# 软件架构文档 (3)

<项目名称>

软件架构文档

**版本** <1.0>

## 修订历史记录

| 日期 | 版本 | 说明 | 作 |
| --- | --- | --- | --- |
| 2022.6.29 | 1.0 | 架构的初步设计 | 周千翔、徐国涛 |
| | | | |
| | | | |
| | | | |

## 目录

# 软件架构文档

## 1. 简介

目的

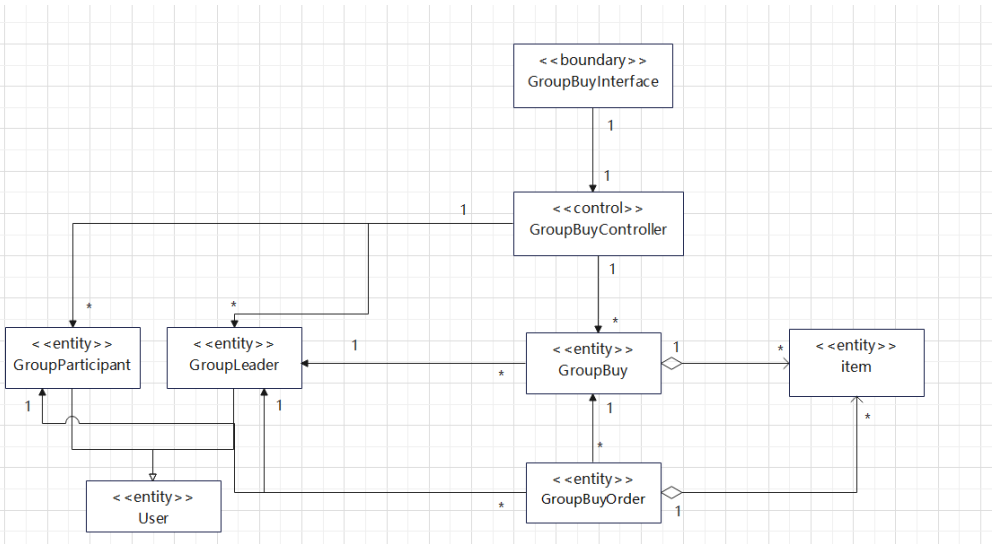
本文档将从构架方面对系统进行综合概述，其中会使用多种不同的构架视图来描述系统的各个方面。它用于记录并表述已对系统的构架方面作出的重要决策。

文档面向对象

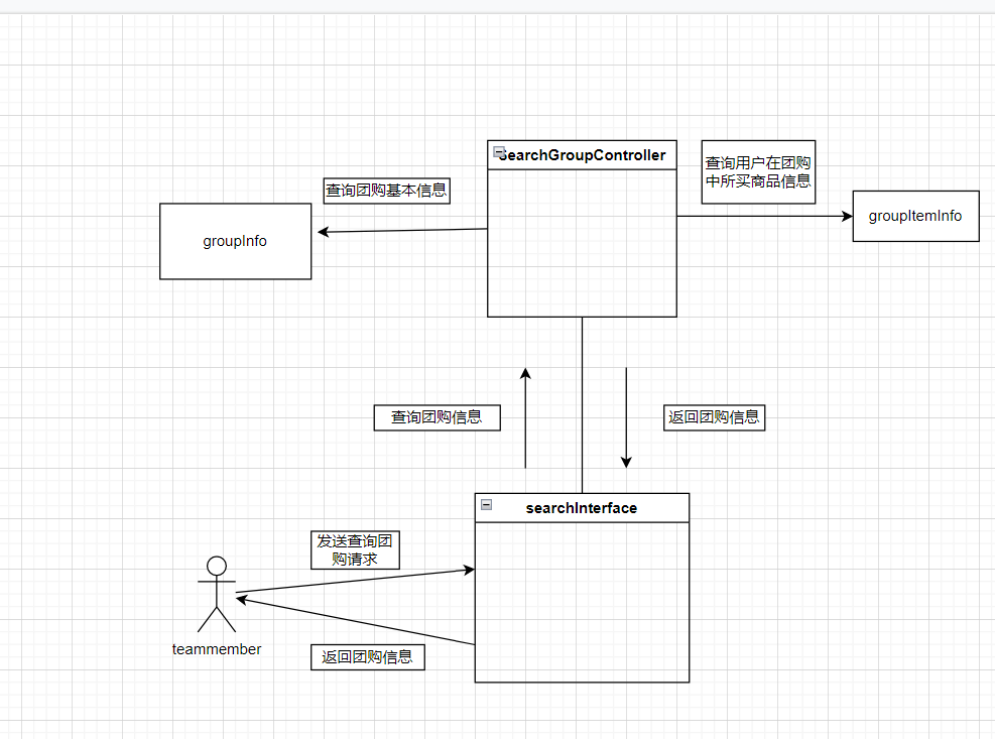
本文档主要面向的对象是团好物APP的开发程序员。

## 2. 用例视图

创建和参与团购VOPC图如下：



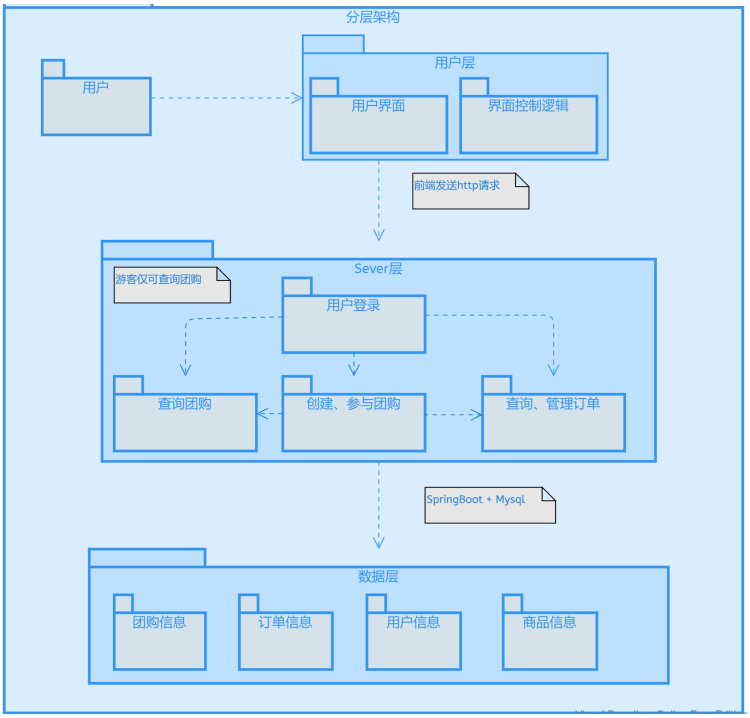

查询团购通信图




## 3. 逻辑视图

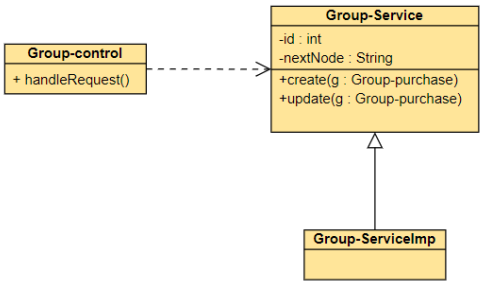
概述

采用3tiers架构，客户端视图包括：login view、home view、order view、information view、 group-purchase view。采用react native框架，用JavaScript构建页面。客户端向服务器端采用HTTP协议发送请求。
应用层功能包括：创建、管理、参与、查询团购，用户登录，以及订单的查询和管理。在实现相应功能时，需对数据层进行相关操作，包括对数据的增删改查。数据层包括多个table，采用Mysql框架，用户信息、订单、团购信息等分别存于不同的table。实现了对数据库的操作后，将所需信息反馈到客户层，客户层根据返回信息，生成相应视图。



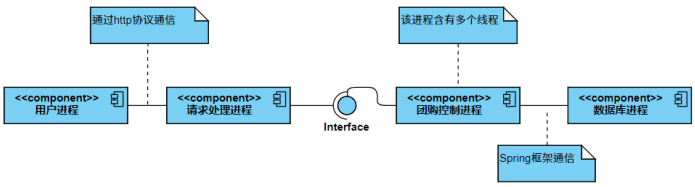

在构架方面具有重要意义的设计包

创建、参与团购包：含有Group-control、Group-Service和Group-ServiceImp类，Group-control类负责接受前序类的参数然后调用Group-Service来处理相应请求。




## 4. 进程视图

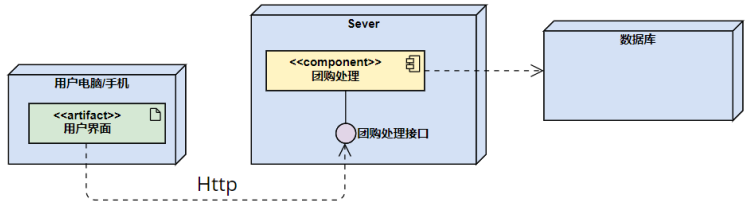
本节为团好物APP的进程视图，着重考虑APP性能、可伸缩性等需求。在进程视图中，包括了四个进程。其中，请求产生进程运行在用户的APP客户端，为轻量级进程；数据库管理系统进程负责管理储存在数据库中的数据；请求处理进程，用于接受APP客户端的请求，并转送给团购控制进程进行处理，为轻量级进程。而团购控制进程为重量级进程，在秒杀和参与团购时采取多线程。




## 5. 部署视图

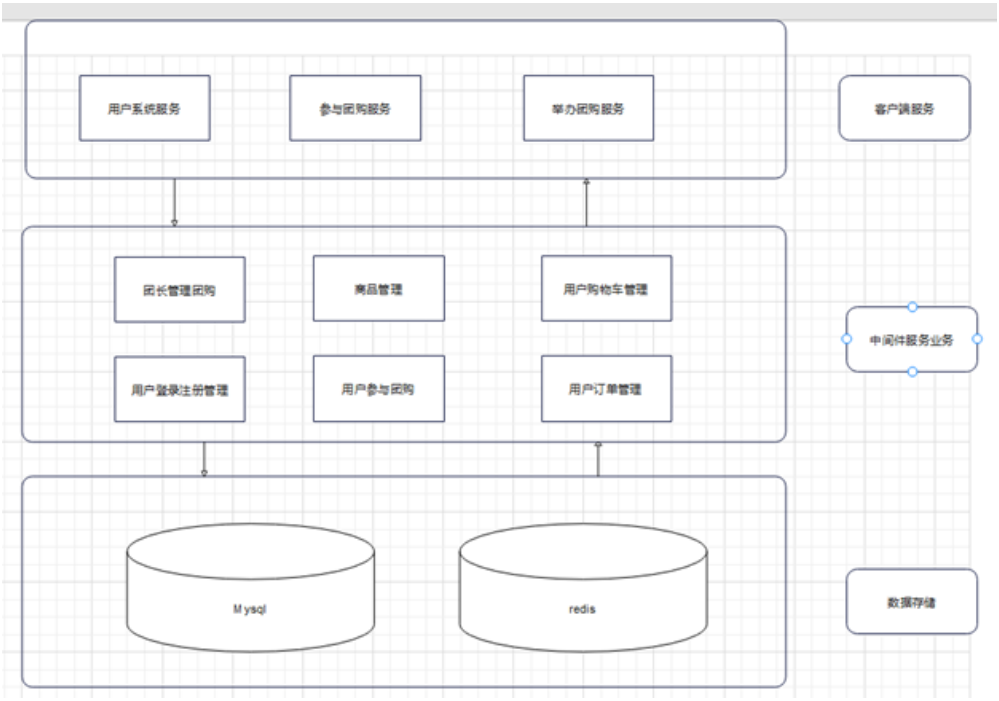
该架构采用client/server模式。用户通过手机等登录APP后，通过互联网以HTTP协议方式发送请求到服务器端，服务器端根据请求实现相应业务，并对数据库进行指定操作，再将信息返

回给用户端。用户端根据收到数据生成相应的页面进行展示。




## 6. 实现视图

实现视图：客户端主要是用户服务系统，包括登录和注册，参与团购和创建举办团购活动
中间层：主要是管理用户登录和注册，团长管理团购，包括其子系统的商品管理，用户参与团购的订单服务和加入购物车服务.
数据的储存：是通过关系型数据库mysql和redis缓存管理




## 7. 技术视图

7.1 Client前端
1)编程语言 JavaScript
2) React 框架 + Html + CSS
3) 使用 WebPack/npm/yarn 等工具打包管理
7.2 Server后端
Spring 技术栈：
1) 使用 Spring 来实现(Spring MVC + Bootstrap + ORM)

2) 使用 Maven 打包管理

7.3 数据库

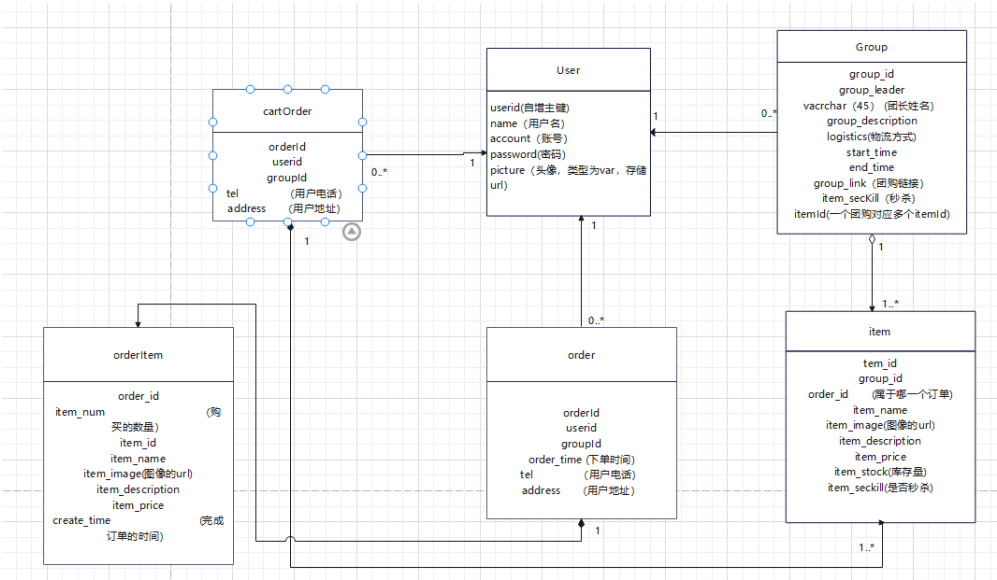
关系型数据库：MySQL，Navicat

7.4 IDE 开发工具

IntelliJ IDEA， VSCode

## 8. 数据视图（可选）

用户可以参与或创建团购group，团购中有多个item的商品信息，每个用户可以拥有订单和购物车订单，购物车订单与一个团购相关联，购物车内item的信息随团购中item的变化而变化。此外用户还有订单数据，是用户支付之后的生成的订单数据。其中的商品信息保存在orderitem的信息是来自支付时团购内item的信息。




## 9. 核心算法设计（可选）

秒杀思路设计：

1、秒杀相关的活动页面相关的接口，所有查询能加缓存的，全部添加redis的缓存；

2、活动相关真实库存、锁定库存、限购、下单处理状态等全放redis；

3、当有请求进来时，进入活动ID为粒度的分布式锁，第一步进行用户购买的重复性校验，满足条件进入下一步，否则返回已下单的提示；

4、第二步，判断当前可锁定的库存是否大于购买的数量，满足条件进入下一步，否则返回已售罄的提示；

5、第三步，锁定当前请求的购买库存，从锁定库存中减除，并将下单的请求放入kafka消息队列；

6、第四步，在redis中标记一个polling的key（用于轮询的请求接口判断用户是否下订单成功），在kafka消费端消费完成创建订单之后需要删除该key，并且维护一个活动id+用户id的key，防止重复购买；

7、第五步，消息队列消费，创建订单，创建订单成功则扣减redis中的真实库存，并且删除polling的key。如果下单过程出现异常，则删除限购的key，返还锁定库存，提示用户下单失败；

8、第六步，提供一个轮询接口，给前端在完成抢购动作后，检查最终下订单操作是否成功，主要判断依据是redis中的polling的key的状态；
9、整个流程会将所有到后端的请求拦截的在redis的缓存层面，除了最终能下订单的库存限制订单会与数据库存在交互外，基本上无其他的交互，将数据库I/O压力降到了最低；

## 10. 质量属性的设计

本系统所采用的软件架构可以很好地支持如下的质量方面及性能方面的需求：

1. 易用性：APP美观、流畅，组件布局合理，符合大众一般的使用习惯，确保99.99%的用户，即便是没有基础手机使用经验的用户在使用本APP上所花费的培训（完成指导试玩）时间也不会超过5分钟；
2. 并发性：本APP可同时支持2000用户进行使用，服务器平均响应时间不超过3秒，可支持团购秒杀功能；
3. 可移植性：本APP的设计可以在所有的Android平台上运行；

4.可靠性：

（1）对用户输入有提示与检查，防止数据异常；

（2）系统健壮性强，应能处理系统运行时出现的异常情况，如：人为输入错误、数据非法、硬件设备错误等，系统应能正常处理。

（3）在1000000次交互中，至多出现1次需重启系统的情况。

（4）可用时间百分比：99.9%

（5）平均故障间隔时间：45天

（6）平均修复时间：1h

（7）精确度：计算至小数点后两位

5.可扩展性：本APP架构设计分为多个模块，采用松耦合，较易拓展新功能。