|  |
| --- |
| S2023-B105 |
| 品读二手书交易平台 |
| 后台详细设计说明书 |

|  |
| --- |
| 乔轲 夏润成 李精文 王晗頔 王子骏  2023-5-9 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **《“品读”二手书交易平台后端详细设计说明书》** | | | | | |
| **编号** | S2023-QK105 | | | **版本** | 1.0 |
| **编制** | 乔轲 夏润成 王子骏 李精文 王晗頔 | **审核** | 乔轲 | **批准** | 乔轲 |
| **日期** | 2023.2.26 | **日期** | 2023.2.27 | **批准日期** | 2023.2.27 |

* 1. 目的

该详细设计说明书明确了后台管理员的分类，及其不同管理员拥有不同权限，并详细介绍了后台结构以及不同管理员登录后的功能。该说明书为设计本项目后台开发人员的详细参考，读者仅开发人员以及项目审核人员。

* 1. 项目背景

项目名称：读品-C2B2C图书交易平台。  
项目代号：S2023-QK105

项目来源：小组成员根据 “多抓鱼”图书平台的发展为灵感，共同商讨得出。

客户群体：读书爱好者

开发背景：本项目开发背景主要是基于以下四方面：

1. 图书装帧的日渐提升，全新图书的高昂价格已然成为大学生等学生群体的负担。且图书本具有良好的二手流通商品的属性。买进图书阅览学习，卖出图书回收资金。本项目的落地实际的解决了学生等低收入群体买卖图书的现实需求。
2. 年轻群体中有大量的文艺群体，这部分用户爱好读书，有大量的图书阅览需求，但并不会积累收藏很多图书。本网站实现图书的交换，吸引更多的图书爱好者，形成良性的以图书为媒介的社群文化。有众多且粘性强的用户群体。
3. 书香社会的创建，2022年“全民阅读”第九次被写入政府工作报告，全民阅读和书香社会建设已成为国家战略规划的重要部分。多读书读好书已然成为了社会新风，也成为了个人的强烈愿景。

四、绿水青山就是金山银山，双碳”战略倡导绿色、环保、低碳的生活方式。国家发展改革委发布《“十四五”循环经济发展规划》，提出了新的目标：到2025年，我国资源循环型产业体系基本建立，覆盖全社会的资源循环利用体系基本建成。符合国家总体发展战略。

* 1. 参考文献

需求说明书，立项任务书，软件开发计划书，后台概要设计说明书

《政府核准的投资项目目录》（2004年本）。

《企业投资项目核准暂行办法》，中华人民共和国国家发展和改革委员会令，（第19号）。

《国家发展改革委关于进一步加强和规范外商投资项目管理的通知》，发改外资[2008]1773 号。

《国务院办公厅关于加强和规范新开工项目管理的通知》，国办发[2007] 64号。

《国家标准软件开发文档规范》

《软件开发流程》，清华大学出版社，2005年1月版

* 1. 程序系统的结构

![图示

描述已自动生成]()

![图示

描述已自动生成]()

图示

描述已自动生成

图示

描述已自动生成

图示

描述已自动生成

图示

描述已自动生成

图示

描述已自动生成

5.接口设计

1. **5.1公共接口**
2. （1）用户的登录与注册
3. （2）管理员的登录
4. **5.2外部接口：**
5. pymysql：Django链接mysql的接口
6. ISBN接口：接口地址：https://api.gugudata.com/text/isbn
7. 豆瓣接口：获取豆瓣图书信息
8. **5.3内部接口：**
9. urlpatterns = [  
    path('admin/', admin.site.urls),  
     
    # people  
    path('api/userlist/', people.views.UserList.as\_view()),  
    path('api/userregister/', people.views.UserRegister.as\_view()),  
    path('api/userlogin/', people.views.UserLogin.as\_view()),  
    path('api/useraddress/', people.views.UserAddressListCreate.as\_view()),  
    # 下面的url：pk=uid并发送get，返回此人的所有记录；pk=useraddressid并发送patch,delete，修改、删除该条记录  
    path('api/useraddress/<int:pk>/', people.views.UserAddressRetrieveUpdateDestroy.as\_view()),  
    path('api/userdetail/<int:pk>/', people.views.UserDetail.as\_view()),  
    path('api/userchangepassword/', people.views.UserChangePw().as\_view()),  
    path('api/userverifypw/', people.views.UserPwVerify.as\_view()),  
    path('api/adminlist/', people.views.AdminListCreate.as\_view()),  
    path('api/adminlogin/', people.views.AdminLogin.as\_view()),  
     
    # book  
    path('api/book/', book.views.BookListCreate.as\_view()),  
    path('api/book/<int:isbn>/', book.views.BookRetrieveUpdateDestroy.as\_view()),  
    path('api/bookisflow/', book.views.BookIsFlowList.as\_view()),  
    # shop的精髓之处  
    path('api/bookcatepresspricesearch/<str:pk>/<str:qk>/<str:ok>/<str:rk>/<str:sk>/',  
    book.views.BookRetrieveByCatePressPriceSearch.as\_view()),  
    path('api/bookcate/<str:pk>/<str:qk>/<str:ok>/<str:rk>/<str:sk>/',  
    book.views.BookCategoryListByCatePressPriceSearch.as\_view()),  
    path('api/bookpress/<str:pk>/<str:qk>/<str:ok>/<str:rk>/<str:sk>/',  
    book.views.BookPresshouseListByCatePressPriceSearch.as\_view()),  
     
    # conditionrate  
    path('api/conditionrate/', book.views.ConditionRateListCreate.as\_view()),  
    path('api/conditionrate/<int:pk>/', book.views.ConditionRateRetreiveUpdateDelete.as\_view()),  
     
    # inventory  
    path('api/inventory/', book.views.InventoryListCreate.as\_view()),  
    path('api/inventory/<int:pk>/', book.views.InventoryRetrieveUpdateDestroy.as\_view()),  
    path('api/inventorydetail/<int:pk>/', book.views.InventoryDetailRetrieve.as\_view()),  
     
    # cart  
    path('api/cart/', buy.views.CartListCreate.as\_view()),  
    # get方法时pk为uid，patch、delete方法时pk为自增id  
    path('api/cart/<int:pk>/', buy.views.CartRetrieveUpdateDestory.as\_view()),  
    # 购物车减少一件  
    path('api/cartminus/', buy.views.CartMinus.as\_view()),  
    # 获得一个用户的一个inventory的数量  
    path('api/cartinventorynum/<int:pk>/<int:qk>/<str:ok>/', buy.views.CartInventoryNumRetrieve.as\_view()),  
     
    # sell  
    path('api/warehouse/', sell.views.WarehouseListCreate.as\_view()),  
    path('api/warehouse/<int:pk>/', sell.views.WarehouseRetrieveDestroy.as\_view()),  
    path('api/warehouse/uid=<int:pk>/', sell.views.WarehouseOfUserRetrieve.as\_view()),  
     
    # buy\_order  
    path('api/buyorder/', buy.views.BuyOrderListCreate.as\_view()),  
    path('api/buyorder/<int:pk>/', buy.views.BuyOrderRetrieveUpdateDelete.as\_view()),  
    # buyorderdetail  
    path('api/buyorderdetail/', buy.views.BuyOrderDetailListCreate.as\_view()),  
    path('api/buyorderdetail/<int:pk>/', buy.views.BuyOrderDetailRetrieveUpdateDestory.as\_view()),  
     
    # sellorder  
    path('api/sellorder/', sell.views.SellOrderListCreate.as\_view()),  
    path('api/sellorder/<int:pk>/', sell.views.SellOrderRetrieveUpdateDestroy.as\_view()),  
    path('api/sellorder/status=<int:pk>/', sell.views.SellOrderStatusList.as\_view()),  
    # sellorderdetail  
    path('api/sellorderdetail/', sell.views.SellOrderDetailListCreate.as\_view()),  
    path('api/sellorderdetail/<int:pk>/', sell.views.SellOrderDetailRetrieveUpdateDestroy.as\_view()),  
    path('api/sellorderdetailpatch/<int:pk>/', sell.views.SellOrderDetailPatch.as\_view()),  
     
    # feedback 这两个是for user  
    path('api/feedback/', feedback.views.FeedbackListCreate.as\_view()),  
    # # GET方法时pk=uid，返回此用户的所有历史问题，包括问题和回复及处理状态  
    path('api/feedback/<int:pk>/', feedback.views.FeedbackRetrieveUpdateDestroy.as\_view()),  
    path('api/feedbackpending/', feedback.views.FeedbackListPending.as\_view()),  
    path('api/feedbackpendingfromasset/', feedback.views.FeedbackListPendingFromAsset.as\_view()),  
    path('api/feedbackfromaftersale/', feedback.views.FeedbackListFromAftersale.as\_view()),  
    path('api/feedbackpendingfromaftersale/', feedback.views.FeedbackListPendingFromAftersale.as\_view()),

6．详细说明

**程序1订单查看模块（buyorderinfo/index.vue;saleorderinfo/index.vue）设计说明：**

1. 程序描述：

购售专员查看平台买/卖书订单。

2. 功能：当购售专员查看平台买/卖书订单时，要显示在页面中，并能点击按钮查看相信信息。

3. 性能：

该程序要求订单信息数据精准保存，反应灵敏且时间极短。

1. 输入项：

查询功能无输入项。

1. 输出项：

订单查看模块输出项

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 字段名 | 数据值的有效范围 | 输出  方式 | 输出位置 | 保密安全特性 |
| 买书订单编号 | bo\_id | int | 列表，界面 | 数据库中的buy\_order表 | 完整性、可用性、不可否认性 |
| 卖书订单编号 | so\_id | int | 列表，界面 | 数据库中的sell\_order表 | 完整性、不可否认性 |
| 用户id | u\_id | int | 列表，界面 | 数据库中的buy\_order/sell\_order表 | 完整性、不可否认性、可用性 |
| 买书订单状态 | Bo\_status | 订单状态(0-未审核,1-已审核 | 列表，界面 | 数据库中的buy\_order表 | 完整性、不可否认性 |
| 卖书订单状态 | So\_status | 订单状态(0-待处理，1-已发货，2-已收货,) | 列表，界面 | 数据库中的sell\_order表 | 整性、不可否认性 |
| 业务管理员id | Id | 1，2，3，4 | 单一字段，界面 | 页面穿参 | 完整性、不可否认性、可用性 |
| 买书ISBN | So\_id | Varchar(100) | 列表，界面 | Sell\_order\_detail表 | 完整性、不可否认性、可用性 |
| 买书  书名 | B\_name | Varchar(30) | 列表，界面 | 外键book表 | 完整性、不可否认性、可用性 |
| 用户定的品相 | W\_customer\_condition | Varchar(20) | 列表，界面 | Sell\_order\_detail表 | 完整性、不可否认性、可用性 |
| 平台鉴定品相 | Sod\_final\_condition | Varchar(20) | 列表，界面 | Sell\_order\_detail表 | 完整性、不可否认性、可用性 |
| 卖书  ISBN | bo\_id | Varchar(100) | 列表，界面 | Buy\_order\_detail表 | 完整性、不可否认性、可用性 |
| 卖书  书名 | B\_name | Varchar(30) | 列表，界面 | Buy\_order\_detail表 | 完整性、不可否认性、可用性 |
| 图书品相 | cr\_condition | Varchar(20) | 列表，界面 | Inventory表 | 完整性、不可否认性、可用性 |
| 购买数量 | I\_book\_num | Int | 列表，界面 | Inventory表 | 完整性、不可否认性、可用性 |

订单详细信息输出项

1. 算法

getsellor getsellorders().then((r) => {

for (let i = 0; i < r.data.length; i++) {

data.push({

id: r.data[i].id,

user\_id: r.data[i].user\_id,

status: r.data[i].status,

a\_id: 2,

})

}

ders().then((r) => {  
 for (let i = 0; i < r.data.length; i++) {  
 data.push({  
 id: r.data[i].id,  
 user\_id: r.data[i].user\_id,  
 status: r.data[i].status,  
 a\_id: 2,  
 })  
 }

7.流程逻辑：

1. 查询订单信息

查询平台卖书/买书订单信息

8.接口：

path('api/buyorder/', buy.views.BuyOrderListCreate.as\_view()),

path('api/buyorderdetail/', buy.views.BuyOrderDetailListCreate.as\_view()),

path('api/sellorder/', sell.views.SellOrderListCreate.as\_view()),

path('api/sellorderdetail/', sell.views.SellOrderDetailListCreate.as\_view()),

9. 存储分配：

本模块在Pycharm中进行编码，直接的内存分配由Pycharm运行时分配。

10. 注释设计：

加在模块首部的注释，包括版权信息、作者/修改者、完成日期；

对核心代码（连接数据库的代码）的简单注释；

11.限制条件：

无

12．测试计划：

（1） 技术要求：进行测试的人员必须懂得基本的数据库知识且熟知测试中常用的临界值等。

（2） 查看订单，查看订单详情：

点击页面触发路由，来到订单页面。点击button查看订单详情。

（3）测试结果：

1. 查询订单信息功能与查看订单详情的测试结果：

（4）进度安排：单个子模块完成后先做单体测试，之后对整个管理员信息管理模块进行测试，最后在整个系统完成之后，再进行系统测试。

（5）人员职责：用户信息管理模块测试由测试员完成，后期系统测试由6位组员共同完成。

（6）设备条件：一台PC机、Windows系统以及浏览器。

（7）驱动程序：无驱动程序要求。

13．尚未解决的问题：

本程序没有尚未解决的问题。

**程序2 购售专员发货模块（send/send.vue）设计说明：**

1. 程序描述：

对订单有查询与筛选功能。购售专员对用户购买的订单进行发货。在确认发货后，修改库存与订单状态。

2. 功能：

查看所有订单信息，查询订单功能，点击按钮进行发货并修改后台数据库。

3. 性能：

该程序要求订单信息数据精准保存，反应灵敏且时间极短。

1. 输入项：

购售专员发货查询输入项

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 名称 | 标识符 | 所采用的组件类型 | 数据值的有效范围 | 输入方式 | 输入数据的来源 | 保密安全特性 |
| order\_id | order\_id | 文本框 | varchar（30） | 键盘输入 | 管理员 | 唯一性、可用性 |
| 用户名 | User\_name | 文本框 | varchar（30） | 键盘输入 | 管理员 | 保密性、可用性 |
| uid | uid | 文本框 | varchar（30） | 键盘输入 | 管理员 | 保密性、可用性 |
| 省份 | region | 文本框 | varchar（30） | 键盘输入 | 管理员 | 保密性、可用性 |
| 地址 | place | 文本框 | varchar（30） | 键盘输入 | 管理员 | 保密性、可用性 |
| 联系方式 | phone | 文本框 | varchar（30） | 键盘输入 | 管理员 | 保密性、可用性 |
| 订单状态 | status | 文本框 | varchar（30） | 键盘输入 | 管理员 | 保密性、可用性 |

购售专员发货输出项

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 标识符 | 数据值的有效范围 | 输出数据的来源 | 保密安全特性 |
| 订单号 | \*bo\_id | varchar（30） | Sell\_order表 | 唯一性、可用性 |
| 用户id | uid | varchar（30） | Sell\_order表 | 保密性、可用性 |
| 订单状态 | Bo\_status | Varchar(30) | Sell\_order表 | 保密性、可用性 |
| 订单提交时间 | Bo\_submit\_time | Varchar(30) | Sell\_order\_detail表 | 保密性、可用性 |
| 订单结束时间 | Bo\_finish\_time | Varchar(30) | Sell\_order\_detail表 | 保密性、可用性 |
| 收件人 | receiver\_name | Varchar(30) | Sell\_order\_detail表 | 保密性、可用性 |
| 联系方式 | receiver\_phone | Varchar(30) | Sell\_order\_detail表 | 保密性、可用性 |
| 收件人地址 | receiver\_place | Varchar(30) | Sell\_order\_detail表 | 保密性、可用性 |
| 订单详情 | Sell\_order\_detail | Varchar(30) | Sell\_order\_detail表 | 保密性、可用性 |
| 库存id | Inventory\_id | Varchar(30) | Inventory表 | 保密性、可用性 |

1. 算法

// 提交发货订单操作  
const confirmEdit = async (index, row) => {  
  
 loading.value = true  
 const finish\_time = new *Date*()  
 *console*.log('row:', row)  
  
 row.detail\_inventory\_id.forEach((i) => {  
 *console*.log("inventory\_id", i)  
 loadInventoryNum(i).then(async (r) => {  
 *console*.log("数量:", r.data.book\_num)  
  
 if (r.data.book\_num < 1) {  
 alert("此书库存不足，无法发货")  
 loading.value = false  
 } else {  
  
 const new\_book\_num = r.data.book\_num - 1  
 await changeInventoryNum(i, {book\_num: new\_book\_num})  
  
 await changeBuyOrder(row.order\_id, {finish\_time: finish\_time})  
 changeBuyOrder(row.order\_id, {status: 1}).then((r) => {  
 data = []  
 getValue()  
 })  
 }  
 })  
 })  
}

6.流程逻辑：

管理员点击发货按钮后，修改后台库存，订单状态。

7. 接口：

path('api/buyorderdetail/<int:pk>/', buy.views.BuyOrderDetailRetrieveUpdateDestory.as\_view()),

path('api/inventory/<int:pk>/', book.views.InventoryRetrieveUpdateDestroy.as\_view()),  
path('api/inventorydetail/<int:pk>/', book.views.InventoryDetailRetrieve.as\_view()),

path('api/buyorder/<int:pk>/', buy.views.BuyOrderRetrieveUpdateDelete.as\_view()),

8. 存储分配：

本模块在Pycharm中进行编码，直接的内存分配由Pycharm运行时分配。

9.注释设计：

无

10．限制条件：

对管理员输入信息的格式有一定的要求。

11.测试计划：

1. 将不同字段输入对应搜索栏，点击筛选按钮
2. 与数据库比对筛选结果是否正确
3. 选择其中一项进行发货操作
4. 与数据库对比发货后是否减少库存并减少订单状态

12.尚未解决的问题：

本程序没有尚未解决的问题。

**程序3 修改价格比率（rules/index.vue）设计说明：**

1. 程序描述：

修改价格比率模块是管理书城中不同品相收书与卖书价格相比原价的比率。其中包括优良品相的平台买书比率、平台卖书比率；品相中等的平台卖书比率、平台卖书比率。

2. 功能：

该程序的功能是：管理员可以通过在不同input栏输入价格比率并进行更新数据库中此字段的比率。

3. 性能：

该程序要求订单信息数据精准保存，反应灵敏且时间极短。

1. 输入项：

修改价格比率功能输入项

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 名称 | 标识符 | 所采用的组件类型 | 数据值的有效范围 | 输入方式 | 输入数据的来源 | 保密安全特性 |
| 品相优良-平台买书 | cr\_rate | 输入框 | Float  0-1 | 键盘输入 | 购售专员 | 完整性、可用性 |
| 品相中等-平台卖书 | S\_rate | 输入框 | Float  0-1 | 键盘输入 | 购售专员 | 完整性、可用性 |
| 品相优良-平台买书 | cr\_rate | 输入框 | Float  0-1 | 键盘输入 | 购售专员 | 完整性、可用性 |
| 品相中等-平台卖书 | S\_rate | 输入框 | Float  0-1 | 键盘输入 | 购售专员 | 完整性、可用性 |

1. 输出项：

修改价格比率输出项

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 字段名 | 数据值的有效范围 | 输出  方式 | 输出位置 | 保密安全特性 |
| 品相优良-平台买书 | cr\_rate | Float  0-1 | 列表 | 数据库中的**mane\_regulation**  表 | 完整性、可用性、不可否认性 |
| 品相中等-平台卖书 | S\_rate | Float  0-1 | 列表 | 数据库中的**mane\_regulation**  表 | 完整性、不可否认性 |
| 品相优良-平台买书 | cr\_rate | Float  0-1 | 列表 | 数据库中的**mane\_regulation**  表 | 完整性、不可否认性 |
| 品相中等-平台卖书 | S\_rate | Float  0-1 | 列表 | 数据库中的**mane\_regulation**  表 | 整性、不可否认性 |

1. 算法

function onSubmit2(formInline) {  
 if (formInline.name3 != null && formInline.name4 != null) {  
 updaterate(1, formInline.name4, formInline.name3)  
 }  
 if (formInline.name3 == null && formInline.name4 != null) {  
 updatebuyrate(1, formInline.name4)  
 }  
 if (formInline.name3 != null && formInline.name4 == null) {  
 updatesellrate(1, formInline.name3)  
 }  
  
 formValue.value = formInline  
 *ElMessage*.success(*JSON*.stringify(formInline))  
}

1. 流程逻辑：

管理员找到想要修改的对应比率位置，填入数据，点击更新按钮更新数据库字段。修改期间可以点击充值清空输入框。

8.接口：

path('api/conditionrate/', book.views.ConditionRateListCreate.as\_view()),  
path('api/conditionrate/<int:pk>/', book.views.ConditionRateRetreiveUpdateDelete.as\_view()),

9. 存储分配：

本模块在Pycharm中进行编码，直接的内存分配由Pycharm运行时分配。

1. 注释设计：

加在模块首部的注释，包括版权信息、作者/修改者、完成日期；

对核心代码（连接数据库的代码）的简单注释；

11．限制条件：

对管理员输入的修改信息有一定的要求。

12．测试计划：

（1） 技术要求：进行测试的人员必须懂得基本的数据库知识且熟知测试中常用的临界值等。

（2） 输入数据：分别输入四个价格比率字段

（3）测试结果：将输入值与数据库中字段进行比对。

（4）进度安排：单个子模块完成后先做单体测试，之后对整个管理员信息管理模块进行测试，最后在整个系统完成之后，再进行系统测试。

（5）人员职责：不同测试流程由开发者以外的不同开发人员与测试用户进行测试。

（6）设备条件：一台PC机、Windows系统以及浏览器。

（7）驱动程序：无驱动程序要求。

13．尚未解决的问题：

本程序没有尚未解决的问题。

**程序4 订单审核模块（buysellmanOrderList/allOrdersbuysellmanOrderList/checkedOrders；buysellmanOrderList/checkWork；buysellmanOrderList/uncheckorders）设计说明：**

1. 程序描述：

程序为订单审核程序，包括以下几部分。

1. 全部订单信息：查看去全部用户卖书订单
2. 未审核订单：查看还未审核的订单，并可以点击审核查看订单详情，并逐一判断订单内图书品相
3. 已审核订单：查看全部已审核订单

2. 功能：

该程序的功能是：第一方面查看订单，包括全部、未审核、已审核。

第二方面是在未审核订单中审核订单内图书。

3. 性能：

该程序要求订单信息数据精准保存，反应灵敏且时间极短。

1. 输入项：

查询订单信息功能输入项

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 名称 | 标识符 | 所采用的组件类型 | 数据值的有效范围 | 输入方式 | 输入数据的来源 | 保密安全特性 |
| 判断品相 | Condition | Select | 选择其中属性 | 点击 | 购售专员 | 完整性、可用性 |

1. 输出项：

全部订单信息、未审核订单信息、已审核订单信息、审核订单

全部订单信息

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 字段名 | 数据值的有效范围 | 输出  方式 | 输出位置 | 保密安全特性 |
| 订单编号 | o\_id | int | 列表 | 数据库中的sell\_order表 | 完整性、可用性、不可否认性 |
| 订单状态 | o\_status | 订单状态(0-待审核；1-已审核) | 列表 | 数据库中sell\_order表 | 完整性、不可否认性 |
| 用户ID | uid | int | 列表 | 数据库中的sell\_order表 | 完整性、不可否认性、可用性 |
| 用户名称 | u\_name | Varchar(30) | 列表 | 数据库中的user表 | 完整性、不可否认性 |
| 日期 | Submit\_time | datetime（） | 列表 | 数据库中的sell\_order表 | 整性、不可否认性 |

未审核信息输出项

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 字段名 | 数据值的有效范围 | 输出  方式 | 输出位置 | 保密安全特性 |
| 订单编号 | o\_id | int | 列表 | 数据库中的sell\_order表 | 完整性、可用性、不可否认性 |
| 订单状态 | o\_status | 订单状态(0-待审核；1-已审核) | 列表 | 数据库中sell\_order表 | 完整性、不可否认性 |
| 用户ID | uid | int | 列表 | 数据库中的sell\_order表 | 完整性、不可否认性、可用性 |
| 用户名称 | u\_name | Varchar(30) | 列表 | 数据库中的user表 | 完整性、不可否认性 |
| 日期 | Submit\_time | datetime（） | 列表 | 数据库中的sell\_order表 | 整性、不可否认性 |

已审核信息输出项

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 字段名 | 数据值的有效范围 | 输出  方式 | 输出位置 | 保密安全特性 |
| 订单编号 | o\_id | int | 列表 | 数据库中的sell\_order表 | 完整性、可用性、不可否认性 |
| 订单状态 | o\_status | 订单状态(0-待审核；1-已审核) | 列表 | 数据库中sell\_order表 | 完整性、不可否认性 |
| 用户ID | uid | int | 列表 | 数据库中的sell\_order表 | 完整性、不可否认性、可用性 |
| 用户名称 | u\_name | Varchar(30) | 列表 | 数据库中的user表 | 完整性、不可否认性 |
| 订单开始日期 | Submit\_time | datetime（） | 列表 | 数据库中的sell\_order表 | 完整性、不可否认性 |
| 订单结束日期 | Finish\_time | datetime（） | 列表 | 数据库中的sell\_order表 | 完整性、不可否认性 |

审核订单输出项

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 字段名 | 数据值的有效范围 | 输出  方式 | 输出位置 | 保密安全特性 |
| 序号 | o\_id | int | 列表 | 数据库中的sell\_order-detail表 | 完整性、可用性、不可否认性 |
| 书名 | B\_name | Varchar(30) | 列表 | 数据库中的book表 | 完整性、可用性、不可否认性 |
| ISBN | so\_id | Varchar(100) | 列表 | 数据库中的sell\_order-detail表 | 完整性、可用性、不可否认性 |
| 作者 | B\_name | Varchar(30) | 列表 | 数据库中的book表 | 完整性、可用性、不可否认性 |
| 原价 | B\_price | Varchar(30) | 列表 | 数据库中的book表 | 完整性、可用性、不可否认性 |
| 用户预期品相 | W\_customer\_condition | Varchar(20) | 列表 | 数据库中的sell\_order-detail表 | 完整性、可用性、不可否认性 |
| 判断品相 | Sod\_final\_condition | Varchar(20) | 列表 | 数据库中的sell\_order-detail表 | 完整性、可用性、不可否认性 |
| 选择品相 | Condition——rate | Varchar(20) | 列表 | 程序写定 | 完整性、可用性、不可否认性 |

1. 算法

function submitForm() {  
 for (let i = 0; i < Form.length; i++) {  
 if (Form[i].final\_price == null) {  
 *ElNotification*({  
 title: '失败啦',  
 message: '订单有书目没有审核！',  
 type: 'error',  
 })  
 return 0  
 }  
 }  
 *console*.log(Form.length)  
 Form.forEach((e, i) => {  
 sellCheckCondition(e.id, {final\_condition: e.final\_condition, final\_price: e.final\_price}).then(r => {  
 *console*.log("rr:", r.data)  
 if (e.final\_condition != 5) {  
 addInventory({book: e.book\_isbn, condition\_rate: e.final\_condition, book\_num: 1})  
 }  
 })  
 })  
 // console.log("formforeach:", form2)  
 *console*.log(isoString)  
 const status = reactive({  
 status: 1,  
 finish\_time: isoString  
 })  
 *console*.log(status)  
 selldetailModify(note\_id, status)  
 modifyBalance(note\_user\_id, sum())  
 *ElNotification*({  
 title: '成功啦',  
 message: '订单审核工作结束！',  
 type: 'success',  
 })  
  
}

1. 流程逻辑：
2. 查询全部、未审核、已审核订单信息

点击左侧导航栏，右侧table显示信息。

1. 审核订单

点击审核button，页面跳转到审核订单详细信息。

在审核订单详细信息页面，点击判断品相，跳出对话框选择具体品相。

在所有书目的品相判断工作结束后点击提交按钮，跳出对话框提示是否选择提交，点击提交，完成审核流程。

1. 接口：

path('api/conditionrate/', book.views.ConditionRateListCreate.as\_view()),

path('api/inventorydetail/<int:pk>/', book.views.InventoryDetailRetrieve.as\_view()),

path('api/sellorderdetailpatch/<int:pk>/', sell.views.SellOrderDetailPatch.as\_view()),

path('api/sellorderdetail/<int:pk>/', sell.views.SellOrderDetailRetrieveUpdateDestroy.as\_view()),

path('api/sellorder/status=<int:pk>/', sell.views.SellOrderStatusList.as\_view()),

path('api/userdetail/<int:pk>/', people.views.UserDetail.as\_view()),

1. 存储分配：

本模块在Pycharm中进行编码，直接的内存分配由Pycharm运行时分配。

1. 注释设计：

加在模块首部的注释，包括版权信息、作者/修改者、完成日期；

对核心代码（连接数据库的代码）的简单注释；

11．限制条件：

对管理员输入的查询以及修改信息有一定的要求。

12．测试计划：

（1） 技术要求：进行测试的人员必须懂得基本的数据库知识且熟知测试中常用的临界值等。

（2） 输入数据：在订单审核页面逐一判断图书品相，在所有图书品相判断结束后，点击提交

（3）测试结果：修改一：订单状态由未审核变为已审核，并在已审核订单信息列表中可以看到，增加了订单完成时间。修改二：数据库中，相应图书的相应品相的库存有所增加。修改三：出售图书的用户钱包的金额数量发生变化。

（4）进度安排：单个子模块完成后先做单体测试，之后对整个管理员信息管理模块进行测试，最后在整个系统完成之后，再进行系统测试。

（5）人员职责：用户信息管理模块测试由测试员完成，后期系统测试由6位组员共同完成。

（6）设备条件：一台PC机、Windows系统以及浏览器。

（7）驱动程序：无驱动程序要求。

13．尚未解决的问题：

本程序没有尚未解决的问题。

**程序5图书管理模块**

1. 程序描述：本程序设计的目的主要为图书信息管理，其目的在于管理员容易管理前台数据，增加或修改图书信息，需要有较强的逻辑覆盖功能以及后台数据库修改、查找和删除的功能

2. 功能：管理员管理图书的修改、查找和删除。

3. 性能：本模块要求对数据字典进行维护。图书表中的图书isbn不可以有重复，所以在添加事要避免冲突信息。要求点击确定之后系统核查反映时间为1秒之内，并在0.5秒跳转回图书总览界面，如果图书库已有，则提示图书库已有书籍、添加失败。

4．功能详述

添加书目：

1. 输入项：图书isbn

2. 输出项：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 标识符 | 所采用的组件类型 | 数据值的有效范围 | 输出方式 | 保密安全特性 |
| 图书isbn | b\_isbn | 文本框 | (12,13] | 列表 | 完整性、保密性 |
| 书名 | b\_name | 文本框 | (0,200] | 列表 | 完整性、保密性 |
| 作者 | b\_writer | 文本框 | (0,20] | 列表 | 完整性、保密性 |
| 简介 | b\_intro | 文本框 | (0,1000] | 列表 | 完整性、保密性 |
| 类别 | b\_category | 文本框 | (0,100] | 列表 | 完整性、保密性 |
| 出版社 | b\_presshouse | 文本框 | (0,100] | 列表 | 完整性、保密性 |
| 出版时间 | b\_pressdate | 文本框 | (0,100] | 列表 | 完整性、保密性 |
| 图书价格 | b\_price | 文本框 | (0,20] | 列表 | 完整性、保密性 |
| 图书流通状态 | b\_flow | 文本框 | [0,1] | 列表 | 完整性、保密性 |
| 图书图片 | b\_img | 文本框 | (0,1000] | 列表 | 完整性、保密性 |

3. 算法：

function handleShowbookdetail() {

// 判断是否为13位数字的ISBN号

console.log(isbnInput.value, /^[0-9]{13}$/.test(isbnInput.value))

const is13DigitISBN = /^[0-9]{13}$/.test(isbnInput)

if (/^[0-9]{13}$/.test(isbnInput.value)) {

// 获取书籍详情等操作

showbookdetail()

show.value = true

} else {

show.value = false

alert('需要13位isbn号')

}

}

function showbookdetail() {

console.log(isbnInput.value)

getdbbook(isbnInput.value).then((res1) => {

console.log(res1)

if (res1.data.status) {

console.log('图书库里有')

bookinfo.intro = res1.data.intro

bookinfo.name = res1.data.name

bookinfo.translator = ''

bookinfo.writer = res1.data.writer

bookinfo.isbn = res1.data.isbn

bookinfo.category = res1.data.category

bookinfo.pressdate = res1.data.pressdate

bookinfo.presshouse = res1.data.presshouse

bookinfo.price = res1.data.price

bookinfo.picture = res1.data.picture

} else {

getoutbook(isbnInput.value).then((res2) => {

console.log('图书库里没有')

bookinfo.intro = res2.data.data.summary

bookinfo.name = res2.data.data.title

bookinfo.translator = res2.data.data.translator

bookinfo.writer = res2.data.data.author

bookinfo.isbn = res2.data.data.isbn

bookinfo.category = res2.data.data.category

bookinfo.pressdate = res2.data.data.pubdate

bookinfo.presshouse = res2.data.data.publisher

bookinfo.price = res2.data.data.price

bookinfo.picture = res2.data.data.img

})

}

})

show.value = true

console.log(show.value)

}

const insertbook = async () => {

// handleClose2(ruleFormRef2)

show.value = false

console.log(show.value)

dialogVisible2.value = false

console.log(bookinfo)

getdbbook(bookinfo.isbn).then((res) => {

console.log(bookinfo.isbn)

console.log(res)

console.log(res.data.isbn)

if (res.data.status) {

alert('图书库里已有')

} else {

console.log('图书库没有的添加isbn', res.data.isbn)

getoutbook(res.data.isbn).then(async (res2) => {

const bookinfo2 = reactive({

name: '',

isflow: null,

price: null,

writer: '',

intro: '',

presshouse: '',

isbn: '',

category: '',

pressdate: '',

translator: '',

picture: '',

})

bookinfo2.intro = res2.data.data.summary

bookinfo2.name = res2.data.data.title

bookinfo2.translator = res2.data.data.translator

bookinfo2.writer = res2.data.data.author

bookinfo2.isbn = res2.data.data.isbn

bookinfo2.category = res2.data.data.category

bookinfo2.pressdate = res2.data.data.pubdate

bookinfo2.presshouse = res2.data.data.publisher

bookinfo2.price = res2.data.data.price

bookinfo2.picture = res2.data.data.img

bookinfo2.isflow = true

bookinfo2.price = bookinfo2.price.toString().substring(0, 5)

console.log(bookinfo2)

await postbooktobook(bookinfo2)

await getdballbook().then((res) => {

const i = 1

console.log(res.data)

simpydata.value = res.data.map((item) => {

return {

pressdate: item['pressdate'],

name: item['name'],

price: item['price'],

isbn: item['isbn'],

presshouse: item['presshouse'],

// sex: i % 2 ? 1 : 0,

checked: true,

id: i + 1,

writer: item['writer'],

intro: item['intro'],

category: item['category'],

isflow: item['isflow'],

picture: item['picture'],

}

})

console.log(simpydata.value)

data.value = simpydata.value.map((item) => {

return {

pressdate: item['pressdate'],

name: item['name'],

price: item['price'],

isbn: item['isbn'],

presshouse: item['presshouse'],

// sex: i % 2 ? 1 : 0,

checked: true,

id: item['id'],

writer: item['writer'],

intro: item['intro'],

category: item['category'],

isflow: item['isflow'],

picture: item['picture'],

}

})

console.log('提交获取所有数据', data)

})

// location.reload()

})

// postbooktobook(bookinfo)

// postbooktobook(res.pra)

}

})

isbnInput.value = ''

bookinfo.name = ''

bookinfo.isflow = null

bookinfo.price = null

bookinfo.writer = ''

bookinfo.intro = ''

bookinfo.presshouse = ''

bookinfo.isbn = ''

bookinfo.category = ''

bookinfo.pressdate = ''

bookinfo.translator = ''

bookinfo.picture = ''

}

查询书目：

1. 输入项：图书isbn，书名，流通状态，出版社，图书类别，价格。（可选择性输入上述字段）

2. 输出项：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 标识符 | 所采用的组件类型 | 数据值的有效范围 | 输出方式 | 保密安全特性 |
| 图书isbn | b\_isbn | 文本框 | (12,13] | 列表 | 完整性、保密性 |
| 书名 | b\_name | 文本框 | (0,200] | 列表 | 完整性、保密性 |
| 作者 | b\_writer | 文本框 | (0,20] | 列表 | 完整性、保密性 |
| 简介 | b\_intro | 文本框 | (0,1000] | 列表 | 完整性、保密性 |
| 类别 | b\_category | 文本框 | (0,100] | 列表 | 完整性、保密性 |
| 出版社 | b\_presshouse | 文本框 | (0,100] | 列表 | 完整性、保密性 |
| 出版时间 | b\_pressdate | 文本框 | (0,100] | 列表 | 完整性、保密性 |
| 图书价格 | b\_price | 文本框 | (0,20] | 列表 | 完整性、保密性 |
| 图书流通状态 | b\_flow | 文本框 | [0,1] | 列表 | 完整性、保密性 |
| 图书图片 | b\_img | 文本框 | (0,1000] | 列表 | 完整性、保密性 |

3.算法

const onSubmit = async (val) => {

await getdballbook().then((res) => {

const i = 1

console.log(res.data)

simpydata.value = res.data.map((item) => {

return {

pressdate: item['pressdate'],

name: item['name'],

price: item['price'],

isbn: item['isbn'],

presshouse: item['presshouse'],

// sex: i % 2 ? 1 : 0,

checked: true,

id: i + 1,

writer: item['writer'],

intro: item['intro'],

category: item['category'],

isflow: item['isflow'],

picture: item['picture'],

}

})

console.log(simpydata.value)

data.value = simpydata.value.map((item) => {

return {

pressdate: item['pressdate'],

name: item['name'],

price: item['price'],

isbn: item['isbn'],

presshouse: item['presshouse'],

// sex: i % 2 ? 1 : 0,

checked: true,

id: item['id'],

writer: item['writer'],

intro: item['intro'],

category: item['category'],

isflow: item['isflow'],

picture: item['picture'],

}

})

console.log('提交获取所有数据', data)

})

console.log('val===', val)

console.log(typeof val.name)

console.log(typeof data.value[0].name)

console.log(val.name)

console.log(data.value[0].name)

data.value = data.value.filter((item) => {

let result = true

if (val.name && String(item.name) !== String(val.name)) {

result = false

}

if (val.isbn && String(item.isbn) !== String(val.isbn)) {

result = false

}

if (val.isflow && item.isflow !== val.isflow) {

result = false

}

if (val.price && String(item.price) !== String(val.price)) {

result = false

}

if (val.category && String(item.category) !== String(val.category)) {

result = false

}

if (val.presshouse && String(item.presshouse) !== String(val.presshouse)) {

result = false

}

return result

})

console.log('筛选数据', data)

ElMessage.success('触发查询方法')

loading.value = true

setTimeout(() => {

loading.value = false

}, 500)

}

查询书目：

1. 输入项：无

2. 输出项：无（仅提示删除成功）

3.算法

const del = (row) => {

console.log('row==', row)

ElMessageBox.confirm('你确定要删除当前项吗?', '温馨提示', {

confirmButtonText: '确定',

cancelButtonText: '取消',

type: 'warning',

draggable: true,

}).then(() => {

console.log(list.value, '删除')

list.value = list.value.filter((item) => item.isbn !== row.isbn)

ElMessage.success('删除成功')

console.log(list.value)

deldbbook(row.isbn)

loading.value = true

setTimeout(() => {

loading.value = false

}, 500)

})

// .catch(() => {})

}

4.流程逻辑：

图示

描述已自动生成

5. 接口：

path('api/book/', book.views.BookListCreate.as\_view()),

path('api/book/<int:isbn>/', book.views.BookRetrieveUpdateDestroy.as\_view()),

path('api/bookisflow/', book.views.BookIsFlowList.as\_view()),

const instance: AxiosInstance = axios.create({

baseURL: 'https://api.ibook.tech/v1/book',

timeout: 5000,

headers: {

'Content-Type': 'application/json',

},

})

export function getoutbook(isbn: string): Promise<any> {

console.log('获取外部', isbn)

return outApi.get(`isbn?isbn=${isbn}&uKey=33b4c461da6946eca5c7fae235c5da9e`)

}

6. 存储分配：本模块在Pycharm中进行编码，直接的内存分配由Pycharm运行时分配。

7. 注释设计：详见本功能模块算法。

8. 限制条件：添加书目和查找信息时，isbn必须13位，否则弹框提示，且不允许添加图书库已有书目，否则弹框提示。

12. 测试计划：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模块名称 | 技术要求 | 人员安排 | 人员职责 | 输入数据 | 进度安排 |
| 后台获取所有图书信息 | 熟悉django\JavaScript\mysql\vue以及测试技术 | 李精文 | 测试，并填写相应记录 | 详见测试用例设计 | 2023.5.7前完成 |
| 后台分类查询图书信息 | 熟悉django\JavaScript\mysql\vue以及测试技术 | 李精文 | 测试，并填写相应记录 | 详见测试用例设计 | 2023.5.3前完成 |
| 后台添加图书信息 | 熟悉django\JavaScript\mysql\vue以及测试技术 | 李精文 | 测试，并填写相应记录 | 详见测试用例设计 | 2023.5.8前完成 |
| 后台删除图书信息 | 熟悉django\JavaScript\mysql\vue以及测试技术 | 李精文 | 测试，并填写相应记录 | 详见测试用例设计 | 2023.5.8前完成 |
| 后台修改图书信息 | 熟悉django\JavaScript\mysql\vue以及测试技术 | 李精文 | 测试，并填写相应记录 | 详见测试用例设计 | 2023.5.8前完成 |

9. 尚未解决的问题：无

**程序6售后管理说明**

1. 程序描述：本程序设计的目的主要为售后管理，其目的在于管理员容易管理前台用户提出的问题数据，反馈问题或向资金专员提出问题，需要有较强的逻辑覆盖功能以及后台数据库修改、查找的功能

2. 功能：管理员管理售后的查找订单问题、反馈用户问题和向资金专员提出问题。

3. 性能：本模块要求对数据字典进行维护。要求点击确定回复消息或向资金专员发出问题之后系统核查反映是否成功，要求在时间为1秒之内，并在0.5秒跳转回售后未回复界面，要求在未回复页面删除刚刚已回复的订单问题。

1. 输入项：反馈用户的回答，向资金专员提出的问题

1. 输出项：

售后所有问题列表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 标识符 | 所采用的组件类型 | 数据值的有效范围 | 输出方式 | 保密安全特性 |
| 问题id | fba2u\_id | 文本框 | (0,10] | 列表 | 完整性、保密性 |
| 用户名 | $u\_name | 文本框 | (0,50] | 列表 | 完整性、保密性 |
| 处理进度：  0:未处理,  1:售后反馈资金,  2:资金反馈售后,  3:售后反馈用户 | fba2u\_condition | 文本框 | (0,20] | 列表 | 完整性、保密性 |
| 用户问题 | fba2u\_content\_u2as | 文本框 | (0,1000] | 列表 | 完整性、保密性 |

售后待处理问题列表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 标识符 | 所采用的组件类型 | 数据值的有效范围 | 输出方式 | 保密安全特性 |
| 问题id | fba2u\_id | 文本框 | (0,10] | 列表 | 完整性、保密性 |
| 售后人员id | $as\_id | 文本框 | (0,10] | 列表 | 完整性、保密性 |
| 用户名 | $u\_name | 文本框 | (0,50] | 列表 | 完整性、保密性 |
| 处理进度：  待处理 | fba2u\_condition | 文本框 | (0,20] | 列表 | 完整性、保密性 |
| 用户问题 | fba2u\_content\_u2as | 文本框 | (0,1000] | 列表 | 完整性、保密性 |

售后待反馈问题列表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 标识符 | 所采用的组件类型 | 数据值的有效范围 | 输出方式 | 保密安全特性 |
| 问题id | fba2u\_id | 文本框 | (0,10] | 列表 | 完整性、保密性 |
| 用户id | $u\_id | 文本框 | (0,10] | 列表 | 完整性、保密性 |
| 售后人员id | $as\_id | 文本框 | (0,50] | 列表 | 完整性、保密性 |
| 处理进度：  0:未处理,  1:售后反馈资金,  2:资金反馈售后,  3:售后反馈用户 | fba2u\_condition | 文本框 | (0,20] | 列表 | 完整性、保密性 |
| 用户问题 | fba2u\_content\_u2as | 文本框 | (0,1000] | 列表 | 完整性、保密性 |
| 售后给用户反馈内容 | fba2u\_content\_as2u | 文本框 | (0,1000] | 列表 | 完整性、保密性 |
| 售后给资金专员的问题 | fba2u\_content\_as2a | 文本框 | (0,1000] | 列表 | 完整性、保密性 |

1. 算法

onBeforeMount(() => {

feedbackList().then((r) => {

r.data.forEach((e, i) => {

data.push(e)

})

data.forEach((e, i) => {

e.user = e.user\_name + '(' + e.user\_id + ')'

if (e.aftersaleman\_id === null) {

e.aftersaleman = '-'

} else {

e.aftersaleman = e.aftersaleman\_name + '(' + e.aftersaleman\_id + ')'

}

if (e.assetman\_id === null) {

e.assetman = '-'

} else {

e.assetman = e.assetman\_name + '(' + e.assetman\_id + ')'

}

if (e.que\_condition == 0) {

e.que\_condition\_text = '待处理'

} else if (e.que\_condition == 1) {

e.que\_condition\_text = '待资金回复'

} else if (e.que\_condition == 2) {

e.que\_condition\_text = '待反馈'

} else if (e.que\_condition == 3) {

e.que\_condition\_text = '已完成'

}

})

console.log(data)

load.value = true

})

})

onBeforeMount(() => {

feedbackListPendingFromAsset().then((r) => {

r.data.forEach((e, i) => {

data.push(e)

})

data.forEach((e, i) => {

e.user = e.user\_name + '(' + e.user\_id + ')'

if (e.aftersaleman\_id === null) {

e.aftersaleman = '-'

} else {

e.aftersaleman = e.aftersaleman\_name + '(' + e.aftersaleman\_id + ')'

}

if (e.assetman\_id === null) {

e.assetman = '-'

} else {

e.assetman = e.assetman\_name + '(' + e.assetman\_id + ')'

}

if (e.que\_condition == 0) {

e.que\_condition\_text = '待处理'

} else if (e.que\_condition == 1) {

e.que\_condition\_text = '待资金回复'

} else if (e.que\_condition == 2) {

e.que\_condition\_text = '待反馈'

} else if (e.que\_condition == 3) {

e.que\_condition\_text = '已完成'

}

})

console.log(data)

load.value = true

})

})

onBeforeMount(() => {

feedbackListPending().then((r) => {

r.data.forEach((e, i) => {

data.push(e)

})

data.forEach((e, i) => {

e.user = e.user\_name + '(' + e.user\_id + ')'

if (e.aftersaleman\_id === null) {

e.aftersaleman = '-'

} else {

e.aftersaleman = e.aftersaleman\_name + '(' + e.aftersaleman\_id + ')'

}

if (e.assetman\_id === null) {

e.assetman = '-'

} else {

e.assetman = e.assetman\_name + '(' + e.assetman\_id + ')'

}

if (e.que\_condition == 0) {

e.que\_condition\_text = '待处理'

} else if (e.que\_condition == 1) {

e.que\_condition\_text = '待资金回复'

} else if (e.que\_condition == 2) {

e.que\_condition\_text = '待反馈'

} else if (e.que\_condition == 3) {

e.que\_condition\_text = '已完成'

}

})

console.log(data)

load.value = true

})

})

const handleClose = async (done: () => void) => {

await ruleFormRef.value.validate((valid, fields) => {

if (valid) {

let obj = {

...ruleForm,

}

list.value.forEach((item) => {

if (item.id === rowObj.value.id) {

// 如果回复用户

if (ruleForm.direction === '1') {

feedbackUpdate(item.id, {

que\_content\_aftersaleman2user: obj.que\_content\_deal,

que\_condition: 3,

aftersaleman: UserStore.id,

}).then((r) => {

console.log(r.data)

})

list.value = list.value.filter((item) => item.id !== rowObj.value.id)

ElMessage.success('回复用户成功啦')

loading.value = true

setTimeout(() => {

loading.value = false

}, 500)

// location.reload()

// 如果联系资金专员

} else {

const body = {

que\_content\_aftersaleman2assetman: obj.que\_content\_deal,

que\_condition: 1,

aftersaleman: UserStore.id,

}

console.log(body)

feedbackUpdate(item.id, body).then((r) => console.log(r.data))

list.value = list.value.filter((item) => item.id !== rowObj.value.id)

ElMessage.success('咨询资金专员成功啦')

loading.value = true

setTimeout(() => {

loading.value = false

}, 500)

// location.reload()

}

}

})

dialogVisible.value = false

console.log('submit!', obj)

} else {

console.log('error submit!', fields)

}

})

}

1. 流程逻辑图

图示

描述已自动生成

1. 接口：

path('api/feedback/', feedback.views.FeedbackListCreate.as\_view()),

path('api/feedback/<int:pk>/', feedback.views.FeedbackRetrieveUpdateDestroy.as\_view()),

path('api/feedbackpending/', feedback.views.FeedbackListPending.as\_view()),

path('api/feedbackpendingfromasset/', feedback.views.FeedbackListPendingFromAsset.as\_view()),

1. 存储分配：本模块在Pycharm中进行编码，直接的内存分配由Pycharm运行时分配。
2. 注释设计：

准备安排在本程序的注释有：按照代码模块化进行各模块化的注释，阐释清楚每个模块的主要功能，同时对于一些格式化设置，可变参数的地方进行进一步注明。

限制条件：只有该管理员才能对订单问题进行反馈，及其对反馈状态进行修改，并且在反馈问题时，必须选择是发送给资金专员还是用户。而且不可以回复栏为空就点击回复按钮

11．模块测试

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模块名称 | 技术要求 | 人员安排 | 人员职责 | 输入数据 | 进度安排 |
| 后台获取所有问题 | 熟悉django\JavaScript\mysql\vue以及测试技术 | 夏润成 | 测试，并填写相应记录 | 详见测试用例设计 | 2023.5.7前完成 |
| 后台获取待处理问题 | 熟悉django\JavaScript\mysql\vue以及测试技术 | 夏润成 | 测试，并填写相应记录 | 详见测试用例设计 | 2023.5.3前完成 |
| 后台获取待反馈问题 | 熟悉django\JavaScript\mysql\vue以及测试技术 | 夏润成 | 测试，并填写相应记录 | 详见测试用例设计 | 2023.5.8前完成 |
| 反馈给用户或提交问题给资金专员 | 熟悉django\JavaScript\mysql\vue以及测试技术 | 夏润成 | 测试，并填写相应记录 | 详见测试用例设计 | 2023.5.8前完成 |

1. 尚未解决的问题：本程序无尚未解决问题。

**程序7资金反馈说明**

1. 程序描述：本程序设计的目的主要为资金反馈，其目的在于管理员容易管理从售后发来的问题

2. 功能：资金管理员管理售后发来的订单问题、反馈给售后回答。

3. 性能：本模块要求对数据字典进行维护。要求点击确定回复消息或向资金专员发出问题之后系统核查反映是否成功，要求在时间为1秒之内，并在0.5秒跳转回售后未回复界面，要求在未回复页面删除刚刚已回复的订单问题。

1. 输入项：回复售后的问题
2. 输出项：

资金所有问题列表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 标识符 | 所采用的组件类型 | 数据值的有效范围 | 输出方式 | 保密安全特性 |
| 问题id | fba2as\_id | 文本框 | (0,10] | 列表 | 完整性、保密性 |
| 售后人员名字 | $as\_name | 文本框 | (0,50] | 列表 | 完整性、保密性 |
| 资金人员名字 | $a\_name | 文本框 | (0,20] | 列表 | 完整性、保密性 |
| 资金人员给售后人员的反馈 | fba2u\_content\_a2as | 文本框 | (0,1000] | 列表 | 完整性、保密性 |
| 问题状态 | 处理进度：,  1:售后反馈资金,  2:资金反馈售后, | 文本框 | (0,10] | 列表 | 完整性、保密性 |

资金待处理问题列表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 标识符 | 所采用的组件类型 | 数据值的有效范围 | 输出方式 | 保密安全特性 |
| 问题id | fba2as\_id | 文本框 | (0,10] | 列表 | 完整性、保密性 |
| 售后人员名字 | $as\_name | 文本框 | (0,50] | 列表 | 完整性、保密性 |
| 资金人员名字 | $a\_name | 文本框 | (0,20] | 列表 | 完整性、保密性 |
| 资金人员给售后人员的反馈 | fba2u\_content\_a2as | 文本框 | (0,1000] | 列表 | 完整性、保密性 |
| 问题状态 | 处理进度：,  1:售后反馈资金, | 文本框 | (0,10] | 列表 | 完整性、保密性 |

1. 算法

onBeforeMount(() => {

feedbackListPendingFromAftersale().then((r) => {

r.data.forEach((e, i) => {

data.push(e)

})

data.forEach((e, i) => {

e.user = e.user\_name + '(' + e.user\_id + ')'

if (e.aftersaleman\_id === null) {

e.aftersaleman = '-'

} else {

e.aftersaleman = e.aftersaleman\_name + '(' + e.aftersaleman\_id + ')'

}

if (e.assetman\_id === null) {

e.assetman = '-'

} else {

e.assetman = e.assetman\_name + '(' + e.assetman\_id + ')'

}

if (e.que\_condition == 0) {

e.que\_condition\_text = '待处理'

} else if (e.que\_condition == 1) {

e.que\_condition\_text = '待回复'

} else if (e.que\_condition == 2) {

e.que\_condition\_text = '已回复'

} else if (e.que\_condition == 3) {

e.que\_condition\_text = '已完成'

}

})

console.log(data)

load.value = true

})

})

const handleClose = async (done: () => void) => {

await ruleFormRef.value.validate((valid, fields) => {

if (valid) {

let obj = {

...ruleForm,

}

list.value.forEach((item) => {

if (item.id === rowObj.value.id) {

feedbackUpdate(item.id, {

que\_content\_assetman2aftersaleman: obj.que\_content\_assetman2aftersaleman,

que\_condition: 2,

assetman: UserStore.id,

}).then((r) => {

console.log(r.data)

})

// location.reload()

list.value = list.value.filter((item) => item.id !== rowObj.value.id)

ElMessage.success('回复成功啦')

loading.value = true

setTimeout(() => {

loading.value = false

}, 500)

}

})

dialogVisible.value = false

console.log('submit!', obj)

} else {

console.log('error submit!', fields)

}

})

}

1. 流程逻辑图

图示

描述已自动生成

1. 接口：

path('api/feedbackfromaftersale/', feedback.views.FeedbackListFromAftersale.as\_view()),

path('api/feedbackpendingfromaftersale/', feedback.views.FeedbackListPendingFromAftersale.as\_view())}

1. 存储分配：本模块在Pycharm中进行编码，直接的内存分配由Pycharm运行时分配。
2. 注释设计：

准备安排在本程序的注释有：按照代码模块化进行各模块化的注释，阐释清楚每个模块的主要功能，同时对于一些格式化设置，可变参数的地方进行进一步注明。

限制条件：不可以回复栏为空就点击回复按钮

13．模块测试

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模块名称 | 技术要求 | 人员安排 | 人员职责 | 输入数据 | 进度安排 |
| 获取售后发来的所有问题 | 熟悉django\JavaScript\mysql\vue以及测试技术 | 夏润成 | 测试，并填写相应记录 | 详见测试用例设计 | 2023.5.7前完成 |
| 获取资金专员待处理问题 | 熟悉django\JavaScript\mysql\vue以及测试技术 | 夏润成 | 测试，并填写相应记录 | 详见测试用例设计 | 2023.5.3前完成 |

14.尚未解决的问题：本程序无尚未解决问题。