****



**研 究 生 毕 业 论 文**

**（申请工程硕士学位）**

|  |  |
| --- | --- |
| **论文题目** | 饿了么体验保障平台的  设计与实现 |
| **作者姓名** | 金思晔 |
| **学科、专业名称** | 工程硕士(软件工程方向) |
| **研究方向** | 软件工程 |
| **指导教师** | 王金庆 高级工程师 |

**2019 年 4 月 日**

**学 号： MF1832073**

**论文答辩日期： 2020 年 月 日**

**指 导 教 师： （签字）**

**饿了么体验保障平台的**

**设计与实现**

|  |  |
| --- | --- |
| **作 者:** | **金思晔** |
| 指导教师: | **王金庆　高级工程师** |

|  |
| --- |
| **南京大学研究生毕业论文** |
| **(申请工程硕士学位)** |

|  |
| --- |
| **南京大学软件学院** |
| **2019年4月** |

**The Design and Implementation of**

**ELEME Experience Guarantee Platform**

**Jin, Si Ye**

**Submitted in partial fulfillment of the requirements for the degree of Master of Engineering**

Supervised by

Senior Engineer **Wang, Jinqing**

Software Institute

**NANJING UNIVERSITY**

Nanjing, China

April, 2019

# 摘 要

# Abstract

# 图目录

# 表目录

目录

[摘 要 V](#_Toc25947911)

[Abstract VI](#_Toc25947912)

[第一章 引言 8](#_Toc25947913)

[1.1 项目背景 8](#_Toc25947914)

[1.2 国内外发展现状 9](#_Toc25947915)

[1.3 本文主要工作 9](#_Toc25947916)

[1.4 本文的组织结构 9](#_Toc25947917)

[第二章 技术综述 9](#_Toc25947918)

[2.1 SpringBoot 9](#_Toc25947919)

[2.2 MaxQ 9](#_Toc25947920)

[2.3 Huskar 10](#_Toc25947921)

[2.4 WorkFlow 10](#_Toc25947922)

[2.5 Redis 10](#_Toc25947923)

[2.6 本章小结 10](#_Toc25947924)

[第三章 系统分析与设计 11](#_Toc25947925)

[3.1 系统总体规划 11](#_Toc25947926)

[3.2 系统需求分析 11](#_Toc25947927)

[3.2.1 规则管理模块 11](#_Toc25947928)

[3.2.2 审核流模块 12](#_Toc25947929)

[3.2.3 业务处理模块 13](#_Toc25947930)

[3.3 非功能性需求 15](#_Toc25947931)

[3.5 数据库设计 15](#_Toc25947932)

[3.6 本章小结 15](#_Toc25947933)

[第四章 系统实现 15](#_Toc25947934)

[4.1 规则配置模块 15](#_Toc25947935)

[4.2 审核流模块 16](#_Toc25947936)

[4.3 业务处理模块 17](#_Toc25947937)

[4.5 系统测试 17](#_Toc25947938)

[4.5.1 测试环境 17](#_Toc25947939)

[4.5.2 测试设计 17](#_Toc25947940)

[4.5.3 测试结果 17](#_Toc25947941)

[第五章 总结与展望 17](#_Toc25947942)

[5.1 总结 17](#_Toc25947943)

[5.2 工作展望 17](#_Toc25947944)

[参 考 文 献 18](#_Toc25947945)

[致 谢 18](#_Toc25947946)

# 第一章 引言

## 1.1 项目背景

论文选题源自于本人实习时参与的“饿了么热斧平台”。

超时是互联网中经常发生的场景，在网上购物，嘀嘀打车，点外卖等场景都是可能发生超时的。对于不同的超时场景，有不同的赔付策略，例如滴滴打车，司机没有在规定时间内到达指定地点，用户可以无责取消这笔订单；用户点外卖后，骑手没有在规定时间内将外卖送到指定地点，那么用户就会收到系统赔付的优惠券等。

饿了么拥有一个超时中心，对于每一笔订单，超时中心都会对其进行判断是否超时，而判断的标准就是运营人员制定的超时规则。当一笔订单被判定为超时的时候，系统就会执行相应的操作，例如赔付，发送消息通知等。

随着市场的不断发展，产品的不断迭代，当前订单超时场景中，除了物流/骑手本身原因导致订单履约超时外，还涉及到商户以及用户的原因。例如：商户没能够即时准备好餐品导致超时、商户原因中断配送导致超时、用户联系不上导致超时等。除此之外，命中超时规则之后的赔付操作，在不同的地区也会有不同的规则，例如对于北上广深等一线以及准一线城市，赔付的标准与三四线城市的标准会有差异。甚至在不同的天气下，赔付的标准都会不同。超时的场景越来越丰富，赔付的标准越来越精细化，这就导致了运营人员需要及时的新增、变更超时规则。而当前超时规则的判断是硬编码在超时中心中，对于每一次的规则变更，都需要经过需求评审、技术方案的编写、技术方案的评审、代码的编写、代码的测试以及产品的发布上线。这当中会耗费很多人力，更重要的是，运营人员的需求不能够得到及时的响应。因此，开发一个能够应对多样性规则，并且能够即时响应需求的系统很有必要。

为了解决上述问题，本文设计和实现了“饿了么热斧系统”，系统通过网页配置规则的方式，解决了规则多样性的问题；通过表达式引擎，可以执行配置完成的规则，能够及时响应需求的变更。

## 1.2 国内外发展现状

规则的可配置化多应用于业务逻辑多变的情况。

网易考拉规则引擎平台是网易旗下的一款规则引擎系统，主要的业务对接方为敏感日志操作判定规则以及业务风控中风控判定规则。系统分为四个层面服务层、引擎层、计算层和存储层。服务层负责接入服务，引擎层主要负责执行规则，计算层负责指标计算和模型训练，存储层负责存储规则、规则执行结果、指标等。其中规则引擎是平台的核心，平台使用了groovy脚本进行规则引擎的开发。

同程艺龙机票自研实时规则引擎，基于groovy实现

阿里巴巴财务平台业务方接入平台，基于groovy实现

美团酒店实时规则引擎，基于aviator实现

饿了么营销平台，基于aviator实现。

## 1.3 本文主要工作

## 1.4 本文的组织结构

# 第二章 技术综述

## 2.1 SpringBoot

Web框架

## 2.2 MaxQ

消息队列

## 2.3 Huskar

配置中心

## 2.4 WorkFlow

任务调度中心

## 2.5 Redis

缓存

## 2.6 本章小结

##

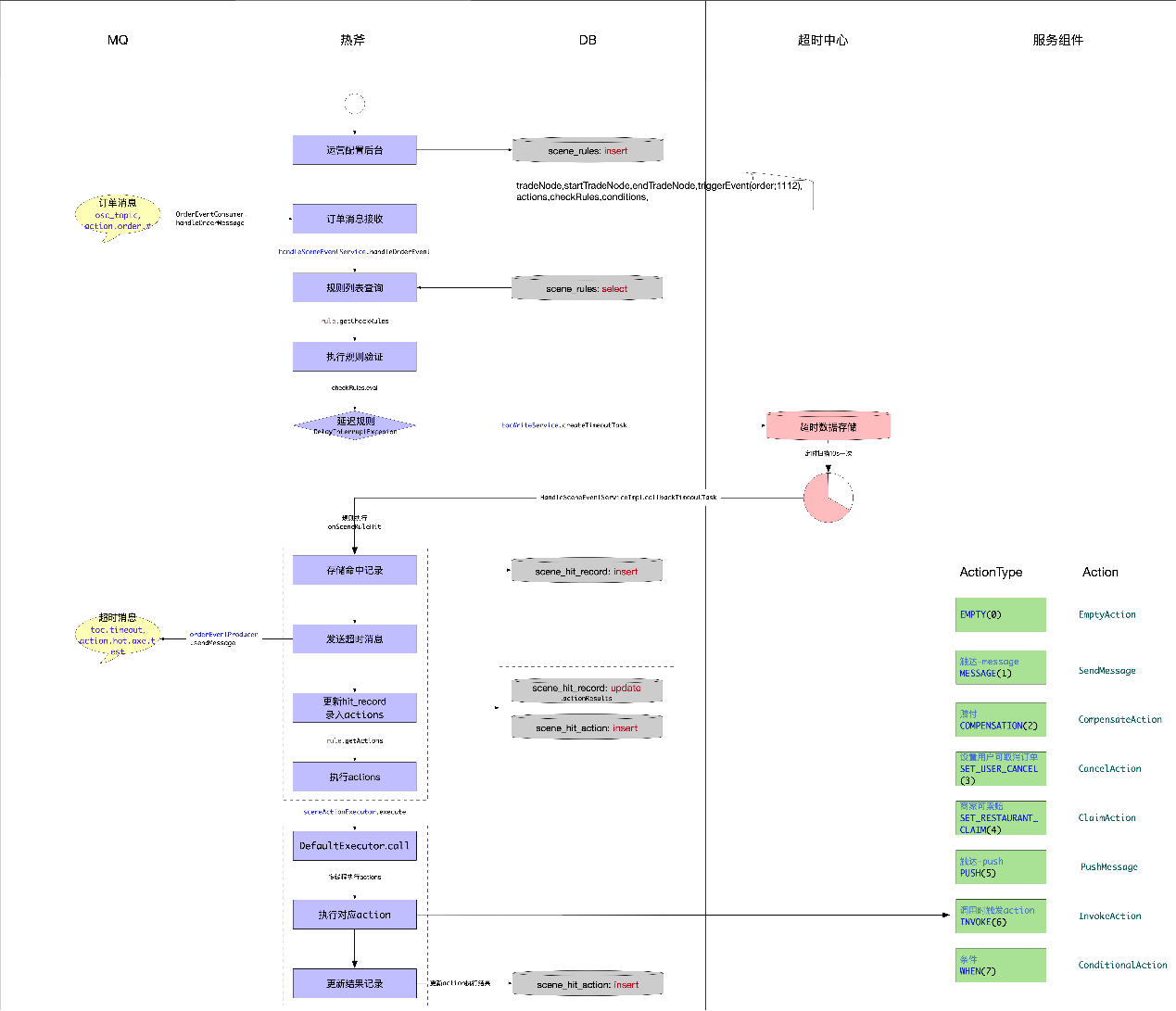
# 第三章 系统分析与设计

## 3.1 系统总体规划

热斧系统是一个规则配置与规则匹配系统，具备快速响应业务需求的能力。

系统拥有一个前端页面，运营人员可以在页面上面新增规则，修改规则，禁用规则，以及根据条件查看规则。在新增和修改规则的时候，系统需要将前端的JSON格式数据转换成表达式引擎能够执行的字符串数据，在此当中需要对表达式引擎进行拓展。除此之外，每当有规则的变动，都需要触发审核流程，审核人员会对规则进行详细审核，确认无误后才会将规则纳入规则库中。

系统还会监听订单消息队列、运单消息队列以及延时任务消息队列。当订单状态发生变化的时候，订单系统会向订单消息队列中发送一条消息，热斧系统会消费此消息，同时会将消息与规则库中的规则进行匹配，如果匹配上规则，会根据规则做命中规则处理。运单消息处理方式与订单消息处理方式类似。规则分为延时规则与非延时规则，当规则是延时规则的时候，将会调用延时中心创建延时任务，等到达预定时间的时候才去做相应的判断。因此需要监听延时任务消息队列。



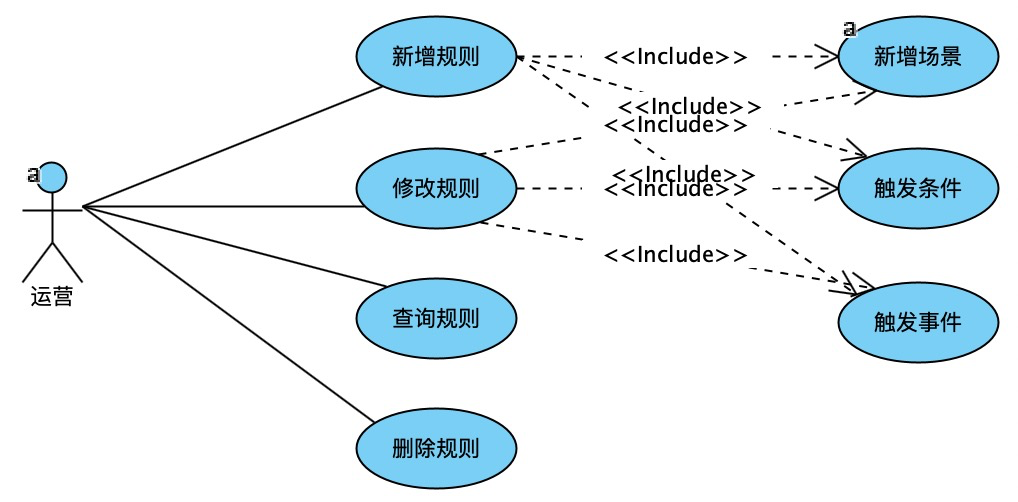
上图是系统大致运行流程。

## 3.2 系统需求分析

基于3.1章节中系统的总体规划，结合热斧系统的实际使用场景，系统需要满足以下的需求：

1. 提供规则配置能力，运营人员在前端页面上能够对规则进行新增，修改，删除，查询等操作。
2. 提供规则预处理能力，运营人员配置完规则后，数据会存在VO对象中，与表达式引擎真正能够执行的字符串有很大差距；除此之外，表达式引擎已有函数不足以支持对规则的解析，需要拓展表达式引擎中的函数。
3. 提供规则审核流管理的能力，当规则有变动的时候，需要触发规则审批流程，同时运营人员需要能在页面上查看规则审批状态，取消规则审批等操作。需要具备流程流转能力，当第一个节点的人审批通过后，系统能够将第一个人流转到第二个人。
4. 提供业务处理能力，系统需要监听订单、运单消息，消费这些消息的同时需要与规则库中的规则进行匹配，并且对匹配后的结果进行下一步的处理。

### 3.2.1 规则管理功能



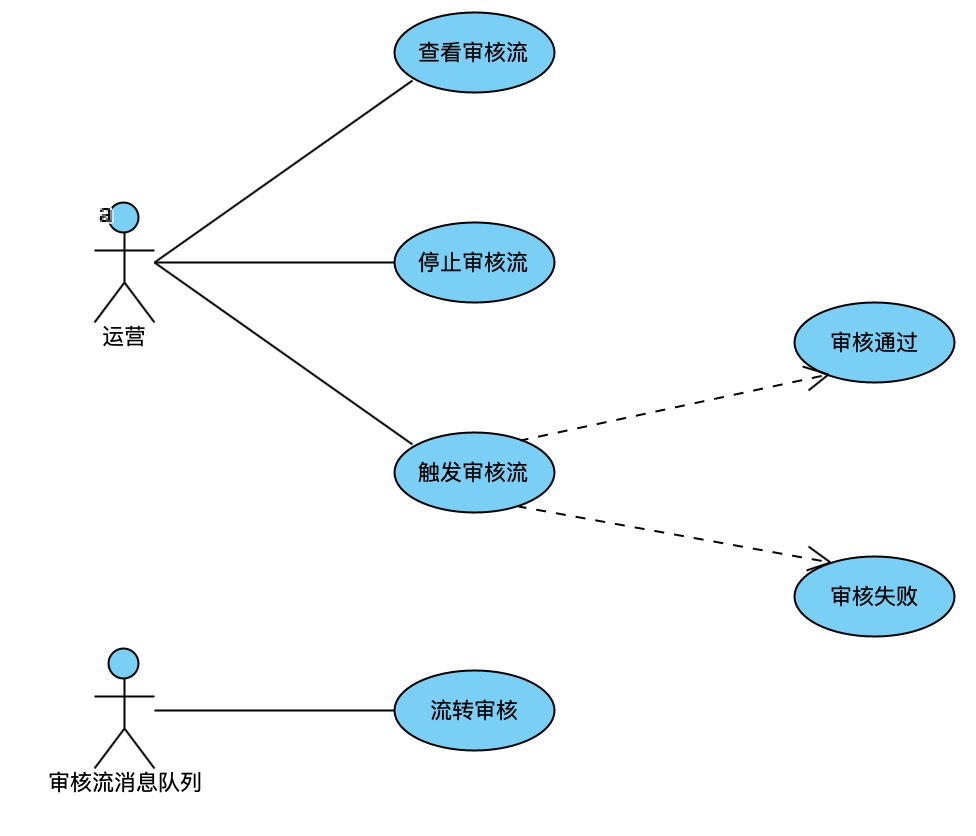
上图是规则管理用例图。规则由于四个部分组成，概括性的描述如下：“主场景”在“满足给定时间”的情况下，没有发生“对应事件”，此时会触发“对应操作”。上述的规则是一个较为通用的模板，所有的规则可以套用这个模板进行编排。例如：用户下单后五分钟内，未发生商家接单，此时指定地区的用户会受到短信，用户可以无责取消订单，并由系统发放赔付红包。在上面的这个例子中，“用户下单”就是主场景，“五分钟”内就是给定时间，“商家接单”就是对应事件，而规则后序的部分就是对应操作。因此在规则新增和规则修改的时候，需要分别去改动上述的四个部分。用户在查询已有规则的时候，可以根据规则名称，主场景类型，是否启用等条件进行查询。

### 规则预处理功能

Aviator支持运算符操作，以及它自带的一些内置函数。然而这些是不能够满足规则解析的需求，例如：aviator没有对集合的操作，需要判断元素是否在集合中的时候，是无法操作的。此外aviator不支持if、when等流程控制语句。不过aviator提供了自定义函数的能力，因此需要定义一些能够解析规则的函数。

除此之外，当运营人员配置完成规则后，规则会以对象的形式存储，此时的规则是不能够被表达式引擎aviator所解析的，aviator只能够执行字符串类型，因此需要将对象类型的规则解析成字符串类型。

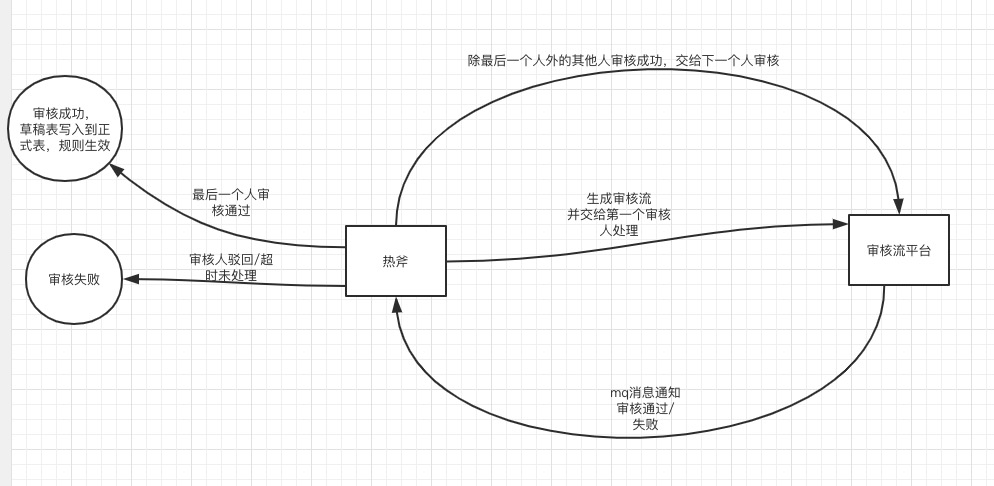
### 审核流程管理功能



上图是审核流程管理的用例图。在规则的配置过程中，有些操作具有高危风险，例如，给用户发送红包，允许商户进行索赔等，如果运营人员在配置完规则后不需要经过审查就直接生效，那么就有可能对公司的资产造成损失。 因此需要有运营人员的上级、上上级等对规则进行审核。

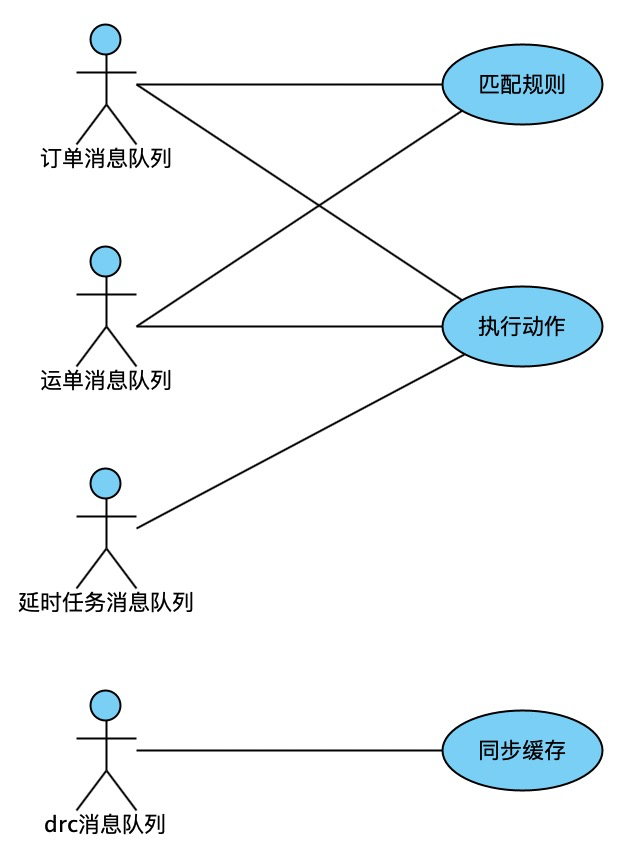
当运营人员新增或修改规则的时候，热斧系统就会在轩辕平台上创建审核流，并且指定审核人员，当审核人员审核通过的时候，轩辕系统会向审核消息对队列中发送一条消息，热斧系统接收到消息后，需要将审核流节点流转给下个审核人，如此循环直到审核均通过或者任意节点被拒绝。此外运营人员可以在页面上查看审核流当前的状态，并且可以选择停止规则的审核。

由于审核流程流转是通过监听消息队列而触发，消息队列存在消息丢失的情况，因此需要一个兜底措施，解决消息丢失的问题。



上图为审核流程流转图

### 业务处理功能



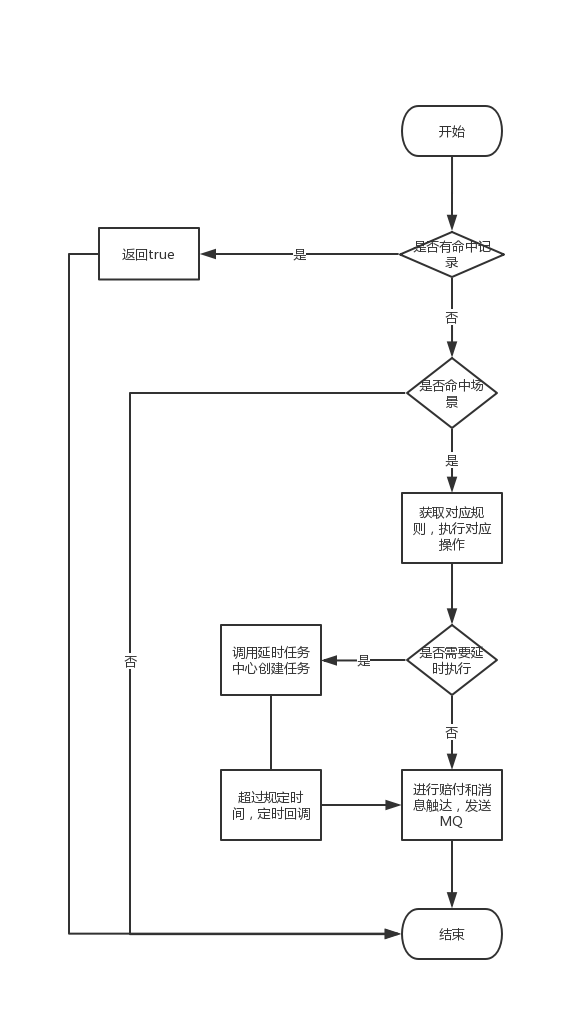
上图为业务处理功能的用例图

业务处理功能包含了订单、运单消息处理，延时任务消息处理，以及drc消息的处理。

订单、运单消息类似，可以放在一起处理，系统接收到订单或运单消息的时候，会根据消息当中的内容与当前已有的规则进行匹配。在匹配的过程中，会判断该规则匹配的时间，如果不需要延时匹配，系统就可以立刻得到匹配结果，根据匹配结果去做相应的操作。

当规则需要延时匹配的时候，系统就会在延时中心创建任务，等到设定时间后，延时中心会向延时任务消息队列中发送数据，系统接收到消息后，会在原有的基础上继续规则匹配，并根据匹配结果进行相应的操作。

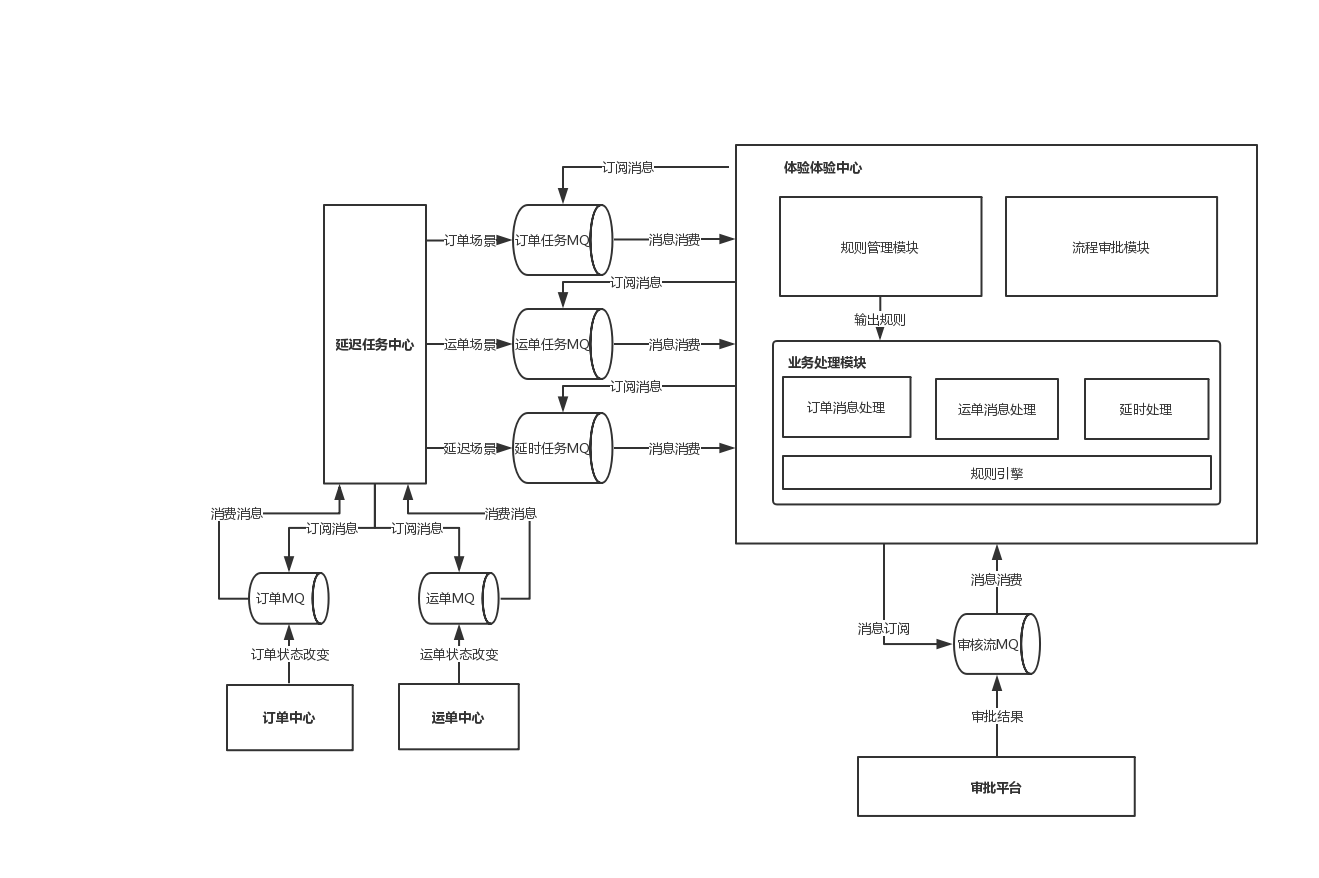
drc消息是异地多活数据同步消息。为了防止一个机房的机子全部宕机，系统一般会在不同的机房里部署，以两个机房为例，用户一次只会影响一个机房的数据，当用户影响a机房的数据的时候，a机房就会给b机房的消息队列发送一条数据同步消息，b机房接收到消息后，就会将b机房的数据就会和a机房的数据进行同步，保证最终一致性，b机房同理。机房之间发送的同步消息就是drc消息。为了避免频繁的访问数据库，数据库中的规则需要缓存到redis中，当数据库中的数据发生变化的时候，redis中的数据也需要同步，监听drc消息为了同步机房的redis消息。



## 3.3 非功能性需求

##

## 3.4 系统总体设计



上图为系统边界

## 3.5 数据库设计

规则表：

|  |  |  |
| --- | --- | --- |
| 字段名 | 类型 | 描述 |
| id | Long | 规则id |
| type | int | 场景类型 |
| description | int | 场景码 |
| Trade\_node | Varchar | 主场景 |
| Start\_trade\_node | Varchar | 业务开始节点 |
| End\_trade\_node | Varchar | 业务结束节点 |
| Trigger\_event | Varchar | 触发事件 |
| Status | Int | 规则状态 |
| Start\_time | Long | 开始时间 |
| End\_time | Long | 结束时间 |
| Check\_rules | Text | 校验规则 |
| Actions | Text | 触发行为 |

规则草稿表

|  |  |  |
| --- | --- | --- |
| 字段名 | 类型 | 描述 |
| id | Long | 规则id |
| type | int | 场景类型 |
| description | int | 场景码 |
| Trade\_node | Varchar | 主场景 |
| Start\_trade\_node | Varchar | 业务开始节点 |
| End\_trade\_node | Varchar | 业务结束节点 |
| Trigger\_event | Varchar | 触发事件 |
| Status | Int | 规则状态 |
| Start\_time | Long | 开始时间 |
| End\_time | Long | 结束时间 |
| Check\_rules | Text | 校验规则 |
| Actions | Text | 触发行为 |

订单命中规则表

|  |  |  |
| --- | --- | --- |
| 字段名 | 类型 | 描述 |
| Id | Long | Id |
| Order\_id | Long | 订单id |
| Scene\_id | Long | 场景id |
| Scene\_code | Int | 场景code |
| Occure\_time | Long | 出现时间 |
| Action\_result | Text | 场景触发的action结果 |
| timeout | Long | 超时时间 |
| Delay | Long | 延迟回调时间 |

场景规则动作表

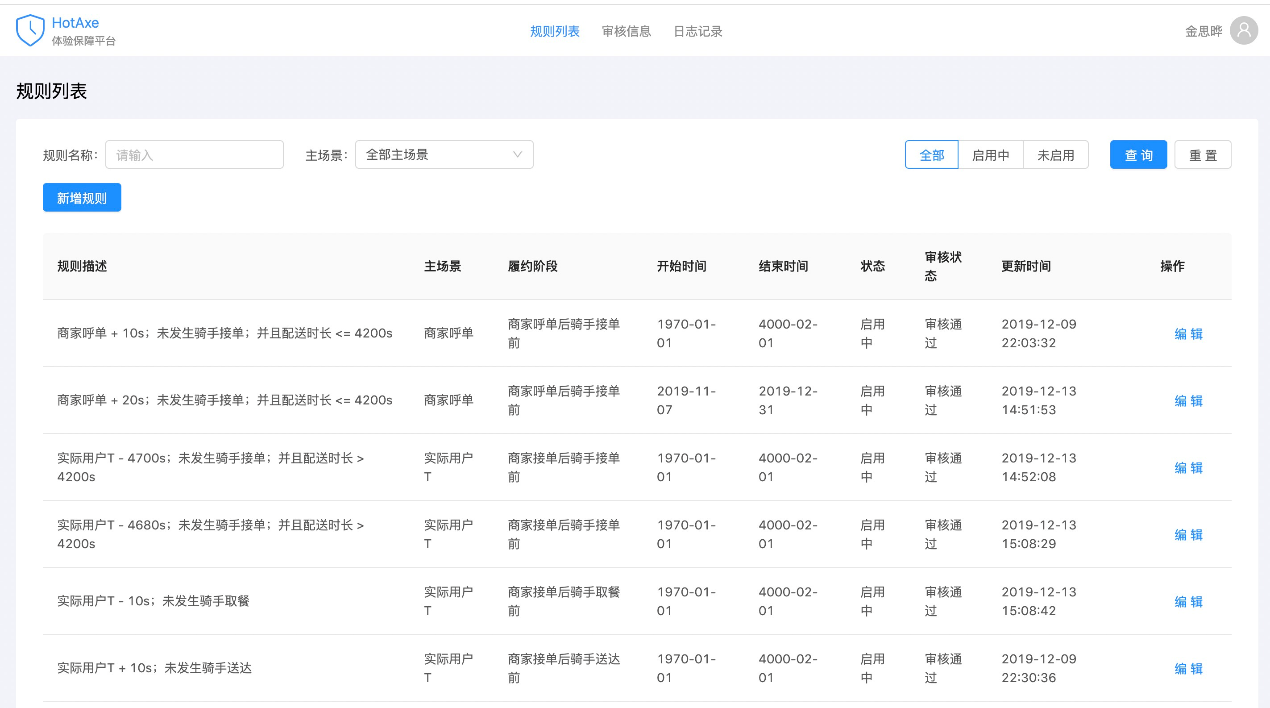
|  |  |  |
| --- | --- | --- |
| 字段名 | 类型 | 描述 |
| id | Long | Id |
| Record\_id | Long | 触发记录id |
| Status | Long | 场景动作状态 |
| Params | Text | 场景动作参数 |
| Results | Text | 场景动作结果 |
| Creator | Varchar | 创建者 |
| updator | Varchar | 更新者 |
| Type | Int | 动作类型 |
| Order\_id | Long | 订单号 |
| trigger\_event | Varchar | 触发事件 |
| Rule\_id | Long | 规则id |
| Name | Varchar | 动作名称 |

## 3.6 本章小结

##

# 第四章 系统实现

## 4.1 规则管理模块







## 4.2 规则预处理模块

## 4.3 审核流模块



## 4.4 业务处理模块

##

## 4.5 系统测试

##

### 4.5.1 测试环境

##

### 4.5.2 测试设计

##

### 4.5.3 测试结果

##

# 第五章 总结与展望

## 5.1 总结

##

## 5.2 工作展望

##

# 参 考 文 献

##

# 致 谢

##