河南财政金融学院

2023 届本科毕业设计

基于 JavaScript GL API 的物流追踪系统

姓　　名:	吴玉配
学　　号:	20192502010341
专　　业:	计算机科学与技术
班　　级:	19 级计科 U(3)
指导教师:	赵景海
教学单位:	计算机与人工智能学院

二〇二三年 五 月 二 十 日

河南财政金融学院

毕业论文（设计）版权使用授权书

本人完全了解河南财政金融学院关于收集、保存和使用学位毕业论文（设计）的规定，同意如下各项内容：按照学校要求提交毕业论文（设计）的印刷本和电子版本；学校有权保存毕业论文（设计）的印刷本和电子版，并采用影印、缩印、扫描、数字化或其它手段保存毕业论文（设计）；学校有权提供目录检索以及提供本毕业论文（设计）全文或者部分的阅览服务；学校有权按有关规定向国家有关部门或者机构送交毕业论文（设计）的复印件和电子版；在不以赢利为目的的前提下，学校可以适当复制毕业论文（设计）的部分或全部内容用于学术活动。

指导教师签名：

毕业论文（设计）作者签名：

年　　月　　日

河南财政金融学院

毕业论文（设计）原创性声明

本人郑重声明：所呈交的毕业论文（设计），是本人在指导教师指导下，进行研究工作所取得的成果。除文中已经注明引用的内容外，本毕业论文（设计）的研究成果不包含任何他人创作的、已公开发表或者没有公开发表的作品的内容。对本毕业论文（设计）所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。本学位毕业论文（设计）原创性声明的法律责任由本人承担。

指导教师签名：

毕业论文（设计）作者签名：

年　　月　　日

目　录

摘　要

本文旨在设计和实现一个基于地图 JavaScript GL API 的物流追踪系统，以解决物流追踪的准确性和效率问题。通过研究物流追踪的基本原理和需求，本文分析了当前物流追踪系统存在的问题和挑战，并提出了基于地图 JavaScript GL API 技术实现的解决方案。本文首先对 JavaScript GL API，可视化等技术进行了深入研究和分析，探讨了其特点、优势和应用。同时，该系统还具有良好的用户体验和操作性，优化物流管理，提升其效率和经济效益。总的来说，本文的研究对于优化物流管理、提高物流追踪的准确性和效率具有重要的实际应用价值。此设计可作为其他物流领域的研究和实践的参考和借鉴，具有一定的指导意义和参考价值。然而，基于地图 JavaScript GL API 技术实现物流追踪系统仍存在一些局限性和挑战，未来研究可以进一步探索如何克服这些挑战，优化系统的设计和实现。

关键词：物流；追踪；JavaScript GL API；可视化

ABSTRACT

This article aims to design and implement a logistics tracking system based on the Map JavaScript GL API to solve the accuracy and efficiency problems in logistics tracking. By studying the basic principles and requirements of logistics tracking, this article analyzes the problems and challenges of current logistics tracking systems, and proposes a solution based on the Map JavaScript GL API technology. This article first conducts in-depth research and analysis on the Map JavaScript GL API technology, discussing its characteristics, advantages, and applications. Overall, this research has important practical application value for optimizing logistics management and improving the accuracy and efficiency of logistics tracking. This design can serve as a reference and reference for other research and practices in the logistics field and has certain guiding significance and reference value. However, the implementation of logistics tracking systems based on Map JavaScript GL API technology still has some limitations and challenges,and future research can further explore how to overcome these challenges and optimize the system design and implementation.

Keywords: logistics,tracking,JavaScript GL API, visualization

1 绪论

1.1 研究背景及意义

1.1.1 研究背景

物流是中国经济发展迅速的代表行业之一，在如今时代，它无处不在，我们可以随时随地看到它的身影，为什么我们能在网上买的东西能在很短的时间里送到我们手中，都是因为有非常多的物流企业存在，然而随着社会经济的发展，人们的购买能力的提升，物流企业信息管理面临着巨大的压力，如果还是照以前采用纸笔管理信息的方式，企业信息管理通常比较繁杂，企业工作的效率实在是太低，此时采用计算机来管理企业信息成为了一种趋势，计算机相比人为操作有更高的效率，更安全，也给企业工作人员降低了工作压力，工作管理人员只需要通过操作电脑，就可以管理企业的信息，但是我们如何通过计算机来管理企业信息呢，这时开发一个物流管系统成为了众多中小型物流企业的第一任务，它对于很多企业是发展中必不可少的一部。

1.1.2 研究意义

1. 提高物流效率：物流追踪系统可以实时监控货物的位置和状态，帮助物流企业更好地管理和调度运输资源，提高物流运输的效率和准确性。

2. 降低物流成本：通过物流追踪系统，物流企业可以实时掌握货物的运输情况，及时解决潜在问题，减少货物的丢失和损坏，降低物流成本。

3. 增强供应链可视性：物流追踪系统可以将整个供应链的信息可视化展示，让供应链各环节的参与者能够更好地协调合作，优化供应链的运作效率。

4. 推动物流行业发展：研究物流追踪系统的应用和技术，可以促进物流行业的创新和发展，推动行业的数字化转型，提升物流企业的竞争力。

1.2 研究内容和技术路线

1.2.1 研究内容

研究了 WEB GIS 技术和车辆路径规划相关技术，有效的优化了物流资源配置，提高了物流配送信息化管理水平、物流配送效率和服务质量。还研究了在当前物流数不胜数的环境下，应该如何高效管理物流，如何实现让物流可视化，可追踪，可拦截等一系列相关物流管理问题。

1.2.2 技术路线

通过参考相关参考文献，本系统选择 NodeJs 为系统的后台开发语言与 Mysql 和 Redis 数据库进行交互，保存数据同时给前端提供接口，前端使用 Vue.js 开发用户界面，使用高德地图 可视化 API 完成物流路线的绘制和地图的生成。另外地图可视化的一些特效用也会到复杂的可视化算计及最短路线算法。

1.3 国内外文献综述

查询到已发表的学术文献，包括学术论文、会议论文和技术报告等相关 JavaScript GL API 和 Web GIS[1]在物流追踪方面的最新应用。发现这些系统都有一些局限性和复杂性。

2019 年周文业《基于移动 GIS 的共享物流系统的设计与实现》，这篇文献介绍了一个基于移动 GIS 的共享物流配送管理系统的设计与实现，该系统结合共享物流的特点和我国货运物流现状，优化资源配置，提高物流配送效率和安全性。文献作者对货运物流现状和共享物流的创新模式进行了研究，提出了一种新的共享货运物流模式，设计和实现了共享物流管理系统。同时，作者分析了移动 GIS 的应用现状、相关理论和技术，搭建了一个服务于货运共享物流模式的移动 GIS 平台。此外，作者还对物流配送车辆路径问题进行了研究和分析，并结合共享货运物流的配送模式，设计实现了适合系统配送模式的车辆路径优化算法，通过优化算法有效的降低了物流配送成本，提高了货运物流配送的质量和效率[2]。文献还进行了系统的测试，测试结果表明该系统功能和性能满足既定要求。

2014 年李晶在现代农产品物流环节中，讨论了二维码技术在农产品物流追溯系统中的应用和优缺点。作者指出，建立农产品物流追溯系统是保障农产品质量安全的必要手段，而二维码技术能够为其提供技术支持[3]。在详细分析了二维码技术的相关特点后，作者提出了农产品物流追溯系统中使用二维码技术可能会带来的问题[4]，并对其在物流追踪领域的发展前景进行了预测。

2019 年吴小力在物流跟踪系统中讲述了基于智慧供应链的电力物流跟踪系统。改论文中并没有像其他基于物联网的技术一样使用传统的 GPS 技术[5]和北斗卫星技术[6]其中，电力物流跟踪系统是电力智慧供应链的核心环节之一。为此，文章介绍了一种基于 RFID 的智慧电力物流跟踪系统，并对其进行了硬件和软件的详细设计。RFID 设备是该智慧电力物流跟踪系统的核心部件，由单片机、RFID 模块、隔离模块

和 GPRS 模块组成。文章还介绍了 RFID 设备的硬件设计和软件流程图[7]，说明了 RFID 设备的工作原理和操作流程。最后，文章提到该系统已经成功应用于地市级电网，运行结果证明了其有效性 8]。

Laura Meade[9]、Ramamurthy、Bhargavi、 ShashiKumar[10]在系统中使用 MVVM 的方式开发的物流系统非常的与时俱进，其中的物流安全和 GPS 技术与孟志军,赵春江,王秀的论文中提到的 GPS 技术以及柴凤伟所提到的物流安全保障问题[11]有非常相似的思想和技术。

最后通过参考徐晶文[12]、苏峰[13]、陆祥瑞[14]、夏绪宏[15]这四位作者的论文，获得了本系统更加广阔的设计思路和技术架构思想。

2 系统分析

2.1 需求分析

2.1.1 功能需求

1. 用户注册和登录功能：用户需注册并登录后才能使用本系统的物流追踪服务。

2. 物流信息录入：管理员可以录入物流信息，包括物流单号、物品名称、发货人信息、收货人信息等。

3. 物流信息查询：用户可以通过输入物流单号等信息查询物流信息，包括物品当前位置和运输状态等。

4. 物流信息更新：管理员可以更新物流信息，包括物品当前位置和状态等。

5. 物流信息展示功能：系统可以将查询到的快递物流信息以地图、列表、图表等多种方式进行展示。

2.1.2 性能需求

1. 系统的响应时间：查询和展示快递物流信息需要在 1 秒内完成。

2. 并发访问量：系统需要支持至少 100 个并发请求。

3. 系统稳定性：系统需要保持 24 小时的稳定性，确保用户可以随时使用系统。

2.1.3 环境需求

1. 部署环境：部署在云服务器上，保证系统的稳定性和可靠性。

2. 操作系统：Windows、Linux。

3.. 数据库：MySQL 数据库管理系统。

2.1.4 用户需求

1. 易用性：系统需要简单易用，不需要太多的操作步骤。

2. 实时性：系统需要提供实时更新的物流信息，保证用户可以及时了解快递的运输情况。

3. 可靠性：系统需要提供准确、及时的物流信息。

4. 安全性：系统需要严格的安全措施[8]，保证用户数据的安全性。

2.1.5 系统时序图及数据流图

1. 时序图

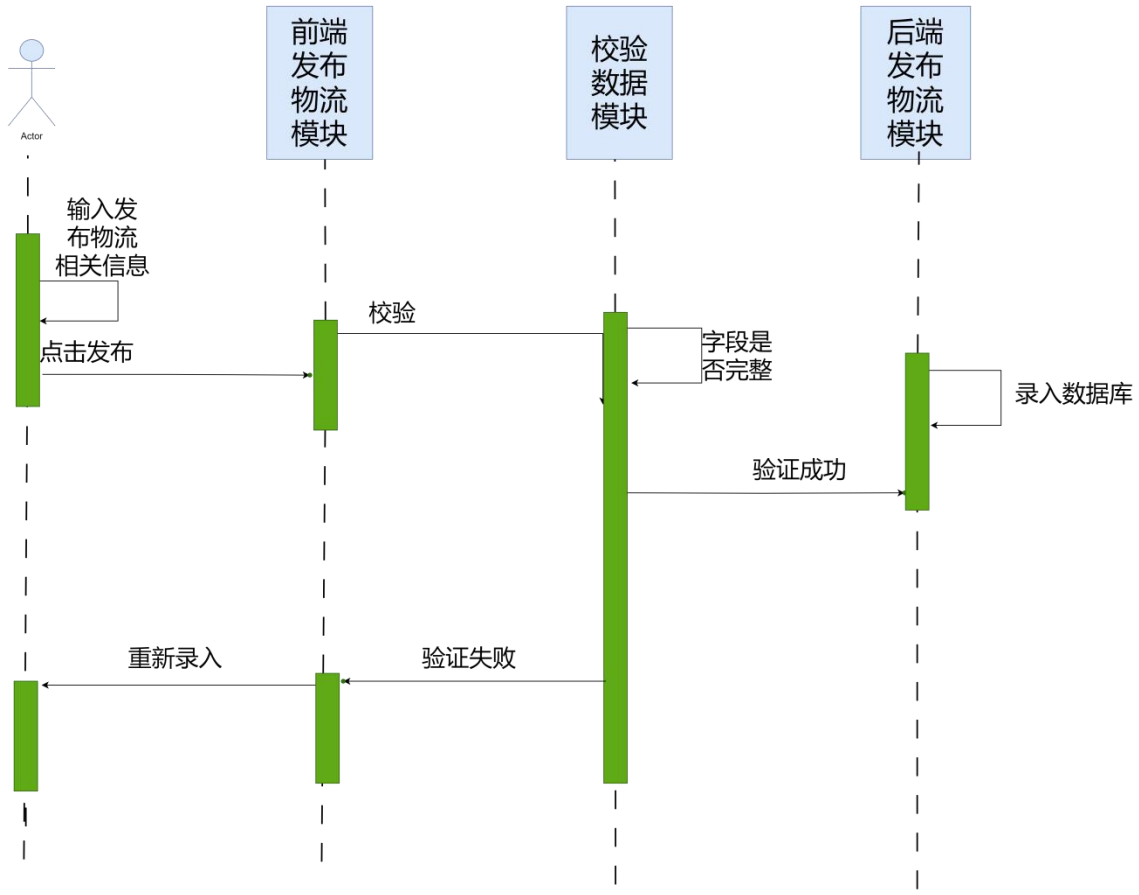

图 1　发布物流时序图

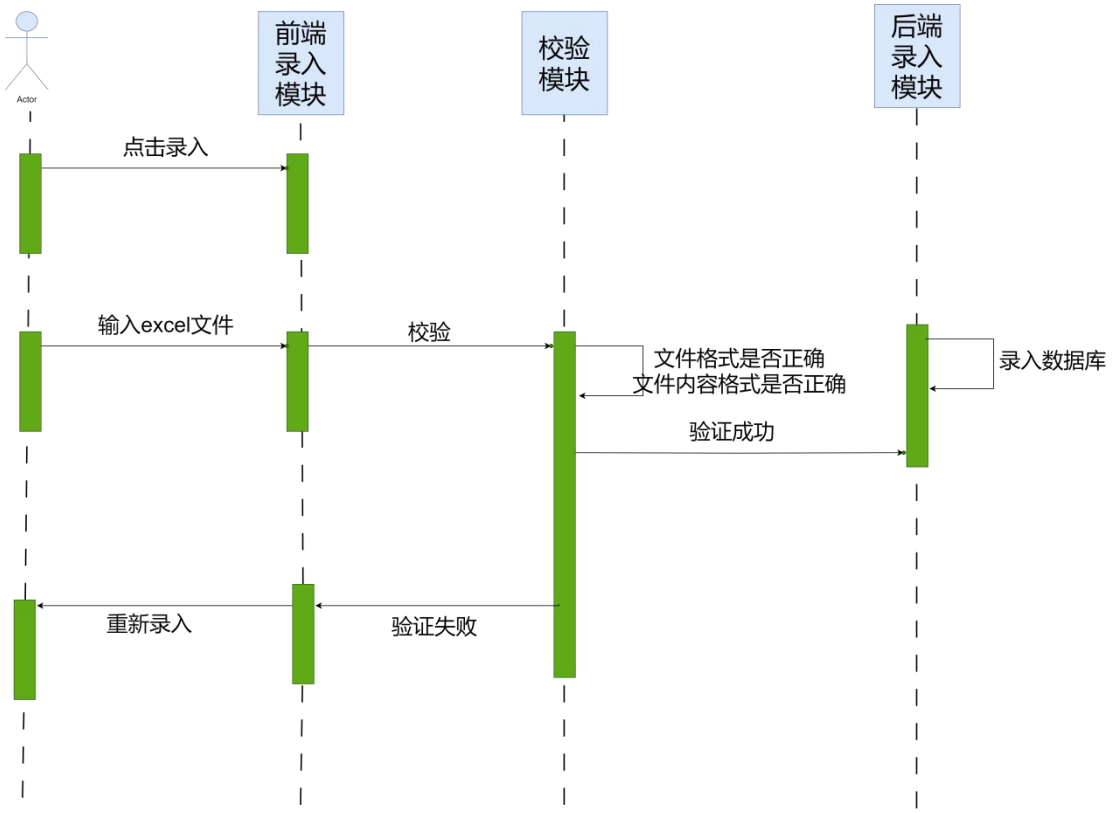

图 2　录入货物时序图

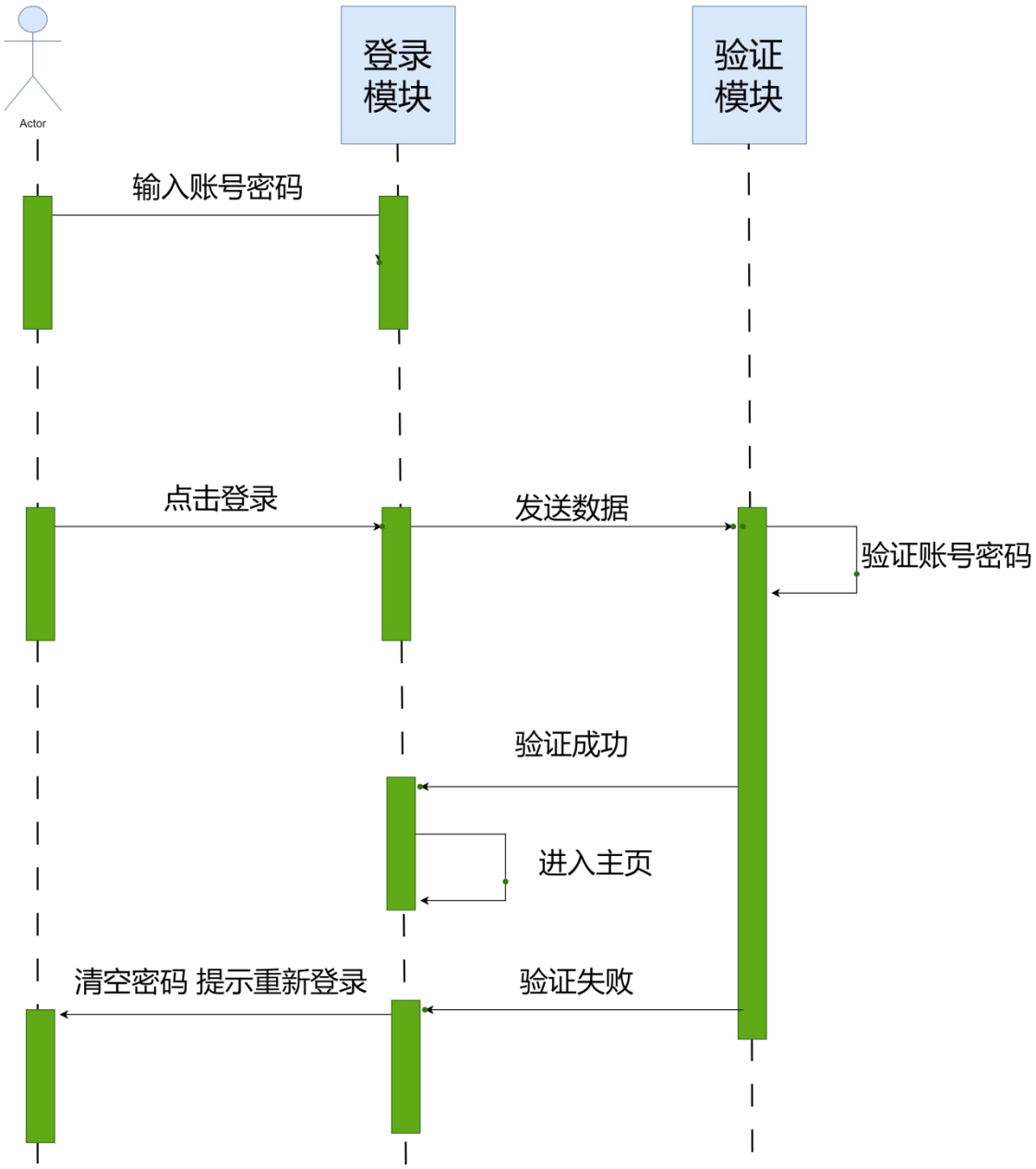

图 3　登录时序图

2. 数据流图

图 4　顶层数据流图

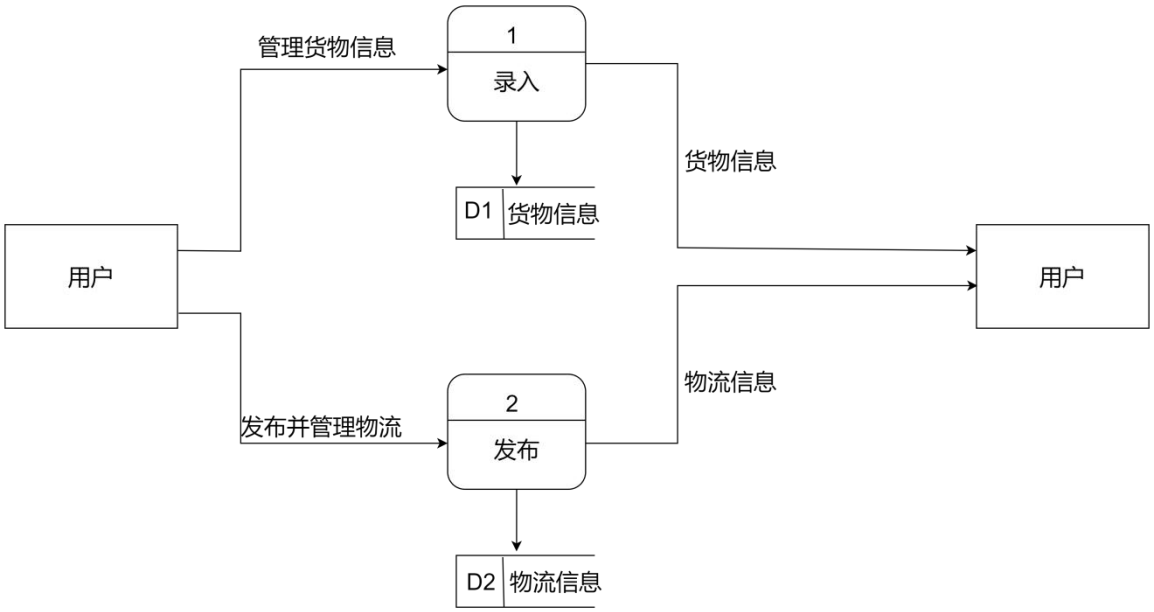

图 5　0 层数据流图

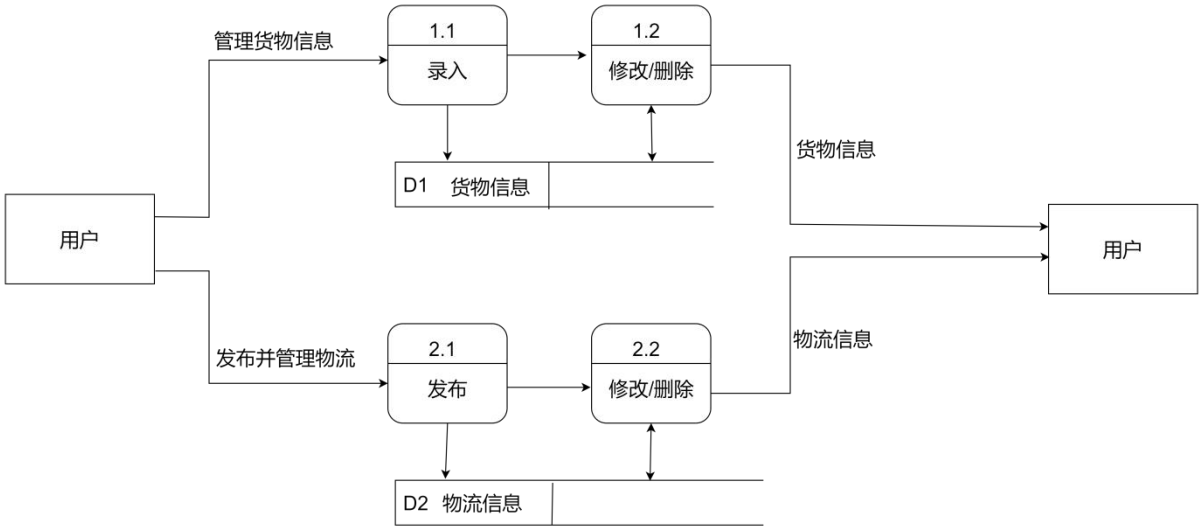

图 6　1 层数据流图

2.2 可行性分析

2.2.1 法律可行性

本系统需要遵守相关的法律法规，包括数据保护法律、互联网安全法律等。同时，还需要获得高德地图 API 的授权使用。因此，在法律可行性上，本系统需要严格遵守相关法律法规，确保系统的合法性和安全性。

2.2.2 技术可行性

本系统采用了流行的前后端分离技术，使用 Node.js 作为后端开发语言，采用 Koa 作为后端框架，使用 Vue3 作为前端框架。同时，为实现实时物流追踪，采用了高德

地图 API。这些技术都是成熟的技术，有丰富的文档和支持，因此在技术可行性上不存在问题。

2.2.3 经济可行性

本系统的开发需要一定的经费投入，包括系统所需的硬件设备和软件授权等。同时，该系统的实施将会带来一定的经济效益，能够提高物流运营的效率和准确性，通过优化物流流程，减少物流成本，提高物流效率和服务质量，增强物流企业在市场上的竞争力。因此，在经济可行性上，本系统具有一定的优势。

3 总体设计

3.1 系统结构设计

该物流追踪系统采用前端和后端相互分离的架构技术，前端主要使用 Vue3 技术，JavaScript 技术栈进行开发，后端采用 Node.js 和 Koa 框架，数据库使用 MySQL，缓存使用 Redis，地图 API 采用第三方高德地图 GL JS API。具体的系统结构设计如图 8 所示：

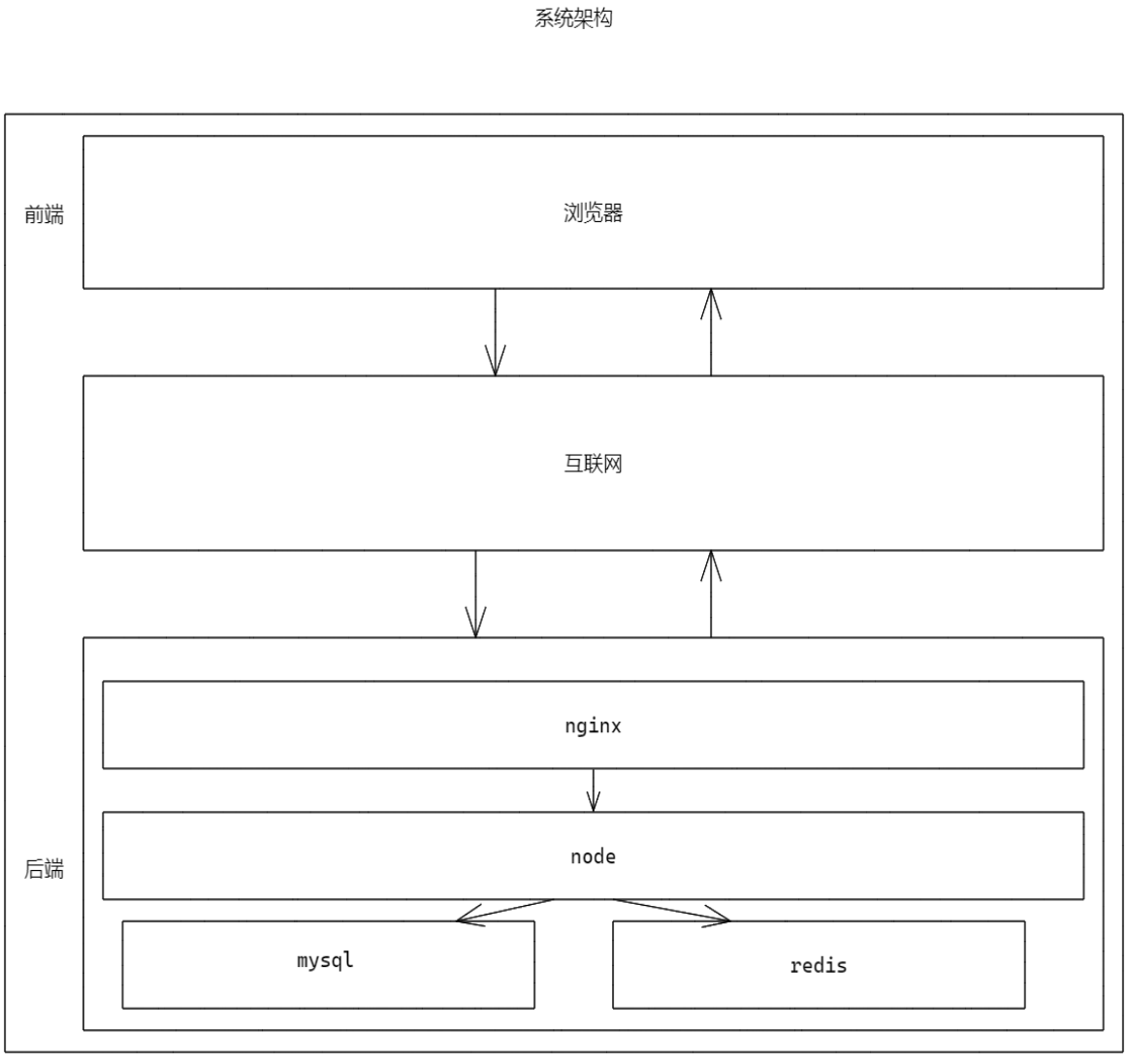

图 7　系统架构图

3.2 系统模块设计

该物流追踪系统的模块分为六个模块如图 8 所示：

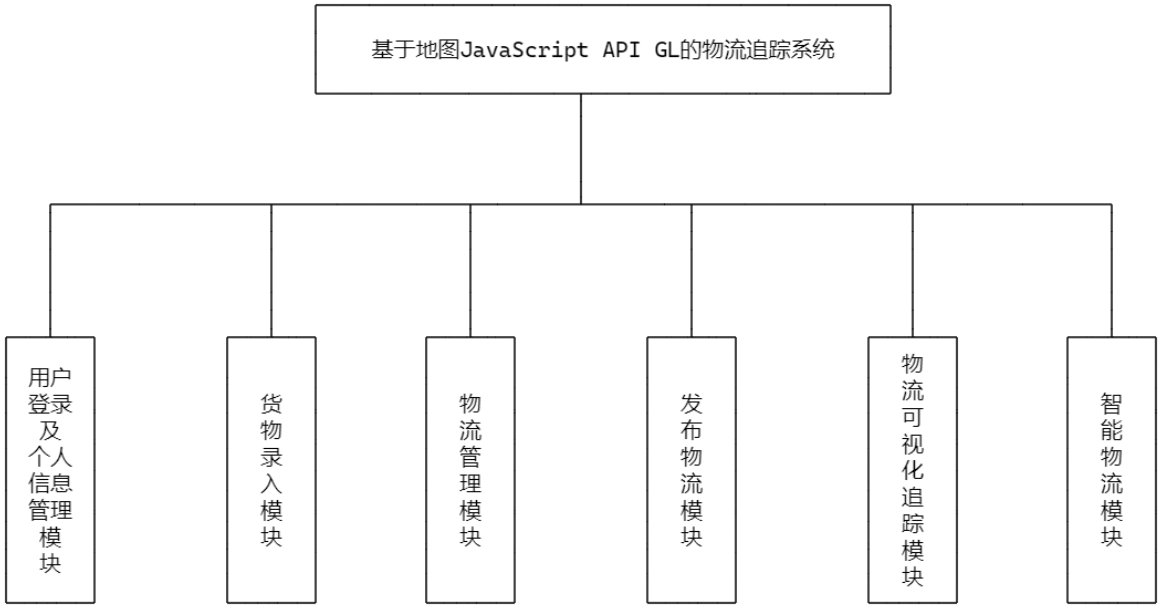

图 8　模块设计图

1. 用户登录及个人信息管理模块：实现用户登录、注册、鉴权等功能，对用户进行身份认证。

2. 物流管理模块：实现查询物流、添加物流、修改物流和删除物流等功能，对物流进行数据操作和处理还可以查看物流的运输路线，运输时间等详细信息。

3. 物流可视化追踪模块：实现物流状态查询和展示，显示订单的实时物流轨迹，支持地图展示和路径规划。实现物流的多种可视化追踪。

4. 发布物流模块：在系统内选择物品进行发布物流。

5. 智能物流模块：一个智能物流机器人，可以向它询问某个物流的具体状态和信息。

6. 货物录入模块：录入货物信息，可以通过导入 excel 文件的方式导入货物信息。

3.4 算法设计

1. 路径规划算法：高德地图 GL API 支持路径规划，可以根据起点、终点、途经点和交通方式等信息生成最优路径。算法设计的关键是根据不同的场景和需求选择合适的路径规划算法，例如 Dijkstra 算法、最短路算法等。

2. 可视化算法：高德地图 GL API 提供了丰富的地图可视化功能，可以展示地图数据、路线、标记点等。在设计可视化算法时，需要考虑数据的类型和数量、展示方式和效果等因素。例如，在展示路线时，可以使用平滑曲线算法来优化路线的显示效

果。

3.3 数据库设计

1. 用户表

表 1 用户表

字段名	类型	长度	描述	能否为空
id	int	10	用户 id	否
account	varchar	11	用户账号	否
uname	varchar	20	用户名	否
password	varchar	20	用户密码	否
img	varchar	30	用户图像	否
address	varchar	50	用户地址	否
address_geo	varchar	20	用户地理坐标	否
create_time	timestamp		创建时间	否

2. 物流表

表 2 物流表

字段名	类型	长度	描述	能否为空
id	int	10	物流 id	否
uid	varchar	10	发货人 id	否
thing_id	int	20	用户名	否
current_time	timestamp		到达最新位置的时间	否
current_position	varchar	30	到达最新位置的地点	否
start_position	varchar	50	发货地点	否
end_position	varchar	20	收获地点	否
current_position_geo	varchar	20	到达最新位置的地理坐标	否
start_position_geo	varchar	20	发货位置的地理坐标	否
end_position_geo	varchar	20	收获位置的地理坐标	否
toName	varchar	20	收获人的名字	否
toPhone	varchar	11	收获人的手机号	否

			码	
time	timestamp		物流创建时的时间	否
qr_code	varchar	20	物流二维码	否
status	int	10	物流状态	否

3. 物流位置记录表

表 3 位置记录表

字段名	类型	长度	描述	能否为空
id	int	10	位置表 id	否
pid	varchar	11	当前物流关联物流 id	否
current_time	varchar	20	到达当前位置时间	否
current_position	varchar	20	到达当前位置地点名称	否
current_position_geo	varchar	30	到达当前位置地理坐标	否

4. 物品表

表 4 物品表

字段名	类型	长度	描述	能否为空
id	int	10	物品 id	否
uid	varchar	11	关联的用户的 id	否
name	varchar	20	物品名称	否
price	varchar	20	物品价格	否
total	varchar	30	总量	否
count	Int	20	剩余数量	否

5. E-R 图

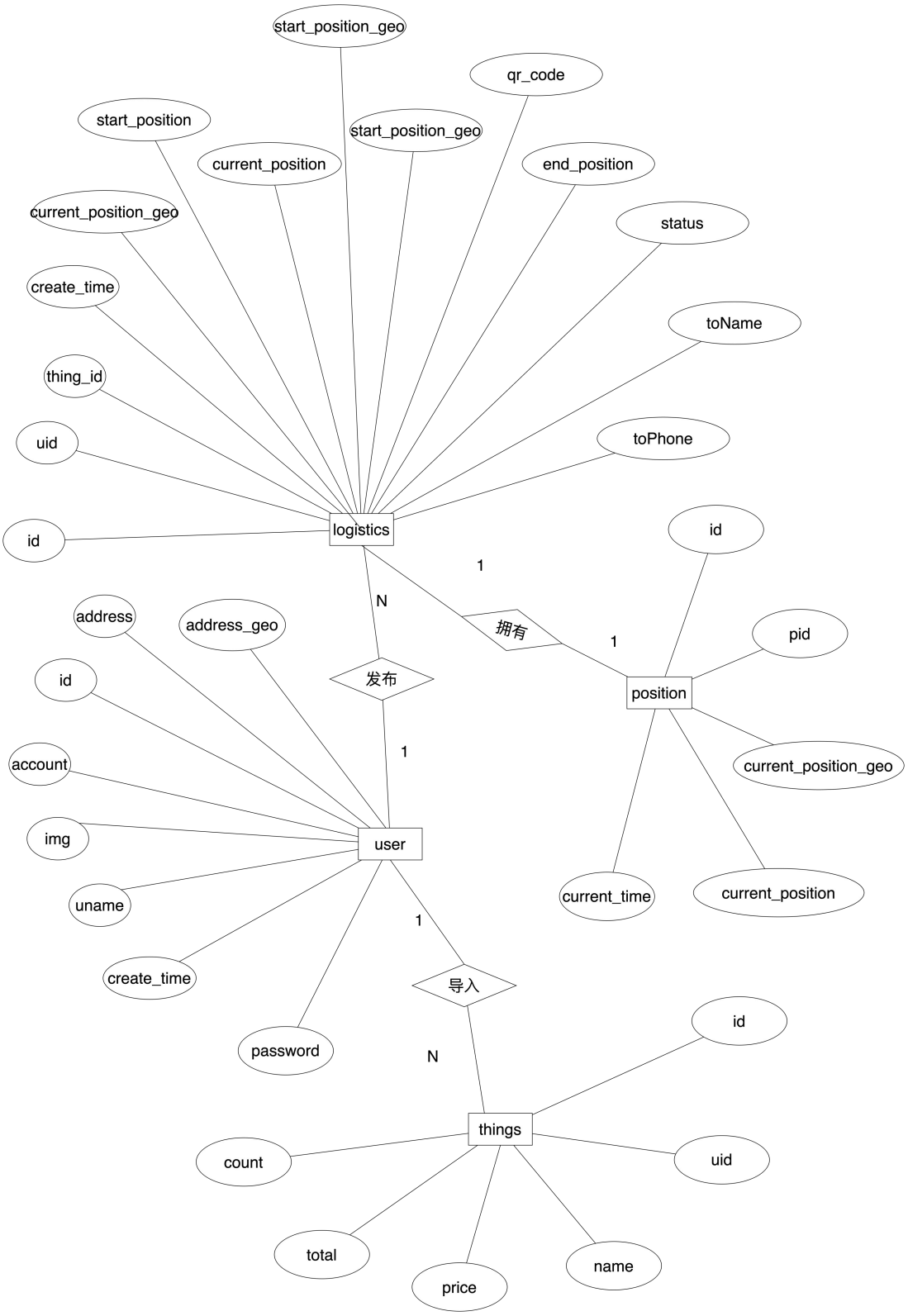
start_position_geo
qr_code
start_position
start_position_geo
current_position
end_position
current_position_geo
status
create_time
toName
thing_id
uid
toPhone
id
logistics
id
1
N
address
address_geo
拥有
1
pid
id
发布
position
1
account
current_position_geo
img
user
uname
current_time
current_position
1
create_time
导入
password
N
id
things
count
uid
total
price
name

图 9 E-R 图

4 详细设计

4.1 用户登录模块及修改个人信息模块详细设计

1. 用户界面：设计一个用户登录界面，包括用户名和密码输入框以及登录按钮。

2. 用户认证：接收用户输入的用户名和密码，并进行身份验证。验证可以使用哈希算法对密码进行加密，并与存储在数据库中的用户信息进行比对。

3. 错误处理：在登录模块中实现适当的错误处理机制，例如显示错误消息或重置输入字段等内容。

5. 会话管理：一旦用户通过身份验证，创建一个会话以跟踪用户的登录状态，并将其存储在服务器端或使用令牌进行身份验证。

4.2 货物录入模块详细设计

1. 货物导入模块：通过解析 excel 文件，将货物导入系统。

2. 货物信息存储模块：将货物的信息保存到数据库中，方便后续的查询和管理。

3. 安全性考虑：

(1) 对于涉及价格、重要描述或其他敏感信息的字段，可以考虑加密存储或访问控制，以确保数据的机密性和保密性。

(2) 防范潜在的安全漏洞，如 SQL 注入攻击和跨站脚本攻击(XSS)，通过使用参数化查询或适当的输入验证来防止这些攻击。

4.3 物流管理模块详细设计

1. 货物信息输入模块：允许用户输入货物的基本信息，包括货物名称、数量、起始地、目的地、收货人、联系电话等。

2. 货物状态设置模块：用户可以根据货物的状态，设置货物的状态，如待发货、已发货、运输中、已送达等。

3. 货物信息存储模块：将货物的信息保存到数据库中，方便后续的查询和管理。

4.4 物流可视化追踪模块详细设计

1. 地图展示模块：物流管理系统可以通过地图展示，直观地显示货物的运输路线和状态。

2. 实时更新模块：物流管理系统可以实时更新货物的位置和状态，以便用户随

时了解货物的最新情况。

3. 数据可视化模块：通过数据可视化技术，物流管理系统可以将货物的信息、状态等数据以图形化的方式呈现出来，方便用户快速了解货物的情况。

4. 状态标识和动画效果：使用不同的标识或符号来表示货物的当前状态，如起始地、目的地、运输中等。可以使用不同的图标、颜色或动画效果来区分不同状态。在货物移动过程中，可以使用动画效果来模拟货物在地图上的移动过程，增强用户体验。

5. 详细信息展示：提供一个信息窗口或侧边栏，显示有关特定货物的详细信息，如货物名称、目的地、预计到达时间、运输历史记录等。

6. 物流数据获取：获取物流运输过程中的相关数据，如货物的起始地、目的地、运输中的路线、运输阶段等。这些数据可以通过物流供应商提供的 API、传感器或其他数据源来获取。设计一个合适的数据接口或逻辑，用于获取和处理物流数据，并将其传递给可视化组件。

4.5 发布物流模块详细设计

1. 发布货物信息模块：用户可以将货物的信息发布到物流管理系统中，以便更好地管理和跟踪。

2. 货物信息录入：设计一个界面，允许用户输入货物的相关信息，如货物名称、数量、起始地、目的地等。对用户输入的数据进行验证，确保数据的有效性和完整性，例如验证数量是否为正数。

3. 生成物流单据模块：物流管理系统可以根据用户的要求，自动生成物流单据，二维码等方便用户进行打印和管理。

4. 数据存储和管理：将用户输入的货物信息、物流计划和状态数据存储到数据库中。设计相应的数据模型或表结构，确保数据的一致性和完整性。实现数据的增删改查功能，以支持对发布的物流信息进行管理和查询。

4.6 智能物流模块详细设计

1. 实现一个智能机器人，向机器人发送物流编号，机器人给你返回对应的物流信息。

5 系统实现

5.1 用户登录及修改个人信息

该功能实现了用户账号的登录和个人信息的修改，包括用户名、密码、联系方式等。该模块的部分代码如图 10 所示。流程图如图 11 所示。

```
class LoginController {
  async login(ctx: any, next: any) {
    const { account, password, type } = ctx.request.body;
    if (type == 1) {
      ctx.body = await loginService.login(account, password);
    } else {
      ctx.body = await loginService.adminLogin(account, password);
    }
  }

  async register(ctx: any, next: any) {
    const { username, password } = ctx.request.body;
    ctx.body = await loginService.register(username, password);
  }

  async change(ctx: any) {
    const data = ctx.request.body;
    ctx.body = await loginService.change(ctx.userInfo.id, data);
  }
}

export default new LoginController();
async login(account: string, password: string) {
      const res: any = await UserModel.findOne({
        where: {
          account,
          password,
        },
      });

      if (res) {
        const token = genToken({ account, password, id: res.id });
        return Object.assign({}, res.dataValues, { msg: '登录成功', code: 200, token, password:
```

图 10　登录模块部分代码

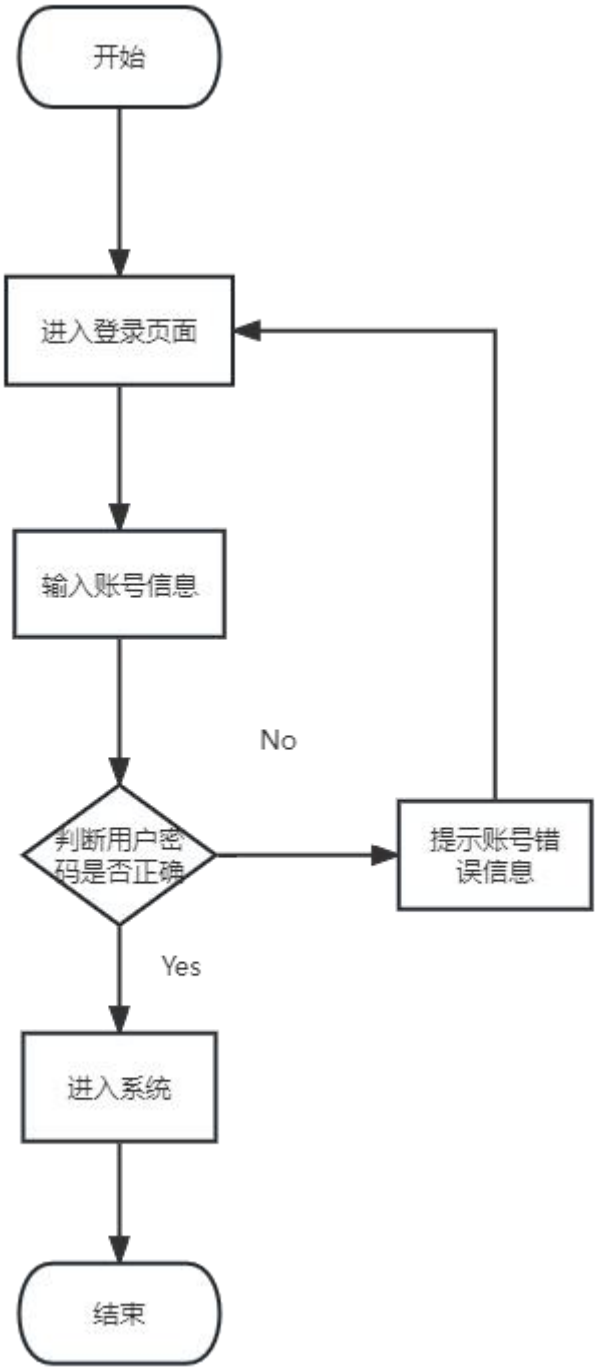

图 11　登录模块流程图

5.2 货物录入

该功能允许用户录入货物信息，包括货物名称、数量、价格等。该模块部分代码如图 12 所示。流程图如图 13 所示。

```
async createThings(ctx: Context) {
        const { data } = ctx.request.body;
        const { id } = ctx.userInfo;
        const res = await shopService.createThings(id, data);
        ctx.body = res;
}
```

图 12　货物录入代码

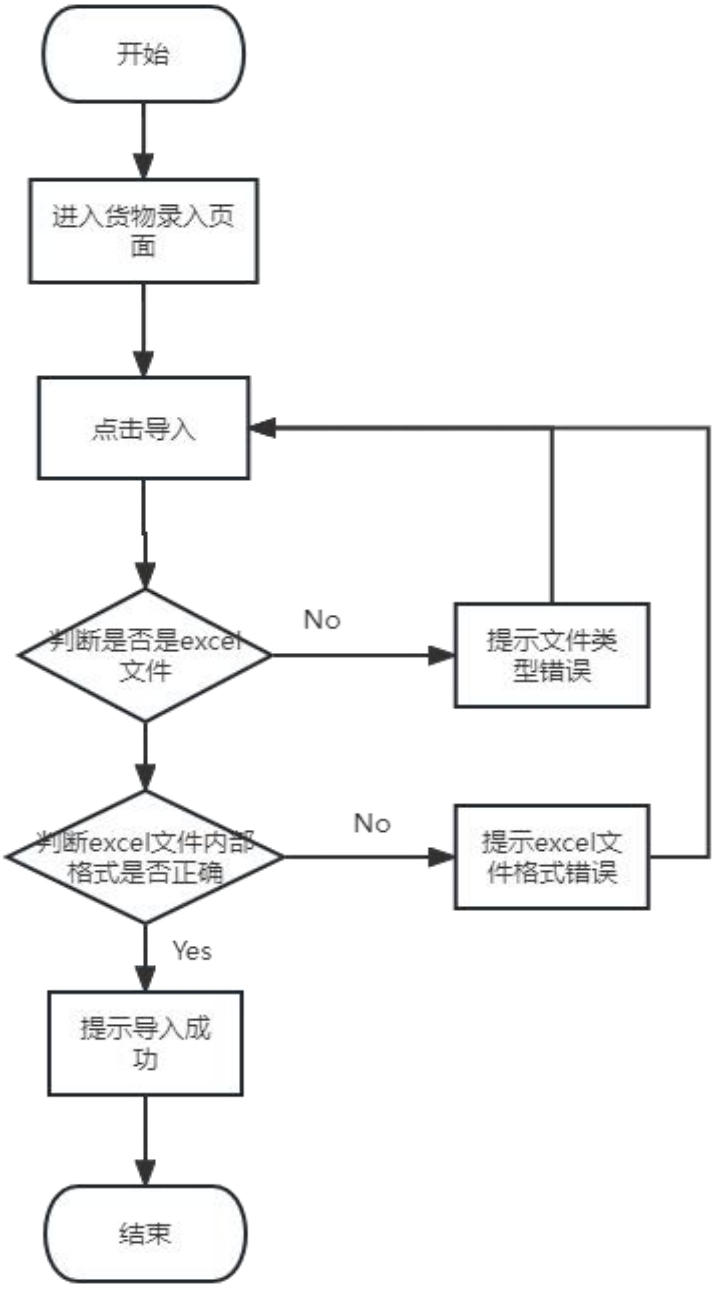

图 13　货物录入模块流程图

5.3 物流管理

该功能实现了对已发货物的管理，包括货物的状态跟踪、物流路线的规划和调整、状态、拦截、删除、更新等操作。该模块部分代码如图 14 所示。

```
// 创建物流
async createShop(ctx: Context, next: any) {
  const { id } = ctx.userInfo;

  const resulte = await shopService.createShop({ ...ctx.request.body, uid: id });
  ctx.body = resulte;
}

// 删除
async delete(ctx: Context, next: any) {
  const { id } = ctx.query;
  const res = await shopService.delete(id);
  ctx.body = res;
}

// 更新地址
async updateAddress(ctx: Context, next: any) {
  const { id, lng, lat, current_position } = ctx.request.body;

  ctx.body = await shopService.updateAddress({ id, lat, lng, current_position });
```

图 14　物流管理代码

5.4 发布物流

该功能允许用户发布货物信息和物流需求，包括货物类型、起点、终点、时间、价格等。该模块部分代码如图 15 所示。 流程图如图 16 所示。

```
const res: any = await ShopModel.create({
    ...shopInfo,
});
let qr_code = await QRCode.toDataURL(String(res.dataValues.id));
try {
  await ShopModel.update(
    { qr_code: qr_code },
```

图 15　物流发布代码

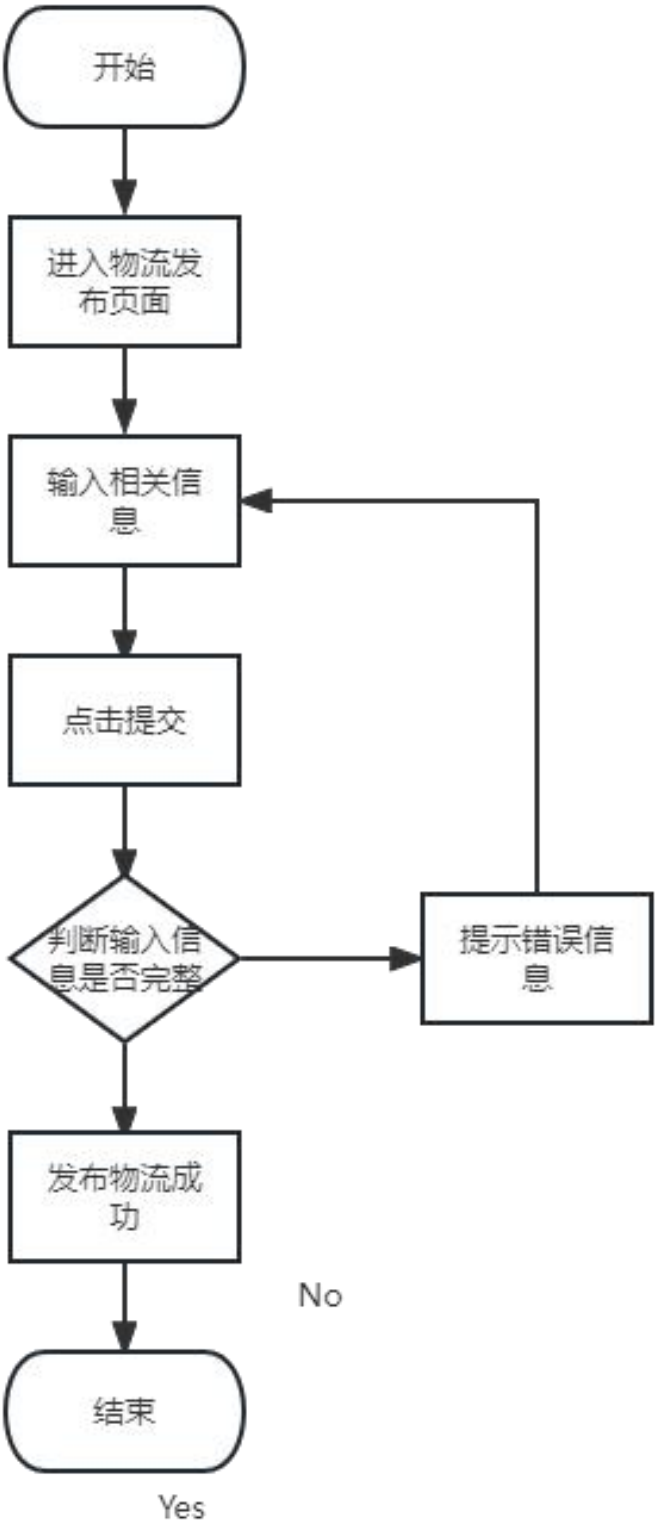

图 16　物流发布流程图

5.5 可视化追踪物流

该功能允许用户对已发布的物流订单进行可视化追踪，包括货物的实时位置、运输情况等改模块运用到了大量的可视化算法例如：运输路径规划算法，飞线图显示算法，流线图显示算法，地图旋转算法，矩阵转换算法等。该模块主要是展示功能，没有流程图。

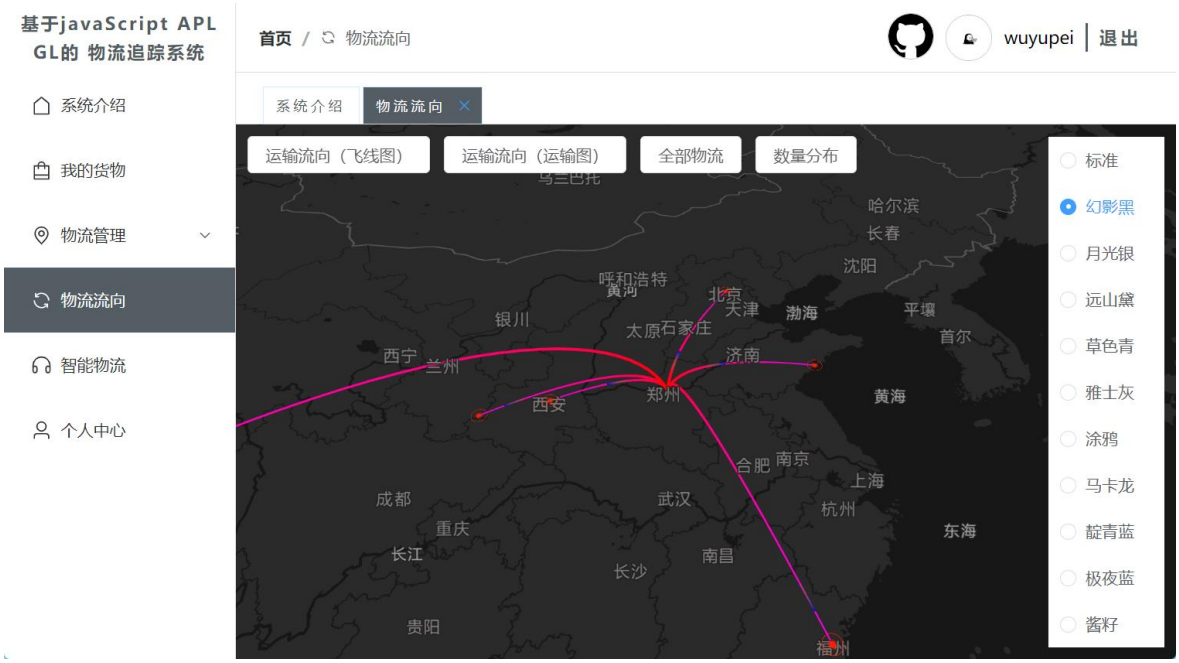

图 17　物流可视化追踪(飞线图)页面图

图 18　物流可视化追踪(物流分布)页面图

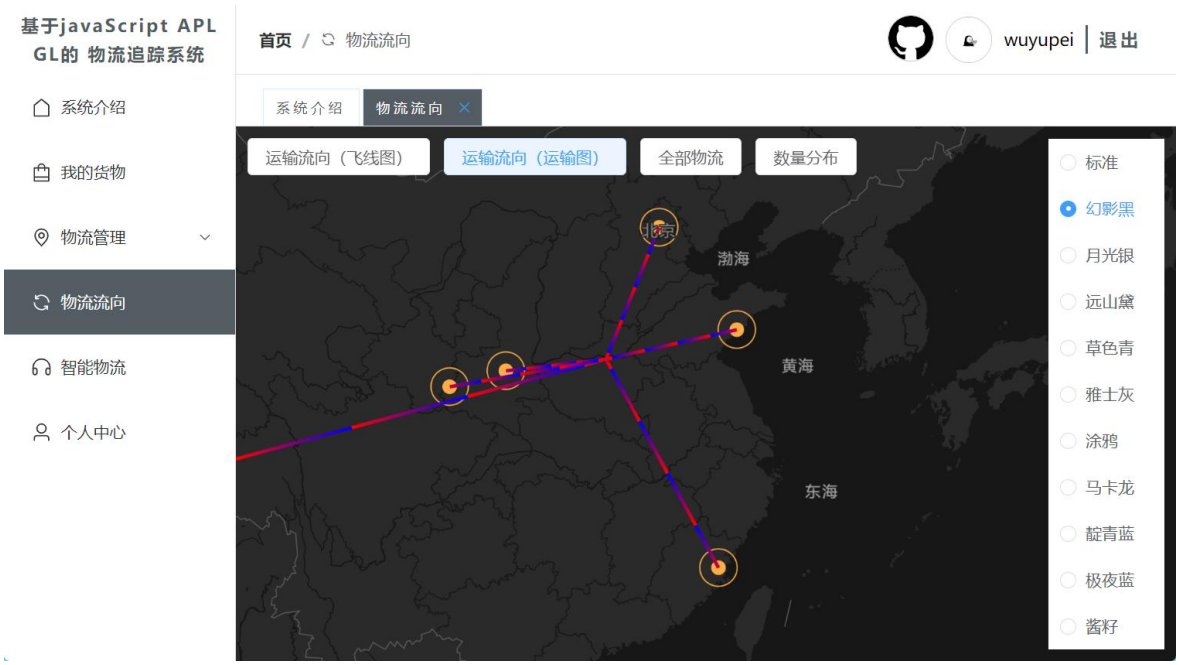

图 19　物流可视化追踪(流向)页面图

5.6 智能物流

该功能允用户根据物流订单号，查询物流信息。该模块部分代码如图 20 所示。具体流程图如图 21 所示。

```
async getInfoById(id: any) {
      const res = await ShopModel.findOne({
        where: {
          id,
        },
      });
const res = await TransportInfoModel.create({
      pid: id,
      current_position,
      current_position_geo,
      current_time,
    });
  }
  async createShop(ctx: Context, next: any) {
    const { id } = ctx.userInfo;

    const resulte = await shopService.createShop({ ...ctx.request.body, uid: id });
    ctx.body = resulte;
  }
```

图 20　智能物流代码

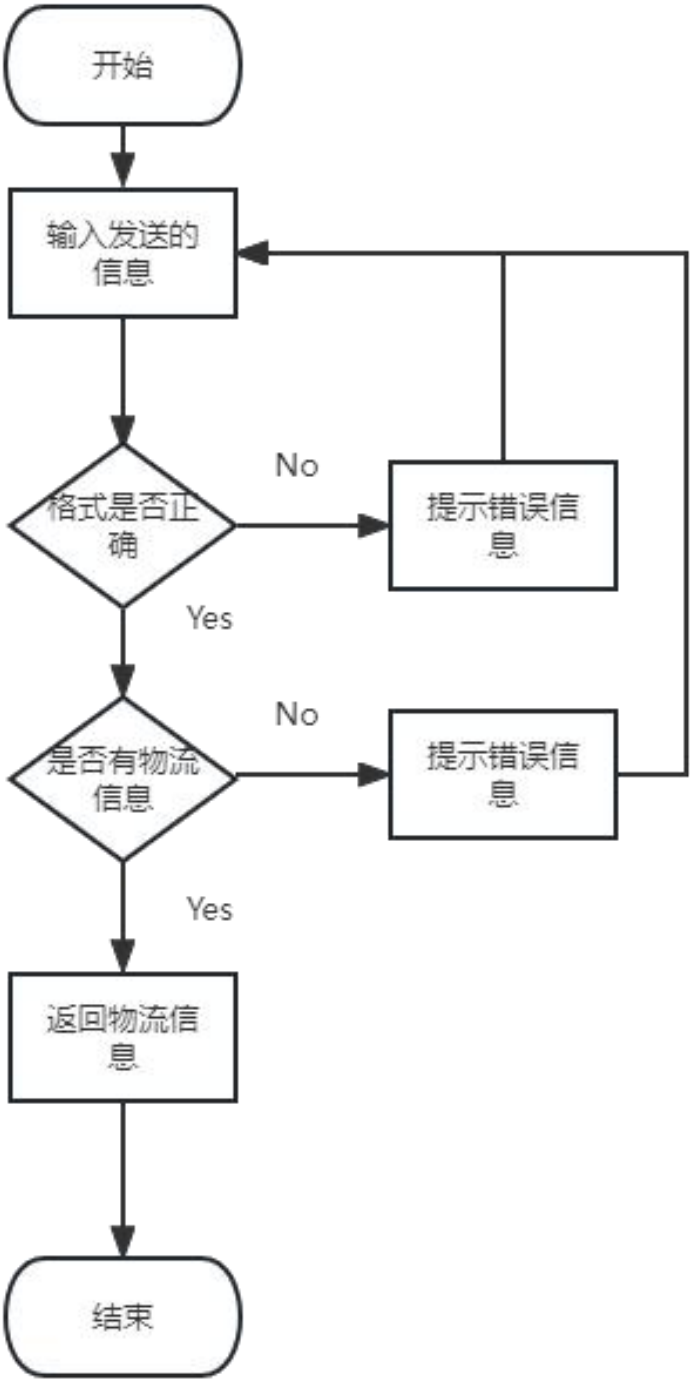
开始
输入发送的信息
格式是否正确
No
提示错误信息
Yes
是否有物流信息
No
提示错误信息
Yes
返回物流信息
结束

图 21 智能物流流程图

6 系统测试

6.1 测试环境

1. 数据库环境

表 5 数据库环境表

硬件环境	软件环境
服务器-cpu:2 核 1.00Ghz	数据库:MySQL_64bit
服务器-内存:2GB	数据库服务器 IP:8.141.63.127
	数据库操作系统:CentOS 7.6 64Bit
	数据库实例名:lch_db
	数据库用户:root

2. 应运环境

表 6 应用环境表

硬件环境	软件环境
服务器-cpu:2 核 1.0Ghz	中间件:Nginx
服务器-内存:2GB	数据库服务器 IP:8.141.63.127
	数据库操作系统:CentOS 7.6 64Bit

3. 客户端环境

表 7 客户端环境表

硬件	软件	网络环境
客户端-cpu:I7-7700HQ 四核 2.8Ghz 客户端-内存:16GB	操作系统:Windows10-64bit 浏览器:IE10、EDGE、Chrome、360	互联网:100/1000M 以太网

6.2 测试目的

验证基于 JavaScript GL API 的物流追踪系统是否能够按预期实现所需的功能。这包括追踪货物、显示地图和路线、提供实时更新等功能的正确性和可靠性。评估系统的用户界面设计和用户体验，了解用户对系统的满意度、易用性和交互性。评估系统的安全性，包括身份验证、数据加密、权限控制和防止潜在攻击的能力。

6.3 测试方法

根据开题报告中的需求，按照预定的测试计划进行测试，验证系统是否符合需求和功能是否可以正常使用。

1. 功能性测试：使用单元测试编写和执行针对系统各个功能模块的单元测试用

例，验证其功能的正确性。

2. 性能测试：使用负载测试模拟不同负载条件下的用户请求，评估系统在高负载情况下的性能表现，如响应时间和吞吐量。

6.4 测试用例

6.4.1 用户登录及修改个人信息测试用例

表 8 用户登陆及修改个人信息测试用例表

测试功能点	步骤	期望输出	实际输出	测试状态
用户登录及修改个人信息	1. 打开浏览器，输入网址 http://127.0.0.1:5173 2. 使用账号密码登录 3. 点击登录 4. 点击修改个人信息查看系统结束，系统操作指南等。	1. 密码输入正确，进入系统 2. 密码输入错误，提示登录失败 3. 修改个人信息	1. 进入系统 2. 提示登录失败 3. 成功修改个人信息	成功

图 22　登录成功测试用例图

图 23　修改个人信息图测试用例图

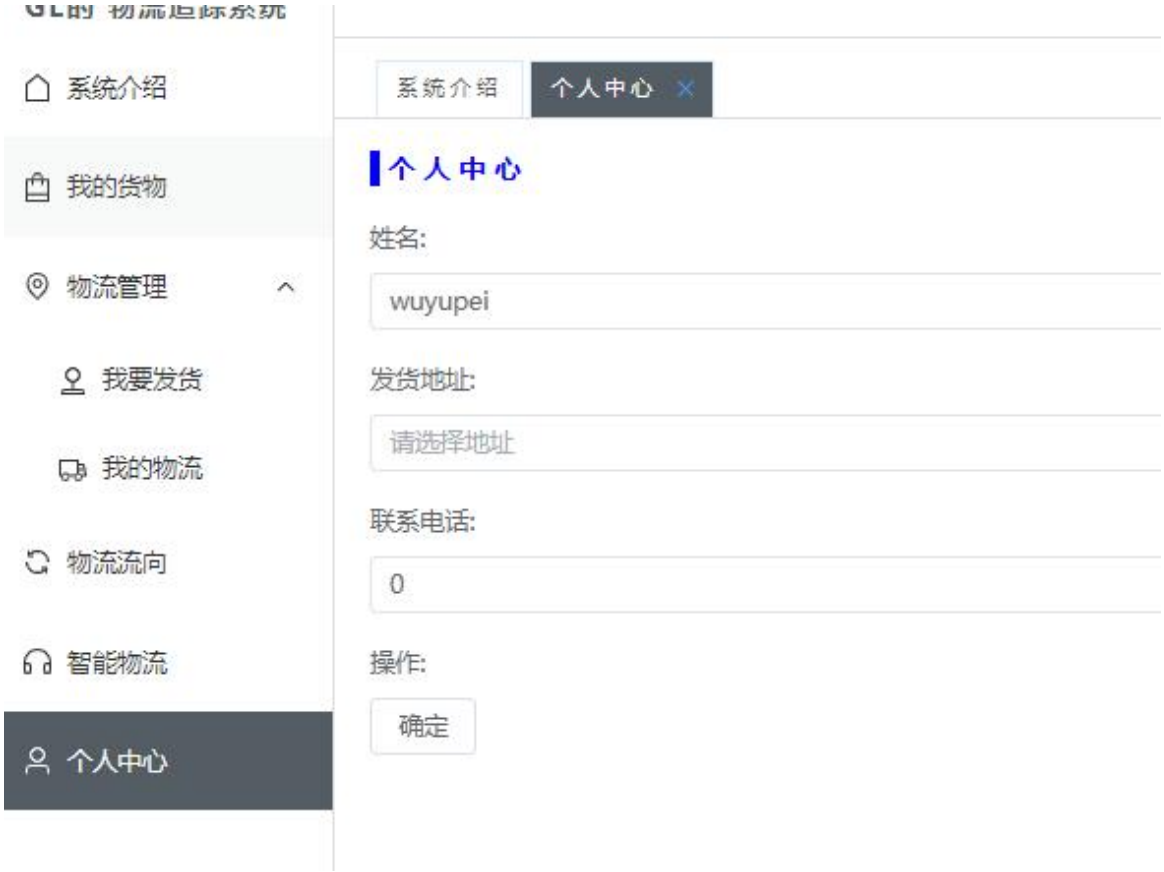

图 24　修改个人信息图测试用例图

图 25　登录失败测试用例图

6.4.2 录入货物测试用例

表 9 录入货物测试用例表

测试功能点	步骤	期望输出	实际输出	测试状态
录入货物	1. 打开浏览器，输入网址 http://127.0.0.1:5173 2. 使用账号密码登录 3. 点击登录 4. 点击编辑 录入 删除	1. 查看用户在平台中的货物 2. 录入货物 3. 编辑货物信息	1. 查看货物 2. 成功录入 3. 成功修改	成功

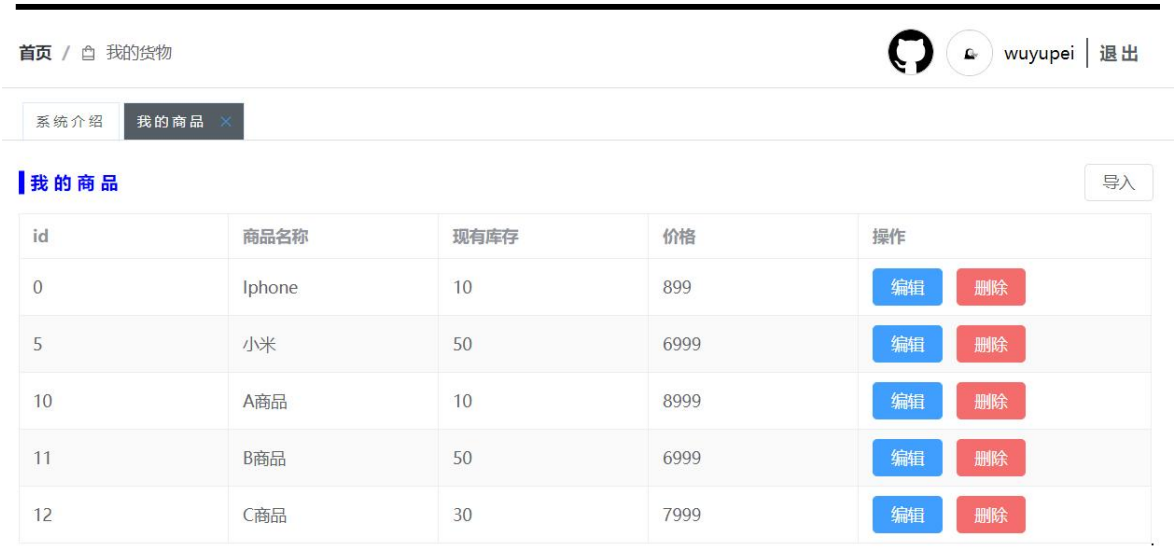

图 26　查看商品测试用例图

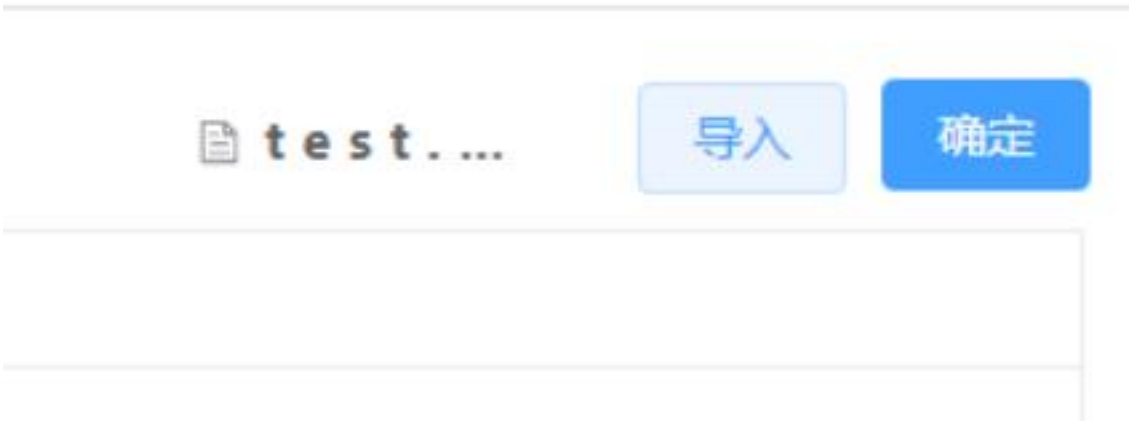

图 27　导入商品测试用例图

图 28　导入成功测试用例图

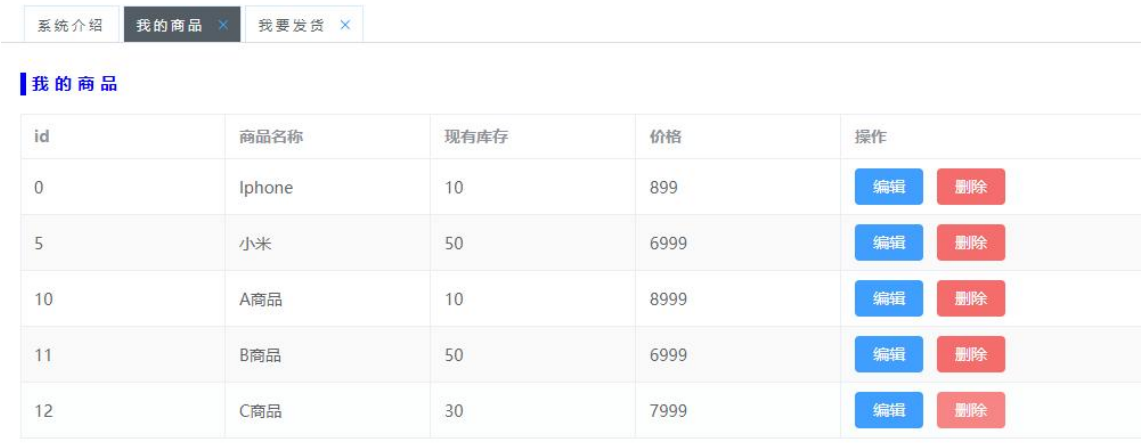

图 29　导入后测试用例图

图 30　编辑测试用例图

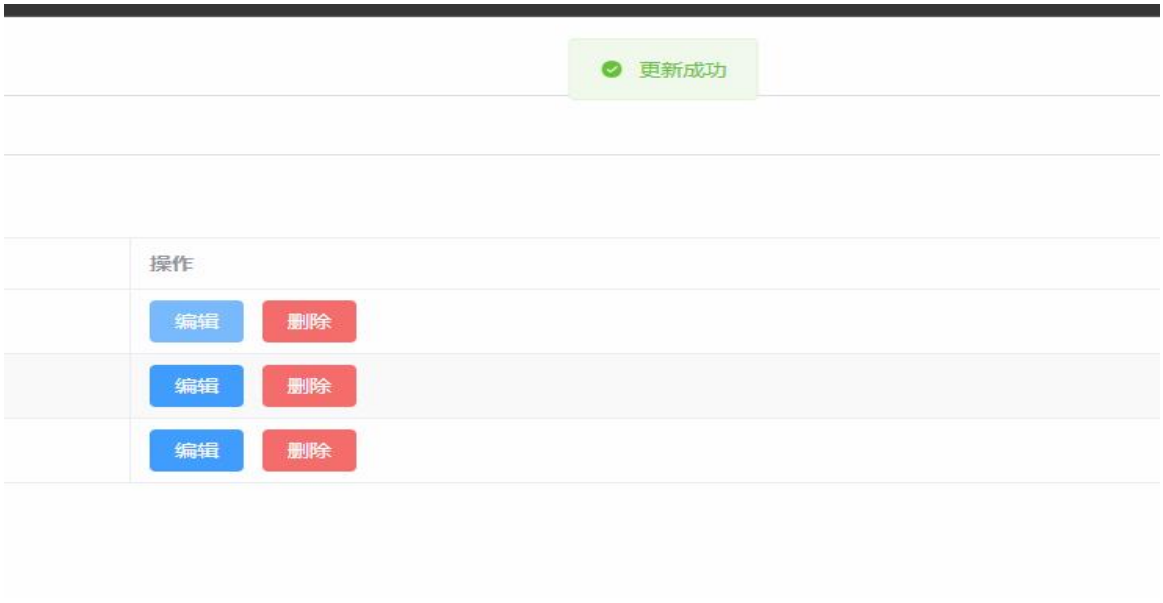

图 31　编辑成功测试用例图

图 32　删除成功测试用例图

6.4.3 发布物流测试用例

表 10 发布物流测试用例表

测试功能点	步骤	期望输出	实际输出	测试状态
发布物流	1. 打开浏览器，输入网址 http://127.0.0.1:5173 2. 使用账号密码登录 3. 点击登录 4. 点击发布物流,输入相关信息	1. 输入完整的相关信息，点击发货 2. 信息输入完整，发货成 3. 信息数不完整提示发货失败，请重新输入信息	1. 发货成功 2. 信息输入不完整,提示发货失败	成功

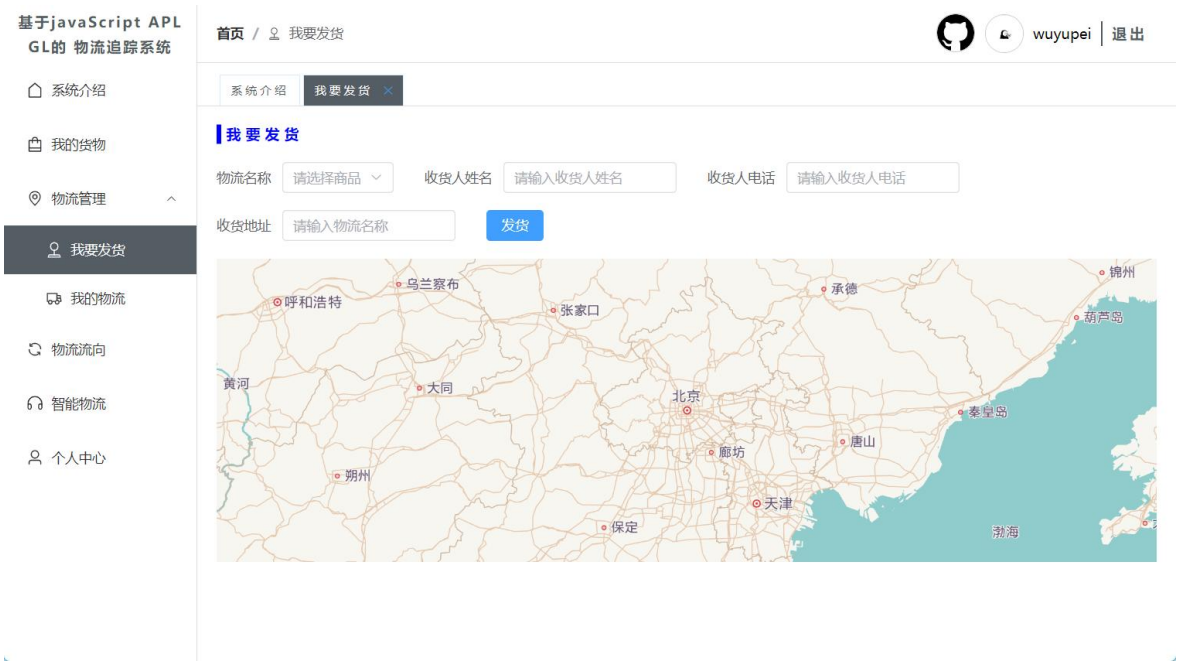

图 33　输入信息测试用例图

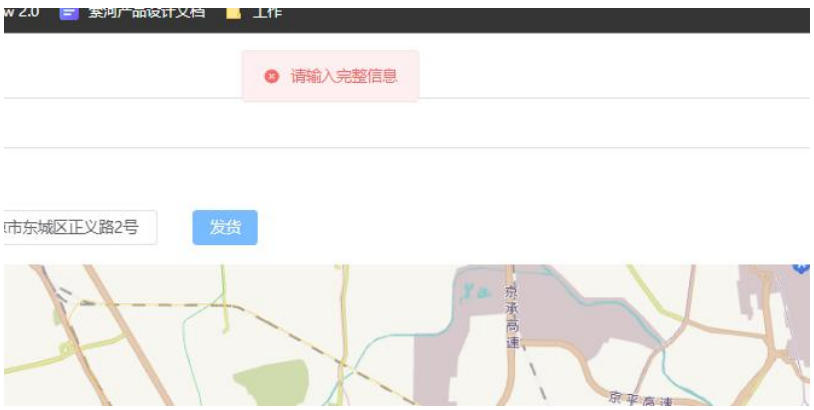

图 35　输入信息不完整测试用例图

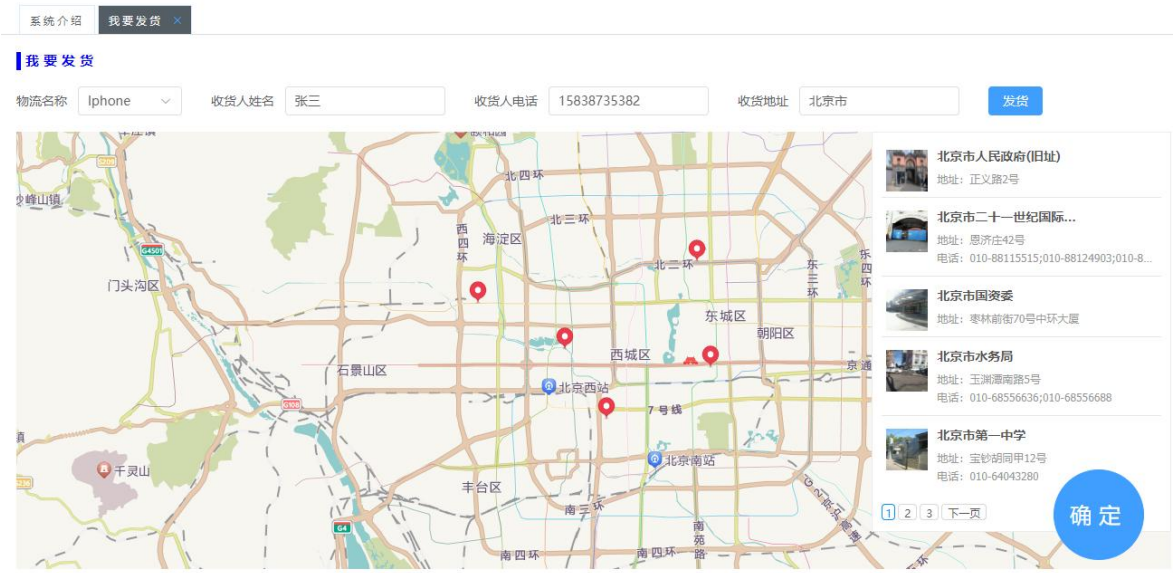

图 34　输入信息测试用例图

6.4.4 物流管理及追踪测试用例

表 11 物流管理及追踪测试用例表

测试功能点	步骤	期望输出	实际输出	测试状态
物流管理及追踪	1. 打开浏览器，输入网址 http://127.0.0.1:5173 2. 使用账号密码登录 3. 点击登录 4. 点击我的物流，进行物流详信息的查看，物流拦截，物流更新位置，物流删除，查看物流运输路线等操作	1. 查看各个物流的详细信息 2. 删除物流信息,更新物流信	1. 查看成功 2. 操作正常	成功

图 36　物流列表测试用例图

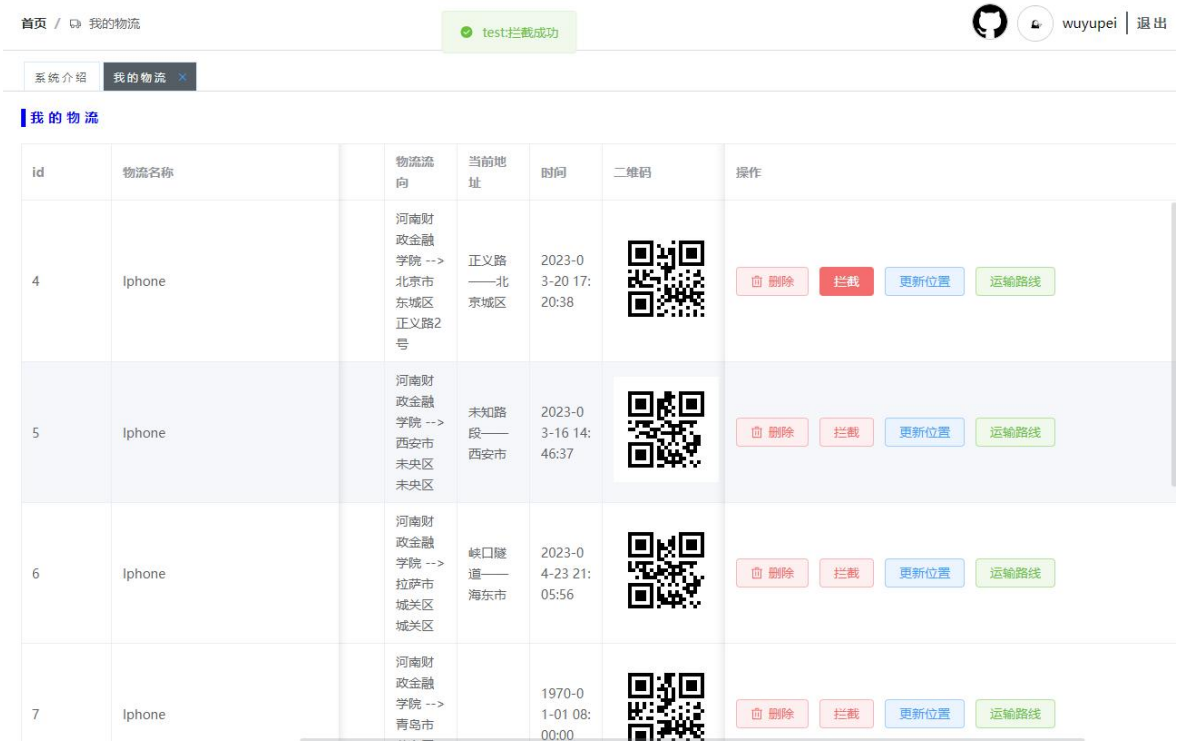

图 37　删除物流测试用例图

图 38　更新物流测试用例图

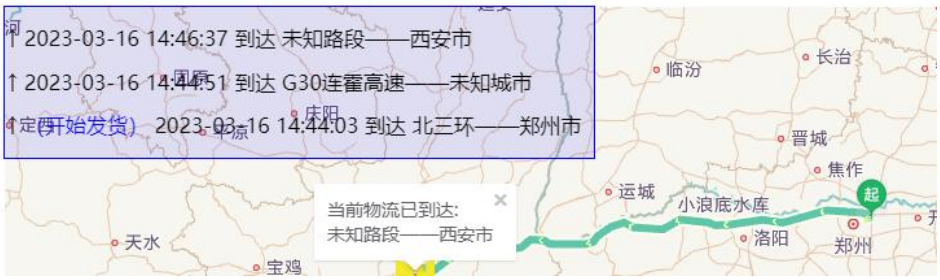

图 39　物流运输路线测试用例图

6.4.5 物流可视化追踪测试用例

表 12 物流可视化追踪测试用例表

测试功能点	步骤	期望输出	实际输出	测试状态
物流可视化追踪	1. 账号密码登录 2. 进入系统,点击物流流向	1. 可视化查看物流的运输路线	1. 查看成功	成功

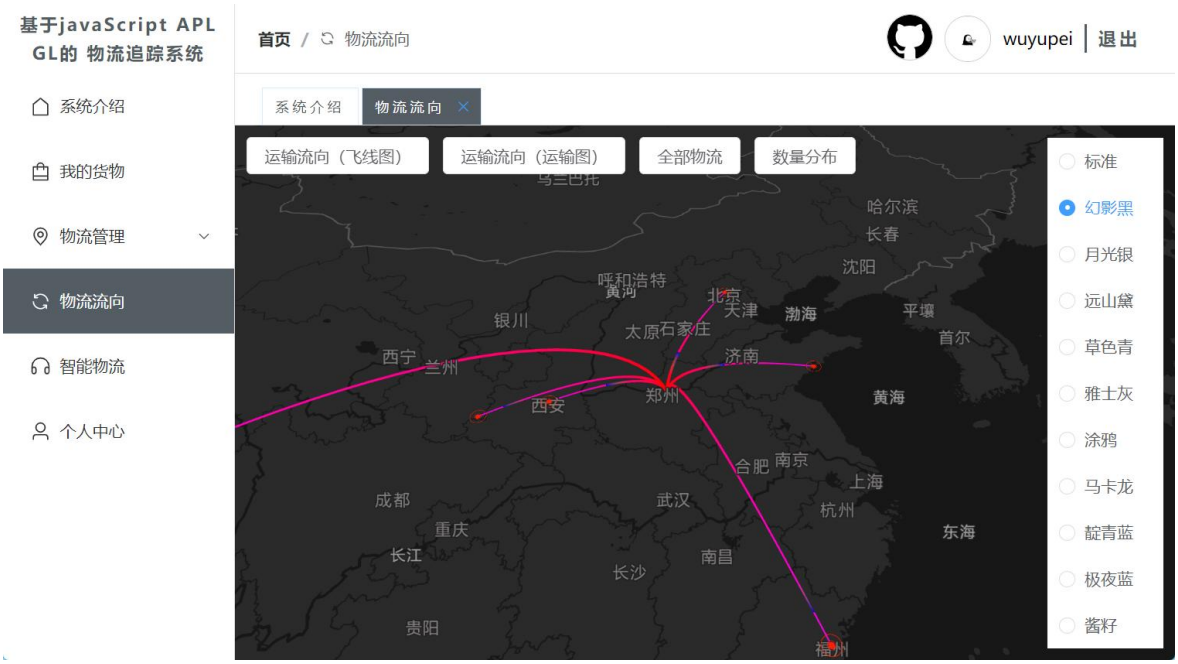

图 40 物流可视化(飞线图)测试用例图

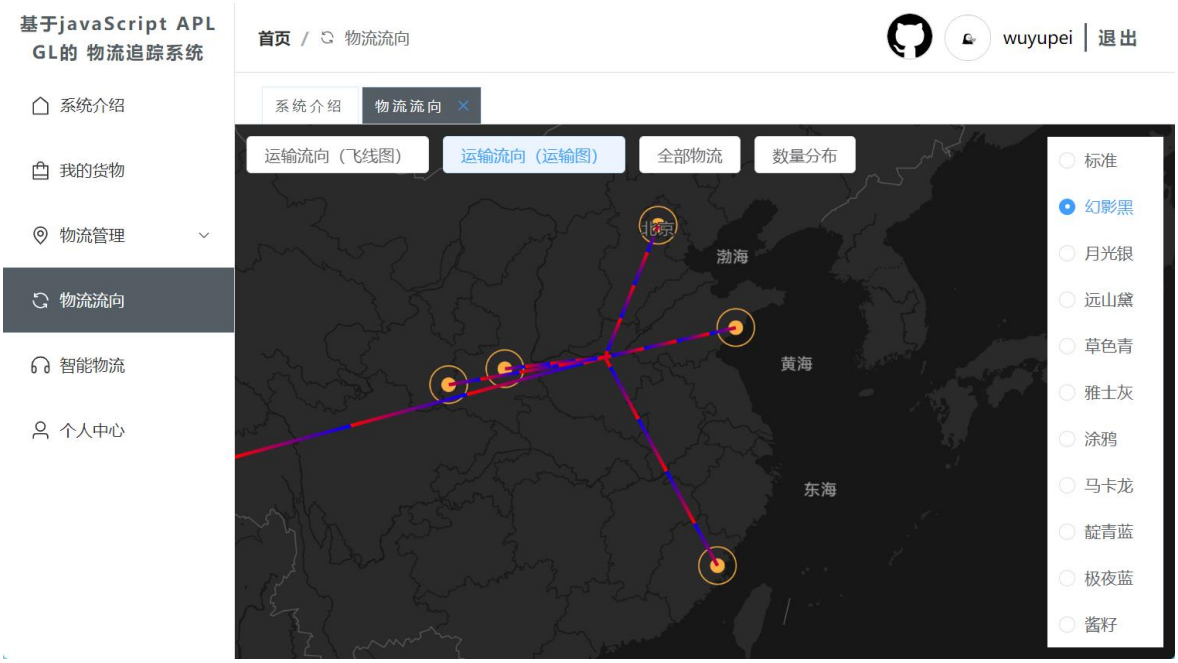

图 41 物流可视化(流向图)测试用例图

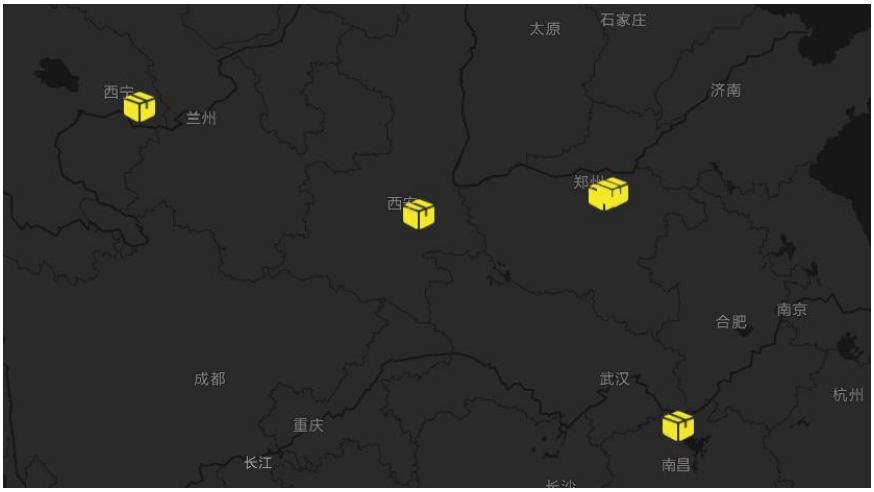

图 42　物流可视化(物流分布)测试用例图

图 43　物流可视化(物流详情)测试用例图

6.4.6 智能物流测试用例

表 13 智能物流测试用例表

测试功能点	步骤	期望输出	实际输出	测试状态
物流可视化追踪	1. 打开浏览器，输入网址 http://127.0.0.1:5173 2. 使用账号密码登录 3. 点击登录,点击智能物流	1. 给智能物流机器人发送物流编号 2. 查询该物流编号的详细信息	1. 查看成功	成功

图 44 智能物流测试用例图

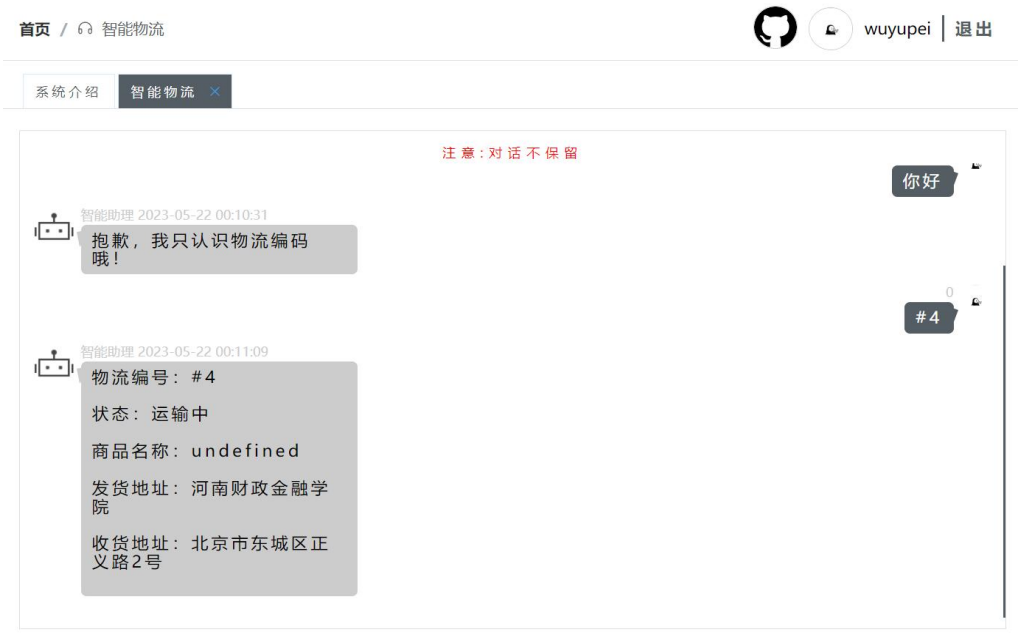

图 45 智能物流测试用例图

6.5 测试结论

6.5.1 用户登录及修改个人信息

1. 用户登录功能正常，可以成功登录系统。

2. 修改个人信息功能正常，可以成功更新个人信息。

3. 鉴权功能正常。

4. 登陆和修改功能是系统安全中最重要的部分，这部分内容确保系统的整体安全性和系统的稳定性。

6.5.2 货物录入

1. 货物录入功能正常，如果用户输入错误的信息，系统会给出友好的提示，输入正确的信息，可以成功添加货物信息。这部分对用户体验部分做了很大的优化，确保用户的体验度。

2. 货物信息字段完整，录入信息符合要求。录入时，需要选择正确的位置信息。

6.5.3 物流管理

1. 物流管理功能正常，可以成功添加、编辑、删除物流信息。

2. 物流信息字段完整，录入信息符合要求。

6.5.4 发布物流

1. 发布物流功能正常，可以成功发布物流信息。

2. 物流信息显示清晰明了，用户可以轻松了解物流详细情况包括物流位置，物流实时动态，物流流向等信息。

6.5.5 可视化追踪物流

1. 可视化追踪物流功能正常，可以实时显示货物位置和运输情况。

2. 地图显示清晰，可以准确反映货物位置和路线。

6.5.6 智能物流

1. 智能物流功能正常，可以发送不同 id 查询不同物流的详细信息。

7 结论与展望

7.1 结论

通过对高德地图 GL API 物流追踪系统的设计与实现，实现了一个功能完善、性能优良的物流追踪系统。系统不仅支持用户登录、物流管理、物流追踪、可视化展示、智能物流以及发送邮箱提醒等多种功能，还通过采用 Node.js 和 Koa 框架、MySQL 和 Redis 数据库等技术栈，保证了系统的高性能和稳定性。同时，系统还采用了高德地图的 GL API，实现了物流地图展示和路径规划等功能。

7.2 展望

未来，随着物流行业的不断发展和创新，物流追踪系统也需要不断地更新迭代。因此，我计划在未来的版本中加入更多的功能和技术。例如，可以采用人工智能技术来优化物流路径规划，提高运输效率和减少成本。同时，还可以引入更多的数据分析和可视化技术，以更好地展示物流数据，并为用户提供更好的体验和服务。此外，本系统将致力于提高系统的性能和稳定性，优化系统架构和数据库设计。相信，在未来的努力和创新下，物流追踪系统将成为物流行业的重要工具之一，为用户带来更好的体验和服务。

参考文献

[1] 谢欢,陈继努. 基于 WebGIS 和 GPRS 的智能交通系统设计与实现[J].计算机科学,2005(04):225-227.

[2] 周文业. 基于移动 GIS 的共享物流系统的设计与实现[D].西安电子科技大学,2018.

[3] 李晶. 二维码技术应用在农产品物流追溯系统中的分析和思考[J].电子技术与软件工程,2014,No.45(19):45+77.

[4] 朱健. 使用二维码要防信息泄露[D].中国防伪报道.2016:30+41.

[5] 孟志军,赵春江,王秀等. 基于 GPS 的农田多源信息采集系统的研究与开发[J].农业工程学报,2003(04):13-18.

[6] 王道军. 基于北斗卫星的物流追踪系统研究[D].北京邮电大学,2014.

[7] 钟聪儿,邱荣祖. RFID 在茶叶物流追踪与追溯中的关键应用技术[J].安徽农业大学学报,2016,43(06):1039-1044

[8] 吴小力. 基于智慧供应链的电力物流跟踪系统[J].机电工程技术,2019,48(12):35-36+83.

[9] Laura Meade.Strategic analysis of logistics and supply chain managmentsystems using the analytical network process Jl.Transportation Research PartE:Logistics and Transportation Review, 1998,34(2):201-215.

[10] Ramamurthy, Bhargavi, ShashiKumar. Development of a Low-Cost GSMS MS-Based Humidity Remote Monitoring and Control system for IndustrialApplications. Journal of Advanced Computer Science and Applications, 2010.019.

[11] 柴凤伟. 货物安全的重要保障[N]. 现代物流报,2007-05-24(008).

[12] 徐晶文. 地区电业局农网台变终端通讯方式研究.湖南大学.2013.

[13] 苏峰. 物流信息技术在平板显示企业材料物流管理应用展望[J].物流工程与管理,2018,40(08):56-57+111.

[14] 陆祥瑞. 物流动态管理信息系统设计、开发及建立[J].计算机仿真,1994(04):17-23.

[15] 夏绪宏. 基于 GPS/GIS 的物流车辆跟踪系统研究[D].北京邮电大学,2010.

致　谢

在此向我的毕业设计导师、辅导员以及所有帮助过我的同学表达我的最深的感激之情。您们的支持和鼓励是我完成这个项目的重要动力。

首先，非常感谢我的毕业设计导师，在我整个设计过程中提供了专业的指导和协助。在我遇到困难和疑惑的时候，导师总是耐心地解答我的问题，给我提供宝贵的建议和意见。他的认真负责和对我的指导和鼓励让我不断提高自己的能力，让我对未来充满信心。

同时，也要感谢我的辅导员，她不仅在我毕业设计的过程中给予了我帮助，还在我的整个大学生活中给予我很多的关心和支持。她的热心帮助和耐心指导让我感受到大家庭的温暖，也让我在大学生活中学会了很多珍贵的经验和知识。

此外，还要感谢我的同学们。在整个设计过程中，他们给予我很多帮助和支持。每当我遇到困难和疑惑的时候，他们总是毫不犹豫地伸出援手，帮助我克服困难。他们的鼓励和支持让我有信心和勇气克服困难，不断提升自己的能力和水平。

最后，再次感谢所有帮助过我的老师、同学和朋友们。你们的支持和激励激发了我战胜困难的信心和勇气。在此，向你们表达最深的谢意，也希望能够在未来的日子里继续与你们保持联系，共同进步，共同成长。

附录 1 核心部分代码

```
import KoaRouter from 'koa-router';
import shopController from '../controller/shop.controller';
import veryToken from '../middleware/veryToken';

const shopRouter: KoaRouter = new KoaRouter();

shopRouter.get('/shop', veryToken, shopController.findByUid);

shopRouter.post('/createshop', veryToken, shopController.createShop);

shopRouter.get('/delete', veryToken, shopController.delete);

shopRouter.post('/updateAddress', veryToken, shopController.updateAddress);

shopRouter.post('/getInfoById', veryToken, shopController.getInfoByid);

shopRouter.post('/addATransportInfo', veryToken, shopController.addATransportInfo);

shopRouter.post('/getTransportInfo', veryToken, shopController.getTransportInfo);

shopRouter.post('/createThings', veryToken, shopController.createThings);

shopRouter.get('/getAllThings', veryToken, shopController.getAllThings);

shopRouter.delete('/deleteOneThing', veryToken, shopController.deleteOneThing);

shopRouter.post('/changeOneThing', veryToken, shopController.changeOneThing);

export default shopRouter;

import adminService from '../service/admin.service';
import { verifyToken } from '../utils';

class LoginController {
  async createUser(ctx: any, next: any) {
    const { account, password, uname, idcard, yue } = ctx.request.body;

    ctx.body = await adminService.createUser(account, password, uname, idcard, yue);
  }

  async changePassword(ctx: any, next: any) {
    const { number, idcard, newpassword } = ctx.request.body;
    ctx.body = await adminService.changePassword(number, idcard, newpassword);
  }

  async look(ctx: any, next: any) {
    const { idcard, number } = ctx.request.body;
    ctx.body = await adminService.look(number);
  }

  async down(ctx: any, next: any) {
    const { number, money } = ctx.request.body;
    ctx.body = await adminService.down(number, money);
  }
```

```
  async zhuan(ctx: any, next: any) {
    const { number, newnumber, money } = ctx.request.body;
    ctx.body = await adminService.zhuan(number, newnumber, money);
  }

  async person(ctx: any, next: any) {
    const token: any = ctx.header?.authorization?.replace('Bearer ', '');
    const res: any = await verifyToken(token);
    ctx.body = await adminService.person(res.id);
  }
}

export default new LoginController();

import * as QRCode from 'qrcode';
import Thing from '../model/thing';
import ShopModel from '../model/shop.model';
import TransportInfoModel from '../model/transportInfo.model';

class ShopService {
  async findByUid(uid: any) {
    const res = await ShopModel.findAll({
      where: {
        uid: uid,
      },
      include: { model: Thing, as: 'thing' },
    });
    return res;
  }

  async createShop(shopInfo: any) {
    console.log(shopInfo);

    const res: any = await ShopModel.create({
      ...shopInfo,
    });

    let qr_code = await QRCode.toDataURL(String(res.dataValues.id));

    try {
      await ShopModel.update(
        { qr_code: qr_code },
        {
          where: { id: String(res.dataValues.id) },
        }
      );
    } catch (error) {
      return error;
    }

    const finalRes = await ShopModel.findOne({
      where: {
        id: res.dataValues.id,
      },
    });

    return { ...finalRes, msg: '发货成功' };

  }
```

```
async delete(id: any) {
  const res = await ShopModel.destroy({
    where: {
      id,
    },
  });

  return { code: 200, msg: '删除成功', id };
}

async updateAddress(info: any) {
  let res = await ShopModel.update(
    {
      current_position_geo: `${info.lng},${info.lat}`,
      current_position: info.current_position,
      current_time: +new Date(),
    },
    {
      where: {
        id: info.id,
      },
    }
  );
  return res;
}

async getInfoById(id: any) {
  const res = await ShopModel.findOne({
    where: {
      id,
    },
  });
  if (!res) {
    return [];
  } else {
    return [res];
  }
}

async addATransportInfo(data: any) {
  const { id, current_position, current_position_geo, current_time } = data;
  const res = await TransportInfoModel.create({
    pid: id,
    current_position,
    current_position_geo,
    current_time,
  });

  return { code: 200, msg: 'add success', data: res };
}

async getTransportInfo(id: any) {
  const res = await TransportInfoModel.findAll({
    where: {
      pid: id,
    },
    order: [['current_time', 'DESC']],
```

```
    });
    return { code: 200, data: res };
  }

  async createThings(uid: any, data: any) {
    const { name, price, total } = data;
    try {
      for (let i = 0; i < name.length; i++) {
        await Thing.create({
          uid: uid,
          name: name[i],
          price: price[i],
          total: total[i],
          count: total[i],
        });
      }
    } catch (error) {
      return { code: 200, msg: error };
    }

    return { code: 200, msg: '导入成功' };
  }

  async getAllThings(id: any) {
    const res = await Thing.findAll({
      where: {
        uid: id,
      },
    });

    return { code: 200, data: res };
  }

  async deleteOneThing(id: any) {
    try {
      let res = await Thing.destroy({
        where: {
          id: id,
        },
      });
      console.log(res);

      return { code: 200, msg: '删除成功' };
    } catch (error) {
      console.log(error);

      return { code: 201, msg: error };
    }
  }

  async changeOneThing(id: any, name: any, price: any, total: any) {
    try {
      let res = await Thing.update(
        { name, price, total },
        {
          where: {
            id,
```

```
            },
          }
        );
        if (res.length) {
          return { code: 200, msg: '更新成功' };
        } else {
          return { code: 201, msg: '更新失败' };
        }
      } catch (error) {
        return { code: 201, msg: '更新失败' };
      }
    }
}

export default new ShopService();
import Application, { Next, Context } from 'koa';
import * as log4js from 'log4js';
import path = require('path');

log4js.configure({
  appenders: {
    file: {
      type: 'file',
      filename: path.resolve(__dirname, '../../../graduation-design-backend-logger/logger.log'),
      encoding: 'utf-8',
      layout: {
        type: 'pattern',
        pattern: '{"date":"%d","level":"%p","category":"%c","host":"%h","pid":"%z","data":\'%m\'}',
      },
      backups: 5,
      compress: false,
      keepFileExt: true,
    },

    datafile: {
      type: 'file',
      filename: path.resolve(__dirname, '../../../graduation-design-backend-logger/datafilelogger.log'),
      encoding: 'utf-8',
      layout: {
        type: 'pattern',
        pattern: '{"date":"%d","level":"%p","category":"%c","host":"%h","pid":"%z","data":\'%m\'}',
      },
      // 日志文件按日期（天）切割
      pattern: '-yyyy-MM-dd',
      // 回滚旧的日志文件时，保证以 .log 结尾 （只有在 alwaysIncludePattern 为 false 生效）
      keepFileExt: true,
      // 输出的日志文件名是都始终包含 pattern 日期结尾
      alwaysIncludePattern: true,
    },

    panel: {
      type: 'stdout',
    },
  },

  categories: {
    default: {
```

```
        appenders: ['file'],
        level: 'All',
      },
      datafile: {
        appenders: ['datafile'],
        level: 'All',
      },

      panel: {
        appenders: ['panel'],
        level: 'All',
      },
    },
});

const logger = log4js.getLogger('datafile');

export default function (app: Application) {
  app.use(async (ctx: Context, next: Next) => {
    logger.info(
      `Ip:${ctx.request.ip.split(':')[3]}      Methou:${ctx.request.method}      Url:${ctx.request.url}
Host:${ctx.request.header.host} Response:${ctx.response.message} Status:${ctx.response.status}`
    );
    await next();
  });
}


import useAppStore from '../../store/app';

import { findShop } from '../../service/home';
const mapRef = ref(null);

let appStore = useAppStore();
let currentStyle = ref('normal');
let styles = [
  { label: '标准', name: 'normal' },
  { label: '幻影黑', name: 'dark' },
  { label: '月光银', name: 'light' },
  { label: '远山黛', name: 'whitesmoke' },
  { label: '草色青', name: 'fresh' },
  { label: '雅士灰', name: 'grey' },
  { label: '涂鸦', name: 'graffiti' },
  { label: '马卡龙', name: 'macaron' },
  { label: '靛青蓝', name: 'blue' },
  { label: '极夜蓝', name: 'darkblue' },
  { label: '酱籽', name: 'wine' },
];
function radioChange(style) {
  console.log(`amap://styles/${style}`);
  mapRef.value.getMap().setMapStyle(`amap://styles/${style}`);
}

let shops = reactive({});
```

```
findShop(appStore.userInfo.id).then((res) => {
  shops = res.data;
});

function showShopFlow() {
  const map = mapRef.value.getMap();
  const loca = mapRef.value.getLoca();
  loca.clear();
  map.setZoom(5);
  map.setPitch(40);

  let linkLayer = new Loca.PulseLinkLayer({
    zIndex: 20,
    opacity: 1,
    visible: true,
    zooms: [2, 22],
  });
  // 线
  let features = shops.map((item, index) => {
    return {
      type: 'Feature',
      properties: {
        type: item.status,
      },
      geometry: {
        type: 'LineString',
        coordinates: [item.start_position_geo.split(','), item.end_position_geo.split(',')],
      },
    };
  });

  const source = new Loca.GeoJSONSource({
    data: {
      type: 'FeatureCollection',
      features,
    },
  });
  linkLayer.setSource(source);
  linkLayer.setStyle({
    unit: 'meter',
    dash: [40000, 0, 40000, 0],
    lineWidth: function () {
      return [15000, 5000];
    },
    height: function (index, feat) {
      return feat.distance / 3 + 10;
    },
    smoothSteps: 30,
    speed: 150000,
    flowLength: 100000,
    lineColors: function (index, feat) {
      return feat.link.properties.type === 0 ? ['#25CDEA'] : ['red', '#ff00ee'];
    },
    maxHeightScale: 0.3, // 弧顶位置比例
    headColor: 'rgba(0, 0, 255, 1)',
    trailColor: 'rgba(255, 255,0,0)',
  });
```

```
loca.add(linkLayer);

// 红色 呼吸点
const pointsRed = shops
  .filter((item, index) => item.status == 0)
  .map((item, index) => {
    return {
      type: 'Feature',
      properties: {
        type: 1,
      },
      geometry: {
        type: 'Point',
        coordinates: item.end_position_geo.split(','),
      },
    };
  });
var geoLeveRed = new Loca.GeoJSONSource({
  data: {
    type: 'FeatureCollection',
    features: pointsRed,
  },
});
var breathRed = new Loca.ScatterLayer({
  loca: loca,
  zIndex: 113,
  opacity: 1,
  visible: true,
  zooms: [2, 22],
});

breathRed.setSource(geoLeveRed);

breathRed.setStyle({
  unit: 'meter',
  size: [100000, 100000],
  borderWidth: 0,
  texture: 'https://a.amap.com/Loca/static/loca-v2/demos/images/breath_yellow.png',
  duration: 500,
  animate: true,
});

// 黄色呼吸点
const pointsYellow = shops
  .filter((item, index) => item.status == 1)
  .map((item, index) => {
    return {
      type: 'Feature',
      properties: {
        type: 1,
      },
      geometry: {
        type: 'Point',
        coordinates: item.end_position_geo.split(','),
      },
    };
  });
```

```
var geoLeveYellow = new Loca.GeoJSONSource({
  data: {
    type: 'FeatureCollection',
    features: pointsYellow,
  },
});
var breathYellow = new Loca.ScatterLayer({
  loca: loca,
  zIndex: 112,
  opacity: 1,
  visible: true,
  zooms: [2, 22],
});
breathYellow.setSource(geoLeveYellow);
breathYellow.setStyle({
  unit: 'meter',
  size: [100000, 100000],
  texture: 'https://a.amap.com/Loca/static/loca-v2/demos/images/breath_red.png',
  borderWidth: 0,
  duration: 1000,
  animate: true,
});

// 启动渲染动画
loca.animate.start();


unction showShopTransport() {
  const map = mapRef.value.getMap();
  map.clearMap();
  mapRef.value.getLoca().clear();
  map.setZoom(6);
  map.setPitch(0);

  // 线
  let inLineSource = new Loca.GeoJSONSource({
    data: shops.reduce(
      (p, cur) => {
        p.features.push({
          type: 'Feature',
          properties: {
            type: 1,
          },
          geometry: {
            type: 'LineString',
            coordinates: [cur.start_position_geo.split(','), cur.end_position_geo.split(',')],
          },
        });
        return p;
      },
      { type: 'FeatureCollection', features: [] }
    ),
  });

  var outLinelayer = new Loca.PulseLineLayer({
    // loca,
    zIndex: 11,
```

```
  opacity: 1,
  visible: true,
  zooms: [2, 22],
});

outLinelayer.setStyle({
  altitude: 0,
  lineWidth: 3,
  headColor: 'red',
  trailColor: 'blue',
  interval: 0.25,
  duration: 5000,
});
outLinelayer.setSource(inLineSource);
mapRef.value.getLoca().add(outLinelayer);

// 呼吸点
var scatter = new Loca.ScatterLayer({
  zIndex: 10,
  opacity: 1,
  visible: true,
  zooms: [2, 22],
});

var scatterGeo = new Loca.GeoJSONSource({
  data: shops.reduce(
    (p, cur) => {
      p.features.push({
        type: 'Feature',
        properties: {
          type: 0,
        },
        geometry: {
          type: 'Point',
          coordinates: cur.end_position_geo.split(','),
        },
      });
      return p;
    },
    { type: 'FeatureCollection', features: [] }
  ),
});
scatter.setSource(scatterGeo);
scatter.setStyle({
  size: [50, 50],
  borderWidth: 0,
  texture: 'https://a.amap.com/Loca/static/loca-v2/demos/images/breath_yellow.png',
  duration: 2000,
  animate: true,
});
mapRef.value.getLoca().add(scatter);

mapRef.value.getLoca().animate.start();
```