# 1. 系统需求分析

## 1.1 可行性分析

在该项目中，我们将采用Kafka技术对货运和物流服务进行重构。通过使用微服务架构、服务发现、熔断器、OAuth2授权服务器、API网关、配置中心和日志追踪等技术，我们可以提高系统的可扩展性、稳定性和安全性。

## 1.2 项目要解决的问题

该项目旨在解决货运和物流过程中的一系列问题，并通过引入微服务架构和相关技术来改善系统的性能、可靠性和用户体验。具体包括货物管理、订单管理、位置追踪、交付安排和运输费用计算等功能。

## 1.3 项目目标

该项目的目标是设计和实现一个基于微服务架构的高性能、可靠、安全的货运和运输网络应用程序。该应用程序将提供货物管理、运输管理、位置追踪、交付安排和运输费用计算等功能。通过API网关和OAuth2授权服务器，用户可以安全地访问系统。配置中心和日志追踪将帮助我们集中管理和跟踪系统的配置和运行状态。

# 2. 功能设计

## 2.1 功能划分

主要功能：

- 用户管理

- 订单管理

- 订单状态追踪

- 交付安排

## 2.2 功能描述

- 用户管理：用户可以注册、登录和注销。已登录用户可以查看和编辑个人信息。

- 订单管理：用户可以创建、修改和删除订单。用户可以查看订单的状态。

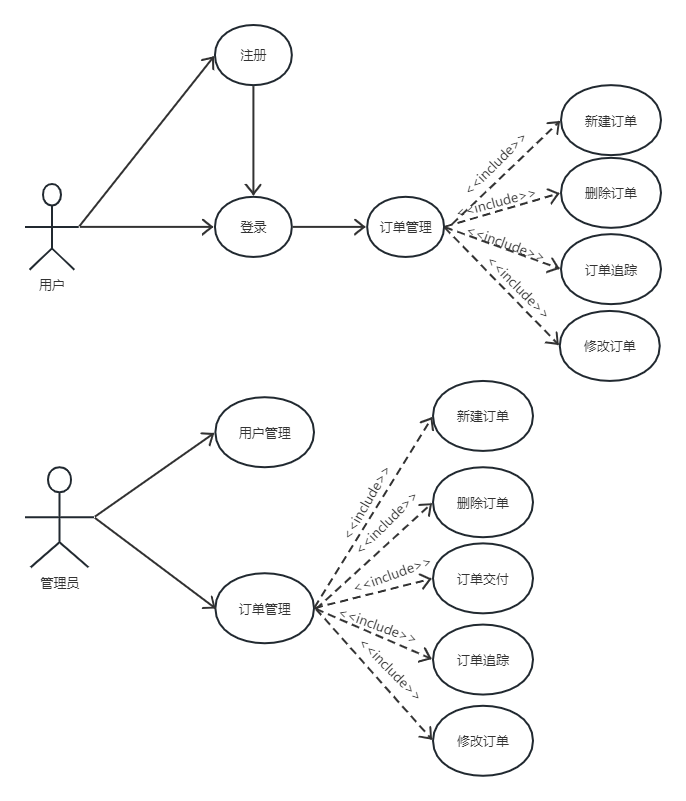
- 订单状态追踪：用户可以查看订单的当前状态和历史轨迹。

- 交付安排：用户可以安排货物的交付并跟踪交付状态。

## 2.3 功能设计

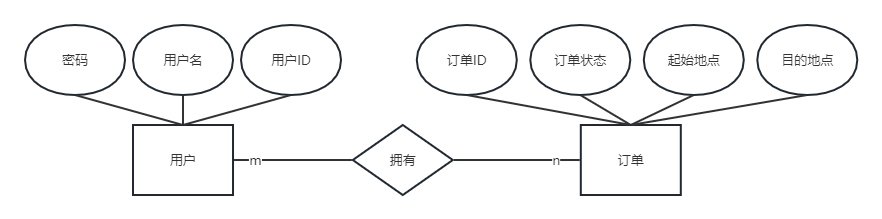
### 2.3.1 用例图

以下是应用程序的用例图：



### 2.3.2 ER图

以下是该应用程序的E-R图：



# 3. 详细设计

## 3.1 后台设计

在后台设计中，我们将使用Kafka技术对原有的Spring Cloud、 Spring MVC和Spring Data JPA/Mybatis进行重构。通过使用微服务架构、服务发现、熔断器和OAuth2授权服务器，我们将提高系统的可靠性、弹性和安全性。在每次添加新功能和组件后，我们将进行连续的单元测试。

## 3.2 前台设计

在前台设计中，我们将继续使用Ajax和Vue.js来实现与后台的数据交互和动态内容的呈现。通过使用这些技术，我们可以实现无刷新页面更新、异步请求数据和动态展示数据等功能。

# 4. 数据操作及表格设计

## 4.1 数据操作

在数据操作方面，我们将继续使用Mybatis Plus，并结合RESTful API进行对象关系映射（ORM）和持久性存储。这样可以使我们的API更加灵活和可扩展。

## 4.2 表格设计

在数据库方面，我们将根据新的需求进行适当的调整，并创建新的表格来满足功能的扩展。

# 5. 开发环境

## 5.1 开发工具

我们将使用以下工具和技术来开发该应用程序：

- JDK 8+

- Spring Cloud

- Spring MVC

- Spring Data JPA/Mybatis

- Spring Security

- OAuth2

- Mybatis Plus

- MySQL

- Ajax

- Vue.js

- Kafka

## 5.2 结构：微服务架构

我们将采用微服务架构，并通过集成Kafka消息队列实现事件驱动架构。每个微服务将充当事件生产者或消费者，以实现松耦合、可扩展的系统架构。通过事件驱动架构，微服务可以通过发布和订阅事件进行异步通信和解耦。这将提高系统的可靠性和弹性，同时支持高吞吐量的事件处理。