数据库系统实现总结报告

题目名称：航游集市——北航学子的交易&社交平台

小组成员：

- 张瀚文 22373321
- 魏鹏程 22373151
- 石伊聪 22371207

组内同学承担任务说明

学生姓名	子任务1 系统功能设计与数据库设计	子任务2 系统服务器端开发	子任务3 系统客户端开发	工作量占比
张瀚文	想法讨论 需求分析	接口文档敲定与测试	前端程序架构设计 前端架构实现 网页测试 前后端代码测试	33.3%
魏鹏程	想法讨论 需求分析	接口文档敲定与测试	前端布局美化 前端架构实现 网页测试 前后端代码测试	33.3%
石伊聪	想法讨论 需求分析 构造数据表	数据库连接 数据库存储定义与实现 后端架构设计与实现	前后端代码测试	33.3%

一、实现环境

1.1 前端

- 框架：Vue3
- 组件：element-plus等
- 实时通信：websocket
- 前后端交互：Axios

1.2 后端

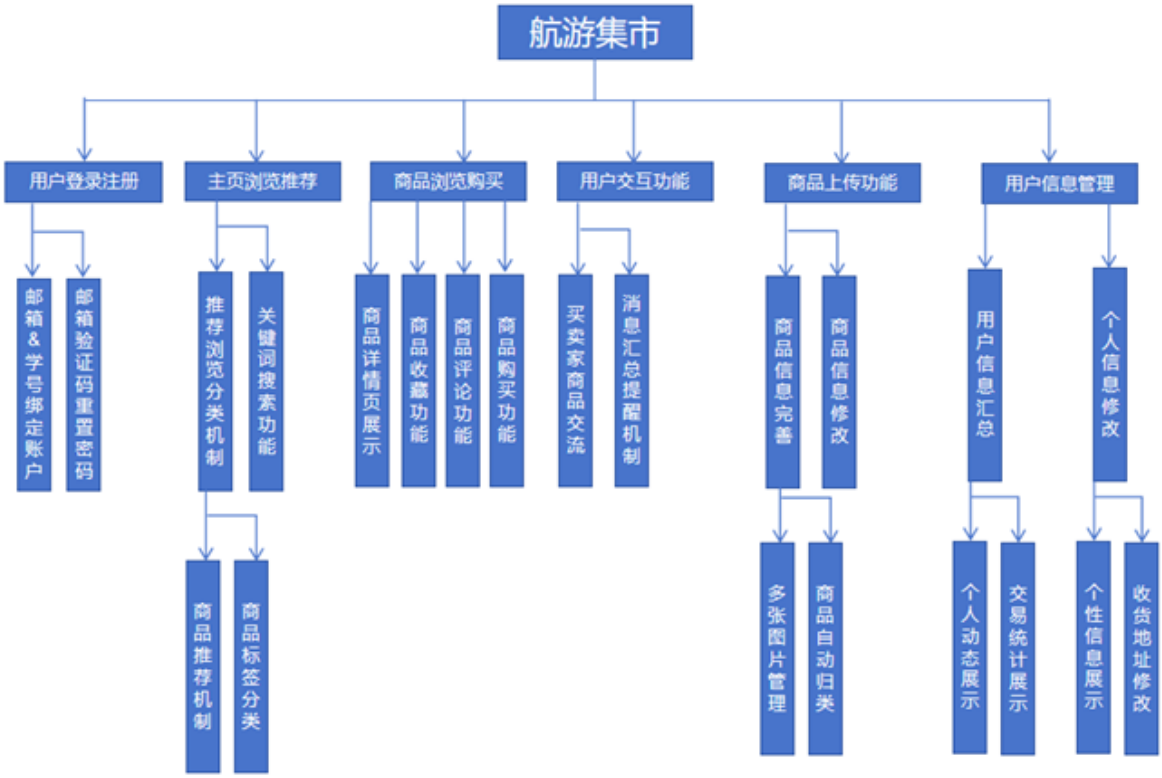
- 语言：C# .NET 9.0
- Web框架： ASP.NET Core
- 数据库：PostgreSQL 16.8
- 后端可跨平台部署，在 Windows 11、Ubuntu 24.04、MacOS 16下经过测试

二、系统功能结构图

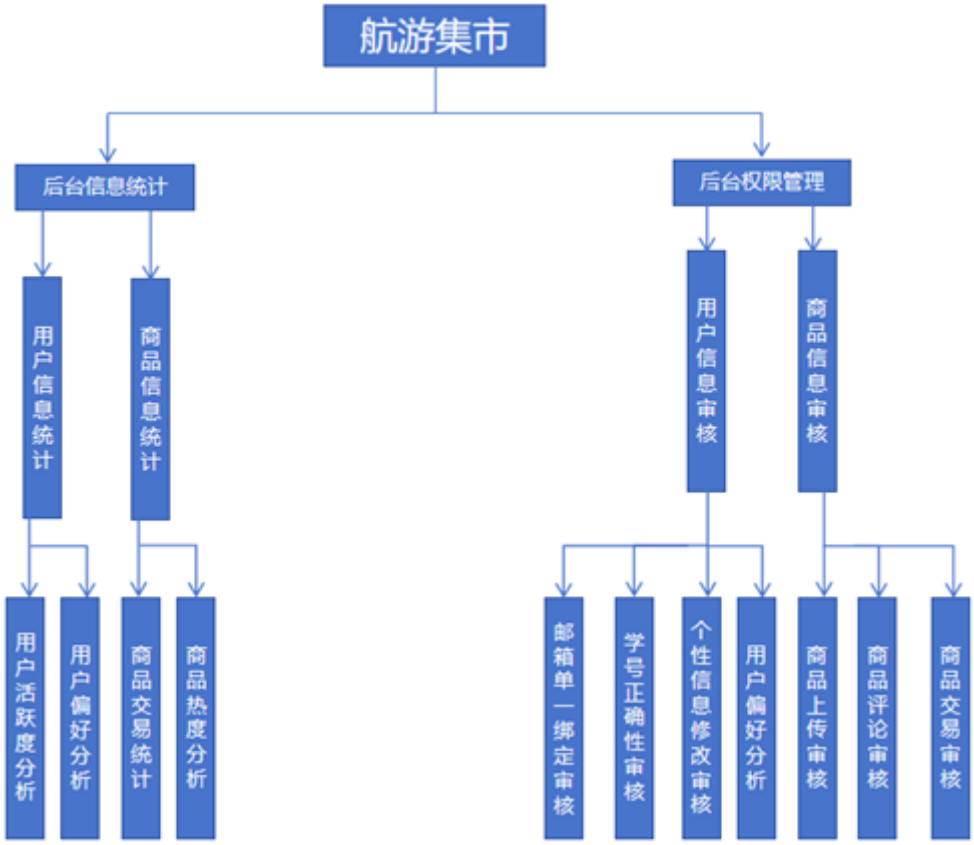
2.1 前端

本项目的系统功能可以从使用者不同身份、使用不同模块展开论述，即身份可分为普通用户和管理员；同时按照不同功能又可进行进一步划分，具体系统功能结构图如下：

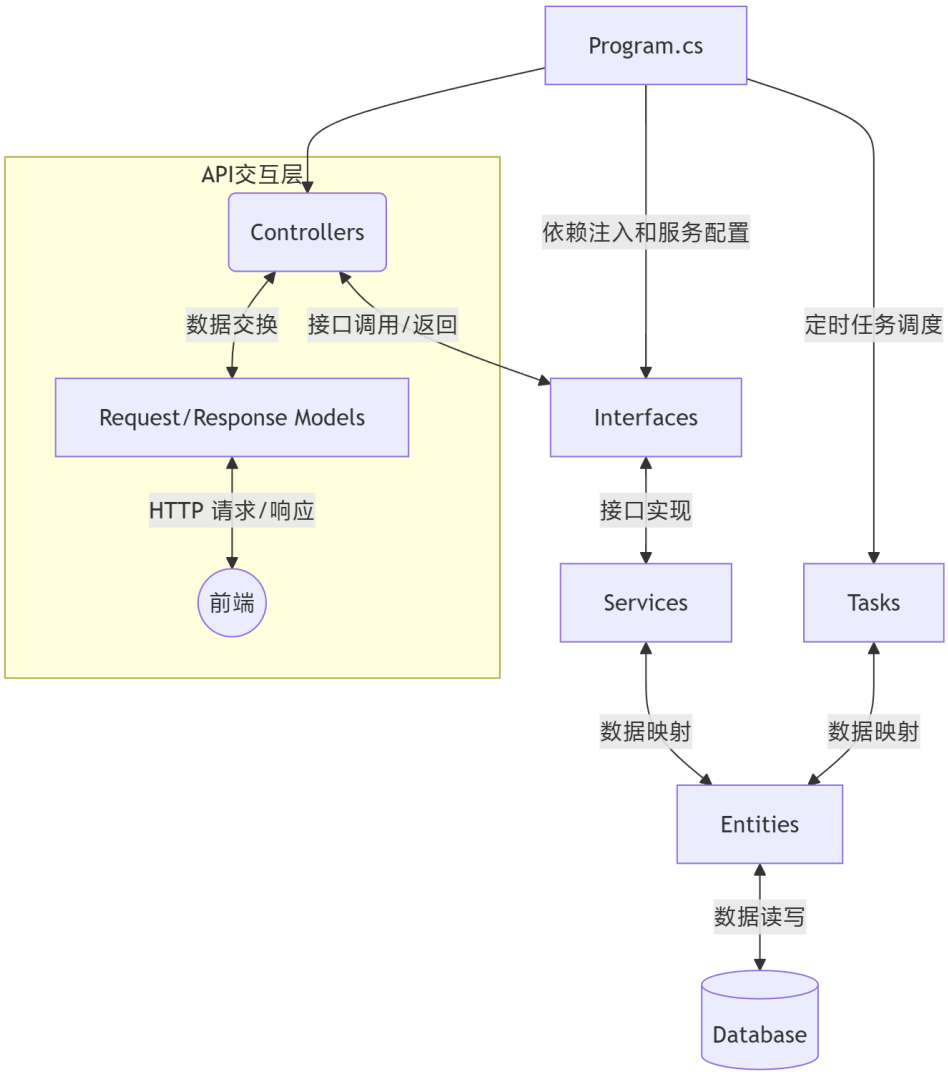
2.1.1 用户层面系统功能结构图：



2.1.2 管理员层面系统功能结构图



2.2 后端



三、数据库相关定义

3.1 基本表、完整性约束定义

3.1.1 用户表

字段名称	数据类型	可否为空	说明
id	VARCHAR(36)	否	主键
avatar	VARCHAR(255)	否	用户头像地址
intro	VARCHAR(255)	否	用户简介
nick_name	VARCHAR(100)	否	用户昵称
username	VARCHAR(100)	否	用户名
email	VARCHAR(100)	否	电子邮箱
student_id	VARCHAR(20)	否	学生ID
password	VARCHAR(100)	否	用户密码

字段名称	数据类型	可否为空	说明
status	INTEGER	是	用户状态
update_time	BIGINT	是	更新时间戳
create_time	BIGINT	是	创建时间戳
address	VARCHAR(10)	是	地址
check_nick_name	VARCHAR(100)	是	昵称审核内容
check_intro	VARCHAR(255)	是	简介审核内容
check_avatar	VARCHAR(255)	是	头像审核内容
check_status	INTEGER	是	审核状态

3.1.2 商品表

字段名称	数据类型	可否为空	说明
user_id	VARCHAR(36)	是	外键，user(id)
title	VARCHAR(100)	是	商品标题
id	VARCHAR(36)	否	主键
intro	TEXT	是	商品介绍
image	TEXT	是	商品图片
price	BIGINT	是	商品价格
original_price	BIGINT	是	商品原价
type_code	VARCHAR(10)	是	外键，product_type(type_code)
type_name	VARCHAR(20)	是	商品类型名称
post_type	INTEGER	是	发布类型
like_count	INTEGER	是	点赞数量
adcode	VARCHAR(10)	是	行政区划代码
province	VARCHAR(10)	是	省份
city	VARCHAR(10)	是	城市
district	VARCHAR(10)	是	区县
status	INTEGER	是	商品状态
create_time	BIGINT	是	创建时间戳
update_time	BIGINT	是	更新时间戳

3.1.3 消息表

字段名称	数据类型	可否为空	说明
id	VARCHAR(36)	否	主键
product_id	VARCHAR(36)	是	外键，product_info(id)
product_image	VARCHAR(255)	是	商品图片
from_user_id	VARCHAR(36)	否	外键，user(id)
from_user_avatar	VARCHAR(255)	是	发送方头像
from_user_nick	VARCHAR(100)	是	发送方昵称
to_user_id	VARCHAR(36)	否	外键，user(id)
to_user_nick	VARCHAR(100)	是	接收方昵称
to_user_avatar	VARCHAR(255)	是	接收方头像
create_time	BIGINT	是	创建时间戳
update_time	BIGINT	是	更新时间戳

3.1.4 评论表

字段名称	数据类型	可否为空	说明
id	VARCHAR(36)	否	主键
product_id	VARCHAR(36)	是	外键，product_info(id)
parent_id	VARCHAR(36)	是	外键，comment(id)
pub_user_id	VARCHAR(36)	是	外键，user(id)
pub_nickname	VARCHAR(100)	是	评论者昵称
parent_user_id	VARCHAR(36)	是	被回复者用户ID
parent_user_nickname	VARCHAR(255)	是	被回复者昵称
content	TEXT	是	评论内容
create_time	BIGINT	是	创建时间戳

3.1.5 订单表

id	VARCHAR(36)	否	主键
order_number	VARCHAR(32)	是	订单号
user_id	VARCHAR(36)	是	外键，user(id)
pay_price	BIGINT	是	支付金额

id	VARCHAR(36)	否	主键
pay_type_id	BIGINT	是	外键，payment_type(id)
pay_type_name	VARCHAR(200)	是	支付类型名称
order_status	INTEGER	是	订单状态
payment_pay_id	VARCHAR(36)	是	支付流水号
payment_status	INTEGER	是	支付状态
payment_type	VARCHAR(20)	是	支付方式
process_status	INTEGER	是	处理状态
time_create	TIMESTAMPTZ	是	创建时间
time_update	TIMESTAMPTZ	是	更新时间，需大于等于创建时间
time_finish	TIMESTAMPTZ	是	完成时间，需大于等于更新时间

3.1.6 角色表

字段名称	数据类型	可否为空	说明
id	VARCHAR(36)	否	主键
role_code	VARCHAR(20)	是	角色编码,唯一
role_name	VARCHAR(20)	是	角色名称
create_time	BIGINT	是	创建时间戳
update_time	BIGINT	是	更新时间戳

3.1.7 管理员表

字段名称	数据类型	可否为空	说明
id	VARCHAR(36)	否	主键
username	VARCHAR(100)	是	用户名
password	VARCHAR(100)	是	密码
name	VARCHAR(20)	是	姓名
role_id	VARCHAR(36)	是	外键，system_role(id)
role_code	VARCHAR(20)	是	角色编码
role_name	VARCHAR(20)	是	角色名称
create_time	BIGINT	是	创建时间戳
update_time	BIGINT	是	更新时间戳

3.1.8 商品类型表

字段名称	数据类型	可否为空	说明
id	VARCHAR(36)	否	主键
type_code	VARCHAR(10)	是	类型编码，唯一约束
type_name	VARCHAR(20)	是	类型名称
create_time	BIGINT	是	创建时间戳
update_time	BIGINT	是	更新时间戳

3.1.9 消息列表表

字段名称	数据类型	可否为空	说明
id	VARCHAR(36)	否	主键
chat_list_id	VARCHAR(36)	否	外键，chat_list(id)
from_user_id	VARCHAR(36)	否	外键，user(id)
to_user_id	VARCHAR(36)	否	外键，user(id)
from_user_nick	VARCHAR(100)	是	发送方昵称
to_user_nick	VARCHAR(100)	是	接收方昵称
content	TEXT	是	消息内容
send_time	BIGINT	是	发送时间戳
is_read	INTEGER	是	是否已读

3.1.10 支付表

字段名称	数据类型	可否为空	说明
id	VARCHAR(36)	否	主键
user_id	VARCHAR(36)	是	外键，user(id)
order_id	VARCHAR(36)	是	外键，payment_order(id)
payment_price	BIGINT	是	支付金额
payment_type	VARCHAR(30)	是	支付方式
payment_result_data	TEXT	是	支付结果数据
payment_time_start	VARCHAR(50)	是	支付开始时间
payment_time_expire	VARCHAR(50)	是	支付过期时间
payment_status	INTEGER	是	支付状态，默认值0

字段名称	数据类型	可否为空	说明
process_status	INTEGER	是	处理状态，默认值0
time_create	TIMESTAMPTZ	是	创建时间
time_update	TIMESTAMPTZ	是	更新时间，需大于等于创建时间
time_finish	TIMESTAMPTZ	是	完成时间，需大于等于更新时间

3.1.11 发布信息表

字段名称	数据类型	可否为空	说明
id	VARCHAR(36)	否	主键
user_id	VARCHAR(36)	是	外键，user(id)
product_id	VARCHAR(36)	是	外键，product_info(id)
create_time	BIGINT	是	创建时间戳
update_time	BIGINT	是	更新时间戳

3.2 索引的定义

```
-- 用户名索引
CREATE INDEX idx_user_username ON "user" (username);
-- 邮箱索引
CREATE INDEX idx_user_email ON "user" (email);
-- 用户发布的商品索引
CREATE INDEX idx_product_user_id ON product_info (user_id);
-- 聊天消息索引
CREATE INDEX idx_chat_list_time ON chat_message (chat_list_id, send_time);
-- 支付订单号索引
CREATE INDEX idx_payment_order_number ON payment_order (order_number);
-- 商品订单号索引
CREATE INDEX idx_product_order_number ON product_order (order_number);
-- 用户收藏索引
CREATE INDEX idx_collect_user ON product_collect (user_id, create_time
DESC);
-- 商品收藏索引
CREATE INDEX idx_collect_product ON product_collect (product_id);
-- 买家订单索引
CREATE INDEX idx_buyer_orders ON product_order (user_id, deal_status);
-- 卖家订单索引
CREATE INDEX idx_seller_orders ON product_order (product_user_id,
deal_status);
-- 支付状态索引
CREATE INDEX idx_order_pay_status ON product_order (pay_status);
```

四、系统安全性设计

系统设置了一般用户和管理员两种角色未登录的用户只能访问 /public 路径下的接口，以 User 身份登录的用户可以访问大部分业务接口，执行查询其他用户的公开商品、发布商品、评论、购买商品、删除或隐藏自己发布的商品等功能，以 System 或 Admin 身份登录的用户可以访问 /admin 路径下的接口，可以查询、删除和修改一般用户产生的数据、删除或禁用用户、审核用户发布的信息。

五、 存储过程、触发器和函数的代码说明

后端程序与数据库的交互通过调用`npgsql`库完成.

1. 首先根据数据表定义数据对象：

```
CREATE TABLE user_address (
    id VARCHAR(36) PRIMARY KEY,
    user_id VARCHAR(36) NOT NULL REFERENCES "user" (id),
    name VARCHAR(10),
    phone VARCHAR(20),
    address VARCHAR(255),
    create_time BIGINT,
    update_time BIGINT
);
```

```
    [Table("user_address")]
 public class UserAddress
 {
     [Key]
     [Column("id")]
     [StringLength(36)]
     public required string Id { get; set; }
     [Column("user_id")]
     [StringLength(36)]
     public required string UserId { get; set; }
     [Column("name")]
     [StringLength(10)]
     public required string Name { get; set; }
     [Column("phone")]
     [StringLength(20)]
     public required string Phone { get; set; }
     [Column("address")]
     [StringLength(255)]
     public required string Address { get; set; }
     [Column("create_time")]
     public long CreateTime { get; set; }
     [Column("update_time")]
     public long UpdateTime { get; set; }
 }
```

2. 然后在业务逻辑中构造对应的数据对象，添加到数据库上下文中：

```
public Result AddUserAddress(UserAddressObj req)
{
    var userid = _tokenService.GetCurrentLoginUserId();
    if (req.Address == null || req.Phone == null || req.Name == null)
```

```
            return Result.Fail(ResultCode.ValidateError);
        var userAddress = new UserAddress
        {
            Id = Guid.NewGuid().ToString(),
            UserId = userid,
            Address = req.Address,
            Phone = req.Phone,
            Name = req.Name,
            CreateTime = DateTimeOffset.UtcNow.ToUnixTimeSeconds(),
            UpdateTime = DateTimeOffset.UtcNow.ToUnixTimeSeconds(),
        };

        _dbContext.UserAddresses.Add(userAddress);
        var save = _dbContext.SaveChanges();
        if (save == 0)
            return Result.Fail(ResultCode.SaveError);
        return Result.Ok();
    }
```

3. 设置了一些触发器来维护数据完整性和业务逻辑:

```
CREATE OR REPLACE FUNCTION update_user_timestamp()
RETURNS TRIGGER AS $$
BEGIN
    NEW.update_time = EXTRACT(EPOCH FROM NOW()) * 1000;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_user_timestamp BEFORE
UPDATE ON "user" FOR EACH ROW
EXECUTE FUNCTION update_user_timestamp ();
```

六、主要技术和模块的论述

6.1 前端

前端采用Vue3框架，同时辅助对应版本的element-plus组件， 以快速构建响应式界面，提升开发效率和用户体验。Vue3的组合式API和响应式系统为开发者提供了更灵活和强大的工具来构建复杂的用户界面，而element-plus则提供了一套丰富的UI组件库，帮助开发者快速实现布局、导航、数据展示等常见功能，两者结合使得前端开发更加高效和便捷。

下面将从主要页面、前后端接口两个方面对项目前端部分进行说明：

6.1.1 主要页面

- 推荐页

  推荐页根据商品标签展示不同类型的商品供用户浏览、选择。

  推荐页将每个商品以卡片的形式封装，使用、<Waterfall>瀑布流组件进行商品卡片的展示。<Waterfall>组件支持水平和垂直的瀑布流布局，以适应不同的页面需求，当窗口大小变化时，布局会自动重排，无需手动调整，同时组件支持移动元素的动画过渡，使得布局调整时效果平滑。当用户点击商品卡片时，通过事件绑定语法@click触发对应的JavaScript代码，打开商品详情页面。推荐页还设置了分类的菜单栏，每当用户点击一个商品类别时，会触发@click切换商品条目。

- 动态页

动态页主要使用element-plus中的时间轴组件<el-timeline>进行商品动态的展示，还通过css文件设置了加载动画，以实现更好的用户体验。<el-timeline>可以拆分成多个按照时间戳排列的activity，时间戳是其区分于其他控件的重要特征；时间轴组件还可以根据实际场景自定义节点的尺寸、颜色，或直接使用图标；并且能够适应不同的屏幕尺寸和布局需求。

除此之外，仍然设置了用户点击商品触发@click显示商品详情窗口

- 发布页

  发布页用于给用户发布商品，用户可以选择上传图片、标题、简介、分类、价格等。其中上传图片使用element-plus中的<el-upload>组件实现该功能，而标题、简介类文本信息使用<el-input>组件，价格、分类等可以选择或者输入的使用<el-select>组件实现。

  当用户点击确认提交时，首先会触发规则检查，检查必填项是否为空。若存在必填项未填写，则发出提醒，提交失败。反之，发布成功，数据传递给后端。

- 商品详情页

  商品详情页用于展示商品详情。左侧使用element-plus中的<el-carousel>走马灯组件进行图片轮播，展示商品图片；右侧展示发布商品的用户头像和昵称、商品简介和商品评论。用户通过点击“收藏按钮”可收藏商品或取消收藏商品；通过在输入框输入评论即可发送评论。当用户点击“我想要”按钮时，会触发聊天界面。聊天界面使用element-plus中的<el-drawer>抽屉组件实现。顶部会显示所要询问的商品，下部有输入框和发送按钮，通过websocket实现了实时通信。

- 消息页

  消息页展示用户接收和发送的消息，在顶部“我的消息”会显示未读消息数量，点击消息列表中的一项，用<el-drawer>实现的聊天侧边框会弹出，用户可随时查看消息以及发送消息。

- “我的”页

  “我的”页用于展示用户个人信息以及各类商品记录。

  用户可以点击“修改资料”来修改个人信息，包括昵称、头像、个人简介、密码等。修改资料页面同样使用<el-drawer>实现。修改头像使用<el-upload>组件实现，剩下的文本信息类使用<el-input>组件实现。类似的，用户还可以点击“添加地址”增添现有地址。

  用户的商品记录与推荐页类似，使用<Waterfall>瀑布流组件实现，除此之外，和推荐页不同的是，商品记录部分通过v-if判断商品状态，进而有对应的标签和按钮显示——如一个商品如果已经上架且还没有卖出，则会显示“已上线”的标签和“下架”按钮。

6.1.2 前后端接口

本项目数据的前后端交互使用Axios实现，Axios是一个基于 Promise 的 HTTP 客户端，用于浏览器和 node.js 环境。它提供了一个简单的 API，用于向 HTTP 服务器发送请求并处理返回的响应。其中，GET请求用于从服务器检索数据，POST请求用于提交数据到服务器，PUT请求用于更新服务器上的资源，DELETE请求用于从服务器删除资源。

本项目在api文件夹下将http请求封装为不同的方法，在页面实现中只需要调用对应的方法就可以发送对应的请求，易于代码复用和理解。

封装的api根据功能可分为以下几类：user类，负责用户信息等的接收和传送；product类，负责商品信息等的接收和传送；chat类，用于更新消息列表等；comment类，用于管理商品评论。例如，当想要获取商品信息时，api文件里会将对应的请求封装为getProductInfo方法，那么vue文件里只需要调用这个方法，即可发送请求，获取数据。

实时通信功能使用原生的Websocket API和后端配合实现。在utils的Websocket文件中，配置由后端提供的URL后websocket即可正常使用。其中new Websocket新建通信，ws.send(msg)用于发送信息，ws.close()为关闭通信。

6.2 后端

6.2.1 技术选型

- **编程语言**：C# .NET 9.0，一个稳定且功能强大的编程语言，适合构建企业级应用。
- **Web框架**：ASP.NET Core，一个高性能的框架，用于构建模块化、跨平台的Web应用。
- **数据库**：PostgreSQL 16.8，一个开源的对象关系数据库系统，以其稳定性和强大的功能而闻名。

6.2.2 后端架构设计

后端采用分层架构设计，包括数据访问层（DAL）、业务逻辑层（BLL）和表现层（API）。这种分层设计使得代码更加模块化，易于维护和扩展。

6.2.3 数据库设计

- **表结构设计**：根据业务需求设计了用户表、商品表、消息表、评论表、订单表等，每个表都有详细的字段定义和约束，确保数据的完整性和一致性。
- **索引定义**：为了提高查询效率，定义了多个索引，如用户名索引、邮箱索引、用户发布的商品索引等。

6.2.4 安全性设计

- **角色权限控制**：系统设置了一般用户和管理员两种角色，不同角色可以访问不同的接口。
- **接口安全**：通过身份验证和授权机制保护接口，确保数据安全。

6.2.5 存储过程、触发器和函数

后端程序与数据库的交互通过调用npgsql库完成。定义了一些存储过程和触发器来维护数据完整性和业务逻辑，例如，更新用户时间戳的触发器。

6.2.6 主要功能实现

- **用户管理**：实现用户注册、登录、信息修改等功能。
- **商品管理**：包括商品的发布、审核、查询、购买等业务逻辑。
- **消息系统**：实现用户之间的实时通信，包括消息的发送、接收和读取状态管理。
- **订单处理**：处理订单的创建、支付、状态更新等逻辑。

6.2.7 跨平台部署

后端可跨平台部署，在Windows 11、Ubuntu 24.04、MacOS 16下经过测试，确保了应用的可移植性。

6.2.8 API设计

后端提供了丰富的API接口供前端调用，包括用户信息、商品信息、订单处理等，通过Axios实现前后端的数据交互。

通过上述内容，我们可以看到后端部分在技术选型、架构设计、数据库设计、安全性设计、存储过程、主要功能实现以及跨平台部署等方面都有详细的规划和实现，确保了整个系统的稳定性和可扩展性。

七、系统运行实例

本项目系统功能大致可分为用户登录注册、商品推荐浏览、商品购买、个性动态展示、用户交互功能、商品发布功能、个人信息展示与修改功能，大致运行实例如下：

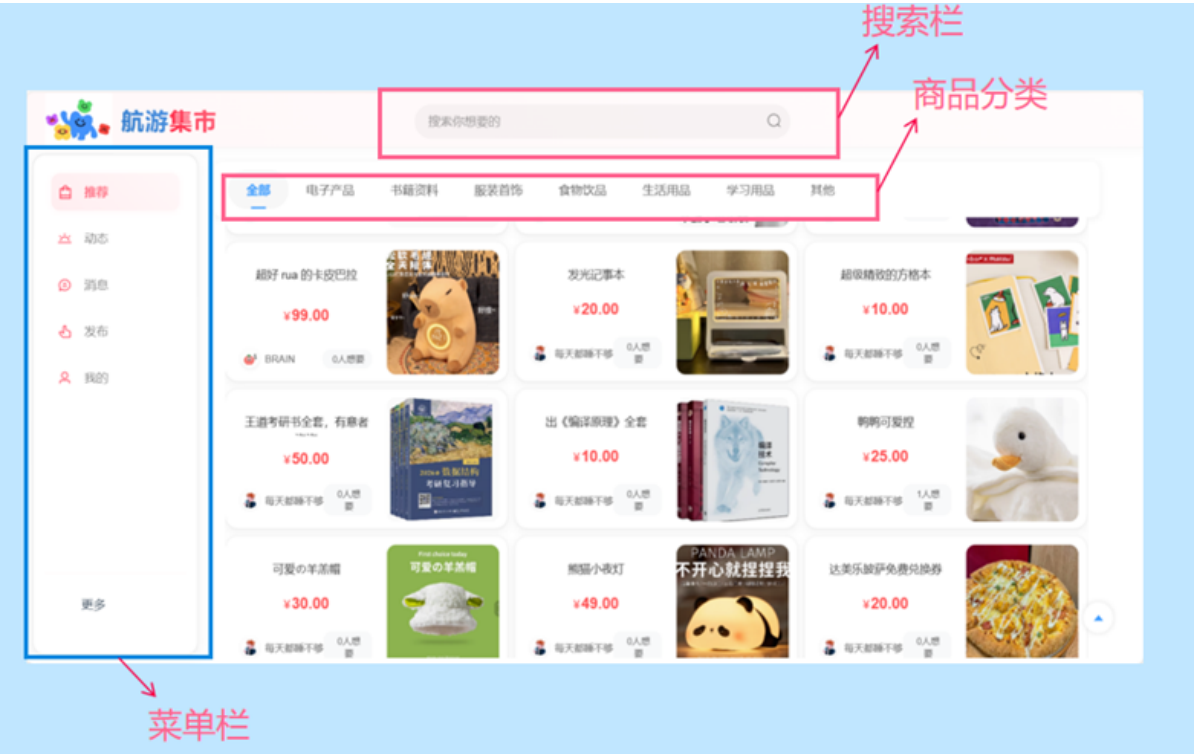
7.1 用户登录注册

用户登录界面如下，输入账号和密码即可进行登录，同时密码设有保护机制：

点击右侧卡片即可跳转到用户注册页面和重置密码页面：

7.2 商品推荐浏览

登录成功后会跳转到网站的主页面。本项目结合个人最近浏览记录、最热商品排序、最新商品排序对用户进行综合个性化推荐。同时主页面设有详细的商品分类，点击即可弹出对应的分类商品浏览，帮助用户进行范围浏览。另外顶部设有搜索栏，输入相关关键词即可搜索对应商品，帮助用户实现精准查询：



（注：这里的蓝色背景只是用于在文档中凸显网页轮廓，实际运行并没有此项背景）

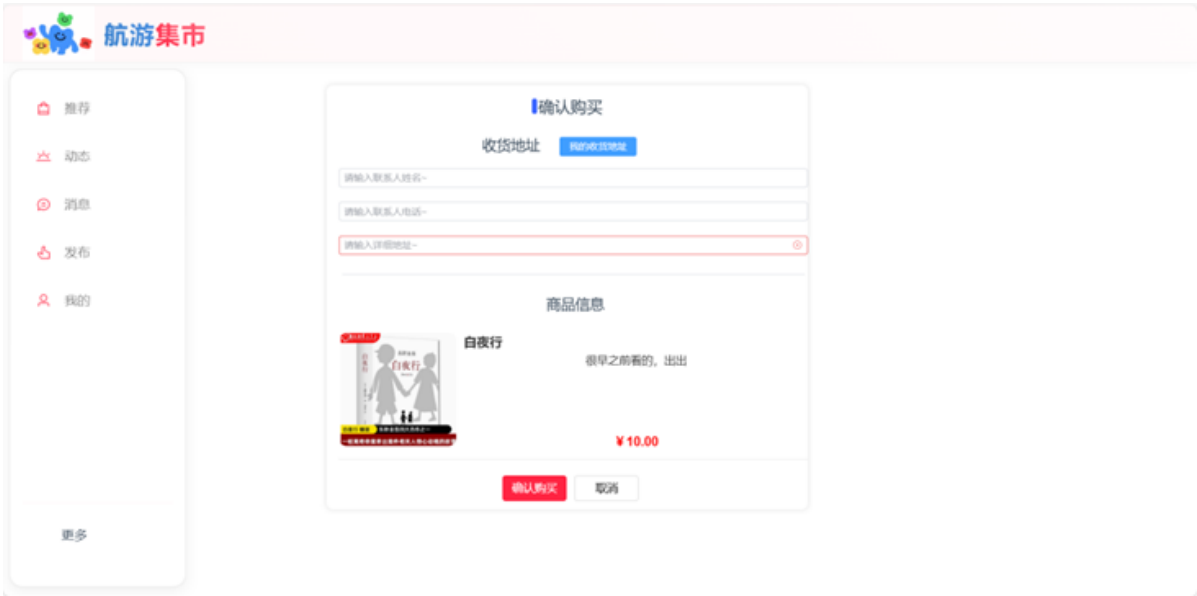
7.3 商品购买

这里以购买整个商品的详细流程为基础，介绍各个功能模块的实现。首先点击某个商品对应的卡片即可弹出商品的详细信息，这里我们没有选择跳转到新的页面，弹出卡片的方式使得整个浏览过程更加快捷、丝滑，能够提升用户体验。点开后左侧会展示商品的图片，多张图片也可以实现自动循环播放；右侧展示有发布者的信息、商品简介。

除了能够对某个商品进行收藏，我们还为每个商品都设置了对应的讨论区，在增强社交性的基础上也更方面买卖家之间的交流协商：

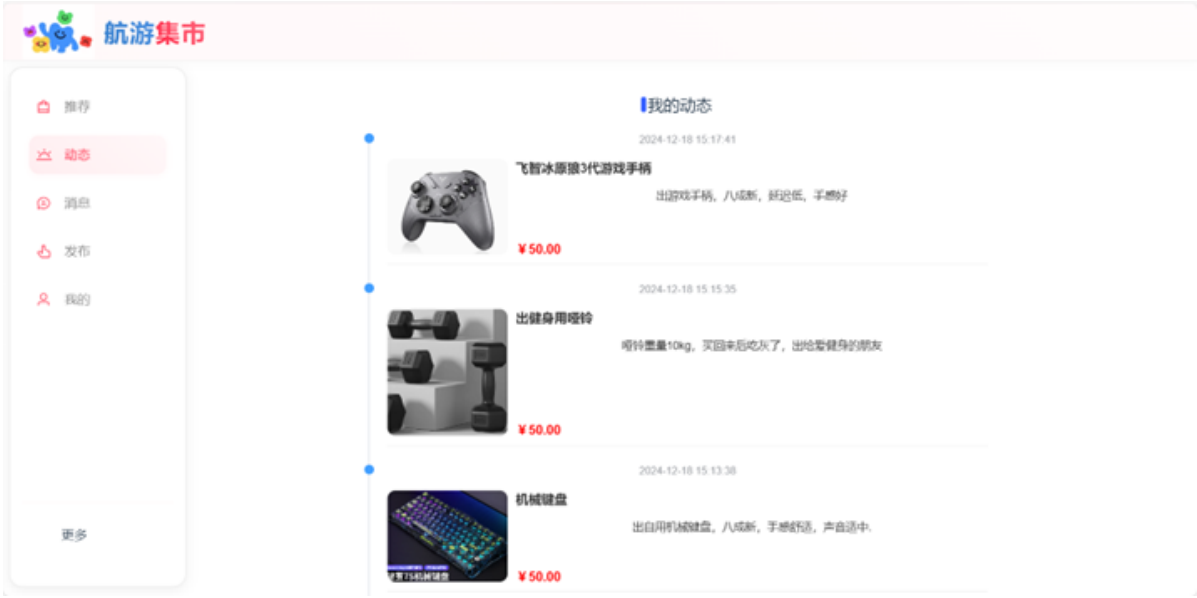
随后点击上侧的“我想要”即可弹出和卖家的对话窗口，在这里双方可以进行实时沟通，同时聊天栏顶部还设置有商品信息，让双方知道是在针对哪一个商品进行交流：

在交流完毕后，点击“购买”即可跳转到对应的完善信息页面和支付页面，点击“确认购买”即可完成购买：



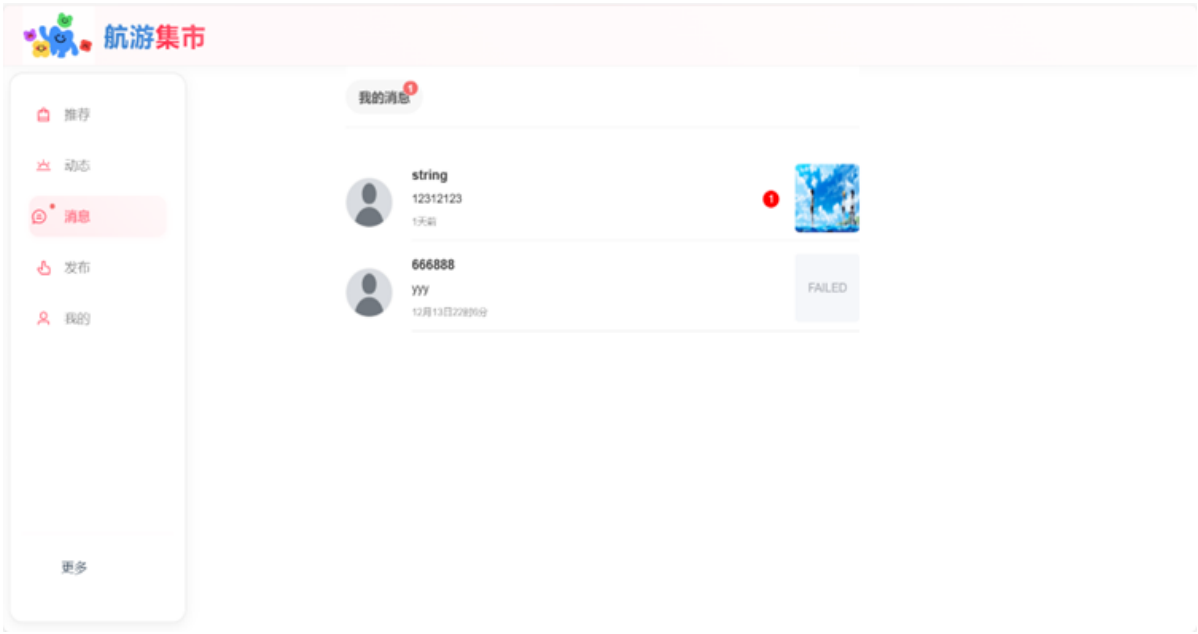
7.4 个性动态展示

在个性动态展示部分以时间脉络展示了用户最近的商品发布/购买等等操作，是对用户在网站活动近况的详细分析，有利于用户对最近的活动进行分析记录。以时间脉络整理商品交易情况，清晰直观：



7.5 用户交互功能

在消息页面展示了用户所有的聊天记录，实现不同用户、不同商品的分区讨论。同时当用户收到买家发来的消息时会进行实时的消息提示，点击即可进入和买家详细的聊天框进行进一步沟通：

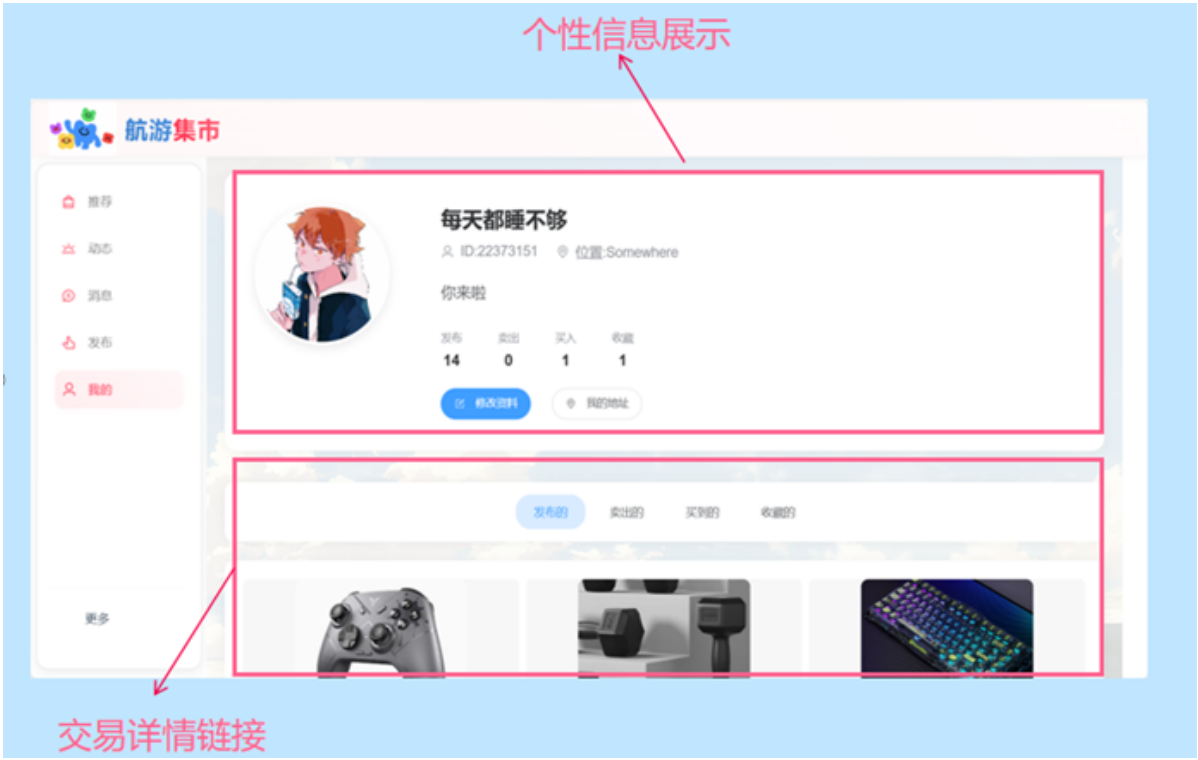


7.6 商品发布功能

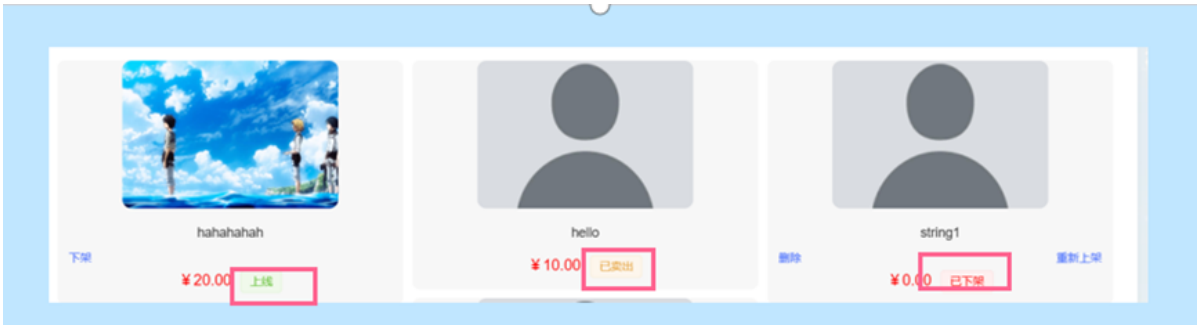
在发布页面用户可以通过完善商品信息进行商品的发布，值得一提的是本项目支持用户发布多张图片，实现商品详情页的循环播放。发布后经管理员审核即可出现在其他用户的推荐页面：

7.7 个人信息展示与修改

最后来到用户的个性页面，点击后会展示出个人的信息、交易记录等详情。值得一提的是我们对用户页面的商品展示和推荐页面的商品展示进行了区分。在个人页面下的商品清晰地展示了个人商品目前的状态：已上线、已下架、已卖出以及一键删除，实现了用户对个人商品更为方便清晰的管理：



商品状态展示如下：



同时在个人主页面也可以对自己的个人信息、收货地址进行完善修改：

以上就是整个用户层面系统功能的简要概述。总结来讲，我们实现了整个从浏览商品、商品交流到购买商品的买家功能；更实现了发布商品、商品管理到信息接收、完成出售的卖家功能。在这里每个人都可以是买家，也可以是卖家，在保证用户功能丰富性的基础上更借助于商品评论、消息汇总等模块保证项目拥有足够完善的社交性，最终集成为一款具有校园特色的、交易社交为一体的集市平台。

八、源程序简要说明

8.1 前端部分

前端部分的文件树如图所示：

- Api文件夹里的各个文件封装了不同用途的前端与后端的接口，包括product类、comment类、user类、image类、chat类等。
- Assets下的image文件夹里存放的是网页展示中的背景图片，以及用户默认头像等不需要与后端交互的图片。
- Components下设三个文件，包括Chat.vue，Comment.vue，UserProduct.vue。其中Chat.vue是聊天界面的实现，Comment.vue负责每个商品的评论的显示，UserProduct.vue则主要管理用户界面“我的”页面里的商品展示。这些放在Components里的vue文件作为组件会在多个页面复用。
- Router文件夹下的index.js负责网页的路由管理。
- Store文件夹进行用户信息的集中管理和响应式更新。
- Utils文件夹下的convert.js文件负责将用户或商品信息里的一些不便于展示的数据转换为易于理解的数据，如时间，价格等等；eventBus.js手动创建了事件总线模式用于组件间通信；request.js封装了axios相关操作，便于api文件夹内的接口使用；screenUtils用于处理页面滚动相关的计算；而websocket.js则是网页中实时通信的基础。
- Views文件夹下的文件是网页主要页面的实现。Login.vue页面实现了登录界面；Index.vue制作了网页的侧边导航栏和头部的header；Dashboard页面是”推荐”页面；followtrend.vue是”动态”页面；main.vue制作了商品详情页面，其会在其他多个页面被引用；message下的message-list.vue负责消息列表的具体展示，而message.vue则负责整个消息界面的排版和布局；release.vue负责“发布”界面；my.vue用于展示用户的个人信息界面；order下的文件则负责订单有关的界面，如订单创建，确认支付等；admin则是后台管理的具体实现。

九、收获和体会

通过本次课程作业，本组成员从初步设立方案到具体实施，实现了从构想到实践的完整流程。整个过程中在知识方面增强了对整个网页全栈开发的理解和熟练度，为成员未来的学习打下基础；在工作方面进一步认识到明确分工、正常密切交流的重要性。

其中一位组员的感悟如下：作为本次数据库大作业的前端选手，通过本次大作业的完成，我全方位学习了Vue3框架的原理及其使用，掌握了web前端设计和开发的技术，以及web前端如何和后端交互。自从完成了大作业的开发，现在再上网冲浪颇有点“看山不是山”的味道，每每见到一个观感舒适的网页总会琢磨应该使用什么组件实现。同时在完成过程中还深刻掌握了各种Vue3出现的bug的原理，如何调试发现bug，以及如何解决bug。可以说是真正接触了以后工作或许会用到的技能。希望以后能积累更多的web前端开发经验，也希望下次能做一些后端的工作，应用一下数据库中学到的理论知识，做一个全栈程序员。