天津大学

《前端开发技术实践》

组　　号 20

学　　院 智算学部

专　　业 软件工程

年　　级 2023 级

2025 年　6 月　15 日

目录

前端项目

实验目的

技术整合能力培养：掌握使用 Vue 框架结合 VueCli 构建前端项目的方法，实现与 Servlet 后端的数据交互，理解前后端分离架构下的开发模式。

前端组件化开发实践：通过开发首页、商家列表、订单确认等页面组件，掌握 Vue 组件的封装、路由配置及状态管理，提升组件化开发能力。

数据交互与渲染实现：利用 Axios 实现前端与后端的 AJAX 通信，完成 JSON 数据的接收与解析，并通过 Vue 的响应式机制实现数据的动态渲染。

用户交互功能开发：实现购物车操作、地址管理、订单确认等业务逻辑，掌握前端表单验证、状态切换及页面跳转等交互功能的开发方法。

跨域问题处理：理解 CORS 跨域资源共享机制，掌握前端在前后端分离开发中处理跨域请求的方法。

前端工程化实践：熟悉前端项目的搭建流程，包括依赖安装、配置文件编写及项目部署，提升前端工程化开发能力。

实验过程

(一) 前端架构设计与基础组件开发

1. 工程目录结构规划
   a) 在 src 目录下创建 components（公共组件）、views（页面组件）、router（路由）、common.js（工具函数）等目录，遵循前后端分离架构。
   b) 在 main.js 中配置全局参数：设置 axios.defaults.baseURL 指向后端接口，挂载工具函数到 Vue.prototype，便于组件调用。
2. 公共组件开发
   a) Footer 组件：创建底部导航栏，包含 “首页”“订单” 等标签，通过$router.push 实现页面跳转。
   b) 工具函数封装：在 common.js 中实现日期获取、本地存储操作（setSessionStorage/getLocalStorage）等功能，方便组件调用。

(二) 核心页面组件开发

1. 首页（Index.vue）
   a) 布局实现：包含搜索栏、分类导航、推荐商家列表，使用 Flex 布局和响应式设计，图片路径引用 assets 目录资源。
   b) 交互逻辑：监听滚动事件，实现搜索栏固定定位；点击分类图标时，通过$router.push 传递 orderTypeId 参数至商家列表页。
2. 商家列表页（BusinessList.vue）
   a) 数据获取：通过$axios.post 调用后端 BusinessController/listBusinessByOrderTypeId 接口，根据分类 ID 获取商家列表。

b) 购物车数量显示：若用户已登录，调用 CartController/listCart 接口查询购物车中该商家的商品数量，并渲染到页面。

3. 商家详情页（BusinessInfo.vue）
   a) 食品列表渲染：调用 FoodController/listFoodByBusinessId 接口获取商品信息，通过 v-for 循环渲染列表。
   b) 购物车操作：实现“+/-”按钮交互，通过$axios 调用 CartController/saveCart 和 updateCart 接口更新购物车状态。
4. 订单确认页（Orders.vue）
   a) 地址管理：从本地存储 localStorage 获取用户地址，未选择时跳转至地址列表页；调用 DeliveryAddressController 接口管理地址数据。
   b) 订单创建：调用 OrdersController/createOrders 接口生成订单，自动计算总价并跳转至支付页。

（三） 路由与跨域配置

1. 路由表配置
   a) 在 router/index.js 中定义路由规则，如/businessInfo 指向商家详情页，/orders 指向订单确认页，启用 history 模式避免 URL 携带。
   b) 添加路由守卫：在 main.js 中通过 router.beforeEach 判断用户登录状态，未登录时禁止访问订单相关页面。
2. 跨域问题处理
   a) 前端无需额外配置，后端通过 CorsFilter 设置 Access-Control-Allow-Origin 头，允许 http://localhost:8081 跨域请求。

（四） 功能测试与调试

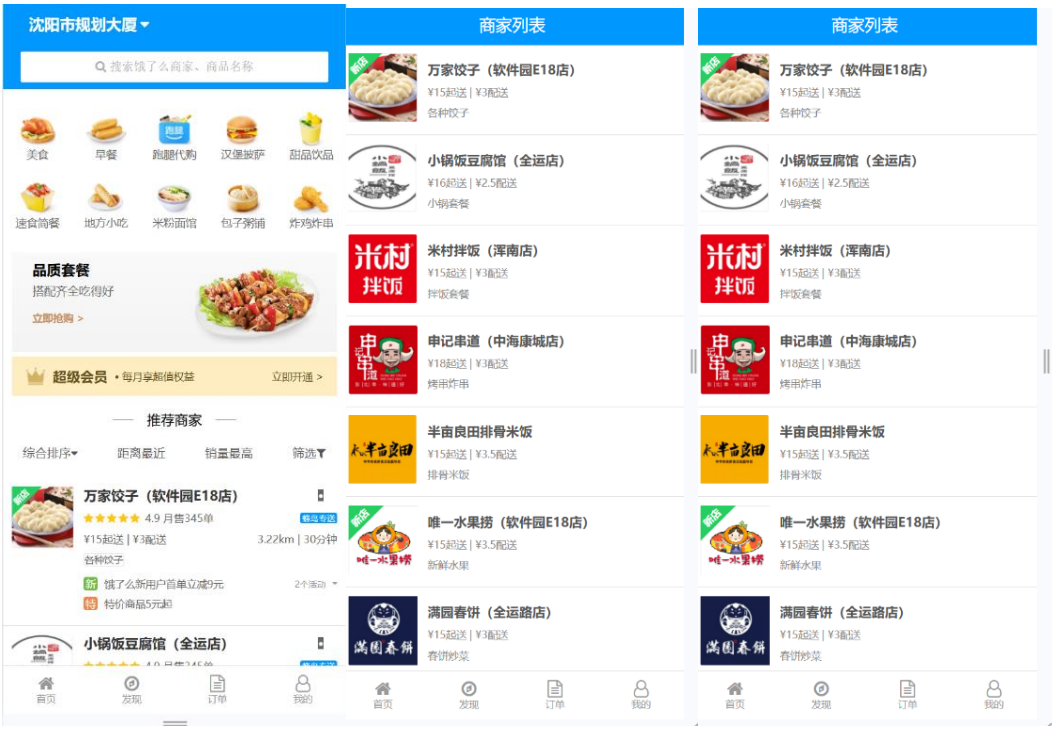
1. 接口联调
   a) 使用$axios 发送请求，通过 then/catch 处理响应，如调用 UserController/getUserByIdByPass 验证登录状态。
   b) 利用浏览器开发者工具检查网络请求，确保参数传递与 JSON 解析正确。
2. 状态管理测试
   a) 验证购物车数量同步：添加商品后刷新页面，通过 localStorage 和后端接口双重校验数据一致性。
   b) 测试订单流程：从选餐、确认地址到支付，确保订单状态（orderState）正确更新。

（五） 项目部署与打包

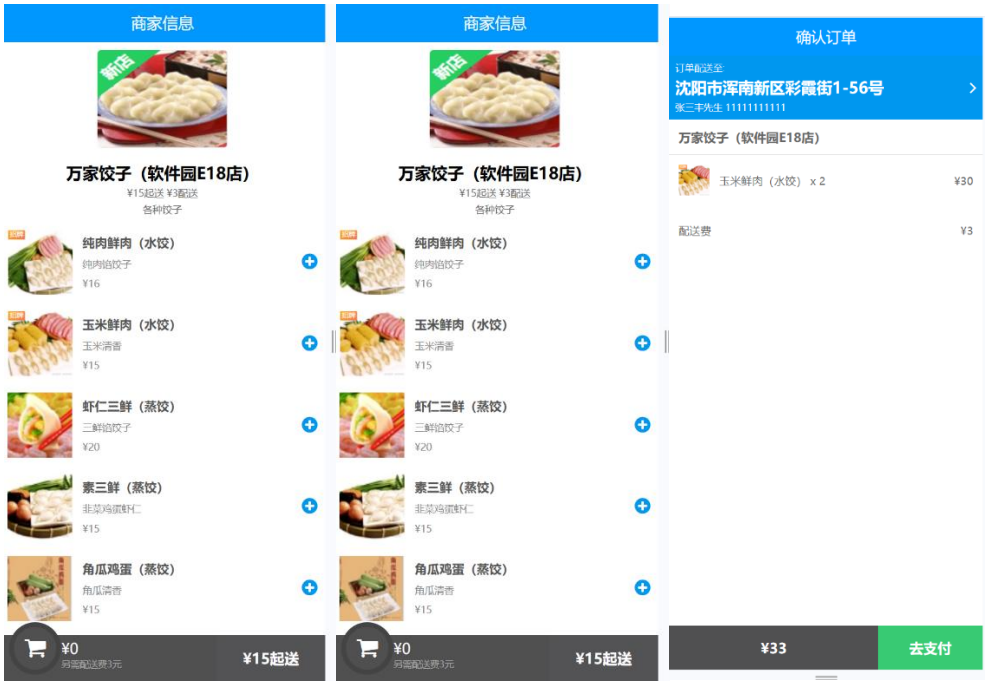
1. 本地调试：执行 npm run serve 启动前端服务，访问 http://localhost:8081 测试功能 -。
2. 前后端联调：在后端部署的前提下启动前端服务，访问 http://localhost:8081 测试功能

实验结果

首页展示，重点实现了各食品类型的商家列表功能



点击商家可查看商品，并可将商品添加至购物车



可以正常结算订单，并跳转到支付界面，如图示，网页会自动计算商品价格

对于未支付的订单，自动保存，并在“我的订单”界面显示，实现了用户登录注册功能

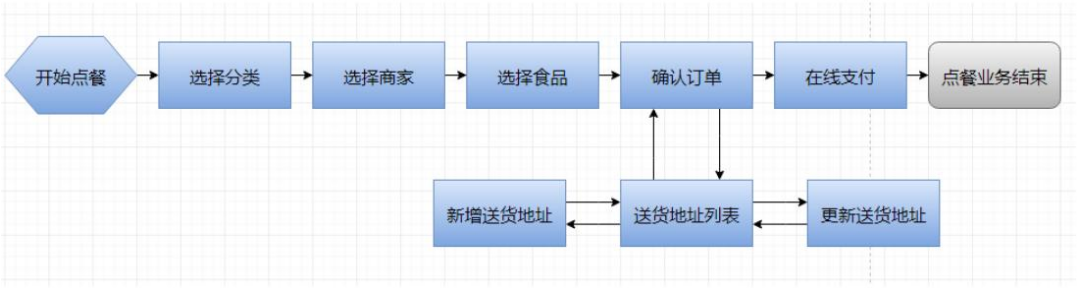
提交订单示需要选择送货地址。用户可以添加送货地

Servlet 后端

项目概述

本项目参照饿了么官网网页版制作，专注于点餐业务线功能，采用前后端分离架构，前端使用 Vue.js 框架，后端使用 Servlet 技术，通过 AJAX 进行前后端数据交互，旨在帮助学习者掌握相关开发技能。

需求分析



图表 1 业务流程图

1. 用户管理：用户可以通过手机号和密码进行注册和登录，系统会验证用户信息的合法性。
2. 商家展示：用户登录后，可以浏览不同分类下的商家和食品信息。
3. 商品购买：用户可以将喜欢的食品加入购物车，也可以修改购物车中的食品数量或删除食品。
4. 订单管理：用户可以确认订单信息，提交订单，并查看订单状态。
5. 地址管理：用户可以添加、修改和删除收货地址，方便订单配送。

项目设计

1. 环境搭建：
2. 开发工具 ： IDEA
3. JDK ： 配置 JDK 21
4. Tomcat ： 配置 Tomcat 8.5
5. 数据库 ： MySQL（创建对应数据库及表结构）
6. 系统架构：
   a) 前端：Vue CLI + Vue Router + Axios

b) 后端：Java Servlet + JDBC + MySQL
c) 通信：RESTful API + JSON 数据格式

7. 数据库设计
a) business：存储所有商家信息
b) food：存储每个商家所拥有的所有食品信息
c) cart：存储每个用户的购物车中的食品信息
d) deliveryaddress：存储每个用户的所有送货地址信息
e) orders：存储每个用户的所有订单信息
f) orderdetailet：存储每个订单中所订购的所有食品信息
g) user：存储所有用户信息

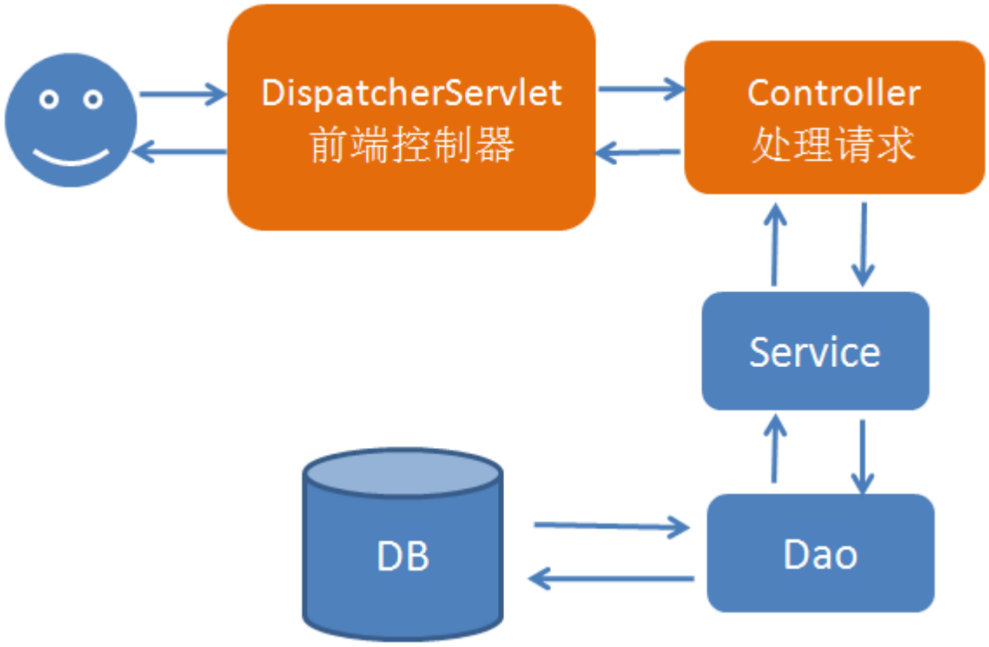
8. 项目结构

```
backend-servlet/
├── reference/
├── src/
│   └── com.neusoft.elm/
│       ├── controller/
│       │   ├── BusinessController.java
│       │   ├── CartController.java
│       │   ├── DeliveryAddressController.java
│       │   ├── FoodController.java
│       │   ├── OrdersController.java
│       │   └── UserController.java
│       ├── dao/
│       │   ├── Impl/
│       │   ├── BusinessDao.java
│       │   ├── CartDao.java
│       │   ├── DeliveryAddressDao.java
│       │   ├── FoodDao.java
│       │   ├── OrderDetailDao.java
│       │   ├── OrdersDao.java
│       │   └── UserDao.java
│       ├── filter/
│       │   └── CorsFilter.java
│       ├── framework/
│       │   └── DispatcherServlet.java
│       ├── po/
│       │   ├── Business.java
│       │   ├── Cart.java
│       │   ├── DeliveryAddress.java
│       │   ├── Food.java
│       │   ├── OrderDetail.java
│       │   ├── Orders.java
│       │   └── User.java
│       ├── service/
│       │   ├── impl/
│       │   │   ├── BusinessServiceImpl.java
│       │   │   ├── CartServiceImpl.java
│       │   │   ├── DeliveryAddressServiceImpl.java
│       │   │   ├── FoodServiceImpl.java
│       │   │   ├── OrdersServiceImpl.java
│       │   │   └── UserServiceImpl.java
│       │   ├── BusinessService.java
│       │   ├── CartService.java
│       │   ├── DeliveryAddressService.java
│       │   ├── FoodService.java
│       │   ├── OrdersService.java
│       │   └── UserService.java
│       └── util/
│           ├── CommonUtil.java
│           └── DBUtil.java
```

项目实现

1. MVC 架构解决方案：

本实验采用基于 servlet 的简易 MVC 架构，前端控制器(DispatcherServlet)部分采用直接请求整体路径，再将整体路径分割为所需的类名和方法名，减少了前后端交互的次数。



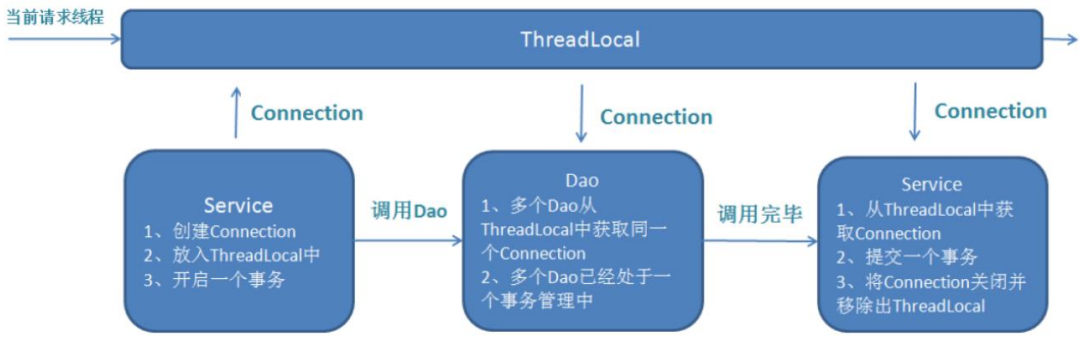
2. 事务处理解决方案：

a) Connection 的创建与销毁放在 service 层。

b) 为了保证在同一次请求处理的线程中，service 层和 dao 层都共用同一个 Connection 对象，需要将 Connection 对象放入 ThreadLocal 中。service 层和 dao 层使用的 Connection 对象一律从 ThreadLocal 获取。

c) dao 层不再处理异常，dao 层产生的异常将直接抛给 service 层进行处理。

d) dao 层负责关闭 PreparedStatement 和 ResultSet，service 层负责关闭 Connection。



3. 其他解决方案：
   a) 本项目中，Servlet 中使用 Jackson 将 java 对象或集合转换为 json 对象或数组后，返回前端。
   b) 对于图片，由于图片较小，本项目通过 Base64 编码方式将图片的二进制数据转化为字符串形式，便于存储、读取和处理。
4. 接口实现:
   a) BusinessController
      i. listBusinessByOrderTypeId:根据点餐分类编号查询所属商家信息
      ii. getBusinessById：根据商家编号查询商家信息
   b) FoodController
      i. listFoodByBusinessId：根据商家编号查询所属食品信息
   c) CartController
      i. listCart：根据用户编号查询用户所有购物车信息；根据用户编号和商家编号查询用户购物车中某个商家的所有购物车信息
      ii. saveCart：向购物车表中添加一条记录
      iii. updateCart：根据用户编号、商家编号、食品编号更新数量
      iv. removeCart：根据用户编号、商家编号、食品编号删除购物车表中的一条食品记录；根据用户编号、商家编号删除购物车表中的条条记录
   d) DeliveryAddressController
      i. listDeliveryAddressByUserId:根据用户编号查询所属送货地址
      ii. getDeliveryAddressById：根据送货地址编号查询送货地址
      iii. saveDeliveryAddress：向送货地址表中添加一条记录
      iv. updateDeliveryAddress:根据送货地址编号更新送货地址信息
      v. removeDeliveryAddress:根据送货地址编号删除一条记录
   e) OrdersController
      i. createOrders:根据用户编号、商家编号、订单总金额、送货地址编号向订单表中添加记录并获取订单编号，批量添加订单明细，删除购物车数据
      ii. getOrdersById:根据订单编号查询订单信息（包含商家

信息和订单明细信息）

iii. listOrdersByUserId：根据用户编号查询此用户的所有订单信息

f) UserController

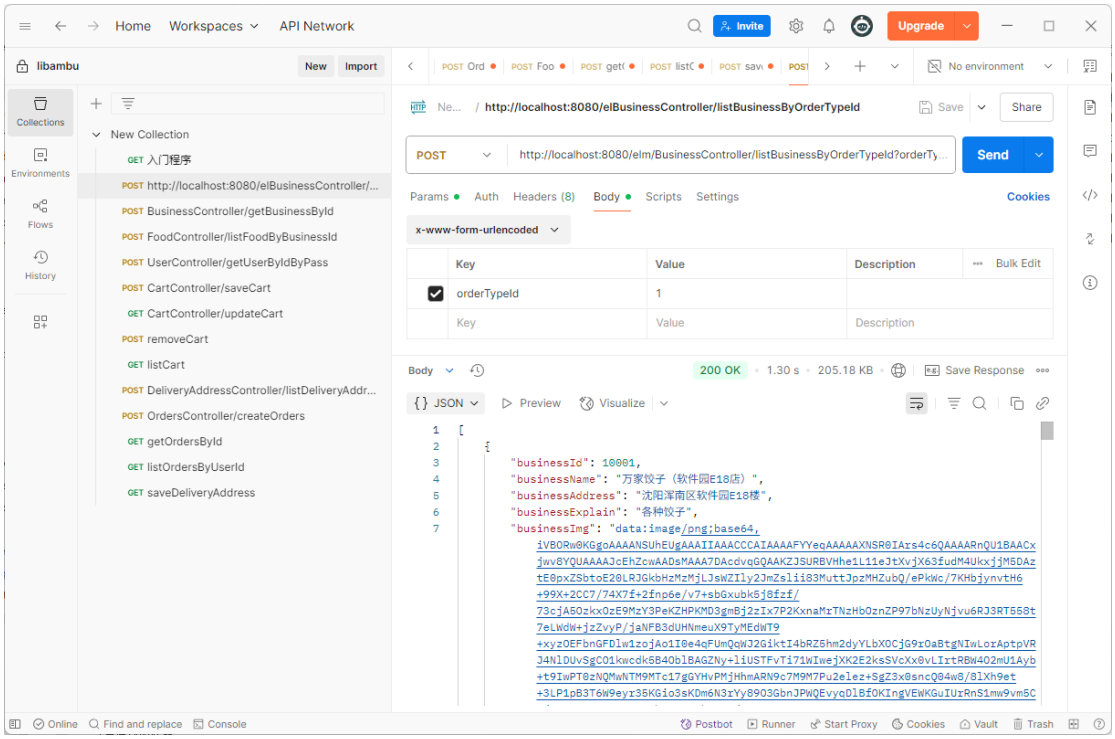
i. getUserByIdByPass:根据用户编号与密码查询用户信息

ii. getUserById：根据用户编号查询用户表返回的行数

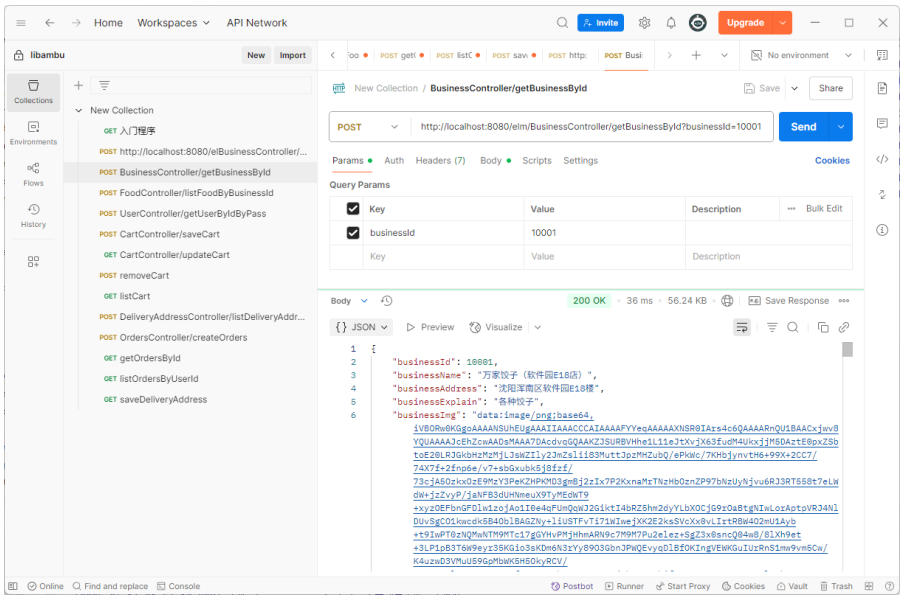
iii. saveUser：向用户表中添加一条记录

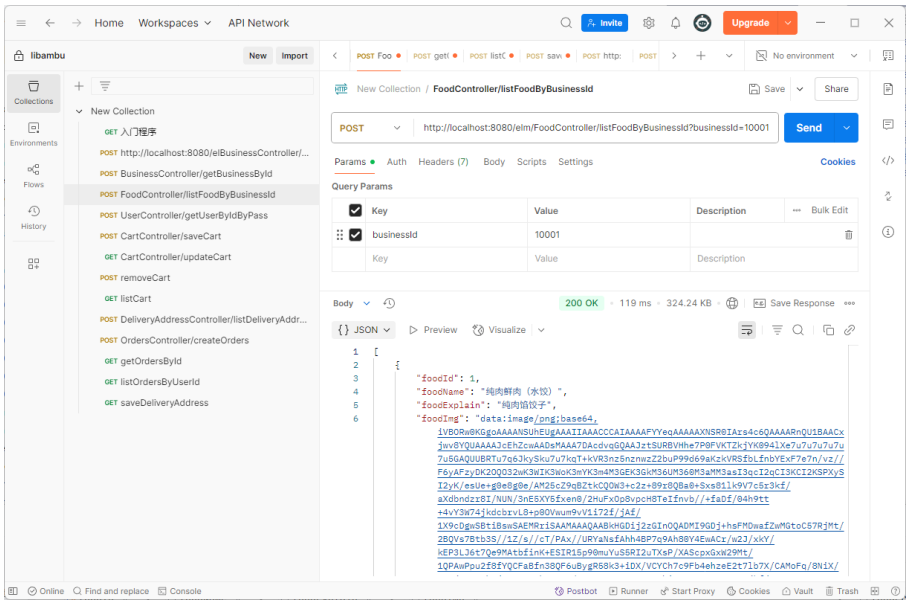
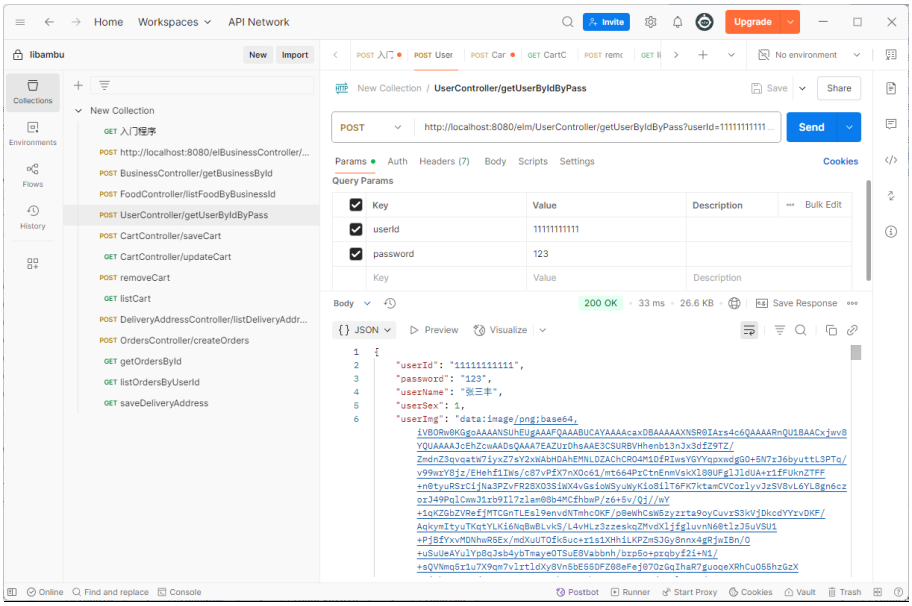
测试结果

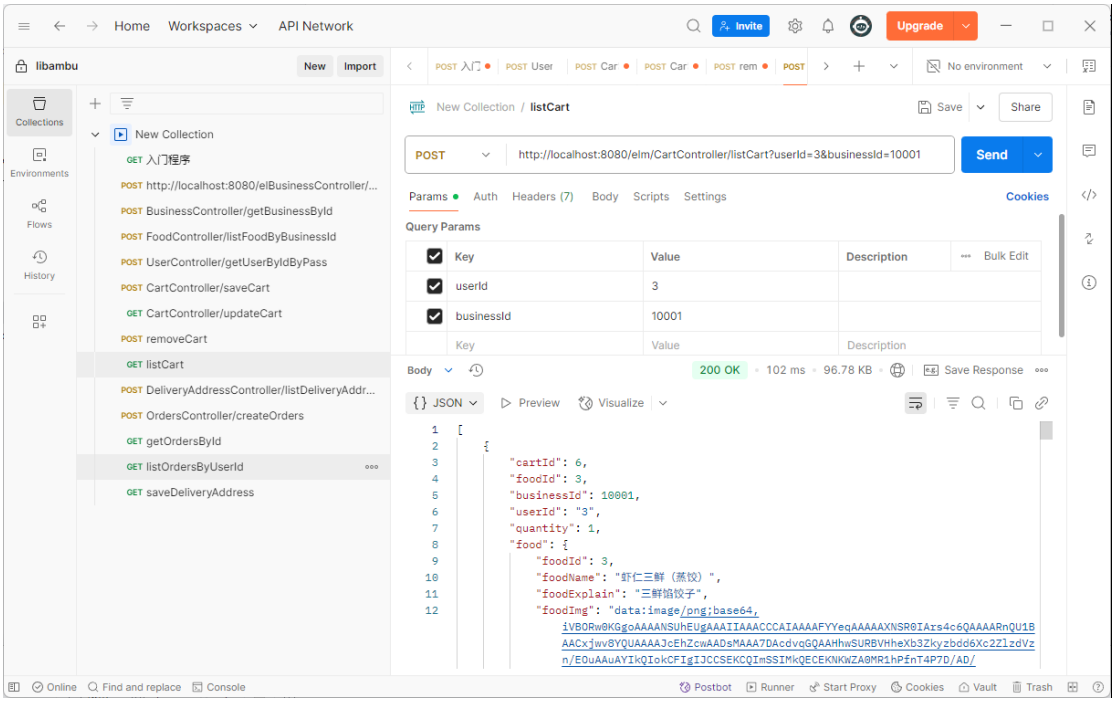
1. 后端接口测试结果

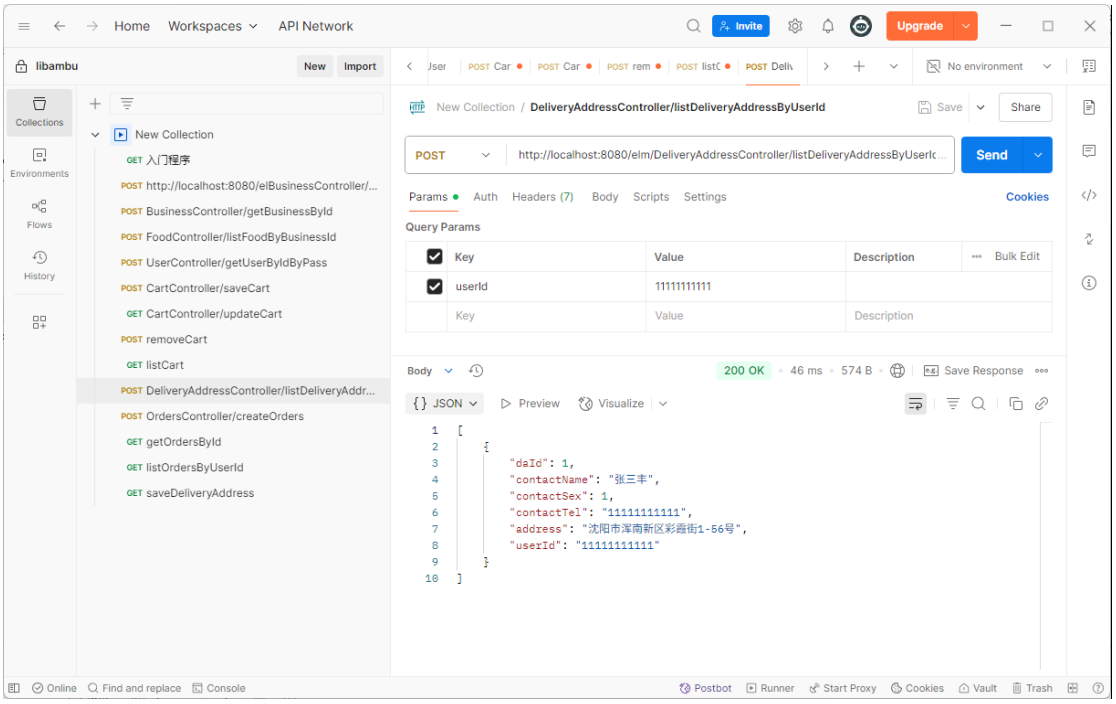


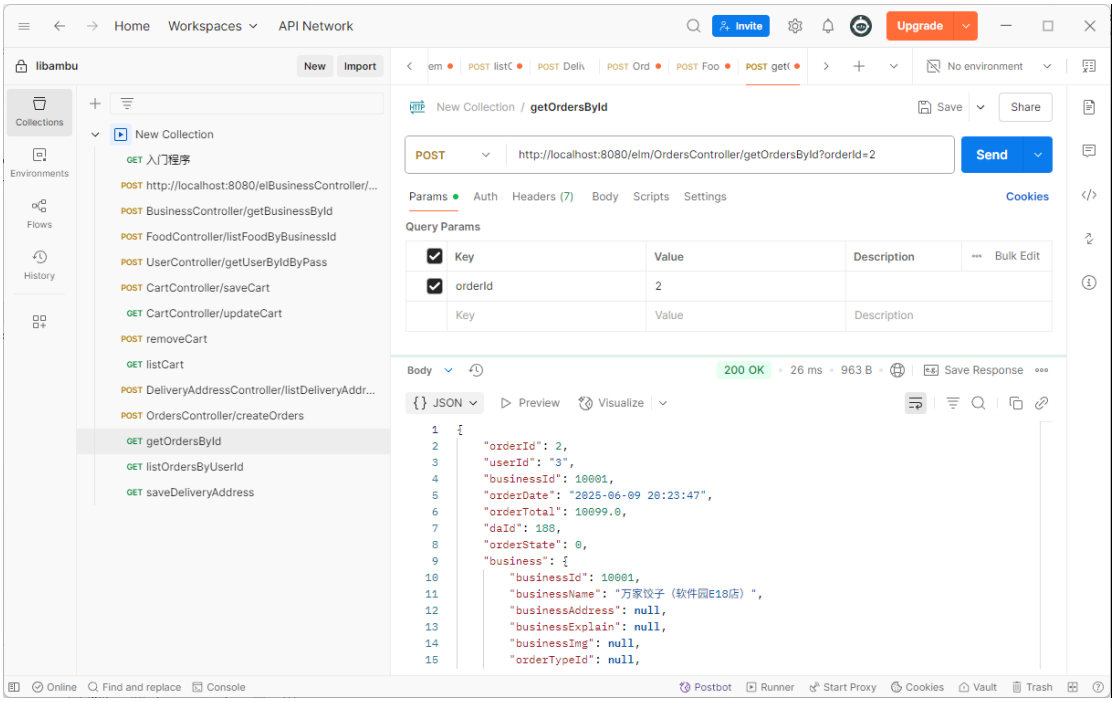
2. 前端接口测试结果

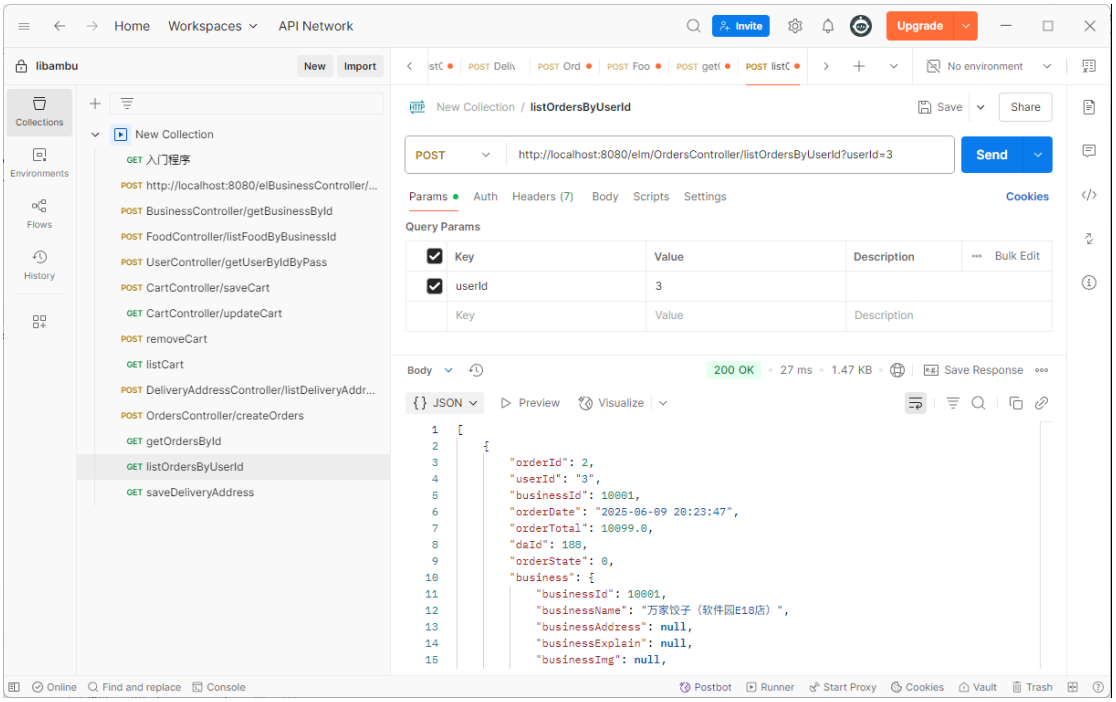



Home
Workspaces
API Network
Invite
Upgrade
libambu
New
Import
Collections
Environments
Flows
History
New Collection
GET 入门程序
POST http://localhost:8080/elBusinessController/...
POST BusinessController/getBusinessById
POST FoodController/listFoodByBusinessId
POST UserController/getUserByIdByPass
POST CartController/saveCart
GET CartController/updateCart
POST removeCart
GET listCart
POST DeliveryAddressController/listDeliveryAddr...
POST OrdersController/createOrders
GET getOrdersById
GET listOrdersByUserId
GET saveDeliveryAddress
No environment
New Collection / FoodController/listFoodByBusinessId
Save
Share
POST
http://localhost:8080/elm/FoodController/listFoodByBusinessId?businessId=10001
Send
Params
Auth
Headers (7)
Body
Scripts
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
businessId
10001
200 OK
119 ms
324.24 KB
Save Response
JSON
Preview
Visualize
[
{
"foodId": 1,
"foodName": "纯肉鲜肉（水饺）",
"foodExplain": "纯肉馅饺子",
"foodImg": "
iVBORw0KGgoAAAANSUhEUgAAAIIAAACCCAIAAAAFYYeqAAAAAXNSR0IArs4c6QAAAARnQU1BAACx
jwv8YQUAAAAJcEhZcwAADsMAAA7DAcdvqGQAAJztSURBVHhe7P9FVKTZkjYK894lXe7u7u7u7u
7uSGAQUUBRTu7q6JkySku7u7kqT+kVR3nz5nznzwzZ2buP99d69aKzKVRSfbLfnbYE7e7n/vz//
F6yAFzyDK2OQO3W3WoK3mYK3m4M3GEK3GkM3aMM3asI3qcI3KCI2KSPXyS
I2yK/esUe+s0e8g0e/AM25cZ9qBZtkCQOW3+c2z+89rz8Q8a8+Sxs8l1k9V7c5z3kf/
aXdbndzr8I/NUN/3nE5XY5fxen8/2HuFxOp8vpcH8TeIfnvb//faDf/04h9tt
+4vY3W74jkdcbrvL8+p0OVwum9vVi172f/jAf/
1X9cDgwSBtiBswSAEMRriSAAMAAAQAABkHGDij2zGInOQADMI9GDj+hsFMDwafZwMGtoC57RjMt/
2BQVs7Btb3S//1Z/s//cT/PAx//URYaNsfAhh4BP7q9Ah80Y4EwACr/w2J/xkY/
kEP3LJ6t7Qe9MAtbfinK+ESIR15p90muYuS5RI2uTXsP/XAScpxGxW29Mt/
1QPAwPou2f8fYQCFaBfn38QF6uBygR58k3+1DX/VCYCh7c9Fb4ehzeE2t7lb7X/CAMoFq/8NiX/



Home
Workspaces
API Network
Invite
Upgrade
libambu
New
Import
Collections
Environments
Flows
History
New Collection
GET 入门程序
POST http://localhost:8080/elBusinessController/...
POST BusinessController/getBusinessById
POST FoodController/listFoodByBusinessId
POST UserController/getUserByIdByPass
POST CartController/saveCart
GET CartController/updateCart
POST removeCart
GET listCart
POST DeliveryAddressController/listDeliveryAddr...
POST OrdersController/createOrders
GET getOrdersById
GET listOrdersByUserId
GET saveDeliveryAddress
No environment
New Collection / listCart
Save
Share
POST
http://localhost:8080/elm/CartController/listCart?userId=3&businessId=10001
Send
Params
Auth
Headers (7)
Body
Scripts
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
userId
3
businessId
10001
200 OK
102 ms
96.78 KB
Save Response
JSON
Preview
Visualize
[
{
"cartId": 6,
"foodId": 3,
"businessId": 10001,
"userId": "3",
"quantity": 1,
"food": {
"foodId": 3,
"foodName": "虾仁三鲜（蒸饺）",
"foodExplain": "三鲜馅饺子",
"foodImg": "
iVBORw0KGgoAAAANSUhEUgAAAIIAAACCCAIAAAAFYYeqAAAAAXNSR0IArs4c6QAAAARnQU1B
AACxjwv8YQUAAAAJcEhZcwAADsMAAA7DAcdvqGQAAHhwSURBVHheXb3ZkyzbdD6Z2ZldD6Xc2ZldVz
n/EOuAAuAYIkQIkCFIgIJCCSEKCQImSSIMkQECEKNKWZA0MR1hPfnT4P7D/
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Vault
Trash


Home
Workspaces
API Network
Invite
Upgrade
libambu
New
Import
Collections
Environments
Flows
History
New Collection
GET 入门程序
POST http://localhost:8080/elBusinessController/...
POST BusinessController/getBusinessById
POST FoodController/listFoodByBusinessId
POST UserController/getUserByIdByPass
POST CartController/saveCart
GET CartController/updateCart
POST removeCart
GET listCart
POST DeliveryAddressController/listDeliveryAddr...
POST OrdersController/createOrders
GET getOrdersById
GET listOrdersByUserId
GET saveDeliveryAddress
No environment
New Collection / DeliveryAddressController/listDeliveryAddressByUserId
Save
Share
POST
http://localhost:8080/elm/DeliveryAddressController/listDeliveryAddressByUserId...
Send
Params
Auth
Headers (7)
Body
Scripts
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
userId
11111111111
200 OK
46 ms
574 B
Save Response
JSON
Preview
Visualize
[
{
"daId": 1,
"contactName": "张三丰",
"contactSex": 1,
"contactTel": "11111111111",
"address": "沈阳市浑南新区彩霞街1-56号",
"userId": "11111111111"
}
]
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Vault
Trash


Home
Workspaces
API Network
Invite
Upgrade
libambu
New
Import
No environment
Collections
Environments
Flows
History
New Collection
GET 入门程序
POST http://localhost:8080/elBusinessController/...
POST BusinessController/getBusinessById
POST FoodController/listFoodByBusinessId
POST UserController/getUserByIdByPass
POST CartController/saveCart
GET CartController/updateCart
POST removeCart
GET listCart
POST DeliveryAddressController/listDeliveryAddr...
POST OrdersController/createOrders
GET getOrdersById
GET listOrdersByUserId
GET saveDeliveryAddress
New Collection / getOrdersById
Save
Share
POST
http://localhost:8080/elm/OrdersController/getOrdersById?orderId=2
Send
Params
Auth
Headers (7)
Body
Scripts
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
orderId
2
Body
200 OK
26 ms
963 B
Save Response
JSON
Preview
Visualize
{
"orderId": 2,
"userId": "3",
"businessId": 10001,
"orderDate": "2025-06-09 20:23:47",
"orderTotal": 10099.0,
"daId": 188,
"orderState": 0,
"business": {
"businessId": 10001,
"businessName": "万家饺子（软件园E18店）",
"businessAddress": null,
"businessExplain": null,
"businessImg": null,
"orderTypeId": null,
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Vault
Trash


Home
Workspaces
API Network
Invite
Upgrade
libambu
New
Import
No environment
Collections
Environments
Flows
History
New Collection
GET 入门程序
POST http://localhost:8080/elBusinessController/...
POST BusinessController/getBusinessById
POST FoodController/listFoodByBusinessId
POST UserController/getUserByIdByPass
POST CartController/saveCart
GET CartController/updateCart
POST removeCart
GET listCart
POST DeliveryAddressController/listDeliveryAddr...
POST OrdersController/createOrders
GET getOrdersById
GET listOrdersByUserId
GET saveDeliveryAddress
New Collection / listOrdersByUserId
Save
Share
POST
http://localhost:8080/elm/OrdersController/listOrdersByUserId?userId=3
Send
Params
Auth
Headers (7)
Body
Scripts
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
userId
3
Body
200 OK
27 ms
1.47 KB
Save Response
JSON
Preview
Visualize
[
{
"orderId": 2,
"userId": "3",
"businessId": 10001,
"orderDate": "2025-06-09 20:23:47",
"orderTotal": 10099.0,
"daId": 188,
"orderState": 0,
"business": {
"businessId": 10001,
"businessName": "万家饺子（软件园E18店）",
"businessAddress": null,
"businessExplain": null,
"businessImg": null,
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Vault
Trash

项目完善

创建独立局域网

首先当后端项目用校园网部署后，前端请求路径 http://172.18.138.84:8080/elm/ 这样是做不到跨主机连接的，这个 ip 其他主机是 ping 不通的

后端服务器电脑换成手机的移动热点，成功获取一个公网 ip 地址，关闭防火墙后可以 ping 通但是无法响应数据

```
以太网适配器 VMware Network Adapter VMnet1:

   连接特定的 DNS 后缀 . . . . . . . :
   本地链接 IPv6 地址. . . . . . . . : fe80::b42c:f5ab:9473:2a42%14
   IPv4 地址 . . . . . . . . . . . . : 192.168.152.1
   子网掩码 . . . . . . . . . . . . : 255.255.255.0
   默认网关. . . . . . . . . . . . . :

以太网适配器 VMware Network Adapter VMnet8:

   连接特定的 DNS 后缀 . . . . . . . :
   本地链接 IPv6 地址. . . . . . . . : fe80::b6c6:f1ab:c2dd:bb0b%5
   IPv4 地址 . . . . . . . . . . . . : 192.168.175.1
   子网掩码 . . . . . . . . . . . . : 255.255.255.0
   默认网关. . . . . . . . . . . . . :

无线局域网适配器 WLAN:

   连接特定的 DNS 后缀 . . . . . . . :
   IPv6 地址 . . . . . . . . . . . . : 2403:ac00:a201:14a:23be:4536:2abf:c36e
   临时 IPv6 地址. . . . . . . . . . : 2403:ac00:a201:14a:7d8c:4208:8463:ef95
   本地链接 IPv6 地址. . . . . . . . : fe80::3f0c:713:4eb2:e0a0%17
   IPv4 地址 . . . . . . . . . . . . : 172.18.138.84
   子网掩码 . . . . . . . . . . . . : 255.255.128.0
   默认网关. . . . . . . . . . . . . : fe80::5ab3:8fff:fed9:5002%17
                                       fe80::723a:a6ff:feaa:3002%17
                                       172.18.128.1
```

后来我们觉得应该是默认网关没有通过我们的信息路由，于是我们想到如果我们主机都在一个子网下，也就是一个局域网，并不走学校网关，应该可以实现连接，以下是我们的解决步骤

1、服务器端关闭防火墙，使用移动热点 WiFi 获取公网 ip，打开电脑的移动热点，并启动后端项目

2、前端服务器连接后端电脑的热点，这样多台电脑形成一个小的局域网，以后端服务器作为默认网关，前端修改 main.js 请求 ip 为后端公网 ip

```
13    getLocalStorage,
14    removeLocalStorage
15  } from './common.js'
16  Vue.config.productionTip = false
17  //设置axios的基础url部分
18  axios.defaults.baseURL = 'http://192.135.1.0:8080/elm/';
19  //将axios挂载到vue实例上，使用时就可以 this.$axios 这样使用了
20  Vue.prototype.$axios = axios;
21  Vue.prototype.$qs = qs;
22  Vue.prototype.$getCurDate = getCurDate;
23  Vue.prototype.$setSessionStorage = setSessionStorage;
```

经测试，三台前端服务器都可以访问后端服务器了，开始我们多人开发第一步

修改地址

修改地址到天津市津南区

完善搜索框

在原先的搜索框中，搜索框是不能用的，只是一个 div 和 p 标签，我们重做了搜索框

并可以向后端传输数据，但是后端还没来得及实现搜索功能

登录注册校验

确保手机号 11 位，在原项目中注册并没有对手机号位数进行检测。我们修改了这一 bug，效果如下图所示

```
// 手机号格式校验
if (!/^1\d{10}$/.test(this.user.userId)) {
    alert('请输入正确的11位手机号! ');
    return;
}
```

实现微信支付

在原项目中支付页面是写死支付宝支付的，我们实现了选择功能

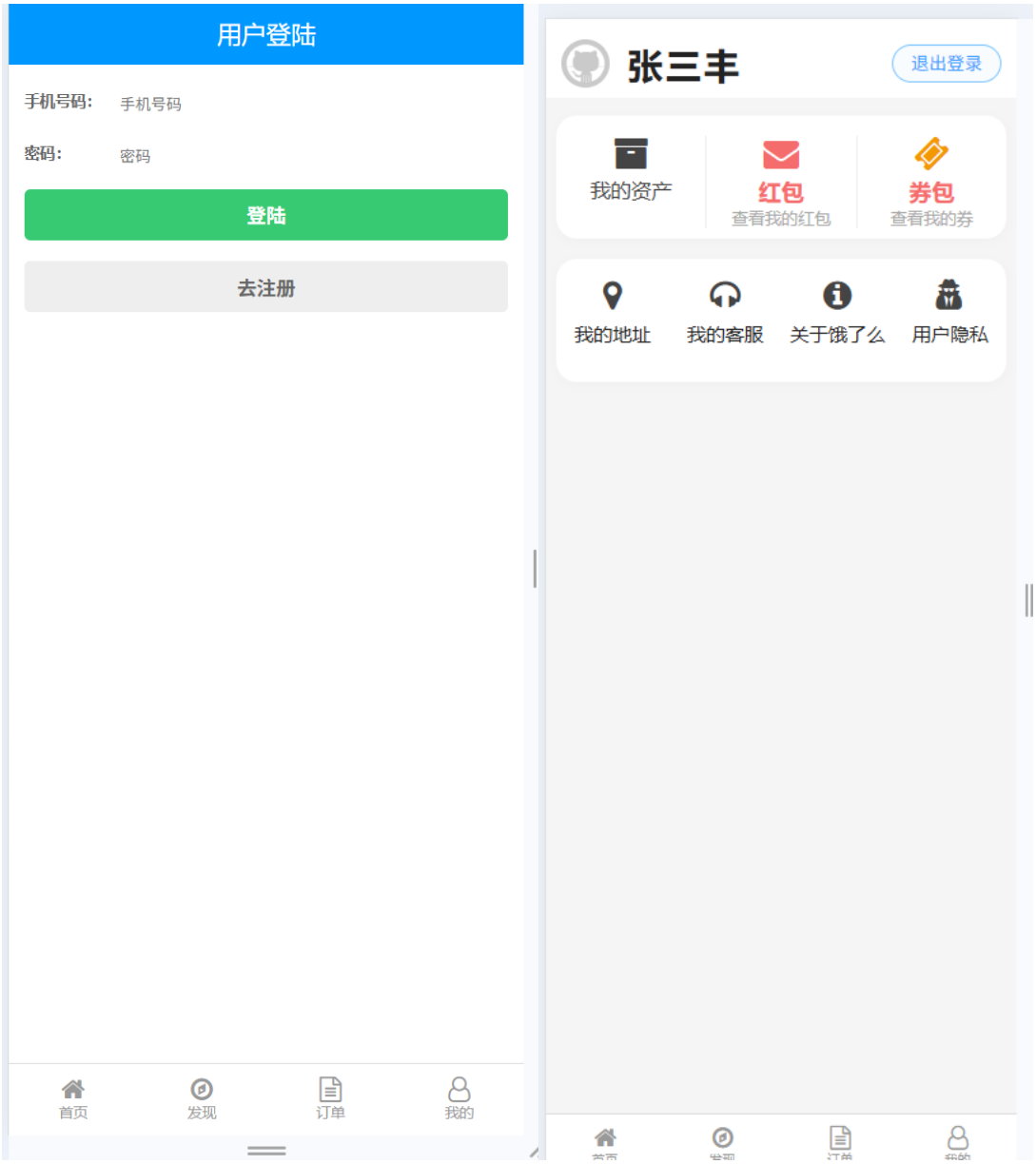
实现支付跳转

在原项目中，我们订单中去支付只是个 p 标签，和实际大不相符合，于是我们实现了点击去支付可以实现支付跳转

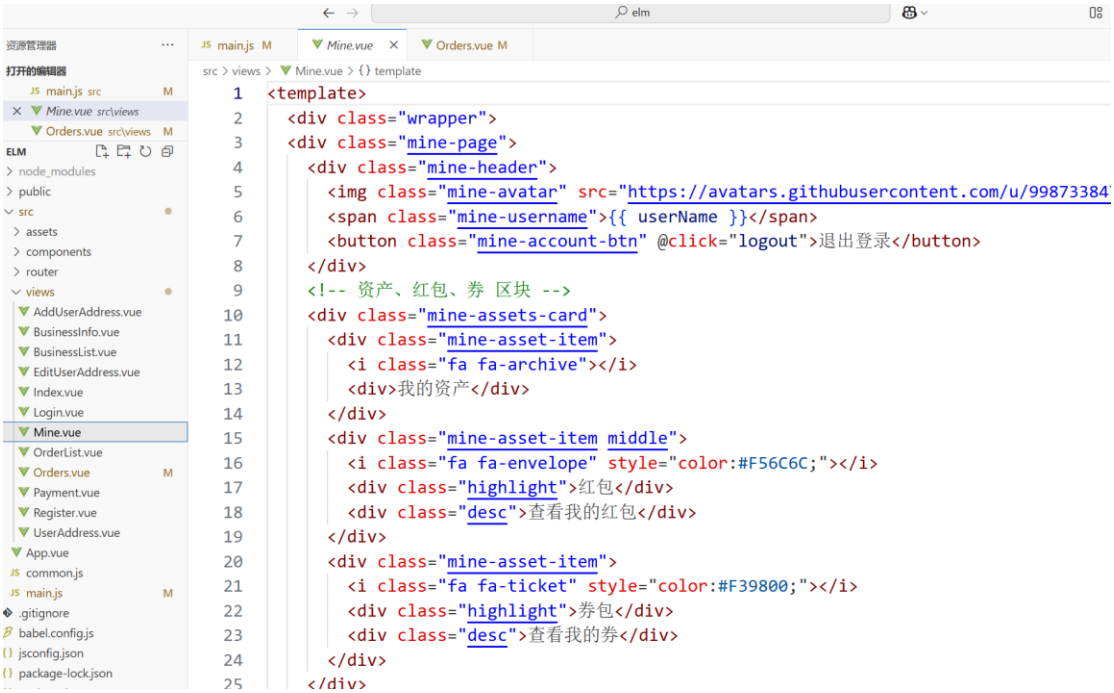
```
Windsurf: Refactor | Explain | Generate Function Comment | ×
goPayment(order) {
    this.$router.push({ path: '/payment', query: { orderId: order.orderId } });
}
```

实现我的页面

首先我们注意到原项目只做了首页和订单功能，并将用户登录注册页面放到了订单页面里，这是很不合常理的，于是我们打开饿了么官方，仿照官方做了独立的新页面，我的页面，在这个页面实现了登录。登录后页面如下图所示



部分代码如下



实现账号切换

原项目中用户一但登录之后就无法退出，也无法切换，这也是明显有问题的，于是我们在我的页面实现了用户退出功能，之后就可以重新登录了

退出登录并重新切换新账号

```
<div class="wrapper">
<div class="mine-page">
  <div class="mine-header">
    <img class="mine-avatar" src="https://avatars.githubusercontent.com/u/998733847?v=4" alt="头像">
    <span class="mine-username">{{ userName }}</span>
    <button class="mine-account-btn" @click="logout">退出登录</button>
  </div>
```

实现地址管理

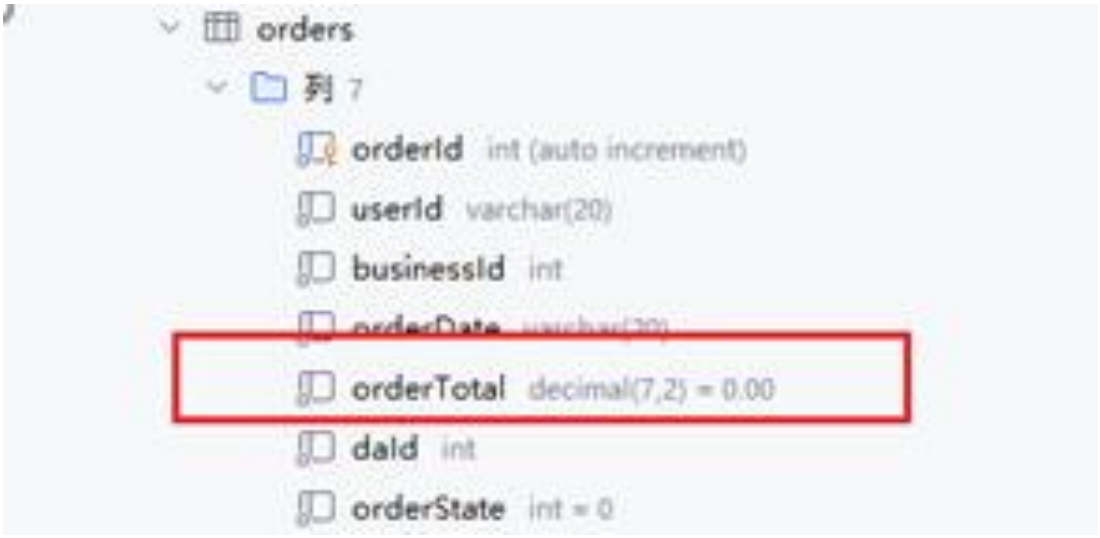
在我的页面点击我的地址就可以实现地址管理，并且可以新增地址


```
<!-- 其他菜单 -->
<div class="mine-menu-card" @click="goAddress">
  <div class="mine-menu-item">
    <i class="fa fa-map-marker"></i>
    <span>我的地址</span>
  </div>
```

精度处理

为了防止数据 double 型 foodprecie 和 int 整型 quantity 相乘计算 Totalprice 有精度丢失的可能，我们对项目做了一下修改

```
        computed: {
        totalPrice() {
            let totalPrice = 0;
            for (let cartItem of this.cartArr) {
                // 将金额转换为整数进行计算
                const itemPrice = cartItem.food.foodPrice * 100;
                const itemTotal = itemPrice * cartItem.quantity;
                totalPrice += itemTotal;
            }
            // 将配送费也转换为整数进行计算
            const deliveryPrice = this.business.deliveryPrice * 100;
            totalPrice += deliveryPrice;
            // 最后转换回两位小数的格式
            return totalPrice / 100;
        }
    },
```



设置后端数据格式为两位小数，现实情况也是这样，金额最精确也是到分，这样在前端中计算可以解决精度丢失问题