基于Vue应急物资保障系统

摘 要

应急物资的充分有效保障是应急物流实施的核心环节，它有利于保障遭遇突发事故人员的生命和财产安全；有利于在最短时间内有效地恢复社会生产、生活秩序；有利于最大限度地消除经济、社会、政治方面的消极影响。目前我国已经具备了一定的应急物资保障能力以应对各类突发性公共事件，但大多数情况下还难以实现第一时间的应急物资保障。综合而言，主要是因为应急物资保障体系不完善。近年来，我国面临自然灾害和突发性公共事件所带来的巨大考验，政府陆续出台了许多应对突发公共事件的政策应急物流得到了国家的资金支持和政策鼓励。频发的突发性公共事件，产生了巨大的应急物流需求。本文在明确了应急物流、应急物资内涵的基础上，简要分析了我国应急物资保障现状，据此提出了构建我国应急物资保障体系的对策与建议。文中从应急物资保障系统的结构与功能入手，对应急物资保障系统的运作流程及管理体系进行分析，归纳当前我国应急物资保障系统存在的问题，并对应急物资保障系统的优化提出建议，以期促进应急物资保障系统的改进。

**关键词**　应急物资；管理系统；Vue

**Vue-based emergency material security system Abstract**

The full and effective guarantee of emergency materials is the core aspect of emergency logistics implementation, which is conducive to safeguarding the lives and property of people who encounter sudden accidents; to effectively restoring social production and living order in the shortest possible time; and to eliminating the negative economic, social and political impacts to the maximum extent. At present, China already has certain emergency material security capacity to deal with various types of public emergencies, but in most cases it is still difficult to achieve the first time emergency material security. On the whole, this is mainly because the emergency material security system is not perfect. In recent years, China has faced huge tests brought about by natural disasters and public emergencies, and the government has introduced many policies to deal with public emergencies one after another. The frequent occurrence of public emergencies has generated a huge demand for emergency logistics. On the basis of the connotation of emergency logistics and emergency materials, this paper briefly analyses the current situation of emergency materials security in China and accordingly puts forward countermeasures and suggestions for constructing an emergency materials security system in China. Starting from the structure and function of the emergency material security system, the paper analyses the operation process and management system of the emergency material security system, summarizes the problems of the current emergency material security system in China, and makes suggestions for the optimization of the emergency material security system in order to promote the improvement of the emergency material security system.

**Keywords** **Emergency supplies**，**Management Systems**，**Vue**

目录

摘要 I

Abstract II

[第1章 绪论 1](#_Toc99788272)

[1.1 课题背景 1](#_Toc99788273)

[1.2 问题的提出 1](#_Toc99788274)

[1.3 项目的意义 1](#_Toc99788275)

[1.4 国内外发展现状 2](#_Toc99788276)

[1.5 本文的结构 2](#_Toc99788277)

[第2章 开发工具与技术 3](#_Toc99788278)

[2.1 IntelliJ IDEA工作平台介绍 3](#_Toc99788279)

[2.2 MySql数据库介绍 3](#_Toc99788280)

[2.2.1 MySQL数据库特点 3](#_Toc99788281)

[2.2.2 数据库设计原则 3](#_Toc99788282)

[2.3 项目使用框架介绍 4](#_Toc99788283)

[2.3.1 前端Vue介绍 4](#_Toc99788284)

[2.3.2 后端Spring Boot框架介绍 4](#_Toc99788285)

[2.4 本章小结 4](#_Toc99788286)

[第3章 系统需求分析 5](#_Toc99788287)

[3.1 系统目标 5](#_Toc99788288)

[3.2 系统的可行性分析 5](#_Toc99788289)

[3.3 系统非功能需求 5](#_Toc99788290)

[3.3.1 系统性能需求 5](#_Toc99788291)

[3.3.2 系统质量需求 6](#_Toc99788292)

[3.3.3 软硬件环境需求 6](#_Toc99788293)

[3.4 系统功能需求 6](#_Toc99788294)

[3.4.1 应急事件管理模块 6](#_Toc99788295)

[3.4.2 应急物资管理模块 7](#_Toc99788296)

[3.4.3 审批管理模块 8](#_Toc99788297)

[3.4.4 运输状态管理模块 9](#_Toc99788298)

[3.4.5 系统模块 10](#_Toc99788299)

[3.5 本章小结 11](#_Toc99788300)

[第4章 系统概要设计 12](#_Toc99788301)

[4.1 系统总体架构设计 12](#_Toc99788302)

[4.2 系统总体功能设计 12](#_Toc99788303)

[4.2.1 应急事件管理模块 13](#_Toc99788304)

[4.2.2 应急物资管理模块 14](#_Toc99788305)

[4.2.3 审批管理模块 14](#_Toc99788306)

[4.2.4 运输状态管理模块 15](#_Toc99788307)

[4.2.5 系统模块 16](#_Toc99788308)

[4.3 数据库总体设计 16](#_Toc99788309)

[4.3.1 系统E-R模型 16](#_Toc99788310)

[4.3.2 数据实体模型 17](#_Toc99788311)

[4.4 本章小结 19](#_Toc99788312)

[第5章 系统详细设计 20](#_Toc99788313)

[5.1 系统的详细设计 20](#_Toc99788314)

[5.1.1 系统模块详细设计 20](#_Toc99788315)

[5.1.2 应急事件管理模块详细设计 21](#_Toc99788316)

[5.1.3 审批管理模块详细设计 23](#_Toc99788317)

[5.2 系统数据库的详细设计 25](#_Toc99788318)

[5.3 本章小结 29](#_Toc99788319)

[第6章 系统的实现与测试 30](#_Toc99788320)

[6.1 系统实现 30](#_Toc99788321)

[6.1.1 应急事件管理模块实现 30](#_Toc99788322)

[6.1.2 应急物资管理模块实现 31](#_Toc99788323)

[6.1.3 审批管理模块实现 33](#_Toc99788324)

[6.1.4 运输状态管理模块 34](#_Toc99788325)

[6.2 系统测试 35](#_Toc99788326)

[6.2.1 软件测试的方法 35](#_Toc99788327)

[6.2.2 软件测试的主要用例 36](#_Toc99788328)

[6.3 本章小结 37](#_Toc99788329)

[结论 38](#_Toc99788330)

[致谢 39](#_Toc99788331)

[参考文献 40](#_Toc99788332)

[附录 42](#_Toc99788333)

# 绪论

## 课题背景

当各种突发事件发生后 相关事件对人、财、物的需求是应急物流运作的推动力，对应急物流系统运作的时间和成本提出要求，即必须在最短的时间内且物流成本控制在最低的范围之内的前提下保障应急物资需求。应急物资保障系统管理体系涉及到政府机关、部队、企业等不同的个体 涉及到收购、储存、运输、配送等多个环节，还涉及到政策、经费、人事、资讯、技术等不同领域，是一个非常复杂的系统。应急物流指挥调度中心的搭建，应结合物联网技术以应急物流运行基础和运行机制为平台。当各类突发事件发生时 即时开启应急物流信息网络平台，通过感知应急物流需求 指挥调度中心制定应急物流方案，以指挥调度中心为领导 联合物资筹集部门、仓储部门、运输部门、配送部门，通过网络链接、信息手段，及时调动各处资源 实现全面的应急联动。

## 问题的提出

努力健全完善统一、协调一致的应急物流政策法规体系，一方面加快空缺方面的立法做到有法可依、有章可循如非自然气象地质等灾害的法律法规的建设；另一方面完善现有的政策法规，做出明晰和周密的规定各级地方政府和国家机关在突发事件中的职责范围要清晰，企业及公民在突发事件中的责任、义务和权利要明确提高政策法规的适用性和可行性，保障应急物流活动的顺利开展。

## 项目的意义

实施应急物流的必要前提就是高效的物流信息网络，要积极搭建手段多样化的应急物流信息网络，保证网络的安全性提高应急物流信息的可靠性，保证信息的畅通；加强信息资源的共享与交流，促进公共信息系统的畅通，依托政府公共信息平台筹建一个独立的、完整的应急物流公共信息网络平台实现信息共享；建立严格的信息公开制度，充分尊重人民群众的知情权 将真实情况及时传达给民众；建立规范的信息转换机制和科学的信息处理机制 防止信息混乱和无序，通过大量数据和信息的分析建立优化模型。

## 国内外发展现状

我国目前应对自然灾害和突发性公共事件的应急物流预案还不够完善。一是应急预案对应急物资的采购、保管、运输、管理等方面只是制定了粗略的规定，对突发事件应急物流的组织实施很难起到有效的指导作用；二是应急预案的原则性过强 不够灵活变通 ，常常出现因与突发事件的实际情况差异大而使操作性下降的情况 在事件刚发生的时候，会出现地区和行业应急预案衔接不畅执行部门没有灵活组织指挥权利等现象 甚至会导致错失救援良机的情况。我国目前尽管已经制定了部分应急物流方面的法律法规，但在应对处理自然灾害和突发性公共事件等方面的立法仍然相对滞后。而现行法律法规存在缺乏体系性和针对性、缺乏完备性和细节性、缺乏必要的配套支撑等问题。

## 本文的结构

第一章介绍了基于Vue应急物资保障系统进行研究现状分析以及对相关技术进行简单介绍。本文的第二章从技术和专业的角度描述了系统开发过程中使用的技术和框架以及开发中使用的工具。第三章对本系统从可行性、市场需求、用例需求和开发环境进行分析，并给出了每个功能的用例。第四章是系统的总体设计，主要介绍系统的每个模块，并给出每个模块的功能结构图。第五章是详细的设计部分，从概念结构和数据结构两个角度对数据库进行设计。第六章主要是关于系统的测试和实现。最后，总结了系统的优缺点，并根据测试结果列出截图。本系统还具有界面友好，易于使用和管理，易维护等特点，如果能借助网络强大的交互性、广泛的传播性、时空的开放性，将进一步提高查询效率和灵活性。具有性能可靠的突出优点，支持面向对象的大型数据库系统。数据安全，存储量大等功能。后台管理系统是该系统的主要系统，其开发主要包括后台数据库的建立维护和应用程序的开发这两个方面。本文主要介绍整体系统构架及数据库的建立，并详细介绍了客户系统的各个功能模块、后台管理系统模块、系统逻辑结构、数据库相关表设计以及页面功能。

# 开发工具与技术

## IntelliJ IDEA工作平台介绍

Java语言开发的集成环境是IntelliJ IDEA，它是业界IntelliJ认可的最好的Java开发工具之一。 特别是在智能，重构，代码助手，代码审查，代码自动提示和创新的GUI设计等功能上，它非常方便并且具有非凡的工作能力。 它的每个方向都是经过特殊设计的，让员工可以最大程度地执行它，并开发出功能强大的静态代码分析，以符合人体工程学设计。 这种发展不仅有效。 在体验中，这也是一个很好的体验。 在工作平台中对源代码建立索引之后，在每种情况下提出相关建议后，请提供相关建议和经验。 通常，这是一个使用率已经上升并超过Eclipse的IDE，并且在开发经验和功能方面大大超过了Eclipse。

## MySql数据库介绍

MySQL是传统数据库，它是一个免费数据库。 它受到人们的广泛关注和使用，并且随着当今技术的逐渐成熟，它支持越来越多的功能，并且对该平台的支持也在不断扩展。 Internet上的大多数中小型网站使用。 由于其成本低，体积小和运行速度快，最重要的一点是它的开源。 许多中小型网站将选择使用此数据库，以节省成本。 此外，它的功能稳定且性能出色，可以免费使用和修改。

### MySQL数据库特点

以书面形式，Linux，Windows，UXMacOS，AIX等提供了用于多种编程语言的API。 为了确保其可移植性，各种编译器均已在C和C ++中进行了测试。 支持多种操作系统，并且在管理和检查中有许多处理结果记录。 由于其较小的内存和较小的尺寸，它分为社区版本和商业版本两个版本。 在此功能中，中小型网站的开发通常选择使用开源。 它最重要的功能之一就是写入和应用数据。

### 数据库设计原则

对于数据库的思想设计，主要原因是要考虑数据库的基本原则性好处。它特别可靠，安全和稳定，因此可以放心地使用和编程该系统。使用数据库时，其原理包括标准化，可伸缩性和可扩展性，完整性，一致性和安全性。它对数据源特别统一，安全性特别高，可以保护数据不致丢失。在结构上，按照设计理念，也具有良好的开发和移植性。协调各种数据源，并遵循标准化理论来泄漏，更改或破坏目标。当存在影响数据的不良词或词汇时，禁止合法用户使用数据库，以确保数据的一致性和有效性。标准化的数据库设计可减少操作（如删除和修改）期间的异常情况和容忍度，并减少数据冗余等。

## 项目使用框架介绍

### 前端Vue介绍

Vue是用于构建用户界面的框架。 特征声明最少，每个框架都有一个特征，它对多个用户有一些要求。 它的命题特点是强项和弱项，其优点将影响该项目的开发和使用。 简而言之，如果您对它的某些功能不熟悉，它将是使用它的用户的技术测试[5]。 但是在其中，您只能使用其他前端来替换它，但是这样做的结果是，它比更改后的框架具有更少的硬性要求，并且只能逐步使用该框架。 对于此可实现的功能，实现开发思想和实现前端与后端分离的实现需要数据绑定和路由配置。 一系列框架（例如项目的编译和打包过程）将使用此框架。

### 后端Spring Boot框架介绍

随着动态语言（Ruby，Groovy，Scala，Node.js）的普及，Java的开发特别繁琐，配置繁多，开发效率低，部署过程复杂且难以集成第三方技术。 在以上环境中，Spring Boot应运而生。 它使用“习惯胜于配置”的概念（项目中有很多配置，并且还有内置的习惯配置，因此您无需手动配置它）来使项目运行 迅速地。

使用Spring Boot，可以轻松地基于Spring框架创建一个独立的（运行jar，嵌入式Servlet容器），准生产级别的项目。 借助Spring Boot，您可以在没有或只有很少的Spring配置的情况下使用Spring Boot。

## 本章小结

本章主要研究开发系统是否可行，同时概述了此过程中需要使用的技术以及所需的开发平台和操作环境。

# 系统需求分析

## 系统目标

本文主要的研究目标如下：

(1) 应急事件管理模块

(2) 应急物资管理模块

(3) 审批管理模块

(4) 运输状态管理模块

(5) 系统模块

## 系统的可行性分析

通常，通过系统的支持条件和主要要求，从技术操作，社会经济等方面进行研究和调查。它的任务是以较低的成本和成本确定项目的前景。 这是系统中一种更全面的方法，可以避免错误。确保新系统的成功开发。

基于Vue应急物资保障系统的设计采用了前端和后端分离的设计思想，并且使用代码实现。系统的后端使用流行的后端语言Java和Spring Boot框架，前端使用具有良好中文社区的Vue框架。如果在前端开发中遇到问题，也可以在社区中提出问题并加以解决，因此基于Vue应急物资保障系统技术可以满足开发要求。

## 系统非功能需求

### 系统性能需求

根据性能要求，绘制系统性能要求表，包括的功能有用户数量，多表和单表请求的响应时间等。系统性能需求如表3-1所示。

表3-1 系统性能需求

|  |  |
| --- | --- |
| 需求项 | 说明 |
| 系统用户数 | 30.0 |
| 系统用户并发数 | 10.0 |

续表3-1

|  |  |
| --- | --- |
| 需求项 | 说明 |
| 单表简单查询响应时间 | <=3秒 |
| 多表复杂查询相应时间 | <=6秒 |
| 系统资源平均占有率 | <=10% |

### 系统质量需求

系统质量要求主要介绍系统的可伸缩性，可靠性，可维护性，可审计性，容错性，可移植性等方面。 质量要求显示了系统的容错能力和可伸缩性。如表3-2所示。

表3-2 系统质量需求

|  |  |
| --- | --- |
| 需求项 | 说明 |
| 可扩展性 | 充分考虑系统未来的业务方向，对未开发的功能保留接口 |
| 可靠性 | 从系统结构设计、数据安全、用户权限等方面考虑系统可靠性 |
| 可维护性 | 操作方便，对技术支持人员有方便维护功能 |
| 可审计性 | 可通过日志和数据备份进行核查 |
| 容错性 | 对于常见错误，可在一定范围内修正 |
| 可移植性 | 主要采用常见技术和工具进行开发，具有很强的跨平台能力 |

### 软硬件环境需求

主要介绍服务器的操作系统，用户界面，操作环境和客户端。软硬件环境需求如表3-3所示。

表3-3 软硬件环境需求

|  |  |
| --- | --- |
| 需求项 | 说明 |
| 服务器操作系统 | Windows |
| 用户界面 | B/S结构 |
| 运行环境 | Vscode+IDEA+JDK1.8+MySQL |
| 客户端 | Windows操作系统、主流浏览器 |

## 系统功能需求

### 应急事件管理模块

该模块作为应急物资调度的先导而存在，可以通过该模块进行应急事件的建立，维护，并可以按照实际生活中的处理进度，更新某个应急事件的处理状态。进行统一的跟进，同时可以在该界面获取到历史的应急事件以及其处理状态，通过多种方式筛选出指定的数据，然后提供了针对某一个应急事件的物资申请发起的功能，可以为这个事件申请物资。

#### 1.应急物资管理员

应急物资管理员登录系统后可以查看已经建立列表，包括ID、事件名称、发生地点、坐标、创建时间、时间等级、创建人ID、详细信息等，可根据事件等级和发生地点进行模糊筛选查询，也可以进行查看现场图片、查看位置和删除操作。应急物资管理员功能用例如图3-1所示。

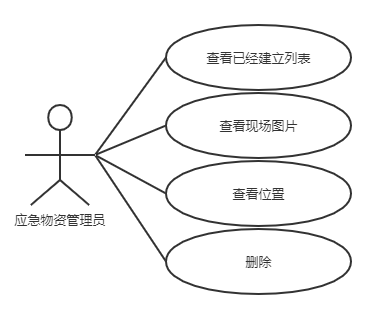


图3-1 应急物资管理员功能用例图

#### 2.应急响应人员

应急响应人员登录后可以实现查看已建立列表、新建应急事件、发起物资申请等功能，其中发起物资申请功能中包括创建用户ID、关联事件ID、指定审批人、申请标题、申请优先级（低\中\高\紧急）、申请内容和选择申请物资，点击提交保存填写。应急响应人员用例如图3-2所示。

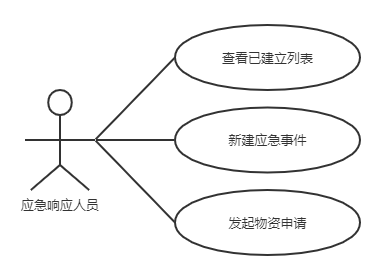


图3-2应急响应人员用例图

### 应急物资管理模块

该模块主要是对应急物资以及其分类进行统一的管理，包含对应急物资的新增，支出，信息更新，删除。并提供了多种筛选方式可以边快速的获取到应急物资的当前情况。针对物资分类也提供了类似的功能。

#### 1.应急物资管理员

应急物资管理员登录系统后可以实现新建应急物资分类、新增物资、应急物资分类列表、物资列表等功能。应急物资管理员用例如图3-3所示。

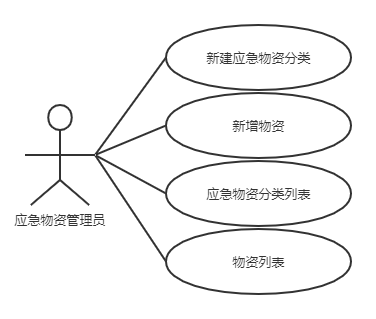


图3-3 应急物资管理员模块用例图

#### 2.运输人员

运输人员登录系统后权限相对较低，可以查看应急物资分类列表和物资列表。运输人员模块用例如图3-4所示。

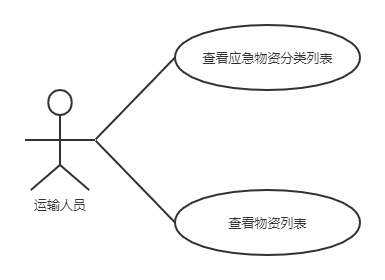


图3-4 运输人员用例图

### 审批管理模块

该模块主要是配合应急事件管理模块，对来自该模块下建立的申请进行审批，结合实际的情况进行处理，选择通过审批或者驳回。只有经过审批之后，相关的流程才可以继续流转下去。

#### 1.应急物资管理员

应急物资管理员登录系统后可以查看已建立申请列表，包括ID、申请标题、详细内容、创建时间、最后更新时间、申请等级、申请状态、关联审批人ID、关联事件ID、创建人ID等，可根据申请等级和申请状态进行模糊筛选查询，也可以进行查看物资和删除操作。应急物资管理员用例图如图3-5所示。

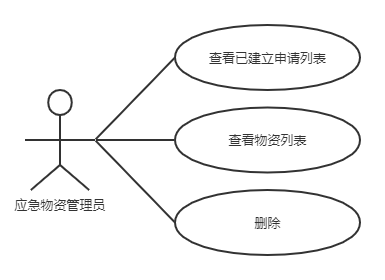


图3-5 应急物资管理员用例图

#### 2.审批人员

审批人员登录系统后可以查看待审批列表和已建立申请列表，并根据申请等级和审批待处理人进行筛选申请。审批人员用例图如图3-6所示。

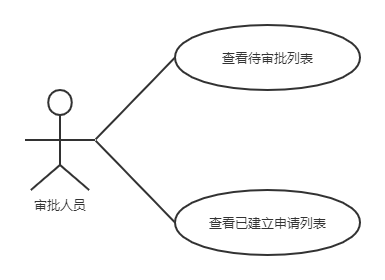


图3-6 审批人员用例图

### 运输状态管理模块

通过该模块可以对已经通过审批的申请建立物资运输单，进而对申请的物资进行运输工作，并且可以在这个过程中对其进行物资的流转等操作，应急事件管理人员也可以在该模块下选择结束流转一个运输单，表示已经收到了物资。

#### 1.应急物资管理员

应急物资管理员登录系统后可以查看已建立配送单，包括ID、关联事件ID、创建时间、预期到达时间、创建人ID、申请状态、接收人ID、运输方式等，并可以进行查看物资、查看流转记录和结束流转功能，和根据接收人模糊筛选查询。应急物资管理员用例图如图3-7所示。

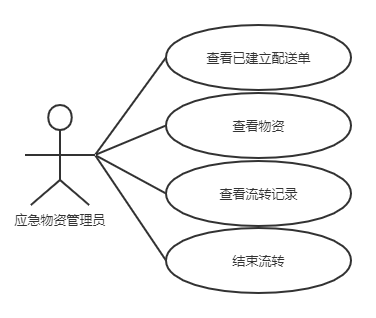


图3-7 应急物资管理员用例图

#### 2.运输人员

运输人员登录系统后可以实现待配送、新建配送单和已建立配送单功能。其中新建配送单功能中包括创建人ID、关联审批标题、接收人ID、物资出库日期、运输方式（汽车\火车\轮船\飞机）和物资列表，点击提交保存填写。运输人员用例图如图3-8所示。

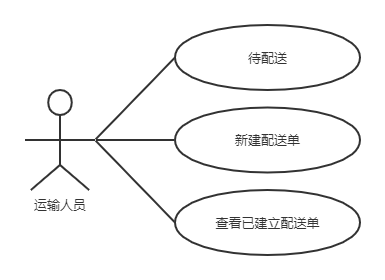


图3-8 运输人员用例图

### 系统模块

通过该模块对用户的信息进行管理，进行登录验证，用户权限判断等操作，并且按照用户的身份为其生成菜单。并记录其操作日志。

## 本章小结

本章首先介绍系统的目的和意义，然后分析系统的可行性，最后给出系统每个功能模块的用例图，并根据用例图描述每个功能模块。

# 系统概要设计

## 系统总体架构设计

系统的总体架构分为客户端，服务层，数据访问层，数据缓存层和数据持久层。服务层执行路由拦截和传输。数据库访问主要使用Mybatis连接到MySQL数据库，并使用Redis缓存数据库。系统的总体架构设计如图4-1所示。



图4-1 系统总体架构图

## 系统总体功能设计

基于Vue应急物资保障系统总共分为五个模块分别是应急事件管理模块、应急物资管理模块、审批管理模块、运输状态管理模块和系统模块。系统整体功能结构图如图4-2所示。

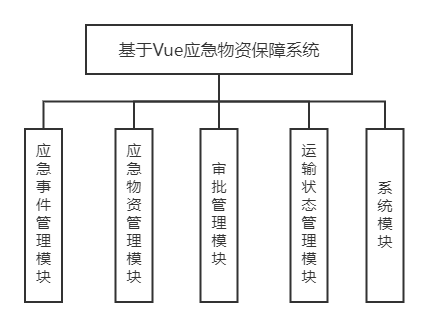


图4-2 系统整体功能结构图

### 应急事件管理模块

该模块作为应急物资调度的先导而存在，可以通过该模块进行应急事件的建立，维护，并可以按照实际生活中的处理进度，更新某个应急事件的处理状态。进行统一的跟进，同时可以在该界面获取到历史的应急事件以及其处理状态，通过多种方式筛选出指定的数据，然后提供了针对某一个应急事件的物资申请发起的功能，可以为这个事件申请物资。

#### 1.应急物资管理员

应急物资管理员登录系统后可以查看已经建立列表，包括ID、事件名称、发生地点、坐标、创建时间、时间等级、创建人ID、详细信息等，可根据事件等级和发生地点进行模糊筛选查询，也可以进行查看现场图片、查看位置和删除操作。

#### 2.应急响应人员

应急响应人员登录后可以实现查看已建立列表、新建应急事件、发起物资申请等功能，其中发起物资申请功能中包括创建用户ID、关联事件ID、指定审批人、申请标题、申请优先级（低\中\高\紧急）、申请内容和选择申请物资，点击提交保存填写。

应急事件管理模块功能结构图如图4-3所示。

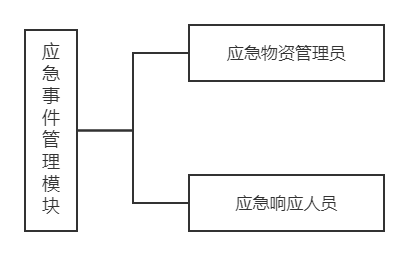


图4-3 应急事件管理模块功能结构图

### 应急物资管理模块

该模块主要是对应急物资以及其分类进行统一的管理，包含对应急物资的新增，支出，信息更新，删除。并提供了多种筛选方式可以边快速的获取到应急物资的当前情况。针对物资分类也提供了类似的功能。

#### 1. 应急物资管理员

应急物资管理员登录系统后可以实现新建应急物资分类、新增物资、应急物资分类列表、物资列表等功能。

#### 2. 运输人员

运输人员登录系统后权限相对较低，可以查看应急物资分类列表和物资列表。

应急物资管理模块功能结构图如图4-4所示。

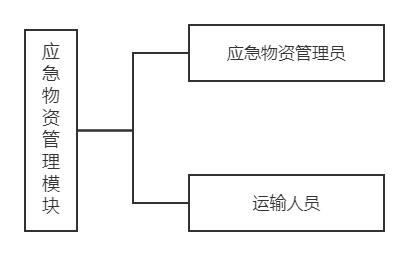


图4-4 应急物资管理模块功能结构图用例图

### 审批管理模块

该模块主要是配合应急事件管理模块，对来自该模块下建立的申请进行审批，结合实际的情况进行处理，选择通过审批或者驳回。只有经过审批之后，相关的流程才可以继续流转下去。

#### 1.应急物资管理员

应急物资管理员登录系统后可以查看已建立申请列表，包括ID、申请标题、详细内容、创建时间、最后更新时间、申请等级、申请状态、关联审批人ID、关联事件ID、创建人ID等，可根据申请等级和申请状态进行模糊筛选查询，也可以进行查看物资和删除操作。

#### 2.审批人员

审批人员登录系统后可以查看待审批列表和已建立申请列表，并根据申请等级和审批待处理人进行筛选申请。

审批管理模块功能结构图如图4-5所示。

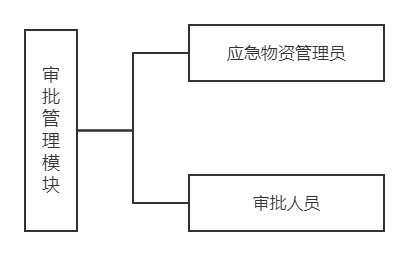


图4-5 审批管理模块功能结构图用例图

### 运输状态管理模块

通过该模块可以对已经通过审批的申请建立物资运输单，进而对申请的物资进行运输工作，并且可以在这个过程中对其进行物资的流转等操作，应急事件管理人员也可以在该模块下选择结束流转一个运输单，表示已经收到了物资。

#### 1.应急物资管理员

应急物资管理员登录系统后可以查看已建立配送单，包括ID、关联事件ID、创建时间、预期到达时间、创建人ID、申请状态、接收人ID、运输方式等，并可以进行查看物资、查看流转记录和结束流转功能，和根据接收人模糊筛选查询。

#### 2.运输人员

运输人员登录系统后可以实现待配送、新建配送单和已建立配送单功能。其中新建配送单功能中包括创建人ID、关联审批标题、接收人ID、物资出库日期、运输方式（汽车\火车\轮船\飞机）和物资列表，点击提交保存填写。

运输状态管理模块功能结构图如图4-6所示。

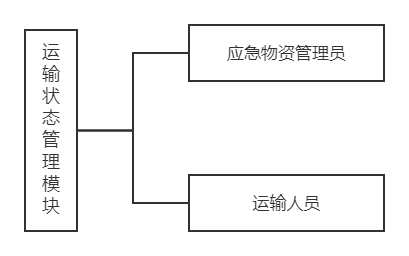


图4-6 运输状态管理模块功能结构图用例图

### 系统模块

通过该模块对用户的信息进行管理，进行登录验证，用户权限判断等操作，并且按照用户的身份为其生成菜单。并记录其操作日志。

系统模块功能结构图如图4-7所示。

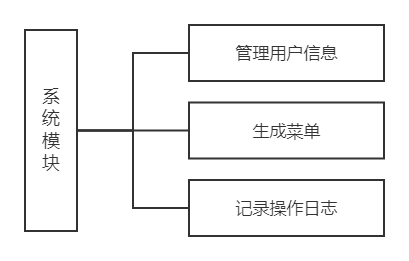


图4-7 系统模块功能结构图用例图

## 数据库总体设计

### 系统E-R模型

系统总体E-R模型包括的实体有普通用户实体、管理员实体、物资实体、运输工具实体、操作日志实体。系统总体E-R图如图4-6所示。

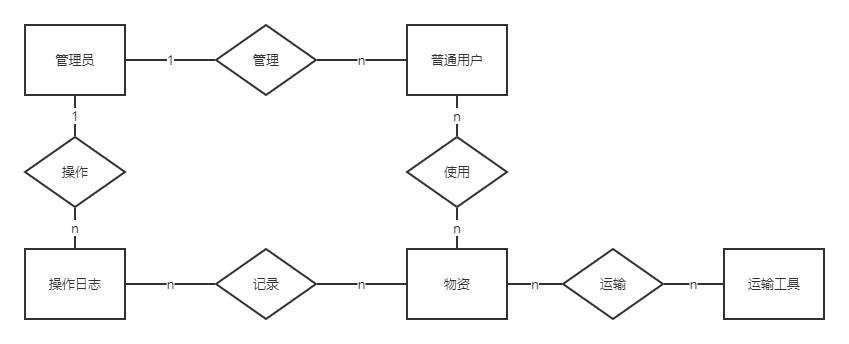


图4-6 系统总体E-R图

### 数据实体模型

1.普通用户实体模型

普通用户实体包括id、姓名、年龄、性别、工作、地点、薪资、婚姻状态。普通用户实体如图 4-7所示。

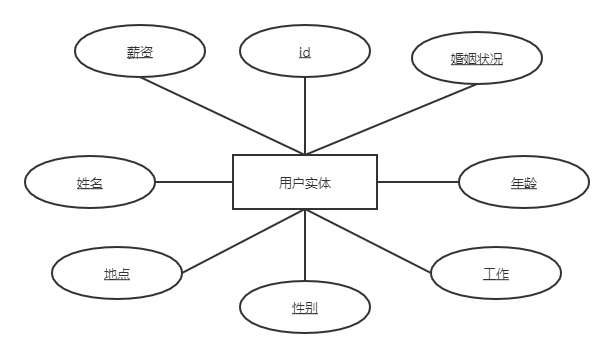


图4-7 普通用户实体图

2.管理员实体模型

管理员实体包括id、用户名、密码、角色。管理员实体如图 4-8所示。

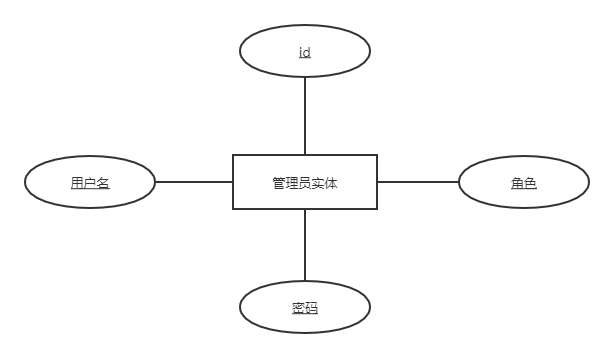


图4-8 管理员实体图

3.物资实体模型

物资实体包括id、物资名称、类别、数量。物资实体如图 4-9所示。

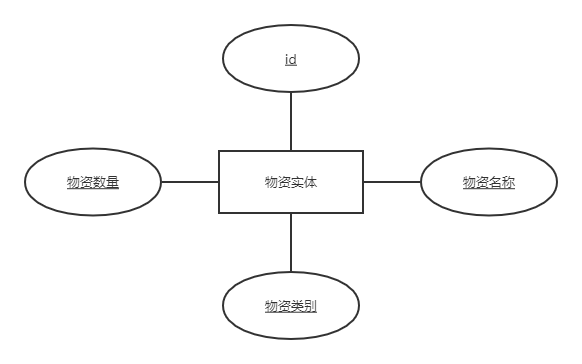


图4-9 物资实体图

4. 运输工具实体模型

运输工具实体包括id、运输工具名称、新增时间、地址。运输工具实体如图 4-10所示。

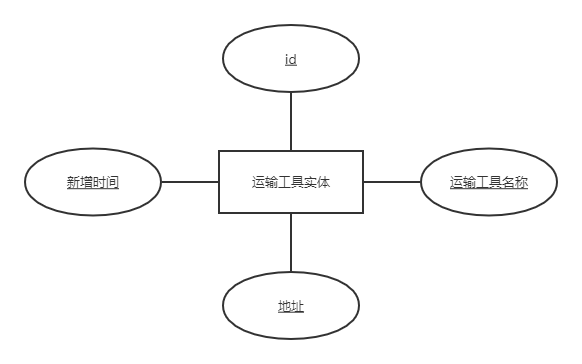


图4-10 运输工具实体图

5.操作日志实体模型

操作日志实体包括id、更新时间、描述。操作日志实体如图 4-11所示。

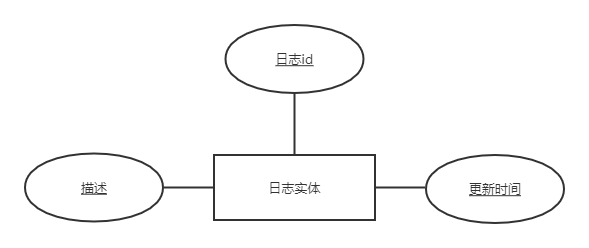


图4-11 操作日志实体图

## 本章小结

本章首先显示系统的总体架构设计，并对系统的模块设计进行简要的分析和描述，以及一些结构图，最后给出整个系统的E-R图并描述每个实体。

# 系统详细设计

## 系统的详细设计

系统详细设计主要包括的模块有：系统模块、应急事件管理模块和审批管理模块。

### 系统模块详细设计

#### 1.系统模块-登录注册详细设计

在使用注册功能时，系统会根据输入的账户名进行校验，由后端对数据库进行比对，如果重复则重新输入新得账户名，否则注册成功，登录到系统主页。用户登录注册模块的流程图如图5-1所示。

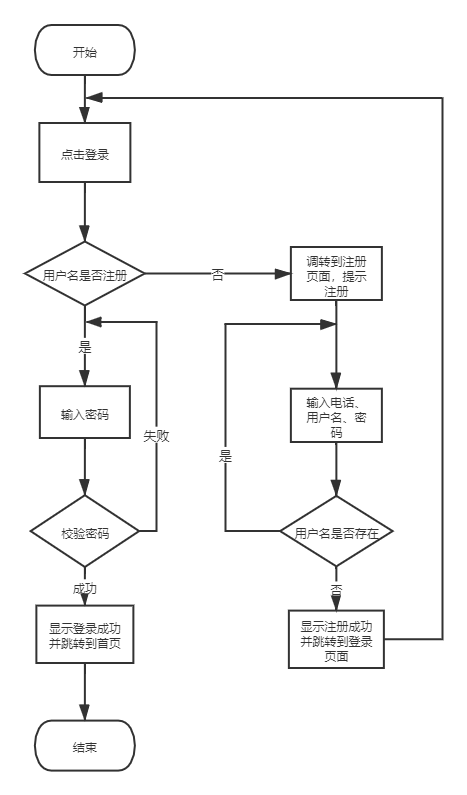


图5-1 登录注册流程图

#### 2.系统模块-退出登录详细设计

用户点击退出登录，系统会跳转到登录页面，清空浏览器缓存以及session。退出登录模块流程图如图5-2所示。

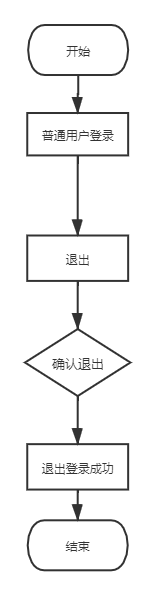


图5-2 退出登录流程图

### 应急事件管理模块详细设计

#### 1. 管理员-查看已经建立列表

应急物资管理员登录系统后可以查看已经建立列表，包括ID、事件名称、发生地点、坐标、创建时间、时间等级、创建人ID、详细信息等，可根据事件等级和发生地点进行模糊筛选查询，也可以进行查看现场图片、查看位置和删除操作。管理员查看已经建立列表流程图如图5-3所示

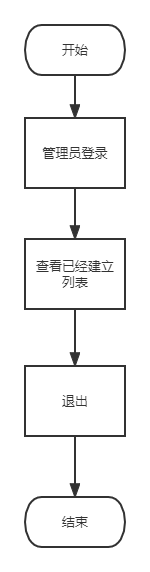


图5-3 管理员查看已经建立列表流程图

#### 2. 应急响应人员-查看已经建立列表

应急响应人员登录后可以实现查看已建立列表、新建应急事件、发起物资申请等功能，其中发起物资申请功能中包括创建用户ID、关联事件ID、指定审批人、申请标题、申请优先级（低\中\高\紧急）、申请内容和选择申请物资，点击提交保存填写。应急响应人员查看已建立列表流程图如图5-4所示

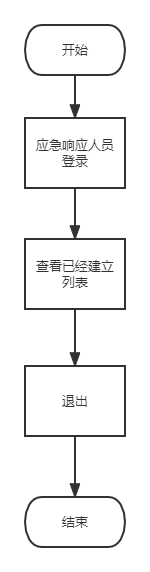


图5-4 应急响应人员查看已建立列表流程图

### 审批管理模块详细设计

#### 1. 应急物资管理员

应急物资管理员登录系统后可以查看已建立申请列表，包括ID、申请标题、详细内容、创建时间、最后更新时间、申请等级、申请状态、关联审批人ID、关联事件ID、创建人ID等，可根据申请等级和申请状态进行模糊筛选查询，也可以进行查看物资和删除操作。管理员查看已经建立申请列表流程图如图5-5所示

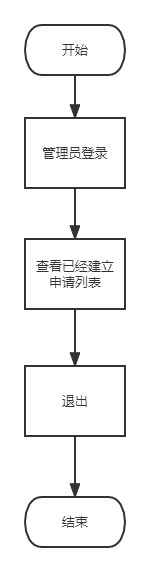


图5-5 管理员查看已经建立申请列表流程图

#### 2. 审批人员

审批人员登录系统后可以查看待审批列表和已建立申请列表，并根据申请等级和审批待处理人进行筛选申请。审批人员登录系统流程图如图5-6所示。

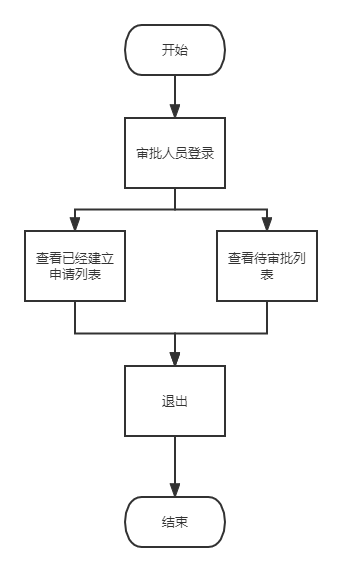


图5-6 审批人员登录系统流程图

## 系统数据库的详细设计

数据库的版本为MySQL5.7，一共包括7张数据表，分别是审批表(t\_approval)、事件表(t\_event)、物资表(t\_material)、物资类别表(t\_material\_category)、操作日志表(t\_operation\_log)、运输表(t\_transport)、用户表(t\_user)。

1.审批表

审批表主要存储用户相关信息，主要包括主键、审批标、审批主体内容、创建时间、最近一次更新时间、审批优先级、审批当前状态、关联审批人ID、物资申请信息、创建人ID、关联应急事件ID。审批表如表5-1所示。

表5-1审批表(t\_approval)

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 含义 |
| id | bigint(20) |  | 主键 |
| title | varchar(255) | 255 | 审批标题 |
| content | text | 65535 | 审批主体内容 |
| create\_time | datetime |  | 创建时间 |
| last\_update\_time | datetime |  | 最近一次更新时间 |
| level | varchar(11) | 11 | 审批优先级 |
| status | varchar(11) | 11 | 审批当前状态 |
| approval\_user\_id | bigint(20) |  | 关联审批人ID |
| material\_apply | text | 65535 | 物资申请信息 |
| create\_user\_id | bigint(20) |  | 创建人ID |
| event\_id | bigint(20) |  | 关联应急事件ID |

2.事件表

事件表包括主键、事件名称、地点、创建时间、创建人ID、等级、受灾图片、详细说明、坐标。事件表如表5-2所示。

表5-2事件表(t\_event)

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 含义 |
| id | bigint(20) |  | 主键 |
| title | varchar(255) | 255 | 事件名称 |
| address | varchar(255) | 255 | 地点 |
| create\_time | datetime |  | 创建时间 |
| create\_user\_id | bigint(20) |  | 创建人ID |
| level | varchar(255) | 255 | 等级 |
| picture | varchar(4096) | 4096 | 受灾图片 |
| detail | text | 65535 | 详细说明 |
| loc | varchar(255) | 255 | 坐标 |

3.物资表

物资表包括主键、物资批次、物资名称、分类名称、库存数量、购进来源、被购方联系人、被购方联系方式、购入合同编号、调配策略、创建日期、创建人。物资表如表5-3所示。

表5-3物资表(t\_material)

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 含义 |
| id | varchar(36) | 36 | 主键 |
| material\_batch\_no | varchar(32) | 32 | 物资批次 |
| material\_name | varchar(32) | 32 | 物资名称 |
| material\_category\_code | varchar(32) | 32 | 分类名称 |
| material\_num | int(32) |  | 库存数量 |
| material\_from | varchar(32) | 32 | 购进来源 |
| material\_contact\_name | varchar(32) | 32 | 被购方联系人 |
| material\_contact\_no | varchar(32) | 32 | 被购方联系方式 |
| material\_contract\_no | varchar(32) | 32 | 购入合同编号 |
| material\_enable | tinyint(1) |  | 调配策略 |
| create\_time | datetime |  | 创建日期 |
| create\_user\_id | bigint(20) |  | 创建人 |

4. 物资类别表

物资类别表包括主键、物资分类编码、物资分类名称、调配策略、创建日期、创建人ID。物资类别表如表5-4所示。

表5-4 物资类别表(t\_material\_category)

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 含义 |
| id | bigint(36) |  | 主键 |
| category\_code | varchar(32) | 32 | 物资分类编码 |
| category\_name | varchar(32) | 32 | 物资分类名称 |
| enable | tinyint(1) |  | 调配策略 |
| create\_time | datetime |  | 创建日期 |
| create\_user\_id | bigint(20) |  | 创建人ID |

5.操作日志表

操作日志表包括主键、请求者IP、模块名称、日志行为简述、方法参数、创建时间、日志产生地方、异常简述、堆栈信息。操作日志表如表5-5所示。

表5-5 操作日志表(t\_operation\_log)

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 含义 |
| id | bigint(16) |  | 主键 |
| ip | varchar(32) | 32 | 请求者IP |
| model\_name | varchar(50) | 50 | 模块名称 |
| log\_description | varchar(50) | 50 | 日志行为简述 |
| action\_args | varchar(1024) | 1024 | 方法参数 |
| create\_time | timestamp |  | 创建时间 |
| log\_location | varchar(256) | 256 | 日志产生地方 |
| error\_msg | varchar(1024) | 1024 | 异常简述 |
| stack\_info | text | 65535 | 堆栈信息 |

6.运输表

运输表包括主键、关联审批ID、创建时间、预期完成时间、创建人、运输状态详情，存储格式、运送物资列表，存储格式、运送状态、接收人信息、运输方式。运输表如表5-6所示。

表5-6运输表(t\_transport)

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 含义 |
| id | bigint(20) |  | 主键 |
| approval\_id | bigint(20) |  | 关联审批ID |
| create\_time | datetime |  | 创建时间 |
| expect\_time | datetime |  | 预期完成时间 |
| create\_user\_id | bigint(20) |  | 创建人 |
| transport\_record | text | 65535 | 运输状态详情，存储格式 |
| material\_record | text | 65535 | 运送物资列表，存储格式 |
| status | varchar(20) | 20 | 运送状态 |
| receiver\_user\_id | bigint(20) |  | 接收人信息 |
| transport\_type | varchar(255) | 255 | 运输方式 |

7.用户表

用户表包括id、账户、密码盐值、昵称、性别、头像地址、地址、身份编码、身份名称、创建时间、账户是否启用，用户表如表5-7所示。

表5-7用户表(t\_user)

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 含义 |
| id | bigint(16) |  | id |
| account | varchar(255) | 255 | 账户 |
| password | varchar(255) | 255 | 密码 |
| salt | varchar(255) | 255 | 盐值 |
| nick\_name | varchar(255) | 255 | 昵称 |
| sex | varchar(255) | 255 | 性别 |
| picture | varchar(255) | 255 | 头像地址 |
| address | varchar(255) | 255 | 地址 |
| identity\_code | int(1) |  | 身份编码 |
| identity\_name | varchar(255) | 255 | 身份名称 |
| create\_time | datetime |  | 创建时间 |
| enable | smallint(1) |  | 账户是否启用 |

## 本章小结

本章介绍的内容通过绘制系统中几个重要模块的流程图，构建了系统的详细设计。同时，从概念结构和数据结构两个角度对数据库进行设计。系统的详细系统设计现已完成。

# 系统的实现与测试

## 系统实现

### 应急事件管理模块实现

#### 1.应急物资管理员

应急物资管理员登录系统后可以查看已经建立列表，包括ID、事件名称、发生地点、坐标、创建时间、时间等级、创建人ID、详细信息等，可根据事件等级和发生地点进行模糊筛选查询，也可以进行查看现场图片、查看位置和删除操作。应急物资管理员登录系统运行示例如图6-1所示。

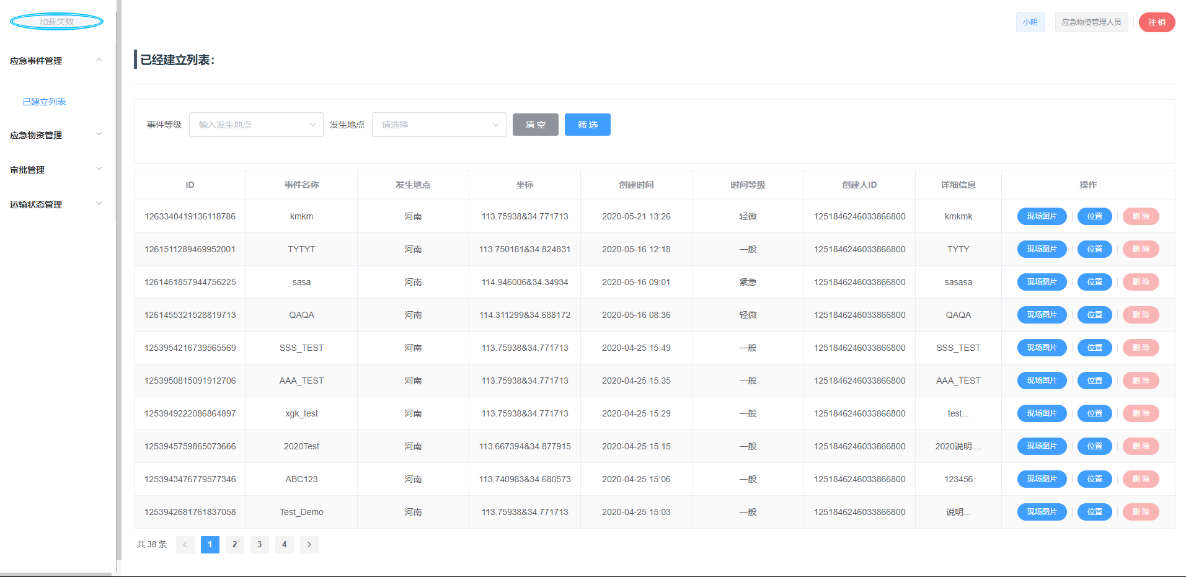


图6-1 应急物资管理员登录系统界面

#### 2. 应急响应人员

应急响应人员登录后可以实现查看已建立列表、新建应急事件、发起物资申请等功能，其中发起物资申请功能中包括创建用户ID、关联事件ID、指定审批人、申请标题、申请优先级（低\中\高\紧急）、申请内容和选择申请物资，点击提交保存填写。应急响应人员登录系统运行示例如图6-2所示。

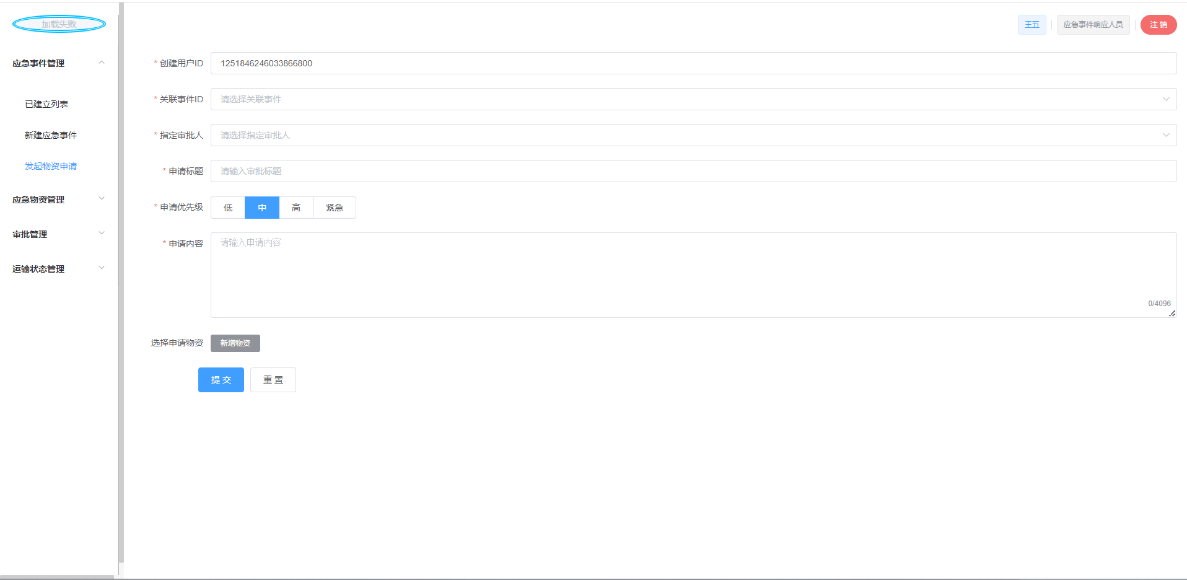


图6-2 应急响应人员登录系统界面

### 应急物资管理模块实现

#### 1. 应急物资管理员

应急物资管理员登录系统后可以实现新建应急物资分类、新增物资、应急物资分类列表、物资列表等功能。应急物资管理员登录系统运行示例如图6-3、6-4、6-5、6-6所示。

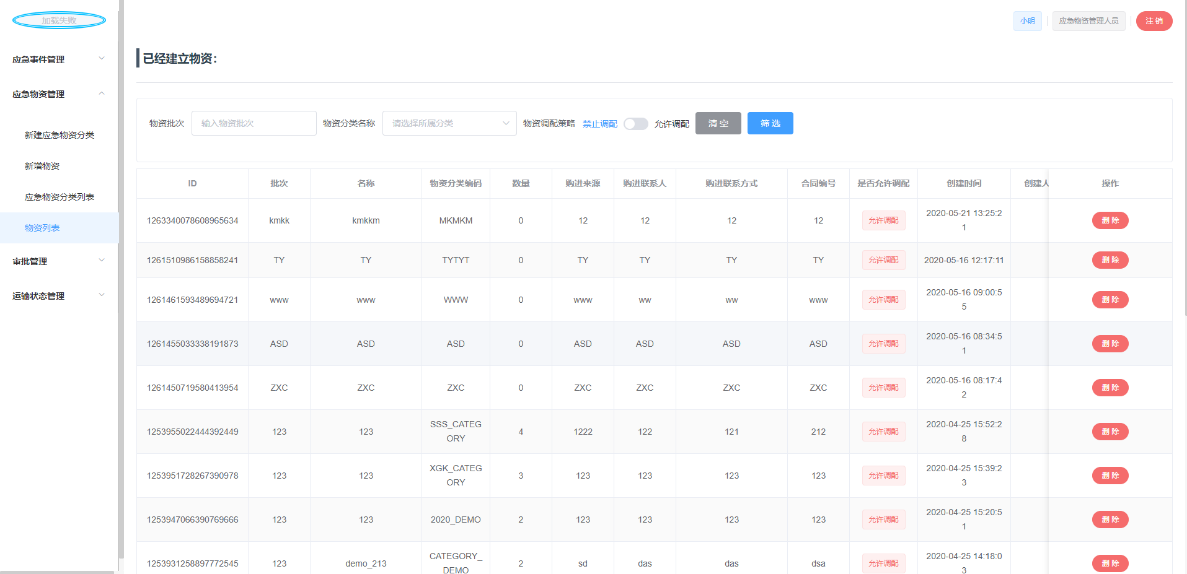


图6-3 应急物资管理员登录系统界面

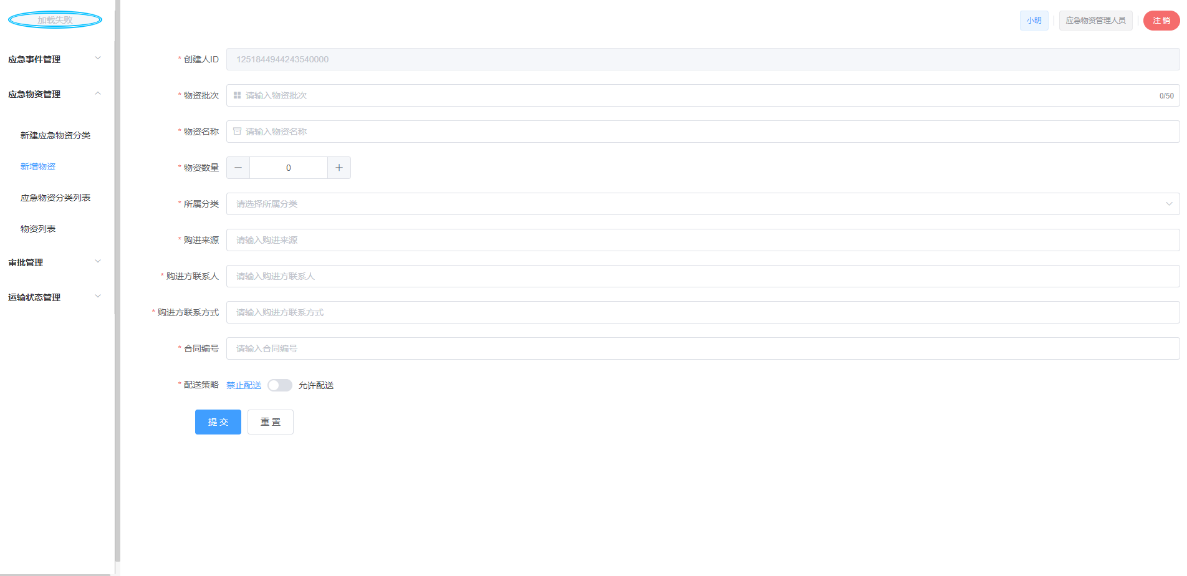


图6-4 应急物资管理员登录系统界面

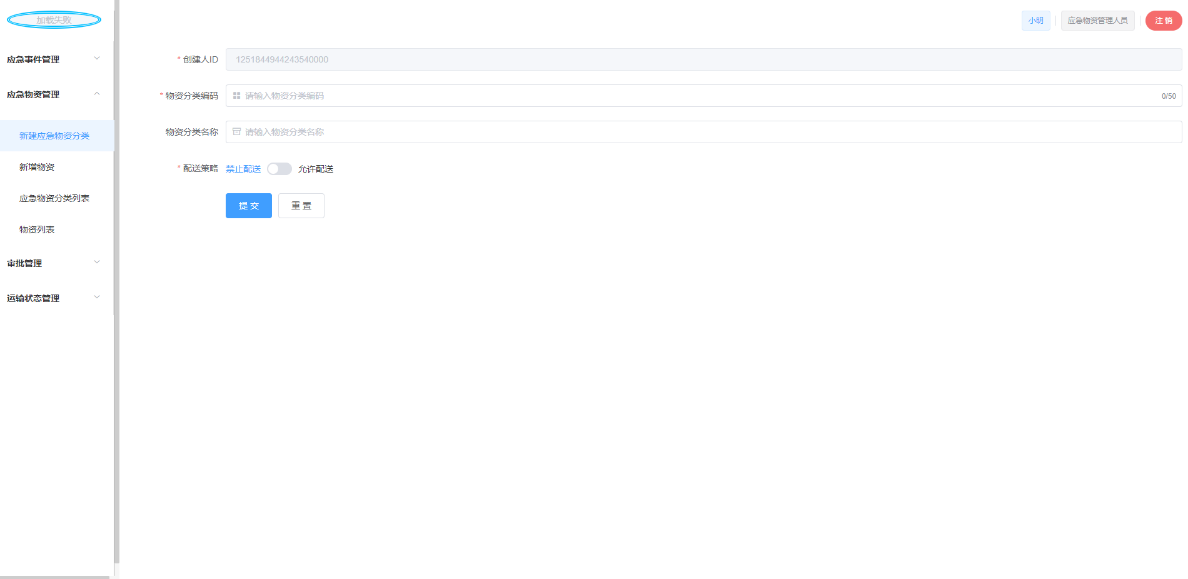


图6-5 应急物资管理员登录系统界面

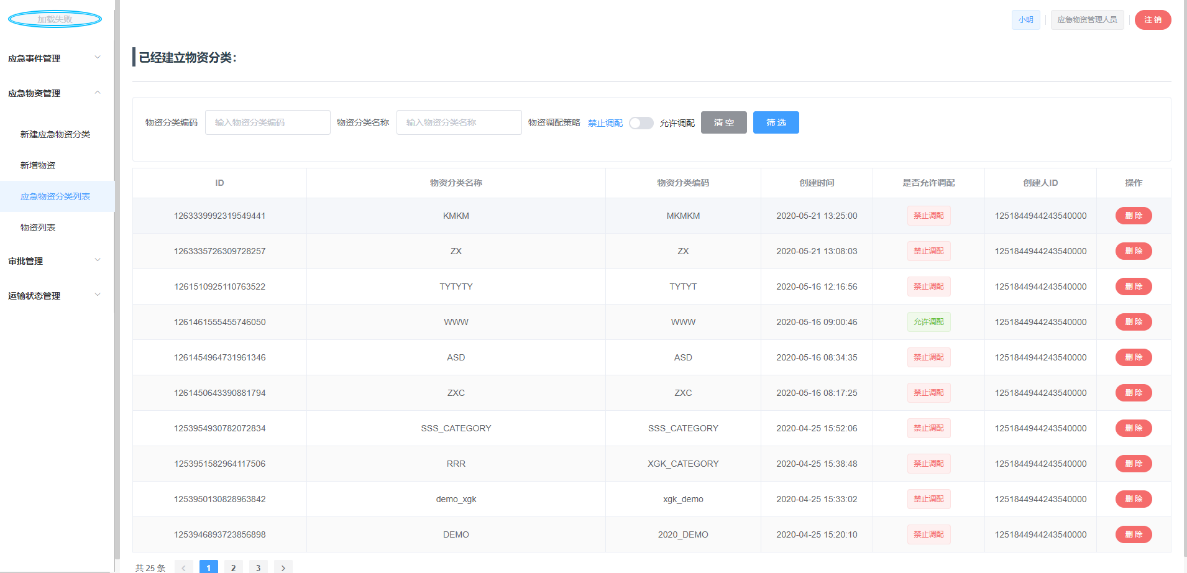


图6-6 应急物资管理员登录系统界面

#### 2. 运输人员

运输人员登录系统后权限相对较低，可以查看应急物资分类列表和物资列表。运输人员登录系统运行示例如图6-7所示。

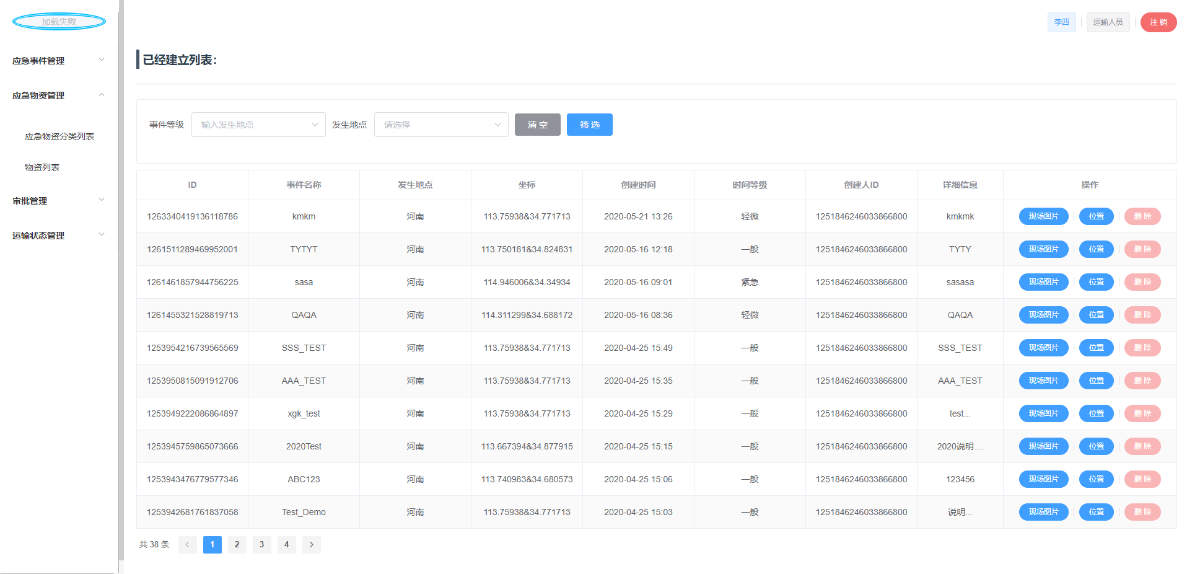


图6-7 运输人员登录系统界面

### 审批管理模块实现

#### 1.应急物资管理员

应急物资管理员登录系统后可以查看已建立申请列表，包括ID、申请标题、详细内容、创建时间、最后更新时间、申请等级、申请状态、关联审批人ID、关联事件ID、创建人ID等，可根据申请等级和申请状态进行模糊筛选查询，也可以进行查看物资和删除操作。应急物资管理员登录系统运行示例如图6-8所示。

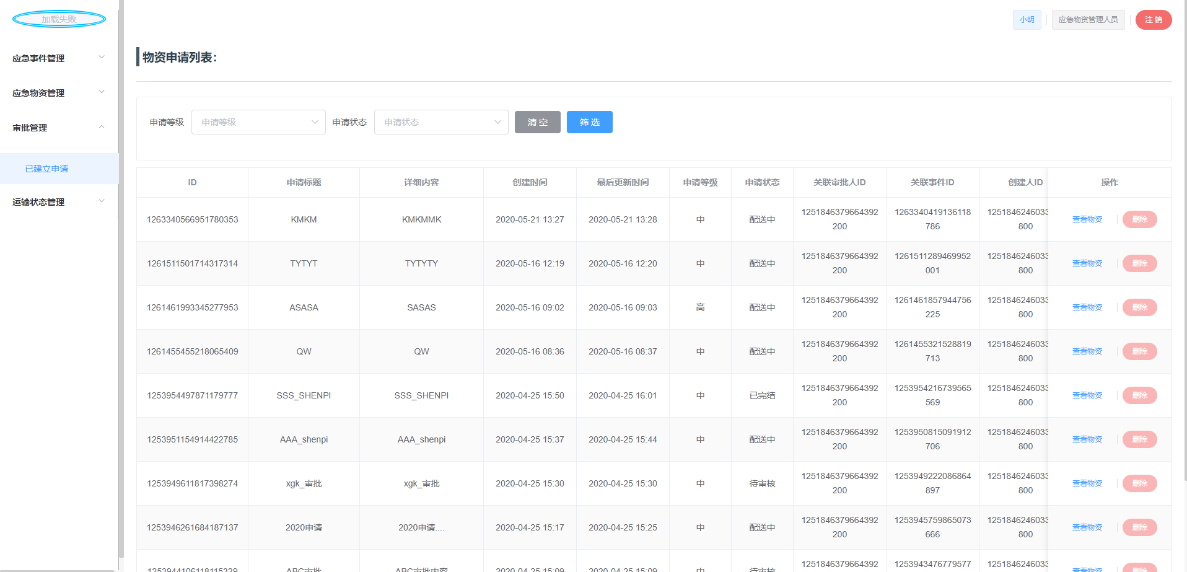


图6-8 应急物资管理员登录系统界面

#### 2.审批人员

审批人员登录系统后可以查看待审批列表和已建立申请列表，并根据申请等级和审批待处理人进行筛选申请。审批人员登录实现如图6-9所示。

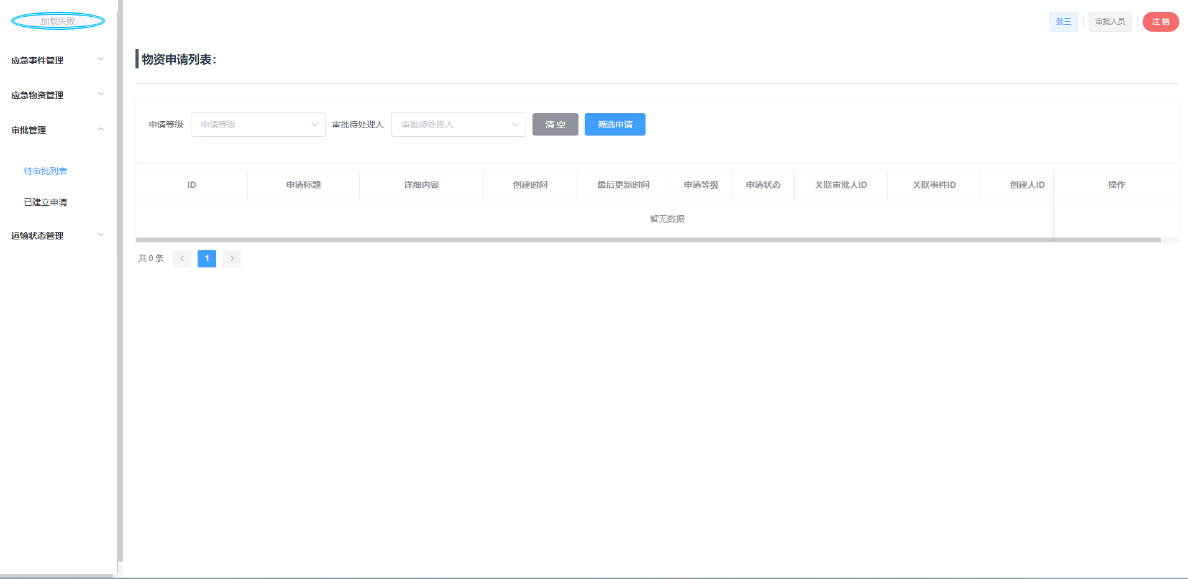


图6-9 审批人员登录界面

### 运输状态管理模块

#### 1.应急物资管理员

应急物资管理员登录系统后可以查看已建立配送单，包括ID、关联事件ID、创建时间、预期到达时间、创建人ID、申请状态、接收人ID、运输方式等，并可以进行查看物资、查看流转记录和结束流转功能，和根据接收人模糊筛选查询。应急物资管理员登录系统运行示例如图6-10所示。

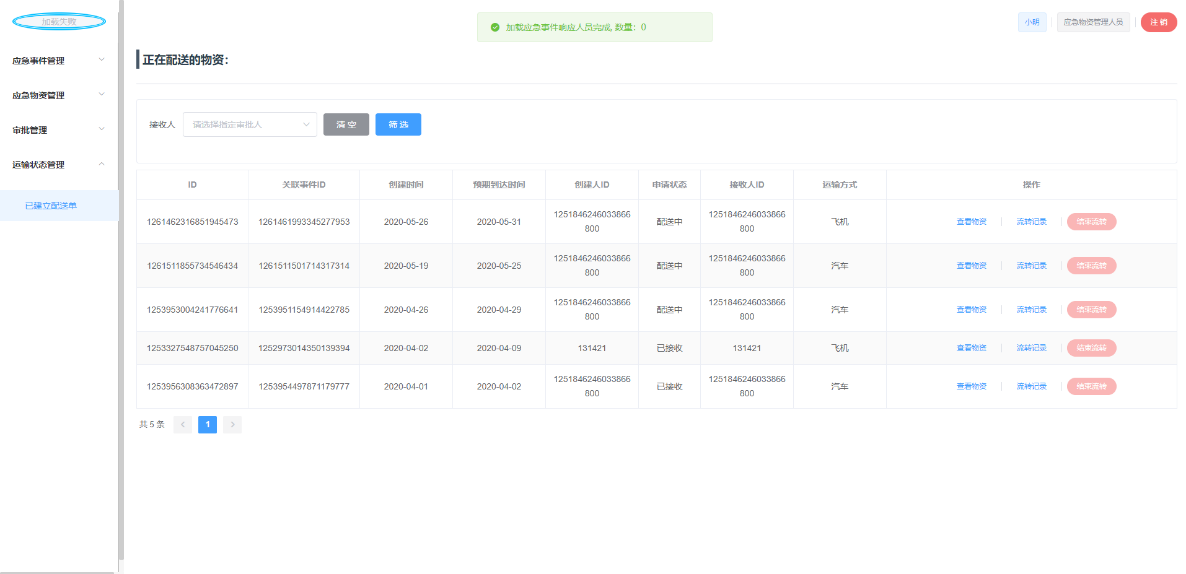


图6-10 应急物资管理员登录系统界面

#### 2.运输人员

运输人员登录系统后可以实现待配送、新建配送单和已建立配送单功能。其中新建配送单功能中包括创建人ID、关联审批标题、接收人ID、物资出库日期、运输方式（汽车\火车\轮船\飞机）和物资列表，点击提交保存填写。运输人员登录实现如图6-11所示。

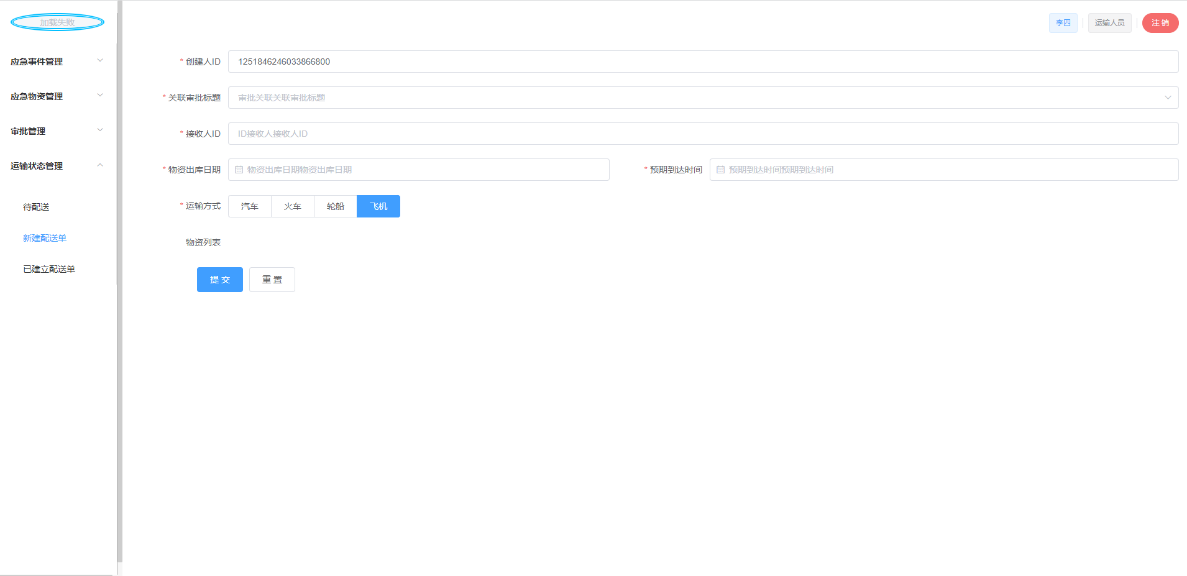


图6-11 运输人员登录界面

## 系统测试

### 软件测试的方法

在该系统的软件测试阶段，使用以下两种测试方法进行测试：一种是当测试人员已经了解了程序的功能时，可以设计测试用例来测试程序的每个功能是否符合逻辑，操作结果是否正确。另一个是，如果测试人员了解了程序的内部结构和操作流程，则他们可以测试程序的内部操作是否合乎逻辑以及结果的正确性。 第一种方法称为黑盒测试技术，第二种方法称为白盒测试技术。

1.黑盒测试技术：黑盒测试方法也称为功能测试方法。 顾名思义，黑盒测试技术将程序视为黑盒。 因此，此时的测试方法是向被测模块输入数据并记录输出数据。 根据输出数据结果并结合系统描述进行检查，从而确定被测模块是否存在问题。

2.白盒测试技术：白盒测试也称为结构测试。 顾名思义，白盒测试技术对程序是完全透明的。 也就是说，测试人员可以查看程序的内部，并了解其内部操作规则，算法和结构。 因此，此时的测试方法是根据程序内部的设计逻辑进行测试，以检查程序中的每个执行路径是否符合该逻辑以及结果是否正确。。

### 软件测试的主要用例

测试用例是在测试系统时执行的实际数据。系统主页和功能点的测试用例如下。

(1) 用户注册测试用例如表 6-1 所示。

表6-1 用户注册测试用例

|  |  |
| --- | --- |
| 用例名称 | 用户注册 |
| 目的 | 测试用户能否正常使用注册功能 |
| 前提 | 用户必须首次使用本系统 |
| 测试流程 | 1.用户输入用户名、姓名、密码 |
| 预期结果 | 当用户输入重复用户名时提示用户名重复 |
| 实际结果 | 实际结果与预期结果一致 |

(2) 用户登录测试用例如表 6-2 所示。

表6-2 用户登录测试用例

|  |  |
| --- | --- |
| 用例名称 | 用户登录 |
| 目的 | 测试用户能否正常使用登录功能 |
| 前提 | 用户已经注册过了 |
| 测试流程 | 1.用户输入正确用户名密码  2.用户输入正确用户名错误密码 |
| 预期结果 | 当用户输入正确用户名密码是可以成功登录系统  当用户输入正确用户名错误密码时提示密码错误 |
| 实际结果 | 实际结果与预期结果一致 |

(3) 退出登录测试用例如表 6-3所示。

表6-3 退出登录测试用例

|  |  |
| --- | --- |
| 用例名称 | 退出登录 |
| 目的 | 测试用户能否正常使用退出登录功能 |
| 前提 | 用户已经注册过了 |
| 测试流程 | 1.用户点击退出登录  2.点击确定退出 |
| 预期结果 | 成功退出系统并返回到系统登录页面 |
| 实际结果 | 实际结果与预期结果一致 |

(4) 审批管理用例测试如表6-4所示。

表6-4 审批管理测试用例

|  |  |
| --- | --- |
| 用例名称 | 审批管理 |
| 目的 | 测试用户能否正常使用审批管理功能 |
| 前提 | 成功登录系统 |
| 测试流程 | 1.修改审批信息  2.点击提交 |
| 预期结果 | 提交成功 |
| 实际结果 | 实际情况与预期结果一致 |

(6) 管理应急物资用例测试如表6-5所示。

表6-5管理应急物资测试用例

|  |  |
| --- | --- |
| 用例名称 | 管理应急物资 |
| 目的 | 测试管理员能否正常使用管理应急物资功能 |
| 前提 | 管理员成功登录 |
| 测试流程 | 1.点击修改  2.确认修改 |
| 预期结果 | 修改成功 |
| 实际结果 | 实际情况与预期结果一致 |

## 本章小结

本章主要介绍应急物资保障系统，通过对系统各个模块的详细测试，可以显示整个系统运行的过程和界面，方便用户更快地了解和使用该程序。通过对系统各种功能的测试，对系统的功能和性能进行了改进，对系统的一些缺陷和错误进行了改进和修改，系统运行测试后,对使用自动化管理系统使得相关业务能够更加信息化、科学化、高效化以及规范化。

结论

该系统的后台设计采用流行的Java语言和Spring Boot框架进行开发。 并使用Mybatis框架与数据库进行交互。前端使用流行的Vue框架。 同时，Vue使用vuex来控制前端的状态，包括组件的传递和变量通过路由的传递。整体实现了客户端：用户注册、用户登录、用户注销、用户管理、分析人员管理、测试数据分析、问卷调查。

该系统的优势在于它使用了前端和后端分离架构。与传统架构相比，后端访问的安全性和效率要快得多，因为Spring Boot使用http进行服务之间的调用，而前端使用流行的Vue框架。界面的舒适度和响应速度比Ajax快得多。通常，系统的安全性和界面的舒适性可以为用户带来更好的体验。该系统已运行并经过测试，结果令人满意。

致谢

随着时间的流逝，毕业季也已经到来，而我们的大学生活也将在这6月的夏天画上一个完美的句号。心中有很多感慨，舍不得我们大学的生活，大学的同学们、老师们。大学四年的酸甜苦辣也都历历在目，大一时期的懵懂，大二时期的“初生牛犊不怕虎”，大三时期的成熟感，以及现在即将步入社会的迷茫。此时此刻的心情由于五味陈杂，百感交集，但更多的还是感激之情。

经过一年的不懈努力，毕业设计终于完成了，在这里，我首先要感谢我的老师们，谢谢你们在我做毕业设计的时候给予我的各种帮助，而且教会了我写毕业论的一些方式方法，而且在毕业设计方面，老师渊博的学识给予了我很大的帮助，有些问题让我百思不得其解的时候，是老师的一句话让犹如在黑夜中找到了一盏明灯，一语惊醒梦中人的感觉！老师有很好的耐心，无论我犯了多么低级的错误，老师仍然会耐心的为我指导，有时候钻牛角尖，老师也会耐心的提示我，让我走出歧途，再次真心的感谢老师！然后我要感谢我亲爱的同学们，在我有困难的时候，他们无论多忙，无论是在忙着学习还是在忙着做毕业设计，都会抽出时间来为我解答我的疑惑。不仅如此，同学们还给我了很多有关于我的项目的资料，文本的、图片的，很多很多。当我心情低落的时候，也是同学们在鼓舞我，让我重新打起精神，和我一起战胜当前的困难，一步一步走到现在，真心的感谢你们，亲爱的同学们！最后我要感谢我的父母，谢谢你们对我生活、精神上的支持，我爱你们！

参考文献

1. 张明森，黄宏敏，詹瑞典．基于Spring Cloud微服务架构的广告系统的设计与实现[J]．电子世界．2020(08)：165-166．
2. 吴旭君．基于Spring Cloud微服务架构的数据服务系统的实践[D]．计算机产品与流通，2020 (03)：93．
3. 吕太之，蒋玉婷，游学军．基于Spring Cloud的面试题库系统设计与实现[J]．电脑知识与技术，2019，15(33)：30-32．
4. 刘金羽．前后端分离的在线考试系统设计与实现[J]．电脑编程技巧与维护．2020(04)：44-46．
5. 孟祥双．前后端分离式WEB应用开发研究[J]．电子元器件与信息技术．2019，3(06)：40-43．
6. 刘金羽．基于Vue.js的前端教学软件设计与实现[J]．电脑编程技巧与维护．2020(02)：23-24．
7. 师明，曾丹．基于Vue.js和Spring Boot的校招日记系统[J]．工业控制计算机．2020，33(01)：95-97．
8. 柴青山．基于MVVM模式的Vue.js框架在物流软件自动化测试系统中的应用研究[D]．北京邮电大学，2019．
9. 陈倩怡，何军．Vue+Springboot+MyBatis技术应用解析[J]．电脑编程技巧与维护．2020(01)：14-15．
10. 李桂林．HTML5在WEB前端开发中的应用研究[J]．计算机产品与流通．2020(08)：17．
11. 徐晓．基于CSS3技术的动画案例浅析[J]．现代计算机．2020(01)：93-96．
12. 李娜．基于Spring Cloud微服务架构的应用[J]．电子技术与软件工程．2019(12)：142．
13. Pengxiang Zhou，Shaohong Xia. Constraints from b-values in the Andaman–Sumatra–Java subduction zone [J]．Atencion Primaria，2020，03(04)：105-110．
14. Stewart，Alan．Reasoning about orchestrations of web services using partial correctness [J]．Keenan．2020，52(3)：79-83．
15. Rudolf Ammann．Recovering the web’s unclaimed legacy of academic text standards: HTML, and the misremediation of quotation[J]．Internet Histories，2020，4(1)：23-27．

附录

英文原文

**Reasoning About Orchestrations of Web Services Using Partial Correctness**

Abstract. A service is a remote computational facility which is made available for general use by means of a wide-area network. Several types of service arise in practice: stateless services, shared state services and services with states which are customised for individual users. A service-based orchestration is a multi-threaded compu- tation which invokes remote services in order to deliver results back to a user (publication). In this paper a means of specifying services and reasoning about the correctness of orchestrations over stateless services is presented. As web services are potentially unreliable the termination of even finite orchestrations cannot be guaranteed. For this reason a partial-correctness powerdomain approach is proposed to capture the semantics of recursive orchestrations.

1.Introduction

In a web environment developers can create applications by orchestrating a set of remote, distributed services. Internet-based software can be used to discover, organise and invoke remote services in order to carry out pre- specified computations. For example, services are used in the cloud computational model to acquire both software and hardware resources. In [SGC06] it is argued that Orc [Mis04, MC07] provides appropriate combinators for constructing service-based computations. For example, Orc provides operators that can be used to time-out non-responsive services and reschedule computations elsewhere. In this paper a predicate-based [BvW, Heh84a, Hoa85] definition of Orc is given.

Conventionally, a specification S is refined into a composition P ; Q (or P I Q ) in such a way that correctness is preserved. An abstract specification may be transformed into code through a process of repeated refinement. In contrast orchestrations are constructed by assembling predefined services. One motivation for giving a predicate semantics to Orc is to enable the correctness of bottom-up designed orchestrations to be addressed.

Services are used as a basis for constructing a semantics for Orc in much the same way that assignment is used to provide a basis for an axiomatic semantics of an imperative programming language [Hoa69]. For exam- ple, a semantics for a conventional programming language can be derived by first defining assignment and then defining the language’s compositional and recursive operators. In a conventional setting an assignment x :∗ e is guaranteed to terminate provided that the expression e is well-defined. Thus, (under the well-defined expression limitation above) a guarantee of program termination follows from proving that all recursive/iterative operators are finite (total correctness).

There is a fundamental distinction between a semantics of a standard programming language and a semantics of Orc; a conventional assignment (within the limitations mentioned above) terminates whereas a service call is, by its very nature, unreliable (and potentially divergent). A service is a software item (whose total correctness may have been established) which is made available in a web environment. The response behaviour of a service may vary, depending on the level of demand. If a large number of requests are made to a service S , within a given time interval, then the underlying hosting infrastructure is placed under stress. If demand exceeds the capacity of the hosting infrastructure then the response time of S may be degraded, perhaps to the point of failure. In practice, orchestrations may call a number of (redundant) services having identical functionalities in order to provide a degree of robustness in the event of service failure. Even a simple non-recursive service call is potentially non- terminating; consequently, a partial-correctness setting is used to define the semantics of Orc expressions. For example, the semantics of a two-thread system which calls a BBC service and a CNN service in parallel must cater for the possibilities that both services are working, that exactly one service is operational or that both of the services are broken. The use of the lower powerdomain to define an orchestration captures this set of possible service behaviours.

Predefined Orc services can be specified as predicates. The approaches of [BvW, Heh84a, Heh84b, Hoa85, HH98] can be modified to provide a semantics for services and orchestrations. Three service variations are con- sidered: stateless, shared state and private state. However, the reasoning framework presented here is designed primarily for orchestrations over stateless services. Such orchestrations arise in practice—e.g. certain kinds of computationally intensive scientific applications may be implemented as orchestrations which utilise (stateless)

linear algebra services. Orc provides three combinators which can be used as glue for assembling pre-specified ser- vices into composite programs: | denotes parallel composition, 》 denotes sequential composition and 《 denotes pruning composition. An orchestration publishes a bag of result values, θ : an orchestration with publication bag θ ∗ {a, b} can informally be identified with a two-threaded computation, one thread publishing a and the other b. The aim of this paper is to expose the nature of orchestration operators and to provide a formal semantics for reasoning about service-based computations.

Several other approaches have been used to give a semantics to Orc [DLSZ06, LZH10, HMM04, NS10, KCM06, VW08, WKCM08]: in both [LZH10] and [NS10] trace-based approaches are used to give a semantics to orchestrations. The trace approach of [LZH10] utilises a total-correctness UTP framework and has similarities with the approach taken in this paper. In [HMM04] orchestrations are mapped onto evaluation trees and the combinators of Orc are defined as tree operations. The work reported here differs in that it provides a mechanism for reasoning about orchestrations over concrete (pre-specified) services. Additionally, various operational and denotational semantics for Orc have been developed [KCM06, VW08, WKCM08]. An operational semantics provides details of service interactions that occur during an orchestration evaluation. In contrast, a predicate- based approach is concerned mainly with the values published by an orchestration (“what” rather than “how”). Although operational, denotational and predicate approaches have different motivations, the techniques have many similarities: for example, the mechanism for instantiating variables that is used in this paper is essen- tially that used in [WKCM08]. The predicate-based semantics given here offers a direct style of reasoning about orchestration input/output behaviour. In addition, a semantics for recursive orchestrations is developed.

A predicate-based framework for specifying services is presented in Sect. 2. Informal and formal (stateless) semantics of Orc are given in Sects. 3 and 4, respectively. Powerdomains are used to derive a semantics of recursive Orc expressions in Sect. 5. In Sect. 6 some suggestions for future work are outlined.

2.Services

A service is a remote, potentially unreliable, computational facility. Services can be stateless or stateful—for example, the service call cloud execute(C , d ) returns the result of remotely executing code C on data d while the service database DBLP utilises a dynamic (state-based) database. A service is called asynchronously from a computational thread. At some later time the service may publish a response [WKCM08]. Service calls may be unsuccessful for a number of reasons: data supplied in a service call may be inappropriate, a service-based computation may diverge, a service may not have the capacity to process simultaneously the current volume of calls, or communications may be “lost” on faulty networks.

Services are specified using predicates (in much the same way that programs are in both UTP [HH98] and predicative programming [Heh84a, Heh84b]). In the remainder of this paper a bag or multiset is denoted usingconventional set notation—{a, a} denotes a bag with two occurrences of element a. The alphabet of a predicate P, αP, comprises the set of variables that occur free in P . The alphabet αP is partitioned into input variables,inαP, and the distinguished output variable θ .

3.Orc

The operators of Orc act as a glue for cementing service calls together. Parallel composition (|) is used to evaluate multiple threads simultaneously, sequential composition (》) can be used to sequence single or multiple threads and pruning composition (《) can be used to non-deterministically select one publication from a set of threads. An otherwise operator (|>) has been proposed recently [Kit09, LZH10] which activates a second orchestration in the event that a first orchestration terminates without publishing. The formal syntax of Orc expressions [Mis04] is given in Fig. 2. An informal description of each of the operators is given below.

4.Orchestration semantics for stateless services

In this section a means of reasoning about orchestrations over stateless services is described. To this end predicate- based definitions of the Orc combinators are given—this allows an Orc expression over pre-defined services to be equated with a relation from inputs to publication bags. For example, an orchestration which is identified with the predicate θ ∗ {a}∨ θ ∗ {a, c} can publish either a or an interleaving of a and c.

The parallel composition of P and Q forms the bag union of the publication bags of P and Q . In the defini- tion of P |Q the publications of (i) P (ii) Q and (iii) P |Q are distinguished as follows: P |Q publishes θ while the publication bags of P and Q are renamed r1 and r2, respectively. This renaming of component publication bags is typical of the remaining semantic definitions.

The sequential composition P > x > Q spawns instantiations Qx for each value v in the publication bag of P —in effect P > x > Q constructs a set of sequenced threads. In the definition of P > x > Q instantiation is generalised from values to bags of values: if b is a publication bag then Qx denotes a parallel composition of conventional (single-valued) instantiations |y∈b Qx .

Evaluation of Q < x < P proceeds by executing P and Q in parallel (the evaluation of some threads within Q may need to be delayed until a binding for x is generated by P ): the first output published by P is used to instantiate x ; at this point evaluation of P is terminated and evaluation of Q is completed. If P diverges then evaluation of Q < x < P evaluates those threads of Q that are independent of x , viz Q ↓x ; otherwise Q < x < P behaves as Q instantiated by some element taken from the publication bag of P . One essential difference between the operators 》 and 《 is that sequential composition uses bag instantiation whereas pruning composition non-deterministically chooses an element from a bag before using conventional single-value instantiation.

5.Recursive declarations

A declaration is a binding of a name to an orchestration expression: the declaration d (x ) ∗df E binds the name d and the formal parameter x to the expression E . If d /∈ αE then declaration d is non-recursive and so d (a) ∗ Ex . Otherwise, d has the recursive form d (x ) ∗ E (d ). In this case a semantics is given to d by treating d (x ) ∗ E (d ) as a fixed-point equation [HH98]. Here the lower powerdomain (see below) is used as a basis for generating a chain of improving approximations [Pi | i ≥ 0] to d . Each predicate Pi is identified with a relation from inputs to publications. The meaning of d is the least upper bound of the approximation sequence [Pi | i ≥ 0].

6.Discussion

In a web environment users can create applications by orchestrating a set of remote, distributed services. In this paper a basis for reasoning about orchestrations over concrete, stateless services has been proposed. The semantic framework includes a set of straightforward rules for reasoning about finite orchestrations. Partial correctness arguments are used to reason about service orchestrations in unreliable environments where failure might occur.An equivalence class ≡l is employed to classify all orchestration outcomes that might arise when web-services fail. The class contains an angelic evaluation outcome, where all services are working, a daemonic evaluation outcome, where the web-environment is entirely broken, as well as a variety of less extreme scenarios. Orc decla- rations can be complex and may involve recursive definitions which spawn parallel threads in addition to more conventional compositional operators. It has been shown here that the operators of Orc are monotonic on the lower powerdomain and so recursive declarations can be given a fixed-point semantics.

Acknowledgments

The authors are grateful to Maurice Clint and the referees for their comments on earlier drafts of this paper. Alan Stewart was partially supported by EPSRC project EP/I03405X/1 (ECHO). Joaquim Gabarro was partially sup- ported by FET pro-active Integrated Project 15964 (AEOLUS), the projects TIN2007-66523 (FORMALISM) and TIN2005-25859-E of “Ministerio de Ciencia e Inovacio´ n y el Fondo Europeo de Desarrollo Regional”.

中文原文

使用部分正确性对Web服务的编排进行推理

摘要. 服务是一种远程计算工具，可通过广域网将其用于一般用途。 在实践中出现了几种类型的服务：无状态服务，共享状态服务以及具有针对单个用户定制的状态的服务。 基于服务的编排是一种多线程计算，它调用远程服务以将结果传递回用户（发布）。 在本文中，提出了一种指定服务的方法以及关于无状态服务的业务流程正确性的推理方法。 由于Web服务可能不可靠，因此即使有限的编排也无法保证终止。 因此，提出了一种部分正确性幂域方法来捕获递归业务流程的语义。

1.简介

在Web环境中，开发人员可以通过编排一组远程分布式服务来创建应用程序。基于Internet的软件可用于发现，组织和调用远程服务，以执行预先指定的计算。例如，在云计算模型中使用服务来获取软件和硬件资源。 [SGC06]认为Orc [Mis04，MC07]提供了适当的组合器，用于构造基于服务的计算。例如，Orc提供的运算符可用于使无响应服务超时并在其他地方重新安排计算时间。本文给出了基于谓词的Orc的[BvW，Heh84a，Hoa85]定义。

常规上，规格S被细化为成分P。 Q（或P I Q）以保持正确性的方式。抽象规范可以通过反复细化的过程转换为代码。相反，业务流程是通过组合预定义的服务来构建的。向Orc赋予谓语语义的一种动机是使自下而上的业务流程的正确性得以解决。

服务被用作为Orc构造语义的基础，与分配被用作为命令式编程语言的公理语义提供基础的方式几乎相同[Hoa69]。例如，可以通过首先定义赋值，然后定义该语言的组合和递归运算符来导出常规编程语言的语义。在常规设置中，只要表达式e定义明确，就可以保证x：\* e终止。因此，（在上面定义明确的表达式限制下）通过证明所有递归/迭代运算符都是有限的（总正确性）可以保证程序终止。

标准编程语言的语义和Orc的语义之间存在根本的区别。常规分配（在上述限制内）终止，而服务调用从其本质上来说是不可靠的（并且可能有分歧）。服务是指在网络环境中可用的软件项目（可能已经确定其完全正确性）。服务的响应行为可能会有所不同，具体取决于需求级别。如果在给定的时间间隔内对服务S提出了大量请求，则基础托管基础结构将承受压力。如果需求超出托管基础架构的容量，则S的响应时间可能会降低，甚至可能会失败。实际上，业务流程可以调用具有相同功能的许多（冗余）服务，以便在服务失败的情况下提供一定程度的健壮性。即使是简单的非递归服务调用也可能会终止。因此，使用部分正确性设置来定义Orc表达式的语义。例如，并行调用BBC服务和CNN服务的双线程系统的语义必须满足两个服务都在工作，一个服务恰好在运行或者两个服务都被破坏的可能性。使用较低的powerdomain定义业务流程可以捕获这组可能的服务行为。

可以将预定义的Orc服务指定为谓词。可以修改[BvW，Heh84a，Heh84b，Hoa85，HH98]的方法，以为服务和业务流程提供语义。考虑了三种服务变体：无状态，共享状态和私有状态。但是，这里介绍的推理框架主要是针对无状态服务的业务流程而设计的。这种编排实际上是在实践中出现的。某些类型的计算密集型科学应用可以实现为利用（无状态）的业务流程线性代数服务。 Orc提供了三种组合器，它们可以用作将预先指定的服务组装到复合程序中的胶水：表示平行组成，》表示顺序组成，《》表示修剪组成。编排发布一袋结果值θ：编排带有发布袋θ∗ {a，b}的编​​排可以通过两线程计算来非正式地标识，一个线程发布a，另一个线程发布b。本文的目的是揭示业务流程运算符的性质，并为推理基于服务的计算提供形式语义。

已使用其他几种方法为Orc赋予语义：[LZH10]和[NS10]基于跟踪的方法都为Orc提供了语义。 [LZH10]的跟踪方法使用了完全正确的UTP框架，并且与本文采用的方法相似。在[HMM04]中，业务流程被映射到评估树上，并且Orc的组合符被定义为树操作。此处报告的工作有所不同，因为它提供了一种针对特定（预定）服务进行业务流程推理的机制。此外，还开发了Orc的各种操作和指称语义[KCM06，VW08，WKCM08]。操作语义提供了业务流程评估期间发生的服务交互的详细信息。相反，基于谓词的方法主要涉及业务流程发布的值（“什么”而不是“如何”）。尽管操作，确定性和谓词方法具有不同的动机，但这些技术有许多相似之处：例如，本文中使用的实例化变量的机制本质上是[WKCM08]中使用的机制。此处给出的基于谓词的语义为编排输入/输出行为提供了一种直接的推理方式。此外，还开发了用于递归业务流程的语义。

2.服务

服务是一种远程的，可能不可靠的计算工具。 服务可以是无状态的，也可以是有状态的，例如，服务调用云execute（C，d）返回在数据d上远程执行代码C的结果，而服务数据库DBLP使用动态（基于状态的）数据库。 从计算线程异步调用服务。 稍后，服务可能会发布响应[WKCM08]。 服务呼叫可能由于以下几个原因而失败：服务呼叫中提供的数据可能不合适，基于服务的计算可能会有所不同，服务可能没有能力同时处理当前呼叫量，或者通信可能是“ 在故障网络上丢失”

服务是使用谓词指定的（与UTP [HH98]和谓词编程[Heh84a，Heh84b]中的程序几乎相同）。 在本文的其余部分，使用以下方式表示袋子或多件套常规设置符号-{a，a}表示出现了两次元素a的袋子。 谓词P的字母αP包含在P中自由出现的变量集。 字母αP被分为输入变量inP和可分辨的输出变量θ。

3.Orc

Orc的运营商充当了固井服务呼叫的粘合剂。 并行组合（|）用于同时评估多个线程，顺序组合（》）可用于对单个或多个线程进行排序，修剪组合（《）》可用于从一组线程中不确定地选择一个发布。 最近，有人提出了一种其他方式的运算符（|>）[Kit09，LZH10]，如果第一个编排终止而没有发布，则该运算符将激活第二个编排。

4.无状态服务的编排语义

在本节中，将介绍有关无状态服务的业务流程推理的方法。为此，给出了Orc组合器的基于谓词的定义，这使Orc表达式对预定义服务的表达等同于从输入到发布袋的关系。例如，用谓词θ{{a}∨θ{{a，c}标识的业务流程可以发布a或a和c的交错。

P和Q的平行组成形成P和Q的出版袋的袋结合。在对P | Q的定义中，对（i）P（ii）Q和（iii）P | Q的发布进行了如下区分：P | Q发布θ，而P和Q的发布袋分别重命名为r1和r2。 ， 分别。组件发布袋的这种重命名是其余语义定义的典型代表。

 顺序组成P> x> Q会为P的发布袋中的每个值v生成实例化Qx-实际上P> x> Q构造了一组序列化线程。在P> x> Q的定义中，实例化从值推广到值的袋：如果b是发布袋，则Qx表示常规（单值）实例|y∈bQx的并行组成。

5.递归声明

声明是名称与业务流程表达式的绑定：声明d（x）\* df E将名称d和形式参数x绑定到表达式E。 如果d /∈αE，则声明d是非递归的，因此d（a）∗ Ex。 否则，d具有递归形式d（x）\* E（d）。 在这种情况下，通过将d（x）∗ E（d）作为不动点方程[HH98]来赋予d语义。 此处，较低的功率域（请参见下文）用作生成改进近似值链的基础。 i≥0]至d。 每个谓词Pi都具有从输入到发布的关系。 d的含义是近似序列[Pi | i≥0]。

6.讨论

在Web环境中，用户可以通过编排一组远程分布式服务来创建应用程序。在本文中，提出了针对具体的无状态服务进行业务流程推理的基础。语义框架包括一组用于推理有限业务流程的简单规则。部分正确性参数用于推断可能发生故障的不可靠环境中的服务编排。等价类≡l用于对Web服务失败时可能出现的所有编排结果进行分类。该类包含一个天使评估结果（其中所有服务都在工作），守护进程评估结果（其中Web环境被完全破坏）以及各种不太极端的场景。 Orc声明可能很复杂，并且可能涉及递归定义，除了更常规的合成运算符外，这些递归定义还产生并行线程。此处已显示Orc的运算符在较低幂域上是单调的，因此可以为递归声明指定定点语义。

致谢

作者感谢Maurice Clint和裁判对本文早期草案的评论。 艾伦·斯图尔特（Alan Stewart）得到了EPSRC项目EP / I03405X / 1（ECHO）的部分支持。 Joaquim Gabarro部分获得了FET积极的集成项目15964（AEOLUS），TIN2007-66523（FORMALISM）和TIN2005-25859-E的“ Ministerio de Ciencia e Inovacio´ y el Fondo Europeo de Desarrollo Regional”项目的支持。