lab4开发

TimeLine

1 小组分工

1.1 人员安排

- 组织：林琰钧
- 数据库：林琰钧、马成
  - 使用Navicat进行Mysql数据库关系模型设计
  - 部署数据库
- 后端：林琰钧、马成
  - SpringBoot
  - 编写controller、service、mapper，以及必要的vo、dto、entity
- 前后端接口：林琰钧、马成、王兆瀚、李咸若
  - 使用APIFox进行前后端接口设计
  - 包括url、请求参数（xxxFVO）、响应参数（xxxVO）
- 前端：王兆瀚、李咸若
  - Vue
  - 根据前后端接口，设计页面
- 测试：马成、李咸若
  - 编写数据库数据脚本
  - 编写单元测试
  - 进行接口测试
- 部署：李咸若
  - 将项目前后端串联并部署

1.2 开发记录

- 5.25 (Thu.)
  - 林琰钧、马成、王兆瀚：下午算法课下课讨论数据库、API设计

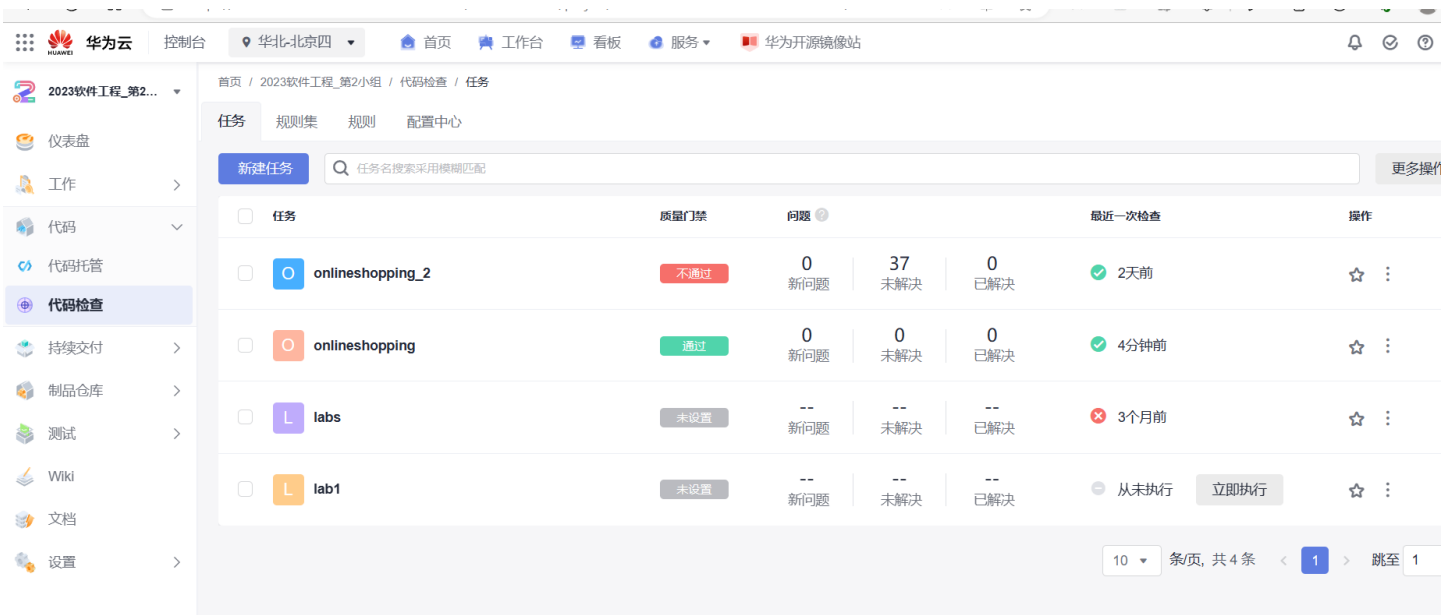
- To 6.6 (Tue.) 23:59.p.m
  - 林琰钧：完成需求分析、系统设计文档
  - 马成：完成后端代码编写、黑盒测试
  - 王兆瀚：完成前端代码编写
  - 李咸若：完成白盒测试

2 问题及解决方案

问题：期末考前发布了最后一个lab……大家都在复习，基本没有时间来做（建议以后2个lab，每个lab一个月，不要有第3个lab了……）

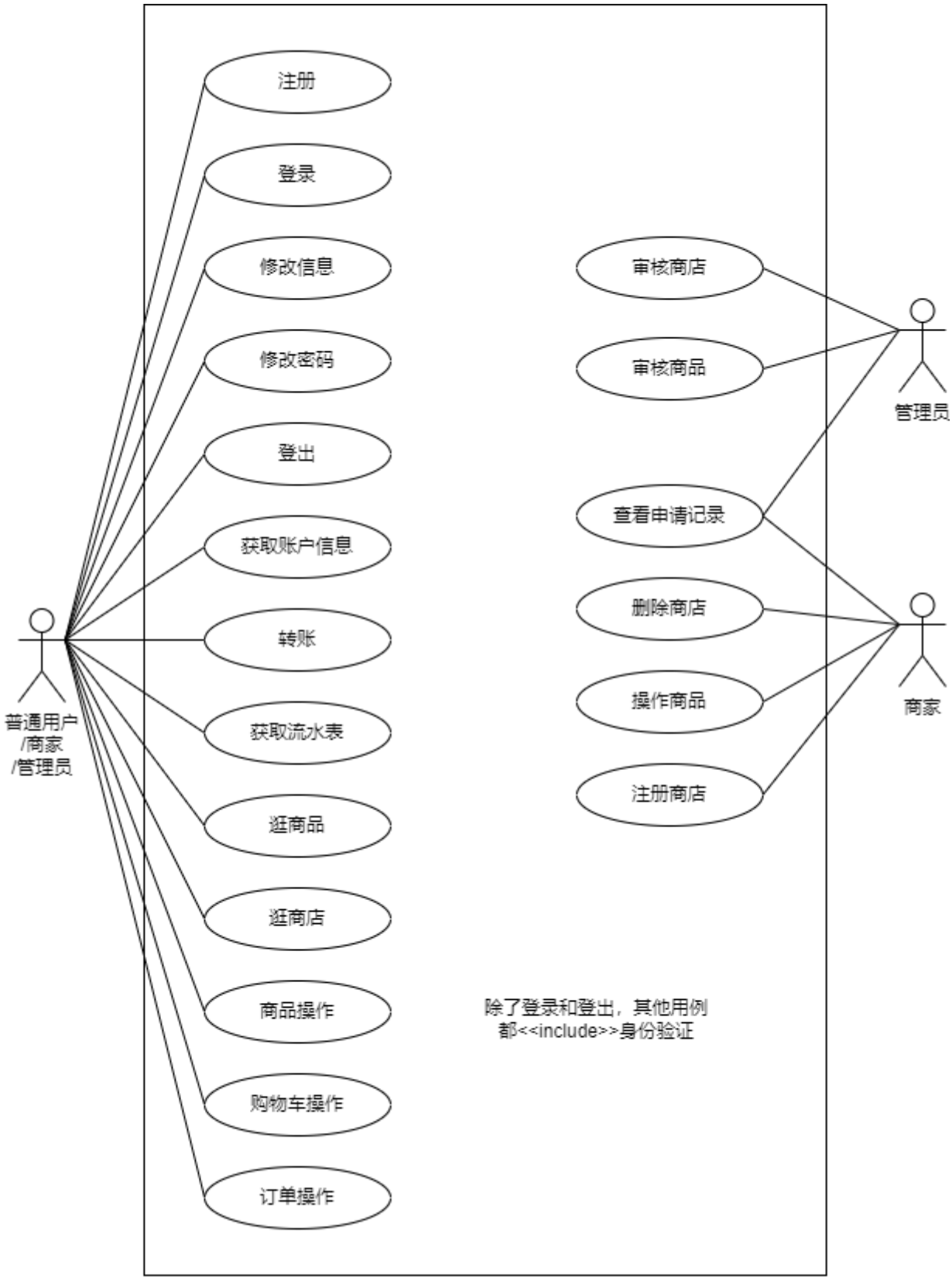
解决方案：赶ddl，最后几天紧赶慢赶赶出来了……

3 代码检查结果



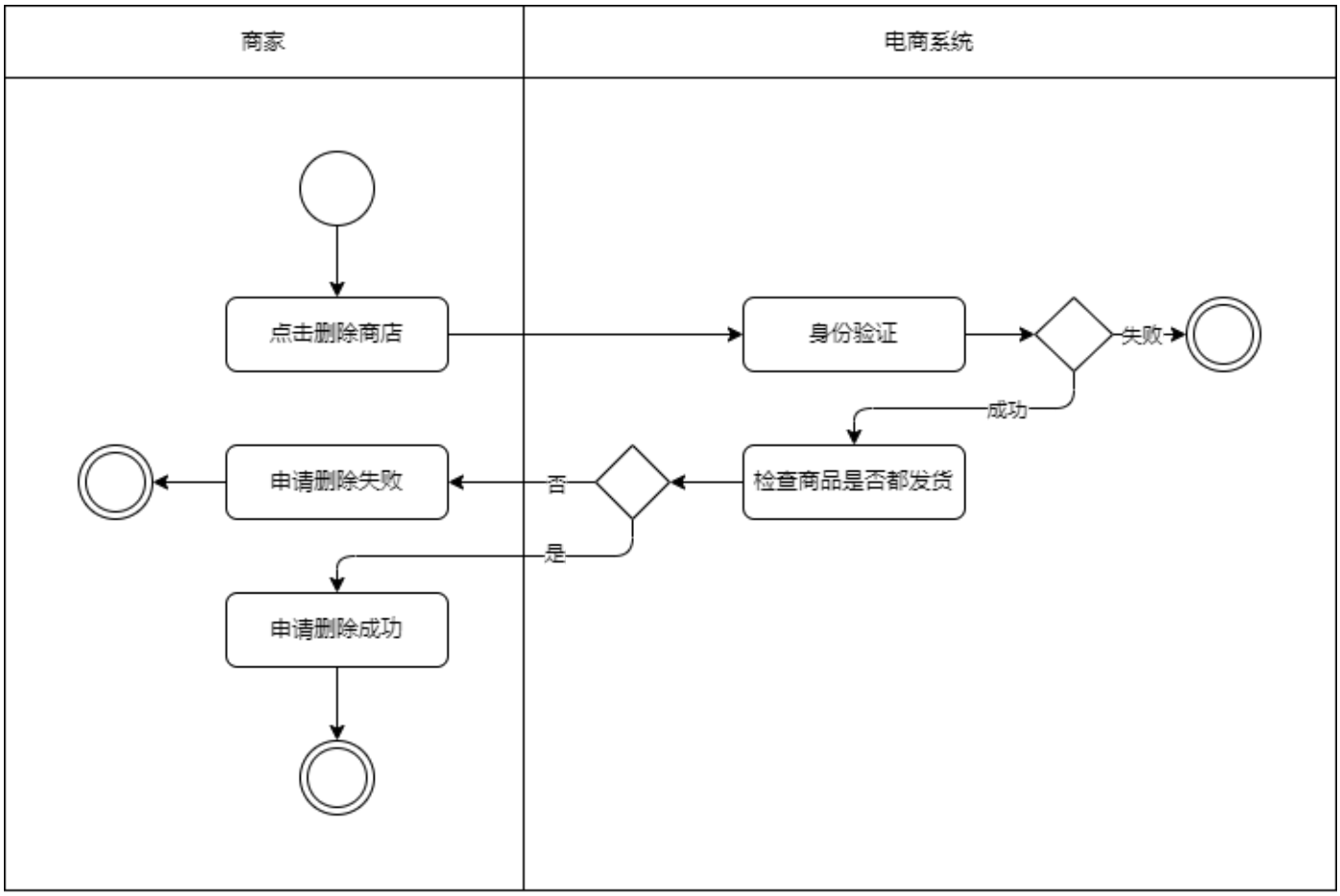
4 软件需求分析

4.1 系统整体用例图

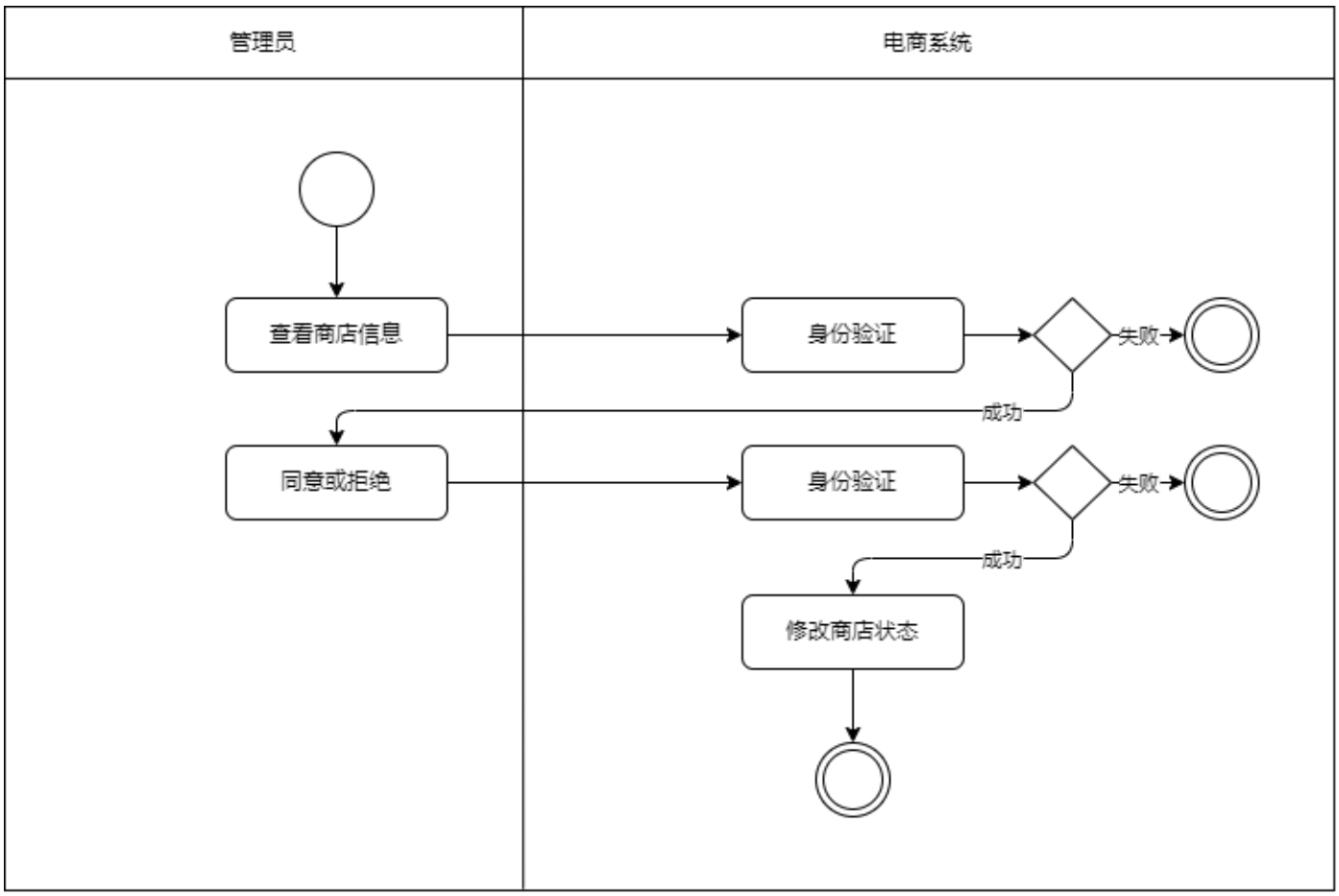


4.2 泳道图

- 商家申请删除商店

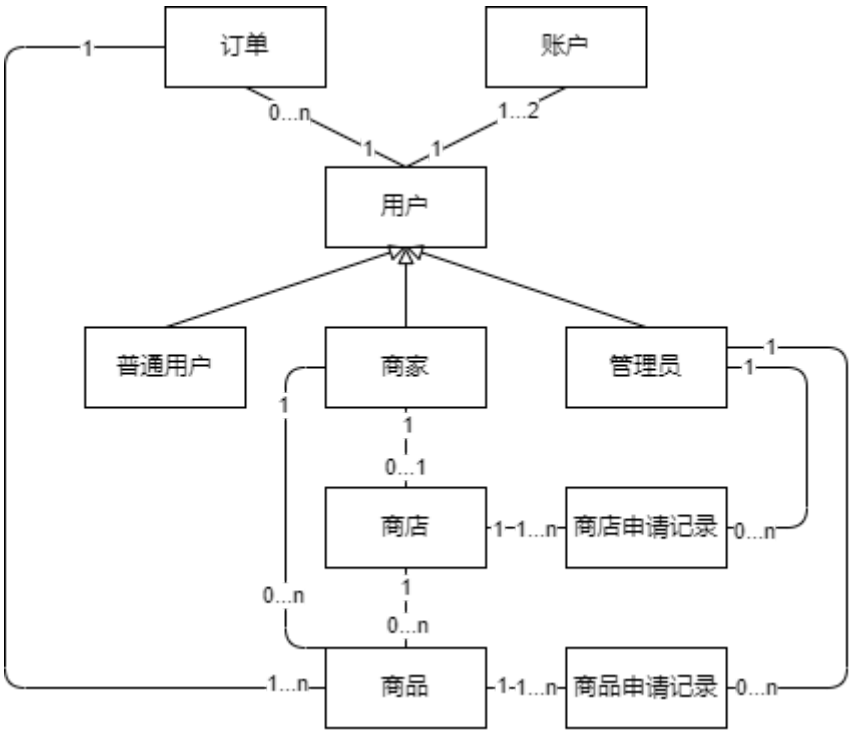


- 管理员处理商店申请



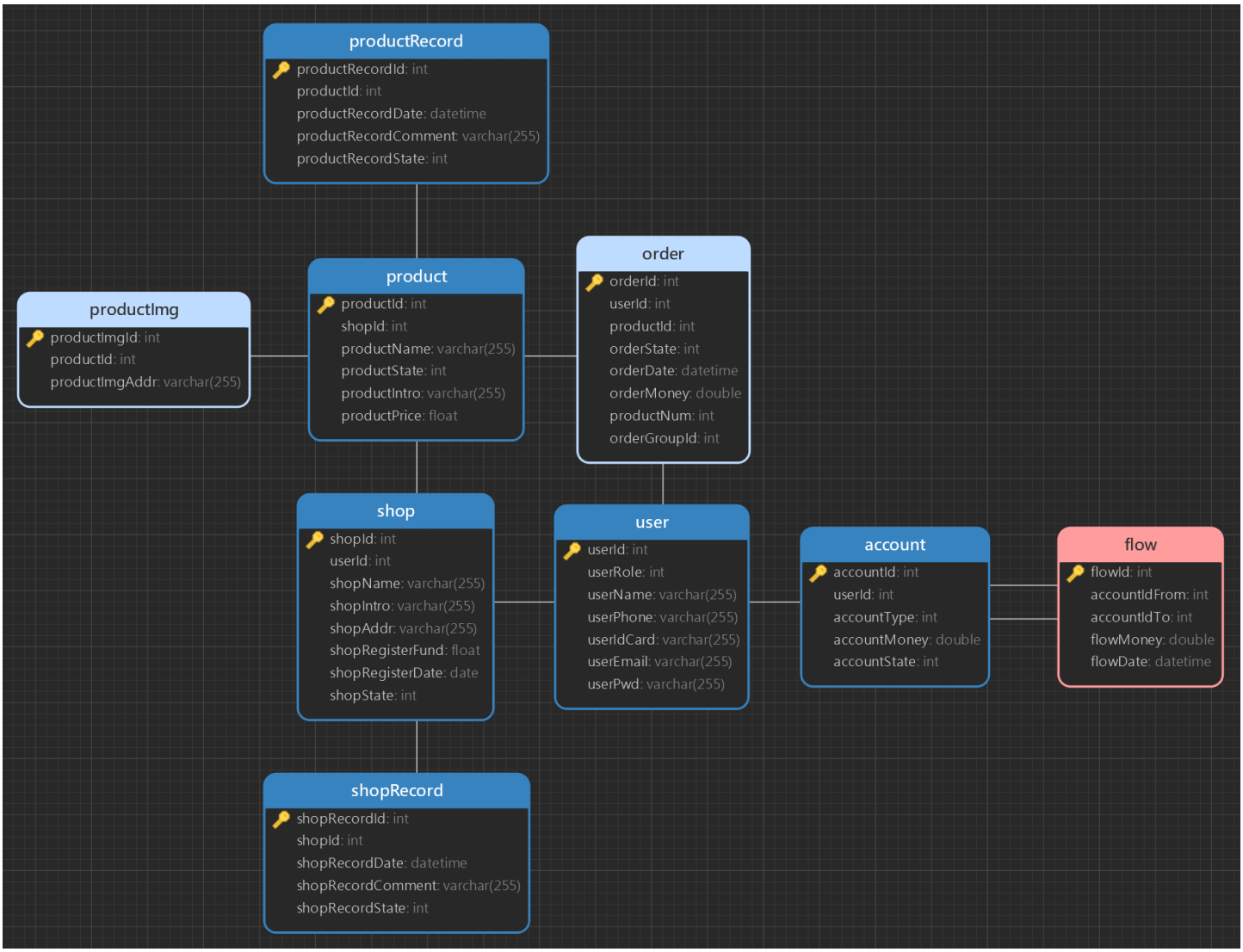
4.3 分析类图

详细的类属性见数据库表……



5 系统设计文档

5.1 数据库设计



- 上述设计中
  - 蓝色表示不能删除entry，但可以增加、修改
    - user本身没有注销接口，不允许删除
    - 所有的record作为归档记录，不希望被删除
    - product、shop、account有外键，删除会导致级联删除，而我们不希望看到record、flow和order中的字段被删除；因此，若非要删除的话，其删除表示为State字段的一个特定值
  - 白色表示可以删除entry
    - Img只能增加不能修改
    - order可以修改State
  - 红色表示可以增加，不能删除、修改
  - 有关图片的信息都存储在后端的某个文件夹，数据库存储图片地址
- user
  - 存储用户、商户、管理员的信息
  - userRole

        - 用户0
        - 商户1
        - 管理员2
    - 特殊id
        - 钱包：userId为1，userName为“wallet”，只是用来占坑的，作为外键引用的对象，login会特判不允许它登录
        - 管理员：userId为2，userName为“root”
- shop
    - 存储商店信息
    - shopState
        - 审核待通过0
        - 审核未通过1
        - 商店开放2（包括删除未通过）
        - 删除待通过3
        - 已删除4
- product
    - 存储所售卖的商品类别
    - productState
        - 审核待通过0
        - 审核未通过1
        - 已上架2（审核通过）
        - 已下架3
        - 已删除4
- account
    - 账户表
    - accountType
        - 个人账户0
        - 商店账户1
        - 商城利润2
        - 商城中间账户3
        - 虚拟账户4，唯一一个，用于充值的流水表accountIdFrom的外键引用

    - accountState
        - 有效0
        - 删除1
    - 特殊的id
        - dummy账户：accountId是1，userId是1。有无限多的钱，充值提现都与该账户有关
        - 中间账户：accountId是2，userId是2
        - 商城利润账户：accountId是3，userId是2
- flow
    - 流水表，用于转账
- order
    - 订单表
    - 把用户订单、商家订单、购物车都以订单表形式表示，表示状态的orderState不同
    - orderState（变动）
        - 加入购物车0
        - 下单未付款1
        - 已付款未收货2
        - 已付款已收货3
        - 已撤销4
    - 新增：
        - productNum：件数
        - orderGroupId：批量选择的商品
        - 批量选择点击购买时不生成orderGroupId，确认下单时才生成orderGroupId
- shopRecord是商店申请记录
    - shopRecordState
        - 开店申请未处理0，商店申请一经提交不允许修改
        - 开店申请通过1
        - 开店申请拒绝2
        - 删除申请未处理3
        - 删除申请通过4
        - 删除申请拒绝5
- productRecord是商品申请记录

    - productRecordState
        - 未处理0，此时如果修改商品信息直接覆盖时间戳
        - 上架申请通过1
        - 上架申请拒绝2
- 数据库脚本
    - cd到/resources/static/design/lab3/lab3-test.sql 所在的文件夹
    - mysql -u root -p <lab3-test.sql

5.2 API设计

- 用户（user）
    - 用户注册
        - 检查注册信息规范
        - 注册成功则生成个人账户
    - 用户登录
        - 生成token给前端
    - 用户信息修改
        - 不能修改角色、身份证号，其他的修改也需要检查规范
        - 修改成功则生成新token给前端
    - 用户密码修改
        - 需要输入原密码和新密码，独立于用户信息的修改
        - 修改成功则生成新token给前端
    - 用户退出
        - 啥也不做
- 账户（account）和流水（flow）
    - 获取账户信息
        - 给后端传一个账户类型
        - 后端通过token获取用户，检查是否有该类型的账户并返回
    - 转账
        - 充值和提现是特殊的转账
            - 充值从dummy账户转账；提现转账到dummy账户
            - dummy账户的金额始终不变，但个人账户或商店账户金额会改变

        - 商店注册申请与审批、商店删除申请与审批、商品购买付款与发货都会涉及与中间账户之间的转账
        - 每次转账都会插入一条flow流水表
    - 流水表
        - 给后端传一个账户类型
        - 后端通过token获取用户，检查是否有该类型的账户并返回出账或入账该账户的流水表
- 逛街（visit）
    - 逛商品（product）
        - 默认分页展示所有在售商品
        - 点击某个商品就会进入该商品的页面
            - 可以查看商品的详细信息
            - 可以加入购物车或直接购买（详情见order）
    - 逛商店（shop）
        - 在商品页面中可以跳转查看商品所属的商店的详细信息
        - 在商店详细信息中可以跳转分页查看该商店的所有在售商品
- 订单（order）
    - 用户（user）
        - 商品页面
            - 可以直接购买，即生成一个只有一种商品但是可以有多件的未付款订单。用户可以选择支付，撤销或暂时不做处理。
            - 可以加入购物车，如果购物车没有该商品则添加一条记录，负责直接修改购物车中该商品的数量，这里可以保证购物车中一个用户一个商品至多有一个购物车纪录
        - 购物车页面
            - 传输购物车列表的时候会传输商品状态，如果商品处于下架状态则不允许用户购买
            - 用户可以选择希望一起购买的若干记录，此时前端传递给后端后后端会返回所有选中订单的详细信息供用户确认。
            - 用户确认后会产生一个有多种商品的未付款订单。用户可以选择支付，撤销或暂时不做处理。
            - 可以在购物车页面修改购物车中某一个记录的商品数量。
        - 订单页面
            - 可以查询用户的订单，会有分页功能，会传输给前端所有有关这个若干个订单的详细商品纪录并且会按照订单号排序，具体的显示方式交给前端操作。

        - 可以查询某个订单的具体情况，会显示所有商品纪录
        - 可以在订单页面支付或撤销某个订单。
    - 物流页面显示用户所有已付款未收货订单
    - 交易完成页面显示用户所有已收货订单
  - 商家（owner）
    - 物流页面显示商家所有已付款未收货订单
      - 点击发货，即将已付款未收货的订单变为已付款已收货的订单；中间账户资金转移至商店账户
    - 交易完成页面显示商家所有已付款已收货订单
- 商户（owner）
  - 商店（shop）
    - 商店注册
      - 商店注册时，提交注册，注册资金从虚拟账户转移到中间账户，在管理员审核之前，商店信息都不允许修改，并且不能添加商品
      - 如果管理员拒绝商店注册，则注册资金从中间账户转移到虚拟账户，此时商户可以修改商店信息，之后重新提交审核，视为重新走了一遍注册流程
      - 如果管理员同意注册，则生成一个商店账户，注册资金从中间账户转移到商城利润账户，之后商店信息就不再允许修改了，此时可以添加、修改、删除商品
    - 商店的删除
      - 检查商品是否都已发货，如果有未发货则不允许删除；
      - 如果都已发货，则将商店状态变为删除待通过，所有已上架的商品状态变为已下架，已提交审核变成审核拒绝，原因是商家申请删除商店
      - 如果审核通过，则将所有未删除的商品转化为已删除，将商店账户的余额转移给虚拟账户，将商店的账户设置为已删除
      - 如果审核未通过，只要把已下架的的商品转化为已上架
  - 商品（product）
    - 商品列表包括审核待通过、审核未通过、已上架的商品
    - 添加商品只能添加商品的文字信息
      - 初始状态是审核待通过
      - 添加商品审核记录
    - 删除商品前需要保证商品都已发货
      - 把商品状态变为已删除

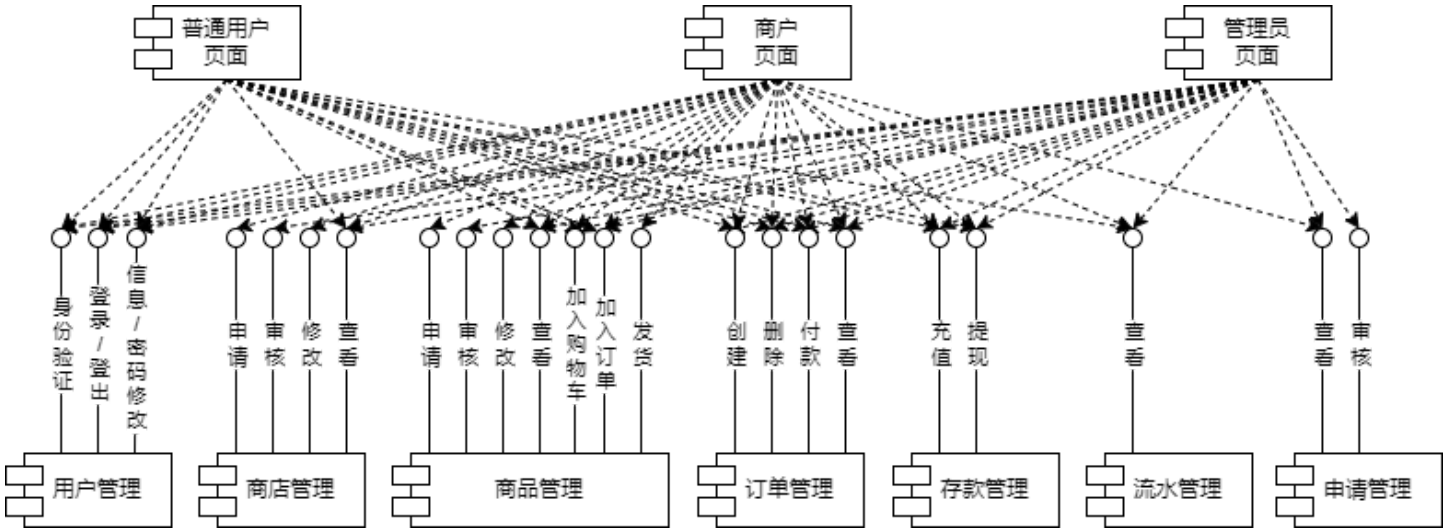
        - 删除与商品关联的所有图片
    - 修改商品前需要保证商品都已发货
        - 可以增加或删除图片，或修改文字信息，这两种修改是独立的
        - 修改会导致状态从审核未通过或已上架变为审核待通过
        - 添加商品审核记录
- 管理员（admin）
    - 商店（shop）
        - 审核申请注册的商店
            - 不通过，则按照商店注册失败流程
            - 通过，则按照商店注册成功流程
        - 审核申请删除的商店
            - 不通过，则按照商店删除失败流程
            - 通过，则按照商店删除成功流程
    - 商品（product）
        - 审核申请上架的商品
            - 不通过，则商品状态变为审核未通过
            - 通过，则商品状态变为已上架
            - 更改商品审核记录
- 申请记录（record）
    - 商店申请记录
        - 商店注册或删除操作都会生成状态为未处理的申请记录
        - 一经提交，在审核之前，都不允许对商店中的信息进行进一步的操作
    - 商品申请记录
        - 商品新增会生成状态为未处理的申请记录
        - 商品修改，如果没有该商品的申请记录，则会生成状态为未处理的申请记录；如果已经是审核待通过，则更新审核记录的时间戳
    - 审核后的记录状态会变为已处理
        - 如果管理员审核通过则会记录“审核通过”
        - 如果管理员审核未通过则会有未通过原因字段
        - 商店申请删除时，那些待审核的商品会自动未通过审核，会记录“申请删除商店”
    - 请求商店纪录

- 如果是Admin获取所有商店记录
- 如果是Owner获取没有删除的店铺所有商店纪录

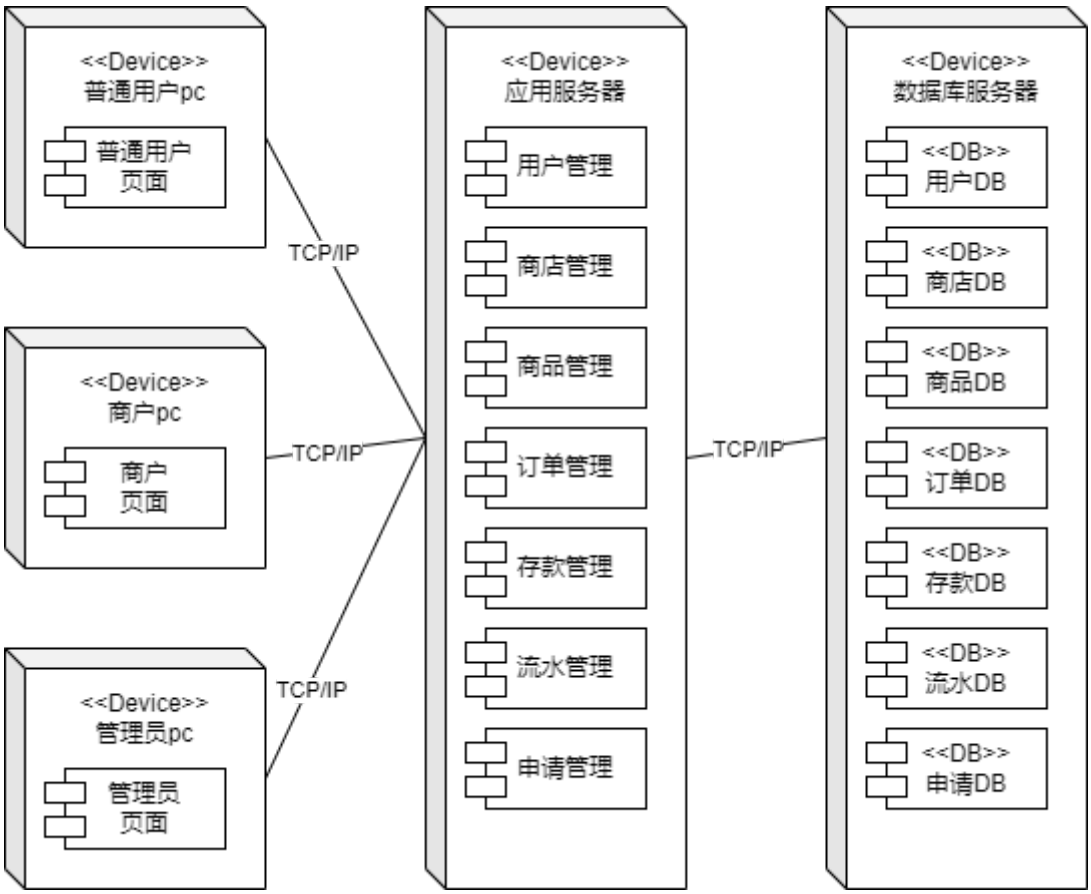
◦ 请求商品纪录

- 如果是Admin获取所有商品记录
- 如果是Owner获取没有删除的店铺所有商品纪录

5.3 构建图



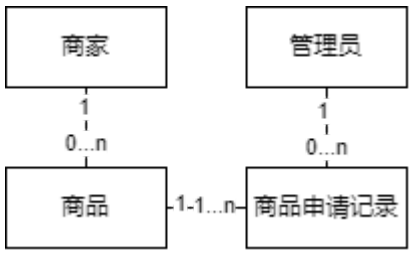
5.4 部署图



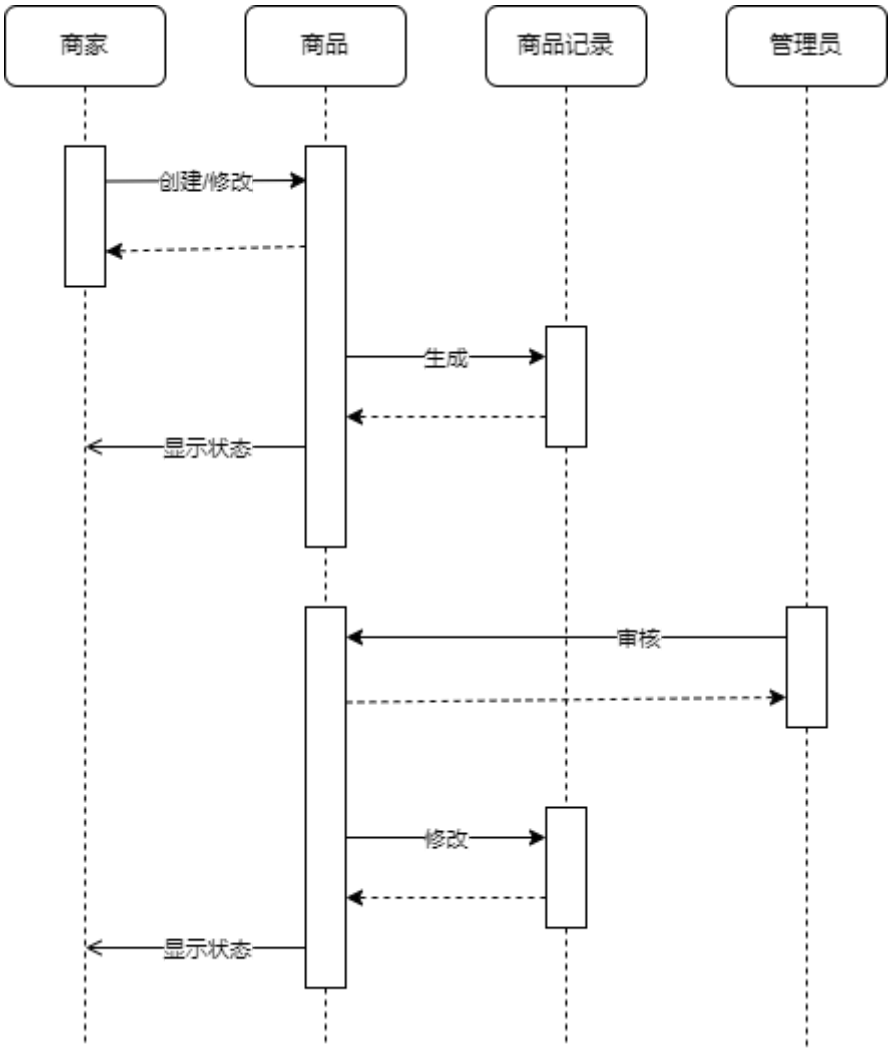
5.5 关键功能示例

以商户创建、修改商品，商家审核为例

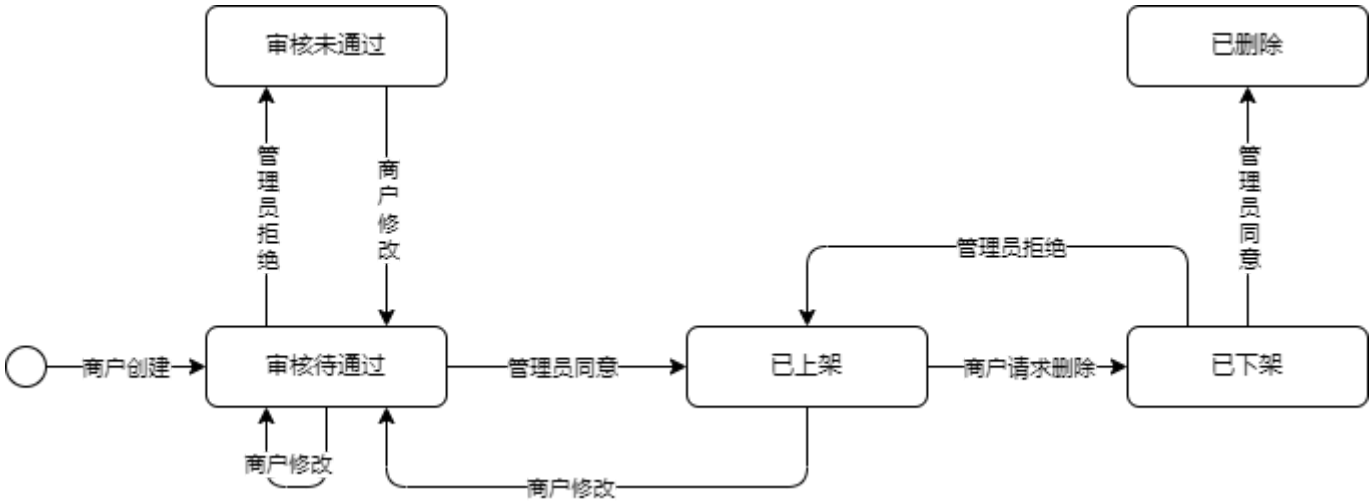
5.5.1 类图



5.5.2 顺序图



5.5.3 状态机图



6 软件测试

6.1 白盒测试

- 使用JUNIT框架对一些Service功能进行测试
- 测试了管理员的展示所有待检测商品的功能，在这里设计了三个测试数据
  - page=1 尝试了一个在范围内的测试数据，代码正常返回了应有的数据
  - page=10000 我们待检测的商品显然没有这么多，代码正常的告知了我们没有这么多页，并且没有返回数据
  - page=-1 我们要求page必须是正整数，page=-1是不符合条件，这里代码成功抛出了一个错误，并给了正确的错误信息
- 测试了注册接口，注册需要有较多的检验根据这个设计了如下数据
  - 正常插入
  - 用户名重复
  - 电话号码重复
  - 身份证重复
  - 邮箱重复
  - 注册角色错误
  - 电话格式错误
  - 身份证号格式错误
  - 邮箱格式错误

6.2 黑盒测试

- 使用APIFOX的自动化测试功能
- 对账户余额进行充值、提现的黑盒测试

- 输入的等价类划分

用户登录	用户名	1，合法用户名 2，非法用户名	访问账户界面	账户类型	5，错误账户 6，正确账户	提现	金额	9，大于等于0，小于等于账户余额 10，大于账户余额
	密码	3，正确密码 4，错误密码	充值	金额	7，非负整数 8，负数			11，负数

- 测试的流程

- 测试用例

数据集名称	name	pwd	添加变量(列)
正常登录	Wilson	97e3b086a573c3c6704ef36a43a	
密码错误	Wilson	123456	
用户名格式错误	WilsonIsTheBestGuyInTheWorld	123456	

（具体的访问参数在测试部署的时候通过随机数生成）

- 测试结果

- 对管理员批准/拒绝商店申请进行了黑盒测试

- 输入的等价类划分

输入参数				
用户权限	1. 管理员 2. 非管理员	通过申请的商店	5. 商店状态是审批 6. 商店状态不是审批	
查询申请中商店页码	3. 页码存在 4. 页码不存在	拒绝申请的商店	商店状态	7. 商店状态是审批 8. 商店状态不是审批
			评论	9. 有评论 10. 评论为空

- 测试的流程

- 测试用例

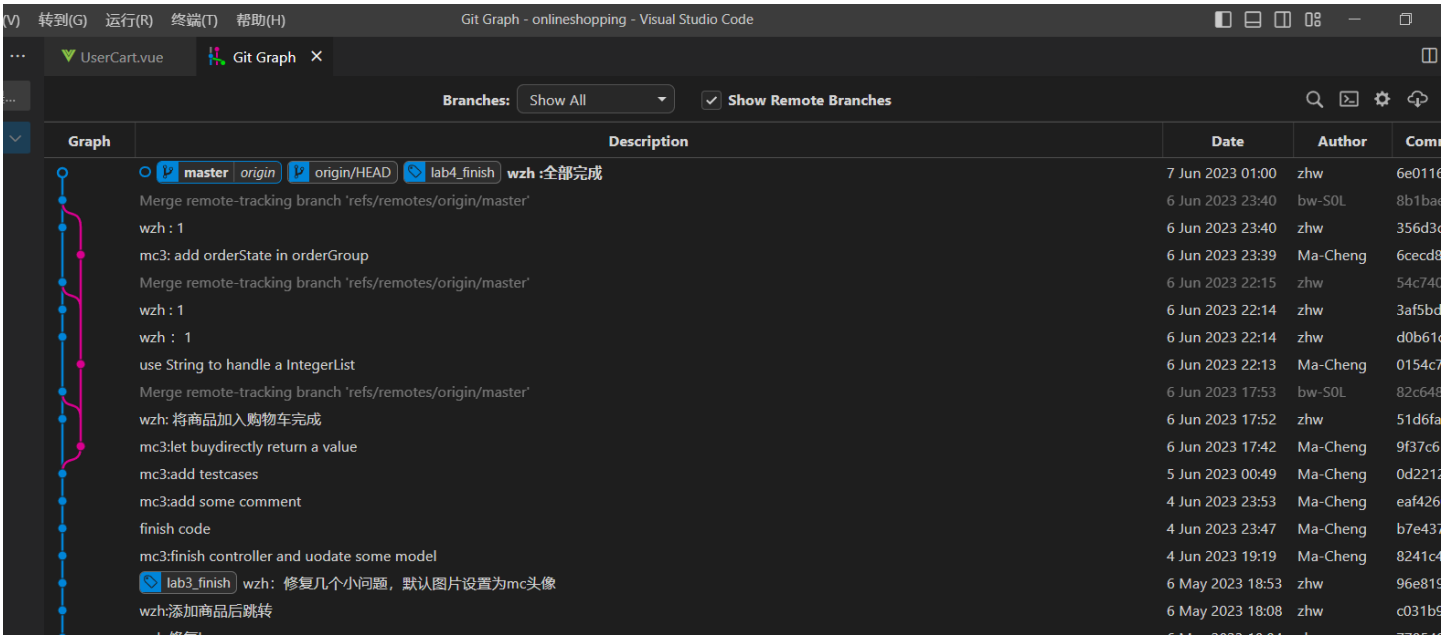
管理员	root	854b6cee1f4387fc1ec89427ef1	
非管理员	Wilson	97e3b086a573c3c6704ef36a43	

（具体的访问参数在测试部署的时候通过随机数生成）

- 测试结果

（一例失败为非管理员跨权访问，此时数据被后端拦截，无返回值，测试器判断为失败）

7 tag截图



8 心得体会

8.1 林琰钧

我负责的内容是：

- 团队组织：进行人员分工与任务安排、制定开发计划、定期召开线下会议讨论进度
- 文档记录：编写项目开发的约定与安排文档、lab开发进度文档
- 架构设计：设计数据库、前后端接口
- 代码编写：编写部分后端代码

我的心得体会是：

- 由于上次lab设计得很充分，因此这次基本没什么改变，只加了几个小API
- 这次很大一部分是文档编写的任务，按照书上的图绘制需求与设计的图，感觉还挺麻烦的

- 建议以后只布置2个lab，每个lab一个月（考虑到其他专业课还有lab，时间不应太短），不要在临近期末考时布置lab（大家都没时间做，最后赶ddl，体验不是很好）

8.2 马成

我负责的内容是：

- 数据库模型设计
- 代码编写：编写所有有关商店、商品、订单及他们申请纪录的增删改的代码
- 测试：编写数据库测试文件以及完成单元测试

我的心得体会是：

我这次承担了最后一次所有的后端代码，因为所有的改动都出现在上一次lab中我编写的部分中，由于上一次我们设计的比较充分并且代码也留下了很多添加功能的地方，这一次lab写的较为轻松，可以看出代码写的较为规范的好处。

最后还编写了一下白盒测试，由于很多较为复杂的Service代码逻辑都需要使用request中的token验证身份，而且使用JUNIT的时候我并不会伪造一个request因此很难测试那些逻辑复杂的Service，因此我只能测试一些不使用request同时也比较复杂的代码。

8.3 王兆瀚

我负责的内容是：

- 前后端接口设计
- 前端页面设计
- 代码编写：前端页面编写

我的心的体会是：

- 良好的设计可以减轻很多工作量
- 前端的很多问题是业务逻辑上的问题，而这往往会导致级联的代码问题
- 这一次的lab前后端分离比较好，可以不关注后端的细节，快速完成前端的开发
- lab的ddl不是很友好，最近考试比较多

8.4 李咸若

我负责的内容是：

- 黑盒测试

我的心得体会是：

- 预先一个lab做好了大部分的工作，所以这次基本没活，其实蛮爽的

- 测试需要全面的考虑问题，系统的设计流程
- 需要一些知道哪里会出问题的直觉和经验