**通达物流系统设计方案**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **版本号** | **日期** | **修改人** | **审阅人** | **摘要** |
| V1.0 | 2020/8/7 | xxx |  | 初稿 |
| V1.1 | 2020/8/30 | xxx |  | 修改 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**技术研发中心**

**2020.08**

目录

[一、 设计背景 3](#_Toc6522)

[1.1、编写目的 3](#_Toc26756)

[1.2、使用范围 3](#_Toc24375)

[1.3、功能概述 3](#_Toc6041)

[1.4、非功能约束 4](#_Toc303)

[二、 系统设计 4](#_Toc5037)

[2.1、物理架构设计 4](#_Toc6183)

[2.2、逻辑架构设计 7](#_Toc6605)

[三、 概要设计 8](#_Toc24269)

[3.1、功能设计 8](#_Toc3171)

[3.2、公用组件 11](#_Toc16019)

[3.3、系统交互设计 11](#_Toc9499)

[四、 详细设计 13](#_Toc12395)

[4.1、时序图 13](#_Toc28464)

[4.2、主要类图 14](#_Toc30832)

[4.3、状态图 14](#_Toc5369)

[4.4、接口设计 15](#_Toc15287)

[4.5、数据模型设计 15](#_Toc8929)

[五、 非功能设计 15](#_Toc10726)

[5.1、性能设计 15](#_Toc13115)

[5.2、安全设计 16](#_Toc66)

[5.3、可靠性设计 16](#_Toc20874)

[5.4、扩展性设计 16](#_Toc6729)

[5.5、可维护性设计 16](#_Toc6454)

## 设计背景

通达是某上市公司全资投资成立的一家物流快递公司，主要进行同城快递业务，公司刚刚成立，组建20人技术部门，准备两个月后系统开发完成上线，你是后端架构师，请你完成系统顶层架构设计，并组织架构评审会议。

### 1.1、编写目的

本文档目的是做什么的

### 1.2、使用范围

本文档哪些人使用

### 1.3、功能概述

* 用户通过app发起快递下单请求并支付
* 快递员通过自己的App上报自己的地理位置，每30秒上报一次
* 系统收到快递请求后，向距离用户直线距离5km内的所有快递员发送通知
* 快递员需要进行抢单，第一个抢单的快递员得到配单，系统向其发送用户详细地址
* 快递员到用户处收取快递，并记录到系统中：已收件
* 快递员将快递送到目的地，并记录到系统中：已送达

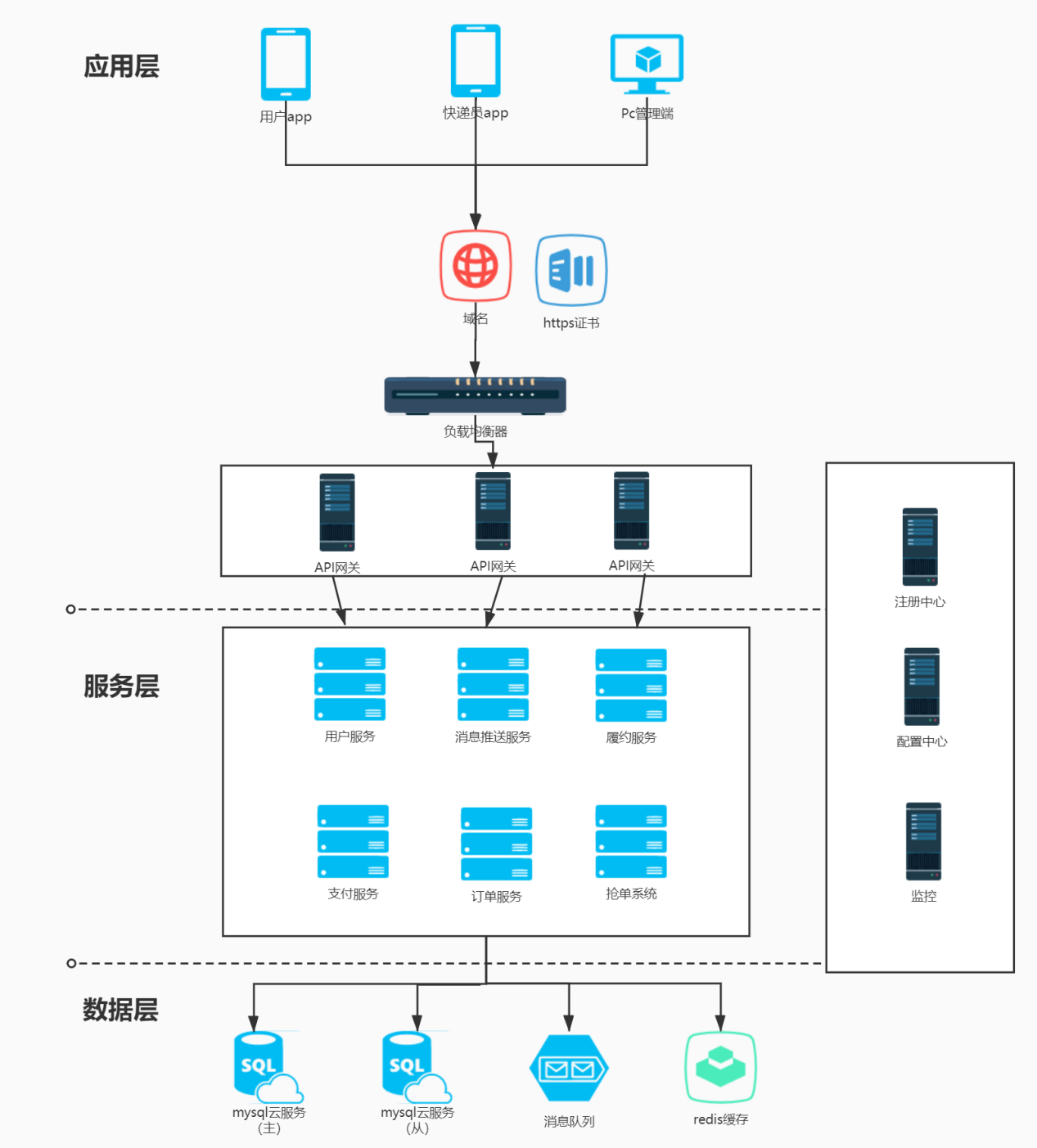
### 1.4、非功能约束

|  |  |
| --- | --- |
| **非功能需求** | **需求描述** |
| 安全性 | 防止DDOS、SQL注入、XSS攻击、CSRF攻击，数据传输采用HTTPS，敏感数据采用加密存储 |
| 可靠性 | 数据库主从设计，保证数据安全存储，冗余备份 |
| 可用性 | 提供99.99%的可用性 |
| 数据量 | 预计上线后三个月日单超过1万，一年日单超过50万 |
| 并发 | QPS 2000/s TPS 500/s |
| 扩展性 | 支持横向扩展 |

## 系统设计

### 2.1、物理架构设计

通达物流系统采用微服务架构进行设计开发，详细部署图如下图：



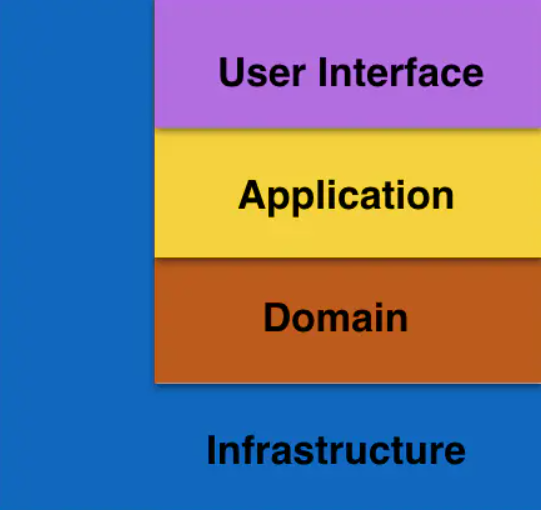
服务器部署情况：

* 用户服务：2台集群
* 消息推送服务：2台集群
* 履约服务：2台集群
* 支付服务：2台集群
* 订单服务：2台集群
* 数据库服务器：2台主从备份

机器配置及域名：

* + 应用服务器（2台）
  + CPU：4C
  + 内存：8G
  + 域名：user.app.sh.tongda.com
  + 硬盘：100G
  + 数据服务器（2台）
  + CPU：8C
  + 内存：16G
  + 域名：db1.sh.tongda.com、db2.sh.tongda.com
  + 硬盘：500G

### 2.2、逻辑架构设计

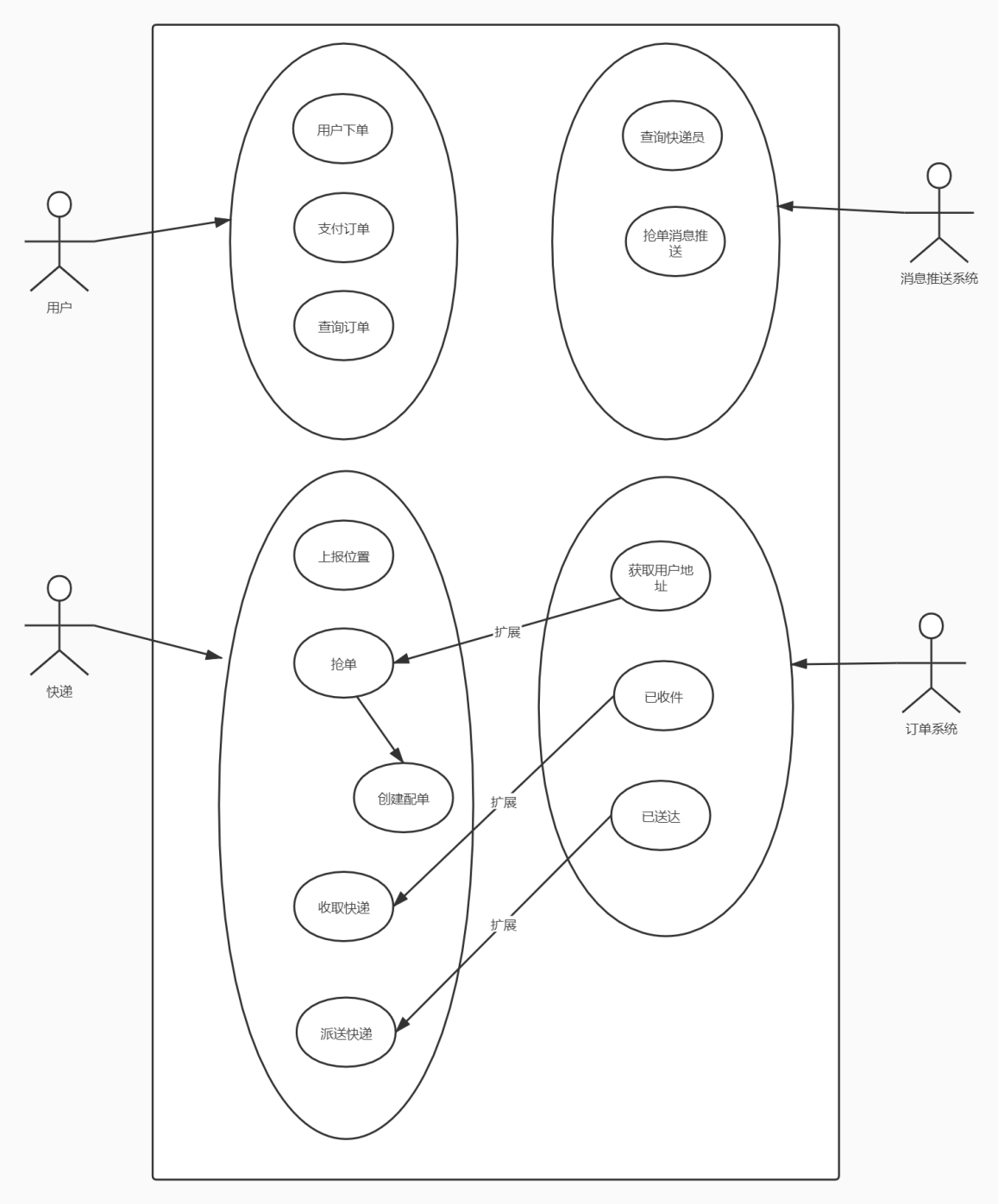


* User Interface为用户界面层（或表示层），负责向用户显示信息和解释用户命令。这里指的用户可以是另一个计算机系统，不一定是使用用户界面的人。
* Application为应用层，定义软件要完成的任务，并且指挥表达领域概念的对象来解决问题。这一层所负责的工作对业务来说意义重大，也是与其它系统的应用层进行交互的必要渠道。应用层要尽量简单，不包含业务规则或者知识，而只为下一层中的领域对象协调任务，分配工作，使它们互相协作。它没有反映业务情况的状态，但是却可以具有另外一种状态，为用户或程序显示某个任务的进度。
* Domain为领域层（或模型层），负责表达业务概念，业务状态信息以及业务规则。尽管保存业务状态的技术细节是由基础设施层实现的，但是反映业务情况的状态是由本层控制并且使用的。领域层是业务软件的核心，领域模型位于这一层。
* Infrastructure层为基础实施层，向其他层提供通用的技术能力：为应用层传递消息，为领域层提供持久化机制，为用户界面层绘制屏幕组件，等等。基础设施层还能够通过架构框架来支持四个层次间的交互模式。

## 概要设计

### 3.1、功能设计

###### 3.1.1、用例视图



###### 3.1.2、用例说明

|  |  |
| --- | --- |
| **用例名称** | **用户下单** |
| 简要说明 | 用户通过APP进行下单操作 |
| 前置条件 | 用户已登录 |
| 后置条件 | 返回订单创建成功，并跳转到支付页面 |
| 基本流程 | 1. 填写订单信息 2. 发送下单请求 3. 订单创建成功 4. 进入支付页面 |
| 扩展流程 | A.订单信息有误  A.系统弹出参数错误  B.订单创建失败  B.系统弹出订单创建失败 |
| 异常流程 | 系统异常，返回错误信息。 |

|  |  |
| --- | --- |
| **用例名称** | **用户支付订单** |
| 简要说明 | 用户通过APP进行订单支付操作 |
| 前置条件 | 用户已登录，订单已创建未支付 |
| 后置条件 | 返回支付成功 |
| 基本流程 | 1. 向履约系统发起订单支付请求 2. 履约系统创建支付流水记录 3. APP通过流水号向第三方支付系统支付 4. 用户在第三方支付系统支付完成 5. 提示用户支付成功 6. 第三方系统回调支付接口，发送已支付订单消息到消息队列 7. 订单系统收到消息后改变订单状态为已支付 8. 消息推送系统收到消息后计算用户周围的快递员向其发送可抢订单消息 9. 抢单系统收到消息后加入可抢资源池中 10. 用户返回APP支付成功 |
| 扩展流程 | 略 |
| 异常流程 | 系统异常，返回错误信息。 |

|  |  |
| --- | --- |
| **用例名称** | **用户查询订单** |
| 简要说明 | 用户通过APP查询订单列表 |
| 前置条件 | 用户已登录 |
| 后置条件 | 返回订单列表 |
| 基本流程 | 略 |
| 扩展流程 | 略 |
| 异常流程 | 系统异常，返回错误信息。 |

|  |  |
| --- | --- |
| **用例名称** | **系统查询快递员** |
| 简要说明 | 系统收到已支付订单消息后，计算用户周围直径5KM的快递员 |
| 前置条件 | 收到已支付订单消息 |
| 后置条件 | 返回符合要求的快递员列表 |
| 基本流程 | 略 |
| 扩展流程 | 略 |
| 异常流程 | 系统异常，返回错误信息。 |

|  |  |
| --- | --- |
| **用例名称** | **抢单消息推送** |
| 简要说明 | 系统收到已支付订单消息后，发送可抢订单消息给符合距离要求的快递员 |
| 前置条件 | 收到已支付订单消息，查询有符合要求的快递员 |
| 后置条件 | 发送可抢订单消息给快递员 |
| 基本流程 | 略 |
| 扩展流程 | 略 |
| 异常流程 | 系统异常，返回错误信息。 |

|  |  |
| --- | --- |
| **用例名称** | **快递员上报位置** |
| 简要说明 | 快递员通过⾃⼰的App上报⾃⼰的地理位置，每30秒上报⼀次 |
| 前置条件 | 快递员已登录 |
| 后置条件 | 上报成功 |
| 基本流程 | 1. 发送位置 2. 存储在redis中 |
| 扩展流程 | 略 |
| 异常流程 | 系统异常，返回错误信息。 |

|  |  |
| --- | --- |
| **用例名称** | **快递员抢单** |
| 简要说明 | 快递员进⾏抢单，抢到订单后，获取用户详细地址，创建配单 |
| 前置条件 | 快递员已登录，收到可抢订单消息 |
| 后置条件 | 抢单成功，配单创建成功 |
| 基本流程 | 1. 收到消息系统推送的可抢订单信息 2. 发起抢单请求 3. 抢单系统收到请求后，进行抢单逻辑计算 4. 抢单成功后，通知履约系统 5. 履约系统收到请求获取用户订单信息和详细地址 6. 为快递员创建配单记录 7. 异步消息通知订单状态：已接单 8. 返回配单信息，抢单成功 |
| 扩展流程 | 1. 资源锁定失败 2. 返回抢单失败 |
| 异常流程 | 系统异常，返回错误信息。 |

|  |  |
| --- | --- |
| **用例名称** | **快递员收取快递** |
| 简要说明 | 快递员到⽤户处收取快递，并记录到系统中：已收件 |
| 前置条件 | 快递员已登录，已有配单信息 |
| 后置条件 | 成功收件，订单状态为已收件 |
| 基本流程 | 略 |
| 扩展流程 | 略 |
| 异常流程 | 系统异常，返回错误信息。 |

|  |  |
| --- | --- |
| **用例名称** | **快递员派送快递** |
| 简要说明 | 快递员将快递送到⽬的地，并记录到系统中：已送达 |
| 前置条件 | 快递员已登录，已有配单信息 |
| 后置条件 | 成功派送，订单状态为已送达 |
| 基本流程 | 略 |
| 扩展流程 | 略 |
| 异常流程 | 系统异常，返回错误信息。 |

### 3.2、公用组件

###### 3.2.1、消息队列

###### 3.2.2、缓存

###### 3.2.3、注册中心

###### 3.2.4、配置中心

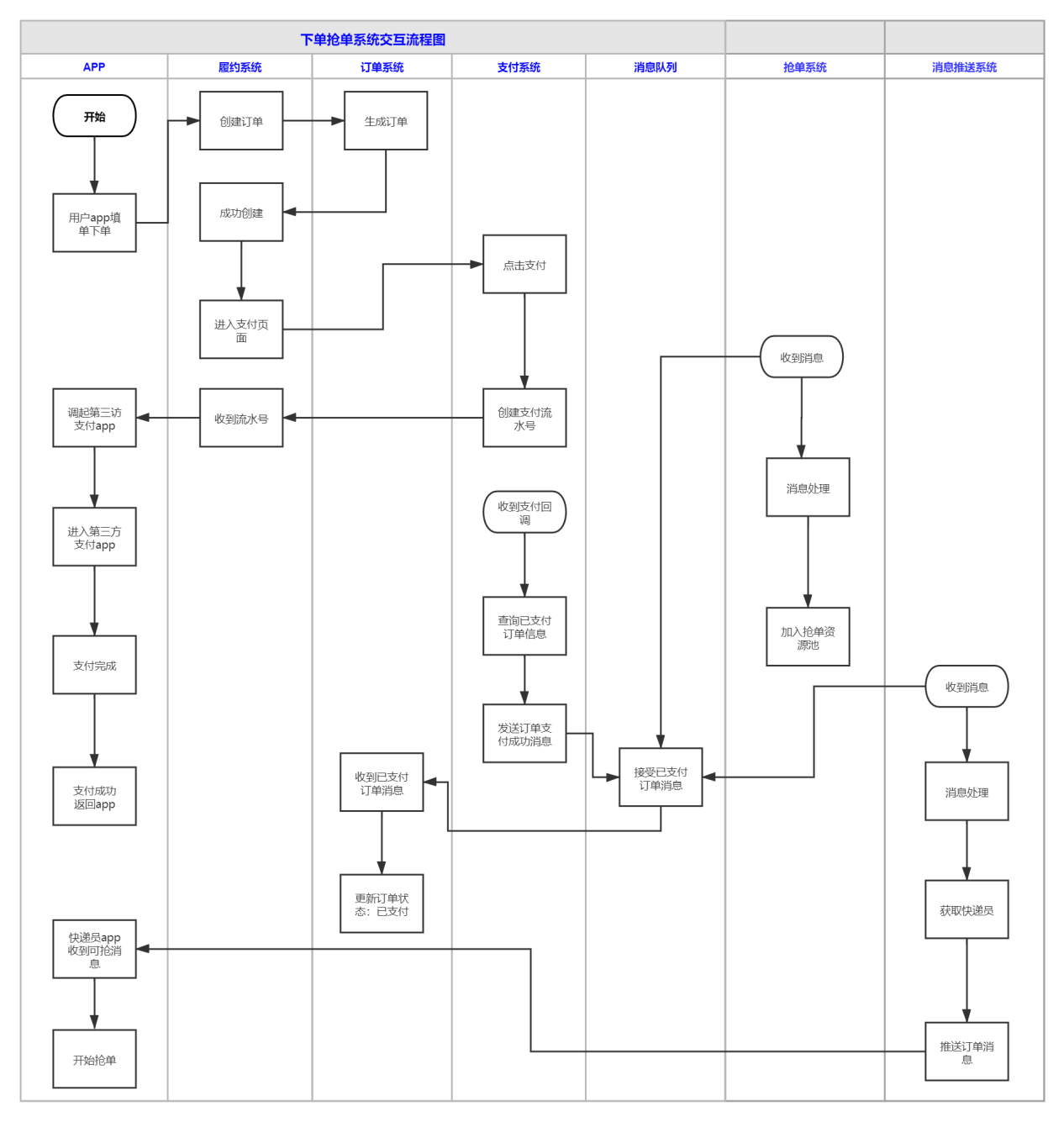
###### 3.2.5、监控中心

### 3.3、系统交互设计

###### 3.3.1、下单流程系统交互

下单流程系统交互涉及的系统如下：

用户APP，快递员APP，履约系统，订单系统，支付系统，消息队列，抢单系统，消息推送系统。



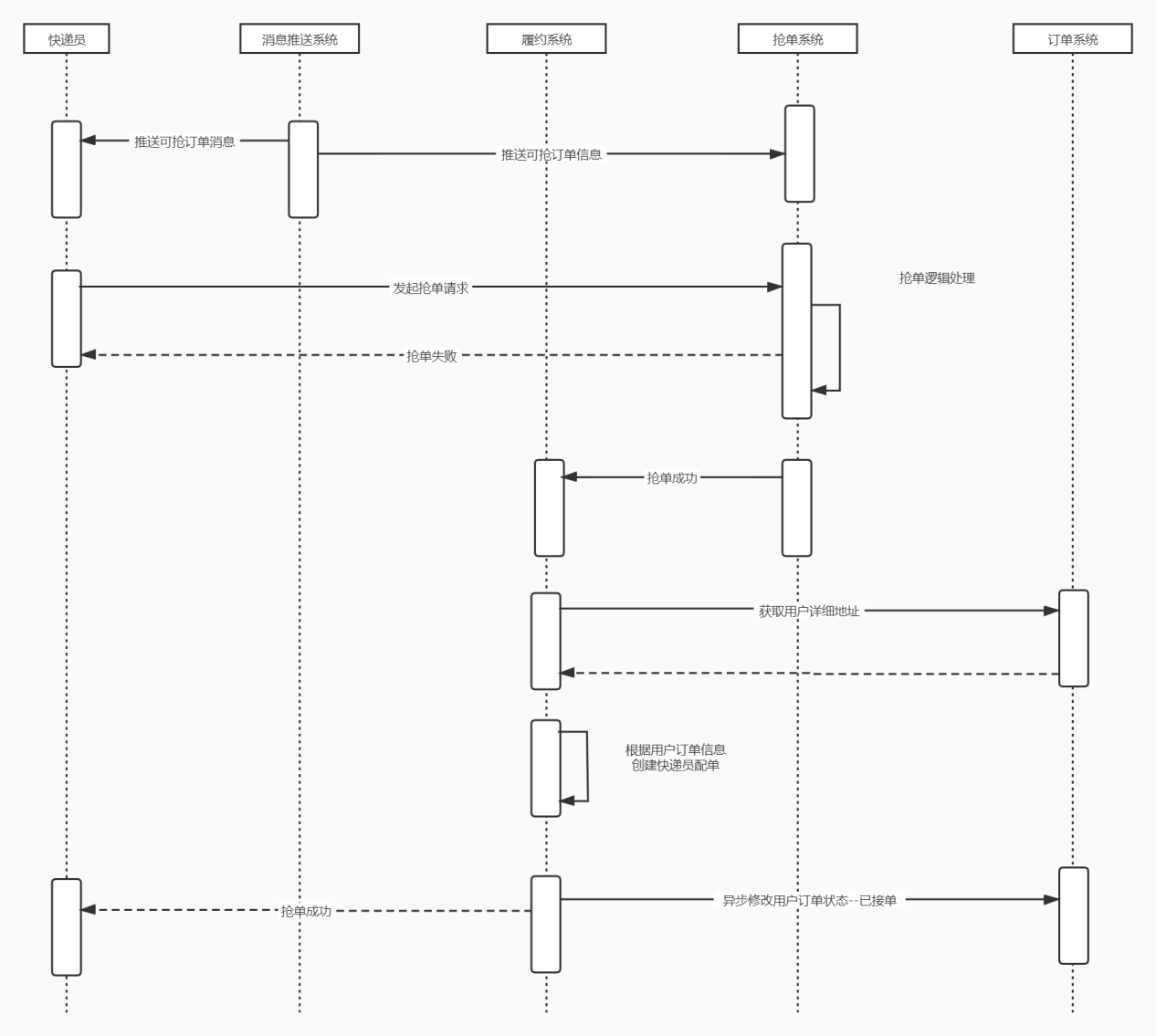
## 详细设计

### 4.1、时序图

###### 4.1.1、抢单时序图

快递员收到消息推送系统发送的可抢订单信息后，向抢单系统进行抢单请求，为了保证抢单系统的无状态性，通过分布式锁的方案进行资源锁定，抢单成功后，通过履约系统获取用户详细地址，创建快递员的配单，通过异步消息修改订单状态为：已接单。

如下图所示：

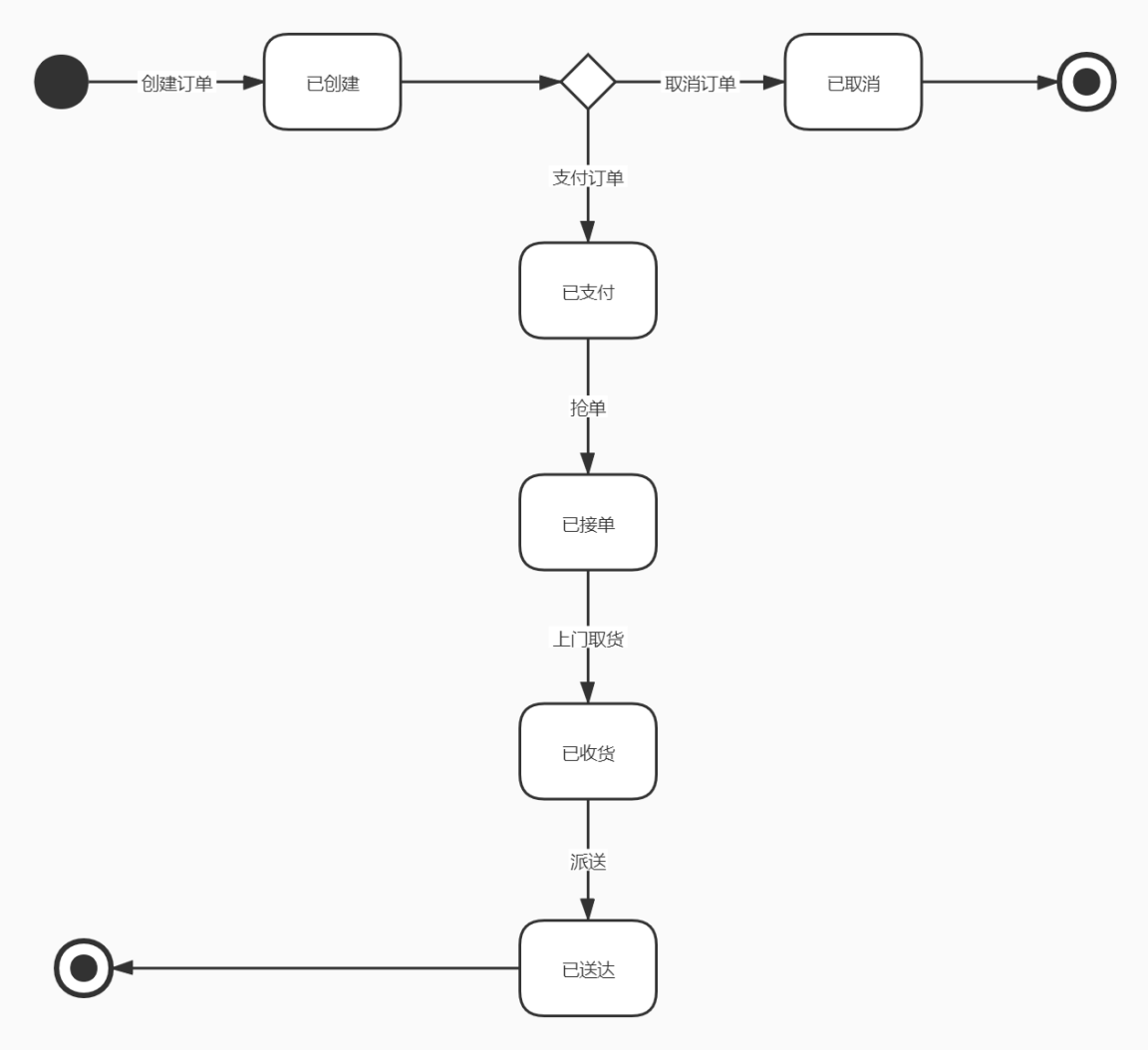


### 4.2、主要类图

### 4.3、状态图

###### 4.3.1、订单状态图

订单状态分为，已创建、已取消、已支付、已接单、已收货、已送达。



### 4.4、接口设计

### 4.5、数据模型设计

###### 4.5.1 账号表

|  |  |  |  |
| --- | --- | --- | --- |
| 账号表 | | | |
| 编号 | 字段 | 类型 | 备注 |
| 1 | id | Int | 自增 |
| 2 | user\_name | varchar | 账号名称 |
| 3 | user\_password | varchar | 账号密码 |
| 4 | user\_email | varchar | 邮箱地址 |
| 5 | status | tinyint | 10启用、20禁用 |
| 6 | is\_system | tinyint | 是否是系统管理员0不是1是 |
| 7 | desc | varchar | 备注 |
| 8 | del\_flag | tinyint | 逻辑删除0没有1有 |
| 9 | create\_time | datetime | 创建时间 |
| 10 | update\_time | datetime | 更新时间 |

## 非功能设计

### 5.1、性能设计

###### 5.1.1、性能目标

业务量：预计上线后三个月日单超过1万，一年日单超过50万

响应时间：200ms/次

并发数量：1000/s

###### 5.1.2、对应设计

数据库索引优化：常用查询关键字设置组合索引，保证查询时索引的使用顺序，减少数据库的数据扫描时间

负载均衡：通过负载均衡器，使请求指向不同Server，分散压力，本系统采用3台服务器进行负载均衡。

### 5.2、安全设计

SQL注入：通过拦截器进行语法拦截

### 5.3、可靠性设计

数据库可靠性：通过数据库的主从复制，达到数据冗余备份，读写分离

### 5.4、扩展性设计

### 5.5、可维护性设计