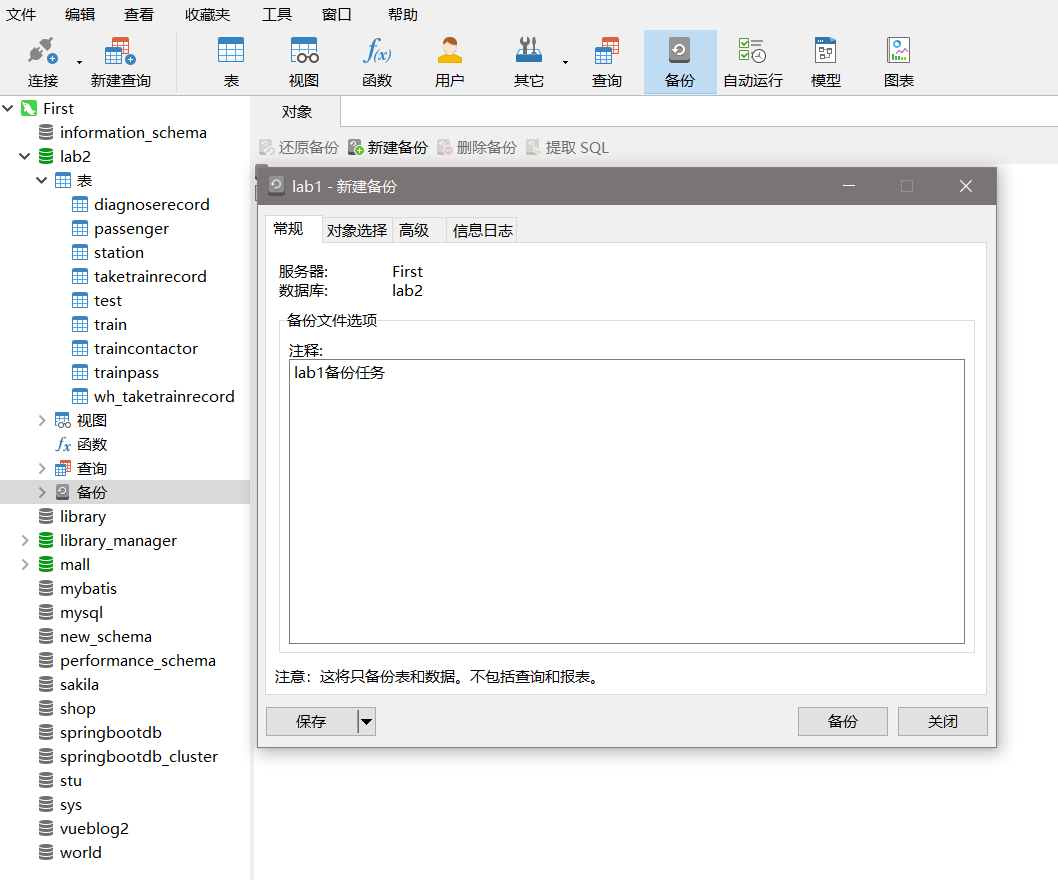


图 2‑1

选择备份后在数据库的备份中生成了一个备份文件，图 2‑2。

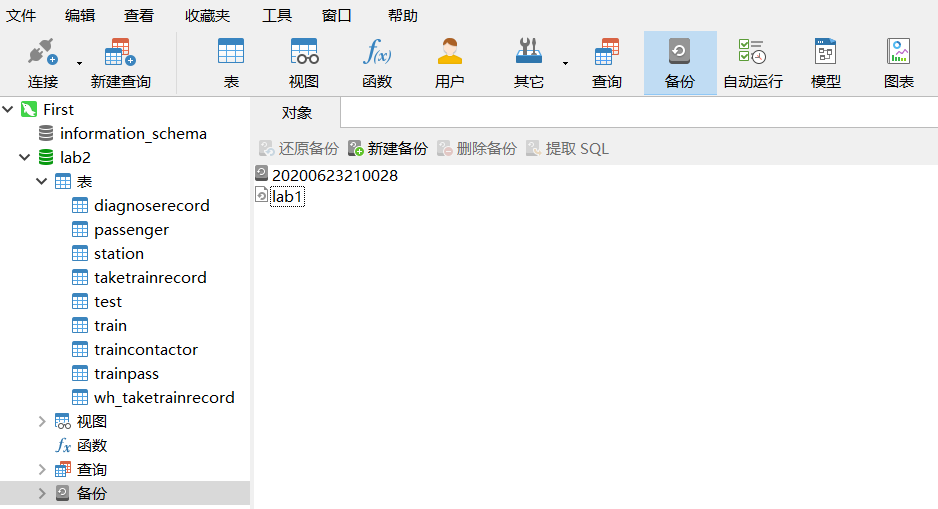


图 2‑2

方法二：

将数据库备份到本地，右键选中需要备份的数据库lab2，在转储SQL文件中可以看到两个备份选项，可以根据自己的需求来选择备份的内容，图 2‑3。

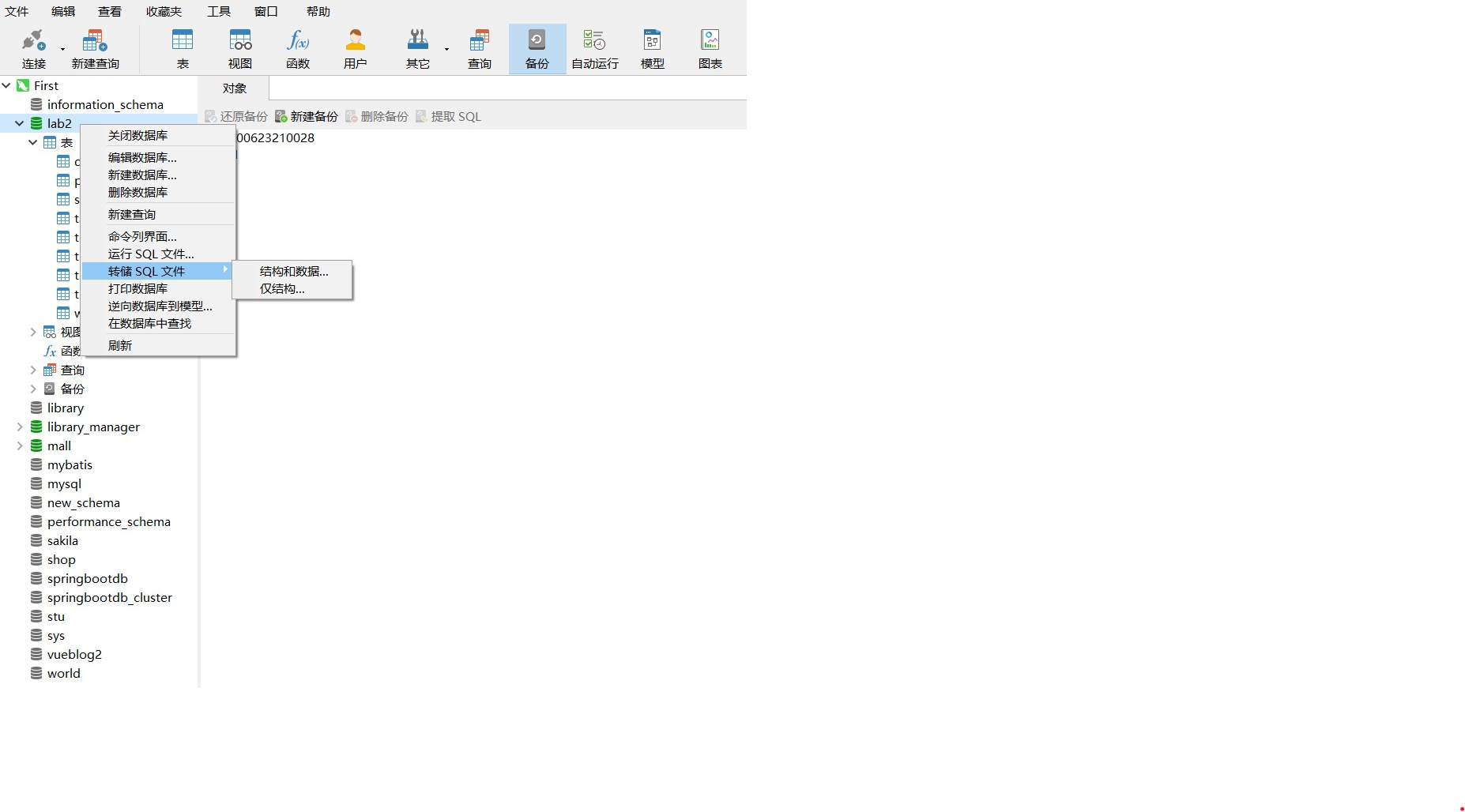


图 2‑3

备份完成后，在本地生成了一个sql文件，可以直接利用该sql文件重建当前备份的数据库，图 2‑4。

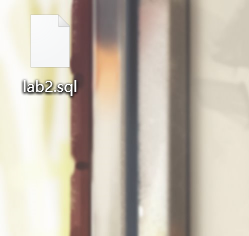


图 2‑4

### 2.2.2练习增加用户并配置权限

在根目录下选择用户，然后选择新建用户，新建一个登录名为sun，密码为123456的新用户图 2‑5。

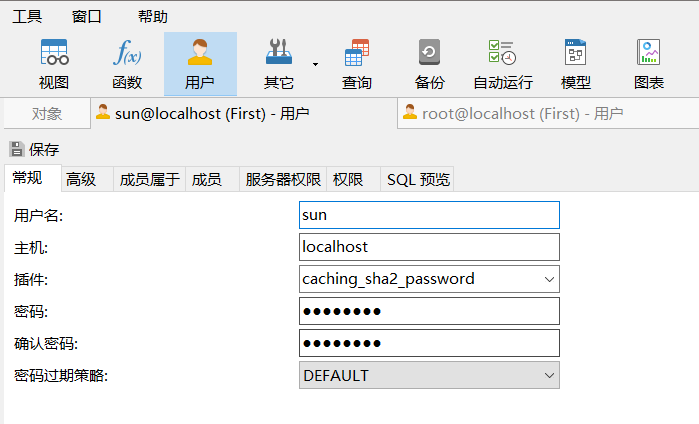


图 2‑5

在服务器权限设置中勾选需要赋予的权限，图 2‑6。

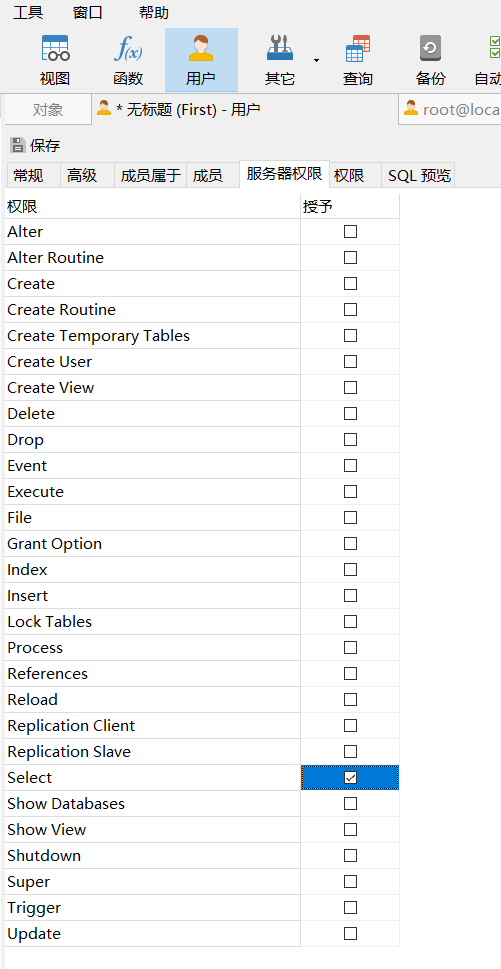


图 2‑6

创建完成后可以在用户界面看到新增了刚才创建的用户nhy，图 2‑7。

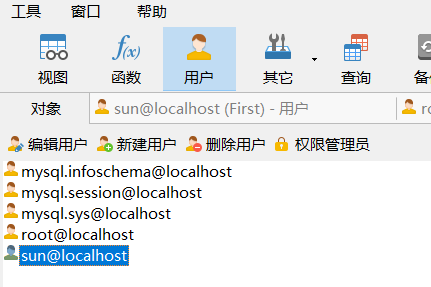


图 2‑7

使用新用户连接数据库，测试新用户的权限问题，因为只给新用户查询权限，所以当该用户在数据库中进行其他更新、修改操作时出现报错，图 2‑8。

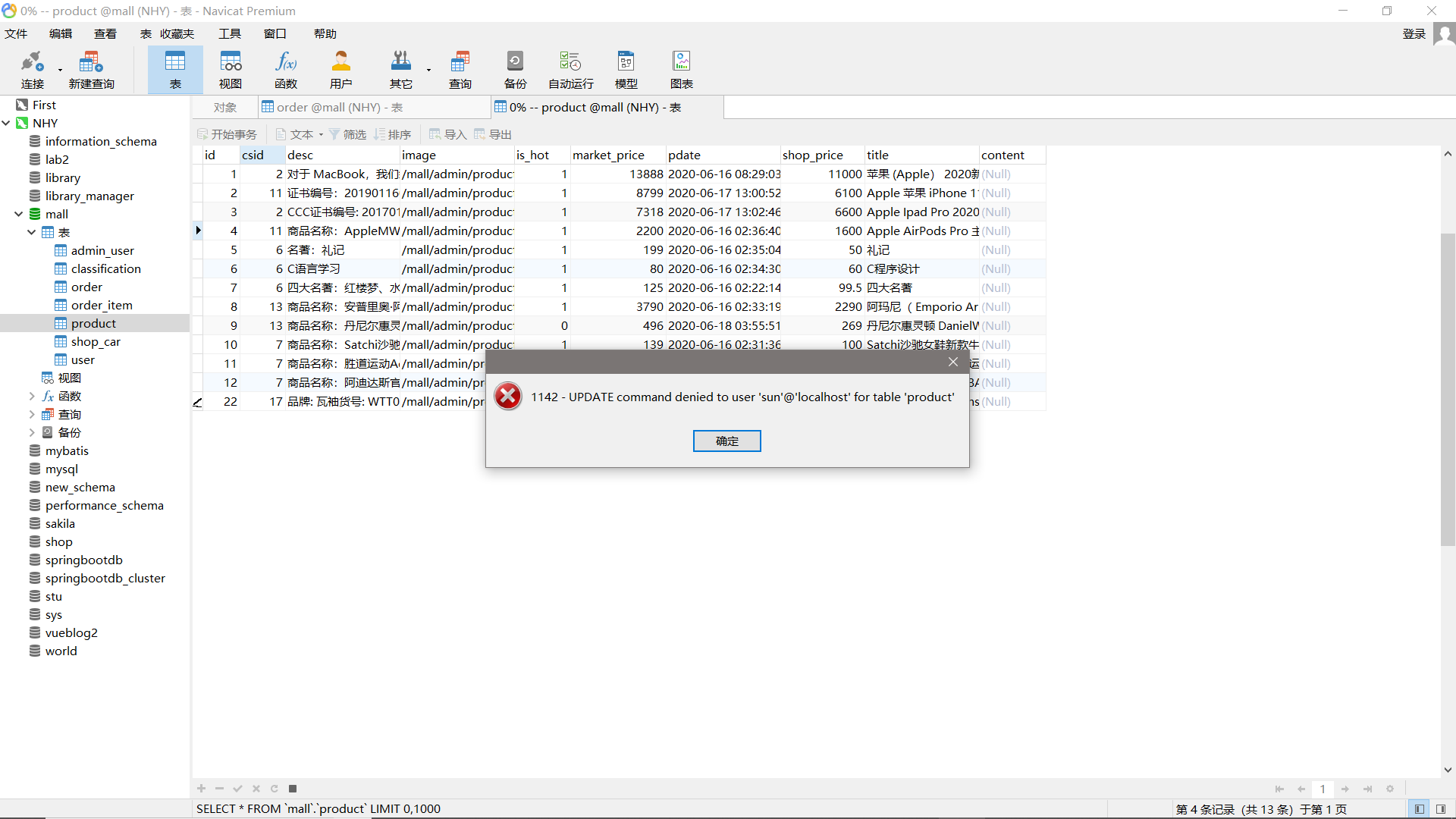


图 2‑8

## 2.3任务总结

在刚开始接触Navicat Premium时，对该软件的操作还不太熟练，有很多的功能都不太清楚，在实验的过程中，通过查询资料学会了如何使用Navicat Premium操作数据库。

# SQL语句

## 3.1任务要求

### 3.1.1建表

1）创建下列跟“疫期乘坐列车”相关的关系，包括主码和外码的说明

车站表【车站编号，车站名，所属城市】

Station (SID int, SName char(20), CityName char(20))

其中，主码为车站编号。

车次表【列车流水号，发车日期，列车名称，起点站编号，终点站编号，开出时刻，终点时刻】

Train (TID int, SDate date, TName char(20), SStationID int, AStationID int, STime datetime, ATime datetime)

其中，TID为主码，(列车名称，发车日期)为候选码；SStationID和AStationID都来源于车站表的SID。

车程表【列车流水号，车站序号，车站编号，到达时刻，离开时刻】

TrainPass (TID int, SNo smallint, SID int, STime datetime, ATime datetime)

其中，主码为(TID,SNo)。SID来源于车站表的SID。

乘客表【乘客身份证号，姓名，性别，年龄】

Passenger (PCardID char(18), PName char(20), Sex bit, Age smallint)

其中，主码为乘客身份证号；性别取值为0/1（“1”表示“男”，“0”表示“女”）。

乘车记录表【记录编号，乘客身份证号，列车流水号，出发站编号，到达站编号，车厢号，席位排号，席位编号，席位状态】

TakeTrainRecord (RID int, PCardID char(18), TID int, SStationID int, AStationID int, CarrigeID smallint, SeatRow smallint，SeatNo char(1)，SStatus int)

其中，主码、外码请依据应用背景合理定义。

CarrigeID若为空，则表示“无座”；

SeatNo只能取值为’A’、’B’、’C’、’E’、’F’，或为空值；

SStatus只能取值’0’（退票）、’1’（正常）、’2’（乘客没上车）。

诊断表【诊断编号，病人身份证号，诊断日期，诊断结果，发病日期】

DiagnoseRecord (DID int, PCardID char(18), DDay date, DStatus smallint, FDay date)

其中，主码为DID；DStatus包括：1：新冠确诊；2：新冠疑似；3：排除新冠

乘客紧密接触者表【接触日期, 被接触者身份证号，状态，病患身份证号】

TrainContactor (CDate date, CCardID char(18), DStatus smallint, PCardID char(18))

其中，主码为全码。DStatus包括：1：新冠确诊；2：新冠疑似；3：排除新冠

2）观察性实验

验证在建立外码时是否一定要参考被参照关系的主码，并在实验报告中简述过程和结果。

3）数据准备

依据后续实验的要求，向上述表格中录入适当数量的实验数据，从而对相关的实验任务能够起到验证的作用。

### 3.1.2数据更新

1）分别用一条sql语句完成对乘车记录表基本的增、删、改的操作；

2）批处理操作

将乘车记录表中的从武汉出发的乘客的乘车记录插入到一个新表WH\_TakeTrainRecord中。

3）数据导入导出

通过查阅DBMS资料学习数据导入导出功能，并将任务2.1所建表格的数据导出到操作系统文件，然后再将这些文件的数据导入到相应空表。

4）观察性实验

建立一个关系，但是不设置主码，然后向该关系中插入重复元组，然后观察在图形化交互界面中对已有数据进行删除和修改时所发生的现象。

5）创建视图

创建一个新冠确诊病人的乘火车记录视图，其中的属性包括：身份证号、姓名、年龄、乘坐列车编号、发车日期、车厢号，席位排号，席位编号。按身份证号升序排序，如果身份证号一致，按发车日期降序排序（注意，如果病人买了票但是没坐车，不计入在内）。

6）触发器实验

编写一个触发器，用于实现以下完整性控制规则：

1) 当新增一个确诊患者时，若该患者在发病前14天内有乘车记录，则将其同排及前后排乘客自动加入“乘客紧密接触者表”，其中：接触日期为乘车日期。

2) 当一个紧密接触者被确诊为新冠时，从“乘客紧密接触者表”中修改他的状态为“1”。

### 3.1.3查询

请分别用一条SQL语句完成下列各个小题的查询需求：

1）查询确诊者“张三”的在发病前14天内的乘车记录；

2）查询所有从城市“武汉”出发的乘客乘列车所到达的城市名；

3）计算每位新冠患者从发病到确诊的时间间隔（天数）及患者身份信息，并将结果按照发病时间天数的降序排列；

4）查询“2020-01-22”从“武汉”发出的所有列车；

5）查询“2020-01-22”途经“武汉”的所有列车；

6）查询“2020-01-22”从武汉离开的所有乘客的身份证号、所到达的城市、到达日期；

7）统计“2020-01-22” 从武汉离开的所有乘客所到达的城市及达到各个城市的武汉人员数。

8）查询2020年1月到达武汉的所有人员；

9） 查询2020年1月乘车途径武汉的外地人员（身份证非“420”开头）；

10）统计“2020-01-22”乘坐过‘G007’号列车的新冠患者在火车上的密切接触乘客人数（每位新冠患者的同车厢人员都算同车密切接触）。

11）查询一趟列车的一节车厢中有3人及以上乘客被确认患上新冠的列车名、出发日期，车厢号；

12）查询没有感染任何周边乘客的新冠乘客的身份证号、姓名、乘车日期；

13）查询到达 “北京”、或“上海”，或“广州”（即终点站）的列车名，要求where子句中除了连接条件只能有一个条件表达式；

14）查询“2020-01-22”从“武汉站”出发，然后当天换乘另一趟车的乘客身份证号和首乘车次号，结果按照首乘车次号降序排列，同车次则按照乘客身份证号升序排列；

15）查询所有新冠患者的身份证号，姓名及其2020年以来所乘坐过的列车名、发车日期，要求即使该患者未乘坐过任何列车也要列出来；

16）查询所有发病日期相同而且确诊日期相同的病患统计信息，包括：发病日期、确诊日期和患者人数，结果按照发病日期降序排列的前提下再按照确诊日期降序排列。

### 3.1.4 了解系统的查询性能分析功能（选做）

选择上述2.3任务中某些较为复杂的SQL语句，查看其执行之前系统给出的分析计划和实际的执行计划，记录观察的结果，并对其进行简单的分析。

### 3.1.5 DBMS函数及存储过程和事务（选做）

1）编写一个依据乘客身份证号计算其在指定年乘列车的乘车次数的自定义函数，并利用其查询2020年至少乘车过3次的乘客。

2）尝试编写DBMS的存储过程，建立每趟列车的乘坐人数的统计表，并通过存储过程更新该表。

3）尝试在DBMS的交互式界面中验证事务机制的执行效果。

## 3.2 完成过程

### 3.2.1 建表

先创建一个叫lab2 的数据库，然后再在这个数据库中创建所有的表格,重要字段表述在代码中用注释的方式写出。

1. 车站表

DROP TABLE IF EXISTS `station`; -- 主键 SID

CREATE TABLE `station` (

`SID` int(11) NOT NULL,

`SName` char(20) NOT NULL,

`CityName` char(20) NOT NULL,

PRIMARY KEY (`SID`) USING BTREE

);

（2）车次表

DROP TABLE IF EXISTS `train`; -- 主键 TID 外键 SStationID—— `station`.(`SID`)、AStationID——`station`.(`SID`)

CREATE TABLE `train` (

`TID` int(11) NOT NULL,

`SDate` date NOT NULL,

`TName` char(20) NOT NULL,

`SStationID` int(11) NOT NULL,

`AStationID` int(11) NOT NULL,

`STime` datetime(0) NOT NULL,

`ATime` datetime(0) NOT NULL,

PRIMARY KEY (`TID`) USING BTREE,

UNIQUE INDEX `TName\_UNIQUE`(`TName`, `SDate`) USING BTREE,

INDEX `SStationID\_idx`(`SStationID`) USING BTREE,

INDEX `AStationID\_idx`(`AStationID`) USING BTREE,

CONSTRAINT `AStationID` FOREIGN KEY (`AStationID`) REFERENCES `station` (`SID`) ON DELETE RESTRICT ON UPDATE RESTRICT,

CONSTRAINT `SStationID` FOREIGN KEY (`SStationID`) REFERENCES `station` (`SID`) ON DELETE RESTRICT ON UPDATE RESTRICT

);

（3）车程表

DROP TABLE IF EXISTS `trainpass`; -- 主键 (TID、SNo) 外键 SID——`station`.(`SID`)

CREATE TABLE `trainpass` (

`TID` int(11) NOT NULL,

`SNo` smallint(6) NOT NULL,

`SID` int(11) NOT NULL,

`STime` datetime(0) NULL DEFAULT NULL,

`ATime` datetime(0) NULL DEFAULT NULL,

PRIMARY KEY (`TID`, `SNo`) USING BTREE,

INDEX `SID\_idx`(`SID`) USING BTREE,

INDEX `TID`(`TID`) USING BTREE,

CONSTRAINT `SID` FOREIGN KEY (`SID`) REFERENCES `station` (`SID`) ON DELETE RESTRICT ON UPDATE RESTRICT

);

（4）乘客表

DROP TABLE IF EXISTS `passenger`; -- 主键 PCardID

CREATE TABLE `passenger` (

`PCardID` char(18) NOT NULL,

`PName` char(20) NOT NULL,

`Sex` int(11) NOT NULL,

`age` smallint(6) NOT NULL,

PRIMARY KEY (`PCardID`) USING BTREE

);

（5）乘车记录表

DROP TABLE IF EXISTS `taketrainrecord`; -- 主键 RID 外键PCardID——`passenger`.(`PCardID`)、SStationID1——`trainpass`.(`SID`)、AStationID1——`trainpass`.(`SID`)、TID——`trainpass`.(`TID`)

CREATE TABLE `taketrainrecord` (

`RID` int(11) NOT NULL,

`PCardID` char(18) NOT NULL,

`TID` int(11) NOT NULL,

`SStationID` int(11) NOT NULL,

`AStationID` int(11) NOT NULL,

`CarrigeID` smallint(6) NOT NULL,

`SeatRow` smallint(6) NOT NULL,

`SeatNo` char(1) NOT NULL,

`SStatus` int(11) NOT NULL,

PRIMARY KEY (`RID`) USING BTREE,

INDEX `PCardID\_idx`(`PCardID`) USING BTREE,

INDEX `TID\_idx`(`TID`) USING BTREE,

INDEX `AStationID1`(`AStationID`) USING BTREE,

INDEX `SStationID1`(`SStationID`) USING BTREE,

CONSTRAINT `AStationID1` FOREIGN KEY (`AStationID`) REFERENCES `trainpass` (`SID`) ON DELETE RESTRICT ON UPDATE RESTRICT,

CONSTRAINT `PCardID` FOREIGN KEY (`PCardID`) REFERENCES `passenger` (`PCardID`) ON DELETE RESTRICT ON UPDATE RESTRICT,

CONSTRAINT `SStationID1` FOREIGN KEY (`SStationID`) REFERENCES `trainpass` (`SID`) ON DELETE RESTRICT ON UPDATE RESTRICT,

CONSTRAINT `TID` FOREIGN KEY (`TID`) REFERENCES `trainpass` (`TID`) ON DELETE RESTRICT ON UPDATE RESTRICT

);

（6）诊断表

DROP TABLE IF EXISTS `diagnoserecord`; -- 主键 DID

CREATE TABLE `diagnoserecord` (

`DID` int(11) NOT NULL,

`PCardID` char(18) NOT NULL,

`DDay` date NOT NULL,

`DStatus` smallint(6) NOT NULL,

`FDay` date NOT NULL,

PRIMARY KEY (`DID`) USING BTREE

);

（7）乘客紧密接触者表

DROP TABLE IF EXISTS `traincontactor`; -- 主键 (CDate、CCardID、DStatus、PCardID)

CREATE TABLE `traincontactor` (

`CDate` date NOT NULL,

`CCardID` char(18) NOT NULL,

`DStatus` smallint(6) NOT NULL,

`PCardID` char(18) NOT NULL,

PRIMARY KEY (`CDate`, `CCardID`, `DStatus`, `PCardID`) USING BTREE

);

10）观察性实验：外码是否一定要参照被参照关系的主码。

创建一个表TEST，设置其中一个外码NAME参照的不是USER表的主码。

参照非主码报错，由报错信息可知：报错的原因是，TEST表中的外码NAME所参照的USER表的NAME在USER表中不是主码或者候选码。

综上所述：外码必须要参照主码或者候选码才可以。

11）数据准备

部分数据可以使用实验时老师提供的数据，通过简单操作后直接将其导入到所建好的表中。具体操作步骤如下：

选中数据库，右击选择任务，再选择导入数据，选择导入数据源你和目的，即可将给出的数据导入到数据库。

没有提供的数据则可以使用python编写代码根据上面提供的数据为约束生成其他表的数据信息，然后同上述导入数据的方法。

### 3.2.2 数据更新

（1）分别用一条sql语句完成对乘车记录表基本的增、删、改的操作；

-- ```mysql

INSERT INTO TABLE A (FIELD 1, FIELD 2) VALUES (FIELD 1, FIELD 2);

DELETE FROM TABLE A [WHERE Clause];

UPDATE TABLE A SET FIELD 1 = new-value1, FIELD 2 = new-value2 [WHERE Clause];

-- ```

（2）批处理操作

-- 将乘车记录表中的从武汉出发的乘客的乘车记录插入到一个新表WH\_TakeTrainRecord中。

-- ```mysql

-- //表不存在时 创建表并插入查询数据

CREATE TABLE WH\_TakeTrainRecord AS SELECT \* FROM taketrainrecord WHERE SStationID IN (SELECT SID FROM station WHERE CityName = "武汉");

-- ```

-- ```mysql

-- //表已存在

INSERT INTO WH\_TakeTrainRecord SELECT \* FROM taketrainrecord WHERE SStationID IN (SELECT SID FROM station WHERE CityName = "武汉");

-- ```

（3）数据导入导出

-- 通过查阅DBMS资料学习数据导入导出功能，并将任务2.1所建表格的数据导出到操作系统文件，然后再将这些文件的数据导入到相应空表。

（4）观察性实验

-- 建立一个关系，但是不设置主码，然后向该关系中插入重复元组，然后观察在图形化交互界面中对已有数据进行删除和修改时所发生的现象。

-- > 当不设主码时，表中可以存在数据项相同的数据存在，例表WH\_TakeTrainRecord设置为无主码，插入两条RID为1的数据，当改变RID=1的数据时，两条数据都会被改变，删除同；

（5）创建视图

-- 创建一个新冠确诊病人的乘火车记录视图，其中的属性包括：身份证号、姓名、年龄、乘坐列车编号、发车日期、车厢号，席位排号，席位编号。按身份证号升序排序，如果身份证号一致，按发车日期降序排序（注意，如果病人买了票但是没坐车，不计入在内）。

-- ```mysql

CREATE VIEW patient\_view as

(SELECT passenger.PName,passenger.PCardID,passenger.age,taketrainrecord.TID,train.STime,CarrigeID,SeatRow,SeatNo

FROM passenger,diagnoserecord,taketrainrecord,train

WHERE passenger.PCardID = diagnoserecord.PCardID AND diagnoserecord.DStatus = 1

AND passenger.PCardID = taketrainrecord.PCardID AND taketrainrecord.TID = train.TID

AND taketrainrecord.SStatus != 2 ORDER BY PCardID ,STime DESC);

-- ```

（6）触发器实验

-- 编写一个触发器，用于实现以下完整性控制规则：

-- 1) 当新增一个确诊患者时，若该患者在发病前14天内有乘车记录，则将其同排及前后排乘客自动加入“乘客紧密接触者表”，其中：接触日期为乘车日期。

CREATE TRIGGER `Insert\_Update` AFTER UPDATE ON `diagnoserecord` FOR EACH ROW BEGIN

IF(new. DStatus = 1) THEN

INSERT INTO traincontactor

SELECT Train.SDate,TRD1.PCardID,2,new.PCardID -- 更新的数据PCardID

FROM Train,TakeTrainRecord TRD1 -- TRD1是密切接触者

WHERE Train.TID = TRD1.TID

AND EXISTS( -- 乘车约束，TRD1的患者14天内乘过车，和该患者在同一车厢，在该患者前排后排，同排的乘客加入密切接触者表

SELECT \* FROM

TakeTrainRecord TRD2 -- 插入诊断表确诊的患者

WHERE new.PCardID = TRD2.PCardID

AND TRD1.TID = TRD2.TID -- 找到该患者

AND TRD2.CarrigeID = TRD1.CarrigeID

AND TRD1.SeatRow IN (TRD2.SeatRow,TRD2.SeatRow + 1 ,TRD2.SeatRow - 1)

AND DATEDIFF(new.FDay,Train.SDate) <= 14 -- 对比日期

AND DATEDIFF(new.FDay,Train.SDate) >= 0

AND TRD1.SStatus != 2 AND TRD2.SStatus != 2

AND TRD1.PCardID <> new.PCardID -- 不是自身

);

END IF;

END;

-- 2) 当一个紧密接触者被确诊为新冠时，从“乘客紧密接触者表”中修改他的状态为“1”。

CREATE TRIGGER `Syn`

ON `diagnoserecord` FOR EACH ROW

AFTER UPDATE

BEGIN

UPDATE `traincontactor` SET DStatus = new.DStatus;

END

### 3.2.3 查询

(1)查询确诊者“李玉兰”在发病前14天内的乘车记录，如下图 3‑1

联合表passager、taketrainrecord、train、diagnoserecord查询姓名为李玉兰，在发病前14天的乘车记录，根据需要的条件写入where语句中进行筛选。

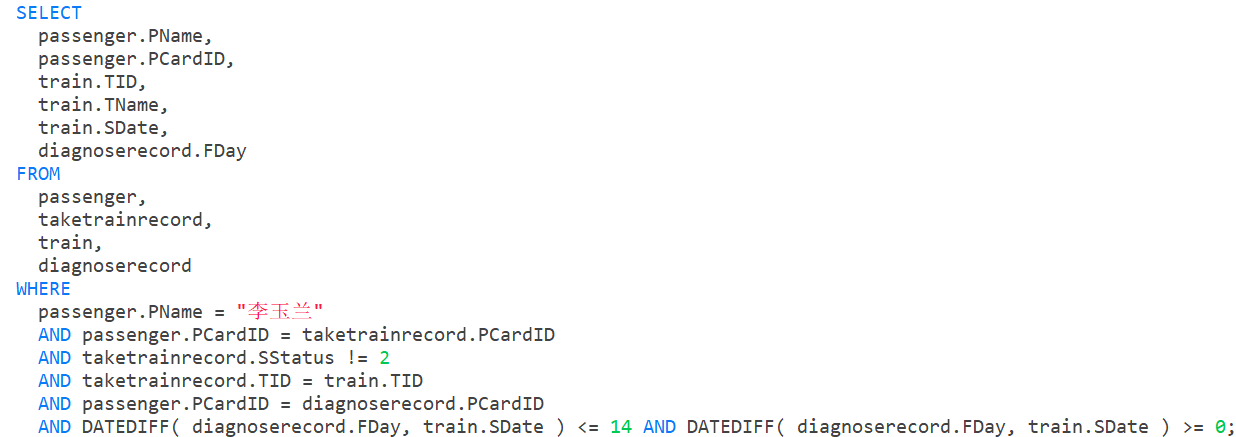


图 3‑1

查询结果，图 3‑2：

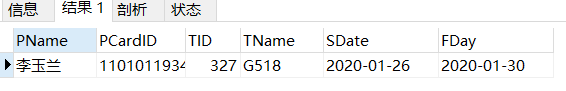


图 3‑2

(2)查询所有从城市“武汉”出发的乘客乘列车所到达的城市名，图 3‑3。

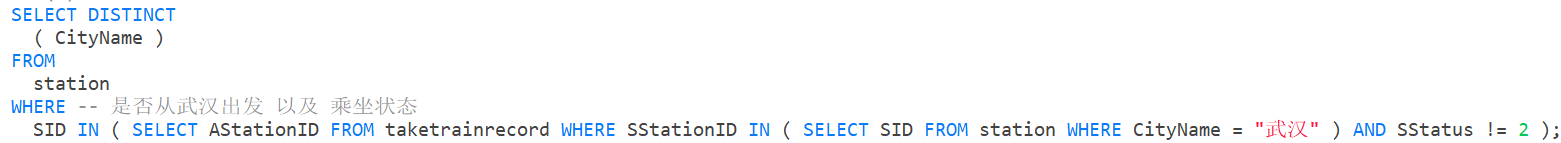


图 3‑3

查询结果，图 3‑4：



图 3‑4

(3)计算每位新冠患者从发病到确诊的时间间隔（天数）及患者身份信息，并将结果按照发病时间天数的降序排列，图 3‑5

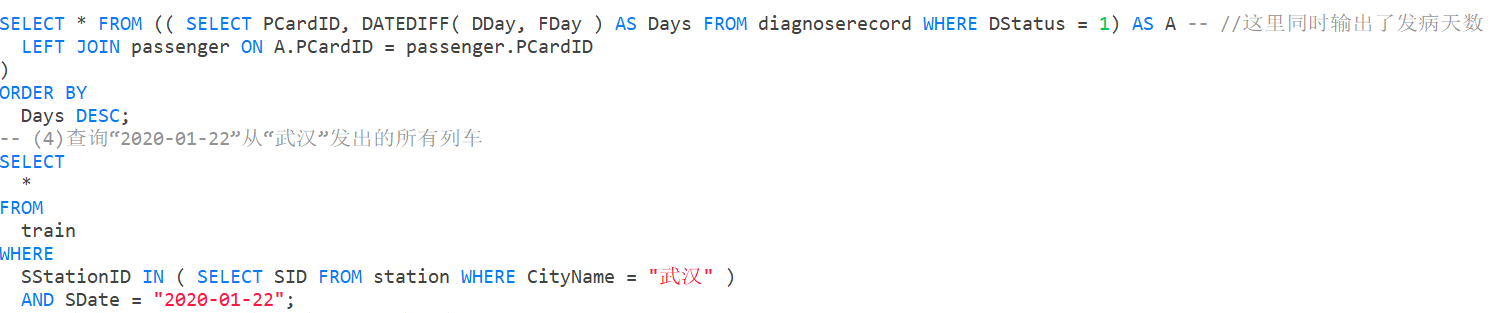


图 3‑5

查询结果，图 3‑6：（8178条）

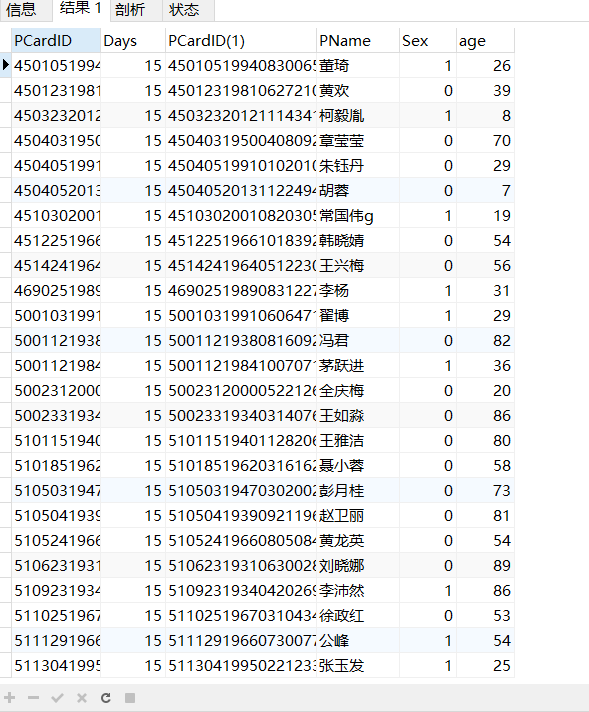


图 3‑6

(4)查询“2020-01-22”从“武汉”发出的所有列车，图 3‑7

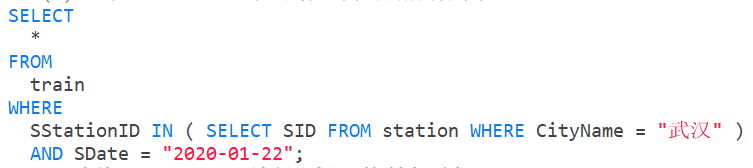


图 3‑7

查询结果，图 3‑8：

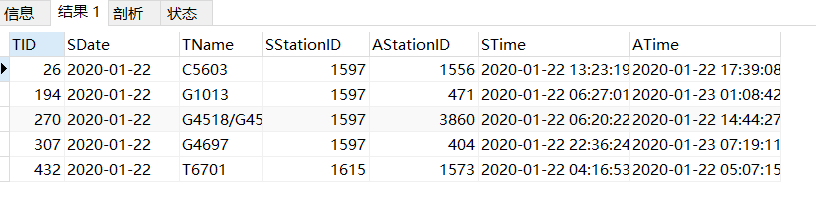


图 3‑8

(5)查询“2020-01-22”途经“武汉”的所有列车，图 3‑9



图 3‑9

查询结果，图 3‑10：



图 3‑10

(6)查询“2020-01-22”从武汉离开的所有乘客的身份证号、所到达的城市、到达日期，图 3‑11

从taketrainrecord查询出"2020-01-22"从武汉站出发的列车以及乘客身份证，然后根据查询到的乘车记录表中的到达站编号查询trainpass中到达站的时间，从station中查询到列车名。输出身份证、到达城市名、到达时间

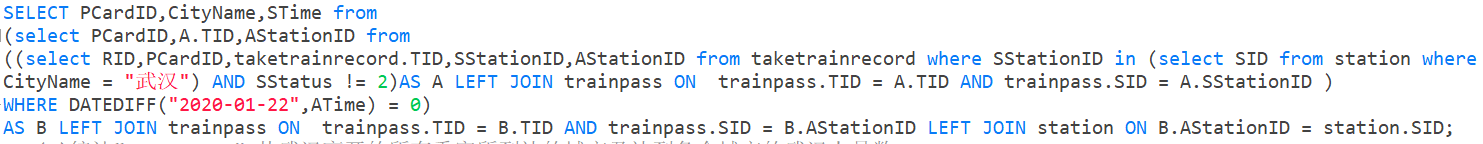


图 3‑11

查询结果：（209条）

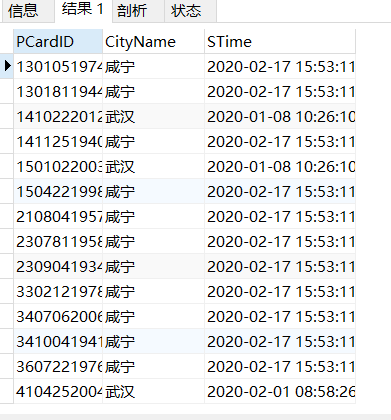


图 3‑12

(7)统计“2020-01-22” 从武汉离开的所有乘客所到达的城市及达到各个城市的武汉人员数图 3‑13

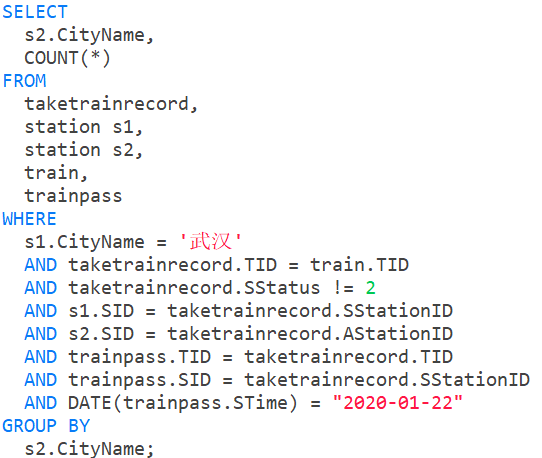


图 3‑13

查询结果图 3‑14：



图 3‑14

(8)查询2020年1月到达武汉的所有人员图 3‑15

从taketrainrecord中查询到达站为武汉车站的乘车记录，再连接trainpass中查询到达站点日期在一月份的记录，根据已经查找到的信息，从passager中查出人员信息

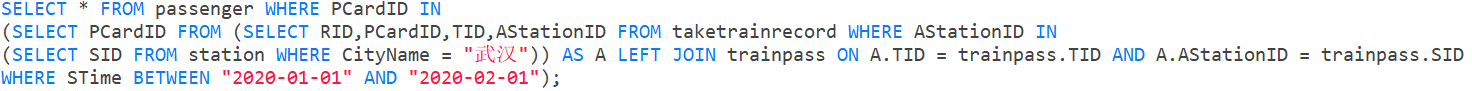


图 3‑15

查询结果图 3‑16：（809条）



图 3‑16

(9)查询2020年1月乘车途径武汉的外地人员（身份证非“420”开头）图 3‑17

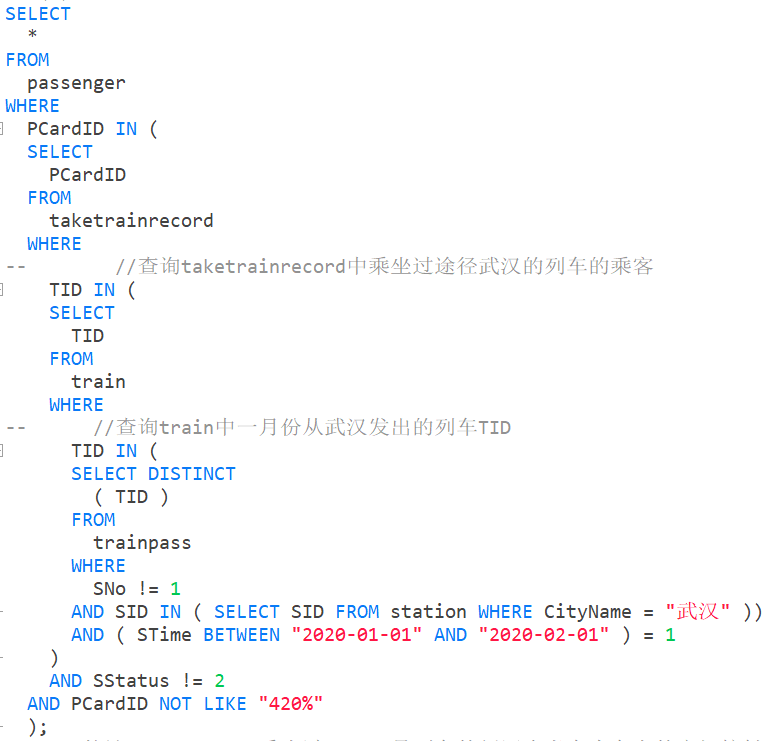


图 3‑17

查询结果图 3‑18：（2020条）



图 3‑18

(10)统计“2020-01-22”乘坐过‘G007’号列车的新冠患者在火车上的密切接触乘客人数（每位新冠患者的同车厢人员都算同车密切接触）图 3‑19

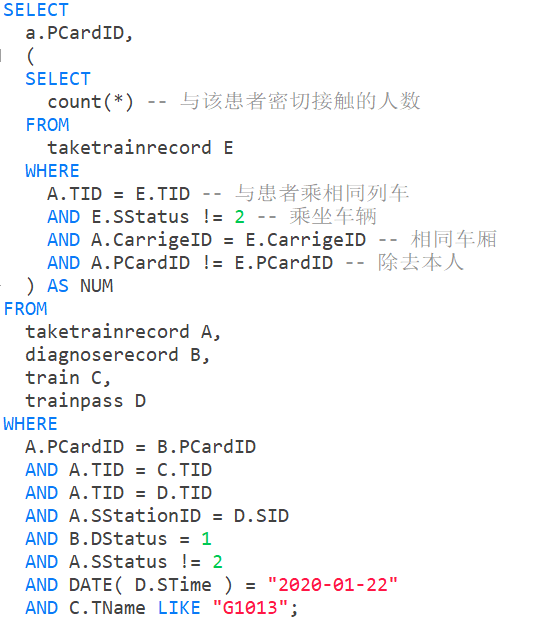


图 3‑19

查询结果图 3‑20：

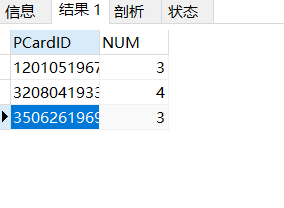


图 3‑20

(11)查询一趟列车的一节车厢中有3人及以上乘客被确认患上新冠的列车名、出发日期，车厢号图 3‑21

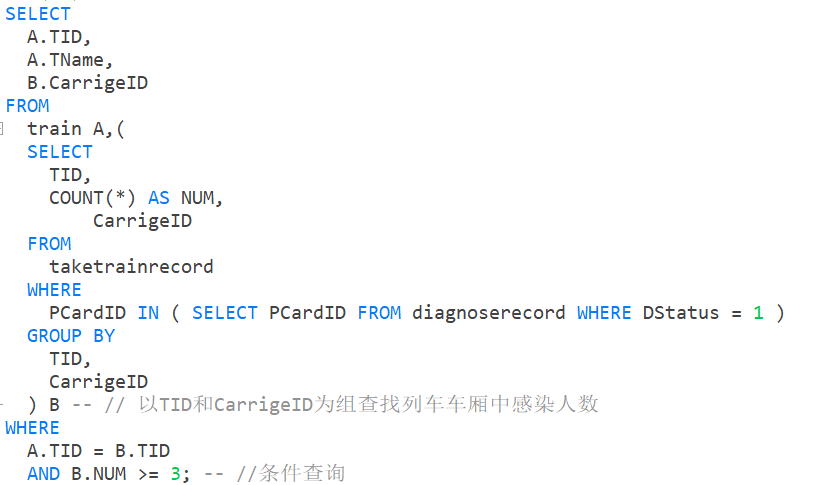


图 3‑21

查询结果图 3‑22：（538条）



图 3‑22

(12)查询没有感染任何周边乘客的新冠乘客的身份证号、姓名、乘车日期图 3‑23

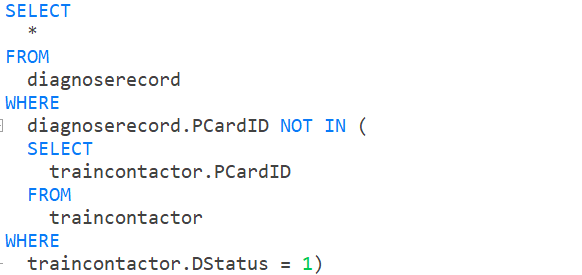


图 3‑23

查询结果图 3‑24：（24367条）

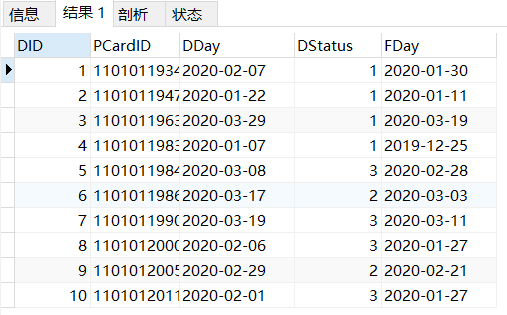


图 3‑24

(13)查询到达 “北京”、或“上海”，或“广州”（即终点站）的列车名，要求where子句中除了连接条件只能有一个条件表达式图 3‑25



图 3‑25

查询结果图 3‑26：（43条）

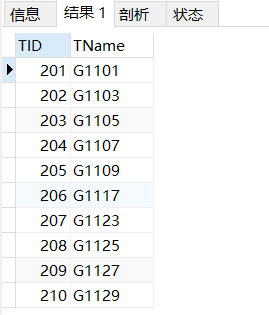


图 3‑26

(14)查询“2020-01-22”从“武汉站”出发，然后当天换乘另一趟车的乘客身份证号和首乘车次号，结果按照首乘车次号降序排列，同车次则按照乘客身份证号升序排列图 3‑27

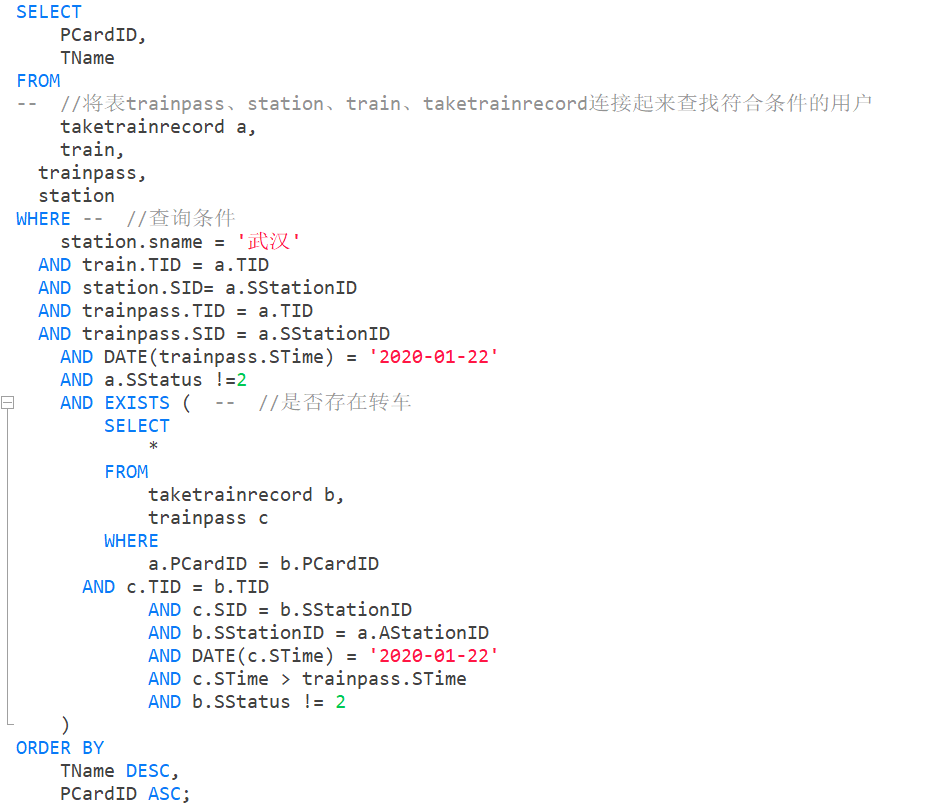


图 3‑27

查询结果图 3‑28：

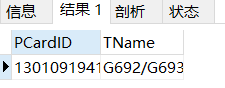


图 3‑28

(15)查询所有新冠患者的身份证号，姓名及其2020年以来所乘坐过的列车名、发车日期，要求即使该患者未乘坐过任何列车也要列出来图 3‑29



图 3‑29

查询结果图 3‑30：（8178条）



图 3‑30

(16)查询所有发病日期相同而且确诊日期相同的病患统计信息，包括：发病日期、确诊日期和患者人数，结果按照发病日期降序排列的前提下再按照确诊日期降序排列图 3‑31

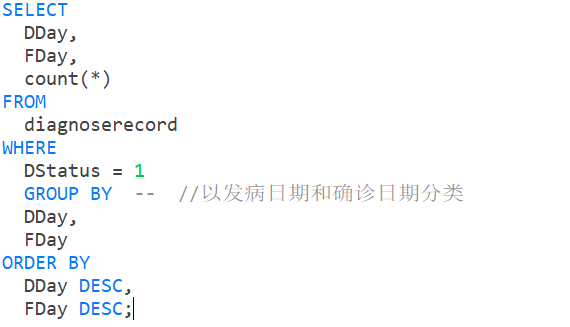


图 3‑31

查询结果图 3‑32：（1887条）

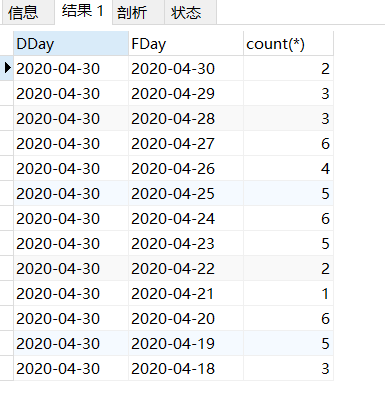


图 3‑32

## 3.3任务总结

(1)利用SQL语言以及图形化的功能创建了数据库表，并往表中导入了数据。

(2)验证了关系的外码一定要参照另外一个关系的主码。

(3)使用SQL语句对表的内容进行了基本的增删改，并通过图形化界面直观的看到这些操作的效果。

(4)实现了批量插入数据的操作。

(5)体验了主码在一个关系中的重要性，测试了没有定义主码的关系，在增删改时的执行情况。

(6)通过定义触发器，来加深对触发器工作机制的理解。

(7)学会了视图的创建

(8)通过实现多种不同的查询的需求，运用了多种查询表达式，加深对查询表达式的熟悉和理解，

# 综合实践任务

## 4.1系统设计目标

随着因特网在全球的迅猛发展，电子商务越来越受到人们的关注，开始渗透到人们生活的各个领域，改变这经济社会的各个方面。电子商务企业的迅速崛起，生产企业、零售企业、金融服务业纷纷开展电子商务。电子商务是一种崭新的商业模式。企业利用前沿信息技术，将传统受制于时空的商务活动转变为跨越时空的虚拟交易，是传统商务经济活动的电子化与信息化。

而随着人们的生活水平的提高，对物质要求也越来越高，网购的消费也逐年增长，网上销售为人们的网上购物提供了一个平台。

## 4.2 需求分析

### 4.2.1功能需求

网上销售系统要求提供包括商品信息管理、查询、订购、销售等功能的网上交易平台，对客户和商店管理员应提供不同的操作界面和使用权限。具体功能包括：

商品信息管理：商店可以对商品信息进行管理，包括商品的类别、名称、描述信息、售价、图片、折扣等。

客户信息管理：客户可以自助注册并管理自己的个人信息。

商品查询：客户可以通过多种方式查询并且挑选网上商店出售的商品，通过在网上填写并确认订单的方式来购买商品。

订单查询：客户可以管理自己的订单信息，查询订单的处理情况。

订单处理：商店可以对客户的订单信息、汇款单信息进行审核，以确定是否发货，并修改订单状态。

### 4.2.2性能需求

用户在使用网站进行操作的过程中响应速度要快、结果精度要高以及运行时资源消耗量要小。

### 4.2.3数据完整性需求

（1）域完整性：一个列的输入有效性，是否允许为空值，对于表中不是重要属性的列不需要设置为not null，允许其空值的存在。

（2）实体完整性：保证表中所有的行唯一。

（3）参照完整性：是保证主关键字（被引用表）和外部关键字（引用表）之间的参照关系，在设计数据库表的时候需要考虑表中数据是否需要设置与其他表之间的联系通过外键来进行联系。

### 4.2.4数据流图

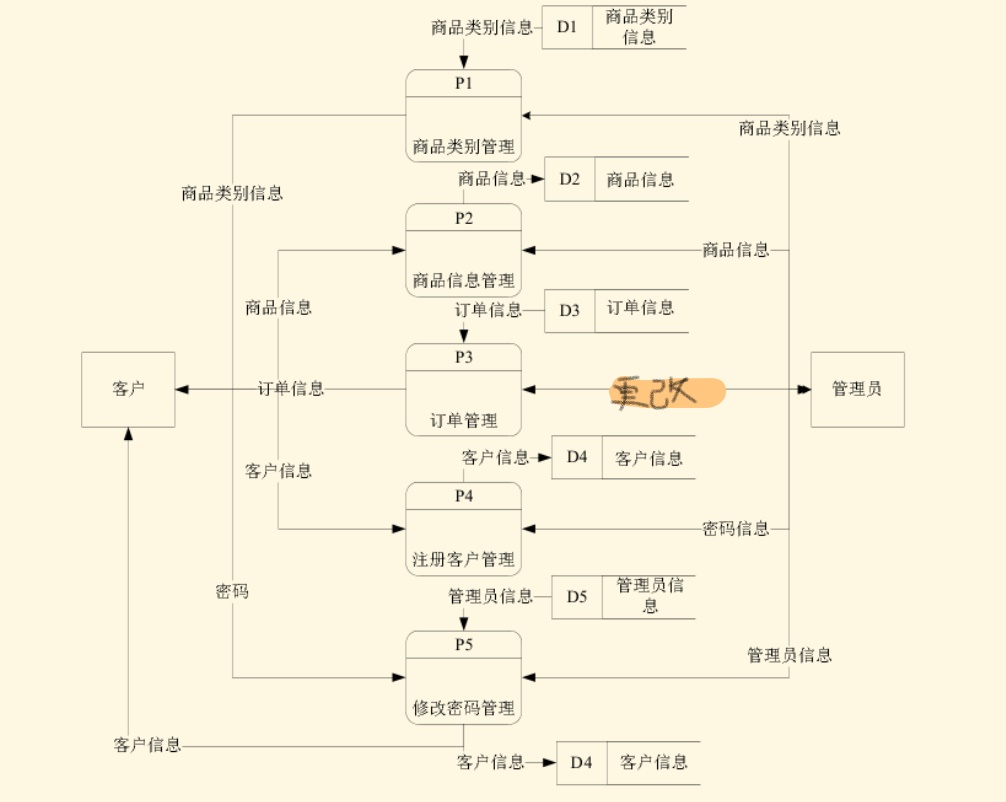


图 4‑1

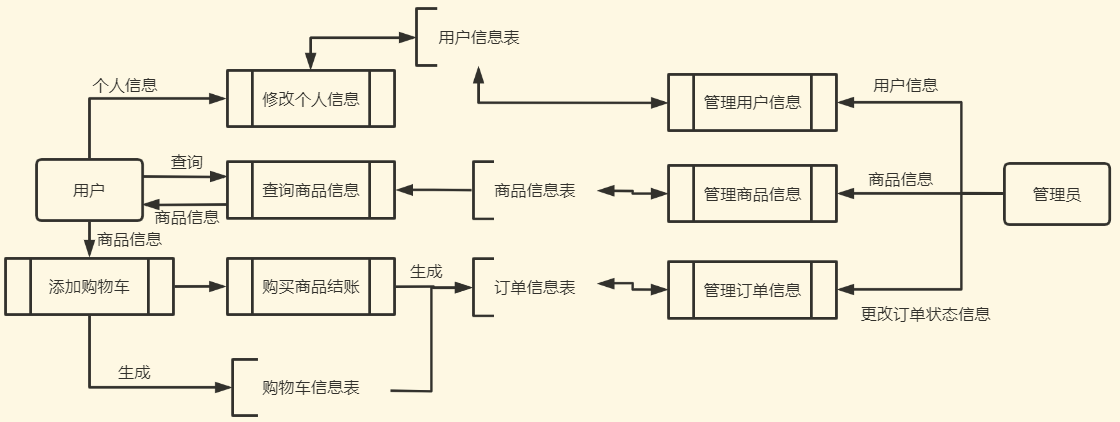


图 4‑2

### 4.2.5数据字典

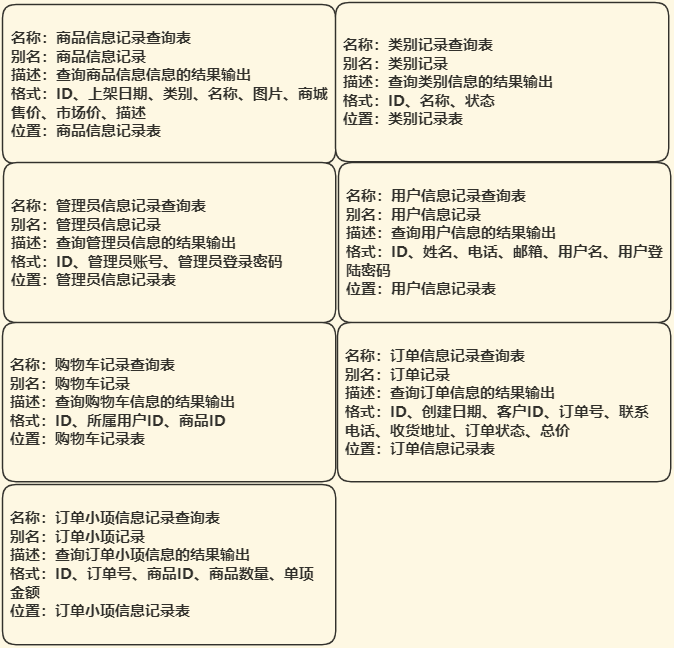


图 4‑3

## 4.3 总体设计

### 4.3.1系统B/S架构图

后端服务器使用技术栈Springboot + JPA + MYSQL + Thymeleaf，目录结构如下：

1.工程启动类(Application.java)：置于com.nhy.demo.mall包下

2.实体类(domain)：置于com.nhy.demo.mall.entity

3.数据访问层(Dao)：置于com.nhy.demo.mall.dao

4.数据服务层(Service)：置于com.nhy.demo.mall.service

5.数据服务接口的实现(serviceImpl)：置于com.nhy.demo.mall.service.impl

6.前端控制器(Controller)：置于com.nhy.demo.mall.controller.(user)/(admin)

7.工具类(utils)：置于com.nhy.demo.mall.utils

8.拦截器(filte)：置于com.nhy.demo.mall.filter（用于在未登陆时，拦截不正确的访问）

前端页面html资源放置在根目录下resource/文件夹下。

简单架构图如下图 4‑4：

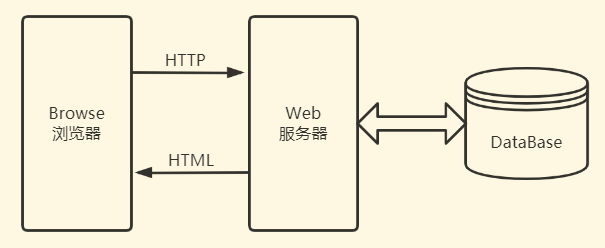


图 4‑4

### 4.3.2功能模块设计

对于管理员和用户有不同的页面供其进行对应操作，管理员的主要功能能够管理用户信息、商品信息以及对订单状态进行更改；而用户则可以对个人信息进行修改以及对商品做出查询、购买等一系列的动作。

将管理员和用户的功能模块设计如下所示图 4‑5。

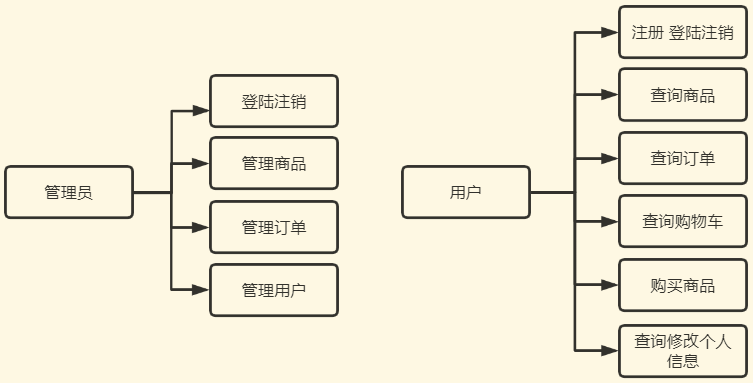


图 4‑5

## 数据库设计

### 4.4.1实体类型设计

| **对象** | **属性** |
| --- | --- |
| 商品 | ID、上架日期、类别、名称、图片、商城售价、市场价、描述 |
| 类别 | ID、名称、状态 |
| 管理员 | ID、管理员账号、管理员登录密码 |
| 用户 | ID、姓名、电话、邮箱、用户名、用户登陆密码、 |
| 购物车 | ID、所属用户ID、商品ID |
| 订单 | ID、创建日期、客户ID、订单号、联系电话、收货地址、订单状态、总价 |
| 订单小项 | ID、订单号、商品ID、商品数量、单项金额 |

（订单小项的设计是因为光凭订单一项无法在数据库表单中存储同一订单中的多个商品，所以再设计一个订单小项用来存储订单中的多个商品 订单号->商品号 == 1->n）

### 4.4.2E-R图

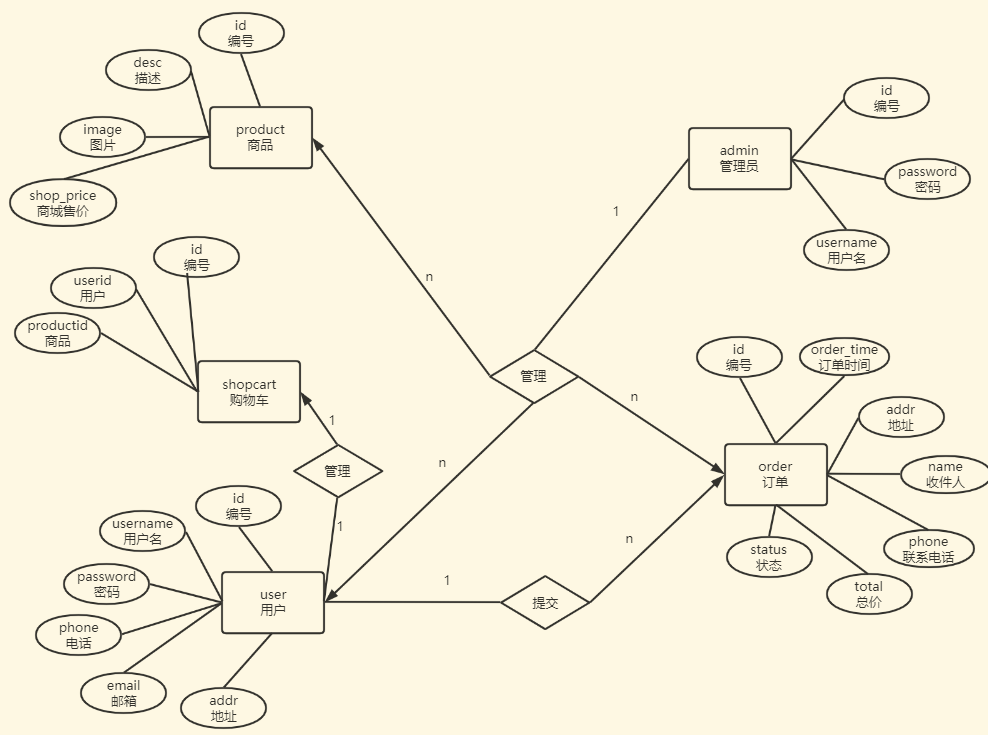


图 4‑6

### 4.4.3数据库逻辑结构设计

**ADMIN( 管理员)**



图 4‑7

**USER（用户）**



图 4‑8

**CLASSIFICATION（类别）**

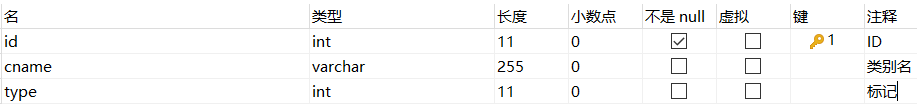


图 4‑9

**PRODUCT（商品）**



图 4‑10

外键设置（商品中的类别字段必须和类别表中的记录匹配）：



图 4‑11

**SHOP\_CAR（购物车）**



图 4‑12

外键设置（购物车需要记录的是用户ID以及商品ID，所以需要对其进行约束）：



图 4‑13

**ORDER（订单）**

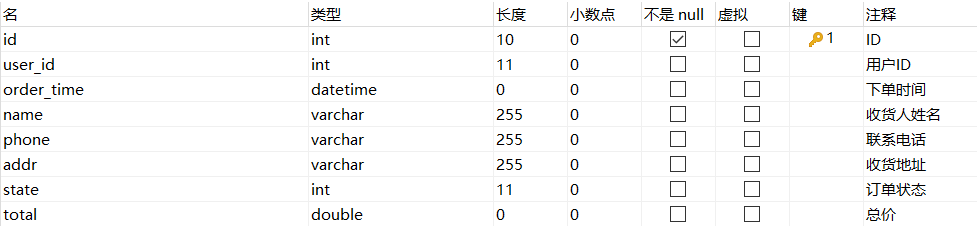


图 4‑14

外键设置（订单的拥有者是用户，对用户ID进行约束）：



图 4‑15

**ORDER\_ITEM（订单小项）**



外键设置（订单小项用于订单与商品的一对多映射，对订单号以及商品ID进行约束）：

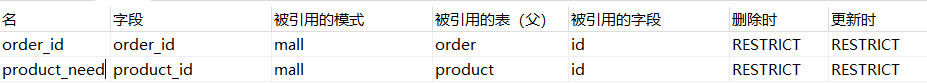


图 4‑16

阐述数据库设计的主要内容，依次是ER图设计及其说明、数据库逻辑结构设计（包括表、视图的逻辑结构说明，含主码、外码说明、用户定义完整性）、数据库物理设计（例如索引）。

## 4.5 详细设计与实现

阐述各主干功能的实现过程，包括主干功能的业务流程图、关键技术和算法说明、数据库事务的定义与实现、数据库函数和触发器的定义与实现等（不允许大段引用源码，如有必要引用必须加详细注释）。

### 4.5.1管理员模块

**登陆注销操作(数据库表admin)**

1. 登陆：前端页面发起登陆请求并传递参数：账号名、密码，后端接收参数，调用方法接口查询admin表中是否存在账号名和密码相同的记录，有则登陆成功，跳转登录成功页面，反之，登陆失败，跳转登陆失败页面，流程图如下图 4‑17。

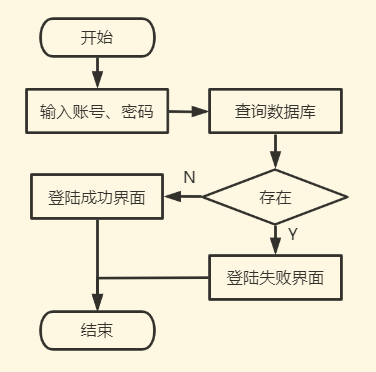


图 4‑17

**管理商品操作（数据库表product）**

1、添加商品：前端页面发起添加商品请求并传递商品所需参数（商品属性有部分可以为null），后端接收到请求和参数后，调用接口向数据库中插入新的商品记录。

2、修改商品：前端页面发起修改商品请求并传递修改商品ID及修改后的信息，后端接收到请求和参数后，调用接口向数据库中更新商品信息记录。

3、删除商品：前端页面发起删除商品请求并传递删除商品ID，后端接收到请求和参数后，调用接口向数据库中删除商品信息记录。

4、展示商品：前端页面发起展示商品请求，后端接收到请求和参数后，调用接口向数据库中查询商品信息记录，前端将后端返回的数据记录渲染到页面上。

**管理订单操作(数据库表order、order\_item)**

1、查询订单信息：首先获取当前登陆用户的用户ID，根据用户ID在表order中进行查询，返回找到的记录并渲染到前端页面。

2、查询订单购买商品：获取当前需要查询的订单号，根据订单号在表 orader\_item中查询记录，将找到的记录返回。

3、修改订单状态：在表order中更新当前更改订单的状态字段。

**管理用户操作（数据库表user）**

1、修改用户信息：更新表user中修改的用户信息。

2、删除用户：在表user中删除选中的用户（根据ID进行查找删除）。

3、展示所有用户：在表user中查询所有用户信息并输出。

### 4.5.2用户模块

**注册、登陆注销操作（数据库表user）**

1、注册用户：想表user中添加用户信息

2、登陆：根据前端页面传回的账号名、密码查询表user中是否有存在的记录，如果有则登陆成功，否则就登陆失败，流程图同图 4‑17

**查询商品操作（数据库表product）**

1、根据条件查询商品：可以根据商品名、类别来查询商品。

**查询订单操作（数据库表order）**

1. 查询登陆用户订单：获取当前登陆用户的ID，在表order中查找user\_id为当前用户的订单记录，并将记录渲染到前端页面展示，如下图图 4‑18。

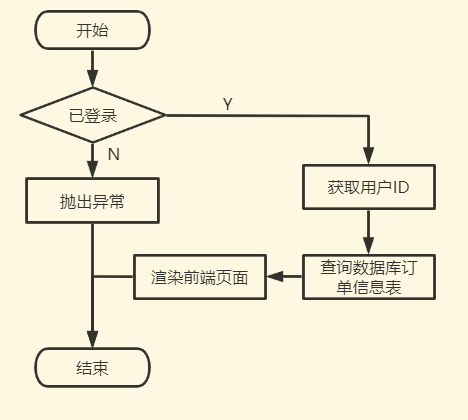


图 4‑18

**查询购物车操作（数据库表shop\_car、product）**

1. 查询登陆用户购物车：获取当前登陆用户的ID，在表shop\_car中查找user\_id为当前用户的购物车商品记录，根据获得的商品信息记录，查询表product，将获取到的商品信息记录渲染到前端页面展示。
2. 删除购物车商品：在表shop\_car中删除指定的商品记录

**购买商品操作（数据库表order、order\_item）**

1. 购买商品：生成订单insert（order），将订单中的商品逐个insert（order\_item），如下流程图图 4‑19。

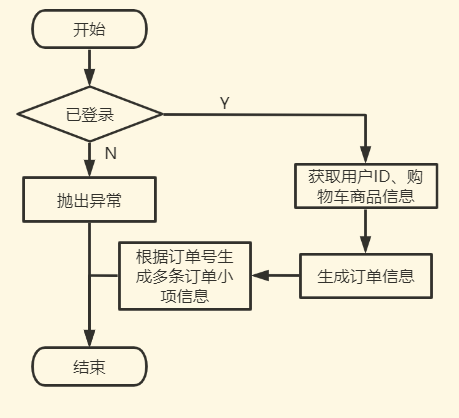


图 4‑19

**查询修改个人信息操作（数据库表user）**

1. 查询个人信息：获取当前登陆用户的ID，在表user中查找user\_id为当前用户的个人信息记录，并将信息渲染到页面上。
2. 修改个人信息：在页面上更新个人信息后，将更新的数据信息更新到表user中。

## 4.6 系统测试

### 4.6.1测试数据说明

添加管理员账号信息



图 4‑20 管理员数据

从网上查找部分商品数据，将其添加进数据库product表中，如下。



图 4‑21 商品数据

### 4.6.1测试

（1）、在网站中输入网址localhost:8080/mall进入首页如下所示，首页展示了部分商品信息。

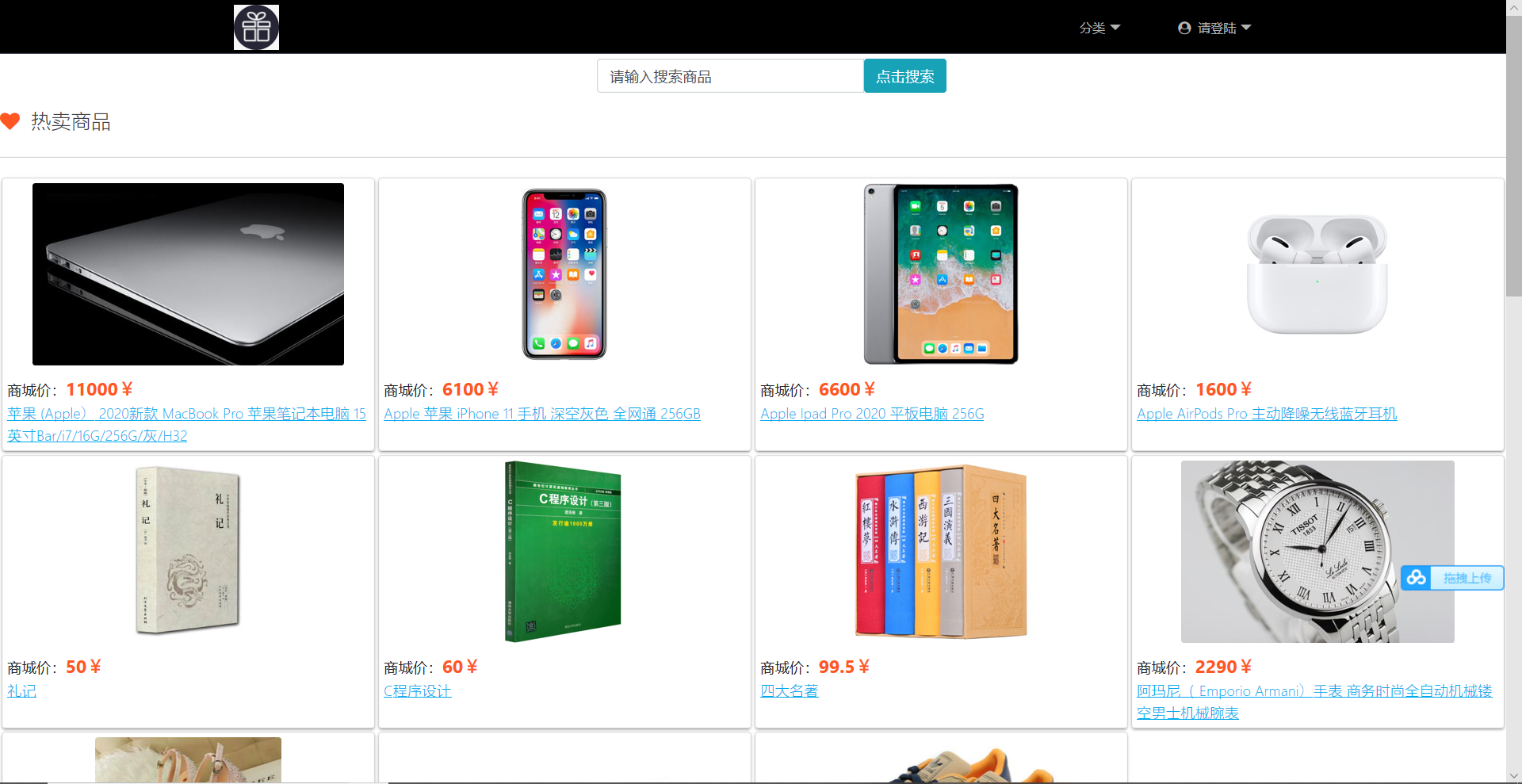


图 4‑22 商城首页（未登陆）

（2）、右上角折叠处可以找到注册用户入口，进入注册页面并填写完注册用户信息后如下图所示。

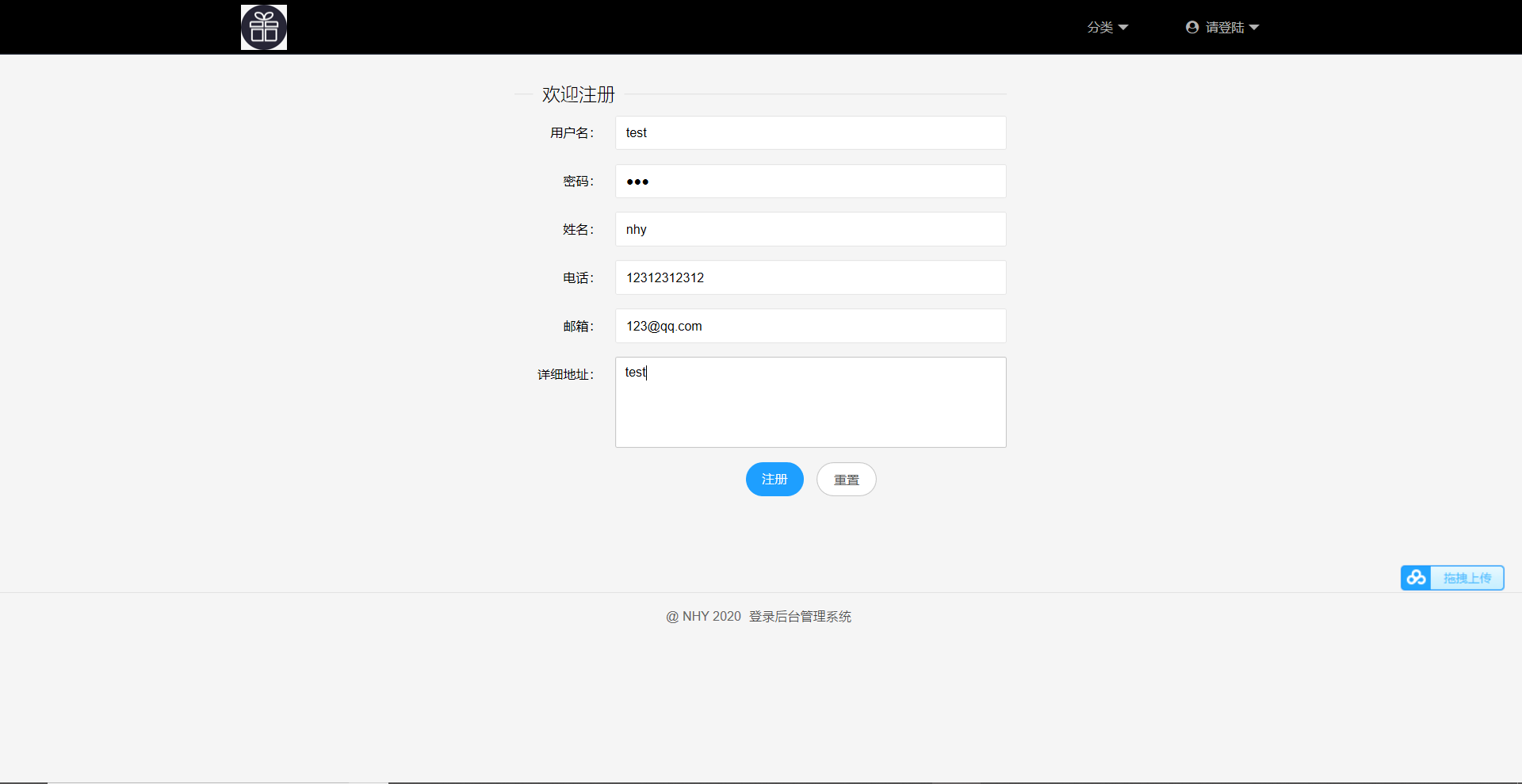


图 4‑23 用户注册

（3）注册完毕后会直接跳转到登陆界面，在登陆界面输入账号名、密码，如下所示。

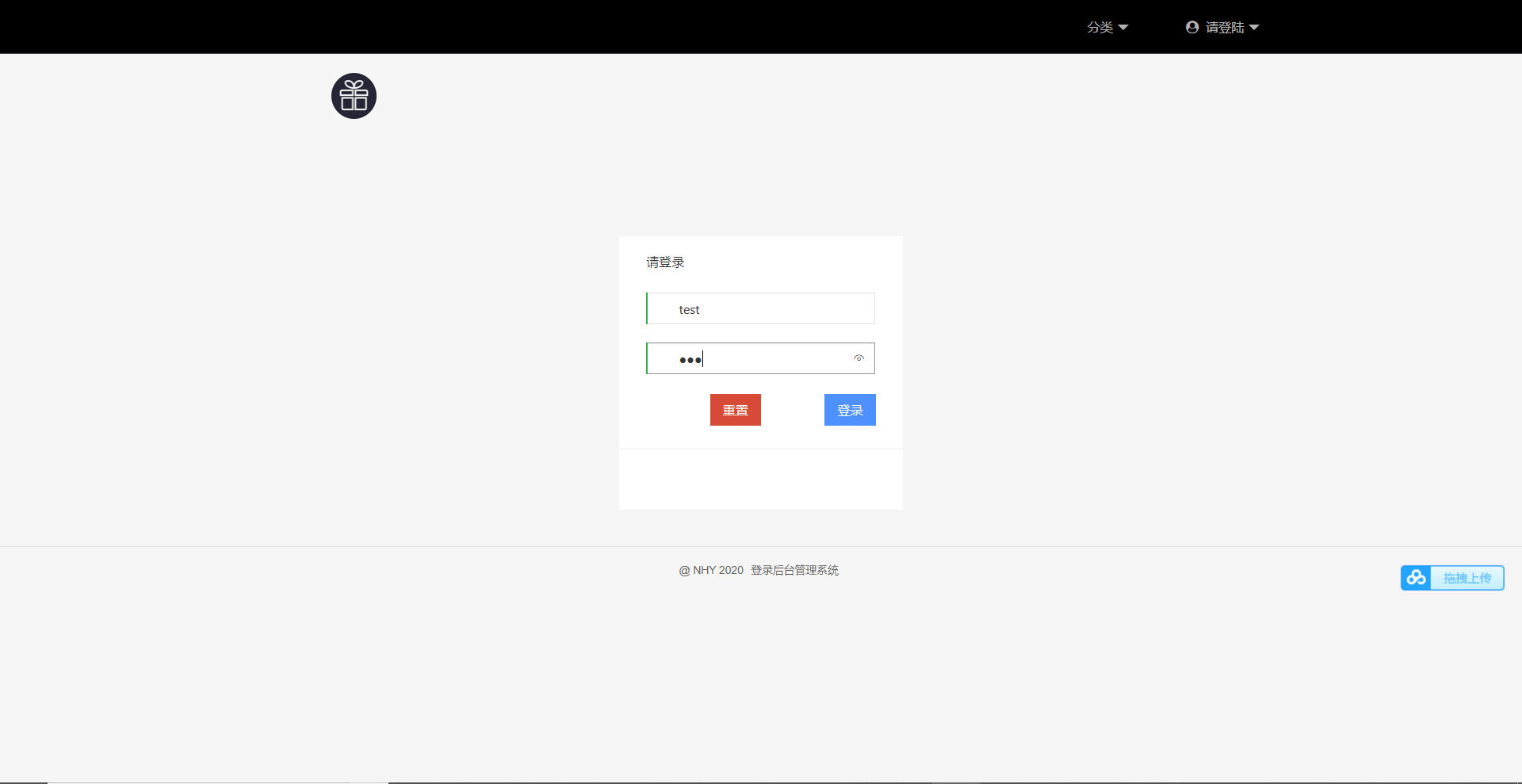


图 4‑24 用户登录

（4）、登陆成功后可以进入商城首页，并可以在右上角看到自己已经登陆的信息，如果账号密码登陆不存在，则登陆失败，会进入404not find页面。

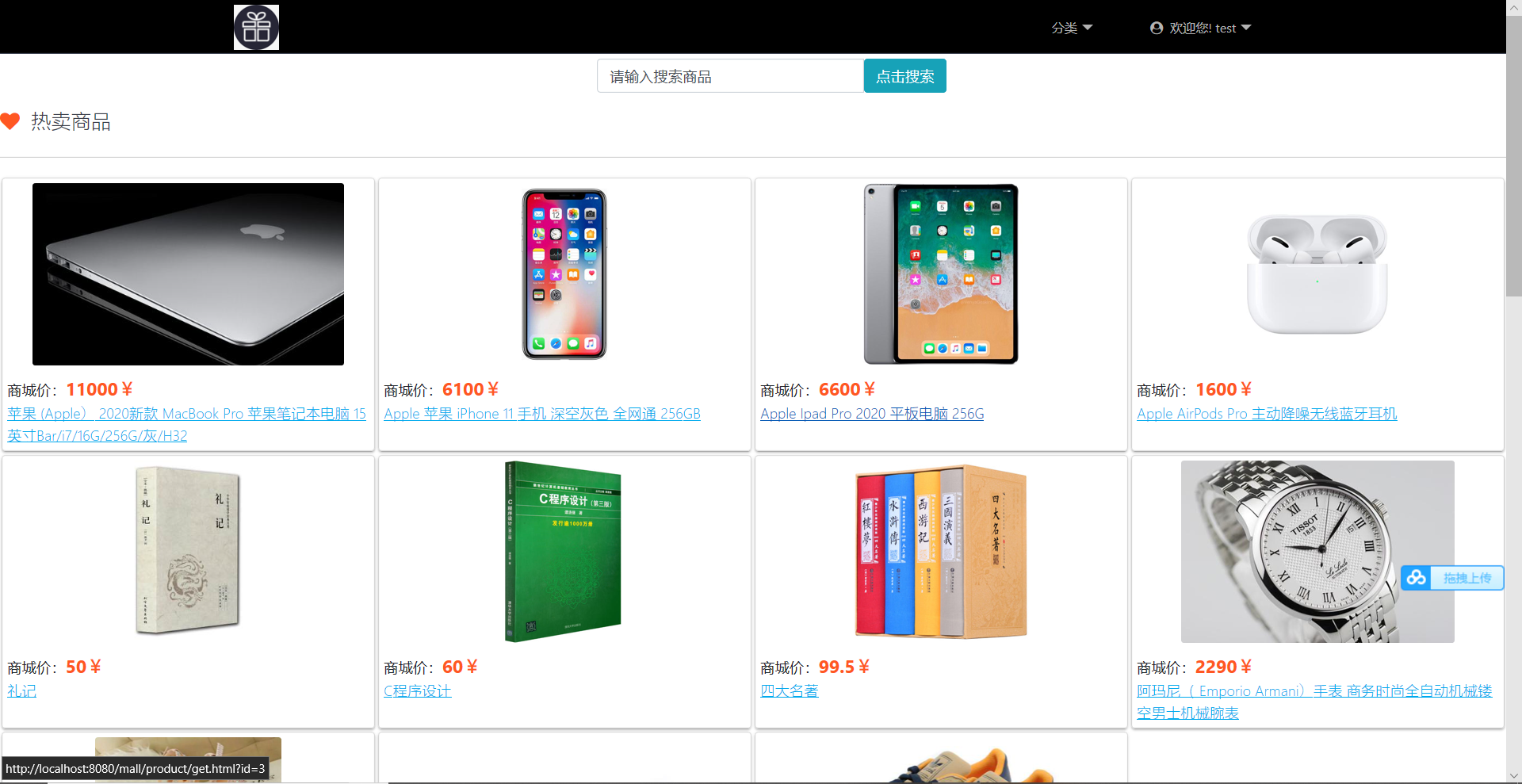


图 4‑25 登陆成功

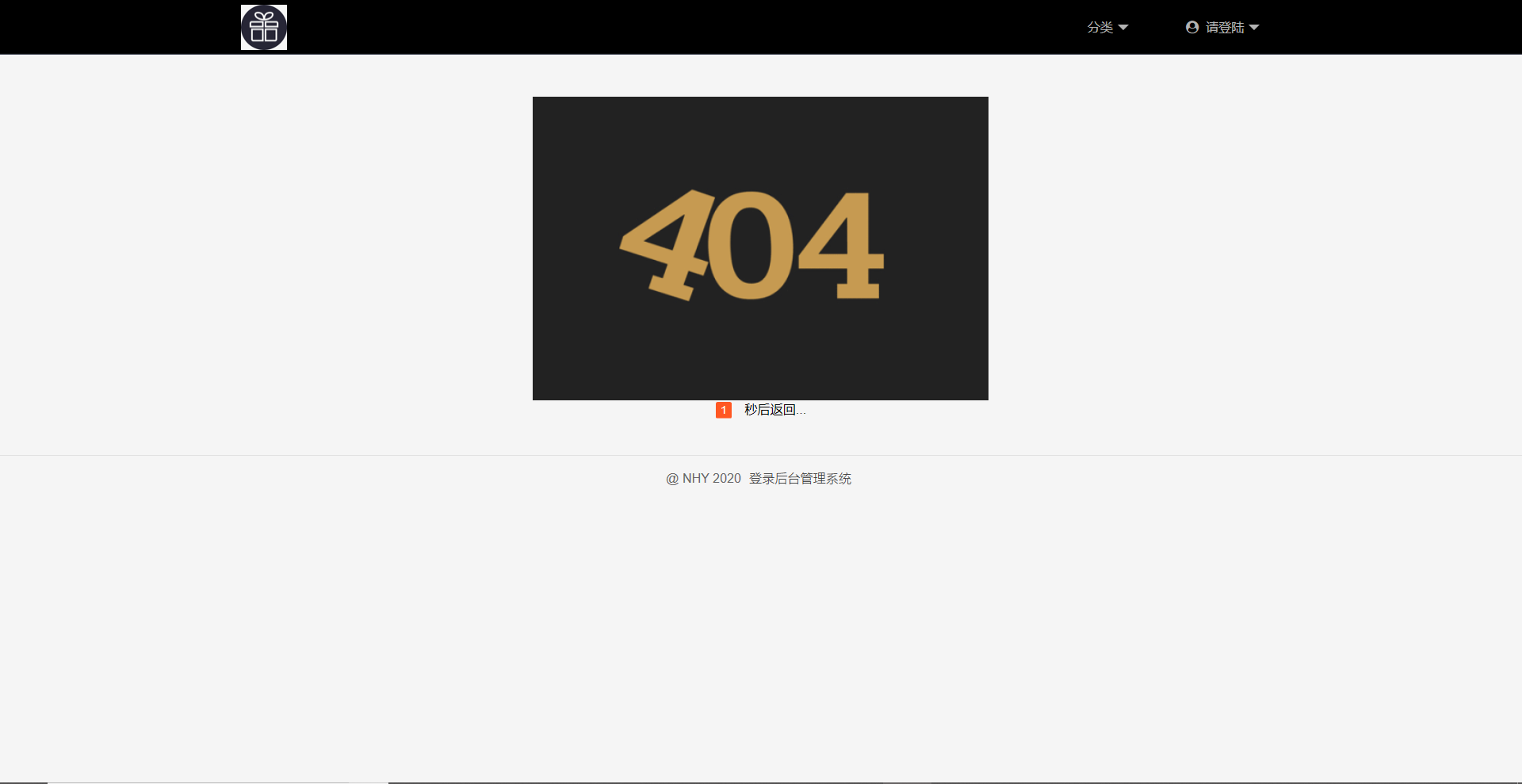


图 4‑26 登陆失败

（5）、展开右上角可以看到有个人信息，点击进入后，出现如下，可以对个人信息进行修改。

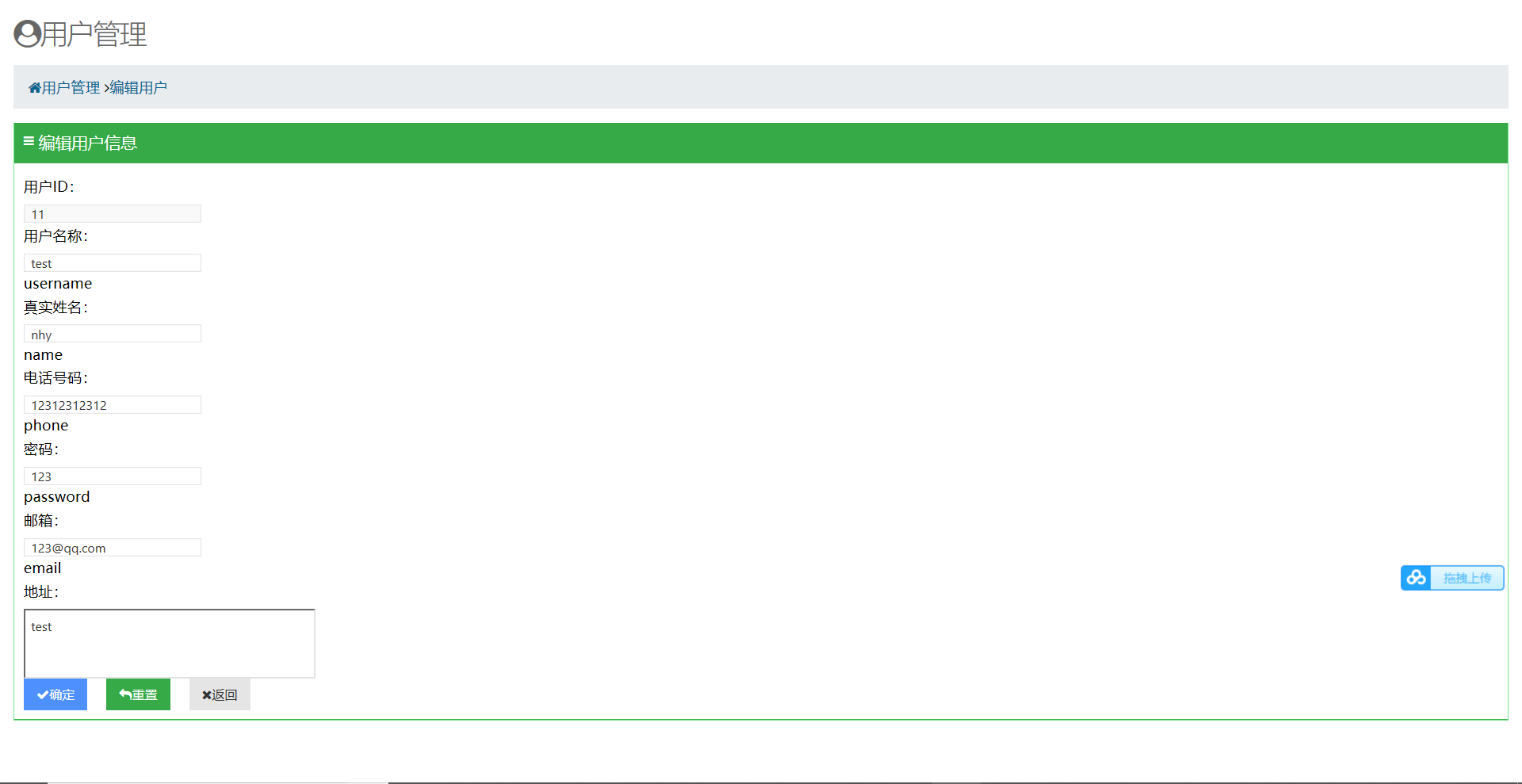


图 4‑27 修改个人信息

（6）、展开右上角分类可以看到多个分类选项，点击电脑可以查询出属于电脑的商品信息，如下图。

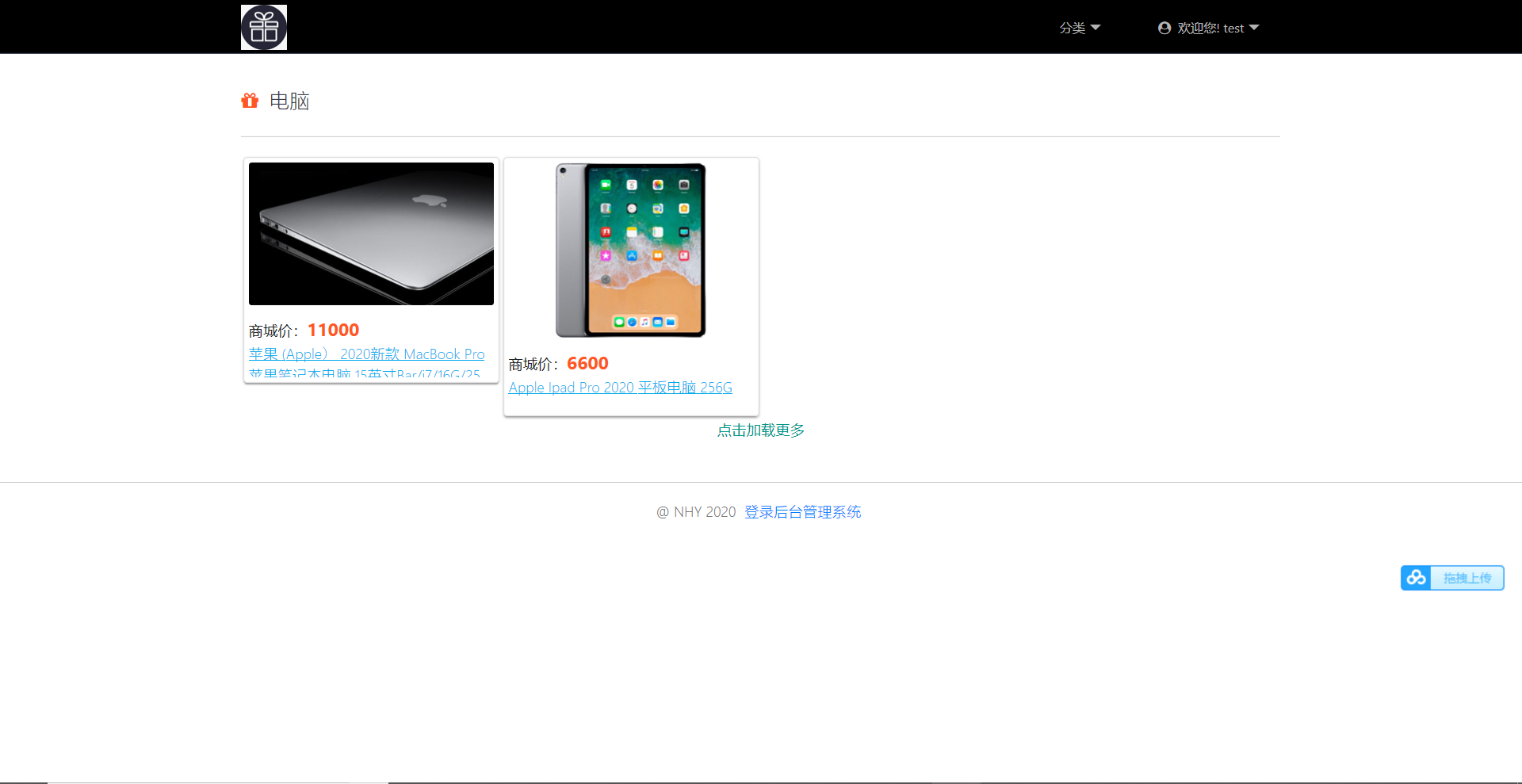


图 4‑28 分类查询

（7）、点击商品，可以进入商品详细信息页面，如下图



图 4‑29 商品详细信息

（8）、点击加入购物车，然后在右上角处进入购物车页面可以看到刚才添加的商品信息，并可以进入结账，如下所示。

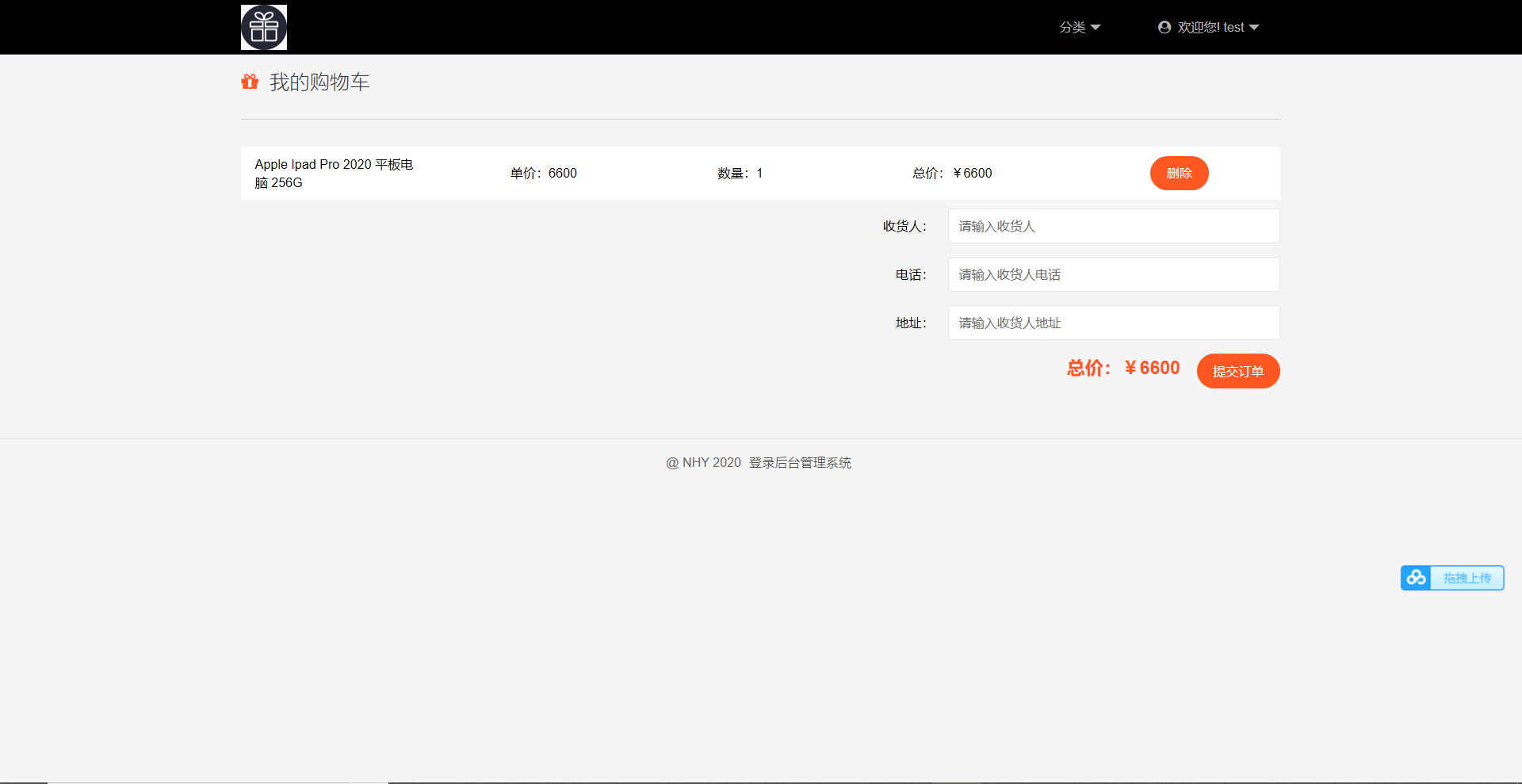


图 4‑30 购物车查询、提交订单

（9）、输入收货人、电话、地址后提交订单，然后在右上角进入我的订单页面，可以看到新生成的订单，如下所示。

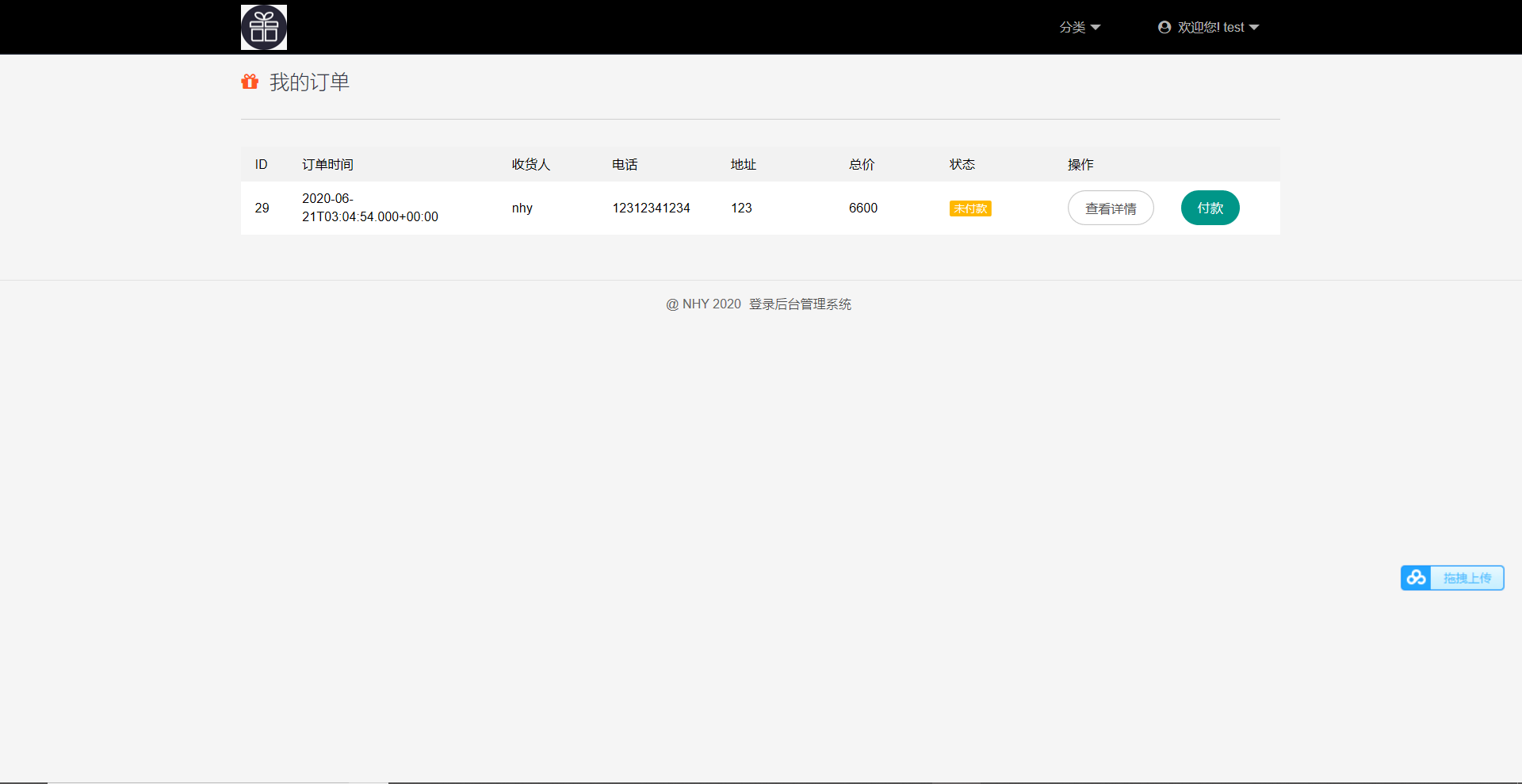


图 4‑31 订单查询

（10）点击付款后等待商家发货

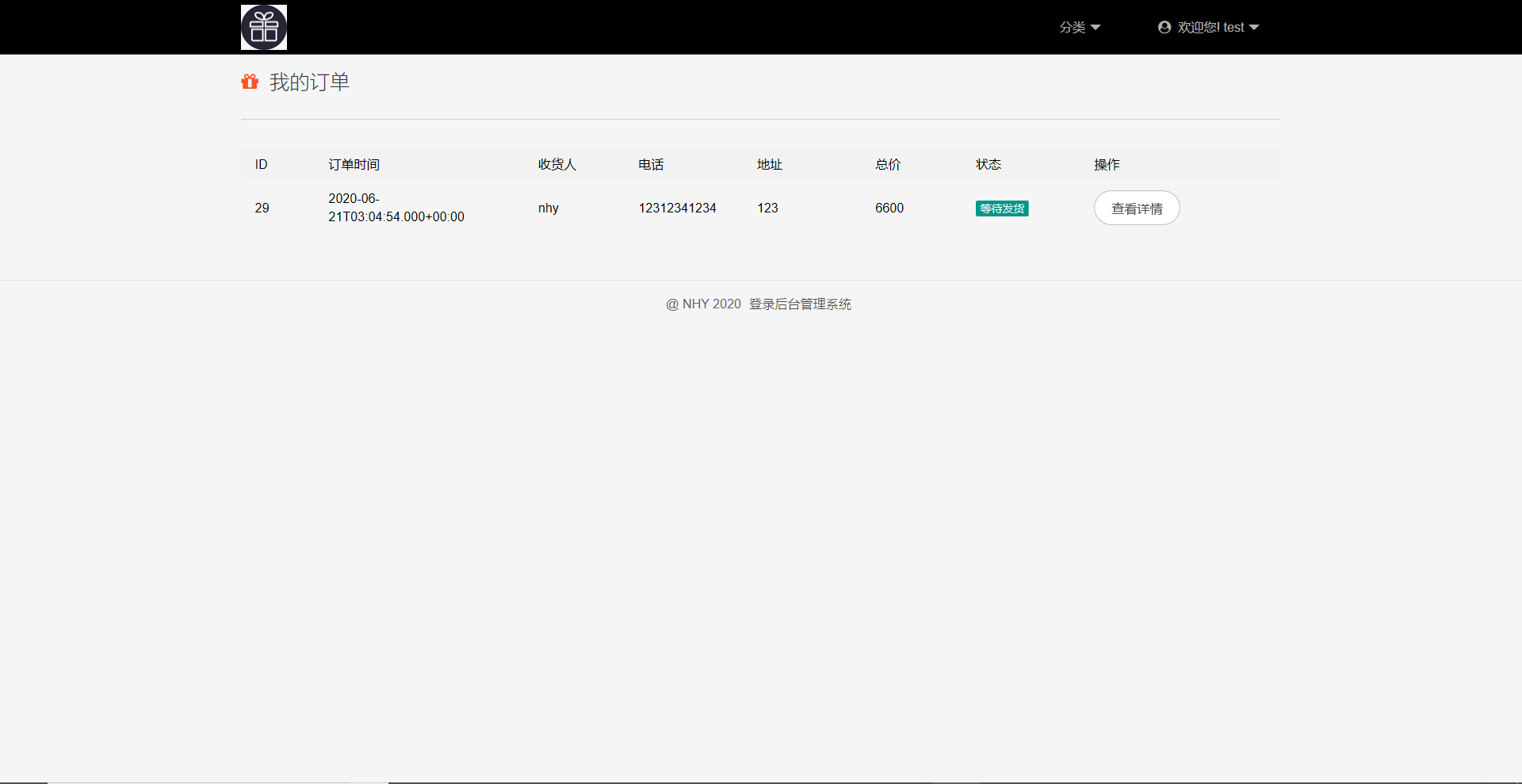


图 4‑32 支付订单

（11）、在所有页面的最下方都能够找到进入管理员操作的登陆界面入口，点击进入后，输入管理员账号密码，如下图所示。

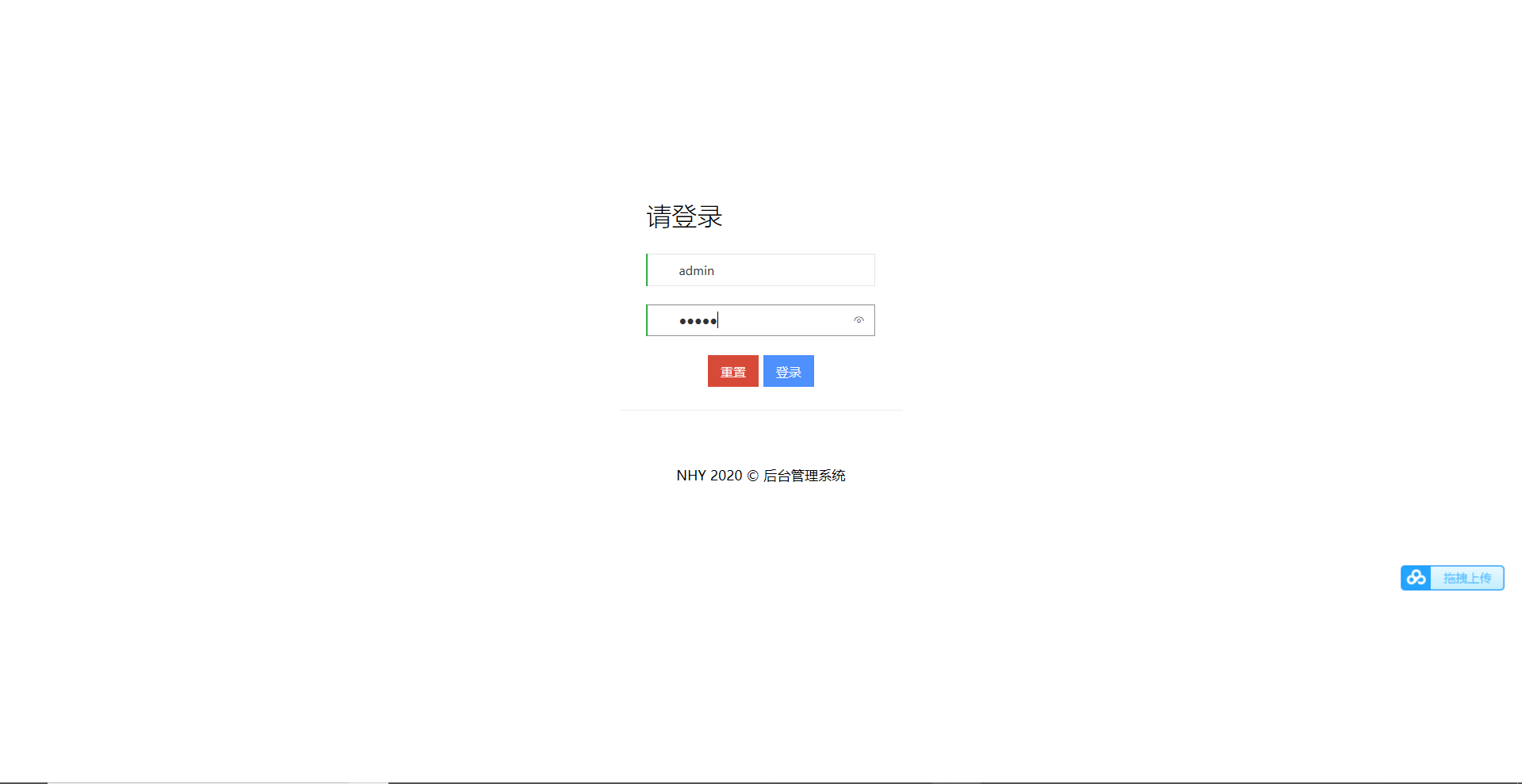


图 4‑33 管理员登陆

（12）、用户信息管理界面如下图所示。

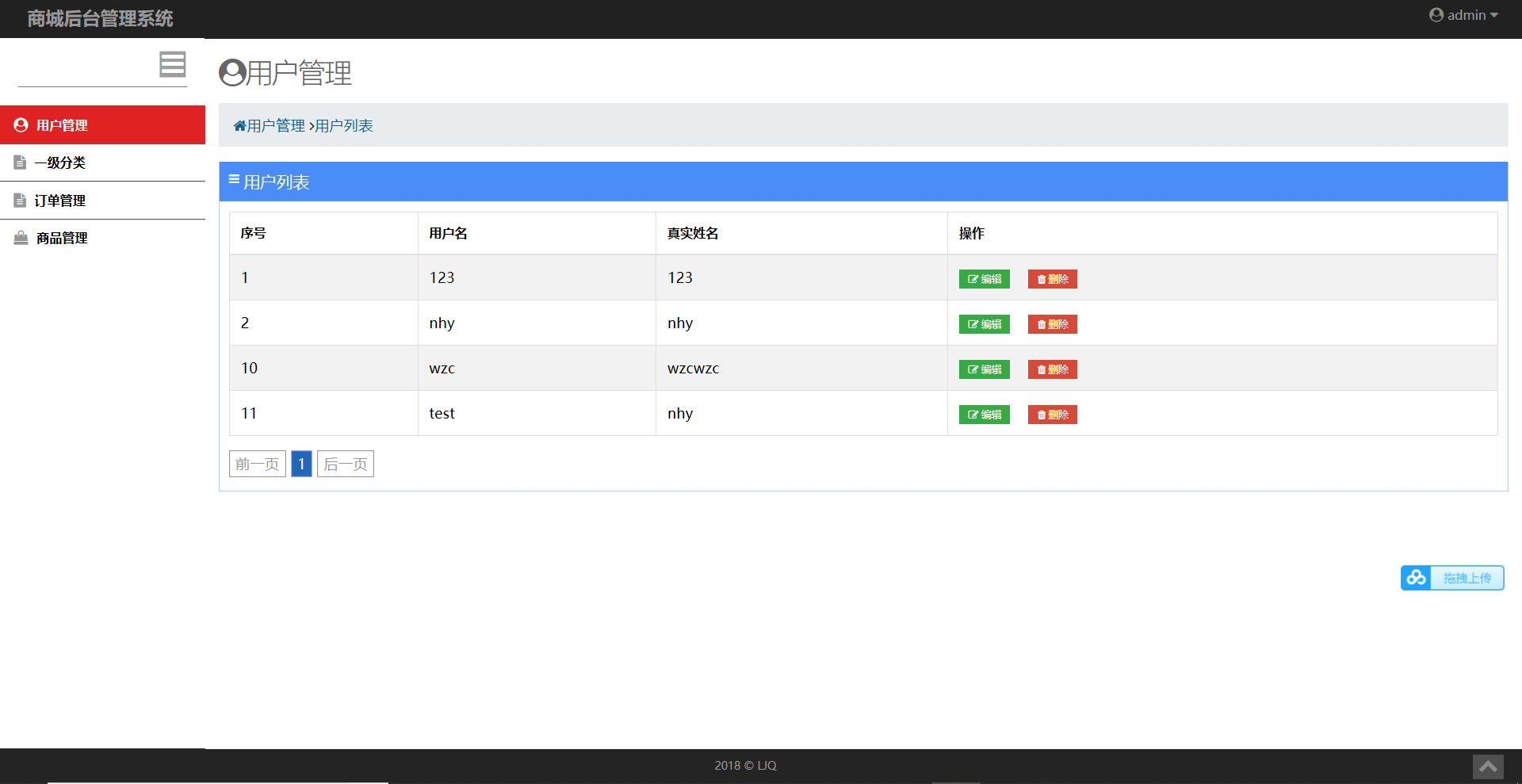


图 4‑34用户展示

（13）点击编辑用户出现如下页面

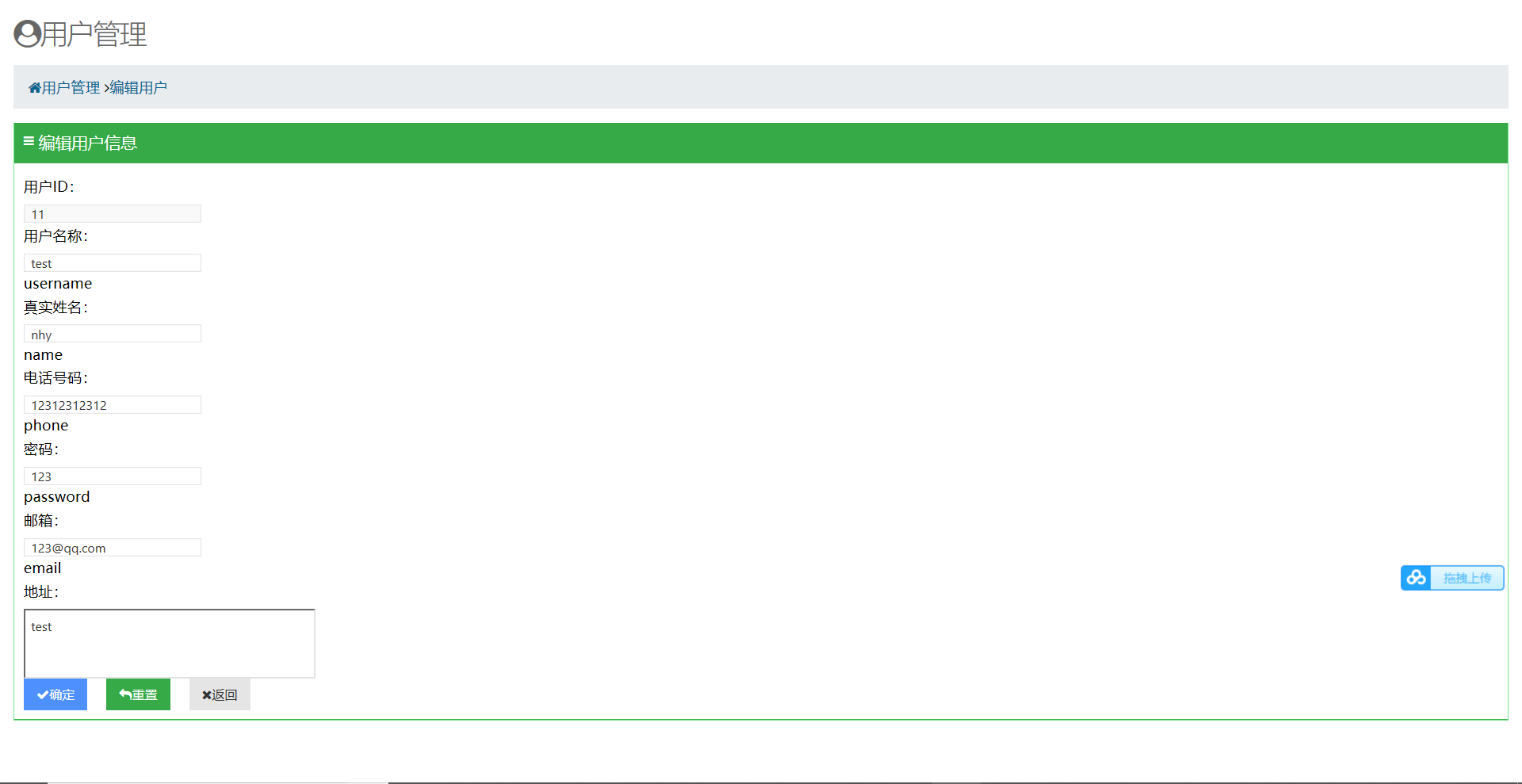


图 4‑35 用户编辑

（14）分类界面如下所示。

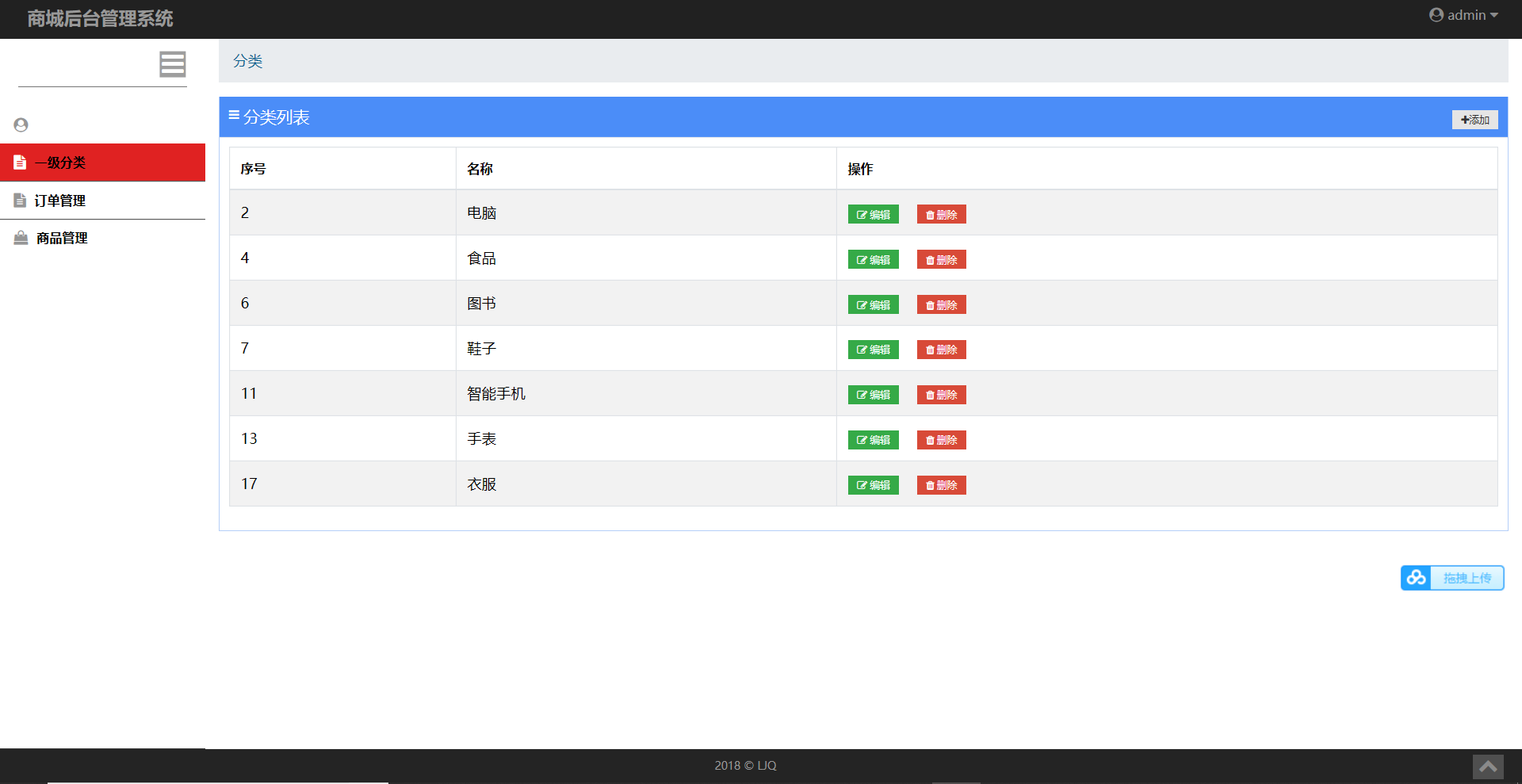


图 4‑36 分类展示

（15）可以进行添加、编辑、删除分类的操作，点击编辑，如下图所示。



图 4‑37 添加类别

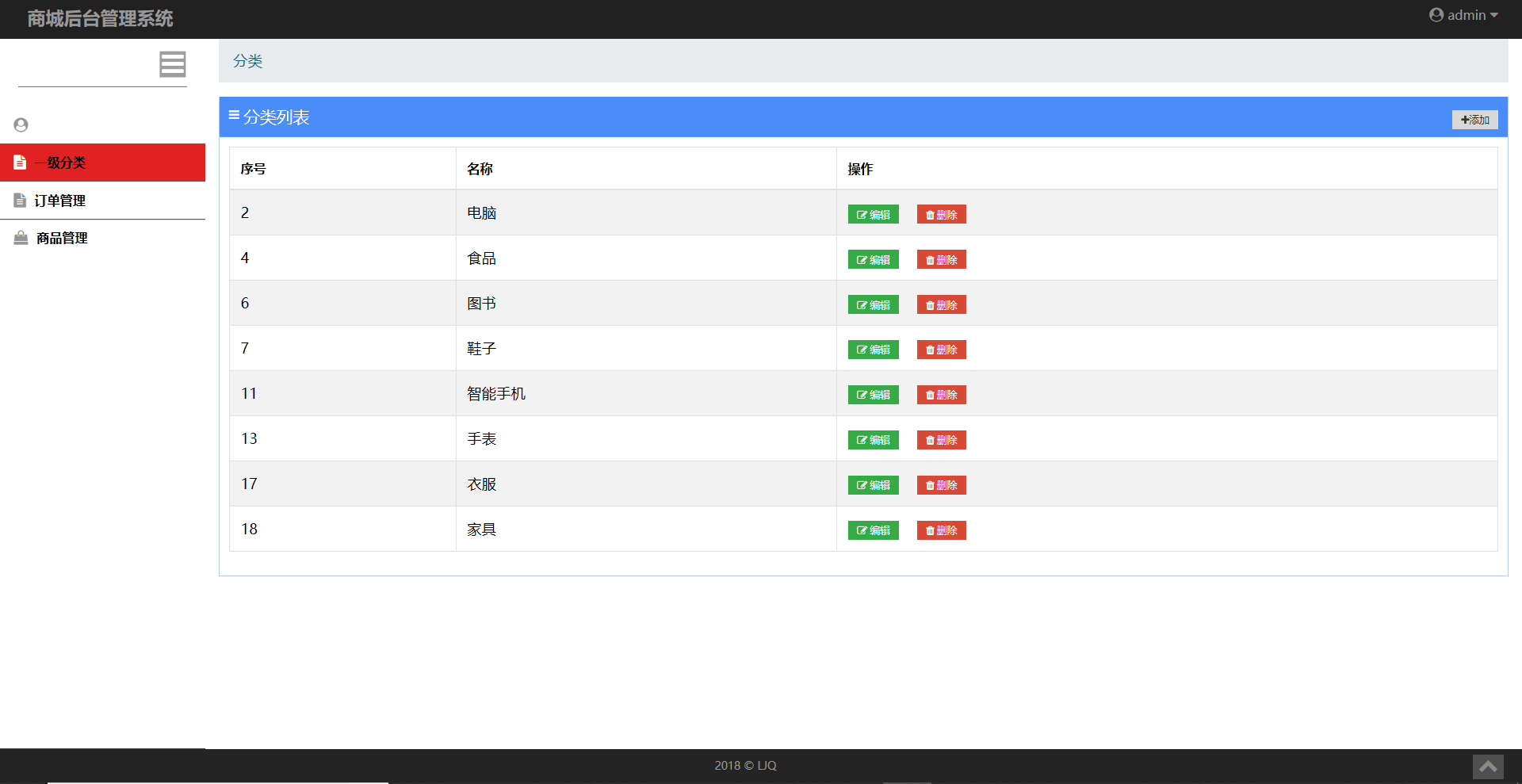


图 4‑38 添加成功

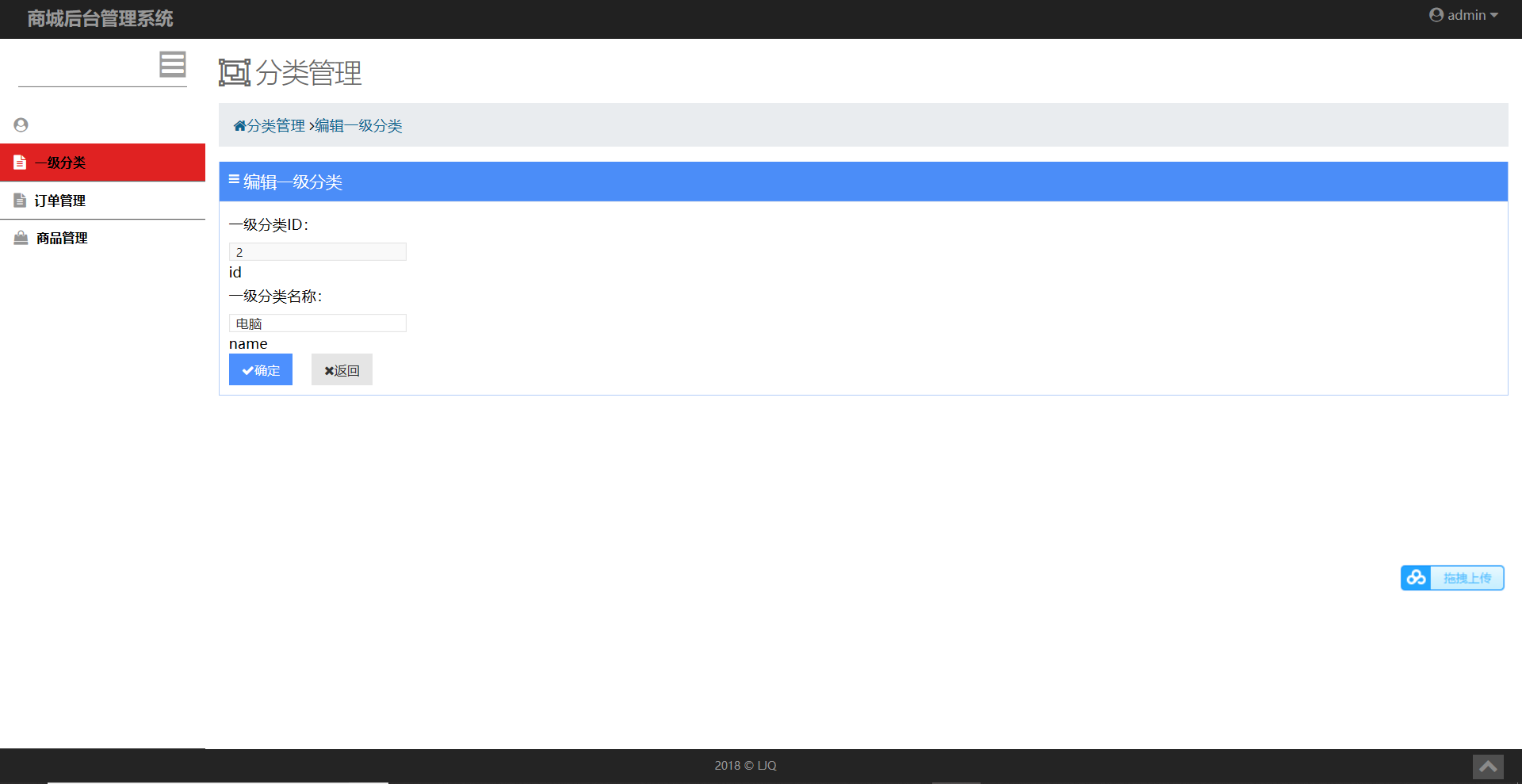


图 4‑39 分类编辑

（16）订单界面如下所示（订单用不同的状态，管理员可以更改）。

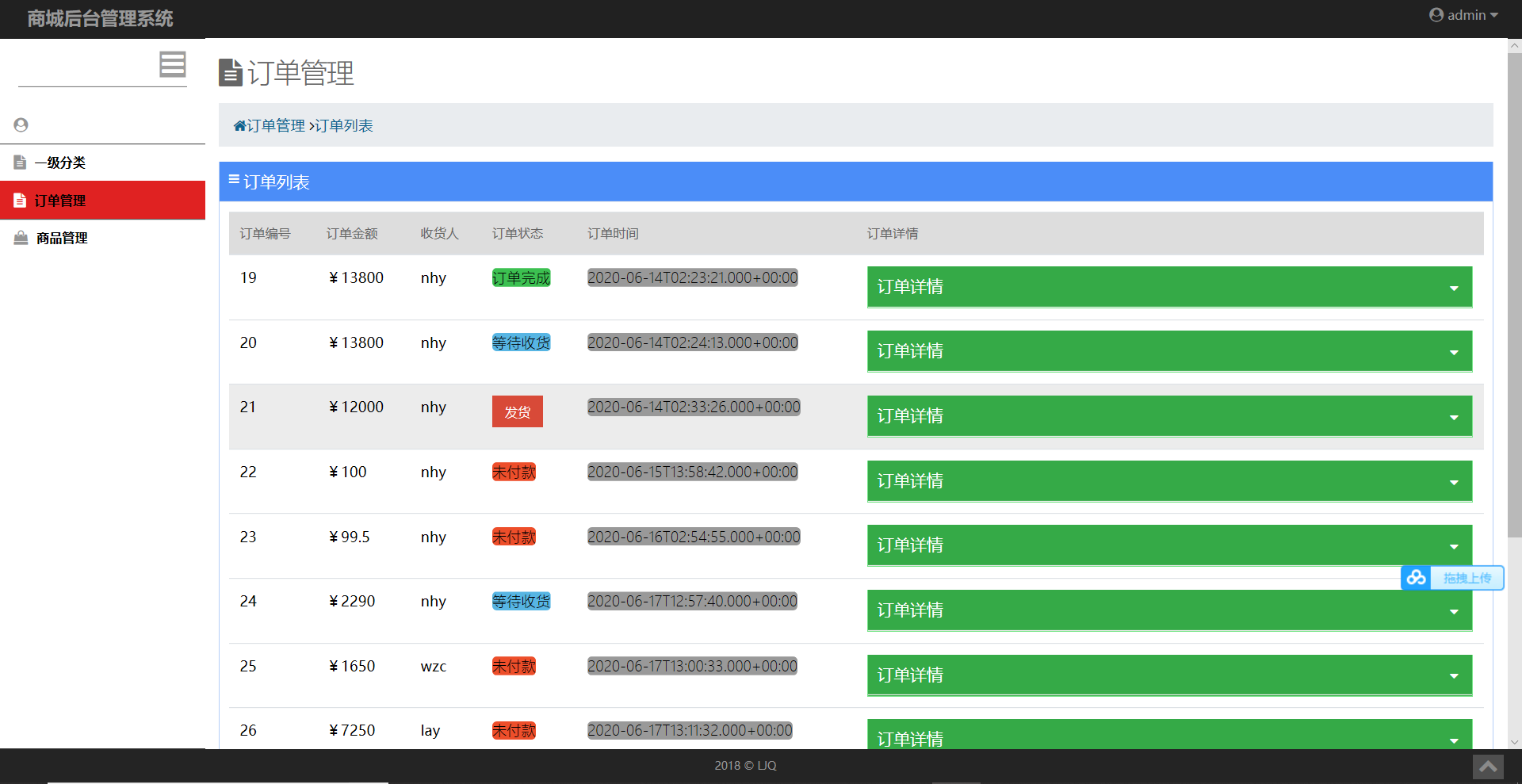


图 4‑40 订单展示）

（17）点击发货



图 4‑41

（18）商品信息界面如下所示。



图 4‑42 商品展示

（19）可以进行添加、编辑、删除商品的操作，点击编辑，如下图所示。

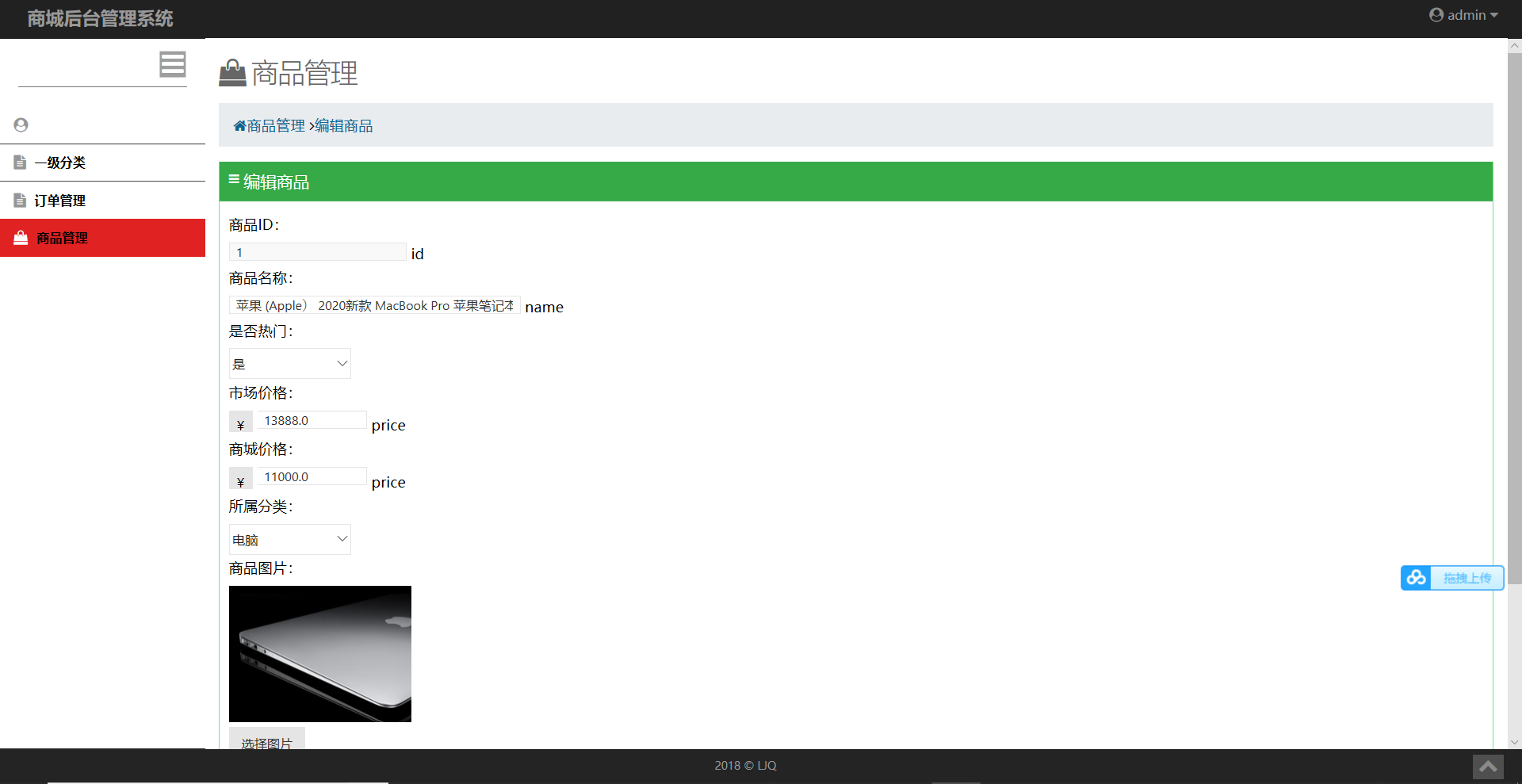


图 4‑43 编辑商品

（20）再次登陆测试账号可以看到订单中，订单状态变成了等待收货。

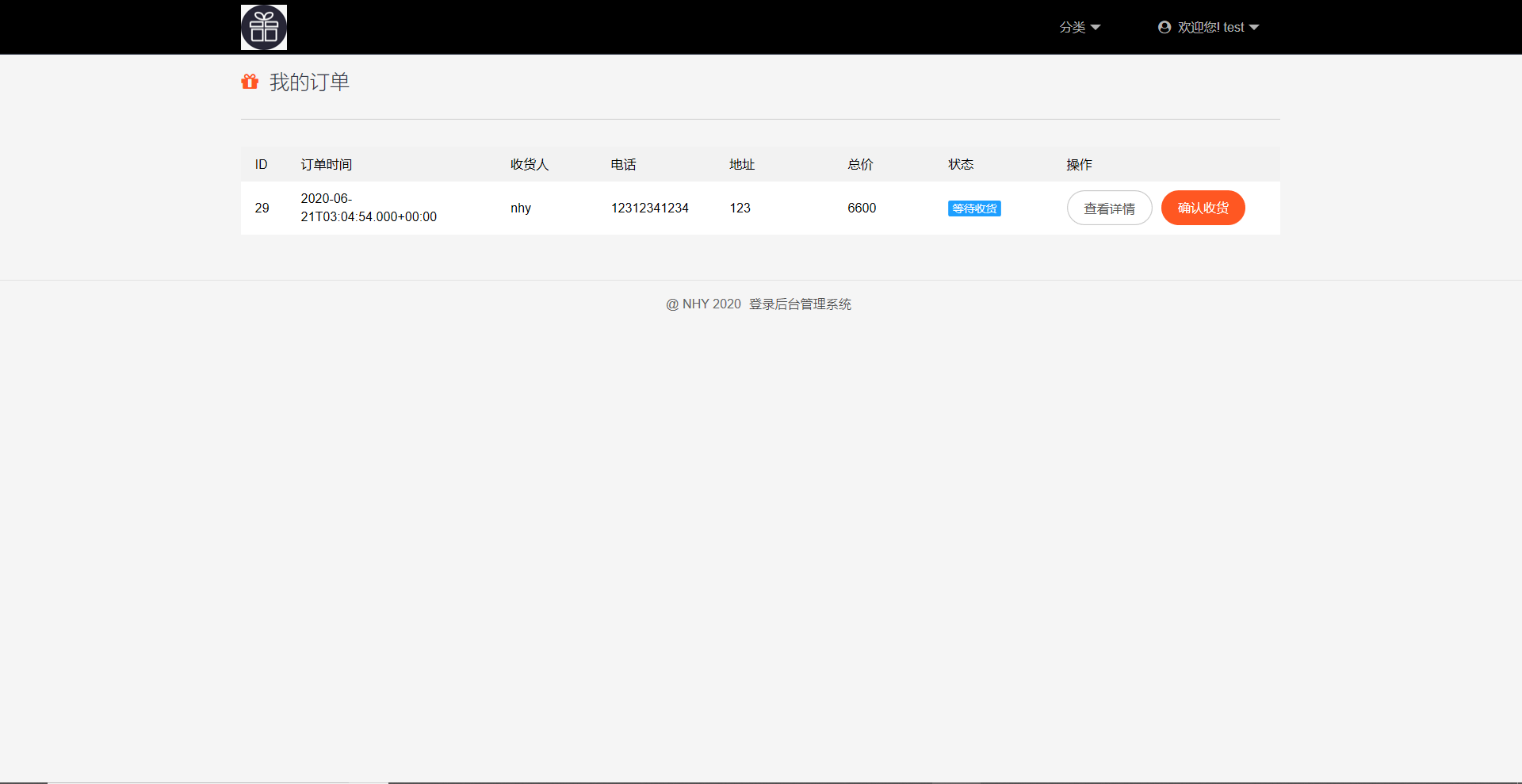


图 4‑44 等待收货

（21）点击确认收货。

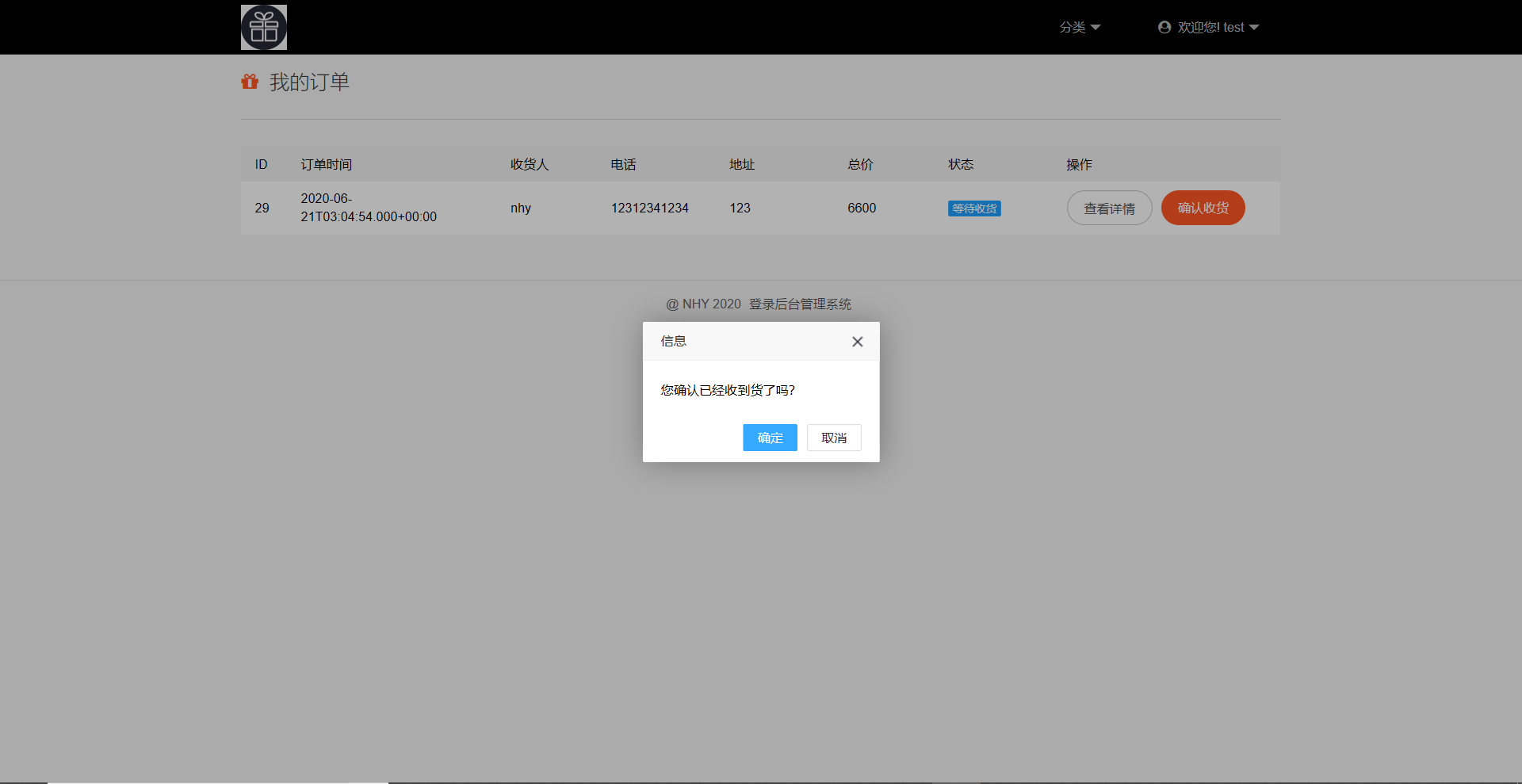


图 4‑45 确认收货

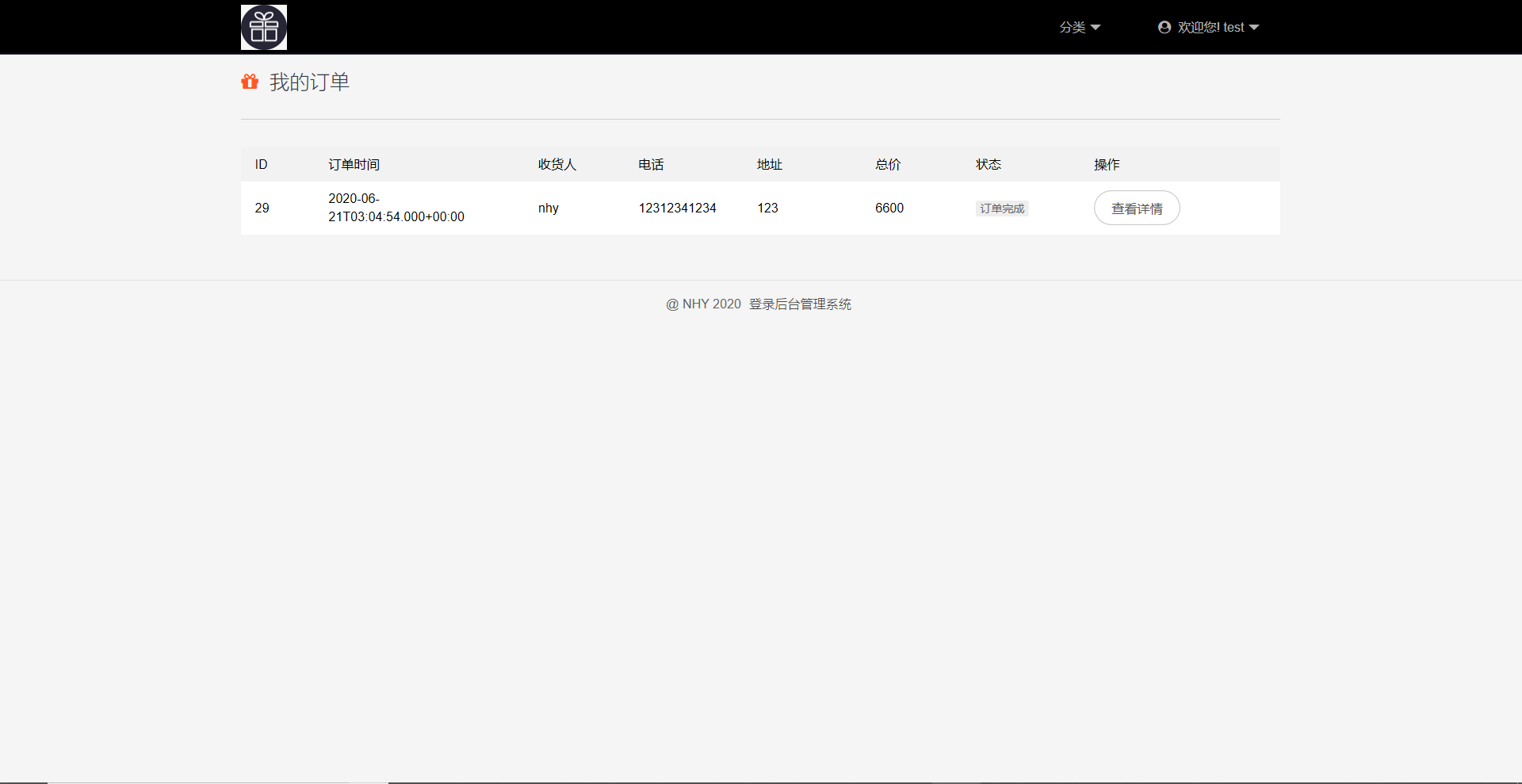


图 4‑46 订单完成

## 4.7 系统设计与实现总结

（1）分析题目需求：根据题目要求以及应用场景分析应用系统需要实现哪些功能。

（2）根据各实体间的关系，绘制了 ER 图，由此设计出了各个数据库表。

（3）根据需求分析的结果，对系统进行了总体设计，确定系统的大致功能。

（4）根据E-R图设计数据表的模式，将属性转化为数据库中的表

（5）向数据库中录入一定数据，进行功能测试，适当修改代码，逐步完善系统。

# 4 课程总结

（1）主要熟悉了 Navicat Premium 的使用，学会了新增用户并设置用户权限；掌握了SQL查询语句的多种用法以及较为复杂的查询逻辑的实现；掌握了触发器的简单使用。

（2）触发器的实现花费了较多时间，因为触发器的语句写法很灵活，而且不同数据库的触发器语法还大不相同。在编程的过程中发现，很多能够在高级语言中实现的功能，其实也可以在数据库中用触发器实现，两者对比，选择用触发器实现更好，可以减少编码量。

（3）软件功能学习部分的选做题没有完成，有时间还需要实现，加深对数据库的理解和操作的熟练程度。

（4）在网上销售系统的数据库设计部分大部分操作采用的是比较简单的查询语句，通过多条SQL语句组合实现功能，实际上应该可以一定程度上通过视图以及触发器的设置来简化操作。

（5）将数据库知识和实际结合，实现了一个功能比较完备的管网上销售系统，满足基本的管理者与用户的不同需求。

（6）接触到后端服务器的编写与前端页面的交互，了解了B/S模式下的设计方法。

# 附录

**环境配置**：

操作系统：Windows 10

IDE：IntelliJ IDEA 2020.1

JAVA ：JDK 1.8

Spring Boot：Spring Boot 2.3.0 released

配置文件application.properties(resource文件夹下)：可在配置文件中配置端口号以及连接MYSQL数据库。

**运行项目**：

1、在本地数据库中导入SQL文件数据，SQL文件存放在根目录data下

2、在配置文件中配置连接自己的本地数据库（MYSQL）或者其他数据库。

注意：该项目中使用的是MYSQL 8.x 连接驱动与 MYSQL 5.x有些许不同

MYSQL 8.x：com.mysql.cj.jdbc.Driver

MYSQL 5.x：com.mysql.jdbc.Driver

3、在IDEA中找到启动类MallApplication，运行即可。

4、访问前端网站首页网址：localhost:8080/mall。

5、在运行过程中，后台服务器会输出对应的SQL语句，可以在配置文件中选择不输出：spring.jpa.properties.hibernate.show\_sql=true