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

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **《“品读”二手书交易平台前端详细设计说明书》** | | | | | |
| **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

该文档在概要设计的基础上，进一步的细化系统结构，展示平台的结构概况，数据结构设计，及算法设计，详细的介绍了系统各个模块是如何实现的，包括涉及到的算法，逻辑流程等。为下一步系统的实现和测试做了一定的准备。

* 1. 项目背景

项目名称：“品读”二手书交易平台；

目的:实现C2B2C模式下的二手书交易，以及后端系统的正常运行；

用户：需要出售、购买二手书的读者朋友；

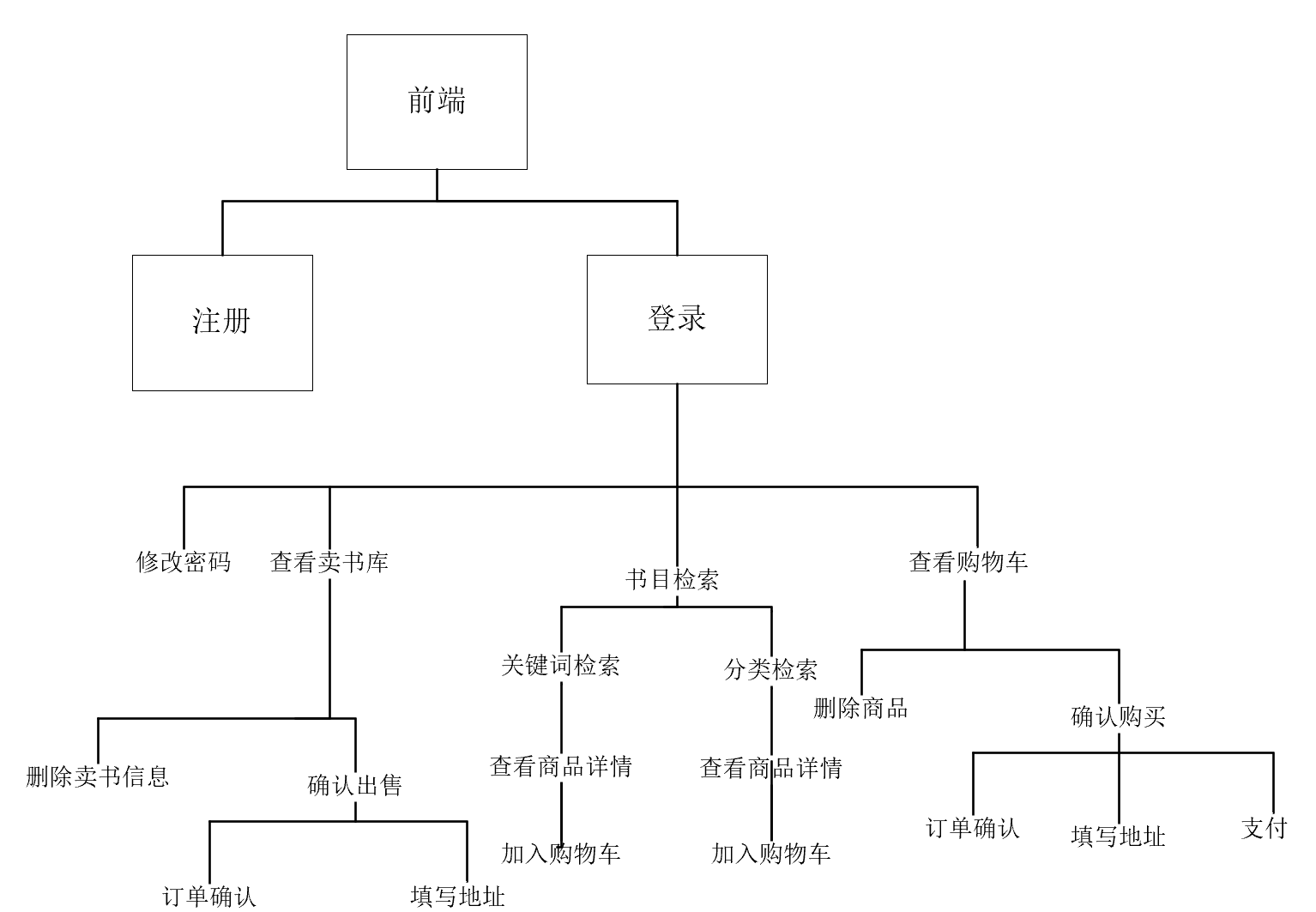
实现该软件的计算中心或计算机网络：小组成员的PC端连接gitee工作台，同时外接腾讯云服务器；

开发者：乔柯、夏润成、王晗頔、李精文、王子骏。

* 1. 参考文献

《立项任务书》、《软件开发计划》、《前端概要设计说明书》、《前端需求说明书》

4.程序系统的结构



5.公共接口

1. 外部接口：pymysql：Django链接mysql的接口
2. ISBN接口：接口地址：https://api.gugudata.com/text/isbn

6.详细说明

**6.1.主页设计说明：**

**6.1.主页：**

1．程序描述：主页界面显示，header部分有跳转链接（商店、购物车、卖书库、登录）；主体部分是热销书目推荐、“猜你喜欢”书目推荐。本程序旨在支持本平台前台的搜索、跳转链接、以及推荐等功能，是用户登录后首先进入的界面，这对于整个平台来说起到了一个统筹聚合的作用。

2．功能：

|  |  |  |
| --- | --- | --- |
| 功能 | 输入/动作 | 输出 |
| 点击跳转功能 | 点击头部标签（商店/购物车/卖书库）；点击主页面中的书目即可跳转至书目信息界面 | 跳转至相应界面 |
| 热销推荐功能 | 进入主页 | 按照用户搜索历史推荐不同种类书 |
| 登录/注册 | 点击登录/注册按钮 | 登录账号后的个性化主页 |

3．性能： ①精度：输入数据的精度：小数点后保留两位有效数字  
输出数据精度：小数点后保留两位有效数字  
传输过程中的精度：小数点后保留两位有效数字

②灵活性：  
适用于WINDOWS、MACOS操作系统

③时间特性：  
热销显示：0.5S以内  
界面跳转：1s以内  
搜索书目：1s以内

分类搜索：1s以内

4．输入项：无

5．输出项：

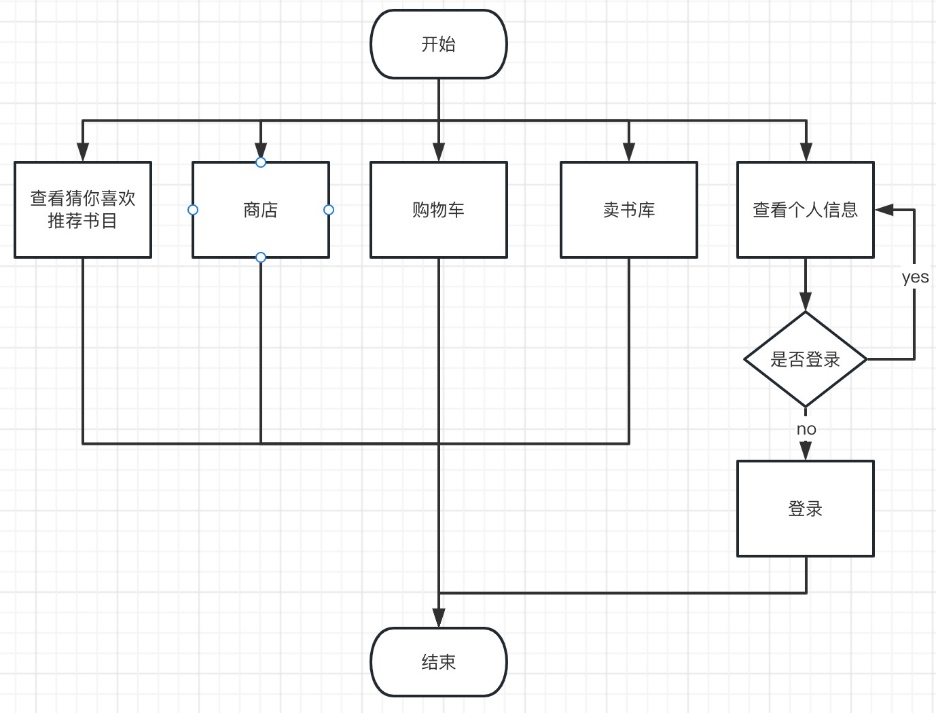
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 标识 | 数据类型 | 取值范围 | 输出方式 |
| 图书名 | B\_name | Varchar(100) | >0 | 自动 |
| 简介 | B\_intro | Varchar(1000) | >0 | 自动 |

6．主要代码：

①跳转链接：

<template>  
 <div id="t">  
 <n-space align="center" justify="space-between">  
 <n-space align="center">  
 <RouterLink to="/" class="routerlink">  
 <n-button text size="large" style="width: 55px;height: 55px;padding-left: 10px;padding-right: 10px">  
 <img src="../assets/logo.png" alt="logo" width="55" height="55" style="padding-top: 7px">  
 </n-button>  
 </RouterLink>  
 <RouterLink to="/shop/" class="routerlink">  
 <n-button text color="white" size="medium" style="font-size: large;font-weight: bolder;width: 60px"><p>  
 商店</p>  
 </n-button>  
 </RouterLink>  
 </n-space>  
   
 <n-space align="center" style="padding-right: 20px">  
 <RouterLink to="/cart" class="routerlink">  
 <n-button text size="large" style="width: 45px; height: 45px; padding-left: 0">  
 <img src="../assets/购物车.png" alt="购物车" width="42" height="42">  
 </n-button>  
 </RouterLink>  
 <RouterLink to="/warehouse" class="routerlink">  
 <n-button text size="large" style="width: 45px; height: 45px; padding-left: 0">  
 <img src="../assets/卖书柜.png" alt="卖书柜" width="38" height="38">  
 </n-button>  
 </RouterLink>  
  
 <RouterLink to="/login/" class="routerlink">  
 <n-button  
 v-if="islogin"  
 size="medium"  
 style="color: white; background-color: #d53f68">  
 登陆/注册  
 </n-button>  
 </RouterLink>  
  
 <template v-if="islogin === false">  
 <RouterLink to="/information/self/">  
 <n-button  
 size="medium"  
 style="color: white; background-color: #d53f68">  
 <p style="font-weight: bold">你好，{{ nickname }}</p>  
 </n-button>  
 </RouterLink>  
 <n-button class="logout" quaternary @click="logout" type="default" size="tiny"  
 style="color: #f2f2f2; background-color: #282828;">  
 登出  
 </n-button>  
 </template>  
 </n-space>  
 </n-space>  
 </div>  
</template>

7．流程逻辑：



8．接口：无

9．存储分配：该系统使用的数据库MySQL，已满足对数据的存储和读取功能。

10．注释设计：①加在各分支点处的注释：用于解释各分支的功能  
②对各变量的功能、范围、缺省条件等所加的注释：具体说明变量的功能，取值范围等其他情况

11．限制条件：系统：Windows10以上系统/MACOS系统  
软件：装有pycharm，webstorm软件并且启动MYSQL，运行npm run dev，启用django；装有edge或chrome

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 |
| 主页信息显示 | 熟悉html，vue，mysql，JavaScript，django以及测试技术 | 夏润成 | 测试，并填写相应记录 | 无 | 2023.5.3前完成 |
| 主页显示“猜你喜欢”推荐书目 | 熟悉html，vue，mysql，JavaScript，django以及测试技术 | 夏润成 | 测试，并填写相应记录 | 无 | 2023.5.3前完成 |
| 主页跳转界面 | 熟悉html，vue，mysql，JavaScript，django以及测试技术 | 夏润成 | 测试，并填写相应记录 | 无 | 2023.5.3前完成 |

13．关键接口功能代码：

export default {  
 Register(username, password, nickname) {  
 return api.post('/userregister/', {username: username, password: password, nickname: nickname})  
 },  
  
 Login(username, password) {  
 return api.post('/userlogin/', {username: username, password: password})  
 },  
  
 GetUserInformation(user\_id) {  
 return api.get(`/userdetail/${user\_id}`)  
 },  
}

**6.2登录/注册页面：**

1．程序描述：在主页NaviBar内点击“登录/注册”，即可进入登录/注册页面，如果用户已有账号即可直接输入账号和密码进行登录。如果没有账户则选择注册，按照注册指导教程完成注册。

2．功能：说明该程序应具有的功能（采用文字的或图表的）。

|  |  |  |
| --- | --- | --- |
| 功能 | 动作/条件 | 输出 |
| 注册 | 点击注册 | 无 |
| 登录 | 输入用户名、密码，点击登录 | 无 |

3．性能：①精度：输入数据的精度：小数点后保留两位有效数字  
输出数据精度：小数点后保留两位有效数字  
传输过程中的精度：小数点后保留两位有效数字

②灵活性：  
适用于WINDOWS、MACOS操作系统

③时间特性：

界面跳转：1s以内

登录：0.5s以内

注册：0.5s以内

4．输入项：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 标识 | 数据类型 | 取值范围 | 输入方式 |
| 用户名 | U\_name | VARCHAR(100) | 5-20 | 手动 |
| 密码 | U\_pwd | VARCHAR(256) | 6-20 | 手动 |
| 昵称 | U\_nickname | VARCHAR(100) | >0 | 手动 |

5．输出项：成功登陆后→主页/成功注册后→登录页面。

6．主要代码：详细说明本程序所采用的主要代码，具体的计算公式及步骤等。

登录：

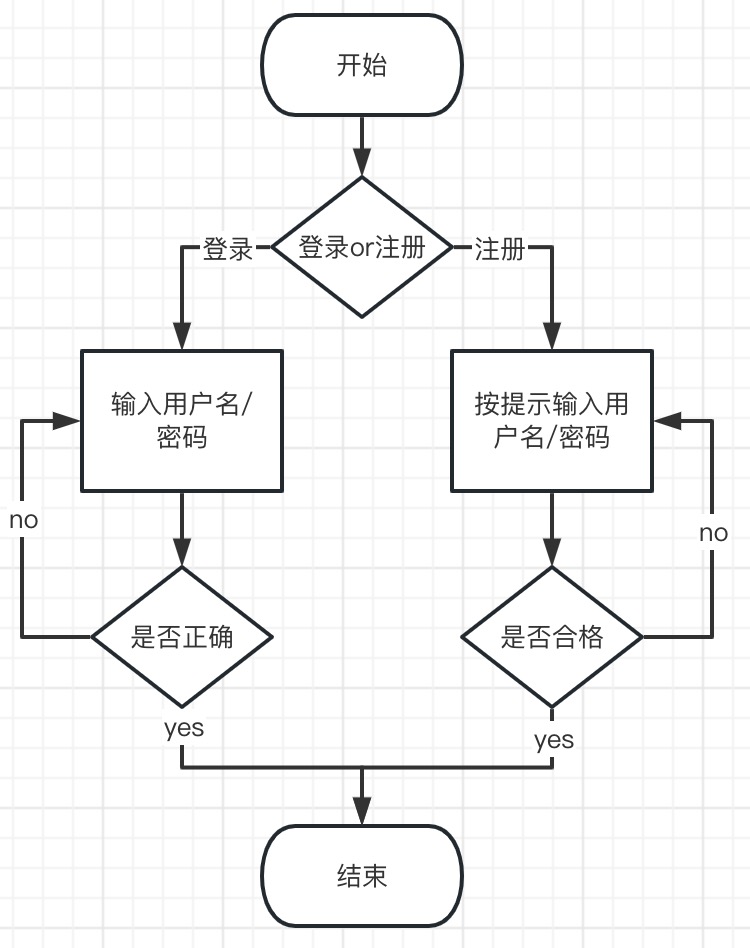
<template>  
 <div style="width: 100%; height: 90%;display: flex;align-items: center;justify-content: center;">  
  
 <div style="">  
 <form style="float: right">  
 <h3>账号：</h3>  
 <input type="text" v-model="***username***">  
 <h3>密码：</h3>  
 <input type="password" v-model="***password***">  
 <br>  
  
 <div style="display: flex; align-items: center; justify-content: space-between; padding-top: 10px">  
 <RouterLink :to="{name: 'register'}" style="color: lightsteelblue; float: left; text-decoration-line: none">  
 注册  
 </RouterLink>  
 <button type="button" @click="login">登陆</button>  
 </div>  
 </form>  
 </div>  
  
 </div>  
  
</template>

注册：

<template>  
 <div style="height: 95%; display: flex; align-items: center; justify-content: center">  
  
 <form style="float: right">  
 <h3>账号：</h3>  
 <input type="text" v-model="***username***">  
 <h3>昵称：</h3>  
 <input type="text" v-model="***nickname***">  
 <h3>密码：</h3>  
 <input type="password" v-model="***password***">  
 <br>  
  
 <div style="display: flex; align-items: center; justify-content: space-between; padding-top: 10px">  
 <RouterLink to="/login/" style="color: white; text-decoration-line: none">  
 返回登录  
 </RouterLink>  
 <button type="button" @click="register">注册</button>  
 </div>  
 </form>  
 </div>  
</template>

const ***username*** = ref("")  
const ***nickname*** = ref("")  
const ***password*** = ref("")  
  
const register = () => {  
 ***console***.log(***username***.value)  
 ***console***.log(***password***.value)  
  
 //校验登录名：只能输入5-20个以字母开头、可带数字的字串  
 const pat = /^[a-zA-Z]([a-zA-Z0-9]){4,19}$/;  
 //校验密码：只能输入6-20个字母、数字、下划线  
 const patrn = /^(\w){6,20}$/  
 const r = pat.test(***username***.value);  
 const rpw = patrn.test(***password***.value)  
  
 ***console***.log(r)  
 ***console***.log(rpw)  
 if (r != null && rpw != null) {  
 if (r && rpw) {  
 // 账密合格  
 loginService.Register(***username***.value, ***password***.value, ***nickname***.value).then(r => {  
 if (r.data.status === true) {  
 alert("注册成功，请登录！")  
 ***router***.push('/login/')  
 } else {  
 // 账号已注册  
 ***username***.value = ''  
 alert("账号已注册，请更换！")  
 }  
 })  
 } else {  
 // 账密格式不对  
 alert("账号或密码格式不对。\n账号：5-20位以字母开头、可带数字\n密码：6-20位字母、数字、下划线")  
 }  
 } else {  
 //账密为空  
 alert("账号密码不能为空")  
 }  
  
}

7．流程逻辑：



8．接口：

path('api/userregister/', people.views.UserRegister.as\_view()),  
path('api/userlogin/', people.views.UserLogin.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： 加在各分支点处的注释；对各变量的功能、范围、缺省条件等所加的注释。

11．限制条件：

系统：Windows10以上系统/MACOS系统

软件：装有phpStudy软件并且启动MYSQL以及Apache功能；装有浏览器

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 |
| 登录 | 熟悉html，vue，mysql，JavaScript，django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 | 2022.4.30前完成 |
| 注册 | 熟悉html，vue，mysql，JavaScript，django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 | 2022.5.3前完成 |

13．关键接口功能代码：

export default {  
 Register(username, password, nickname) {  
 return api.post('/userregister/', {username: username, password: password, nickname: nickname})  
 },  
  
 Login(username, password) {  
 return api.post('/userlogin/', {username: username, password: password})  
 },  
  
 GetUserInformation(user\_id) {  
 return api.get(`/userdetail/${user\_id}`)  
 },  
}

**6.3购物车设计说明：**

1．程序描述：在主页内点击“我的购物车”，即可通过用户ID进入该用户的购物车，并能对购物车内的商品进行删除，提交订单等操作，实现二手书的购买。本界面需要能调整购买书的数量，界面简洁明了，反映速度较快。

2．功能：说明该程序应具有的功能（采用文字的或图表的）。

|  |  |  |
| --- | --- | --- |
| 功能 | 动作/条件 | 输出 |
| 购物车界面展示 | 该用户购物车内无内容 | “暂无数据”界面 |
| 该用户购物车内有内容 | 该用户购物车内的内容 |
| 删除购物车内书籍 | 点击该书籍前的删除 | 删除该书籍，更新界面 |
| 选择商品提交订单 | 点击书籍前面的多选按钮  （书豆余额大于书籍总金额） | 在确认订单界面显示所选书籍 |
| 点击书籍前面的多选按钮  （书豆余额小于书籍总金额） | 提示“余额不足”，并返回购物车界面 |
| 点击确认支付 | 选择了商品并提交了订单 | 提示“支付成功”，更新数据库，返回更新后的购物车界面 |

3．性能：①精度：输入数据的精度：小数点后保留两位有效数字  
输出数据精度：小数点后保留两位有效数字  
传输过程中的精度：小数点后保留两位有效数字

②灵活性：  
适用于WINDOWS、MACOS操作系统

③时间特性：

界面跳转：5s以内

4．输入项：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 标识 | 数据类型 | 取值范围 | 输入方式 |
| 购物车 | Bookcart | VARCHAR(MYSQL) | >=0 | 自动查询 |
| 提交订单 | Submit | VARCHAR(MYSQL) | >=0 | 自动获取 |
| 填写地址 | Address | VARCHAR(MYSQL) | >=0 | 手动输入 |
| 提交反馈 | Feedback | VARCHAR(MYSQL) | >=0 | 手动输入 |

6．算法：详细说明本程序所采用的算法，具体的计算公式及步骤等。

显示购物车：

mounted() {  
 getCartbook(***window***.sessionStorage.getItem("id")).then((r) => {  
 ***console***.log("数据库的", r.data.length)  
 if (r.data.length) {  
 ***console***.log("数组不为空", this.cartData)  
 this.inventry = r.data.map(s => {  
 return {  
 id: s["inventory"],  
 total: s["book\_num"],  
 cartid: s["id"]  
 };  
 })  
 ***console***.log("传过来的inventry", this.inventry)  
 let idex = 0;  
 let fin\_price = 0;  
 this.inventry.forEach(item => {  
 ***console***.log("穿过来的inventry的id", item.id)  
 getCartbookdetail(item.id).then(res => {  
 ***console***.log("idex为", idex)  
 ***console***.log("传过来的书目具体信息", res.data)  
 ***console***.log(res.data.book\_price)  
 ***console***.log(res.data.condition\_rate\_customerrate)  
 fin\_price = res.data.book\_price \* res.data.condition\_rate\_customerrate  
 ***console***.log(fin\_price)  
 if (idex === 0) {  
 ***console***.log("idex为0的传入")  
 this.cartData[idex].cartid = item.cartid  
 this.cartData[idex].price = fin\_price.toFixed(2)  
 this.cartData[idex].id = res.data.id  
 this.cartData[idex].isbn = res.data.book\_isbn  
 this.cartData[idex].name = res.data.book\_name  
 this.cartData[idex].img = res.data.book\_pic  
 this.cartData[idex].pinxiang = res.data.condition\_rate\_name  
 this.cartData[idex].quantity = item.total  
 this.cartData[idex].author = res.data.book\_writer  
 } else {  
 ***console***.log("idex不为0", idex)  
 this.cartData.push({  
 cartid: item.cartid,  
 price: fin\_price.toFixed(2),  
 id: res.data.id,  
 isbn: res.data.book\_isbn,  
 name: res.data.book\_name,  
 img: res.data.book\_pic,  
 pinxiang: res.data.condition\_rate\_name,  
 quantity: item.total,  
 author: res.data.book\_writer  
 })  
 }  
 ***console***.log("数据库的", r.data.length)  
 if (r.data.length) {  
 ***console***.log("数组不为空", this.cartData)  
 this.inventry = r.data.map(s => {  
 return {  
 id: s["inventory"],  
 total: s["book\_num"],  
 cartid: s["id"]  
 };  
 })  
 ***console***.log("传过来的inventry", this.inventry)  
 let idex = 0;  
 let fin\_price = 0;  
 this.inventry.forEach(item => {  
 ***console***.log("穿过来的inventry的id", item.id)  
 getCartbookdetail(item.id).then(res => {  
 ***console***.log("idex为", idex)  
 ***console***.log("传过来的书目具体信息", res.data)  
 ***console***.log(res.data.book\_price)  
 ***console***.log(res.data.condition\_rate\_customerrate)  
 fin\_price = res.data.book\_price \* res.data.condition\_rate\_customerrate  
 ***console***.log(fin\_price)  
 if (idex === 0) {  
 ***console***.log("idex为0的传入")  
 this.cartData[idex].cartid = item.cartid  
 this.cartData[idex].price = fin\_price.toFixed()  
 this.cartData[idex].id = res.data.id  
 this.cartData[idex].isbn = res.data.book\_isbn  
 this.cartData[idex].name = res.data.book\_name  
 this.cartData[idex].img = res.data.book\_pic  
 this.cartData[idex].pinxiang = res.data.condition\_rate\_name  
 this.cartData[idex].quantity = item.total  
 this.cartData[idex].author = res.data.book\_writer  
 } else {  
 ***console***.log("idex不为0", idex)  
 this.cartData.push({  
 cartid: item.cartid,  
 price: fin\_price.toFixed(2),  
 id: res.data.id,  
 isbn: res.data.book\_isbn,  
 name: res.data.book\_name,  
 img: res.data.book\_pic,  
 pinxiang: res.data.condition\_rate\_name,  
 quantity: item.total,  
 author: res.data.book\_writer  
 })  
 }  
 ***console***.log("cartData第idex数据", idex, this.cartData[idex])  
 ***console***.log("cartData所有数据", this.cartData[idex])  
 idex = idex + 1  
 })  
 })  
 ***console***.log(this.cartData)  
 } else {  
 ***console***.log("数组为空", this.cartData)  
 this.cartData.pop()  
 ***console***.log(this.cartData)  
 }  
 });  
 })  
 }  
 })

购物车内计算总金额、总买书数量、是否选中所有。

computed: {  
 totalPrice() {  
 let total = 0;  
 this.cartData.forEach((item) => {  
 if (item.isSelected) {  
 total += item.price \* item.quantity;  
 }  
 });  
 return total.toFixed(2);  
 },  
 totalQuantity() {  
 let total = 0;  
 this.cartData.forEach((item) => {  
 if (item.isSelected) {  
 total += item.quantity;  
 }  
 });  
 return total;  
 },  
 isSelectedAll: {  
 get() {  
 let result = true;  
 let t = 0;  
 this.cartData.forEach((item) => {  
 if (!item.isSelected) {  
 result = false;  
 }  
 });  
 return result;  
 },  
 set(value) {  
 this.cartData.forEach((item) => {  
 item.isSelected = value;  
 });  
 },  
 },  
},

删除购物车内的书

removeSelectedProducts() {  
 const deltodb = this.cartData.filter((item) => item.isSelected)  
 ***console***.log("要删除的数据", deltodb)  
 this.delId = deltodb.map(s => {  
 return {  
 id: s["cartid"],  
 };  
 })  
 ***console***.log("要删除的id", this.delId)  
 this.delId.forEach((item) => {  
 ***console***.log("准备删除的id为", item.id)  
 deleteCartBook(item.id)  
 })  
 this.cartData = this.cartData.filter((item) => !item.isSelected);  
 },  
 },  
}

加减购物车内商品数量

addQuantity(index) {  
 this.cartData[index].quantity++;  
 ***console***.log(this.cartData[index].id)  
 patchbooktoCart(***window***.sessionStorage.getItem("id"), this.cartData[index].isbn, this.cartData[index].pinxiang)  
},  
reduceQuantity(index) {  
 if (this.cartData[index].quantity > 1) {  
 this.cartData[index].quantity--;  
 ***console***.log(this.cartData[index])  
 patchreducebooktoCart(***window***.sessionStorage.getItem("id"), this.cartData[index].id)  
 }  
},

购物车结算界面：

methods: {  
 handleSell() {  
 const selectedItems = this.cartData.filter(item => item.isSelected);  
 if (selectedItems.length === 0) {  
 alert('请至少选择一个商品');  
 } else {  
 this.$router.push({  
 name: 'purchasecheck',  
 query: {myArray: ***JSON***.stringify(selectedItems)}  
 });  
 }  
 },

>

填写地址：

function addAddress() {  
 if (form.name === '' || form.phone === '' || form.region === '' || form.place === '') {  
 ***console***.log("error")  
 } else {  
 const data = {  
 user\_id: ***window***.sessionStorage.getItem("id"),  
 receiver\_name: form.name,  
 receiver\_phone: form.phone,  
 receiver\_region: form.region,  
 receiver\_place: form.place  
 }  
 ***console***.log(data)  
 addUserAddress(data)  
 dialogFormVisible.value = false;  
 ***location***.reload()  
 }  
}

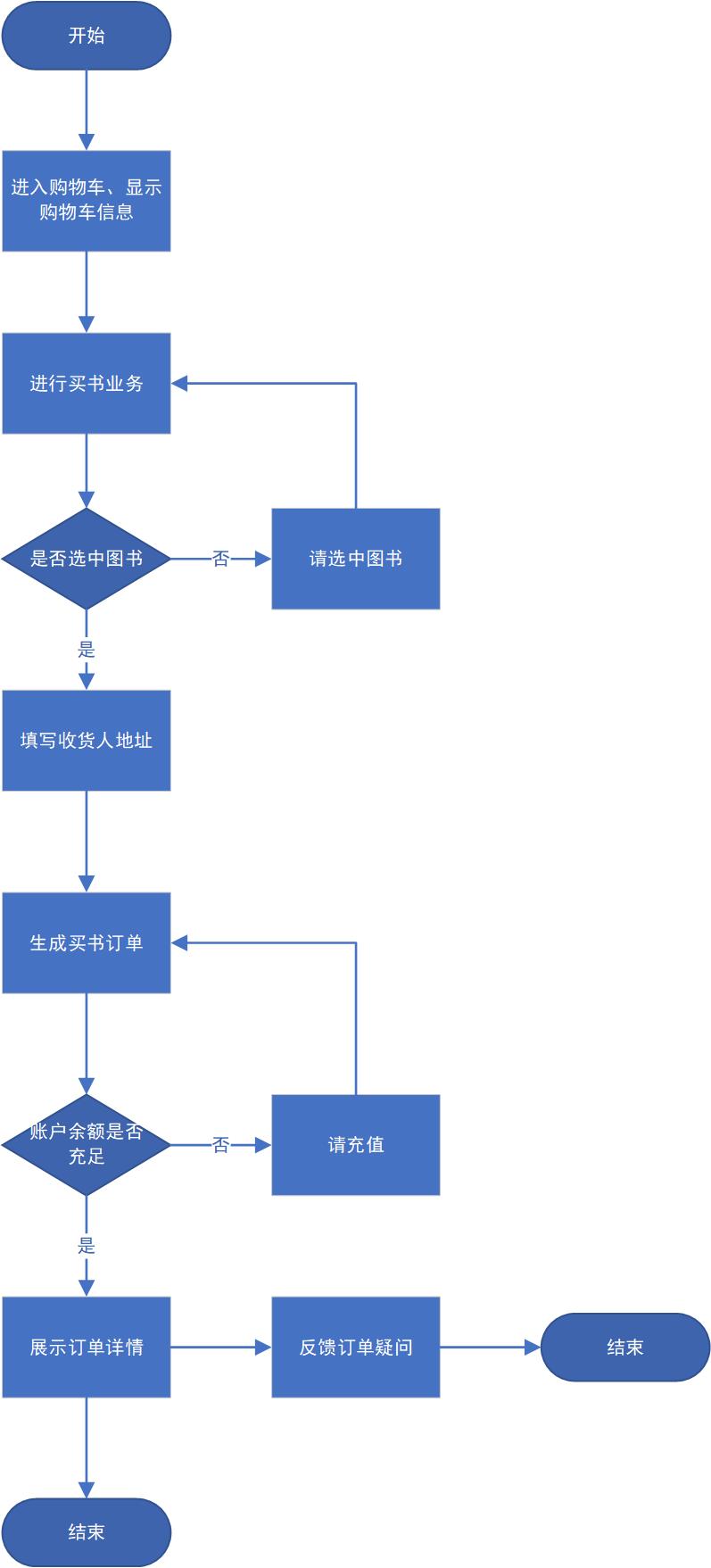
提交订单：

function isAbleToBuy() {  
 const user\_id = ***window***.sessionStorage.getItem("id")  
 userDetail(user\_id).then(resp => {  
 ***console***.log(Number(resp.data.balance), Number(totalprice.value))  
 if (Number(resp.data.balance) < Number(totalprice.value)) {  
 alert("余额不足")  
 } else {  
 addBuyOrderDetail()  
 }  
 })  
}

联系客服

const open1 = () => {  
 const uid = ***window***.sessionStorage.getItem("id")  
 feedback(uid, question.value)  
 feedbackdialog.value = false  
 if (question.value === '') {  
 ***ElNotification***({  
 title: '失败啦',  
 message: '请在输入栏中输入您的问题！',  
 type: 'error',  
 })  
 } else  
 ***ElNotification***({  
 title: '成功啦',  
 message: '已将您的意见发给专员，将在1-3个工作日内回复您！',  
 type: 'success',  
 })  
}

7.流程图：



8．接口：

# 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()),

# 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()),

# 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())

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： 加在各分支点处的注释；对各变量的功能、范围、缺省条件等所加的注释。

11．限制条件：

系统：Windows10以上系统/MACOS系统

软件：装有phpStudy软件并且启动MYSQL以及Apache功能；装有浏览器

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 |
| 查看购物车模块 | 熟悉vue\JavaScript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 | 2022.4.30前完成 |
| 购物车选商品确认订单模块 | 熟悉vue\JavaScript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 | 2022.5.2前完成 |
| 填写订单信息，立即购买模块 | 熟悉vue\JavaScript以及测试技术 | 王晗頔 | 测试，并填写相应记录 | 详见测试用例设计 | 2022.5.3前完成 |
| 查看订单详情，联系客服模块 | 熟悉vue\JavaScript以及测试技术 | 王子骏 | 测试，并填写相应记录 | 详见测试用例设计 | 2022.5.4前完成 |

13．尚未解决的问题：暂无

**6.4卖书设计说明：**

1. 程序描述：

（1）根据uid显示卖书库，用户可以输入isbn向卖书库中添加图书，选择品相后显示卖书金额，确认卖书后卖书库中书消失，生成卖书订单。

（2）确认卖书后生成填写订单地址，提交订单后显示卖书订单信息，可以进行反馈。

2．功能：

|  |  |  |  |
| --- | --- | --- | --- |
| 功能 | 动作 | 条件 | 输出 |
| 卖书库添加图书 | 输入ISBN向卖书库添加图书信息 | ISBN真实存在 | 卖书库增加书目 |
| 选择书目品相 | 书目品相合格 | 输出对于该书的预估售价 |
| 确认卖书 | 点击“确认卖书”按钮 | 已选中相关图书 | 跳转到填写订单信息界面 |
| 填写地址 | 选择填写地址或新建地址 | 已确认卖书 | 允许生成订单 |
| 生成订单 | 点击“生成订单”按钮 | 已填写地址 | 在数据库中生成订单，输出订单详细信息 |
| 提交反馈 | 填写反馈问题，点击“提交反馈”按钮 | 已生成订单 | 提交反馈成功，可在个人通知界面查看客服回复和反馈处理进度。 |

3．性能：

①精度：输入数据的精度：小数点后保留两位有效数字

输出数据精度：小数点后保留两位有效数字  
 传输过程中的精度：小数点后保留两位有效数字

②灵活性：  
 适用于WINDOWS、MACOS操作系统

③时间特性：  
 界面跳转：5s以内

4．输入项：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 标识 | 数据类型 | 取值范围 | 输入方式 |
| 购物车 | Bookcart | VARCHAR(MYSQL) | >=0 | 自动查询 |
| 提交订单 | Submit | VARCHAR(MYSQL) | >=0 | 自动获取 |
| 填写地址 | Address | VARCHAR(MYSQL) | >=0 | 手动输入 |
| 提交反馈 | Feedback | VARCHAR(MYSQL) | >=0 | 手动输入 |

5．输出项：alert

6．算法：

根据isbn加入图书

findbookinfo() {  
if (this.inputtext.length !== 13 || !/^\d{13}$/.test(this.inputtext)) {  
 alert("ISBN号应为13位数字，请重新输入！");  
 } else{  
 ***console***.log(this.inputtext);  
 this.centerDialogVisible = true  
 ***console***.log("发啊啊啊啊啊啊啊")  
 //fetchbook↓  
 getdbbook(this.inputtext).then(res => {  
 ***console***.log("触发啊啊啊啊啊啊啊")  
 ***console***.log(res);  
 let res1 = res;  
 if (res1.data.status) {  
 ***console***.log("图书库里有")  
 ***console***.log(res1.data)  
 this.bookinfo.summary = res1.data.intro;  
 this.bookinfo.title = res1.data.name;  
 this.bookinfo.translator = '';  
 this.bookinfo.author = res1.data.writer;  
 this.bookinfo.isbn = res1.data.isbn;  
 this.bookinfo.category = res1.data.category;  
 this.bookinfo.pubdate = res1.data.pressdate;  
 this.bookinfo.publisher = res1.data.presshouse;  
 this.bookinfo.price = res1.data.price;  
 this.bookinfo.img = res1.data.picture;  
 } else {  
 getoutbook(this.inputtext).then(res2 => {  
 ***console***.log("图书库里没有")  
  
 this.bookinfo.summary = res2.data.data.summary;  
 this.bookinfo.title = res2.data.data.title;  
 this.bookinfo.translator = res2.data.data.translator;  
 this.bookinfo.author = res2.data.data.author;  
 this.bookinfo.isbn = res2.data.data.isbn;  
 this.bookinfo.category = res2.data.data.category;  
 this.bookinfo.pubdate = res2.data.data.pubdate;  
 this.bookinfo.publisher = res2.data.data.publisher;  
 this.bookinfo.price = res2.data.data.price;  
 this.bookinfo.img = res2.data.data.img;  
  
 this.booktodb.name = this.bookinfo.title;  
 this.booktodb.writer = this.bookinfo.author  
 this.booktodb.isbn = this.bookinfo.isbn  
 this.booktodb.category = this.bookinfo.category  
 this.booktodb.pressdate = this.bookinfo.pubdate  
 this.booktodb.presshouse = this.bookinfo.publisher  
 this.booktodb.picture = this.bookinfo.img  
 this.booktodb.price = this.bookinfo.price.toString().substring(0, 5)  
 this.booktodb.intro = this.bookinfo.summary  
 this.booktodb.customer\_condition = this.pinxiang  
 this.booktodb.user\_id = ***window***.sessionStorage.getItem("id")  
  
 ***console***.log("price todb:", this.booktodb.price)  
 ***console***.log("price info:", this.bookinfo.price)  
 postbooktobook(this.booktodb)  
  
 })  
 }  
 })  
  
 }  
 }

确认卖书：

handleSell() {  
 const selectedItems = this.cartData.filter(item => item.isSelected);  
 if (selectedItems.length === 0) {  
 alert('请至少选择一个商品');  
 } else {  
 this.$router.push({  
 name: 'sellcheck',  
 query: {myArray: ***JSON***.stringify(selectedItems)}  
 });  
 }  
}

填写地址

function addAddress() {  
 if (form.name === '' || form.phone === '' || form.region === '' || form.place === '') {  
 ***console***.log("error")  
 } else {  
 const data = {  
 user\_id: ***window***.sessionStorage.getItem("id"),  
 receiver\_name: form.name,  
 receiver\_phone: form.phone,  
 receiver\_region: form.region,  
 receiver\_place: form.place  
 }  
 ***console***.log(data)  
 addUserAddress(data)  
 dialogFormVisible.value = false;  
 ***location***.reload()  
 }  
}

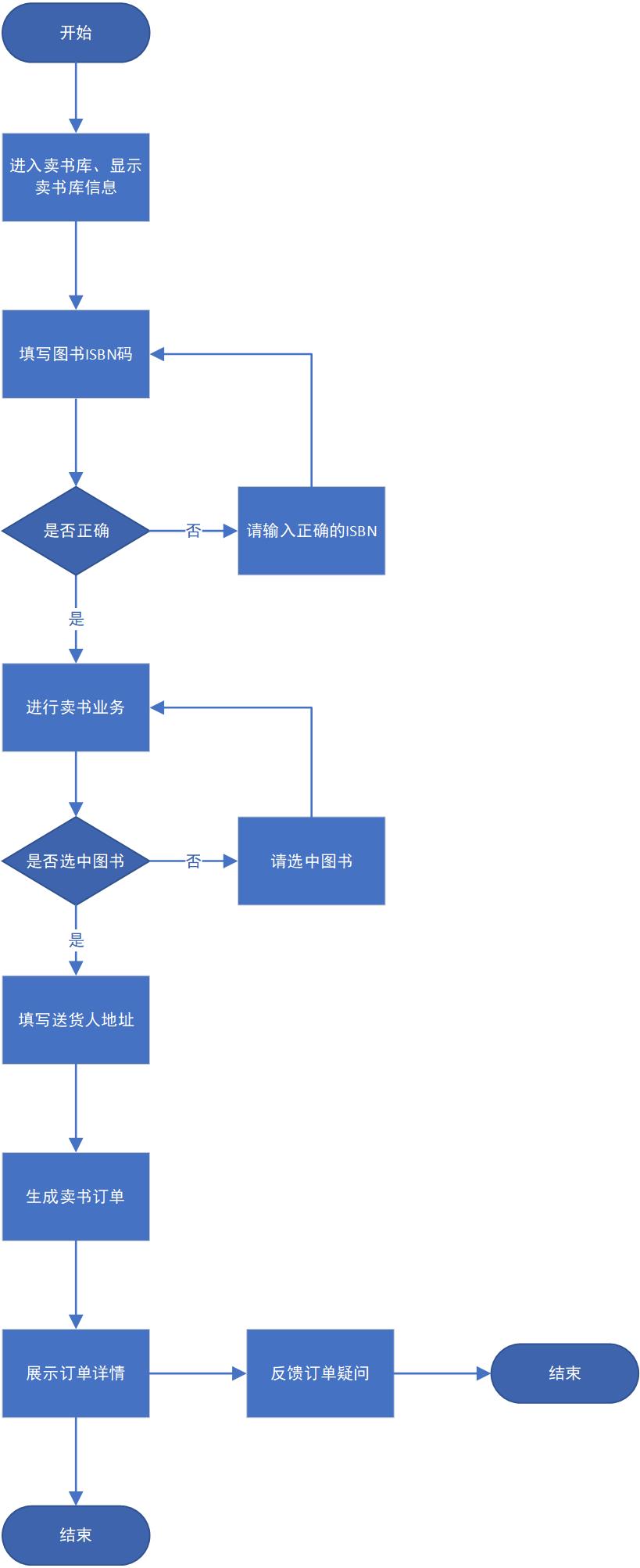
生成订单

function addSellOrderDetail() {  
 let order\_id = null  
 addSellOrder().then(resp => {  
 ***console***.log(resp.data)  
 order\_id = resp.data.id  
 let sod\_data = book\_data.map(function (item) {  
 return {book\_isbn: item.isbn, predict\_price: item.price, customer\_condition: item.pinxiang}  
 })  
  
 sod\_data.forEach(t => {  
 t.final\_price = null  
 t.final\_condition = null  
 t.order\_id = order\_id  
 })  
 sendData = {...sendData, ...sod\_data}  
 ***console***.log(sod\_data)  
 sod\_data.forEach(t => {  
 createSellOrderDetail(t)  
 })  
  
 router.push({name: 'sellorder', state: {data: sendData}})  
  
 //删除卖书库中的对应信息  
 //*todo 用户地址没进入数据库* const deltodb = book\_data  
 ***console***.log("要删除的数据", deltodb)  
  
 let delId = deltodb.map(s => {  
 return {  
 warehouseid: s["id"],  
 };  
 })  
  
 ***console***.log("要删除的id", delId)  
 delId.forEach((item) => {  
 ***console***.log("准备删除的id为", item.warehouseid)  
 deleteWareHouseBook(item.warehouseid)  
 })  
 })  
  
}

提交反馈

const open1 = () => {  
 const uid = ***window***.sessionStorage.getItem("id")  
 ***console***.log(question.value)  
 ***console***.log(uid)  
 feedback(uid,question.value)  
 ***console***.log(feedbackdialog)  
 feedbackdialog.value = false  
 ***console***.log(feedbackdialog)  
 ***console***.log(question)  
 if (question.value === '') {  
 ***ElNotification***({  
 title: '失败啦',  
 message: '请在输入栏中输入您的问题！',  
 type: 'error',  
 })  
 } else  
 ***ElNotification***({  
 title: '成功啦',  
 message: '已将您的意见发给专员，将在1-3个工作日内回复您！',  
 type: 'success',  
 })  
}

1. 流程逻辑：



接口：

# 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()),

# 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())

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。

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。

——加在各分支点处的注释。

——对各变量的功能、范围、缺省条件等所加的注释。

1. 限制条件：

系统：Windows10以上系统/MACOS系统

软件：启动MYSQL以及Apache功能；装有浏览器

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 |
| 查看卖书库模块 | 熟悉vue\JavaScript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 | 2023.4.30前完成 |
| 卖书库选商品确认订单模块 | 熟悉vue\JavaScript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 | 2023.4.30前完成 |
| 填写订单信息，立即出售模块 | 熟悉vue\JavaScript以及测试技术 | 王晗頔 | 测试，并填写相应记录 | 详见测试用例设计 | 2023.5.2前完成 |
| 查看订单详情，联系客服模块 | 熟悉vue\JavaScript以及测试技术 | 王子骏 | 测试，并填写相应记录 | 详见测试用例设计 | 2023.5.4前完成 |

1. 尚未解决的问题：暂无。

**6.5商店：**

1. 程序描述：

用户在商店页面可以按照图书类别、出版社、价格区间、有无库存筛选图书，也可以在搜索栏搜索图书。

2．功能：

|  |  |  |  |
| --- | --- | --- | --- |
| 功能 | 动作 | 条件 | 输出 |
| 搜索 | 输入书名 | 输出书名并且触发enter | 显示查找书目信息 |
| 分类 | 点击图书类别、出版社、价格区间、有无库存等按钮 | 点击按钮后触发 | 显示查找书目信息 |

3．性能：

①精度：输入数据的精度：小数点后保留两位有效数字

输出数据精度：小数点后保留两位有效数字  
 传输过程中的精度：小数点后保留两位有效数字

②灵活性：  
 适用于WINDOWS、MACOS操作系统

③时间特性：  
 显示分类/搜索结果：1s以内

4．输入项：

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 标识 | 数据类型 | 输入方式 |
| 书名 | B\_name | varchar | 用户输入 |

5．输出项：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 标识 | 数据类型 | 取值范围 | 输出方式 |
| 图书名 | B\_name | Varchar(100) | >0 | 自动 |
| 简介 | B\_intro | Varchar(1000) | >0 | 自动 |
| 价格 | B\_price | float | >0 | 自动 |

1. 主要代码：

**搜索栏：**

<div  
 style="width: 90%; height: 100%;background-color: white; padding: 20px 20px 5px;color: white; display: flex;align-items: center;justify-content: space-between">  
 <h1 style="font-weight: bolder; color: black; padding-bottom: 8px;display: inline">品读 · 商店</h1>  
 <div style="float: right">  
 <n-input round placeholder="搜索一本书嘿嘿" :value="search" @input="search = ***$event***" style="width: 230px"  
 @keydown.enter.native="handleSearch" clearable>  
 <template #suffix>  
 <n-icon :component="FlashOutline"/>  
 </template>  
 </n-input>  
 </div>  
</div>

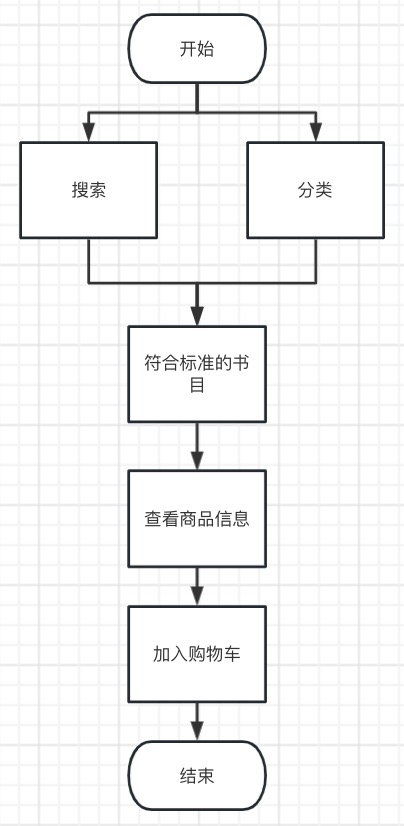
**分类器（部分）:**

<p style="font-weight: bold; color: black; margin: 2px -5px 2px 5px">图书类别：</p>  
<p style="white-space: nowrap;color: #cb3b3b;font-weight: bold; margin-right: 15px">({{ selected.cate }}) </p>  
<RouterLink @click="changeSelectedCate('-')" style="margin:0" to>  
 <p style="white-space: nowrap;margin: 0;">  
 全部  
 </p>  
 <p style="margin: 0 10px 0 0; color: deeppink">  
 ({{ books.length }})  
 </p>  
</RouterLink>  
<RouterLink v-for="(value, key) in category" style="margin:0"  
 @click="changeSelectedCate(value.category)" to>  
 <p style="white-space: nowrap;margin: 0;">  
 {{ value.category }}  
 </p>  
 <p style="margin: 0 10px 0 0; color: deeppink">  
 ({{ value.total }})  
 </p>  
</RouterLink>  
<br>

**显示商品信息：**

<div style="width: 100%;display: flex;align-items: center;justify-content: center; ">  
 <div style="height: 90%; width: 90%; background-color: white; padding: 20px 20px; ">  
 <el-row v-for="(page, index) of ***page\_num***" :key="***index***" :gutter="20" justify="space-evenly" align="middle"  
 style="padding-bottom: 20px">  
 <el-col :span="4" v-for="(item, innerindex) of ***page***" :key="item.isbn">  
 <BookItemShop :book-isbn="item.isbn" :book-cover="item.picture" :book-title="item.name"  
 :book-introduction="item.intro" :book-price="item.price"/>  
 </el-col>  
 </el-row>  
 <div style="background-color: #ffffff; height:auto;display: flex;align-items: center;justify-content: center;">  
 <el-pagination v-model="***currentPage***" :page-size="18" :total="***books***.length" @size-change="handleSizeChange"  
 @current-change="handleCurrentChange" style="background-color: white" size="large"/>  
 </div>  
 </div>  
</div>

1. 流程逻辑：



1. 接口：

# 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()),

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。

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。

——加在各分支点处的注释。

——对各变量的功能、范围、缺省条件等所加的注释。

1. 限制条件：

系统：Windows10以上系统/MACOS系统

软件：启动MYSQL以及Apache功能；装有浏览器

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 |
| 分类器 | 熟悉html，vue，mysql，JavaScript，django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 | 2021.4.30前完成 |
| 搜索 | 熟悉html，vue，mysql，JavaScript，django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 | 2021.4.30前完成 |

1. 关键接口代码实现：

export async function getdbbook(isbn) {  
 return api.get((`book/${isbn}/`))  
}  
  
export async function getdballbook() {  
 return api.get((`book/`))  
}  
  
export async function getbuyrate(info){  
 return api.get(`conditionrate/`,info)  
}  
  
export async function postbooktobook(info) {  
 return api.post('book/', info)  
}  
  
export async function postbooktowarehouse(info) {  
 return api.post('warehouse/', info)  
}  
  
export function getBookIsFlow() {  
 return api.get('bookisflow')  
}  
  
export async function getBookCondition() {  
 return api.get('conditionrate/')  
}  
export async function getBookInventory(isbn) {  
 return api.get(`inventory/${isbn}`)  
}  
  
export async function getCartNum(user\_id,isbn,condition) {  
 return api.get(`cartinventorynum/${user\_id}/${isbn}/${condition}`)  
}  
export async function getBookCategory(cate, press, price, search, havinv) {  
 let s = '-'  
 if (search !== '')  
 s = search  
 return api.get(`bookcate/${cate}/${press}/${price}/${s}/${havinv}/`)  
}  
  
export async function getBookPresshouse(cate, press, price, search, havinv) {  
 let s = '-'  
 if (search !== '')  
 s = search  
 return api.get(`bookpress/${cate}/${press}/${price}/${s}/${havinv}/`)  
}  
  
export async function getBookByCategoryPresshousePriceSearchHaveinv(cate, press, price, search, havinv) {  
 let s = '-'  
 if (search !== '')  
 s = search  
 return api.get(`bookcatepresspricesearch/${cate}/${press}/${price}/${s}/${havinv}/`)  
}

**6.6修改个人信息**

1. 程序描述：

用户在成功登录系统后，可以点击自己的用户名按钮，进入编辑个人信息页面。

2．功能：

|  |  |  |  |
| --- | --- | --- | --- |
| 功能 | 动作 | 条件 | 输出 |
| 修改个人信息 | 修改用户名、性别、生日 | 点击确认修改 | 新的用户信息 |

3．性能：

①精度：输入数据的精度：小数点后保留两位有效数字

输出数据精度：小数点后保留两位有效数字  
 传输过程中的精度：小数点后保留两位有效数字

②灵活性：  
 适用于WINDOWS、MACOS操作系统

③时间特性：  
 修改结果：1s以内

4．输入项：

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 标识 | 数据类型 | 输入方式 |
| 姓名 | U\_name | varchar | 用户输入 |
| 性别 | U\_gender | Varchar | 用户输入 |
| 生日 | U\_birthday | Varchar | 用户输入 |

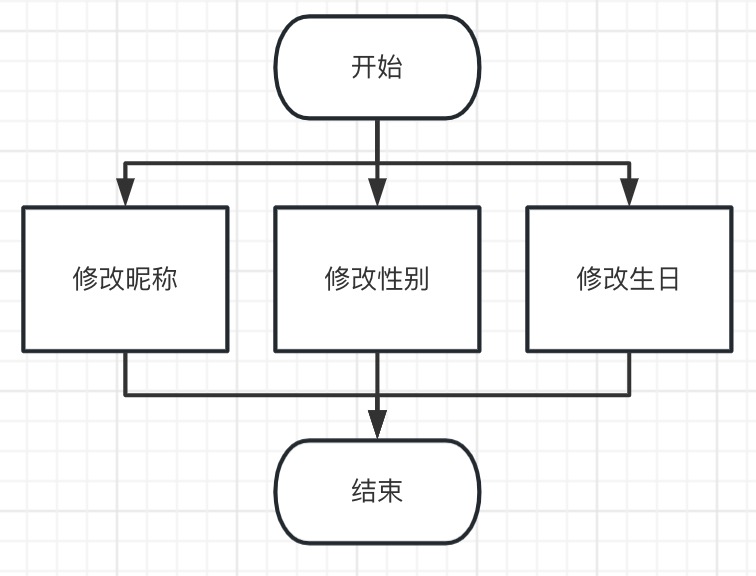
5．输出项：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 标识 | 数据类型 | 取值范围 | 输出方式 |
| 姓名 | U\_name | varchar | >0 | 自动 |
| 性别 | U\_gender | Varchar | >0 | 自动 |
| 生日 | U\_birthday | Varchar | >0 | 自动 |

1. 主要代码：

<el-form  
 ref="ruleFormRef"  
 :model="ruleForm"  
 status-icon  
 :rules="rules"  
 label-width="120px"  
 class="ruleForm">  
 <el-row justify="center">  
 <el-col span="1" :pull="3">  
 <div>  
 <el-form-item label="姓名" prop="first\_name" style="width: 100%">  
 <el-input v-model="ruleForm.first\_name" placeholder="Please input"/>  
 </el-form-item>  
 </div>  
 </el-col>  
 </el-row>  
 <br>  
 <el-row justify="center">  
 <el-col span="1" :pull="3">  
 <div>  
 <el-form-item label="性别" prop="gender">  
 <el-select v-model="ruleForm.gender" placeholder="男">  
 <el-option label="男" value="男"></el-option>  
 <el-option label="女" value="女"></el-option>  
 </el-select>  
 </el-form-item>  
 </div>  
 </el-col>  
 </el-row>  
 <br>  
 <el-row justify="center">  
 <el-col span="1" :pull="3">  
 <el-form-item label="生日" prop="birth">  
 <el-date-picker  
 v-model="ruleForm.birth"  
 type="date"  
 label="pick a date"  
 placeholder="Pick a day"  
 style="width: 100%"  
 format="YYYY/MM/DD"  
 value-format="YYYY-MM-DD"  
 />  
 </el-form-item>  
 </el-col>  
 </el-row>  
 <br>  
 <el-row justify="center">  
 <el-col span="1" :pull="3">  
 <el-form-item>  
 <el-button type="danger" @click="resetForm()">重置</el-button>  
 </el-form-item>  
 </el-col>  
 <el-col span="1" :pull="3">  
 <el-form-item>  
 <router-link :to="{name: 'self'}">  
 <el-button type="primary" @click="submitForm(ruleForm)">确认修改</el-button>  
 </router-link>  
 </el-form-item>  
 </el-col>  
 <el-col span="1" :pull="3">  
 <el-form-item>  
 <router-link :to="{name: 'self'}">  
 <el-button type="default" >返回</el-button>  
 </router-link>  
 </el-form-item>  
 </el-col>  
 </el-row>  
</el-form>

1. 流程逻辑：



8. 接口：

path('api/userdetail/<int:pk>/', people.views.UserDetail.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。

——加在各分支点处的注释。

——对各变量的功能、范围、缺省条件等所加的注释。

11．限制条件：

系统：Windows10以上系统/MACOS系统

软件：启动MYSQL以及Apache功能；装有浏览器

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 |
| 修改信息 | 熟悉html，vue，mysql，JavaScript，django以及测试技术 | 乔轲 | 测试，并填写相应记录 | 详见测试用例设计 | 2023.4.30前完成 |

1. 关键接口代码实现：

export async function userDetail(user\_id) {  
 return ***api***.get(`userdetail/${user\_id}`)  
}  
  
export async function userModify(user\_id, data) {  
 return ***api***.patch(`userdetail/${user\_id}/`, data)  
}

**6.7地址管理**

1. 程序描述：

用户在成功登录系统后，可以点击自己的用户名按钮，进入编辑个人信息页面。在左侧选择“我的信息”→“地址管理”进入页面

2．功能：

|  |  |  |  |
| --- | --- | --- | --- |
| 功能 | 动作 | 条件 | 输出 |
| 查看/新增/修改/删除地址 | 查看/新增/修改/删除收件人姓名、联系方式、省市、详细地址 | **新增**：点击新增收货人地址按钮，在录入信息后提交  **编辑**：点击编辑按钮，在录入信息后提交  **删除**：选择要删除的信息，点击删除按钮 | 新的满足条件的地址信息 |

3．性能：

①精度：输入数据的精度：小数点后保留两位有效数字

输出数据精度：小数点后保留两位有效数字  
 传输过程中的精度：小数点后保留两位有效数字

②灵活性：  
 适用于WINDOWS、MACOS操作系统

③时间特性：  
 修改结果：1s以内

4．输入项：

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 标识 | 数据类型 | 输入方式 |
| 收件人姓名 | receiver\_name | varchar | 用户输入 |
| 联系方式 | receiver\_phone | Varchar | 用户输入 |
| 省市 | receiver\_region | Varchar | 用户输入 |
| 详细地址 | receiver\_place | Varchar | 用户输入 |

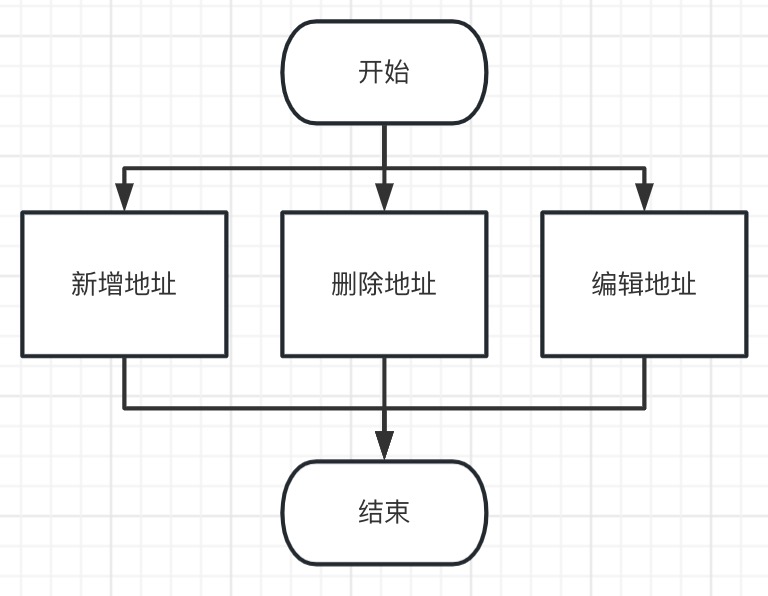
5．输出项：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 标识 | 数据类型 | 取值范围 | 输出方式 |
| 收件人姓名 | receiver\_name | varchar | >0 | 自动 |
| 联系方式 | receiver\_phone | Varchar | >0 | 自动 |
| 省市 | receiver\_region | Varchar | >0 | 自动 |
| 详细地址 | receiver\_place | Varchar | >0 | 自动 |

6.主要代码：

<el-form  
 :model="form1"  
 ref="ruleFormRef"  
 :rules="rules"  
 label-width="120px"  
 class="demo-ruleForm"  
 :size="formSize"  
 status-icon  
>  
 <el-form-item label="收件人姓名" :label-width="***formLabelWidth***"  
 prop="form2.receiver\_name">  
 <el-input v-model="form1.receiver\_name" autocomplete="off"/>  
 </el-form-item>  
  
 <el-form-item label="联系方式" :label-width="***formLabelWidth***"  
 prop="form2.receiver\_phone">  
 <el-input v-model="form1.receiver\_phone" autocomplete="off"/>  
 </el-form-item>  
  
 <el-form-item label="省市" :label-width="***formLabelWidth***" prop="form2.receiver\_region">  
 <el-select v-model="form1.receiver\_region" placeholder="选择省市"  
 :options="***options***">  
 <el-option label="上海市" value="上海市"/>  
 <el-option label="北京市" value="北京市"/>  
 <el-option label="天津市" value="天津市"/>  
 <el-option label="甘肃省" value="甘肃省"/>  
 </el-select>  
 </el-form-item>  
  
 <el-form-item label="详细地址" :label-width="***formLabelWidth***"  
 prop="form2.receiver\_place">  
 <el-input v-model="form1.receiver\_place" autocomplete="off"  
 placeholder="区、县/街道/门牌号"/>  
 </el-form-item>  
</el-form>

1. 流程逻辑：



8. 接口：

path('api/useraddress/<int:pk>/', people.views.UserAddressRetrieveUpdateDestroy.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。

——加在各分支点处的注释。

——对各变量的功能、范围、缺省条件等所加的注释。

11．限制条件：

系统：Windows10以上系统/MACOS系统

软件：启动MYSQL以及Apache功能；装有浏览器

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 |
| 编辑收货人地址 | 熟悉html，vue，mysql，JavaScript，django以及测试技术 | 乔轲 | 测试，并填写相应记录 | 详见测试用例设计 | 2023.4.30前完成 |

1. 关键接口代码实现：

export async function addUserAddress(data) {  
 return ***api***.post('useraddress/', data)  
}  
  
export async function loadUserAddress(uid) {  
 return ***api***.get(`useraddress/${uid}`)  
}  
  
export async function modifyUserAddress(uid, data) {  
 return ***api***.patch(`useraddress/${uid}/`, data)  
}  
  
export async function deleteUserAddress(uid) {  
 return ***api***.patch(`useraddress/${uid}/`)  
}

**6.8钱包管理**

1. 程序描述：

用户在成功登录系统后，可以点击自己的用户名按钮，进入编辑个人信息页面。在左侧选择“我的信息”→“我的钱包”进入页面

2．功能：

|  |  |  |  |
| --- | --- | --- | --- |
| 功能 | 动作 | 条件 | 输出 |
| 钱包管理 | 充值、提现 | **充值**：点击充值按钮，选择充值金额，点击确定  **提现**：点击提现按钮，选择提现金额，点击确定 | 实时显示钱包余额 |

3．性能：

①精度：输入数据的精度：小数点后保留两位有效数字

输出数据精度：小数点后保留两位有效数字  
 传输过程中的精度：小数点后保留两位有效数字

②灵活性：  
 适用于WINDOWS、MACOS操作系统

③时间特性：  
 修改结果：1s以内

4．输入项：

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 标识 | 数据类型 | 输入方式 |
| 充值金额 | Recharge | Float | 用户输入 |
| 提现金额 | withdraw | Float | 用户输入 |

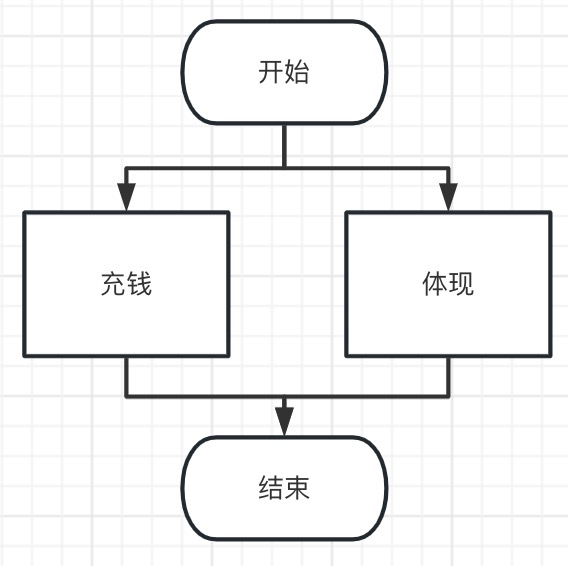
5．输出项：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 标识 | 数据类型 | 取值范围 | 输出方式 |
| 钱包余额 | Balance | Float | >0 | 自动 |

6.主要代码：

<el-form  
 ref="formRef"  
 :model="numberValidateForm"  
 label-width="100px"  
 class="ruleForm"  
>  
 <el-form-item  
 label="金额"  
 prop="balance2"  
 >  
 <el-input-number v-model="numberValidateForm.balance" :step='10' precision="0.1"  
 placeholder="输入充值金额"/>  
 </el-form-item>  
</el-form>

1. 流程逻辑：



8. 接口：

path('api/userdetail/<int:pk>/', people.views.UserDetail.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。

——加在各分支点处的注释。

——对各变量的功能、范围、缺省条件等所加的注释。

11．限制条件：

系统：Windows10以上系统/MACOS系统

软件：启动MYSQL以及Apache功能；装有浏览器

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 |
| 钱包充值提现 | 熟悉html，vue，mysql，JavaScript，django以及测试技术 | 乔轲 | 测试，并填写相应记录 | 详见测试用例设计 | 2023.4.30前完成 |

1. 关键接口代码实现：

export async function userDetail(user\_id) {  
 return ***api***.get(`userdetail/${user\_id}`)  
}  
  
export async function userModify(user\_id, data) {  
 return ***api***.patch(`userdetail/${user\_id}/`, data)  
}

**6.9查看通知**

1.程序描述：

用户在成功登录系统后，可以点击自己的用户名按钮，进入编辑个人信息页面。在左侧选择“我的信息”→“通知”进入页面

2．功能：

|  |  |  |  |
| --- | --- | --- | --- |
| 功能 | 动作 | 条件 | 输出 |
| 查看通知 | 查看发给售后人员的问题反馈 | 进入通知页面即可查看 | 问题的内容、反馈状态、回复信息 |

3．性能：

①精度：输入数据的精度：小数点后保留两位有效数字

输出数据精度：小数点后保留两位有效数字  
 传输过程中的精度：小数点后保留两位有效数字

②灵活性：  
 适用于WINDOWS、MACOS操作系统

③时间特性：  
 显示结果：1s以内

4．输入项：无

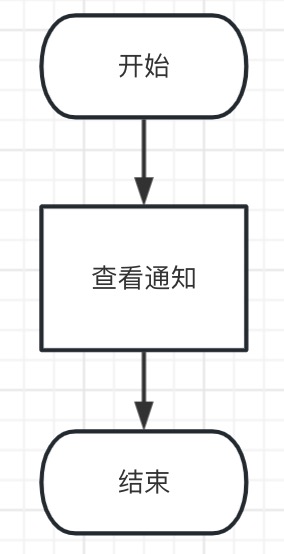
5．输出项：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 标识 | 数据类型 | 取值范围 | 输出方式 |
| 反馈状态 | Que\_condition | Varchar | >0 | 自动 |
| 问题内容 | que\_content\_user2aftersaleman | Varchar | >0 | 自动 |
| 回复内容 | que\_content\_aftersaleman2user | Varchar | >0 | 自动 |

6.主要代码：

<el-main>  
 <div v-if="get">  
 <el-scrollbar height="600px">  
 <el-table :data="tableData" style="width: 100%">  
 <el-table-column label="反馈状态" prop="condition"/>  
 <el-table-column label="问题" prop="question"/>  
 <el-table-column label="回复" prop="answer"/>  
 </el-table>  
 </el-scrollbar>  
 </div>  
</el-main>

7.流程逻辑：



8. 接口：

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

path('api/feedbackfromaftersale/', feedback.views.FeedbackListFromAftersale.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。

——加在各分支点处的注释。

——对各变量的功能、范围、缺省条件等所加的注释。

11．限制条件：

系统：Windows10以上系统/MACOS系统

软件：启动MYSQL以及Apache功能；装有浏览器

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 |
| 查看通知 | 熟悉html，vue，mysql，JavaScript，django以及测试技术 | 乔轲 | 测试，并填写相应记录 | 详见测试用例设计 | 2023.4.30前完成 |

1. 关键接口代码实现：

export async function getfeedback(uid){  
 return ***api***.get(`feedback/${uid}/`)  
}

**6.10查看买/卖书订单**

1.程序描述：

用户在成功登录系统后，可以点击自己的用户名按钮，进入编辑个人信息页面。在左侧选择“我的订单”→“买过的书”/“卖过的书”进入页面

2．功能：

|  |  |  |  |
| --- | --- | --- | --- |
| 功能 | 动作 | 条件 | 输出 |
| 查看买/卖书订单 | 查看订单详细信息 | 进入订单页面即可查看 | 下单时间、订单号、订单状态 |

3．性能：

①精度：输入数据的精度：小数点后保留两位有效数字

输出数据精度：小数点后保留两位有效数字  
 传输过程中的精度：小数点后保留两位有效数字

②灵活性：  
 适用于WINDOWS、MACOS操作系统

③时间特性：  
 显示结果：1.5s以内

4．输入项：无

5．输出项：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 标识 | 数据类型 | 取值范围 | 输出方式 |
| 下单时间 | Order\_time | Varchar |  | 自动 |
| 订单号 | Order\_id | Int |  | 自动 |
| 订单状态 | Order\_attitude | Varchar | 已完成、进行中 | 自动 |

6.主要代码：

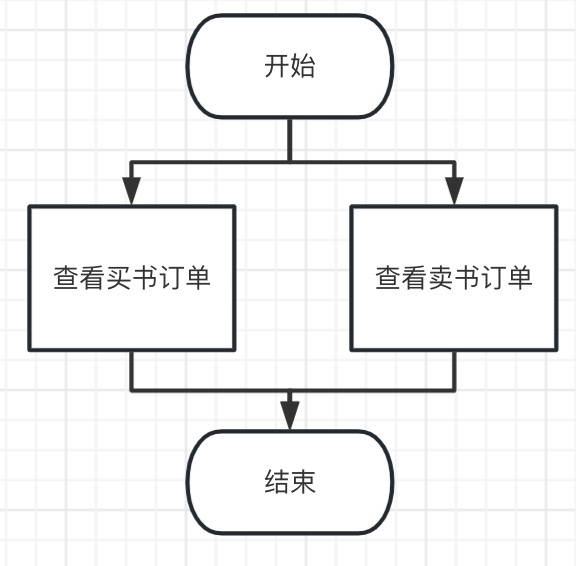
**查看买书订单：**

<el-table :data="***tableData***" name="orders">  
 <el-table-column label="时间" prop="date"  
 >  
<!-- :formatter="dateFormat"-->  
 </el-table-column>  
 <el-table-column label="订单号" prop="order">  
 </el-table-column>  
 <el-table-column label="订单状态" prop="status">  
 </el-table-column>  
 <el-table-column align="right">  
 <template v-slot:default="{row}">  
 <router-link :to="{name: 'boughtorderinfo' ,params:{oid:row.order.toString()} }">  
 <el-button size="default" type="success" >  
 查看详情  
 </el-button>  
 </router-link>  
 </template>  
 </el-table-column>  
 </el-table>

**查看卖书订单：**

<el-table :data="***tableData***" name="orders">  
 <el-table-column label="日期" prop="date">  
  
 </el-table-column>  
 <el-table-column label="订单号" prop="order">  
  
 </el-table-column>  
 <el-table-column label="订单状态" prop="status">  
  
 </el-table-column>  
 <el-table-column align="right">  
 <template v-slot:default="{row}">  
 <router-link :to="{name: 'soldorderinfo' ,params:{oid:row.order.toString()} }">  
 <el-button size="default" type="success" >  
 查看详情  
 </el-button>  
 </router-link>  
 </template>  
 </el-table-column>  
</el-table>

7. 流程逻辑：



8. 接口：

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

path('api/sellorder/<int:pk>/', sell.views.SellOrderRetrieveUpdateDestroy.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。

——加在各分支点处的注释。

——对各变量的功能、范围、缺省条件等所加的注释。

11．限制条件：

系统：Windows10以上系统/MACOS系统

软件：启动MYSQL以及Apache功能；装有浏览器

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 |
| 查看买/卖书订单 | 熟悉html，vue，mysql，  JavaScript，django以及测试技术 | 王子骏 | 测试，并填写相应记录 | 详见测试用例设计 | 2023.5.3前完成 |

1. 关键接口代码实现：

export async function getsellorders(uid){  
 return ***api***.get(`sellorder/${uid}/`)  
 }  
export async function getbuyorders(uid){  
 return ***api***.get(`buyorder/${uid}/`)  
 }

**6.11查看买/卖书详细订单**

1. 程序描述：

用户在成功登录系统后，可以点击自己的用户名按钮，进入编辑个人信息页面。在左侧选择“我的订单”→“买过的书”/“卖过的书”→选择“查看详情 ”按钮进入页面

2．功能：

|  |  |  |  |
| --- | --- | --- | --- |
| 功能 | 动作 | 条件 | 输出 |
| 查看买/卖书订单详细信息 | 查看订单详细信息，包括商品信息，下单时间，预期到货时间，订单状态等 | 进入订单详细页面即可查看 | 下单时间、订单状态、商品信息（图片、书名、品相、作者、数量、价格） |

3．性能：

①精度：输入数据的精度：小数点后保留两位有效数字

输出数据精度：小数点后保留两位有效数字  
 传输过程中的精度：小数点后保留两位有效数字

②灵活性：  
 适用于WINDOWS、MACOS操作系统

③时间特性：  
 显示结果：1.5s以内

4．输入项：无

5．输出项：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 标识 | 数据类型 | 取值范围 | 输出方式 |
| 书名 | B\_name | Varchar | >0 | 自动 |
| 品相 | B\_condition | Varchar | 优良、中等 | 自动 |
| 数量 | B\_num | Int | >0 | 自动 |
| 价格 | B\_price | Float | >0 | 自动 |
| 订单状态 | Order\_attitude | Varchar | >0 | 自动 |

6.主要代码：

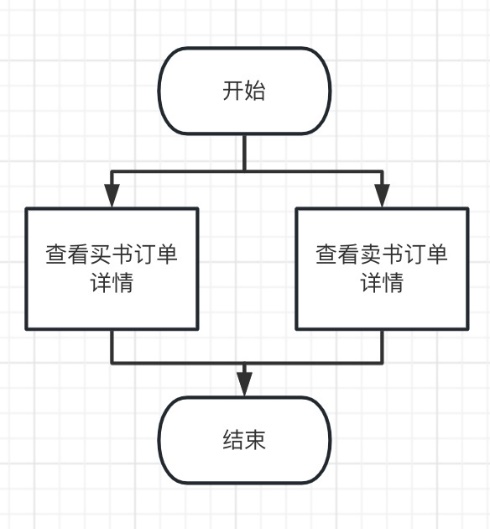
**查看买书订单：**

<el-table :data="***tableData***" name="orders">  
 <el-table-column label="时间" prop="date"  
 >  
<!-- :formatter="dateFormat"-->  
 </el-table-column>  
 <el-table-column label="订单号" prop="order">  
 </el-table-column>  
 <el-table-column label="订单状态" prop="status">  
 </el-table-column>  
 <el-table-column align="right">  
 <template v-slot:default="{row}">  
 <router-link :to="{name: 'boughtorderinfo' ,params:{oid:row.order.toString()} }">  
 <el-button size="default" type="success" >  
 查看详情  
 </el-button>  
 </router-link>  
 </template>  
 </el-table-column>  
 </el-table>

**查看卖书订单：**

<el-table :data="***tableData***" name="orders">  
 <el-table-column label="日期" prop="date">  
  
 </el-table-column>  
 <el-table-column label="订单号" prop="order">  
  
 </el-table-column>  
 <el-table-column label="订单状态" prop="status">  
  
 </el-table-column>  
 <el-table-column align="right">  
 <template v-slot:default="{row}">  
 <router-link :to="{name: 'soldorderinfo' ,params:{oid:row.order.toString()} }">  
 <el-button size="default" type="success" >  
 查看详情  
 </el-button>  
 </router-link>  
 </template>  
 </el-table-column>  
</el-table>

7. 流程逻辑：



8. 接口：

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

path('api/sellorder/<int:pk>/', sell.views.SellOrderRetrieveUpdateDestroy.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。

——加在各分支点处的注释。

——对各变量的功能、范围、缺省条件等所加的注释。

11．限制条件：

系统：Windows10以上系统/MACOS系统

软件：启动MYSQL以及Apache功能；装有浏览器

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 |
| 查看买/卖书订单  详细信息 | 熟悉html，vue，mysql，JavaScript  django以及测试技术 | 王子骏、王晗頔 | 测试，并填写相应记录 | 详见测试用例设计 | 2023.5.3前完成 |

1. 关键接口代码实现：

export async function getsellorderdetail(order\_id) {  
 return ***api***.get((`sellorderdetail/${order\_id}/`))  
}  
export async function getbuyorderdetail(id) {  
 return ***api***.get((`buyorderdetail/${id}/`))  
}

**6.12修改密码**

1. 程序描述：

用户在成功登录系统后，可以点击自己的用户名按钮，进入编辑个人信息页面。在左侧选择“设置”即可进入修改密码页面

2．功能：

|  |  |  |  |
| --- | --- | --- | --- |
| 功能 | 动作 | 条件 | 输出 |
| 修改密码 | 在旧密码/新密码/确认密码三个input内输入数据 | 进入修改密码页面，按照标准填入数据并提交 | 无 |

3．性能：

①精度：输入数据的精度：小数点后保留两位有效数字

输出数据精度：小数点后保留两位有效数字  
 传输过程中的精度：小数点后保留两位有效数字

②灵活性：  
 适用于WINDOWS、MACOS操作系统

③时间特性：  
 显示结果：1.5s以内

4．输入项：

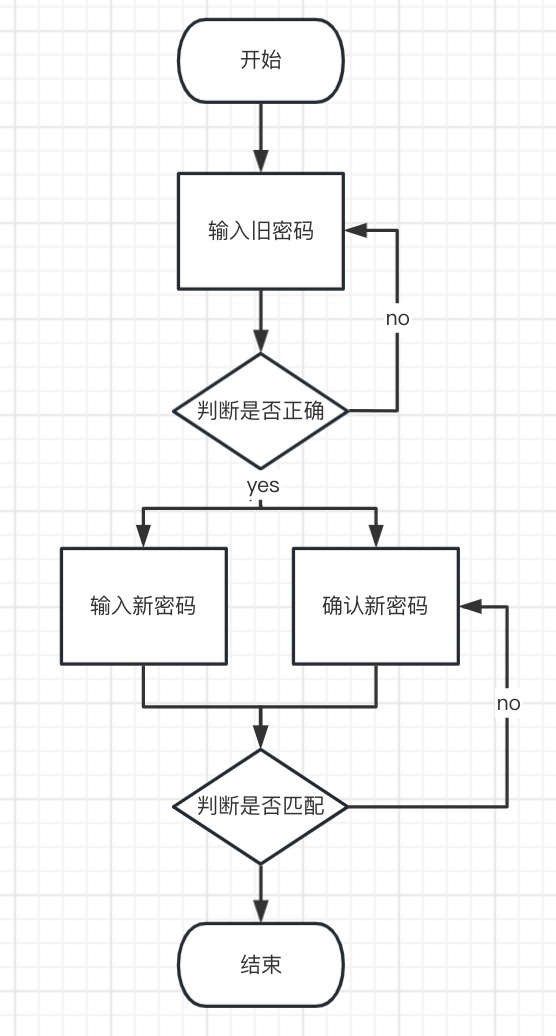
|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 标识 | 数据类型 | 取值范围 |
| 旧密码 | B\_pwd\_old | Varchar | 6-20位字母、数字、下划线 |
| 新密码 | B\_pwd\_new | Varchar | 6-20位字母、数字、下划线 |
| 确认密码 | B\_pwd\_confirm | Varchar | 6-20位字母、数字、下划线 |

5．输出项：无

6. 主要代码：

<el-form ref="formRef" :model="form" label-width="80px">  
 <el-form-item label="旧密码" prop="oldPassword">  
 <el-input style="width: 200px" v-model="form.old\_pw"></el-input>  
 </el-form-item>  
 <el-form-item label="新密码" prop="newPassword">  
 <el-input style="width: 200px" type="password" v-model="form.new\_pw"></el-input>  
 </el-form-item>  
 <el-form-item label="确认密码" prop="confirmPassword">  
 <el-input style="width: 200px" type="password" v-model="form.confirm\_pw"></el-input>  
 </el-form-item>  
 <el-form-item>  
 <el-button type="primary" @click="submitForm()">提交</el-button>  
 </el-form-item>  
</el-form>

7. 流程逻辑：



8. 接口：

path('api/userchangepassword/', people.views.UserChangePw().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。

——加在各分支点处的注释。

——对各变量的功能、范围、缺省条件等所加的注释。

11．限制条件：

系统：Windows10以上系统/MACOS系统

软件：启动MYSQL以及Apache功能；装有浏览器

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 |
| 修改密码 | 熟悉html，vue，mysql，  JavaScript，django以及测试技术 | 乔轲 | 测试，并填写相应记录 | 详见测试用例设计 | 2023.5.1前完成 |

1. 关键接口代码实现：

export async function userCPassword(data) {  
 return ***api***.post('userchangepassword/', data)  
}  
  
export async function userTPassword(data) {  
 return ***api***.post('userverifypw/', data)  
}

**6.13书目详情**

1. 程序描述：

用户在成功登录系统后，可以通过在商店点击书目进入书目详情页面

2．功能：

|  |  |  |  |
| --- | --- | --- | --- |
| 功能 | 动作 | 条件 | 输出 |
| 查看书目详情→选择品相→添加购物车 | 进入书目详情页面，查看商品是否有库存，选择品相，添加购物车 | 进入书目详情页面；  添加购物车需要该商品有库存 | 无 |

3．性能：

①精度：输入数据的精度：小数点后保留两位有效数字

输出数据精度：小数点后保留两位有效数字  
 传输过程中的精度：小数点后保留两位有效数字

②灵活性：  
 适用于WINDOWS、MACOS操作系统

③时间特性：  
 显示结果：1.5s以内

4．输入项：无

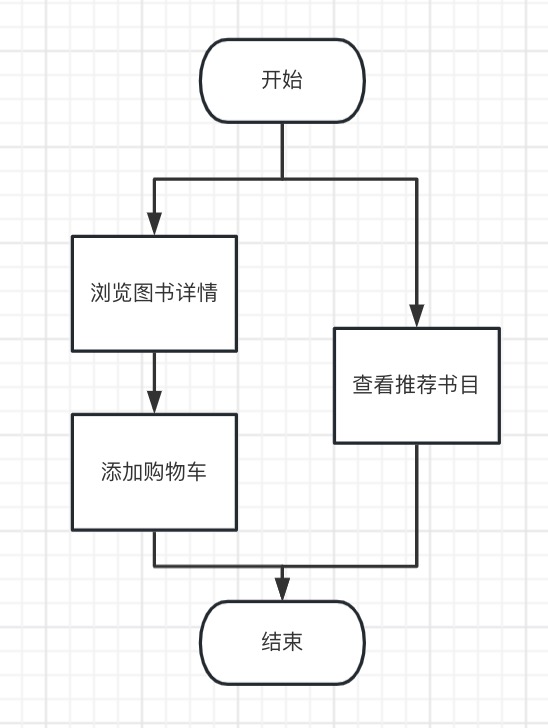
5．输出项：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 标识 | 数据类型 | 取值范围 | 输出方式 |
| 书名 | B\_name | Varchar | >0 | 自动 |
| 原价 | B\_price\_origin | Float | >0 | 自动 |
| 售价 | B\_price\_onsale | Float | >0 | 自动 |
| 品相 | B\_condition | Varchar | >0 | 自动 |
| ISBN | B\_isbn | Varchar | 13 | 自动 |
| 作者 | B\_author | Varchar | >0 | 自动 |
| 出版社 | B\_publisher | Varchar | >0 | 自动 |
| 出版时间 | B\_publish\_time | Varchar | 10 | 自动 |
| 类别 | B\_type | Varchar | >0 | 自动 |
| 内容简介 | B\_content | Varchar | >0 | 自动 |
| 相关推荐 | Recommendation |  |  | 自动 |

6. 主要代码：

<el-descriptions style=" padding-top: 10px " direction='vertical' :column="2" :border='true'>  
 <el-descriptions-item label='品相'>  
 <div v-for="condition in ***condition\_inventory***" :key="condition.id">  
 <el-tag>{{ condition.condition\_rate }}</el-tag>  
 </div>  
 </el-descriptions-item>  
 <el-descriptions-item label="原价">  
 ￥{{ book.b\_price }}  
 </el-descriptions-item>  
 <el-descriptions-item label="ISBN">  
 {{ book.b\_isbn }}  
 </el-descriptions-item>  
 <el-descriptions-item label="作者">  
 {{ book.b\_writer }}  
 </el-descriptions-item>  
 <el-descriptions-item label='出版社'>  
 {{ book.b\_presshouse }}  
 </el-descriptions-item>  
 <el-descriptions-item label='出版时间'>  
 {{ book.b\_pressdate }}  
 </el-descriptions-item>  
 <el-descriptions-item label='类别'>  
 {{ book.b\_category }}  
 </el-descriptions-item>  
</el-descriptions>  
<br><br>  
<div class="brief">  
 <dl>  
 <dt>内容简介:</dt>  
 <dd style="font-size: large">  
 {{ book.b\_introduction }}  
 </dd>  
 </dl>  
</div>

7. 流程逻辑：



8. 接口：

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

path('api/inventory/<int:pk>/', book.views.InventoryRetrieveUpdateDestroy.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。

——加在各分支点处的注释。

——对各变量的功能、范围、缺省条件等所加的注释。

11．限制条件：

系统：Windows10以上系统/MACOS系统

软件：启动MYSQL以及Apache功能；装有浏览器

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 |
| 商品详情 | 熟悉html，vue，mysql，JavaScript，  django以及测试技术 | 王子骏 | 测试，并填写  相应记录 | 详见测试  用例设计 | 2023.5.1前完成 |

1. 关键接口代码实现：

export async function postbooktoCart(info) {  
 return ***api***.post('cart/', info)  
}

export async function getBookInventory(isbn) {  
 return ***api***.get(`inventory/${isbn}`)  
}