# 数据库实训项目报告

**组长：**

姓名：马熙弘 学号：201830662165 班级：软件工程4班

**成员：**

姓名：刘 柽 学号：201830662028 班级：软件工程4班

姓名：骆家森 学号：201830662134 班级：软件工程4班

姓名：温灿雄 学号：201830663124 班级：软件工程4班

目录

[数据库实训项目报告 1](#_Toc44625989)

[一、项目简介 3](#_Toc44625990)

[二、项目开发平台和工具 3](#_Toc44625991)

[三、页面整体架构 3](#_Toc44625992)

[四、页面交互功能 6](#_Toc44625993)

[五、系统业务功能与技术设计 8](#_Toc44625994)

[六、后端服务器搭建 17](#_Toc44625995)

[七、后端数据库设计 18](#_Toc44625996)

[八、系统功能测试 23](#_Toc44625997)

## 一、项目简介

我们小组所选项目为**校园二手交易平台**。

主要实现了：用户注册与登录、物品上架出售与分类展示、交易过程中线上联系交流、求物品区、购物车、个人信息展示与修改等功能。

## 二、项目开发平台和工具

### 1.整体架构。

整体采用B/S架构。

### 2.前端。

前端采用浏览器展示网页，采用的开发语言有：HTML、CSS、JavaScript。

还使用了少部分简单的Vue框架下的语句。

例如，简单的数据绑定{{user\_name}}，v-if条件渲染，v-for列表渲染，v-on:click点击事件绑定，v-model绑定表单数据。

### 3.后端服务器部署。

使用springboot框架、Java开发语言，使用JDBC连接数据库。

### 4.数据库。

使用mySQL建立数据库。

## 三、页面整体架构

### 1.可导航页面：

可导航页面，是每个页面顶部都有导航栏，可以点击直接跳转的页面。

有：首页，全部物品页，求物品区，购物车，个人中心页。

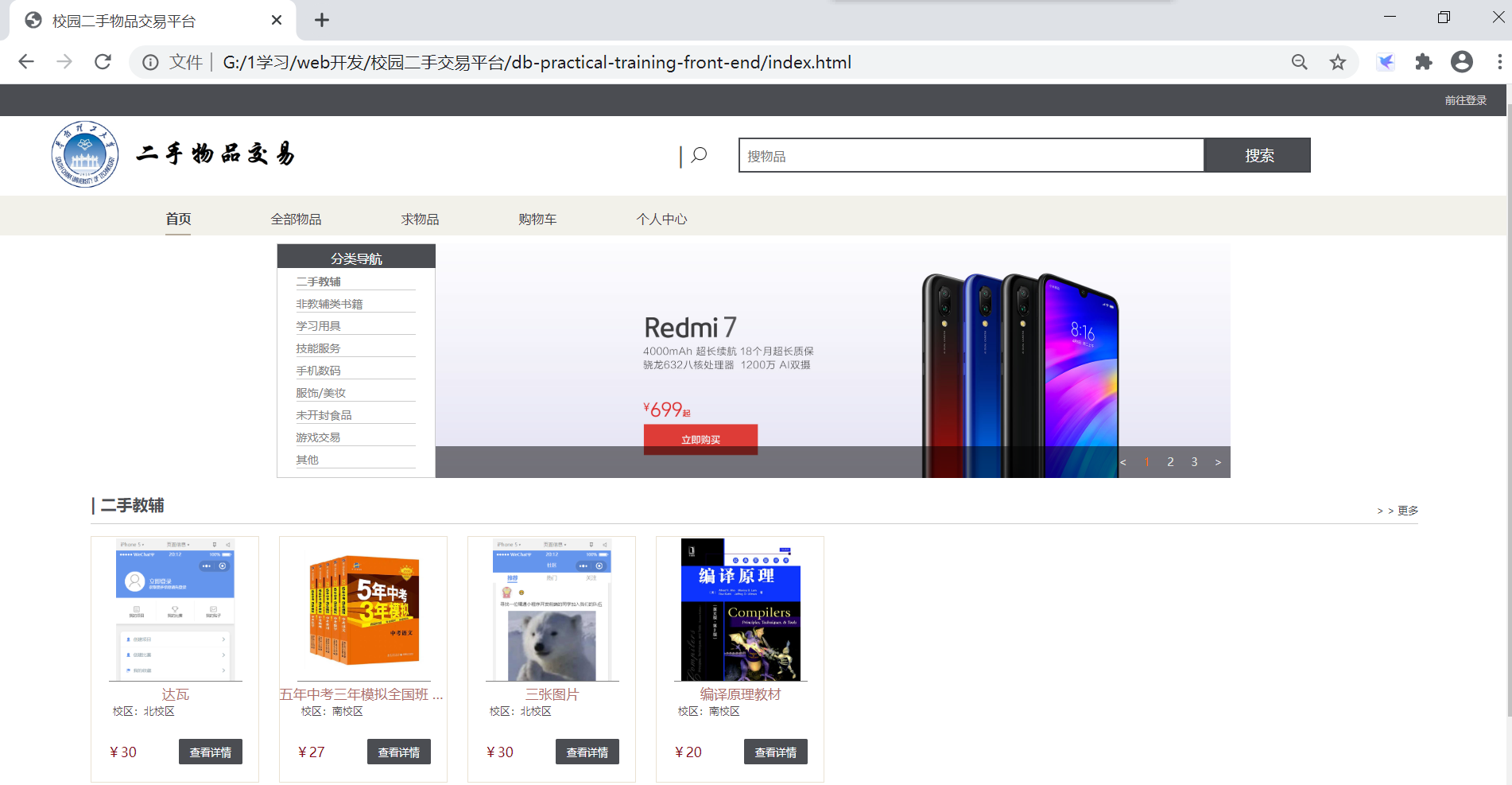


图1：首页

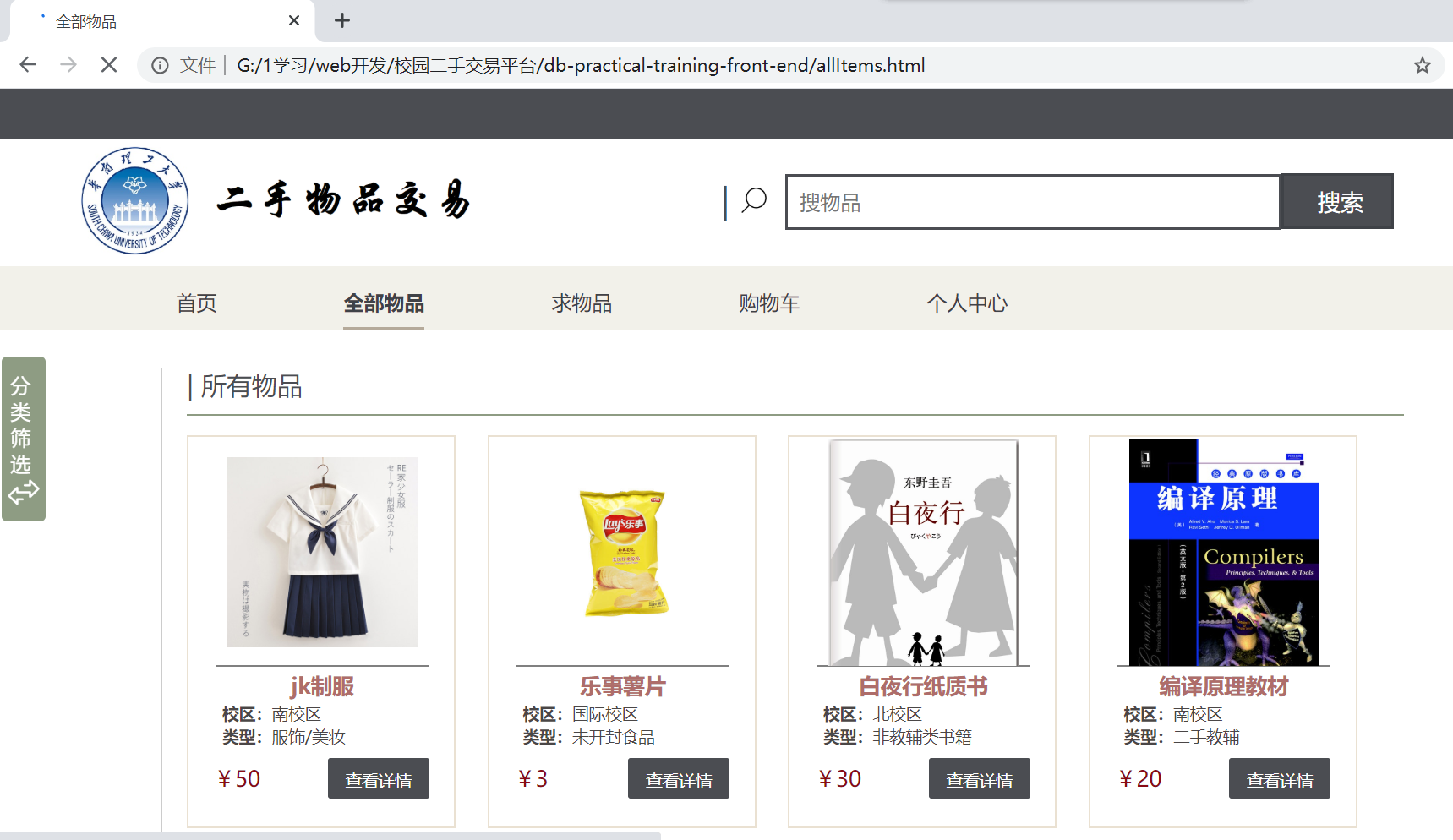


图2：全部物品页



图3：求物品区

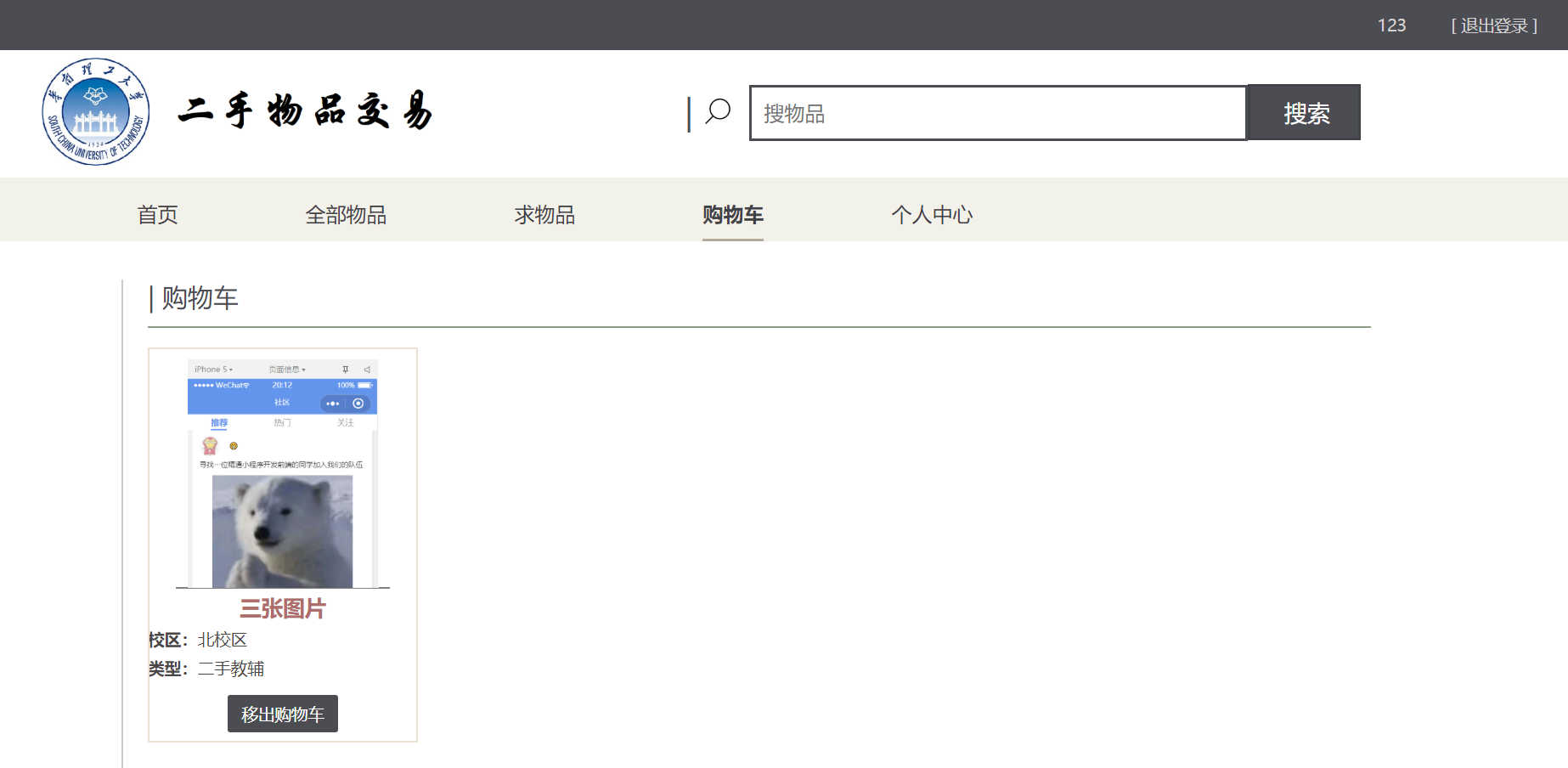


图4：购物车

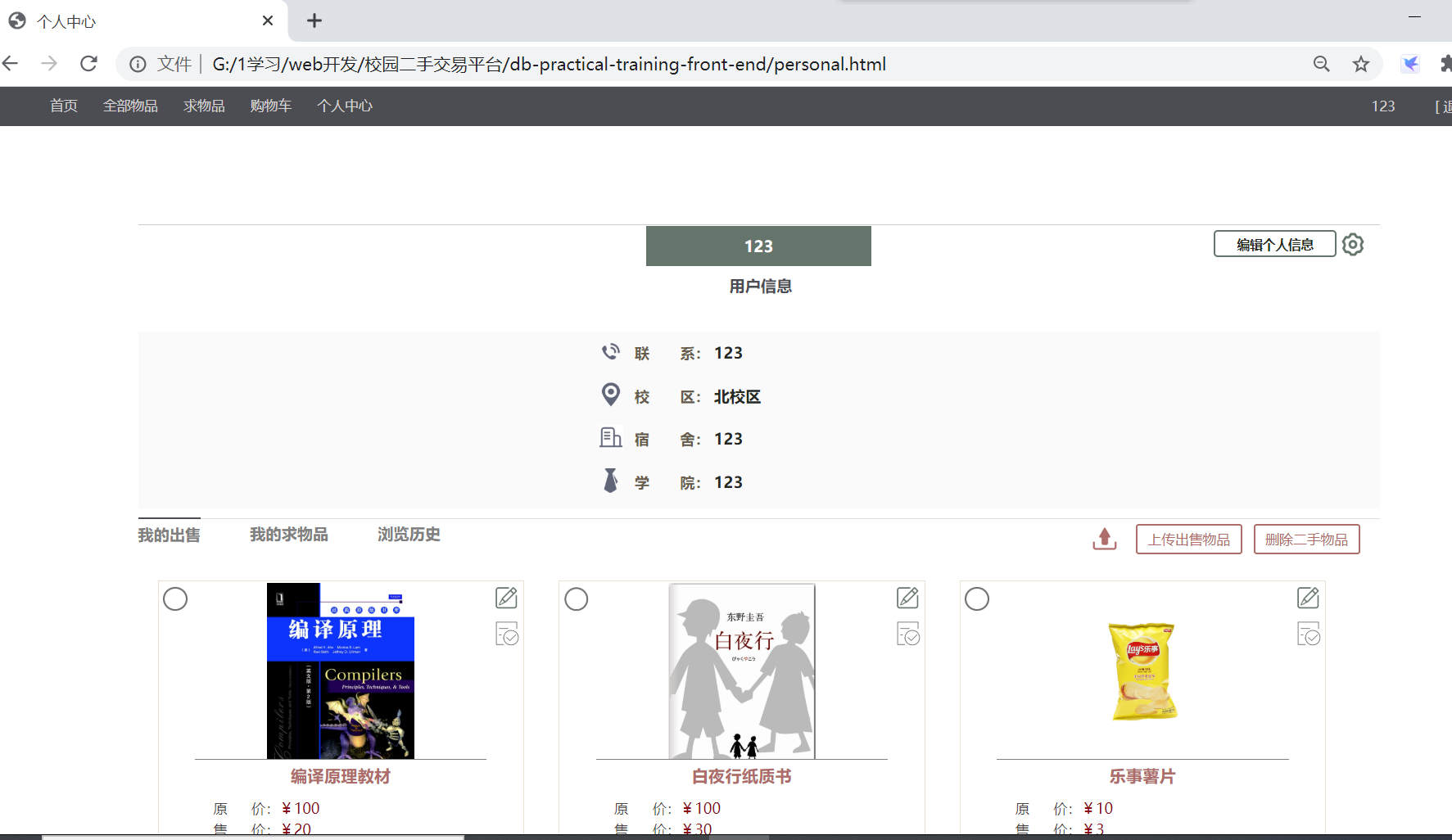


图4：个人中心

### 2.只可跳转进入的页面：

只可跳转页面只能从可导航页点击特定位置跳转。

有：物品详情页（包括出售物品、请求物品），上传出售物品页，上传请求物品页。



图4：物品详情页

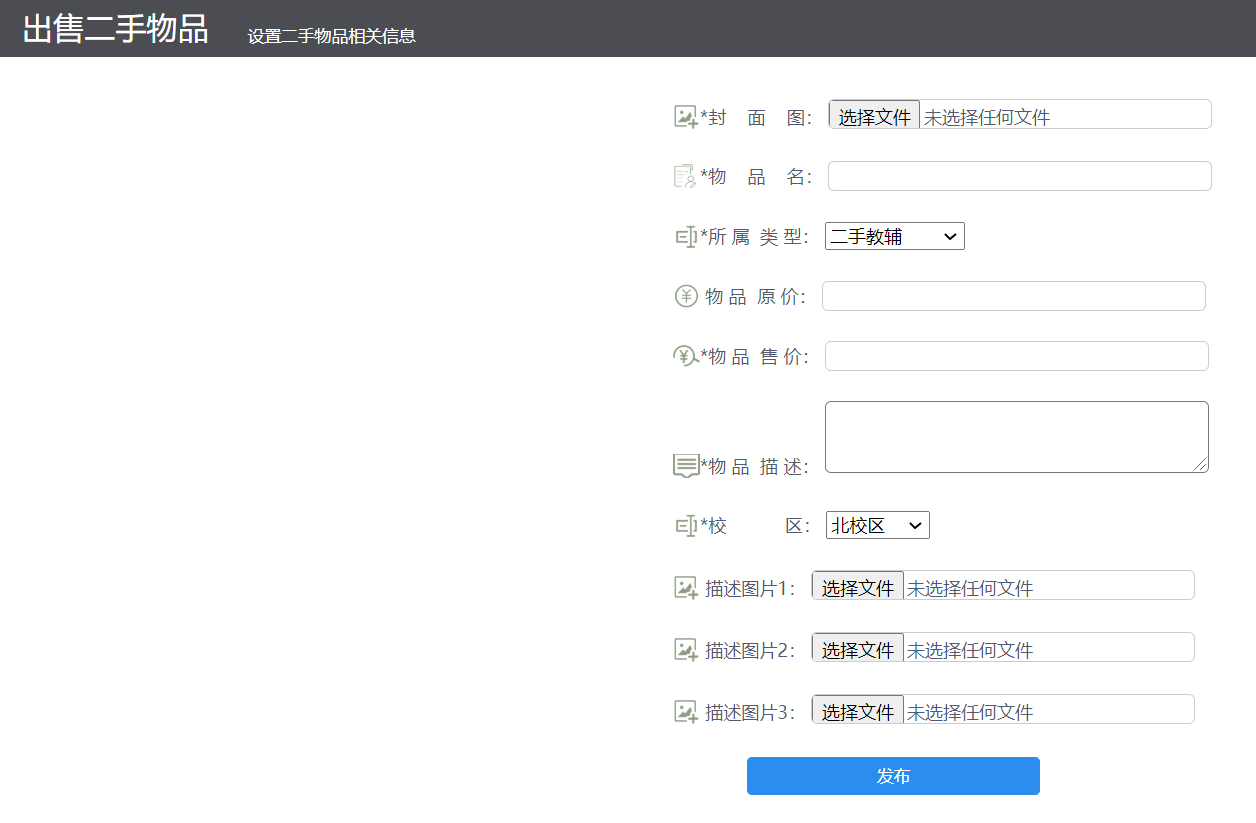


图5：出售物品上传页



图6：请求物品上传页

### 3.各个页面都包含的部件：

a.每个页面顶部的登录状态栏，用于判定用户的登录状态。

b.每个页面顶部的搜索框，用于搜索出售物品。目前仅支持搜索出售物品的信息，不支持搜索请求物品的信息。



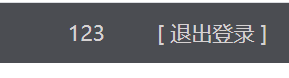


图7、8：登录状态



图9：搜索框

## 四、页面交互功能

网站有多个界面，具体内容如下：

### 1.首页。

使用者前往网页首先看到的是首页。

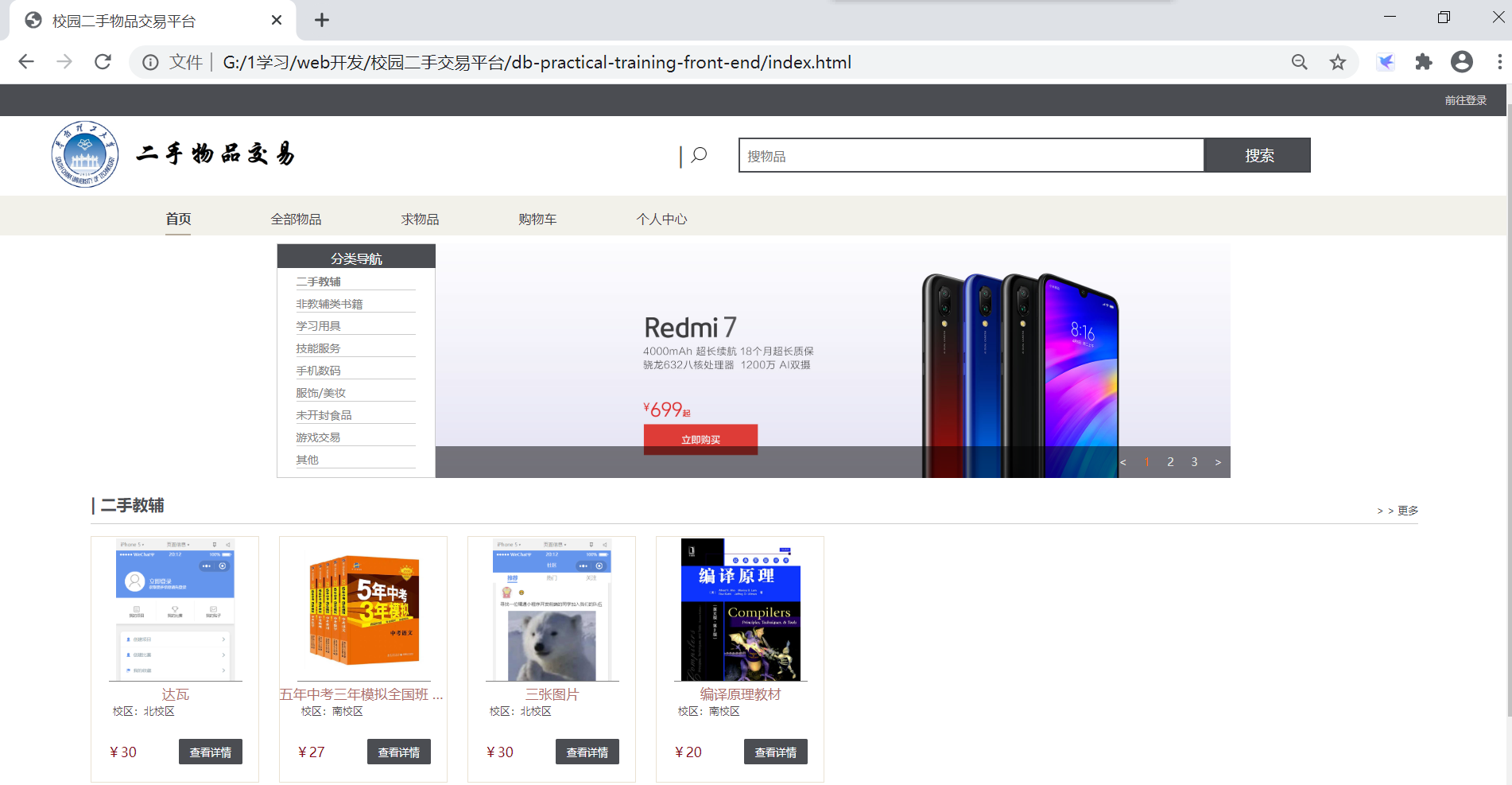


图10：首页

首页顶端导航栏显示用户的登录信息，如果已登录则显示用户名信息；如果没登录则有“前往登录”的按钮，点击后可以进入登录页面。



图11、12：登录状态栏

往下左边是网站标题“二手物品交易”，右边是搜索栏，通过搜索栏，用户可以对数据库里已经有的物品名称和描述进行查询，数据库返回符合关键词的物品，页面跳转到全部物品页显示。



图13：搜索框



图14：搜索功能

往下左侧是分类导航，每个物品都有其分类属性。点击分类导航里的某个具体分类，数据库返回该分类属性的物品并显示给用户。

其右是轮播图，轮流随机展示几个物品供使用者查看，如果使用者看到心仪的物品，即可点击直接进入物品页面。

再往下则是为每个物品分类单独分出一栏，显示部分该分类的物品。点击右侧的“更多”按钮，跳转到全部物品页面，显示该分类的全部物品。

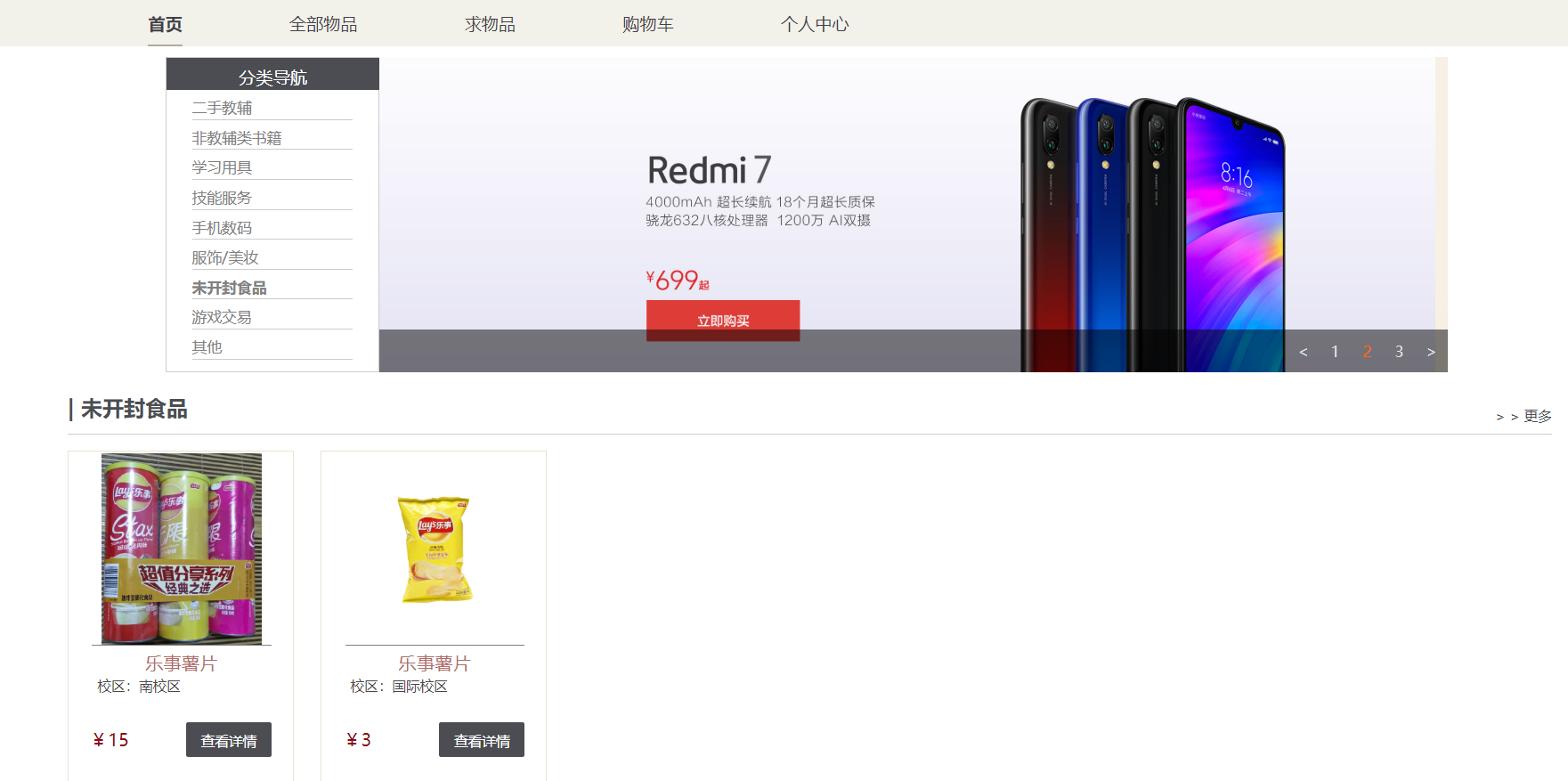


图15：首页的分类导航、轮播图

### 2.登录界面。

在首页点击前往登录后会进入登录界面。

登录界面内容简单，右上角有“返回首页”的按钮。

画面中央有两个文本输入框，分别输入账号和密码，下方共有3个按钮，分别是“没有账号？前往注册”、“忘记密码？”、“登录”。

“没有账号？前往注册”可以进入注册界面；

“忘记密码”按钮暂时没有功能；

“登录”按钮会向数据库发出查询请求，寻找是否存在相符的账号密码，有则登录成功，进入用该账户登录成功后的首页；

“返回首页”按钮则是放弃登录，直接以未登录状态返回首页。

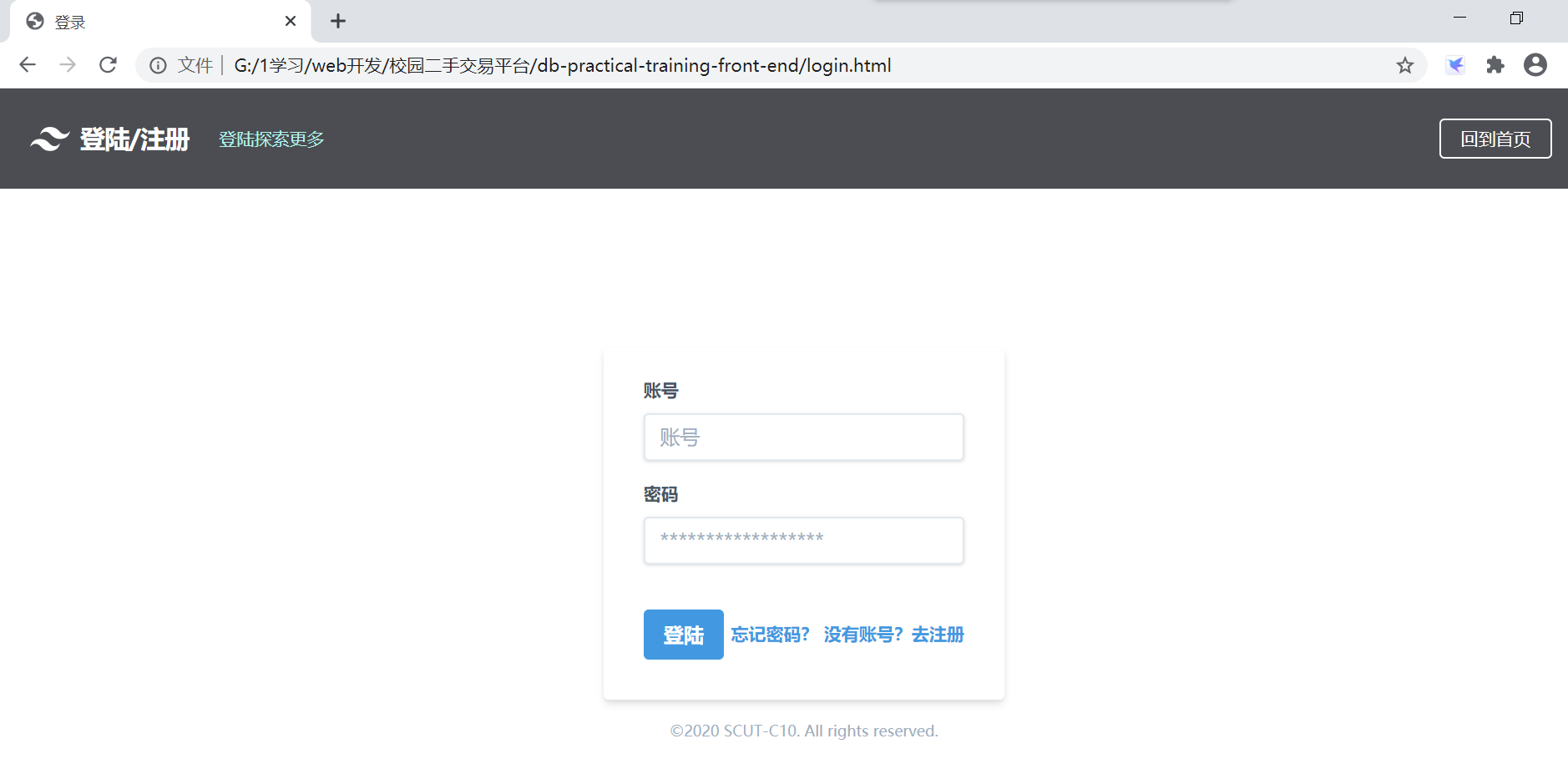


图16：登录页面



图17：注册页面

### 3.搜索界面。

在各个页面的搜索框输入物品名称，点击搜索按钮，网页向数据库查询物品名称，数据库返回目标物品的数据。然后跳转到全部物品页面上显示搜索到的具体物品图片、价格等简略信息。图片和物品名称是超链接，点击可以打开物品详情页面。



图14：搜索功能

### 4.物品详情页面。

从首页、全部物品页、求物品页、购物车、个人中心的个人创建物品处，点击物品卡片，都能跳转到物品详情页。

物品详情页面也有导航栏：“首页”、“全部物品”、“求物品”、“购物车”、“个人中心”和搜索框。

下方为物品具体描述：左为物品封面图；中为物品名称、物品所属分类、和文字物品描述，再往下是用户选填的3张物品描述图；右为物品发布人的信息，包括联系方式、校区、宿舍、学院等。

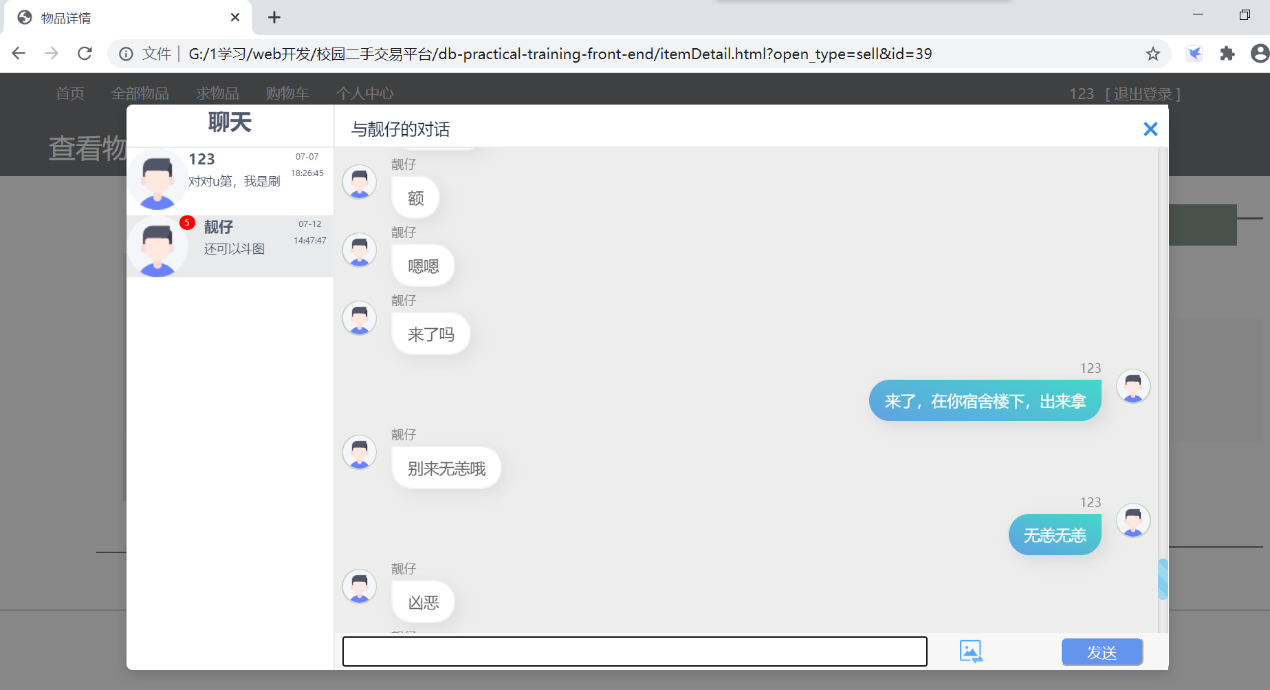


图15：物品详情页

加入购物车按钮，点击可将物品加入购物车。

有一个“立即联系”按钮。如果用户有意愿购买，可立即线上进行联系，也可根据右侧联系方式进行线下联系交易。

点击“立即联系”按钮，会弹出聊天框，用户可在线交流，聊天框功能强大，可发送文字、图片、聊天表情，聊天信息实时更新。



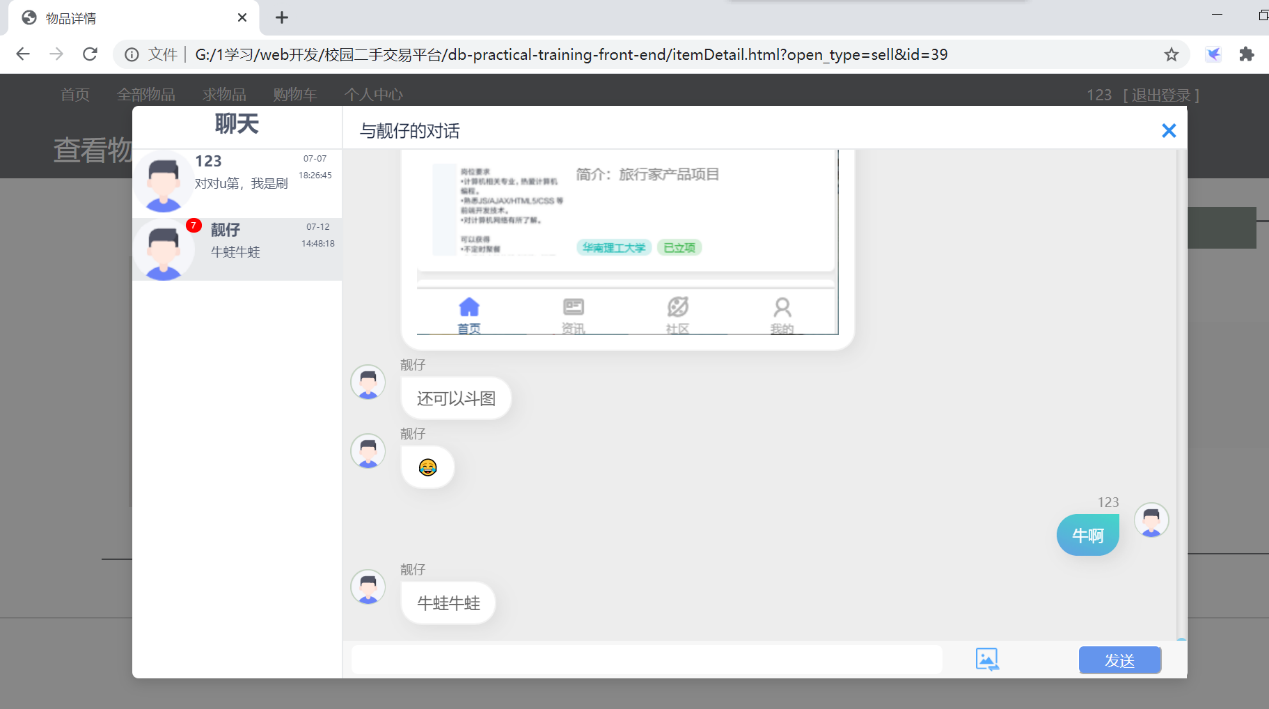




图16、17、18：在线交流功能

### 5.全部物品页面。

在各个页面导航栏点击“所有物品”按钮可以打开所有物品页面。

该页面直接展示所有还未出售的已发布二手物品。左侧是分类筛选菜单。用户可以点击图片或物品名称进入物品详情页面。

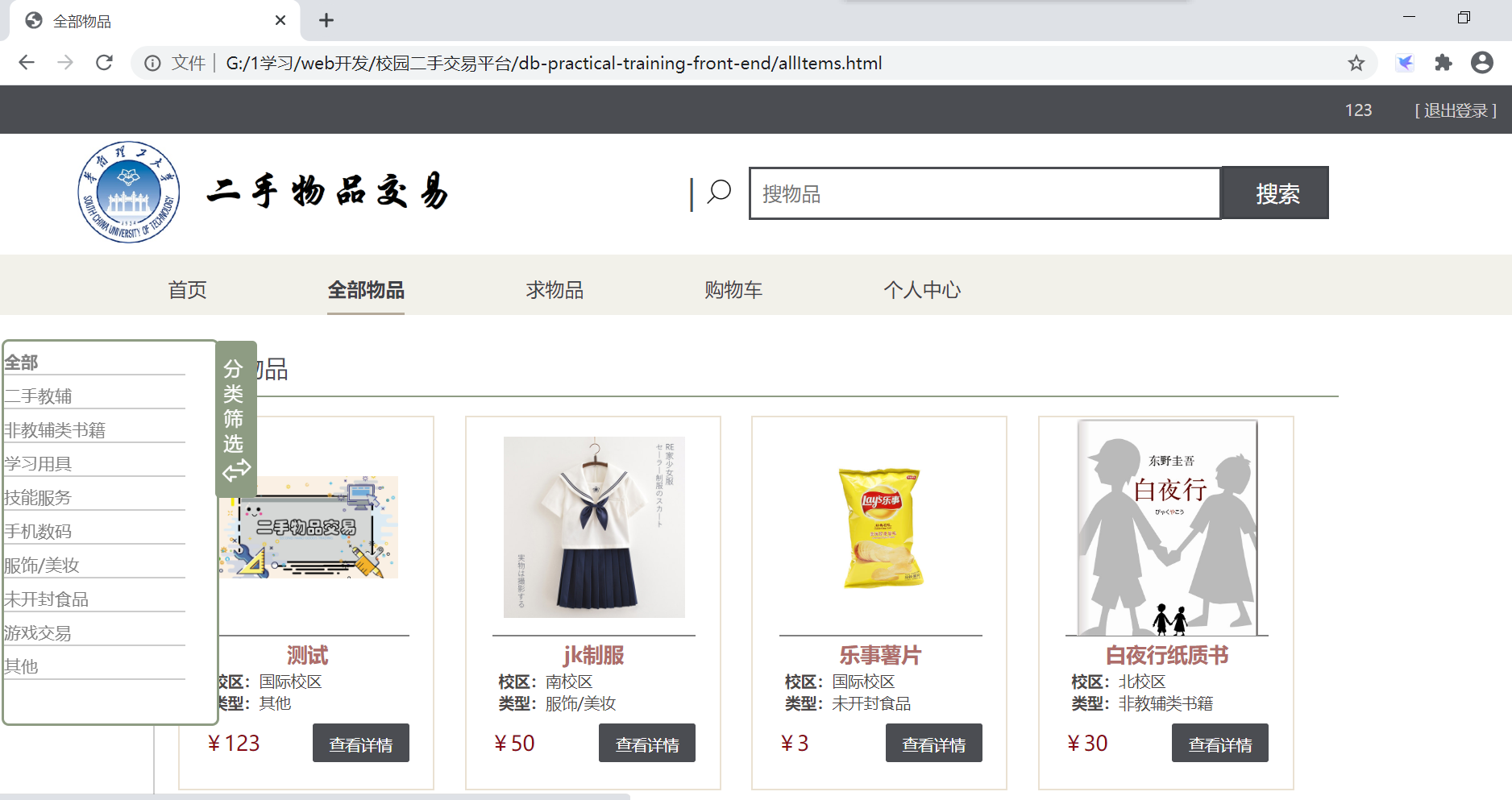


图19：全部物品页

### 6.求物品页面。

在各个页面导航栏点击“求物品”按钮可以打开求物品页面。页面将显示需要某种物品的同学所发布的请求，可以点击进入具体页面。该页面没有分类功能。

请求的物品也可进入详情页。



图20：求物品页



图21：求物品详情页

### 7.购物车。

在各个页面导航栏点击“购物车”按钮可以打开购物车页面。

页面显示目前已加入购物车的物品图片和名称。如果不想买了也可以通过点击“移出购物车”按钮将物品移出购物车。

购物车功能需登录才可以使用，否则会提示未登录。同样，可以通过点击物品图片或名称进入具体物品页面

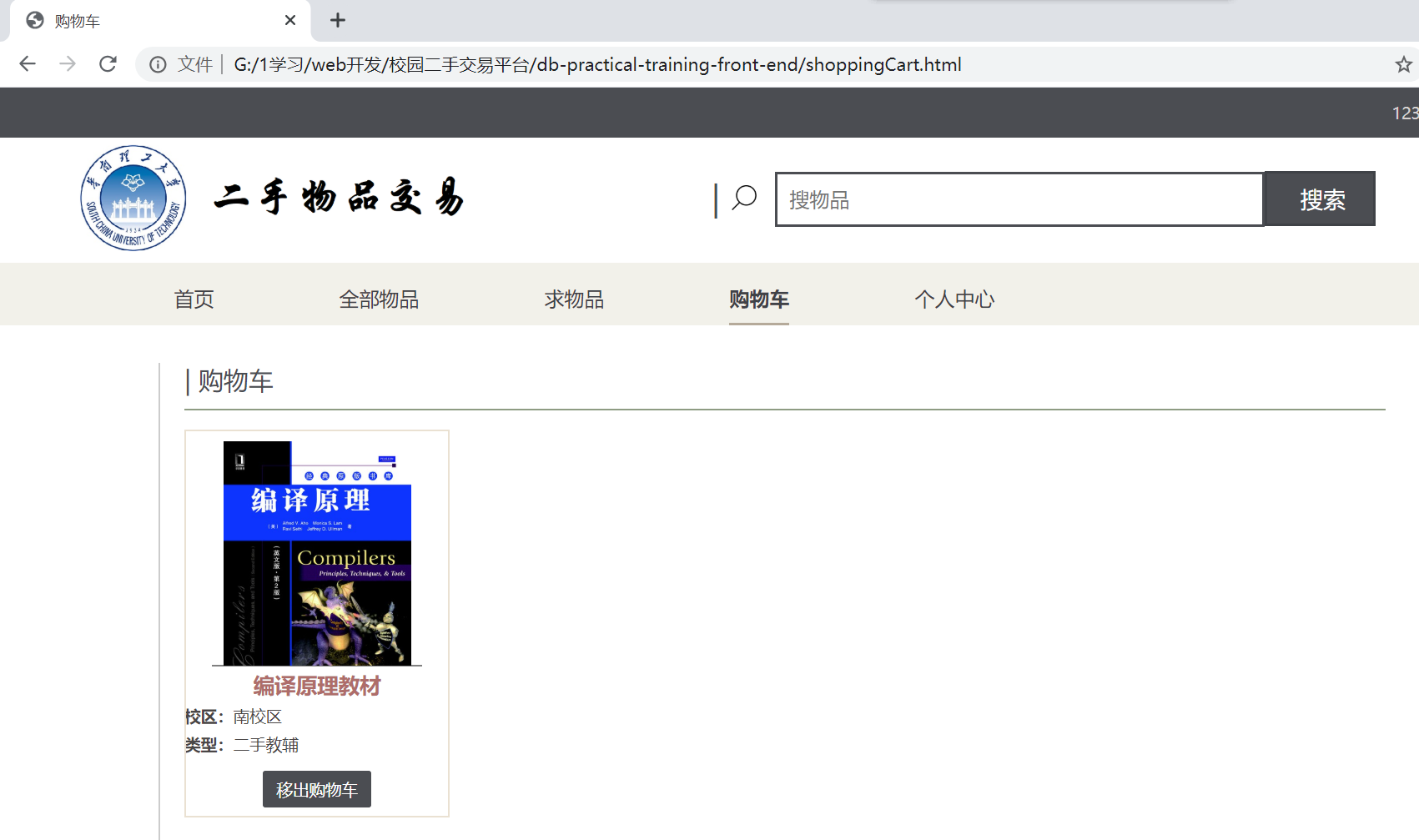


图22：购物车页

### 8.个人页面。

点击“个人页面”按钮可以进入个人页面。

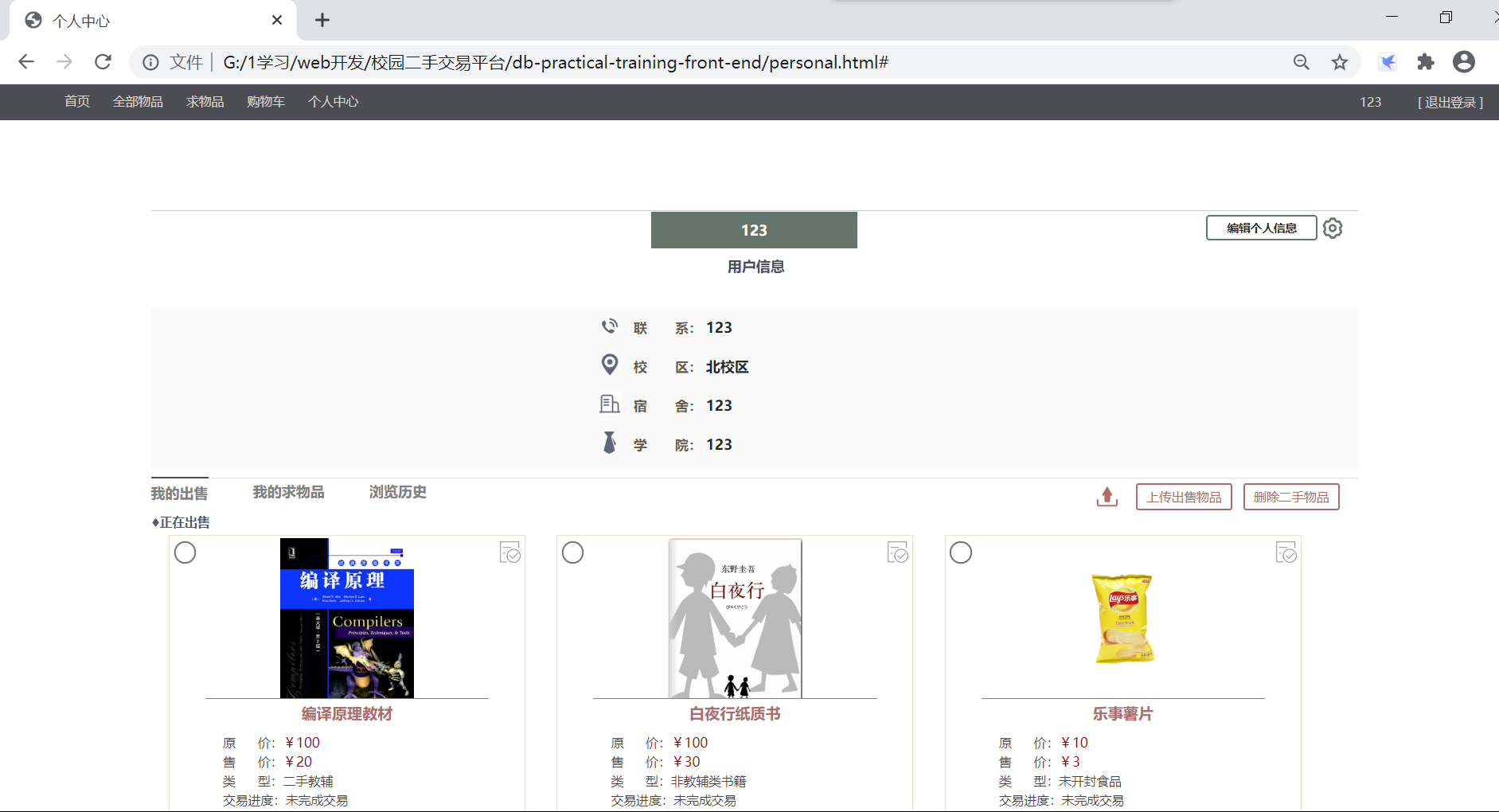


图23：个人中心页

个人页面显示个人信息，可以点击编辑，修改自己的个人信息，包括用户名、联系方式、校区、宿舍、学院。

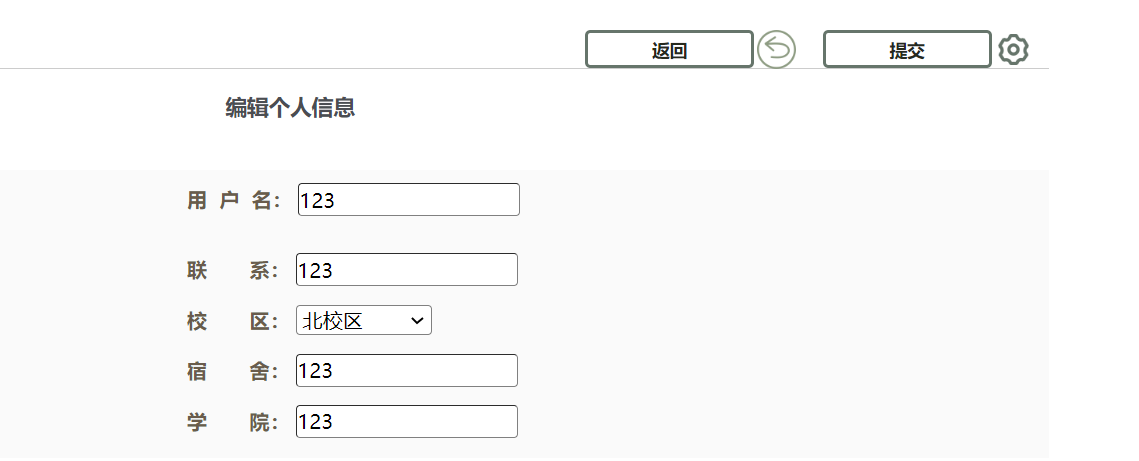


图24：修改个人信息

同时可以查看自己当前正在出售的物品、求物品信息、浏览历史。通过点击下方的“我的出售”、“我的求物品”、“浏览历史”即可。

在“我的出售”标签里，右方还有“上传出售物品”和“删除二手物品”两个按钮，上传出售物品可以在网站上张贴新的待出售物品，点击后进入出售物品上传页面；点击物品卡片左侧，可选择物品，点击删除按钮，可是删除物品，删除二手物品后物品将不再展示，创建者和其他用户都看不见，从数据库中彻底删除。

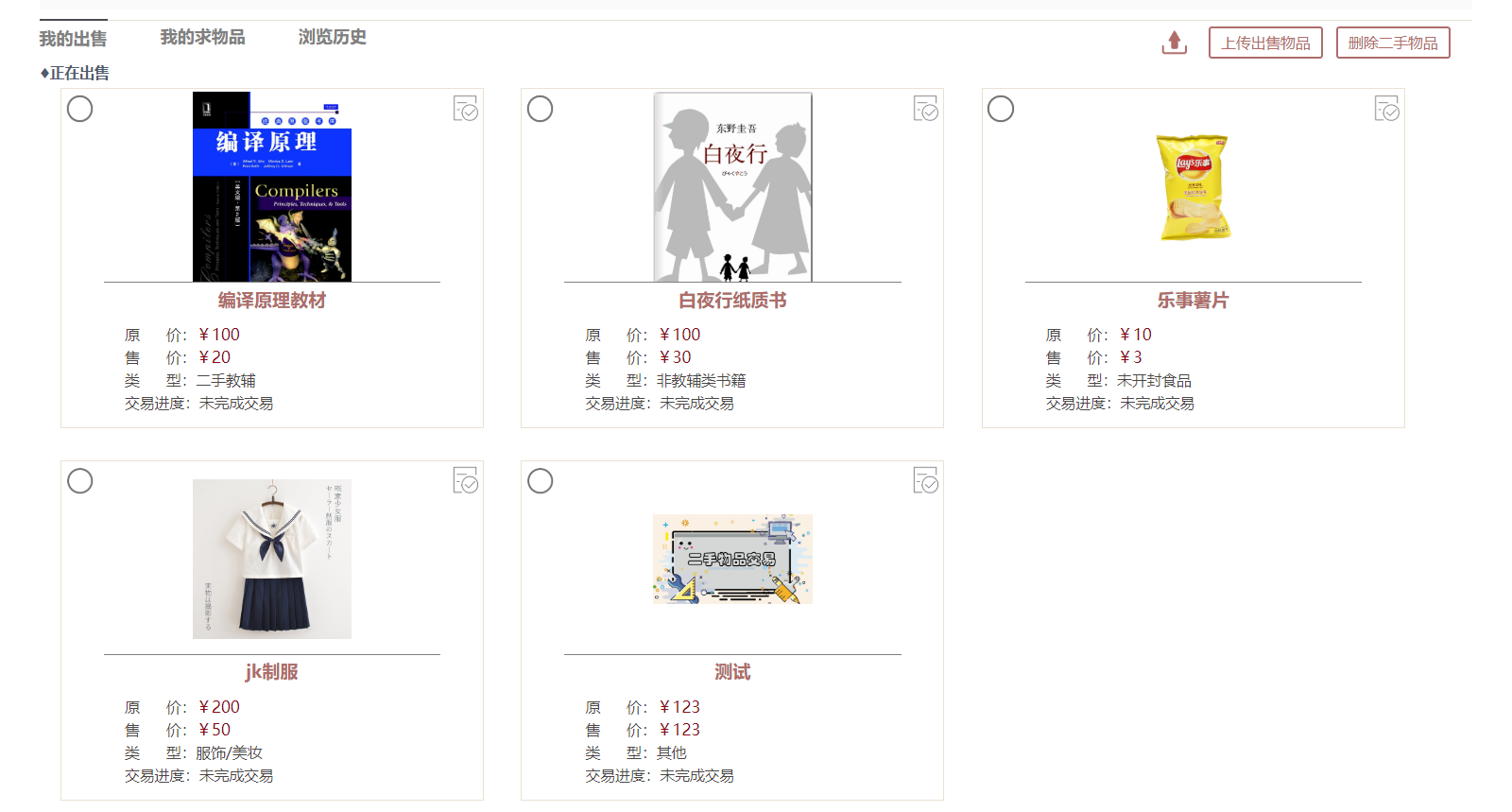


图25：我的出售物品



图26：选中物品

物品卡片右侧有完成交易按钮，点击后，物品进入完成交易列表，此后仅对创建者可见，不再在物品展示页面显示。

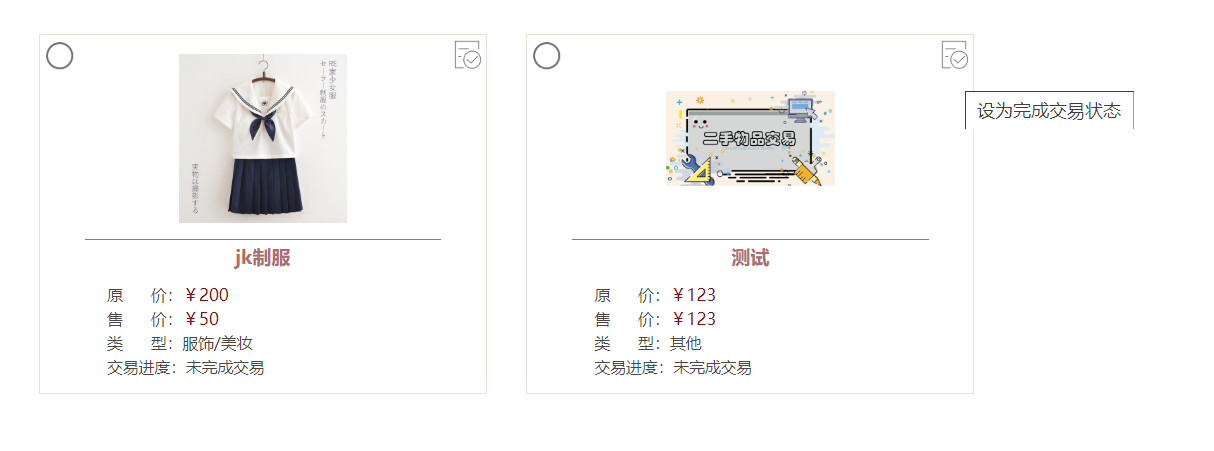


图27：物品完成交易按钮

往下，是已完成交易的物品，这些物品仅对创建者可见，不再在物品展示页面显示。

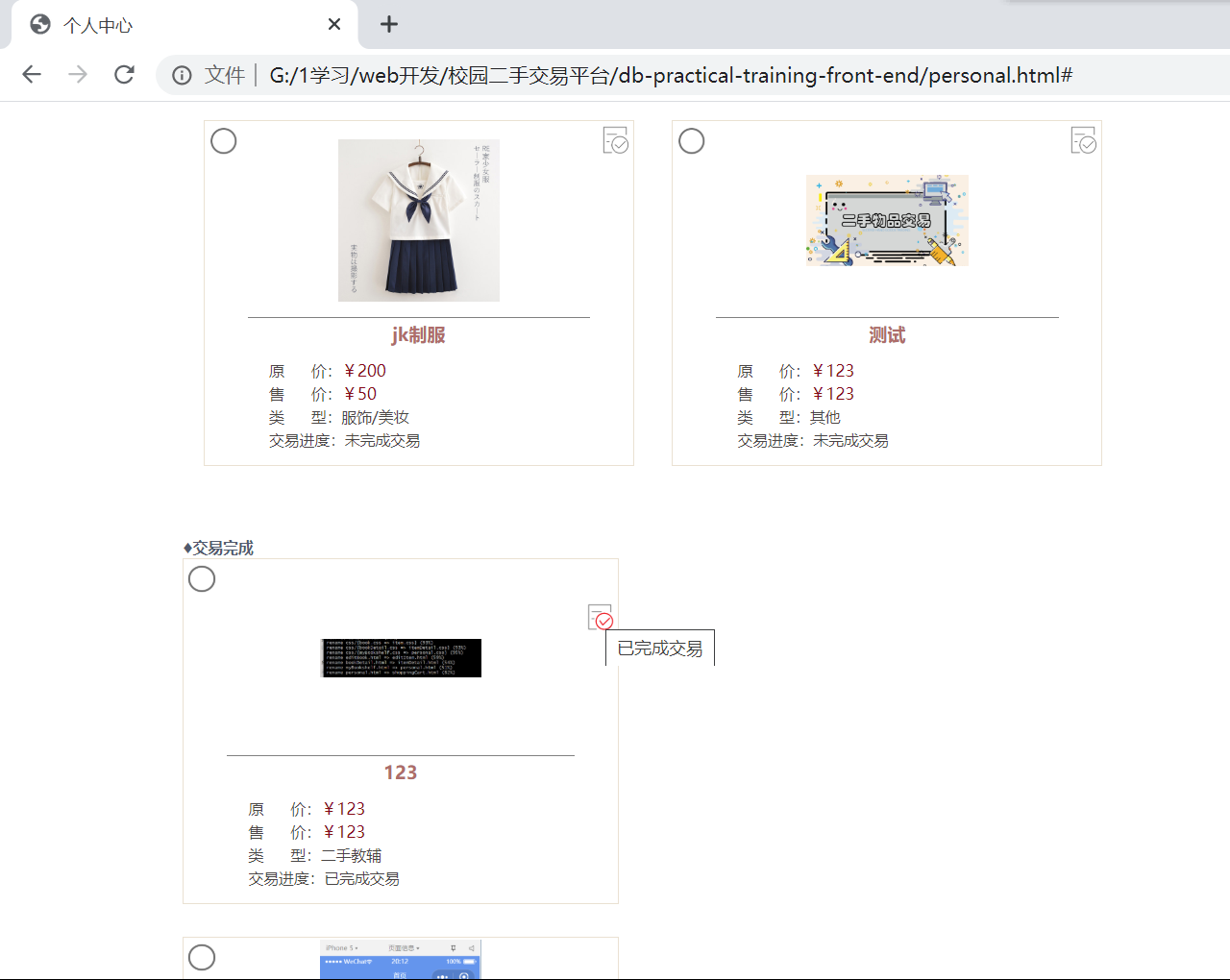


图28：完成交易物品

在“我的求物品”标签里，右方有“上传求物品信息”和“删除求物品”按钮，点击“上传求物品信息”可以进入求物品上传页面，点击“删除求物品按钮可以取消求物品。



图29：我的求物品

在“浏览历史”标签里，用户可看到自己浏览过的物品，该列表的物品也可删除。

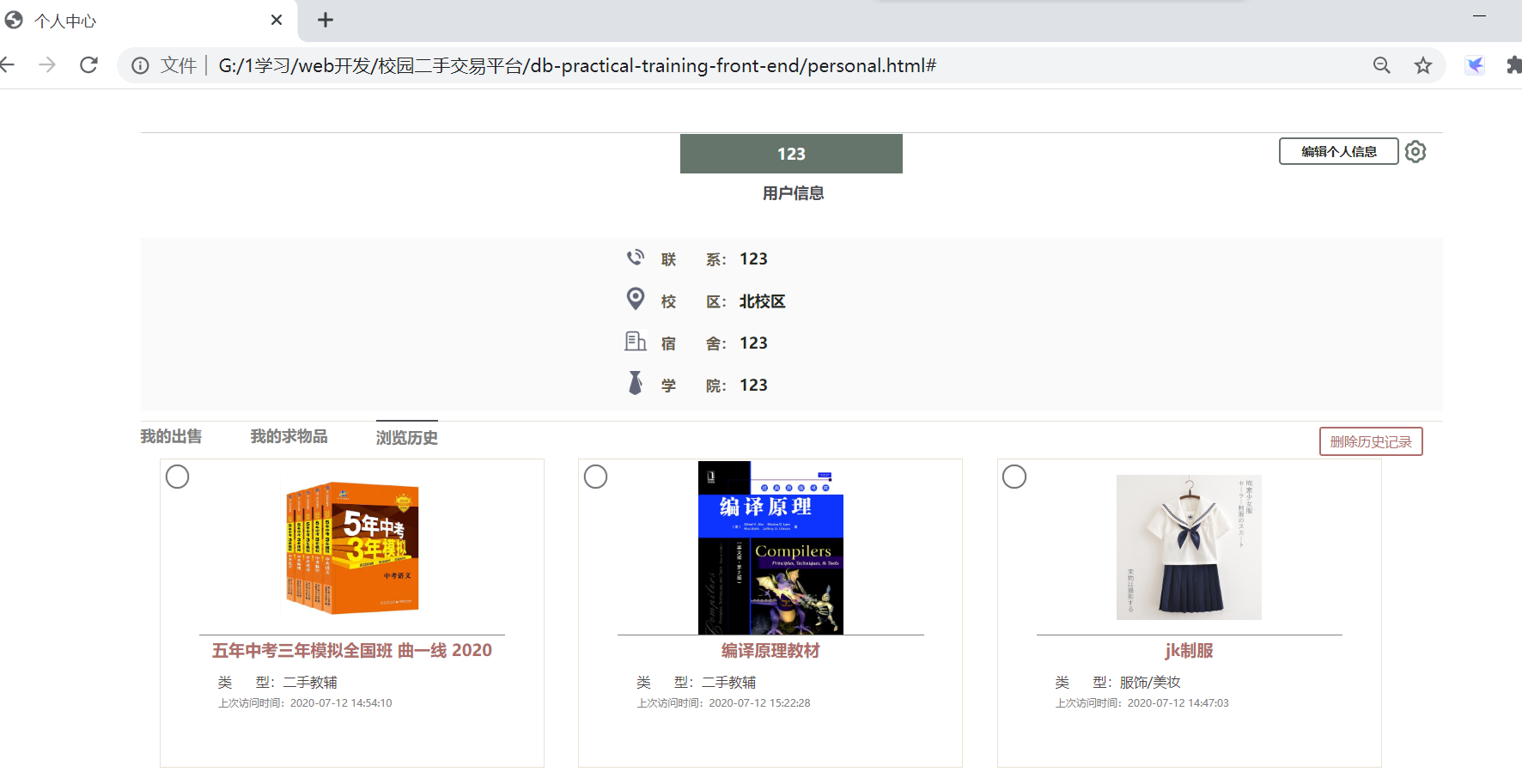


图30：我的求物品

### 9.出售物品上传页面。

该页面有多个输入框，依提示分别为封面图、物品名、所属类型、物品原价、物品售价、物品描述、描述图片1、描述图片2、描述图片3。

用户上传封面图片，并输入所有必填的信息（打\*号）后，便可点击“提交”按钮上传。上传后的物品可以被其他人浏览到，同时可以在个人页面里查看。

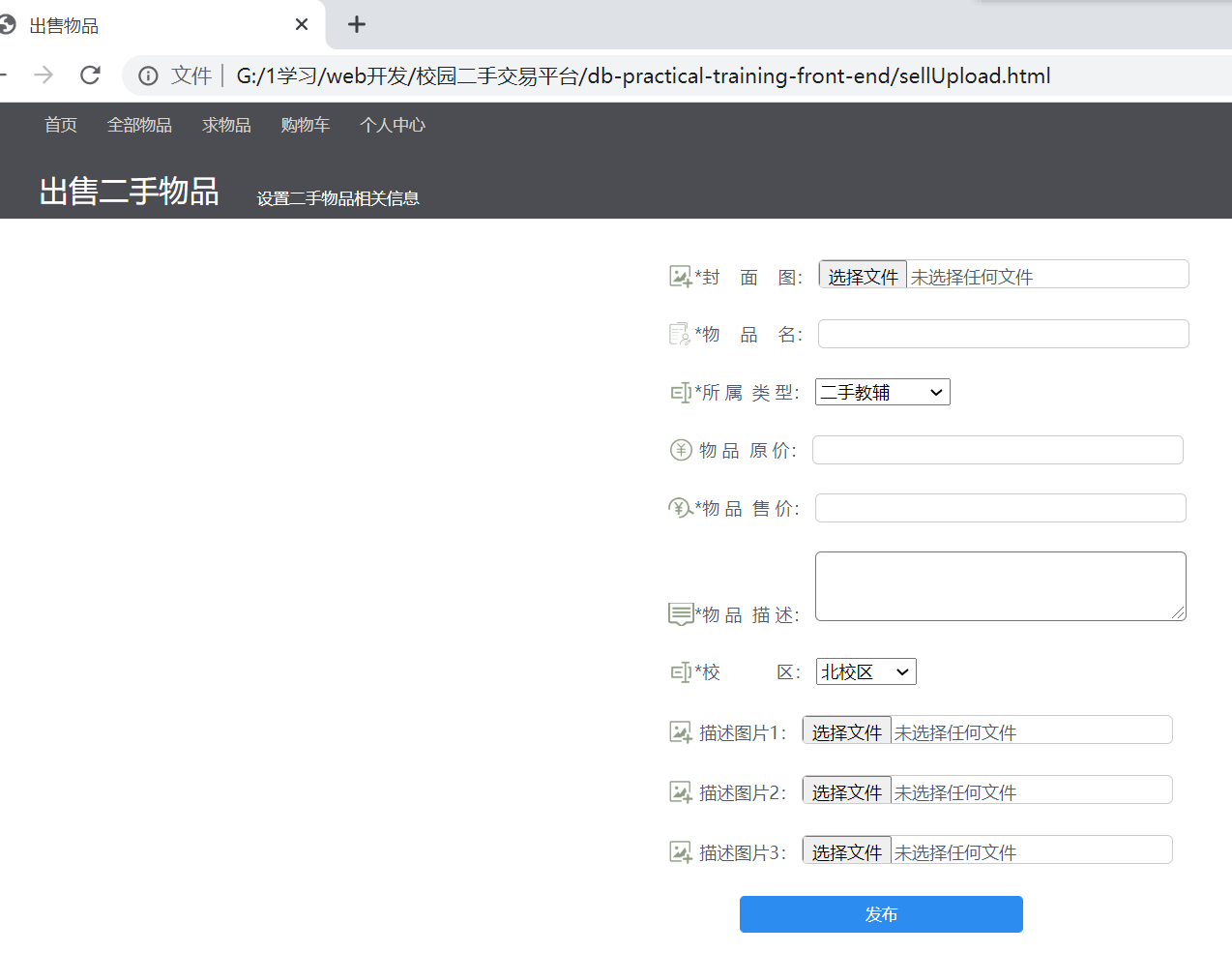


图31：出售物品上传页

### 10.求物品上传页面。

该页面类似出售物品上传页面，输入信息后可以提交上传。上传后的求物品可以被其他人浏览到，同时可以在个人页面查看。



图32：请求物品上传页

## 五、后端服务器搭建

### 1.框架：

基于Java的Springboot框架(MVC三层架构)：

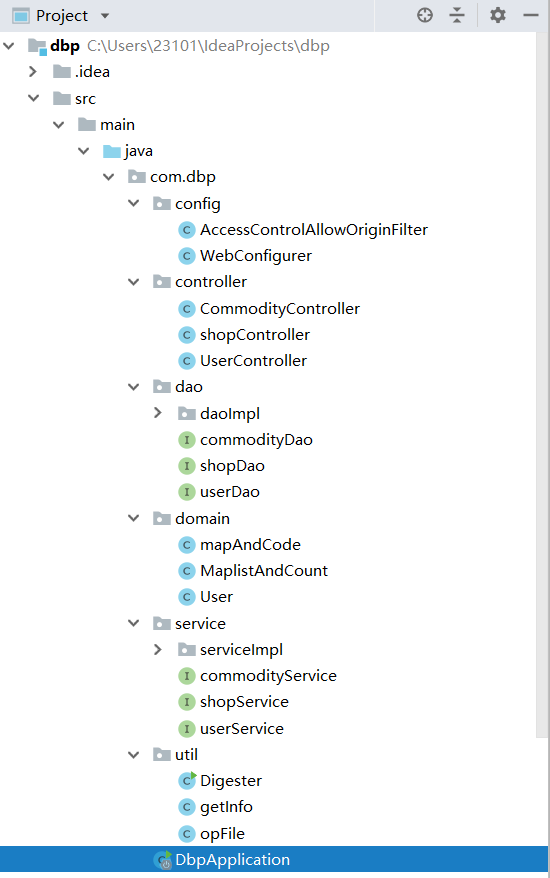


图16：后端工程目录结构

### 2.运行环境

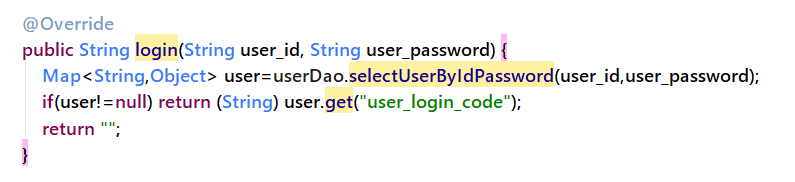
Centos7，Mysql,Tomcat

### 3.运行方式

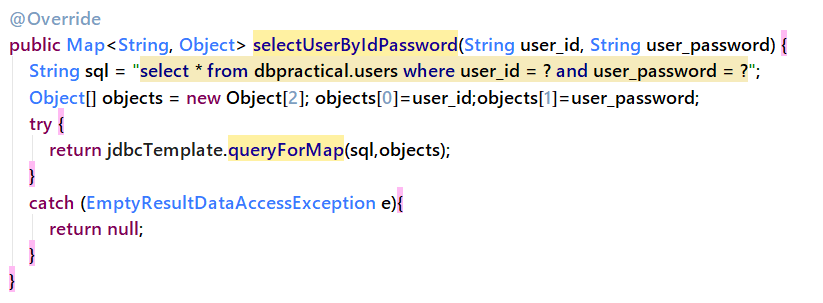
在Controller层接收请求，通过service层定义的服务来调用Dao层对应的数据库访问方法。接口示例：



UserController的登陆接口



UserService的登陆服务



Dao的数据库访问方法

使用Map<String,Object>而代替了Bean，减少了代码量。

## 六、后端数据库设计

### 1.页面分析，得出数据

根据“四、页面交互功能”中的详细页面分析，我们得到所有所需记录信息如下：

**出售物品信息：（\*号必填）**

物品名\*

所属类型\*

原价

售价\* （求物品没有价格）

描述\*

封面图片

描述图片1

描述图片2

描述图片3

物品状态：是否已完成交易

**需求物品信息：（\*号必填）**

物品名\*

所属类型\*

描述\*

封面图片

*注：求物品没有价格、描述图片、完成交易状态。*

**用户信息：**

账户，密码，每次操作用的code令牌，

昵称，联系方式，校区，宿舍，学院

**用户连接的信息：（联系集）**

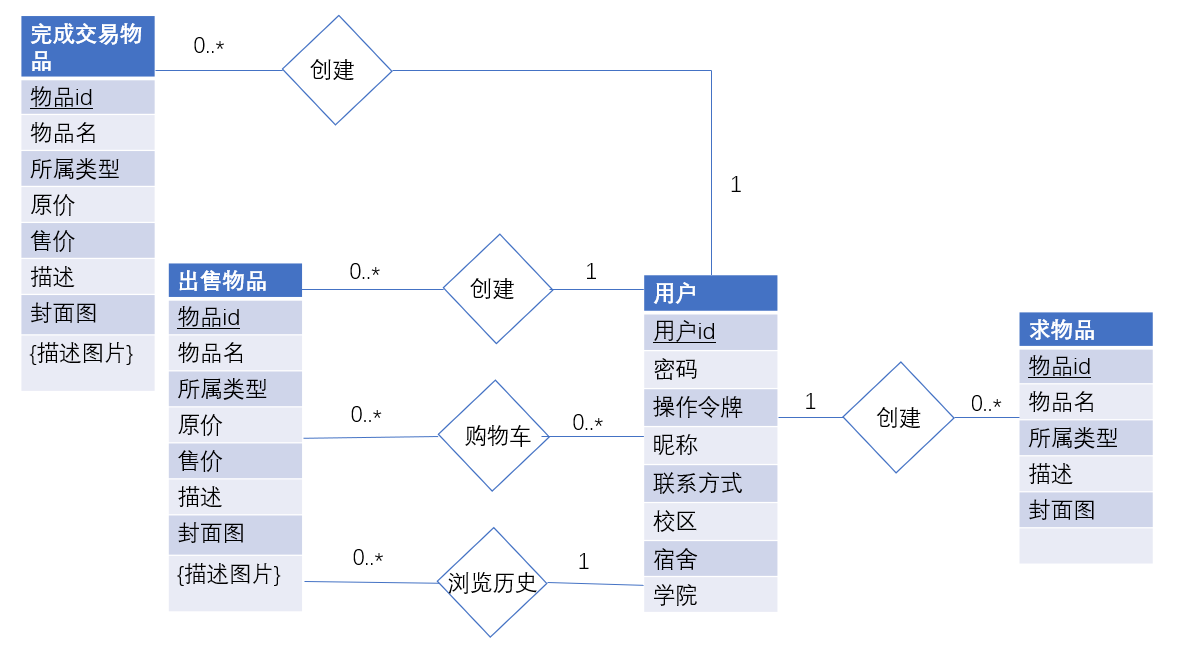
加入购物车的物品id，购物车只有出售物品，没有请求物品（多对多）

自己创建的出售物品id（一对多）

自己创建的请求物品id（一对多）

用户浏览历史：出售物品的id（一对多）

### 2.E-R图及分析



图：数据库E-R图

**E-R图分析：**

**实体集有：**

出售物品、已完成交易的出售物品、请求的物品、用户。

**联系集有：**

用户创建的出售物品、已完成交易的物品、求物品，用户浏览历史，用户将物品加入购物车。

**E-R图设计分析：**

1.物品的是否完成**交易状态**，采用分表机制实现，当用户将正在出售的物品设置为已完成交易状态，数据库将该物品从出售物品表移动到已完成交易物品表。

2.用户创建的出售物品、已完成交易的出售物品、请求的物品，都是**“一对多”**，用户是“一”方，物品是“多”方，转换为关系模式时，将“一”方，即用户的主码放进物品表中，从而**表示此联系集**。

3.**购物车的联系集**，是“多对多”，需要建立新的表，表中存有用户表主码和物品表主码。

4.出售物品、已完成交易物品的描述图片，是**多值属性**，转换为关系模式时，需要多建立一个表，存放物品id，图片地址。

5.浏览历史怎么搞的？？？？？？？？？？？？？？？？？？？？？？？？？？？？

### 3.数据库DDL

具体的数据定义语句如下：

create database dbpractical;

use dbpractical;

drop table commodity\_images;

drop table commodity;

drop table done\_commodity\_images;

drop table done\_commodity;

drop table required\_commodity;

drop table users;

注解：创建数据库，把原有的表删除，重新创建新的表。

create table users(

user\_id varchar(15) primary key,

user\_name varchar(255),

user\_password varchar(31),

user\_login\_code char(40),

tel char(11),

user\_area char(4),

dormitory varchar(7),

academy varchar(15),

check (user\_area in ('北校区','南校区','国际校区'))

);

注解：用户信息表。

信息有：账户id，用户昵称，账号密码，用户操作令牌，联系方式，校区，宿舍，学院。

其中，账户id是主码，校区限定3个：'北校区','南校区','国际校区'。

create view seller\_commodity as(

select \* from users natural join commodity

);

注解：创建视图。

将用户表与物品表自然连接（相同属性为用户id），用于物品详情页展示数据。

（我们采用的数据库是mySQL，支持自然连接语句）

create table commodity(

commodity\_id int unsigned auto\_increment primary key,

commodity\_name varchar(31),

commodity\_type varchar(15),

commodity\_original\_price int2,

commodity\_price int2,

commodity\_description varchar(1023),

commodity\_cover\_pic\_url varchar(2083),

publish\_time\_stamp char(16),

area char(4),

user\_id varchar(15),

foreign key(user\_id) references users(user\_id) on delete cascade,

check (area in ('北校区','南校区','国际校区')),

FULLTEXT (commodity\_name,commodity\_description,commodity\_type) WITH PARSER ngram,

FULLTEXT (commodity\_type) WITH PARSER ngram

);

注解：出售物品表。

用户上传的待出售物品信息存于此表。

信息有：物品id，物品名，物品类型，物品原价，物品出价，物品描述，物品封面图，校区，出售者用户id，publish\_time\_stamp？？？？

其中，主码是物品id，每次创建新物品，id自增；用户id是参照用户表上的外码，采用级联删除；

校区限定在：'北校区','南校区','国际校区'；

Fulltext？？？？？？？？？？？？？？？？？？？？？？？？？？？？？

由于用户创建出售物品信息，是一对多的联系集，对于联系集的表示，我们将“一”方的主码：用户id，放到“多”方的出售物品表中，从而表示出联系集。

create table commodity\_images(

commodity\_id int unsigned,

image\_url varchar(2083),

foreign key(commodity\_id) references commodity(commodity\_id) on delete cascade

);

注解：物品描述图片表。

由于物品的描述图片属性是个多值属性，在转换为关系模式时，需要用另外一个表。

包含：物品id，多个图片的地址；

其中物品id是参照物品表的外码，级联删除。

create table done\_commodity(

commodity\_id int unsigned primary key,

commodity\_name varchar(31),

commodity\_type varchar(15),

commodity\_original\_price int2,

commodity\_price int2,

commodity\_description varchar(1023),

commodity\_cover\_pic\_url varchar(2083),

publish\_time\_stamp char(16),

area char(4),

user\_id varchar(15),

foreign key(user\_id) references users(user\_id) on delete cascade

);

注解：已完成交易的物品表。

交易流程设计中，若用户已完成线下交易，可将该物品设为已完成交易状态。

数据库实现采用分表的方式，将已完成交易的物品移至此表。

已完成交易的物品只对上传物品的用户可见，不再显示给其他用户。

表的设计与出售物品表相同。

create table done\_commodity\_images(

commodity\_id int unsigned,

image\_url varchar(2083),

foreign key(commodity\_id) references done\_commodity(commodity\_id) on delete cascade

);

注解：已完成交易物品的描述图片表。

由于物品的描述图片属性是个多值属性，在转换为关系模式时，需要用另外一个表。

包含：已完成物品id，多个图片的地址；

其中已完成物品id是参照已完成物品表的外码，级联删除。

create table required\_commodity(

commodity\_id int unsigned auto\_increment primary key,

commodity\_name varchar(31),

commodity\_type varchar(15),

commodity\_description varchar(1023),

commodity\_cover\_pic\_url varchar(2083),

publish\_time\_stamp char(16),

area char(4),

user\_id varchar(15),

foreign key(user\_id) references users(user\_id) on delete cascade,

check (area in ('北校区','南校区','国际校区'))

);

注解：请求物品表。

我们设计了请求物品区。用户上传的求物品信息存于此表。

信息有：请求物品id，每次创建新物品，id自增；物品名，物品类型，物品描述，物品封面图，校区，创建用户id。

publish\_time\_stamp？？？

其中，主码是物品id，用户id是参照用户表的外码，校区限定在'北校区','南校区','国际校区'。

由于用户创建求物品信息，是一对多的联系集，对于联系集的表示，我们将“一”方的主码：用户id，放到“多”方的求物品表中，从而表示出联系集。

create table shopping\_cart(

user\_id varchar(15),

commodity\_id int unsigned,

primary key(user\_id,commodity\_id),

foreign key(user\_id) references users(user\_id) on delete cascade,

foreign key(commodity\_id) references commodity(commodity\_id) on delete cascade

);

注解：购物车。

购物车是作为联系集存在的，用户可将出售物品加入到购物车中。

联系集是多对多，所以转换为关系模式时，建立一个表。

包含：用户id，物品id。

主码：用户id和物品id。用户id是参照用户表的外码，级联删除；物品id是参照物品表的外码，级联删除。

### 4.数据库索引

1)在用户经常访问的数据表上，建立了关于用户id的索引

alter table users add index user\_id\_index(user\_id);

alter table commodity add index commodity\_user\_id\_index(user\_id);

alter table done\_commodity add index done\_commodity\_user\_id\_index(user\_id);

alter table required\_commodity add index required\_commodity\_user\_id\_index(user\_id);

2)为了提供关键字搜索、类型筛选的功能。我们还在一些字段上建立了全文索引以提高效率：



图：全文索引

### 5.查询优化

1）在编写SQL语句访问数据库过程中，为了提高查询效率,我们极少直接使用Natural join。取而代之的是先筛选后链接，示例如下：

with com as

(select \* from dbpractical.commodity where commodity\_id = ?)

select *\** from com natural join users

2）分页

为了避免前端数据冗余，我们使用了分页机制，编写如下SQL语句：

select \* from dbpractical.commodity order by commodity\_id desc limit 0,10

从表中得到最新的10条数据，在接下来请求时，则有如下SQL语句：

select \* from dbpractical.commodity order by commodity\_id desc limit 10,10

select \* from dbpractical.commodity order by commodity\_id desc limit 20,10

select \* from dbpractical.commodity order by commodity\_id desc limit 30,10

………

以此类推，每次请求得到10条数据。

就算在分次请求间隔有新的数据插入表中，此过程也不会受影响：我们记录了第一次访问时表中的行数，使其后每次访问都带上此行数，通过此行数和请求时行数判断limit的第一个参数取值再执行即可。

### 6.数据库安全

为了防止SQL注入，后端数据库的执行一律采用预编译模式执行，示例如下：

String sql=**"insert into required\_commodity(commodity\_name,commodity\_type,commodity\_description,commodity\_cover\_pic\_url,publish\_time\_stamp,area,user\_id) "** +  
 **"values(?,?,?,?,?,?,?)"**;  
Object[] objects = **new** Object[7];  
objects[0]=commodity\_name;  
objects[1]=commodity\_type;  
objects[2]=commodity\_description;  
objects[3]=commodity\_cover\_pic\_url;  
objects[4]=publish\_time\_stamp;  
objects[5]=area;  
objects[6]=user\_id;  
 **jdbcTemplate**.update(sql,objects); //执行SQL语句

### 7.数据库规范

经检验，数据库符合BC范式，且在逻辑设计上有良好的设计

外码约束示例：



图19：commodity\_images的外码约束

当此物品完成交易时，描述图片的image\_url字段也自动删除。

## 七、系统功能测试、数据库查询、技术实现

### 1.注册与登录、用户操作令牌

用户要想进行个人用户操作（如加入购物车、创建物品等而非浏览），需要先注册。

注册时，服务器会先检查有无已存在的用户id，没有才能成功注册。

注册和登录用的http请求是post方法而不是get，以保证信息的安全。

以下为登录的JavaScript代码：

*//登录，发送账号密码，返回操作用的code，和用户nickname*login:**function**(){  
 **let** param = **new** URLSearchParams();  
 param.append(**"user\_id"**,**this**.**userID**)  
 param.append(**"user\_password"**,**this**.**password**)  
 **let** self=**this** axios.post(**localStorage**.serverUrl+**'User/login'**, param)  
 .then(**function** (response) {  
 **console**.log(**'res:'**)  
 **console**.log(response);  
 *//下面设置用户登录态* **localStorage**.setItem(**'user\_id'**,response.data.map.**user\_id**);  
 **localStorage**.setItem(**'operation\_code'**,response.data.opcode);  
 **localStorage**.setItem(**'nickname'**,response.data.map.**user\_name**);  
 **localStorage**.setItem(**'userInfo'**,**JSON**.stringify(response.data.map));  
  
 **window**.**location**.**href**=**"index.html"**;  
 })  
 .catch(**function** (error) {  
 **console**.log(error);  
 });  
  
},

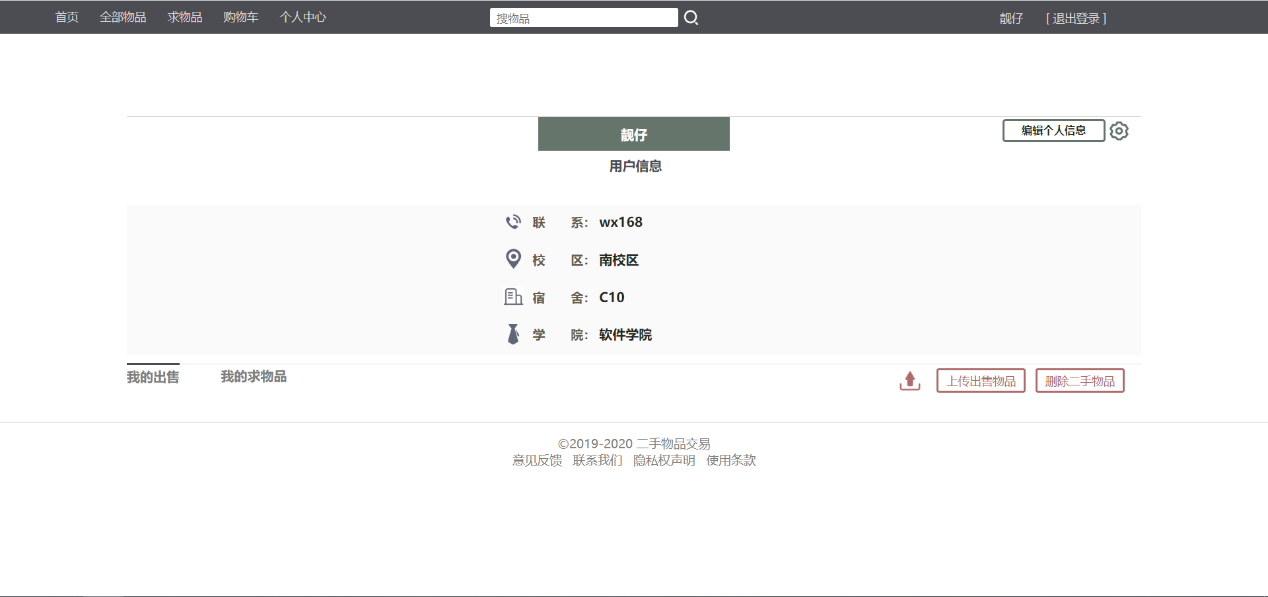
分析：

post方法发送用户id，密码，返回的operation\_code，是用户操作令牌，后端服务器生成，用于保障数据库信息安全。

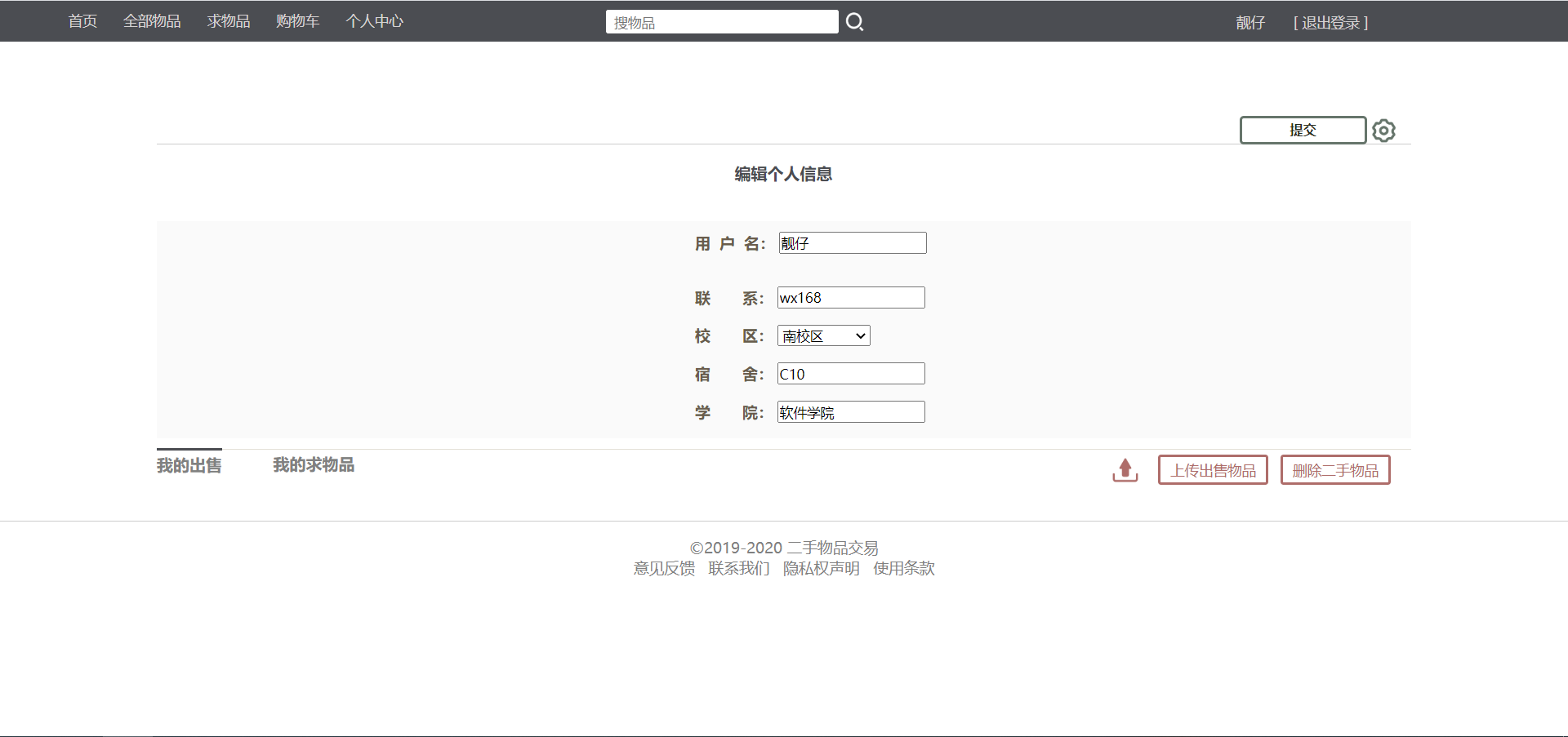
因为，如果前端进行用户操作（发布物品信息、线上交流），如果发送网络请求只需要用户id，那么任何人只需要获取别人的用户id就可以直接用别人的账户进行恶意操作。

所有，操作令牌可以有效防止这种情况。用户操作令牌，是每次用户登录时后端服务器生成的，用户前端保存一份，后端保存一份，进行用户操作时，会进行匹配，匹配成功才能进行用户操作。

### 2.个人信息展示与修改

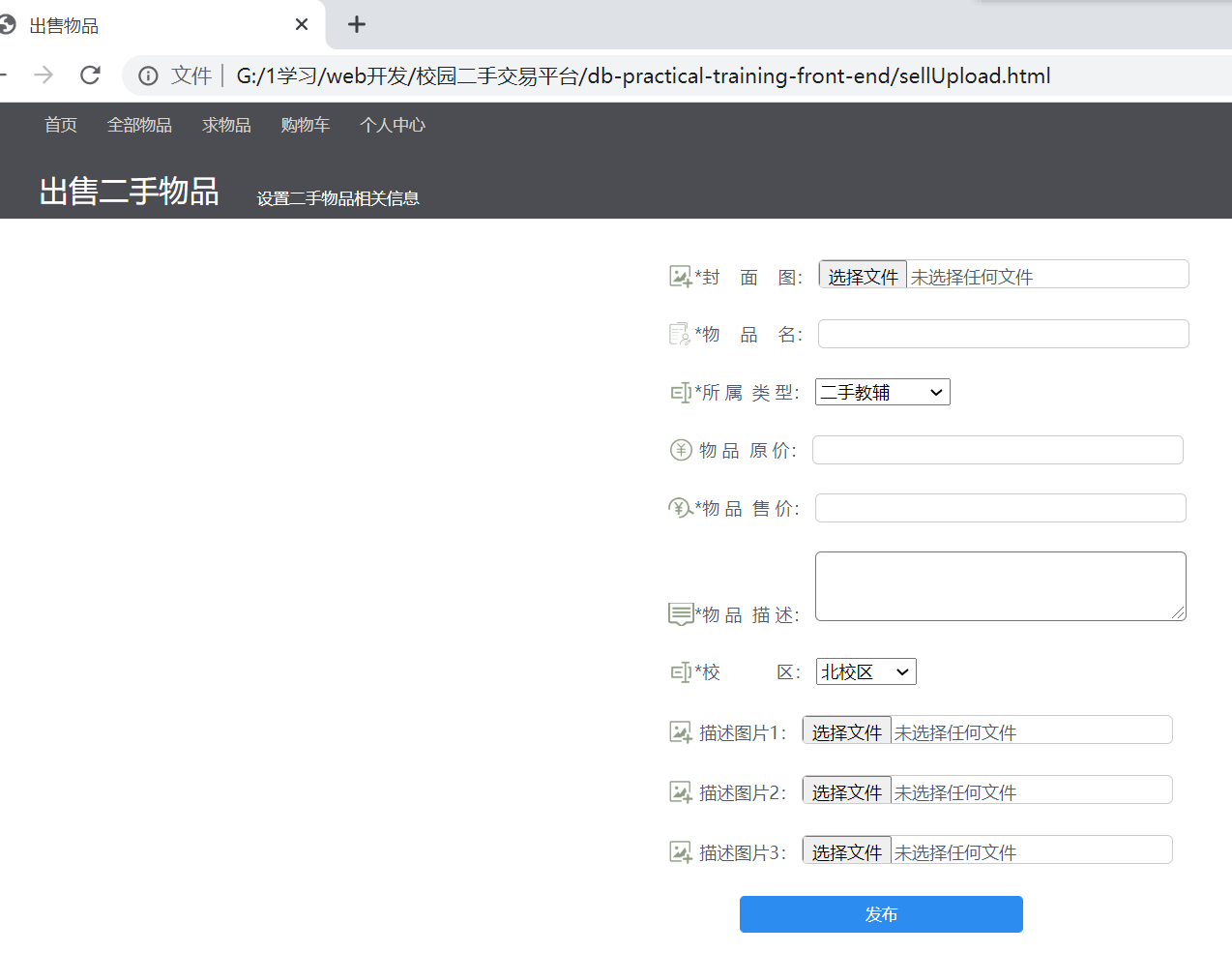
 

图：个人信息页



图：修改个人信息

### 3.物品的上架



19：点击上传出售物品图

部分html代码：

<**div class="container"**>  
 <**form class="upload-form" id="formContain"**>  
 <**p**>  
 <**span class="pic-icon"**>**&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;**</**span**>\*封**&nbsp;&nbsp;&nbsp;&nbsp;**面**&nbsp;&nbsp;&nbsp;&nbsp;**图：  
 <**input @change='add\_img' type="file"**>  
 *<!-- <img src="cover\_pic" id="cover\_pic"> -->* </**p**>  
  
 <**p**>  
 <**span class="author-icon"**>**&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;**</**span**>\*物**&nbsp;&nbsp;&nbsp;&nbsp;**品**&nbsp;&nbsp;&nbsp;&nbsp;**名：  
 <**input type="text" v-model="commodity\_name" name="commodity\_name"**>  
 </**p**>  
 <**p**>  
 <**span class="name-icon"**>**&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;**</**span**>\*所**&nbsp;**属**&nbsp;&nbsp;**类**&nbsp;**型：  
 <**select name="commodity\_type" v-model="commodity\_type"**>  
 <**option disabled value=""**>请选择</**option**>  
 <**option value="二手教辅"**>二手教辅</**option**>  
 <**option value="非教辅类书籍"**>非教辅类书籍</**option**>  
 <**option value="学习用具"**>学习用具</**option**>  
 <**option value="技能服务"** >技能服务</**option**>  
 <**option value="手机数码"**>手机数码</**option**>  
 <**option value="服饰/美妆"**>服饰/美妆</**option**>  
 <**option value="未开封食品"**>未开封食品</**option**>  
 <**option value="玩具/游戏交易"**>游戏交易</**option**>  
 <**option value="其他"**>其他</**option**>  
</**select**>  
 </**p**>  
  
 <**p**>  
 <**span class="orign-icon"**>**&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;**</**span**> 物**&nbsp;**品**&nbsp;&nbsp;**原**&nbsp;**价：  
 <**input type="text" v-model="commodity\_original\_price" name="commodity\_original\_price"**>  
 </**p**>  
 <**p**>  
 <**span class="price-icon"**>**&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;**</**span**>\*物**&nbsp;**品**&nbsp;&nbsp;**售**&nbsp;**价：  
 <**input type="text" v-model="commodity\_price" name="commodity\_price"**>  
 </**p**>  
  
 <**p**>  
 <**span class="desc-icon"**>**&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;**</**span**>\*物**&nbsp;**品**&nbsp;&nbsp;**描**&nbsp;**述：  
 <**textarea type="text" v-model="commodity\_description" name="commodity\_description"**></**textarea**>  
 </**p**>  
  
 <**p**>  
 <**span class="name-icon"**>**&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;**</**span**>\*校**&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;**区：  
 <**select name="area" v-model="area"**>  
 <**option disabled value=""**>请选择</**option**>  
 <**option value="北校区"**>北校区</**option**>  
 <**option value="南校区"**>南校区</**option**>  
 <**option value="国际校区"**>国际校区</**option**>  
</**select**>  
 </**p**>  
  
 <**p**>  
 <**span class="pic-icon"**>**&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;**</**span**> 描述图片1： <**input type="file" name="desc\_img1" @change='add\_desc\_img'**>  
 *<!-- <img src="desc\_img1" id="desc\_img1"> -->* </**p**>  
 <**p**>  
 <**span class="pic-icon"**>**&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;**</**span**> 描述图片2： <**input type="file" name="desc\_img2" @change='add\_desc\_img'**>  
 *<!-- <img src="desc\_img2" id="desc\_img2"> -->* </**p**>  
 <**p**>  
 <**span class="pic-icon"**>**&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;**</**span**> 描述图片3： <**input type="file" name="desc\_img3" @change='add\_desc\_img'**>  
 *<!-- <img src="desc\_img3" id="desc\_img3"> -->* </**p**>  
  
 <**i-button type="primary" @click="uploadToServer"**>发布</**i-button**>  
  
 </**form**>  
  
 *<!-- 模态窗口 -->* <**Modal title="View Image" v-model="modalVisible"**>  
 <**img :src="'https://o5wwk8baw.qnssl.com/' + imgName + '/large'" v-if="visible" style="width**: 100%**"**>  
 </**Modal**>  
  
  
</**div**>

分析：

以上代码是表单标签<form>的内容，

form中使用了<input type=”file”>输入图片，

<input type=”text”>输入文字，

<textarea>输入大段文本，

<select>和<option>表示选择器。

同时，每个标签，使用v-model绑定表单数据，用于上传服务器。

部分JavaScript代码：

uploadToServer:**function**(){  
 **if**(!**this**.checkIfLogin()||!**this**.checkContain()) **return  
  
 let** form = **new FormData**();  
 form.append(**'cover\_pic'**, **this**.**cover\_pic**,**this**.**cover\_pic**.name);  
 **if**(**this**.**desc\_img1**!=**''**) form.append(**'desc\_img1'**, **this**.**desc\_img1**, **this**.**desc\_img1**.name);  
 **if**(**this**.**desc\_img2**!=**''**) form.append(**'desc\_img2'**, **this**.**desc\_img2**, **this**.**desc\_img2**.name);  
 **if**(**this**.**desc\_img3**!=**''**)form.append(**'desc\_img3'**, **this**.**desc\_img3**, **this**.**desc\_img3**.name);  
 form.append(**'commodity\_name'**,**this**.**commodity\_name**)  
 form.append(**'commodity\_type'**,**this**.**commodity\_type**)  
 form.append(**'commodity\_original\_price'**,**this**.**commodity\_original\_price**)  
 form.append(**'commodity\_price'**,**this**.**commodity\_price**)  
 form.append(**'commodity\_description'**,**this**.**commodity\_description**)  
 form.append(**'area'**,**this**.area)  
 form.append(**'user\_id'**,**localStorage**.**user\_id**)  
 form.append(**'user\_login\_code'**,**localStorage**.**operation\_code**)  
  
 **const** msg = **this**.**$Message**.loading({  
 **content**: **'Loading...'**,  
 **duration**: 0  
 });  
 setTimeout(msg, 10000);  
 **var** self = **this  
 axios**.post(**localStorage**.serverUrl+**'commodity/addCommodity'**, form, {**headers**: {**'Content-Type'**: **'multipart/form-data'**}})  
 .then(**function** (response) {  
 **if**(response.data==**'0'**){  
 setTimeout(msg, 1);  
 self.**$Modal**.error({  
 **title**: **'发布结果'**,  
 **content**: **'发布失败，请检查内容后重试！'** });  
 }  
 **else**{  
 setTimeout(msg, 1);  
 self.**$Modal**.success({  
 **title**: **'发布结果'**,  
 **content**: **'发布成功！'** });  
 }  
 })  
 .catch(**function** (error) {  
 **console**.log(error);  
 });  
  
},

分析：

以上，是上传出售物品的前端接口，checkContain检测有无必填项没有填写，

然后设置需要发送的数据，

然后发送到服务器地址：**localStorage**.serverUrl+**'commodity/addCommodity'**

### 4.物品的分类展示



图：物品分类展示

前端JavaScript代码：

*//改变分类*change\_classify:**function**(index){  
 **if**(**this**.**chooseIndex**==index) **return**;  
 **this**.**chooseIndex**=index;  
 **this**.**alreadyGot**=0;  
 **this**.**count**=0;  
 **var** self=**this**;  
 *//index是classify的下标，发送请求，获取itemList信息.requestType统一设为sell* **if**(index==0) {  
 *//获取全部类型的全部物品信息，* **axios**.get(**localStorage**.serverUrl+**'commodity/getCommodities'**)   
 .then(**function** (response) {  
 **if**(response.data.**list**.**length**>0){  
 self.**itemList**=response.data.**list**;  
 self.**count**=response.data.count;  
 self.**alreadyGot**+=response.data.**list**.**length**;  
 }   
 **else if**(response.data.**length**==0) self.**noMore**=**true** *//console.log(response);* **console**.log( self.**itemList**);  
 })  
 .catch(**function** (error) {  
 **console**.log(error);  
 });  
 }  
 **else**{  
 *//否则获取某一类型的全部物品信息* **axios**.get(**localStorage**.serverUrl+**'commodity/queryCommoditiesByType?commodity\_type='**+self.**classify**[self.**chooseIndex**] )   
 .then(**function** (response) {  
 **if**(response.data.**length**>0) {  
 self.**itemList**=response.data ;  
 self.**alreadyGot**+=response.data.**length**;  
 }  
 **console**.log(response);  
 })  
 .catch(**function** (error) {  
 *// console.log(error);* });  
 }  
  
},

分析：当用户点击分类按钮，就会根据下标值index，获取用户点击的物品类型self.**classify**[self.**chooseIndex**]，发送http请求到服务器。

地址**localStorage**.serverUrl+**'commodity/queryCommoditiesByType**，**localStorage**.serverUrl是已经设定好的服务器主地址。

参数：**’?commodity\_type=’** +self.**classify**[self.**chooseIndex**]。

后端数据库查询语句，DML：？？？？？？？？？？？？？？？？？？？？？？？？？

### 5.交易流程

用户在各个页面，点击物品卡片即可进入详情页，

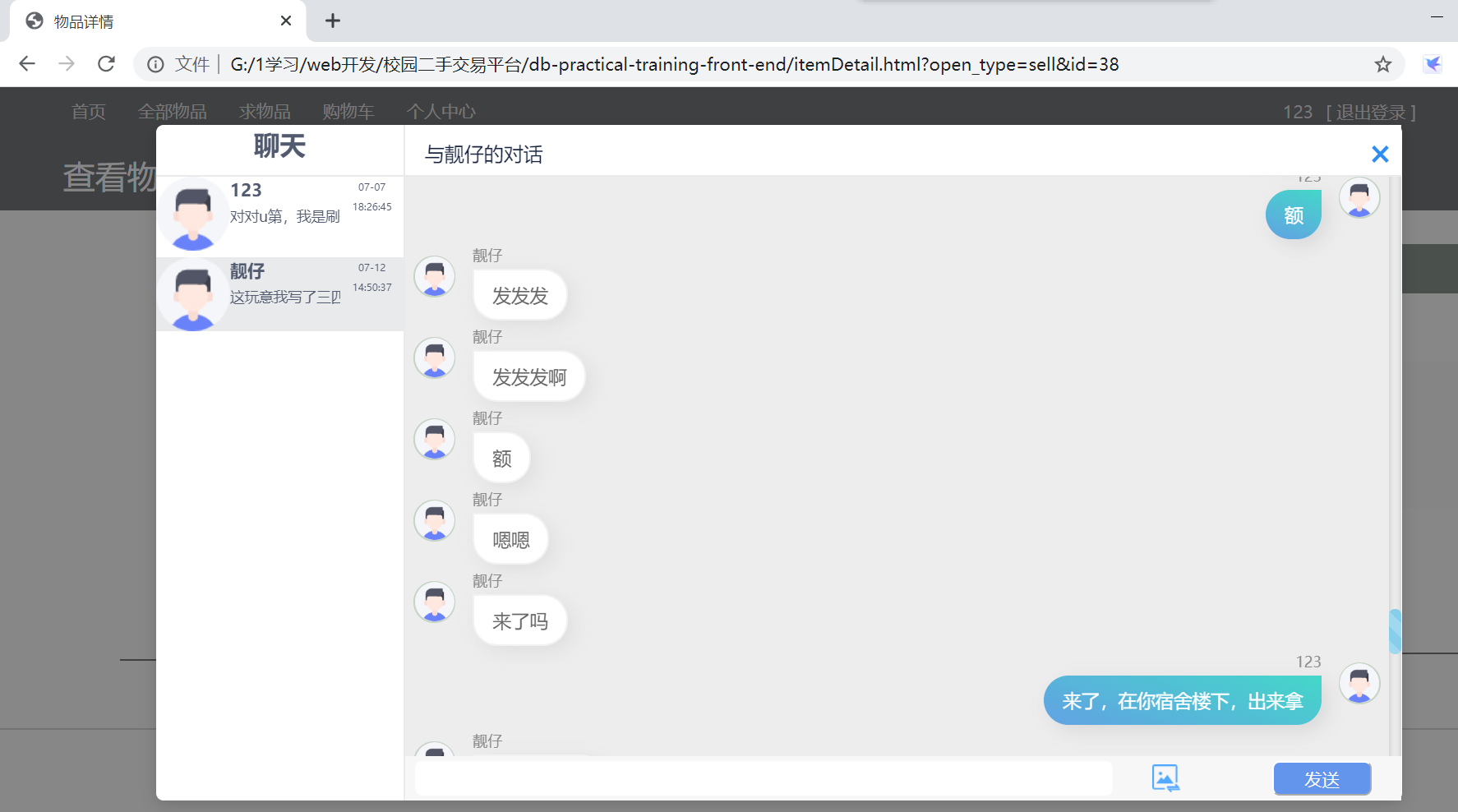


图：物品详情

可加入购物车。

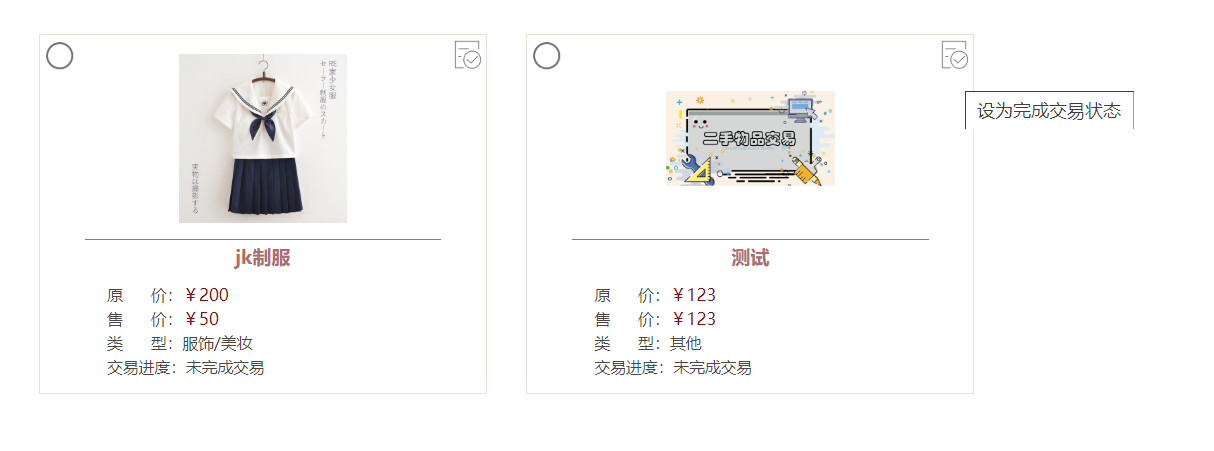
可根据右侧卖家的联系方式联系，进行线下交易。

也可点击“在线联系”按钮，进行在线联系。



图：在线联系

交易完成后，在个人中心，自己出售的物品下点击完成交易按钮，则该物品下架，进入完成交易列表，此后仅对创建者可见，不再在物品展示页面显示。



图：物品完成交易按钮



图：已完成交易的物品

物品详情页，前端请求代码：

axios.get(**localStorage**.serverUrl+**'commodity/getCommodityById?commodity\_id='**+self.**item\_id**)  
 .then(**function** (response) {  
 **console**.log(response);  
 *//创建者信息* self.**owner\_id**=response.data.map.**user\_id**;  
 self.**nickname**=response.data.map.**user\_name**;  
 self.**contact**=response.data.map.**tel**;  
 self.**area**=response.data.map.**user\_area**;  
 self.**dormitory**=response.data.map.**dormitory**;  
 self.**department**=response.data.map.**academy**;  
 *//物品信息* self.**title**=response.data.map.**commodity\_name**;  
 self.**cover**=response.data.map.commodity\_cover\_pic\_url;  
 self.**item\_type**=response.data.map.**commodity\_type**;  
 self.**now\_price**=response.data.map.**commodity\_price**;  
 self.**old\_price**=response.data.map.**commodity\_original\_price**;  
 self.**description**=response.data.map.**commodity\_description**;  
 self.**other\_pic**=response.data.imgList;  
 self.**cart\_status**=**'加入购物车'** })  
 .catch(**function** (error) {  
 **console**.log(error);  
 });

分析：

页面跳转时，传送了物品的id。

发送到服务器，参数的物品id，服务器返回详细信息，包括创建者信息（昵称，宿舍，校区，学院等）和物品信息（名称，封面图，描述，价格等）

后端数据库查询语句，DML：？？？？？？？？？？？？？？？？？？？？？？？？？

聊天框的实现，？？？