

基于全信创环境下的物流信息数据可视化中台

技术文档

摘要： 现代信息化时代，国家大力发展信创(信息技术应用创新)产业链，包括芯片、系统软件、中间件和应用软件。针对国产中间件与传统架构适配度较低，开发难度大、时间长的问题，提出一套基于国产操作系统、核心中间件的物流信息数据可视化中台系统作为解决方案。

本系统共包含数据采集、数据清洗、数据分析、数据可视化、数据共享五大核心功能，同时集成国产核心中间件 **TongWeb** 容器、**TongRDS** 缓存、**apache shenyu** 网关、达梦数据库、**nacos** 服务注册。同时数据采集功能支持大部分国产数据库（人大金仓、达梦、神州通用、南大通用）以及对象存储和 **Hadoop hdfs** 多种数据源；数据分析功能引入基于深度可分离卷积神经网络的轻量级预测模型，对物流数据进行预测，准确率达到 **94.13%**，可满足对港口资源调配的需求。

目录

1 引言	1
1.1 目的	1
1.2 项目背景	1
2 架构设计	1
2.1 系统概述	4
2.2 分层结构	5
2.3 组件及其功能	6
2.4 通信机制	7
2.5 部署架构	8
3 算法设计	9
3.1 算法概述	9
3.1.1 数据采集模块	6
3.1.2 数据治理模块	6
3.1.3 数据分析模块	7
3.1.4 数据共享模块	8
3.2 核心算法	13
3.2.1 数据采集模块	9
3.2.2 数据治理模块	12
3.2.3 数据分析模块	14
3.1.4 数据共享模块	16
3.3 性能考虑	21
4 接口设计	23
4.1 外部接口	23
4.2 内部接口	23
4.3 数据交换格式	26
5 数据存储结构	26
5.1 数据库设计	27
5.2 数据表结构	27
5.3 数据存储方案	32
5.4 数据访问策略	32
6 界面设计	33
7.1 用户界面概述	33
7.2 用户界面组件	43
7.3 用户交互流程	43
7.4 设计原则和准则	44
7 测试策略	45

7.1 测试目标	45
7.2 测试环境	45
7.3 验收标准	45
8 总结	47

1 引言

1.1 项目背景

现代信息化时代，国家大力发展信创(信息技术应用创新)产业链，包括芯片、系统软件、中间件和应用软件，但一直还存在国产中间件与传统架构适配度较低，开发难度大、时间长等问题，同时又伴随着物流行业的快速发展和信息化进程的加速推进，传统的物流信息管理系统已经不能满足日益增长的需求和复杂的业务场景。

结合以上我们提出了一套基于全信创环境下的可视化物流信息数据可视化中台系统。

1.2 文档概述

本文档旨在提供系统设计的全面指南，以支持开发团队在构建新系统或对现有系统进行重构时的决策和实施。通过详细介绍系统的架构、设计考虑、组件和部署方案，本文档旨在帮助开发人员、测试人员、项目经理以及其他相关利益相关者理解系统的整体结构和工作原理。

1、文档目的

提供全面的技术文档支持，帮助开发人员理解软件的设计和实现原理。

提供详细的功能说明和操作指南，帮助管理员和用户正确使用和管理软件。

提供必要的参考资料和技术支持，以便开发人员进行系统定制和二次开发。

2、使用范围

软件开发人员：了解软件的设计和实现原理，进行系统定制和二次开发。

系统管理员：掌握软件的部署和管理方法，保障系统的稳定运行。

最终用户：掌握软件的基本功能和操作方法，提高工作效率。

3、阅读建议

首先阅读引言部分，了解本文档的整体内容和结构。

在阅读过程中，可以根据需要参考附录部分和其他相关章节。

1.3 读者对象

1. 开发人员

- 开发人员可能对系统架构、API 设计、技术栈选择、代码示例等感兴趣，以便了解如何构建系统的各个组件。

2. 二次开发人员

- 我们的系统属于中台系统，仅负责数据采集与分析，只针对专业人员，二次开发人员经过二次开发供大众使用

2. 测试人员

- 测试人员关注系统的功能性和非功能性需求，他们可能对测试策略、测试用例、性能测试结果、安全漏洞等内容感兴趣。

3. 项目经理

- 项目经理可能对项目进度、资源分配、风险管理、成本估算等方面感兴趣，以确保项目按时交付并在预算内完成。

4. 产品经理

- 产品经理关注系统的功能需求、用户体验、市场竞争等方面，他们可能对功能规格、用户界面设计、市场分析报告等内容感兴趣。

5. 运维人员

- 运维人员关注系统的部署、监控、维护和故障排除等方面，他们可能对部署文档、监控指南、故障处理流程等内容感兴趣。

6. 安全团队

- 安全团队关注系统的安全性和合规性，他们可能对安全设计、漏洞报告、安全审计结果等内容感兴趣。

8. 外部合作伙伴

- 外部合作伙伴可能对系统集成、API 文档、数据格式、服务级别协议等内容感兴趣，以便与系统进行集成或交互。

1.4 术语定义

1. 用户

- 定义： 使用系统的个体或实体。用户可以是终端用户、管理员、系统集成商等。

2. API（应用程序接口）

- 定义： 一组定义了软件应用程序如何与其他软件组件进行交互的规范。API 通常包括预定义的函数、方法和数据结构。

3. 负载

- 定义： 指系统承载的工作量，可以是网络流量、请求量、数据量等。负载的大小直接影响系统的性能。

4. 数据库索引

- 定义： 用于加快数据库查询速度的数据结构。索引通常是表中一行或多列的值，数据库引擎使用这些索引来快速定位数据。

5. 水平扩展

- 定义： 增加系统容量或能力的一种方式，通过增加更多的节点或服务器来实现。水平扩展通常涉及将负载分布到多个节点上。

6. 垂直扩展

- 定义： 增加单个节点或服务器的处理能力或资源，以处理更多的负载。垂直扩展通常涉及增加 CPU、内存、存储等硬件资源。

7. RESTful API

- 定义： 一种设计风格，用于构建网络服务，基于 HTTP 协议，通常使用 GET、POST、PUT 和 DELETE 等 HTTP 方法进行通信，支持资源的增删改查操作。

8. CDN（内容分发网络）

- 定义： 一种分布式网络架构，用于将静态内容（如图片、视频、CSS 文件等）缓存到位于全球各地的服务器上，以提高内容传输速度和用户体验。

9. 容器化

- 定义： 使用容器技术（如 Docker）将应用程序及其依赖项打包成一个独立的可移植容器，以实现跨平台部署和运行。

10. 国产信创

- 定义： 国产的信息技术创新，主要指由中国本土企业或个人开发的、具有自主知识产权的信息技术产品或解决方案。

11. 微服务

- 定义： 微服务是一种软件架构风格，其中软件系统被构建为一组小型独立的服务，每个服务都运行在自己的进程中，并通过轻量级的通信机制（通常是 HTTP API）进行通信。每个微服务都专注于一个特定的业务功能，可以独立部署、扩展和维护。

12. 中台系统

- 定义： 指在企业信息化建设中，将前台（客户端、用户界面）与后台（数据存储、业务逻辑）进行解耦，引入一个中间层，用于统一管理业务逻辑、数据和基础设施，从而实现业务的快速开发、灵活部署和高效运维的一种信息技术架构。

这些术语定义将帮助读者更好地理解文档中的内容，并确保在整个文档中使用了一致的术语和概念。

2 架构设计

2.1 系统概述

本系统采用国产操作系统银河麒麟 v10 以及国产核心中间件（包括 Apache Shenyu 网关、Tongweb 容器、TongRDS 缓存、Nacos 注册中心和达梦 8 数据库），并基于微服务分布式架构构建，同时基于团队自主开发的多线程连接池缓存技术。

微服务框架基于 Spring Cloud Alibaba，使用 OpenFeign 远程接口调用实现微服务节点间的通讯，系统内技术涉及 JDBC 连接、mybatis-plus、数据源连接池

多线程连接池缓存技术：利用连接池批量对数据库读写操作，自主研发的一套连接池数量管理与性能优化问题的调优措施。

- **业务背景：**

每天，物流航运领域产生大量的数据，需要一个专门的数据中心来实现对这些数据的统一管理。因此，我们设计了一个数据中台，旨在对不同地区的数据进行统一管理。该数据中台涵盖了数据清洗、分析和预测等关键功能，并将最终的结果通过可视化方式进行展示。

- **用户群体：**

本系统的用户包括以下几个主要群体：

物流公司数据管理员： 负责对所有数据进行集中管理，包括数据采集、数据治理、数据分析和数据预测等工作。

决策人员（高管）： 主要通过可视化图表来了解物流宏观趋势，以便做出正确的决策。这些可视化图表将提供对物流运营状况的直观展示和分析。

普通员工： 可通过 SQL 工作台进行高度定制的数据查询，并且能够导出数据以便进行

进一步的分析和利用。这样的操作能够帮助他们更好地理解数据和做出相关决策。

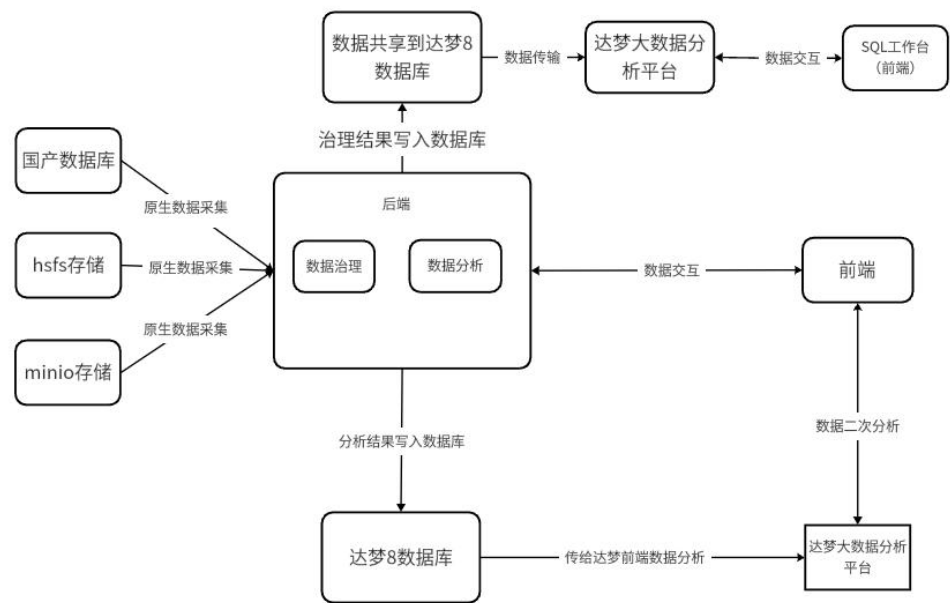
通过满足不同用户群体的需求，本系统旨在提供全面的数据管理和分析解决方案，以支持物流公司的运营和决策工作。

2.2 分层结构

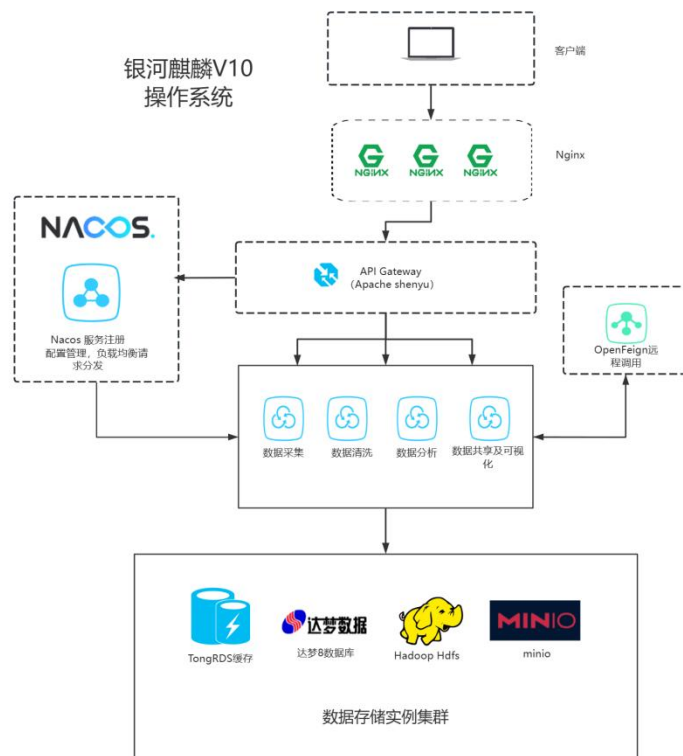
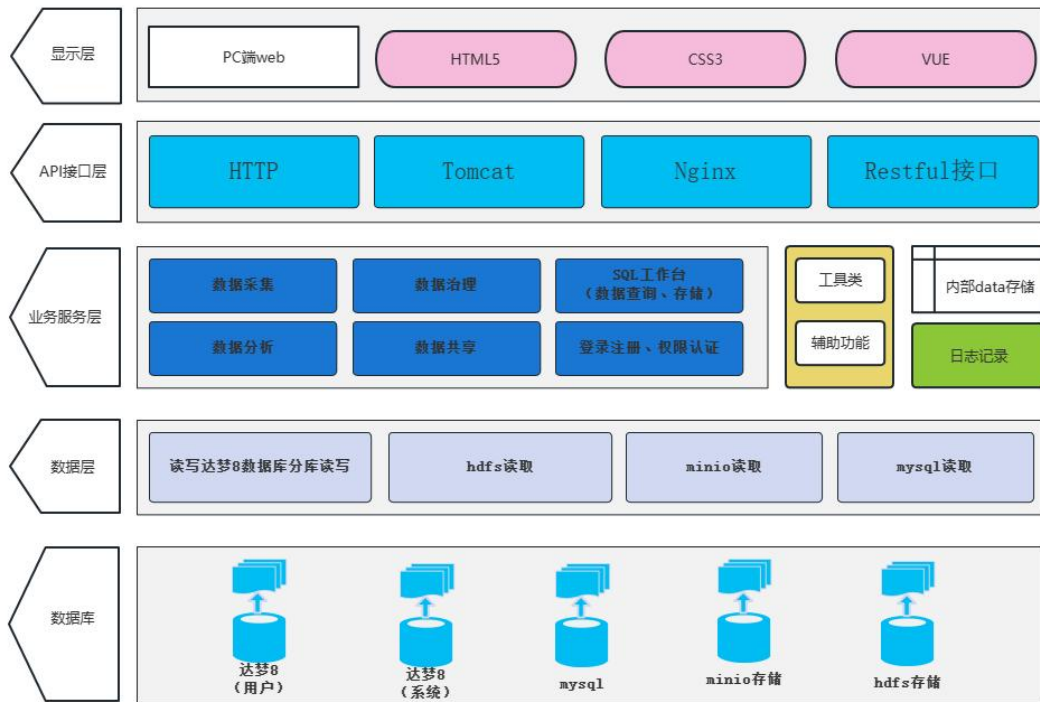
此数据中台主要分为四个部分：

- 数据源层（人大金仓、达梦、神州通用、南大通用、对象存储、hdfs 存储、minio 存储）
- 前端（qianduan 容器）
- 后端（TongWeb 容器）
- 数据库（达梦 8 数据库系统数据库）
- 数据库（达梦 8 数据库用户数据库）

数据中台系统工作流程图



数据中台系统架构



2.3 组件及其功能

- 达梦 8 数据库（系统数据）：国产化数据库，用于存储数据，利用它完成治理合规数据、分析结果持久化的功能。
- 达梦 8 数据库（用户数据）：国产化数据库，数据共享模块服务实现，SQL 工作台查询数据持久化。
- TongWeb 容器：国产化容器，不仅用于部署后端，实现数据源链接、数据采集、数据治理、数据分析、分析结果持久化功能。同时又做到微服务适配方案。项目中，为集成 TongWeb 容器，我们在 springboot 中配置 TongWeb 属性，手动编写 TongWeb 的控制器类来接受处理 http 请求，再调用各种注解和 api 来定义和处理路由、参数、模版等。
- Nacos 注册中心：国产化服务，采用分布式微服务的架构模型，利用 Naocs 实现服务发现与注册，动态配置管理，前端请求来的负载均衡处理。
- TongRDS 缓存：国产化服务，实现数据分布式存储与访问，系统中实现数据查询缓存，以减少数据库访问压力。
- Apache Shenyu 网关：国产化分布式网关与服务框架，用于构建和管理系统各各模块的 API 网关。旨为我们的微服务框架指定统一的入口和流量控制。为做到配置信息可视化，我们将配置信息存储在数据库。为保证微服务的安全性，对于安全认证，我们做了基于密码的身份验证、基于令牌的身份验证。对于访问控制，我们限制特定用户、IP 地址、请求路径等访问某些服务或接口，从而保护系统的安全性和隐私性。在数据传输过程中，我们对数据加密解密处理。
- 银河麒麟操作系统：国产化操作系统，提供了项目运行所需的基础环境和资源管理。
- PyTorch 框架：用于构建和训练各种类型的深度学习模型，在项目中负责构建卷积神经网络

2.4 通信机制

系统采用 HTTP 协议进行通信，通过 RESTful API 接口实现服务之间的交互。服务注册与发现采用 Nacos，实现了服务自动发现和动态路由。数据传输过程中，客户端向服务器发起 HTTP 请求，服务器根据请求进行处理，并返回相应的 HTTP 响应。为保证通信安全，采用 HTTPS 协议进行数据加密和认证。系统在通信过程中处理各种可能出现的错误情况，并通过性能优化策略提高通信效率，确保通信的可靠性和稳定性。

2.5 部署架构

该项目的部署架构基于 Kubernetes (k8s) 的容器化技术，共涉及五个应用：达梦 8 数据库（系统数据）、达梦 8 数据库（用户数据）、Nginx 容器、TongWeb 容器以及微服务部署(分布式集成服务)。

- 达梦 8 数据库容器：

达梦 8 数据库容器是用来存储和管理系统的数据，提供数据的持久化存储。通过 Kubernetes 的 Pod 和 Deployment 资源，创建达梦 8 数据库容器并配置相应的存储卷。

达梦 8 数据库容器与 TongWeb 容器进行通信，以提供数据存储和读取的支持。

- Nginx 容器：

Nginx 容器用于部署系统的前端，提供用户界面和静态文件的访问。通过 Kubernetes 的 Pod 和 Deployment 资源，创建 Nginx 容器，并配置相关的网络和端口映射。

Nginx 容器与用户浏览器之间通过 HTTP 协议进行通信，提供用户界面的访问和交互。

- TongWeb 容器：

TongWeb 容器是后端模块的部署单元，用于实现业务功能，包括数据采集、治理、存储、分析和展示等。

通过 Kubernetes 的 Pod 和 Deployment 资源，创建 TongWeb 容器，并配置与达梦

8 数据库容器的通信连接。

TongWeb 容器与达梦 8 数据库容器进行数据读写操作，同时与 Nginx 容器通过 HTTP 协议进行通信，提供 API 接口和数据交互。

- 微服务框架：

利用微服务框架部署项目，实现将业务功能分模块部署，实现了各模块的独立性与高流量问题。

在 Kubernetes 上部署微服务，将每个微服务容器化为 Docker 镜像，然后编写 Kubernetes 的 YAML 配置文件，描述微服务的部署和服务，包括 Deployment、Service 等对象的定义，接着使用 kubectl 命令应用这些配置文件到 Kubernetes 集群中，最后通过监控工具和持续集成部署工具确保微服务的稳定运行和持续交付。

部署架构的优势在于使用 Kubernetes 进行容器编排和管理，实现了系统的弹性伸缩和高可用性，同时提供了前后端分离的部署模式。达梦 8 数据库容器提供了稳定的数据存储和读写能力，Nginx 容器提供了高效的用户界面访问，TongWeb 容器实现了系统的业务功能。微服务技术实现了各功能模块的独立性。通过这样的部署架构，能够确保系统的可靠性、性能和可扩展性。

3 算法设计

3.1 算法概述

3.1.1 数据采集

遍历各种数据源集合（国产数据库、HDFS 存储、Minio 存储），利用对应的数据采集方法从每个数据源中采集原始数据，并将采集到的数据存储到本地目录 "/opt"。

对于 MySQL 数据库，使用适当的连接库和 SQL 查询语句，通过遍历数据库表获取数据，并将结果存储到本地文件。

对于 HDFS 存储，使用 Hadoop 文件系统接口，通过遍历 HDFS 上的文件和目录，获取数据内容，并将数据保存到本地目录。

对于 Minio 存储，使用 Minio 客户端 API，通过遍历存储桶和对象，下载对象的内容，并将数据存储到本地目录。

在遍历过程中，根据每种数据源的特性和访问方式，选择合适的迭代方式（如循环或递归）来遍历数据源集合。同时涉及多种格式数据源，因此需要对每个数据源进行处理分析，有考虑到数据采集的效率问题，因此我采取了连接池缓存技术与多线程模式采集数据，实现了毫秒内完成数据采集工作。对于每个数据源，使用对应的采集方法，通过连接、查询或下载等操作获取原始数据。最后，将采集到的数据按照要求存储到本地目录"/opt"，可以使用合适的文件格式（如文本文件、CSV 文件等）来保存数据。

3.1.2 数据治理

根据用户选择的治理规则（数据去重、异常值检测），对原始数据进行二次遍历。数据治理条件的优先级按照如下顺序进行处理：身份证号有效性 > 物流信息是否在客户信息中 > 异常值检测 > 数据去重。

通过按照优先级顺序进行数据治理条件的处理，确保数据治理过程的准确性和完整性。根据不同的条件结果，将数据标记为异常数据或进行相应的处理。这样的算法描述清晰地说明了数据治理条件的优先级和处理顺序，使读者更容易理解和实施。

- 身份证号有效性：

1. 首先，检查身份证号的长度是否为 18 位。
2. 将前 17 位数字分别乘以对应的权重（依次为 7、9、10、5、8、4、2、1、6、3、7、9、10、5、8、4、2）。
3. 将乘积结果相加得到总和。
4. 将总和除以 11 取余数，得到余数值。
5. 根据余数值在对应的映射表中查找校验码。
6. 将身份证号的最后一位与校验码进行比较，如果相符则身份证号正确，否则错误。

- 物流信息是否在客户信息中：

优先遍历客户信息表，将客户信息加载到 Map 中，身份证号为 Key，客户信息对象为 value。遍历物流信息表单时，判断物流信息货主代码字段内容是否存在于客户信息 Map 的 key 中。

- 异常值检测：

电话号码的异常检测（是否为 11 位），集装箱尺寸的异常检测（是否为 20 或 40）

- 数据去重：

遍历数据时与先前遍历过的数据进行比对，客户信息比对身份证号，物流信息、装货表和卸货表比对提单号。如有重复，将次数据存放至不合规数据中。

3.1.3 数据分析与预测

- 港口的吞吐量分析：

遍历装货表、卸货表根据提单号查询物流信息表货运量，将港口名称、货运量传给统计函数进行吞吐量的统计。

- 港口不同类型货物吞吐趋势：

趋势用最近年份中最近月份的环比进行表示，将 2022 年 11、12 月港口货物货运量传给统计函数进行吞吐量统计，存到 Map 中，最后遍历 Map 计算各港口每种货物 2022 年 12 月环比，并将结果持久化到达梦 8 数据库中。

- 港口货物吞吐同比环比：

将年份、港口、货物、月份、货运量传给统计函数进行吞吐量统计，存到 Map 中，最后遍历 Map 计算每年各港口各月同比环比，并将结果持久化到达梦 8 数据库中。

- 不同货物吞吐占比：

将货物、货运量传给统计函数进行吞吐量统计，存到 Map 中，遍历 Map 计算总吞吐量，再次遍历 Map 算出所占百分比，并将结果持久化到达梦 8 数据库中。

- 不同货物流向分析：

将货物、港口、货运量传给统计函数进行吞吐量统计，存到 Map 中，遍历 Map 计算每种货物流向不同港口比重，并将结果持久化到达梦 8 数据库中。

- 不同类型货物堆场流转周期分析：

将货物、堆场流转时间（long 类型，不足一天按一天计算）传给统计函数进行吞吐量统计，存到 Map 中，遍历 Map 计算每种货物流向不同港口比重，并将结果持久化到达梦 8 数据库中。

- 数据预测：

在数据分析功能中，我们引入了基于深度可分离卷积神经网络的轻量级预测模型，对物流数据进行预测分析，经测试准确率高达 94.13%，可满足对港口资源调配的需求。

一方面为保证准确率，我们引入前六个月的港口货物流量数据分析后续流量走势情况，并且投放到前端图标数据，供用户直观感受。另一方面，用户可以自定义选择性的导入数据，可以是某个时段，也可以是某批量数据分析，最后我们给出未来可能的流量走势图。

3.1.4 数据共享

- 合规数据展示

我们给用户暴露四个接口，分别是客户信息、物流信息、装货表、卸货表，同时我们将接口参数做成动态可配，也就是手动输入他们自己的参数信息，这样可供一些用户进行二次开发，同时我们又新增 SQL 工作台，供用户查询数据库。

返回信息形式共两种：表格化展示、JSON 字符串。

每次访问接口后端查询数据库，将查询数据返还给前端。

- SQL 工作台

前端将用户的 sql 代码传到后端，后端使用 jdbc 连接数据库，数据库注册特定权限用户，将查询信息转换成 List<Map<String,Object>>结构回传前端，若 sql 语法有误，返回则为报错信息。

3.2 核心算法

3.2.1 数据采集模块

数据采集总接口

```
/**
 * 数据采集接口
 */
1 个用法
public void dataAcquisition() throws Exception {
    mysqlData();
    if (minioDataSource.size() != 0){
        minioDataSource.forEach(m -> {
            minioData(m.getEndpoint(), m.getAccessKey(), m.getSecretKey(), m.getBucketName());
        });
    }

    if (hdfsDataSource.size() != 0){
        hdfsDataSource.forEach(h -> {
            try {
                hdfsData(h.getUrl(), h.getTargetPath());
            } catch (Exception e) {
                logger.error(e.getMessage() + e);
            }
        });
    }
}
```

Mysql 数据采集

```
public void mysqlData(){
    //查询mysql中所有表单名称
    List<String> tables = mysqlMapper.selectTables();
    int i = 1;
    for (String tableName : tables) {
        //查询该表单所有字段
        List<String> columns = mysqlMapper.selectColumns(tableName);
        //查询客户信息
        List<CustomerInformation> list = mysqlMapper.selectCustomerInformation(columns, tableName);
        customerMap.put(i, list);
        list.forEach(System.out::println);
        i++;
    }
}
```

Minio 数据采集

1 个用法

```
@Value("/opt")
private String downloadPath;
```

2 个用法

```
public AmazonS3 getClient(String endpoint, String accessKey, String secretKey){
    // 创建Minio客户端
    AWSCredentials awsCredentials = new BasicAWSCredentials(accessKey, secretKey);
    AwsClientBuilder.EndpointConfiguration endpointConfiguration =
        new AwsClientBuilder.EndpointConfiguration(endpoint, signingRegion: null);
    return AmazonS3ClientBuilder.standard()
        .withCredentials(new AWSStaticCredentialsProvider(awsCredentials))
        .withEndpointConfiguration(endpointConfiguration).build();
}
```

```
public void downloadFiles(String endpoint, String accessKey, String secretKey, String bucketName) {
    AmazonS3 client = getClient(endpoint, accessKey, secretKey);

    // 列出文件
    List<String> fileName = new ArrayList<>();
    ObjectListing objectListing = client.listObjects(bucketName);

    List<S3ObjectSummary> objectSummaries = objectListing.getObjectSummaries();

    objectSummaries.forEach(s3ObjectSummary -> fileName.add(s3ObjectSummary.getKey()));
    while (objectListing.isTruncated()) {
        objectListing = client.listNextBatchOfObjects(objectListing);
        objectSummaries = objectListing.getObjectSummaries();
        objectSummaries.forEach(s3ObjectSummary -> fileName.add(s3ObjectSummary.getKey()));
    }
}
```

```
//文件下载
File dir = new File(downloadPath);
//创建文件夹
dir.mkdirs();
fileName.forEach(key -> {
    if (!key.equals("mc")) {

        String fileNameEnglish = "";
        if (key.contains("集装箱动态")) {
            fileNameEnglish = "ContainerDynamics" + key.substring(beginIndex: 5);
        } else if (key.contains("物流公司")) {
            fileNameEnglish = "LogisticsCompanies" + key.substring(beginIndex: key.length() - 9);
        }

        File file = new File(pathname: dir.getAbsolutePath() + File.separator + fileNameEnglish);

        // dir/sss/aaa/ww/obj.xx 防止此类文件下载时出错
        file.getParentFile().mkdirs();
    }
});
```

```
try (S3Object object = client.getObject(bucketName, key);
     OutputStream ops = new BufferedOutputStream(Files.newOutputStream(file.toPath())) {
    S3ObjectInputStream objectContent = object.getObjectContent();
    //创建缓冲区 (1M = 1024000)
    byte[] buff = new byte[1 << 20];
    int length;
    while ((length = objectContent.read(buff)) != -1) {
        ops.write(buff, off: 0, length);
    }
    ops.flush();
    objectContent.close();
    System.out.println(key + "下载成功");
} catch (IOException e) {
    e.printStackTrace();
}
});
```

Hdfs 数据采集

```
public void downloadHdfsFiles(String url, String path) throws Exception {
    //建立客户端
    FileSystem fs = getFileSystem(url);
    // 读取文件列表的目标路径
    Path targetPath = new Path(path);
    // 递归找到所有文件
    RemoteIterator<LocatedFileStatus> filesList = fs.listFiles(targetPath, recursive: true);
    while (filesList.hasNext()) {
        LocatedFileStatus next = filesList.next();

        //获取文件名称
        String fileName = next.getPath().getName();
        String fileNameEnglish = "";
        if (fileName.contains("物流信息")) {
            fileNameEnglish = "LogisticsInformation" + fileName.substring(beginIndex: 4);
        } else if (fileName.contains("卸货表")) {
            fileNameEnglish = "UnloadingTable" + fileName.substring(beginIndex: 3);
        } else if (fileName.contains("装货表")) {
            fileNameEnglish = "LoadingTable" + fileName.substring(beginIndex: 3);
        }
    }
}
```

```

//拼接本地文件地址
File localFile = new File(pathname: downloadPath + File.separator + fileNameEnglish);

System.out.println("localFile:" + localFile.getAbsolutePath());

try (FSDataInputStream open = fs.open(next.getPath());
    OutputStream ops = new BufferedOutputStream(Files.newOutputStream(localFile.toPath())))
{
    //创建缓冲区 (1M = 1024000)
    byte[] buff = new byte[1 << 20];
    int length;
    while ((length = open.read(buff)) != -1) {
        ops.write(buff, 0, length);
    }
    ops.flush();
    System.out.println(localFile.getName() + "--downloadSucceed");
} catch (IOException e) {
    System.out.println("文件下载失败: " + e.getMessage());
}

}
fs.close();
}

```

3.2.2 数据治理模块

客户信息治理

```

private void customerInformation(boolean repeat, boolean examine){
    //遍历客户信息，根据筛选条件，添加到数据
    for (int i : customerMap.keySet()) {
        for (CustomerInformation c : customerMap.get(i)) {

            //判断身份证号是否合规
            if(!IdCardUtil.isIdCardNo(c.getId())){
                c.setRemark("身份证号不合规");
                customerList_false.add(c);
                continue;
            }
            //异常值检测
            if (examine){
                if (c.getPhoneNumber().length() != 11){
                    c.setRemark("电话号异常");
                    customerList_false.add(c);
                    continue;
                }
            }
            //去重
            if (repeat & customer_true.containsKey(c.getId())){
                c.setRemark("数据重复");
                customerList_false.add(c);
                continue;
            }

            customerList_true.add(c);
            customer_true.put(c.getId(),c);
        }
    }
}

```

```

//判断身份证号是否有误
if (!IdCardUtil.isIdCardNo(log.getOwnerId())){
    log.setRemark("身份证号不合规");
    logisticsList_false.add(log);
    continue;
}

//判断物流信息中客户是否存在
if (!(customer_true.containsKey(log.getOwnerId()))){
    log.setRemark("客户不在客户信息中");
    logisticsList_false.add(log);
    continue;
}

//去重
if(repeat & logistics_true.containsKey(log.getId())){
    log.setRemark("数据重复");
    logisticsList_false.add(log);
    continue;
}

```

装货表治理

```

//异常值检测
if (examine & (loadingTable.getContainerSize() != 20 && loadingTable.getContainerSize() != 40)){
    loadingTable.setRemark("集装箱尺寸异常");
    loadingList_false.add(loadingTable);
    continue;
}

//去重
if (repeat & loading_true.contains(loadingTable.getOrderId())){
    loadingTable.setRemark("数据重复");
    loadingList_false.add(loadingTable);
    continue;
}

```

卸货表治理


```

//异常值检测
if (examine & (unloadingTable.getContainerSize() != 20 && unloadingTable.getContainerSize() != 40)){
    unloadingTable.setRemark("集装箱尺寸异常");
    unloadingList_false.add(unloadingTable);
    continue;
}

//去重
if (repeat & unloading_true.contains(unloadingTable.getOrderId())){
    unloadingTable.setRemark("数据重复");
    unloadingList_false.add(unloadingTable);
    continue;
}

```

3.2.3 数据分析模块

数据分析前清除统计结果和数据库中持久化的分析结果

```

public void deleteData(){

    //清除数据库
    dmMapper.deleteTable( tableName: "GOODS_PORT_PUT");
    dmMapper.deleteTable( tableName: "GOODS_THROUGHPUT_PERCENTAGE");
    dmMapper.deleteTable( tableName: "GOODS_TIME");
    dmMapper.deleteTable( tableName: "PORT_GOODS_TREND");
    dmMapper.deleteTable( tableName: "PORT_THROUGHPUT");
    dmMapper.deleteTable( tableName: "YEAR_PORT_MONTH_YOY_QOQ");

    //统计数据
    port_throughput.clear();
    portGoodsMonth_throughput.clear();
    goods_throughput.clear();
    port_goods_throughput.clear();
    year_port_goods_throughput.clear();
    goods_port_put.clear();
    yearPortMonth_throughput.clear();
    goods_time.clear();
}

```

数据统计

```
//计算货物堆场流转时间
goodsTimeCalculation(goods, loadingTable.getWorkBeginTime().getTime(),
    loadingTable.getDepartureTime().getTime());

//港口吞吐量计算
portThroughputCalculation(port, weight);

//各港口各货物吞吐量计算
portGoodsThroughputCalculation(port, goods, weight);

//每年各月吞吐量计算
yearPortMonthThroughputCalculation(year, port, month, weight);

//每年各港口各货物吞吐量计算
yearPortGoodsThroughputCalculation(year, port, goods, weight);

//货物吞吐统计
goodsThroughputCalculation(goods, weight);

//2022年港口货物每月吞吐量
if (year == 2022 & (month == 12 || month == 11))
    portGoodsMonthThroughputCalculation(port, goods, month, weight);
```

结果计算

```
//计算货物堆场流转时间
goodsTimeCalculation(goods, unloadingTable.getArriveTime().getTime(),
    unloadingTable.getWorkEndTime().getTime());

//港口吞吐量计算
portThroughputCalculation(port, weight);

//各港口各货物吞吐量计算
portGoodsThroughputCalculation(port, goods, weight);

//每年各月吞吐量计算
yearPortMonthThroughputCalculation(year, port, month, weight);

//每年各港口各货物吞吐量计算
yearPortGoodsThroughputCalculation(year, port, goods, weight);

//货物吞吐统计
goodsThroughputCalculation(goods, weight);

//各货物流向分析
goodsPortThroughputCalculation(goods, port, weight);

//2022年港口货物11月12月吞吐量
if (year == 2022 & (month == 12 || month == 11))
    portGoodsMonthThroughputCalculation(port, goods, month, weight);
```


3.2.4 数据共享模块

SQL 工作台接口：

```
/**
 * sql工作台 (jdbc)
 * 查询失败返回sql错误信息
 */
1个用法
@PostMapping("/selectTable")
public String selectTable(String username,String sql){
    if (!(sql.contains("SELECT") || sql.contains("select"))){
        return JSON.toJSONString(new Result( code: 404, message: "查询失败! ", object: "权限不足! "));
    }
    try {
        List<Map<String, Object>> list = openSqlService.selectTable(username, sql);
        //将查询的结果存储到Map中
        return JSON.toJSONString(new Result( code: 0, message: "查询成功! ", list));
    }catch (Exception e){
        logger.error(e.getMessage());
        return JSON.toJSONString(new Result( code: 404, message: "查询失败! ", e.getMessage()));
    }
}
```

查询成功返回查询信息，查询失败返回报错信息（强制只能执行 select 语句）

可视化报表配置：

```
/**
 * 持久化到数据库
 */
0个用法
@PostMapping("/saveResult")
public String saveResult(String username){

    //建立数据库表
    openSqlService.createTable(username);
    //持久化到数据库
    openSqlService.insertData(username);

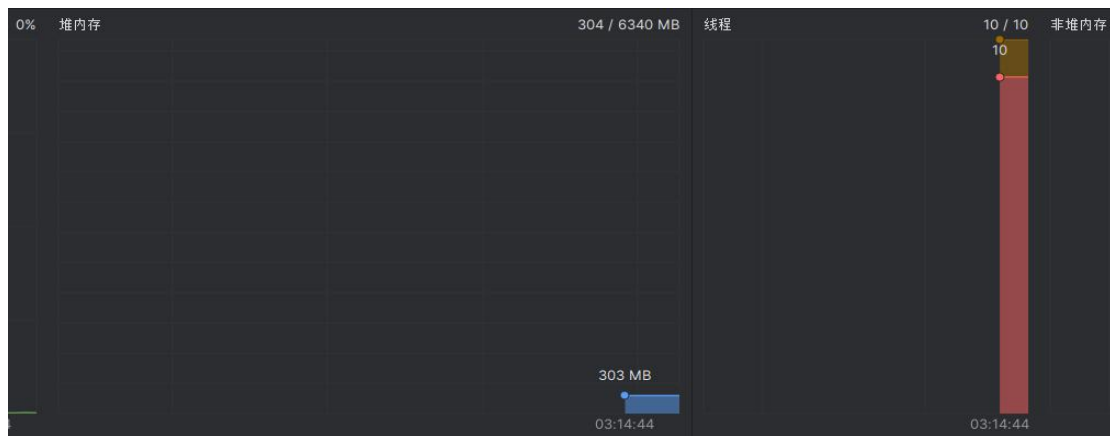
    return JSON.toJSONString(new Result( code: 0, message: "保存成功! ", object: null));
}
```

将用户查询信息持久化到达梦数据库，跳转链接到达梦大数据分析平台。

3.3 性能考虑

项目中，为兼顾效率与性能的问题，我们运用线程池缓存，选择批量的存入数据源，完成数据库读写操作，具体是项目中创建线程池（始发 3 个，最大 10 个数量级别），自动化的增加与销毁对象，形成一种线程池的管理模式

运行时堆内存在 300mb 左右，容器分配性能为 cpu1.5 核心、内存 4gb，分配留有余地。



4 接口设计

4.1 外部接口

数据共享接口

```

@RequestMapping(🌐"/displayData")
public class OpenDataController implements Data {
|

/**
 * 错误数据
 */
0 个用法
@PostMapping(🌐"/errorData")
public String errorData(){...}

/**
 * 客户信息分页
 */
0 个用法
@PostMapping(🌐"/customer")
public String customer(int page, int number){...}

/**
 * 物流信息分页
 */
0 个用法
@PostMapping(🌐"/logistics")
public String logistics(int page, int number){...}

/**
 * 卸货表分页
 */
0 个用法
@PostMapping(🌐"/unloading")
public String unloading(int page, int number){...}

```

```

/**
 * sql工作台 (jdbc)
 * 查询失败返回sql错误信息
 */
1 个用法
@PostMapping(🔗"/selectTable")
public String selectTable(String username,String sql){
    if (!(sql.contains("SELECT") || sql.contains("select"))){
        return JSON.toJSONString(new Result( code: 404, message: "查询失败! ", object: "权限不足! "));
    }
    try {
        List<Map<String, Object>> list = openSqlService.selectTable(username, sql);
        //将查询的结果存储到Map中
        return JSON.toJSONString(new Result( code: 0, message: "查询成功! ", list));
    }catch (Exception e){
        logger.error(e.getMessage());
        return JSON.toJSONString(new Result( code: 404, message: "查询失败! ", e.getMessage()));
    }
}

/**
 * 持久化到数据库
 */
0 个用法
@PostMapping(🔗"/saveResult")
public String saveResult(String username){

    //建立数据库表
    openSqlService.createTable(username);
    //持久化到数据库
    openSqlService.insertData(username);

    return JSON.toJSONString(new Result( code: 0, message: "保存成功! ", object: null));
}

```

数据源接口

```

/**
 * 数据源链接测试
 */
0 个用法
@PostMapping(🔗"/dataSourceTest")
@ResponseBody
public String dataSourceConnectionTest(String type, String url, String username,
                                       String password,String bucketName, String path)
    throws SQLException, ClassNotFoundException {...}

/**
 * 保存数据源 (数据源信息存储到内存)
 */
0 个用法
@PostMapping(🔗"/dataSourceSave")
@ResponseBody
public String dataSourceSave(String type, String url, String username, String password, String bucketName, String path){...}

```

4.2 内部接口

数据分析接口

```
/**
 * 数据采集接口
 */
0 个用法
@PostMapping(Ⓜ"/dataAcquisition")
public String dataAcquisition() throws InterruptedException {...}

/**
 * 数据清洗（过滤）接口
 */
0 个用法
@PostMapping(Ⓜ"/dataClean")
public String dataClean(boolean repeat, boolean examine) throws IOException, ParseException {...}

/**
 * 客户信息修改不合规数据
 */
0 个用法
@PostMapping(Ⓜ"/updateCustomerData")
public String updateCustomerData(boolean repeat, boolean examine,
                                String name, String id, String phoneNumber, String address, String remark){...}

/**
 * 物流信息修改不合规数据
 */
0 个用法
@PostMapping(Ⓜ"/updateLogistics")
public String updateLogistics(
    boolean repeat, boolean examine, String id, String owner, String ownerId, String companyId, String containerId,
    String goodsName, int weight, String remark)
{...}
```

```
/**
 * 装货表
 */
0 个用法
@PostMapping(Ⓜ"/updateLoadingTable")
public String updateLoadingTable(
    boolean repeat, boolean examine, String shipCompanies, String shipName, String workBeginTime, String workEndTime,
    String departureTime, String arriveTime, String port, String orderId, String containerId, String containerSize,
    String departurePlace, String destinationPlace, String remark
) throws ParseException {...}

/**
 * 卸货表
 */
0 个用法
@PostMapping(Ⓜ"/updateUnloadingTable")
public String updateUnloadingTable(
    boolean repeat, boolean examine, String shipCompanies, String shipName, String workBeginTime, String workEndTime,
    String departureTime, String arriveTime, String port, String orderId, String containerId, String containerSize,
    String departurePlace, String destinationPlace, String remark
) throws ParseException {...}
```

4.3 数据交换格式

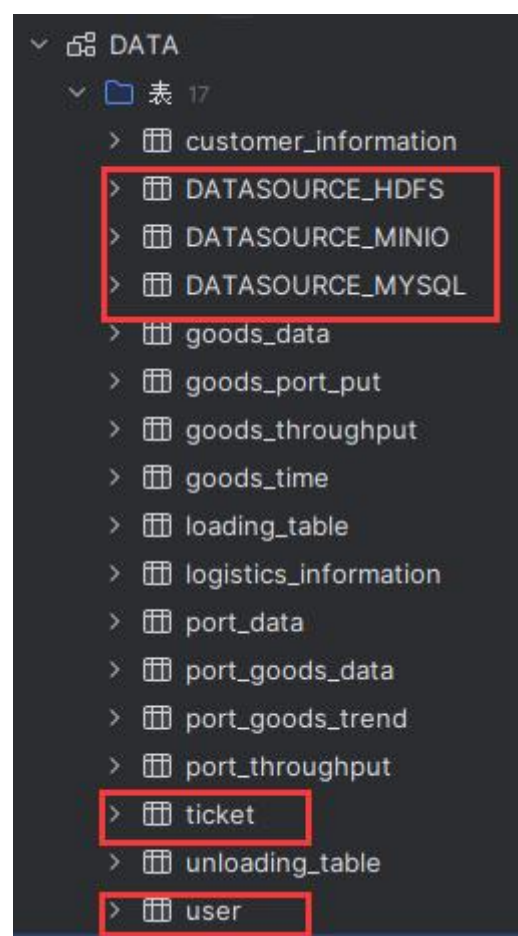
整体架构遵循前后端分离的原则，通过 RESTful API 实现前后端的交互和数据传输。前端通过 Vue 框架发起请求，后端接收请求并处理相应的业务逻辑，最后将结果以 JSON 形式返回给前端进行展示。

5 数据存储结构

5.1 数据库设计

数据库共 17 张表，红框标注出的是系统数据，其均为数据共享模块的共享数据资源。

系统数据中 user 为用户信息表，ticket 为登录凭证表，DATASOURCE_HDFS、DATASOURCE_MINIO、DATASOURCE_MYSQL 分别为数据源信息表（hdfs 存储、minio 存储、mysql 数据库）



5.2 数据表结构

```
1 CREATE DATABASE DATA;
2
3 USE DATA;
4
5 create table DATA.GOODS_PORT_PUT
6 (
7     GOODS      VARCHAR(10),
8     PORT       VARCHAR(10),
9     PUT        INTEGER,
10    PERCENTAGE  FLOAT(53)
11 );
12
13 create table DATA.GOODS_THROUGHPUT_PERCENTAGE
14 (
15     GOODS      VARCHAR(10),
16     PERCENTAGE  FLOAT(53)
17 );
18
19 create table DATA.GOODS_TIME
20 (
21     GOODS VARCHAR(10),
22     TIME  DOUBLE PRECISION(53)
23 );
24
25 create table DATA.PORT_GOODS_TREND
26 (
27     PORT VARCHAR(10),
28     GOODS VARCHAR(10),
29     TREND FLOAT(53)
30 );
31
32 create table DATA.PORT_THROUGHPUT
33 (
34     PORT      VARCHAR(10),
35     THROUGHPUT INT(10)
36 );
37
38 create table DATA.YEAR_PORT_MONTH_YOY_QOQ
39 (
40     YEAR  INTEGER,
41     PORT  VARCHAR(100),
42     MONTH INTEGER,
43     YOY   FLOAT(53),
44     QOQ   FLOAT(53)
45 );
```



```
-- 用户信息
create table DATA."customer_information"
(
    "name"          VARCHAR(8188),
    "id"            VARCHAR(8188),
    "phone_number"  VARCHAR(8188),
    "address"       VARCHAR(8188),
    "remark"        VARCHAR(8188)
);

-- 物流信息
create table DATA."logistics_information"
(
    "id"            VARCHAR(8188),
    "owner"         VARCHAR(8188),
    "owner_id"      VARCHAR(8188),
    "company_id"    VARCHAR(8188),
    "container_id"  VARCHAR(8188),
    "goods_name"    VARCHAR(8188),
    "weight"        int,
    "remark"        VARCHAR(8188),
);
```

```

-- 卸货表
create table DATA."unloading_table"
(
    "ship_companies"    VARCHAR(8188),
    "ship_name"         VARCHAR(8188),
    "work_begin_time"   DATE,
    "work_end_time"     DATE,
    "departure_time"    DATE,
    "arrive_time"       DATE,
    "port"              VARCHAR(8188),
    "order_id"          VARCHAR(8188),
    "container_id"      VARCHAR(8188),
    "container_size"    INT,
    "departure_place"   VARCHAR(8188),
    "destination_place" VARCHAR(8188),
    "remark"            VARCHAR(8188),
);

-- 分析结果表
-- 港口信息表
create table DATA."port_data"
(
    "port"              VARCHAR(255),
    "year"              int,
    "month"             int,
    "throughput"        int,
    "yoy"               float,
    "qoq"               float
);

```

```
-- 货物信息表
create table DATA."goods_data"
(
    "goods"          VARCHAR(255),
    "year"            int,
    "month"           int,
    "throughput"      int
);

create table DATA."port_goods_data"
(
    "port"            VARCHAR(255),
    "goods"            VARCHAR(255),
    "year"            int,
    "month"           int,
    "throughput"      int
);

create table DATA."user"
(
    id                integer PRIMARY KEY AUTO_INCREMENT,
    username          varchar,
    password          varchar,
    real_name         varchar,
    id_card           varchar,
    phone            varchar,
    active            integer
);
```

```
create table DATA."ticket"
(
    id                integer PRIMARY KEY AUTO_INCREMENT,
    user_id          int,
    ticket           varchar,
    power            int
);
```

5.3 数据存储方案

根据项目需求，采集的原始数据和中间统计结果等数据存储在内存在，而分析完成的结果存储在达梦 8 数据库中。因此，可以采用以下数据存储方案：

1. 原始数据和中间统计结果：

- 存储位置：将这些数据存储在内存在的数据结构（如数组、哈希表等）或缓存中，以提高访问速度和性能。
- 存储格式：根据数据类型和结构的特点选择合适的内存存储方式，如对象、键值对等。

2. 分析结果：

- 存储位置：将分析完成的结果存储在达梦 8 数据库中，以确保数据的持久性和可靠性。
- 存储结构：根据分析结果的特点和查询需求，设计合适的数据库表结构，包括字段、索引等。
- 数据库管理：使用达梦 8 数据库提供的管理工具，对数据进行管理、备份和恢复，确保数据的完整性和安全性。

5.4 数据访问策略

1. 原始数据和中间统计结果：

- 内存访问：通过程序的内存操作方式，直接从内存中访问和处理这些数据。
- 缓存访问：将数据缓存在内存中的缓存系统中，如 Redis 等，以提高读取速度和响应性能。

2. 分析结果：

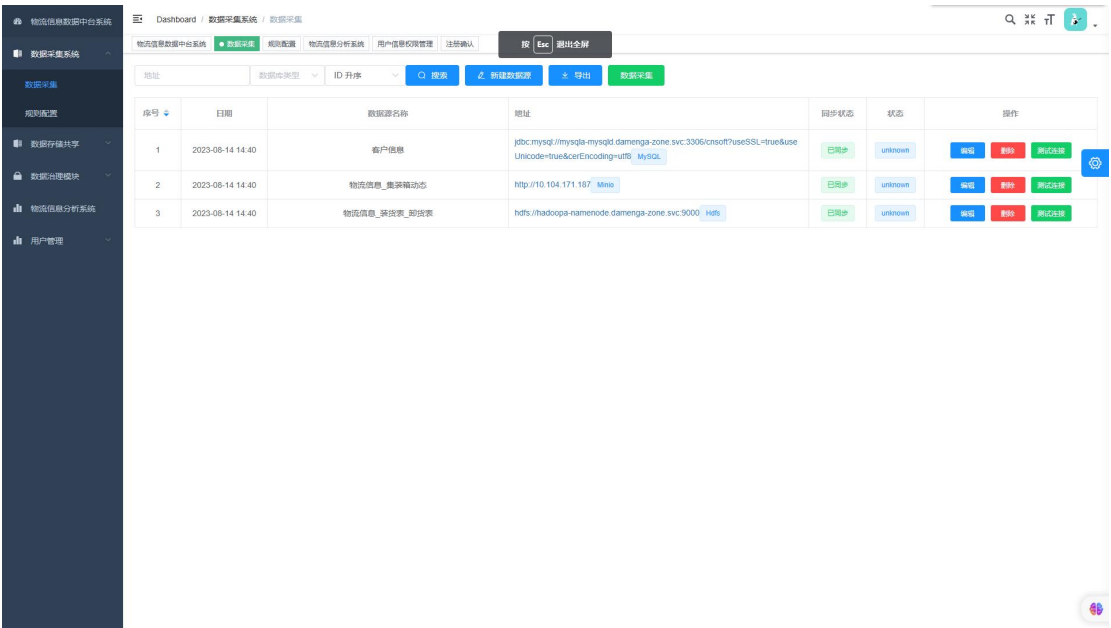
- SQL 查询：使用 SQL 语句从达梦 8 数据库中查询和获取分析结果。
- 数据库连接：通过数据库连接池或数据库客户端连接到达梦 8 数据库，并执行相应的查询操作。

根据具体的数据访问需求和性能要求，合理选择数据访问策略，以实现数据的高效访问和处理。同时，根据项目的实际情况，可以考虑使用缓存技术、数据索引优化等手段来提升数据访问的性能和效率。

6 界面设计

6.1 用户界面概述

数据采集界面：包含添加数据源、数据源连接测试以及开始数据采集功能。



数据治理界面：包括数据治理规则选择页面和错误数据展示页面。



错误数据展示页面可对错误数据进行编辑，提交修改后会对修改信息进行反馈（修改成功或失败原因），以及下一步数据分析操作

物流信息数据中台系统

数据平台系统

数据引擎

规则配置

数据存储服务

数据治理模块

异常数据处理

物流信息分析系统

用户管理

Dashboard / 数据治理模块 / 异常数据处理

物流信息数据中台系统 | 数据引擎 | 规则配置 | 物流信息分析系统 | 用户信息权限管理 | 注册确认 | 异常数据处理

异常数据处理

物流信息

客户信息

缺失表

异常表

提单号	货主名称	货主代码	物流公司	集装箱箱号	货物名称	货重(吨)	原因	操作
TEST1234567	中国达梦	420111200001010016	达梦测试单可以删除	TEST0001	数据库	1000	客户不在客户信息中	修复

Total 1

导出

开始分析

Edit

×

* 提单号

TEST1234567

* 货主名称

中国达梦

* 货主代码

420111200001010016

* 物流公司

达梦测试单可以删除

* 集装箱箱号

TEST0001

* 货物名称

数据库

* 货重(吨)

1000

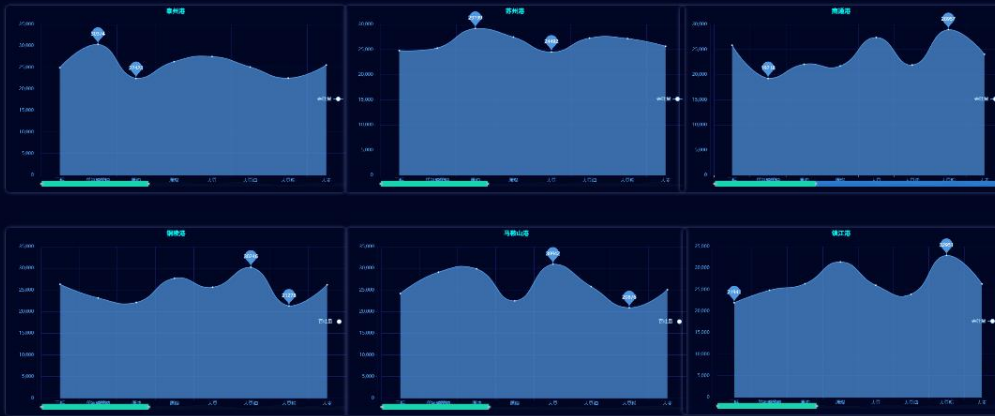
* 备注

客户不在客户信息中

取消

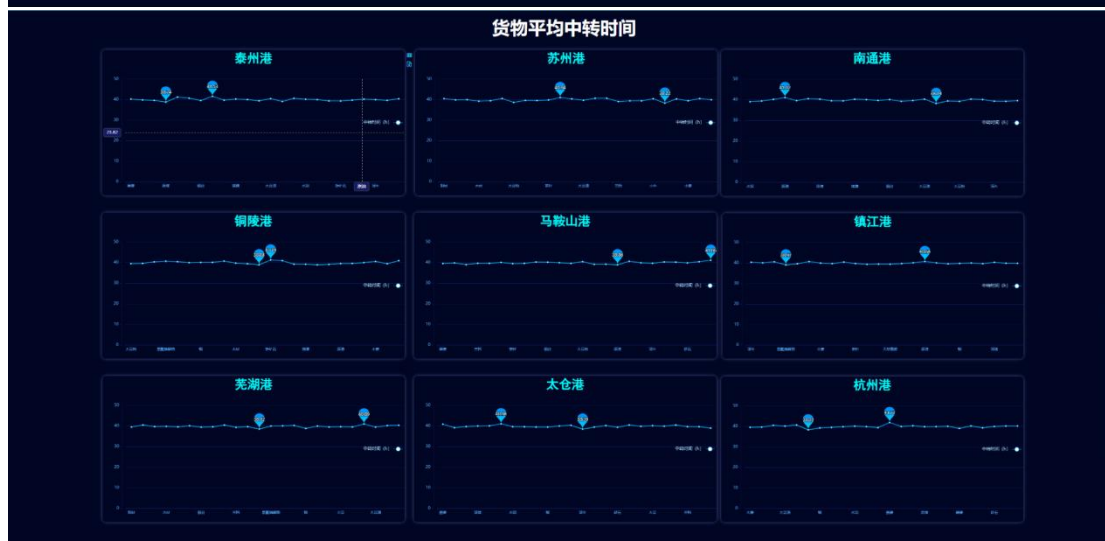
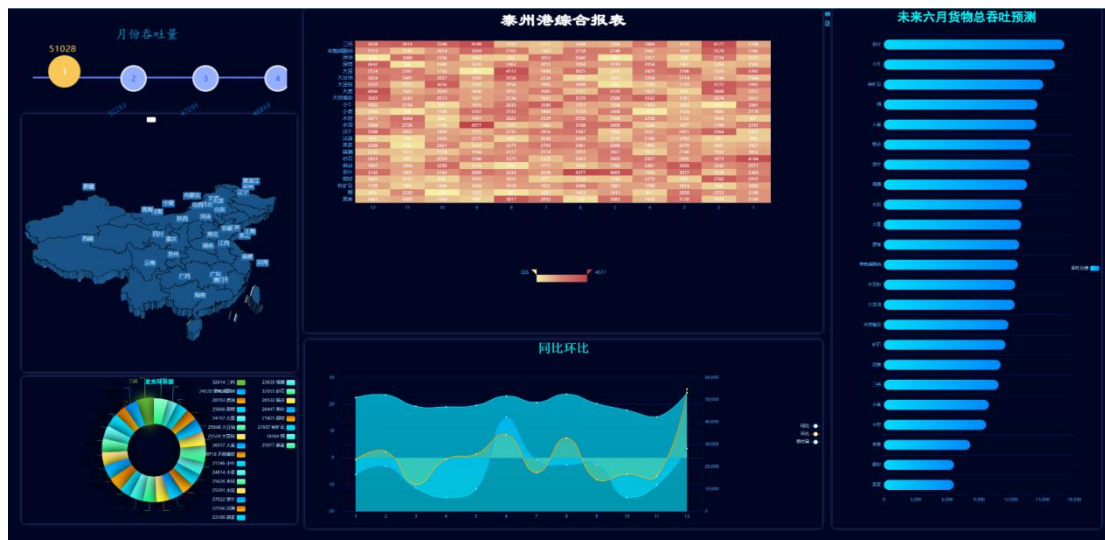
修改

港口货物入口量一览



2021年港口吞吐量同比环比



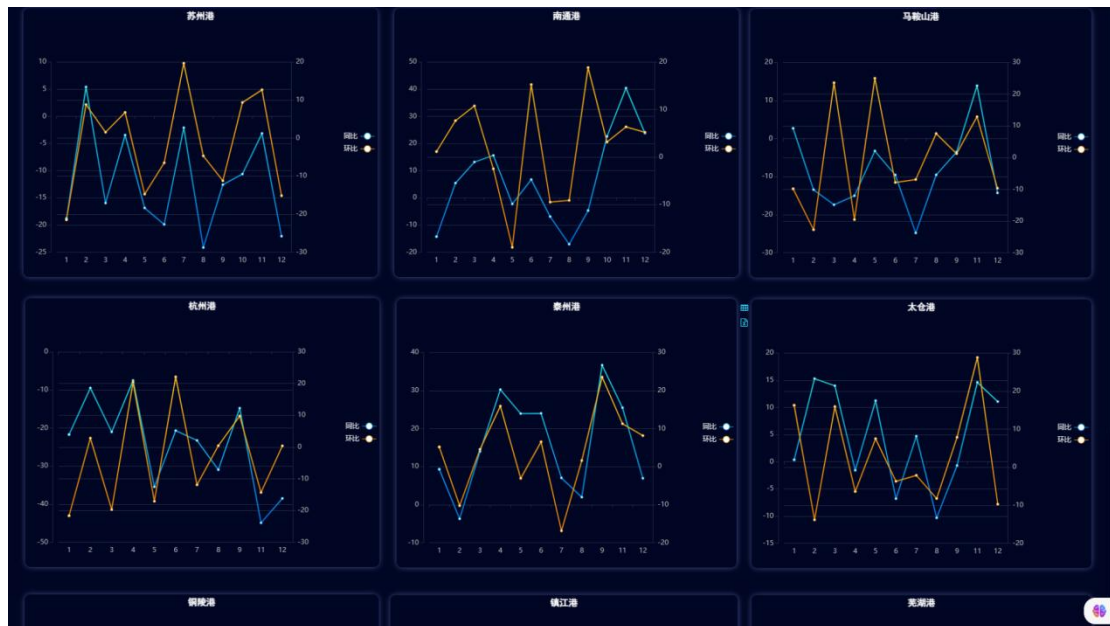


(注: 该页面上方会提示本系统在达梦大数据分析平台的系统账号, 如果用户在达梦大数据分析平台没有注册账号可用系统账号进行访问!)

账号:tsytytyt

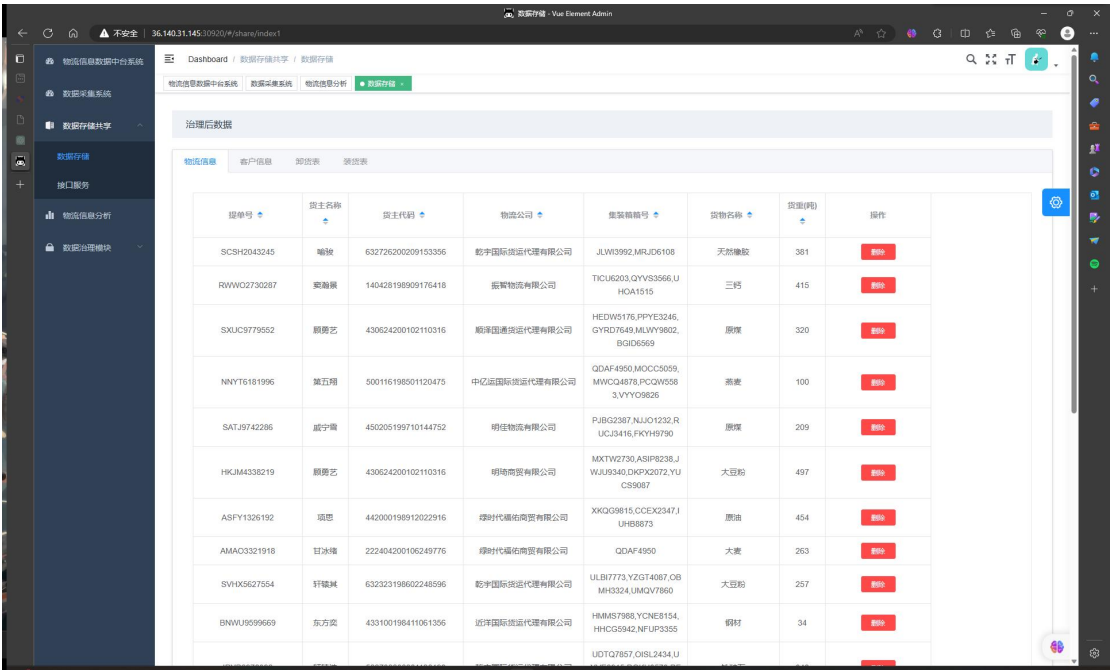
密码:admin123



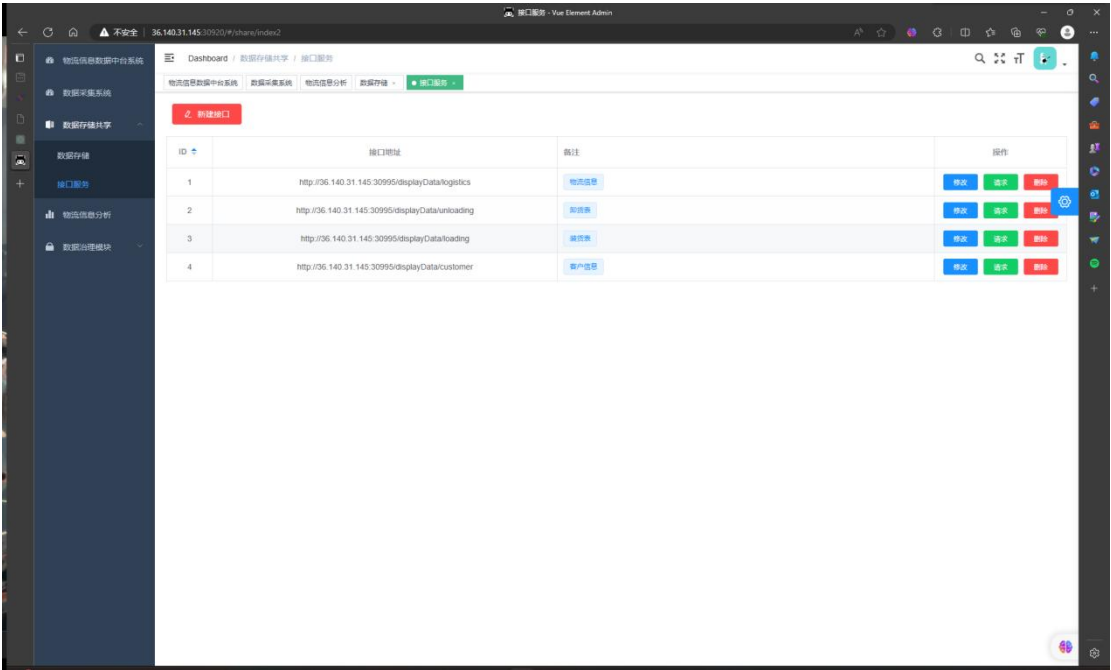


数据共享界面：

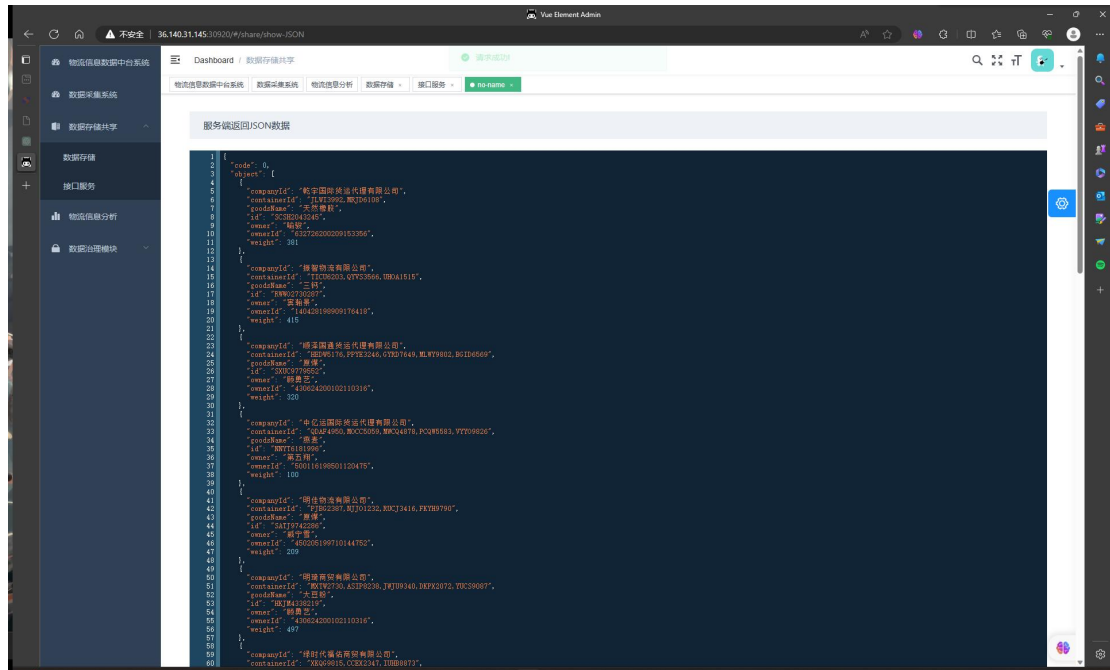
数据存储和接口服务两个页面，数据存储负责展示四张表经过数据治理后的正确数据，接口服务负责展示对外提供数据服务的接口，对外的接口可动态配置页码和每页信息数量。



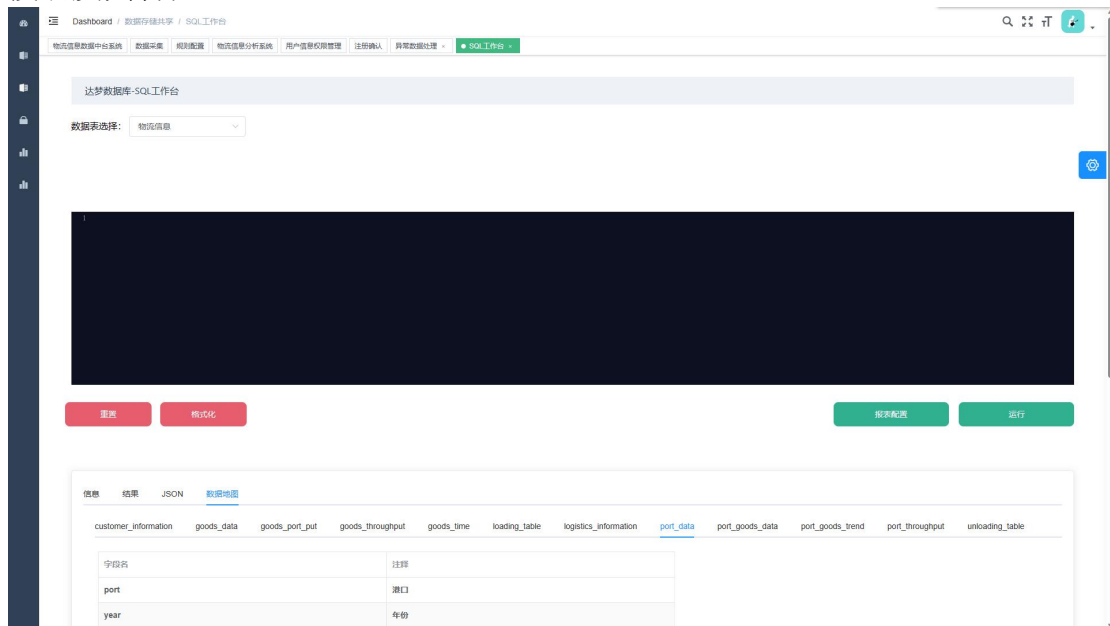
提单号	货主名称	货主代码	物流公司	集装箱号	货物名称	货重(吨)	操作
SCSH2043245	喻敏	632726200209153356	顺丰国际货运代理有限公司	JLW33992.MRJD6108	天然橡胶	381	删除
RWWQ2730287	荣海斌	140428198909176418	蓝箭物流有限公司	TICU6203.QYV33566.UHOA1515	三钙	415	删除
SXUC9779552	顾勇艺	430624200102110316	顺泽国通货运代理有限公司	HEDW5176.PPYE3246.GYRD7549.MLWY9602.BGIG6569	原煤	320	删除
HNNT6181996	第五期	500116198501120475	中亿国际货运代理有限公司	QDAF4950.MOCC5059.MWCC4878.PCQW3583.VYYYQ9626	燕麦	100	删除
SATJ9742286	戴宁南	450205199710144752	明达物流有限公司	PJBG2287.NLJO1232.RUCJ3416.FKXH9790	原煤	209	删除
HKJMA338219	顾勇艺	430624200102110316	明达商贸有限公司	MXTW2730.ASPR6238.JWJUS340.DKPK2072.YUCS9087	大豆粉	497	删除
ASFY1326192	项思	442000198912022916	德时代福佑商贸有限公司	XKQG9815.CCEX2347.JUH88673	原油	454	删除
AMAO3321918	甘冰梅	222404200106249776	德时代福佑商贸有限公司	QDAF4950	大豆	263	删除
SVH05627554	舒毓斌	632323198602248596	顺丰国际货运代理有限公司	ULB17773.YZGT4087.OBMH3324.UMQV7860	大豆粉	257	删除
BNWU8599669	东方炎	433100198411061356	近洋国际货运代理有限公司	HMM57368.YCNE5154.HHCG5942.NFUP3355	钢材	34	删除
				UDTQ7857.OISL2434.U			删除



ID	接口地址	备注	操作
1	http://36.140.31.145:30995/displayData/logistics	物流信息	修改 删除 新增
2	http://36.140.31.145:30995/displayData/unloading	卸货表	修改 删除 新增
3	http://36.140.31.145:30995/displayData/loading	装货表	修改 删除 新增
4	http://36.140.31.145:30995/displayData/customer	客户信息	修改 删除 新增



SQL 工作台可供用户使用 sql 代码访问本系统数据库查询所需数据，共 12 张表格的数据对外开放。为便于查询在下方提供数据地图服务（提示所有表名称、字段名及其备注）



查询结果会在“结果”中以列表形式展示，如果 sql 语法有误则会提示在“信息”中。

```
1 SELECT 'port','goods','month','throughput' FROM 'port_goods_data'
2 WHERE 'year' = 2022
```

重置

格式化

报表配置

运行

信息结果JSON数据地图

month	port	goods	throughput
12	苏州港	小牛	1449
11	苏州港	小牛	1969
10	苏州港	小牛	1543
9	苏州港	小牛	1766
8	苏州港	小牛	889
7	苏州港	小牛	1603

点击“报表配置”后用户可将查询出的信息配置成为报表，具体形式可参考下图（注：报表配置只能使用平台提供的系统账号！）



系统账户如下图：

提示

×

!

跳转至报表配置网站!配置账号:tsytyty。密码:admin123。当前表名:admin1hxeob

取消

跳转

6.2 用户界面组件

用户界面组件包括：

1. 按钮：用于触发特定的操作或功能。
2. 输入框：允许用户输入文本、数字或其他数据。
3. 复选框和单选按钮：用于选择一个或多个选项。
4. 下拉菜单和列表框：提供选项列表供用户选择。
5. 标签和文本区域：用于展示文本或信息。
6. 图像和图标：用于展示图形或图标以增加可视化效果。
7. 进度条和滑块：显示任务的进度或允许用户选择一个范围。
8. 表格和列表：展示数据以表格或列表的形式。
9. 对话框和弹出框：用于显示提示、警告或请求用户确认的消息框。
10. 导航菜单和选项卡：提供导航和切换不同功能模块的选项。

通过这些用户界面组件的布局和交互设计，帮助用户直观地操作系统并获取所需的信息和功能。

6.3 用户交互流程

管理员用户交互流程包括数据源配置、数据采集、数据治理规则配置、数据治理、异常数据的展示与编辑、数据分析、分析结果可视化显示、数据共享（sql 工作台、报表配置、合规数据查看）。

普通用户只能访问数据共享功能。

6.4 设计原则和准则

1. 一致性：保持界面元素的一致性，包括布局、颜色、字体等，使用户能够轻松理解和使用界面。
 2. 可视化层次：使用合适的视觉元素和布局来传达信息的层次结构，帮助用户快速理解和导航界面。
 3. 易学性：界面应该易于学习和掌握，通过简洁明了的设计和直观的操作方式，减少用户的学习成本。
 4. 反馈机制：及时提供反馈信息，如操作状态、错误提示等，让用户知道他们的行为和操作是否成功。
 5. 弹性和容错性：设计界面时要考虑到用户的不同操作方式和可能的错误输入，提供合理的容错机制和恢复功能。
 6. 导航与布局：合理的导航和布局帮助用户快速找到所需内容，减少混乱和迷失的可能性。
 7. 易用性：简化复杂任务，提供明确的操作流程和引导，使用户能够快速完成任务。
 8. 可访问性：考虑到残障用户的需求，提供辅助功能和易于访问的界面设计。
- 这些原则旨在提供一个直观、高效、易用和愉悦的界面，以满足用户的期望和需求。

7 测试策略

7.1 测试目标

在本地环境下搭建 minio、hdfs 存储、minio 存储，在达梦启云云原生大数据平台搭建达梦 8 数据库容器。通过 Java、springboot 项目与 springcloud 微服务技术，连接 4 端数据源进行本地开发预测试，测试无误后将 jar 包打包成镜像推送至平台进行线上环境测试。最后经过版本迭代，将最终稳定版本前后端代码部署到平台上。

7.2 测试环境

测试环境主要为自建 mysql、minio 存储、hdfs 存储，地址如下图：

```
secondary:
  driver-class: com.mysql.cj.jdbc.Driver
  jdbc-url: jdbc:mysql://localhost:3306?useSSL=true&useUnicode=true&characterEncoding=utf8
  username: root
  password: 123456

minio:
  endpoint: http://123.56.252.188:9000
  accessKey: test
  secretKey: 
  bucketName: tsytyty

hdfs:
  url: hdfs://192.168.10.101:9820
```

本地通前后端开发人员通过使用 EasyN2N 开源软件，建立远程局域网，通过虚拟 IP 映射技术进行联调，大大节约调试成本。

7.4 验收标准

1. 功能完整性：所有主要功能和模块已经按照需求规格说明书中的要求实现，并且能够正常运行，没有明显的功能缺陷或错误。
2. 数据准确性：系统能够正确采集、治理和分析数据，并生成准确的结果。经过多组测试数据验证，数据的处理过程和结果与预期一致。
3. 性能和稳定性：系统能够在正常工作负载下保持稳定，并具备合理的响应

时间。系统在长时间运行和高负载情况下没有出现严重的性能问题和崩溃。

4. 用户界面友好性：系统的用户界面设计简洁明了，操作流程合理，能够提供良好的用户体验。用户能够轻松理解和操作系统的各项功能。

5. 安全性和数据保护：系统具备必要的安全措施，能够保护用户数据和系统的机密性、完整性和可用性。系统能够防止未经授权的访问和数据泄露。

6. 文档和培训支持：系统的相关文档和用户手册完备，清晰地描述了系统的安装、配置和使用方法。此外，培训支持提供了系统使用的相关培训和技术支持。

根据项目的具体要求和客户需求，可以进一步细化和扩展以上的验收标准，确保对程序的验收评估具有全面性和准确性。

8 总结

此物流信息数据可视化中台基于国产系统（银河麒麟 v10）和全国产核心中间件（Apach shenyu 网关、TongWeb 容器、TongRDS 缓存、nacos 注册中心、TongHttpServer 反向代理、达梦 8 数据库），采用分布式微服务系统架构，在满足信创要求自主可控的同时引入了企业级分布式微服务系统架构。

虽然国产产品生态差，中间件整合困难，但我们不畏困难，敢于挑战。我们坚信，正是在面对挑战和困难的时候，我们才能够展现出真正的实力和勇气。因此，我们毅然选择了国产系统和核心中间件，并不断努力，不断尝试，不断创新。希望这次的参赛可以给国产信创领域带入新动力，增强国产信创行业的自信心。给国产信创带来新模范。