ICS 35.240.60
CCS L 70

T/CS

中 国 商 品 学 会 团 体 标 准

T/CS 067—2025

智慧物流网络货运系统

Intelligent logistics network freight system

2025－06－11 发布　　2025－06－11 实施

中国商品学会　　发 布

目　　次

前言 II
1　范围 1
2　规范性引用文件 1
3　术语和定义 1
4　系统架构设计 1
5　系统架构 2
6　安全要求 5

前　言

本文件按照GB/T 1.1—2020《标准化工作导则　第1部分：标准化文件的结构和起草规则》的规定起草。

请注意本文件的某些内容可能涉及专利。本文件的发布机构不承担识别专利的责任。

本文件由江苏星通北斗航天科技有限公司提出。

本文件由中国商品学会归口。

本文件起草单位：江苏星通北斗航天科技有限公司、安徽星通北斗航天科技有限公司、上海原子风暴科技有限公司、唐山芦台经济开发区仰望物流有限公司。

本文件主要起草人：周松林、周伟、谢雨、宋朝辉。

智慧物流网络货运系统

1 范围

本文件规定了智慧物流网络货运系统的系统架构设计、系统架构、安全要求。
本文件适用于智慧物流网络货运系统的构建和运行。

2 规范性引用文件

本文件没有规范性引用文件。

3 术语和定义

本文件没有需要界定的术语和定义。

4 系统架构设计

4.1 目标与需求

4.1.1 实时监控：系统需要能够实时监控企业的资金流入和流出，确保财务数据的实时更新和准确性。
4.1.2 数据分析：系统应具备对财务数据进行分析的能力，帮助企业做出基于数据的决策。
4.1.3 合规性检查：系统需要确保所有财务操作符合相关法律法规，减少企业合规风险。
4.1.4 用户友好：系统应提供直观的用户界面，确保不同技术背景的用户都能轻松使用。
4.1.5 可扩展性：系统架构应支持未来的功能扩展和升级，以适应企业的发展需求。

4.2 技术选型

4.2.1 前端技术选型（Vue.js）

4.2.1.1 响应式设计优势：Vue.js 支持响应式布局，可使系统界面在不同设备上自适应显示，为用户提供一致且优质的体验，方便物流人员随时随地通过多种终端设备访问系统。
4.2.1.2 组件化开发高效性：其组件化特性允许将复杂界面拆分为多个独立、可复用的组件，提升开发效率，降低维护成本。例如，可创建独立的货物跟踪组件、订单管理组件等，每个组件单独开发、测试与维护。
4.2.1.3 丰富生态助力拓展：拥有庞大的社区和丰富的生态系统，涵盖各类插件与工具，便于快速拓展系统功能。如利用相关插件实现地图导航展示车辆位置、数据可视化展示货运分析结果等功能。
4.2.1.4 卓越性能保障体验：在处理大量数据和复杂交互时性能出色，能确保系统在实时更新货运信息等场景下保持流畅运行，避免卡顿，提升用户操作体验

4.2.2 后端技术选型（Spring Boot）

4.2.2.1 快速开发特性：Spring Boot 简化了 Spring 应用的搭建与开发流程，开发人员可快速构建后端服务，缩短项目开发周期，使系统能更快上线投入使用，满足物流企业对业务快速响应的需求。
4.2.2.2 自动配置提升效率：自动配置功能大幅减少配置文件编写量，开发人员无需花费大量时间在繁琐的配置工作上，可将更多精力聚焦于业务逻辑实现，提高开发效率。
4.2.2.3 微服务架构支持：对微服务架构的良好支持，便于将系统拆分为多个独立的微服务（如订单服务、车辆管理服务、数据统计服务等），每个微服务独立开发、部署与扩展，提升系统的可维护性和扩展性，以适应物流业务的多样化和动态变化。
4.2.2.4 高稳定性与成熟度：经过广泛的生产环境验证，稳定性高、成熟度好，能为物流系统的稳定运行提供坚实保障，确保业务连续性。

4.2.3 数据库选型（MySQL）

4.2.3.1 成本效益显著：MySQL 是开源关系数据库管理系统，企业使用无需支付高昂的软件授权费用，可有效降低成本，同时能获得活跃社区的免费技术支持与丰富资源。
4.2.3.2 高性能与可靠性：以高性能和可靠性著称，具备强大的并发处理能力，能够稳定存储和快速检索海量的物流数据（如大量的订单信息、车辆行驶轨迹数据等），满足物流业务高并发的操作需求。
4.2.3.3 维护简便易行：维护相对简单，社区资源丰富，遇到问题时，开发人员可轻松在社区找到解决方案，降低维护难度与成本。
4.2.3.4 广泛应用便于学习：在全球广泛应用，拥有大量的文档和教程，开发人员学习成本低，便于快速掌握其开发与使用技巧，加速项目开发进程。

5 系统架构

5.1 基本架构

5.1.1 本系统的架构设计采用了分层的设计思想，将系统分为表现层、业务逻辑层、数据访问层和数据层四个主要层次，以实现职责清晰、高内聚低耦合的系统设计。

5.1.2 表现层

负责与用户交互，提供可视化界面，接收用户输入指令，展示系统处理结果。运用 Vue.js 框架构建单页面应用（SPA），实现动态交互效果，无需频繁重新加载页面即可更新内容，提升用户操作流畅性。

5.1.3 业务逻辑层

承担业务逻辑处理和事务管理工作。Spring Boot 框架在此层发挥核心作用，提供 RESTful API 与前端交互数据，并处理物流业务逻辑，如订单调度、车辆分配、货运路线规划、费用结算等。

5.1.4 数据访问层

作为业务逻辑层与数据层的连接桥梁，负责执行数据的创建、读取、更新、删除（CRUD）操作。采用 Spring Data JPA 或 MyBatis 等 ORM 工具，简化数据库操作，提高开发效率，确保业务逻辑层能高效地访问和管理数据。

5.1.5 数据层

负责数据的持久化存储与检索。MySQL 数据库系统存储物流业务数据，如订单数据、车辆信息、用户信息等，提供高效的数据存储和快速检索能力，保障数据的安全性与完整性。

5.2 前后端分离架构

5.2.1 前端架构

基于 Vue.js 框架，结合 Element UI 组件库构建响应式、组件化的用户界面。Vue.js 的单文件组件结构让开发人员可在同一文件内编写模板、逻辑和样式，简化开发流程。其虚拟 DOM 和响应式数据绑定机制，提高页面渲染效率和数据更新性能，确保前端界面快速响应后端数据变化。

5.2.2 后端架构

运用 Spring Boot 框架搭建 RESTful 服务。Spring Boot 的自动配置和起步依赖功能降低传统 Spring 应用的配置复杂度，使后端服务能快速启动和稳定运行。后端通过定义清晰的 API 接口与前端交互数据，接口遵循 REST 原则，便于前端开发人员理解和调用。

5.2.3 通信机制

前后端通过 HTTP 协议进行通信，前端借助 Axios 库向后端发送请求，并接收 JSON 格式的响应数据。这种通信方式简单高效，易于调试和测试，保障数据传输的稳定性和准确性。

5.2.4 安全性保障

采用 JWT（JSON Web Tokens）进行身份验证和授权，确保前后端通信安全。Spring Security 框架提供全面的安全控制，涵盖认证、授权、防止 CSRF 攻击等功能，保障系统免受各类安全威胁，确保物流数据和业务操作的安全性。通过前后端分离架构，系统能够实现快速迭代开发，提升可维护性和可扩展性，为后续功能升级和业务拓展提供便利。

5.3 前端技术栈

5.3.1 Vue.js 框架应用

5.3.1.1 组件化开发：将前端界面划分为多个独立、可复用的组件，每个组件负责特定功能模块的 UI 渲染。例如，创建订单列表组件展示待处理订单、车辆跟踪组件实时显示车辆位置信息等，提高代码可维护性和可测试性。
5.3.1.2 响应式数据绑定：其响应式系统自动追踪数据变化并同步更新 DOM，减少手动操作 DOM 的工作量，提高开发效率。在物流系统中，货物运输状态、车辆位置等数据实时变化，Vue.js 的响应式数据绑定确保前端界面及时准确地展示最新信息。
5.3.1.3 集成：Vue.js 可灵活集成到现有项目或与其他技术栈协同工作。若物流企业已有部分系统模块，Vue.js 能方便地与之集成，实现系统功能的逐步升级和扩展。

5.3.2 前端路由管理

5.3.2.1 Vue Router 功能：Vue Router 作为 Vue.js 官方路由管理器，通过 URL 控制页面渲染。在本系统中，用于管理登录页、订单管理页、货运监控页、报表生成页等不同页面的导航，实现页面间快速切换，无需重新加载整个页面，提升用户操作体验。
5.3.2.2 页面懒加载：支持页面懒加载技术，仅在用户访问特定页面时才加载该页面资源，有效提高系统加载速度，减少服务器负载，尤其适用于包含大量数据和复杂组件的页面，如报表生成页。
5.3.2.3 嵌套路由：支持嵌套路由，满足复杂应用的导航需求。在财务资金流转监控软件中，可能存在多个层级的导航，如在监控仪表板下有多个子页面，每个子页面显示不同类型的资金流转信息。

5.3.3 前端状态管理

5.3.3.1 Vuex 集中式存储：Vuex 作为状态管理模式和库，采用集中式存储管理应用各组件的状态。在本系统中，用于管理用户登录状态、货运订单数据、车辆状态等全局状态，确保数据在不同组件间的一致性和共享性。
5.3.3.2 模块化状态管理：支持模块化，将不同功能的状态划分为独立模块进行管理。如将用户管理状态、订单管理状态、货运监控状态分别独立管理，使状态管理更清晰、有条理，便于维护和扩展。
5.3.3.3 严格模式辅助开发：严格模式下，Vuex 可追踪状态变化，在开发过程中帮助开发者快速发现和修复状态管理中的错误，提高开发效率和代码质量。

5.4 后端技术栈

5.4.1 Spring Boot 框架应用

5.4.1.1 自动配置简化流程：自动配置功能减少大量配置工作，开发人员只需进行简单配置，即可自动装配数据库连接、消息队列等常用组件，将更多精力投入到业务逻辑开发中。
5.4.1.2 微服务架构灵活拓展：支持微服务架构，可将系统拆分为多个独立微服务，如订单服务负责订单的创建、处理与跟踪；车辆管理服务管理车辆信息、调度和维护；数据统计服务进行货运数据的分析与报表生成。各微服务独立开发、部署和扩展，提升系统的灵活性和可维护性。
5.4.1.3 RESTful API 开发支持：提供强大的 RESTful API 开发功能，包括路径映射、请求处理等。后端通过 RESTful API 与前端交互数据，为前端提供各类物流数据接口，如订单详情查询接口、车辆位置获取接口等。
5.4.1.4 集成测试保障稳定：支持集成测试，开发人员可在模拟服务器环境中对整个应用进行测试，确保系统在不同场景下的稳定性和可靠性，减少上线后的故障风险。

5.4.2 数据接口设计

5.4.2.1 统一接口风格：遵循 RESTful 原则，设计统一风格的 API 接口。使用标准 HTTP 方法（GET、POST、PUT、DELETE）处理不同类型的资源请求，便于前端开发人员理解和使用，提高开发效率。
5.4.2.2 数据序列化规范：采用 JSON 作为数据交换格式，借助 Jackson 等库实现对象与 JSON 之间的序列化和反序列化，确保数据在前后端之间高效、准确地传输，保证数据的一致性和完整性。
5.4.2.3 版本控制策略：在 API 路径中包含版本号（如/api/v1/），便于未来对 API 进行升级和变更时，保持向后兼容性，不影响现有前端应用的正常使用，保障系统的稳定性和可持续发展。
5.4.2.4 错误处理机制：建立统一的错误处理机制，后端在发生错误时返回清晰、一致的错误信息给前端，包括错误代码、错误描述等。前端根据这些信息进行相应处理，如提示用户错误原因、引导用户进行正确操作等，提升用户体验。

5.4.3 安全性与认证机制

5.4.3.1 JWT 认证保障访问安全：采用 JSON Web Tokens（JWT）进行用户认证，用户登录成功后，系统生成 JWT 并返回给前端。前端在后续请求中携带 JWT，后端验证 JWT 的有效性，确保只有合法用户才能访问受限资源，防止非法访问和数据泄露。
5.4.3.2 RBAC 实现权限控制：借助 Spring Security 框架实现基于角色的访问控制（RBAC）。为不同用户角色（如管理员、调度员、司机等）分配相应的操作权限和数据访问权限，确保用户只能执行其角色允许的操作，访问相应的数据，保障系统操作的安全性和数据的保密性。
5.4.3.3 数据加密保护敏感信息：使用 SSL/TLS 加密数据传输，防止数据在网络传输过程中被窃取或篡改。对系统中的敏感数据（如用户密码、支付信息等）进行加密存储，采用合适的加密算法（如 BCrypt），确保数据的安全性。
5.4.3.4 防范常见安全威胁：利用 Spring Security 提供的防护机制，有效防止 SQL 注入、跨站脚本攻击（XSS）、跨站请求伪造（CSRF）等常见安全威胁。定期进行安全漏洞扫描和修复，保障系统的安全性和稳定性。
5.4.3.5 审计日志记录关键操作：记录关键操作的审计日志，包括用户登录、订单创建与修改、车辆调度等操作，以及系统异常信息。审计日志用于事后分析和追踪问题，便于排查安全事故原因、监督用户操作行为，保障系统安全运行。

5.5 数据库设计

5.5.1 MySQL 数据库选择依据

5.5.1.1 开源成本与社区支持：作为开源数据库，可降低企业软件采购成本。同时，拥有活跃的社区，开发人员能获取丰富的技术支持、文档资料和解决方案，便于解决开发和运维过程中遇到的问题。
5.5.1.2 可扩展性满足业务增长：支持从单服务器到复杂的多服务器配置，包括分区、复制和集群技术。随着物流业务量的增长，可灵活扩展数据库架构，满足大规模数据存储和高并发访问的需求。
5.5.1.3 性能优化提升效率：提供多种性能优化工具和技术，如索引优化、查询缓存、存储过程等。通过合理使用这些技术，可提高物流数据的检索和处理速度，确保系统快速响应业务请求，提升用户体验。
5.5.1.4 数据完整性保障数据质量：支持外键、触发器和事务处理，能够维护物流数据的完整性和一致性。例如，在订单处理过程中，通过事务确保订单创建、车辆分配、货物装卸等操作的原子性，避免数据不一致问题。
5.5.1.5 多平台支持灵活部署：可在多种操作系统（如 Linux、Windows、macOS）上运行，为系统部署提供灵活性，企业可根据自身技术架构和服务器环境选择合适的操作系统进行数据库部署。

5.5.2 数据库模型设计细节

5.5.2.1 实体关系模型（ER Model）构建：采用 ER 模型组织数据，主要实体包括用户（User）、订单（Order）、车辆（Vehicle）、货物（Goods）、运输线路（Route）等。通过外键约束建立实体之间的关联关系，如用户与订单之间的一对多关系，车辆与订单之间的多对多关系（通过中间表实现）。
5.5.2.2 表结构设计合理规划：每个实体对应一个或多个数据库表。例如，用户表（User）包含用户 ID、用户名、密码、角色、联系方式等字段；订单表（Order）包含订单 ID、用户 ID、车辆 ID、货物 ID、运输线路 ID、订单状态、下单时间、预计送达时间等字段。

5.5.2.3　索引优化提升查询性能：对经常用于查询条件的字段建立索引，如订单表中的订单 ID、用户 ID，车辆表中的车牌号等。索引可加速数据检索，提高系统查询效率，满足物流业务对数据快速查询的需求。

5.5.2.4　视图设计简化复杂查询：为简化复杂查询操作，设计多个视图（View）。如创建一个视图展示特定时间段内所有订单的运输进度汇总信息，方便企业管理人员快速掌握物流整体情况，生成相关报表。

5.5.2.5　存储过程和触发器实现自动化操作：定义存储过程（Stored Procedure）和触发器（Trigger）实现自动化操作。例如，通过存储过程实现批量订单数据的导入和处理；利用触发器在订单状态变更时自动更新相关统计数据，确保数据的实时性和准确性。

5.5.2.6　数据备份与恢复策略保障数据安全：制定完善的数据备份与恢复策略，定期对数据库进行备份，包括全量备份和增量备份。在数据丢失或损坏时，可依据备份数据快速恢复系统，保障物流业务的连续性，减少数据损失。

5.5.2.7　数据一致性和完整性保障措施：运用事务（Transaction）确保操作的原子性，保证数据一致性。通过外键约束和其他约束条件，保证数据的引用完整性，防止非法数据插入和更新，维护数据库中数据的准确性和可靠性。

5.5.2.8　安全性设计保护敏感数据：严格控制数据库用户权限，为不同用户角色分配不同的权限，确保只有授权用户才能访问敏感数据。对密码等敏感数据进行加密存储，防止数据泄露，保障数据安全。

6　安全要求

6.1　网络安全

6.1.1　网络架构安全：采用合理的网络架构，划分不同的网络区域，设置防火墙、入侵检测系统等网络安全设备，防止外部网络攻击。

6.1.2　网络通信安全：对网络通信数据进行加密传输，防止数据被窃取或篡改。

6.1.3　网络访问控制：实施严格的网络访问控制策略，限制内部和外部用户对系统网络资源的访问权限。

6.2　数据安全

6.2.1　数据存储安全：采用可靠的数据存储技术，对数据进行备份和冗余存储，防止数据丢失。同时，对存储的数据进行加密处理，确保数据的安全性。

6.2.2　数据传输安全：在数据传输过程中，采用安全的传输协议，如 SSL/TLS 等，对数据进行加密传输，防止数据泄露。

6.2.3　数据使用安全：对数据的使用进行严格的权限管理，只有经过授权的用户才能访问和使用相关数据。同时，对数据的使用行为进行审计和记录，以便追溯和管理。

6.3　应用安全

6.3.1　应用程序安全：对系统的应用程序进行安全开发和测试，确保应用程序不存在漏洞和安全隐患。同时，定期对应用程序进行安全更新和升级，修复已知的安全问题。

6.3.2　用户认证与授权安全：采用安全可靠的用户认证和授权机制，防止用户身份被冒用和非法访问系统资源。

6.3.3　安全审计：建立安全审计机制，对系统的操作行为进行审计和记录，及时发现和处理安全事件。