

1.项目介绍

学习目标

- 系统的功能、背景、场景及需求
- 在架构角度思索系统可能面临的问题以及解决方案
- 运用中间件特性，完成架构设计
- 主业务源码分析
- 微服务的部署与动态扩容

1.1项目概述

1.1.1概述

京东的红包雨大家可能都参与过，在某段时间内随机发放不同的红包，如果公司让你设计类似系统，如何做？

本项目扩展为一个通用的红包雨模式抽奖系统，兼容多种规则。

由管理后台配置相关活动和奖品等信息，前端用户通过参与活动，完成抽奖。

1.1.2应用场景

1) 时间随机

在一段时间内，设置一批礼品，这些礼品不定时的出现，尽量在这段时间内均匀抛出，一旦出现，就可以被抓走。类似抓红包。

2) 瞬间秒杀

用于抢单或者秒杀场景，到点后，用户一起抽奖，机会均等，谁抢的快算谁的。这个并发比较高。但是活动时间相对较短。

3) 机会随机

常见于转盘类活动。不同等级的用户，设定不同的中奖概率，一般配合设置用户最大可抽奖次数，比如5次机会，能不能中奖，根据概率判定。一般活动时间设置的较长，比如几天。

1.1.3系统要求

1) 并发性

抽奖系统比如涉及到访问量大的问题。系统涉及所面临的第一关，即活动开始的瞬间，大批用户点击的涌入。怎样设计系统以达到如此高并发情况下的及时响应是本项目的重中之重。

2) 库存控制

抽奖面临的必然是奖品。数量控制是必须要做到精准吻合。不允许出现设置了5个奖品，最终6人中奖这种类似的问题出现。其中的本质是奖品库存的控制。

3) 投放策略

在活动时间段内，管理员设置好的一堆奖品如何投放？红包何时出现？什么时候可以被抽中？这些都涉及到投放策略。

4) 边界控制

活动何时开始？何时结束？倒计时如何控制。这涉及到活动的边界。开始前要提防用户提前进入抽奖。结束后要即使反馈结果给用户，告知活动已结束。

5) 活动自由配置

活动的配置由后台管理员完成，可以自由配置活动的开始结束时间，主题、活动简介、有哪些奖品、不同等级的用户中奖的策略。这就要求系统必须具备足够的业务灵活度。

6) 中奖策略

每个用户参与抽奖后，要遵从后台管理员所设定的中奖策略，典型的场景是最大抽奖次数和最大中奖次数。

1.2功能展示

1.2.1管理后台

1) 会员管理

功能：用户查询、用户新增、删除、修改密码

用户管理为管理员提供基本的用户录入。本项目可参与抽奖的用户由管理员后台直接录入，不允许私自注册其他非法账号。已录入的账号可以在抽奖前端页面中登录，参与抽奖。

在电商面向C端用户的情况下，新增一个注册接口，允许用户自行注册参与抽奖。

首页会员管理 x

用户名: 姓名: 身份证号: 手机号码:

等级: 注册时间: ~

+ 增加 编辑 删除 修改密码 导出

搜索 重置

	用户名	姓名	身份证号	手机号码	等级	注册时间	更新时间
1	☐ test	张艳新	15042619890518175	15201251947	普通会员	2019-09-27 02:38:42	2019-09-27 02:38:42
2	☐ 0	0	0	0	普通会员	2019-09-16 10:19:53	2019-09-16 10:19:53
3	☐ 4	4	4	4	四级会员	2019-09-16 10:19:40	2019-09-16 10:19:40
4	☐ 3	3	3	3	三级会员	2019-09-16 10:19:26	2019-09-16 10:19:26
5	☐ 2	2	2	2	二级会员	2019-09-16 10:19:12	2019-09-16 10:19:12
6	☐ 1	1	1	1	一级会员	2019-09-16 10:19:01	2019-09-16 10:19:01

2) 会员等级

功能：等级新增、删除、编辑

不同等级的会员有不同的中奖策略设置。比如高级别的会员中奖次数更多。详细会涉及下面活动配置中策略配置一节

首页

会员管理 x

等级管理 x

等级名称:

+ 增加

✎ 编辑

✕ 删除

📍 详情

	☐	等级代号	等级名称	创建时间
1	☐	4	四级会员	2019-08-24 03:29:25
2	☐	3	三级会员	2019-08-23 05:07:23
3	☐	2	二级会员	2019-08-23 05:07:22
4	☐	1	一级会员	2019-08-23 05:07:20
5	☐	0	普通会员	2019-08-23 05:07:19

3) 活动管理

基础信息配置

功能：新增活动，修改活动，删除活动，配置活动基本信息（开始结束时间，标题，说明）

活动的基本信息管理功能

首页

会员管理 x

等级管理 x

创建活动 x

活动主题:

开始时间:

~

类型:

+ 增加

✎ 编辑

✕ 删除

📍 详情

📄 导出

🔍 搜索

🔄 重置

	☐	活动主题	开始时间	结束时间	类型	操作
1	☐	2019抢房节	2019-09-09 10:11:35	2019-09-09 16:30:51	随机类	奖品配置 策略配置
2	☐	年会抽奖第二轮	2019-09-05 10:12:15	2019-08-29 00:00:00	随机类	奖品配置 策略配置
3	☐	年会抽奖第一轮	2019-09-05 10:12:21	2019-08-29 00:00:00	随机类	奖品配置 策略配置

首页 | 创建活动 x

活动主题: 开始时间: ~ 类型:

+ 增加 / 编辑 / 删除 / 详情 / 导出

	活动主题	开始时间	结束时间	类型	操作
1	2019抢房节	2019-09-09 10:11:35	2019-09-09 16:30:51	随机类	奖品配置 策略配置
2	年会抽奖				
3	年会抽奖				

活动配置-增加

活动主题:

活动简介:

开始时间:

结束时间:

类型:

保存 / 取消

策略配置

功能: 新增, 修改, 删除策略

策略涉及到用户的中奖次数, 可以为不同等级的用户设置不同的最大中奖机会。不设置或者设置为0表示次数不限。

首页 | 创建活动 x

活动主题: 开始时间: ~ 类型:

+ 增加 / 编辑 / 删除 / 详情 / 导出

搜索 / 重置

	活动主题	开始时间	结束时间	类型	操作
1	2019抢房节	2019-09-09 10:11:35	2019-09-09 16:30:51	随机类	奖品配置 策略配置
2	年会抽奖第二轮	2019-09-05 10:12:15	2019-08-29 00:00:00	随机类	奖品配置 策略配置
3	年会抽奖第一轮	2019-09-05 10:12:21	2019-08-29 00:00:00	随机类	奖品配置 策略配置

子页面

+ 增加 / 编辑 / 删除

活动	会员等级	可抽奖次数 (0为不限)	最大中奖次数 (0为不限)
2019抢房节	四级会员	0	100
2019抢房节	三级会员	1	2
2019抢房节	二级会员	31	0
2019抢房节	一级会员	2	0

奖品配置

功能: 添加, 删除, 编辑奖品

为活动配置响应的奖品, 可以添加多个不同的奖品, 并为每个奖品设置单独的数量。

创建活动

活动主题: 开始时间: ~ 结束时间: 类型: 加 编辑 删除 详情 导出 搜索 重置

活动主题	开始时间	结束时间	类型	操作
2019抢房节	2019-09-09 10:11:35	2019-09-09 16:30:51	随机类	奖品配置 策略配置
年会抽奖第二轮	2019-09-05 10:12:15	2019-08-29 00:00:00	随机类	奖品配置 策略配置
年会抽奖第一轮	2019-09-05 10:12:21	2019-08-29 00:00:00	随机类	奖品配置 策略配置

子页面
+ 增加 编辑 删除

	活动	奖品	数量
1	☐ 2019抢房节	哈士奇一只	4
2	☐ 2019抢房节	二环内四合院一套	6
3	☐ 2019抢房节	地下车位一个	5
4	☐ 2019抢房节	美国商务办公室一套	5

4) 奖品管理

奖品管理

功能：奖品增加，编辑，删除

录入奖品的基本信息，可供多个活动引用。

首页 会员管理 等级管理 创建活动 奖品管理

奖品名称: 市场价: ~ 加 编辑 删除 详情 导出 搜索 重置

	奖品名称	图片	简介	市场价
1	☐ 不长胖奶茶一杯		奶茶原为中国北方游牧	50
2	☐ 狗狗一只		简介：中华田园犬，传	1000
3	☐ 地下车位一个		地库设计导则 - 说明？	500
4	☐ 美国商务办公室一套		白宫（英语：The Wh	1000
5	☐ 劳斯莱斯一辆		劳斯莱斯（Rolls-Roy	5000000
6	☐ 二环内四合院一套		四合院，又称四合房，	50000000
7	☐ 哈士奇一只		西伯利亚雪橇犬是原	1000

5) 信息管理

中奖统计

功能：只有按条件查询，不涉及其他操作

统计每个活动的奖品总数，以及被抽走的数量。该功能只涉及数据的统计，不涉及新增修改删除，属于只读操作。

首页

会员管理 x

等级管理 x

创建活动 x

奖品管理 x

中奖统计 x

活动主题:

开始时间:

~

详情

导出

搜索

重置

	☐ 活动主题	开始时间	结束时间	活动类型	总奖品数	已抽中	
1	☐ 财务自由就靠这了	2019-09-16 16:41:46	2019-09-16 16:43:46	随机类	40		
2	☐ 2019抢房节	2019-09-09 10:11:35	2019-09-09 16:30:51	随机类	20		
3	☐ 年会抽奖第二轮	2019-09-05 10:12:15	2019-08-29 00:00:00	随机类	36		
4	☐ 年会抽奖第一轮	2019-09-05 10:12:21	2019-08-29 00:00:00	随机类	14		
5	☐ 10周年庆典	2019-09-24 15:48:12	2019-09-24 15:55:12	随机类	42	5	

中奖列表

功能：基于各种条件查询中奖详情

可以根据所需条件，查询到相关的中奖信息，中奖人信息，奖品信息，中奖时间等。该功能只涉及数据的统计，不涉及新增修改删除，属于只读操作。

首页

会员管理 x

等级管理 x

创建活动 x

奖品管理 x

中奖统计 x

中奖列表 x

活动主题:

用户名:

手机号码:

奖品名称:

中奖时间: ~

详情

导出

搜索

重置

	☐ 活动主题	活动类型	用户名	姓名	身份证号	手机号码	会员等级	奖品名称	市场价	中奖时间
1	☐ 10周年庆典	随机类	2	2	2	2	二级会员	美国商务办公室一套	1000	2019-09-24 15:27:11
2	☐ 10周年庆典	随机类	2	2	2	2	二级会员	二环内四合院一套	50000000	2019-09-24 15:27:11
3	☐ 10周年庆典	随机类	2	2	2	2	二级会员	哈士奇一只	1000	2019-09-24 15:27:10
4	☐ 10周年庆典	随机类	2	2	2	2	二级会员	iPhoneX一部	5000	2019-09-24 15:26:46
5	☐ 10周年庆典	随机类	2	2	2	2	二级会员	iPhoneX一部	5000	2019-09-24 15:26:45

6) 系统管理

操作日志

功能：查询管理员的操作日志

该功能用于记录管理员的操作。可以根据ip，操作时间内容，以及操作人查询到在后台中的行为。只涉及数据的统计，不涉及新增修改删除，属于只读操作。

首页

会员管理 x

等级管理 x

创建活动 x

奖品管理 x

中奖统计 x

中奖列表 x

操作日志 x

用户:

▼

操作内容:

ip:

操作时间:

~

导出

	☐ 用户	操作内容	ip	操作时间	
1	☐ test	登陆系统	113.44.46.127	2019-10-07 10:49:50	
2	☐ zcurd	登陆系统	14.131.252.229	2019-09-29 09:06:18	
3	☐ zcurd	登陆系统	14.131.250.25	2019-09-28 12:36:32	
4	☐ admin	登陆系统	14.131.250.25	2019-09-28 10:44:14	
5	☐ test	退出系统	14.131.250.25	2019-09-28 10:44:01	
6	☐ test	登陆系统	14.131.250.25	2019-09-28 09:27:36	
7	☐ zcurd	登陆系统	14.131.250.25	2019-09-27 21:15:44	
8	☐ test	登陆系统	14.131.250.25	2019-09-27 15:53:00	
9	☐ zcurd	登陆系统	14.131.250.25	2019-09-27 15:25:57	
10	☐ zcurd	[会员test] 增加	14.131.250.25	2019-09-27 10:38:42	
11	☐ zcurd	登陆系统	14.131.250.25	2019-09-27 10:36:49	
12	☐ zcurd	登陆系统	14.131.250.25	2019-09-27 10:10:40	
13	☐ zcurd	登陆系统	14.131.250.25	2019-09-26 17:29:19	
14	☐ test	登陆系统	14.131.250.25	2019-09-26 17:22:28	
15	☐ test	登陆系统	14.131.250.25	2019-09-25 11:25:37	
16	☐ zcurd	登陆系统	14.131.250.25	2019-09-25 10:47:27	
17	☐ test	登陆系统	14.131.250.25	2019-09-25 09:54:06	

1.2.2前台展示

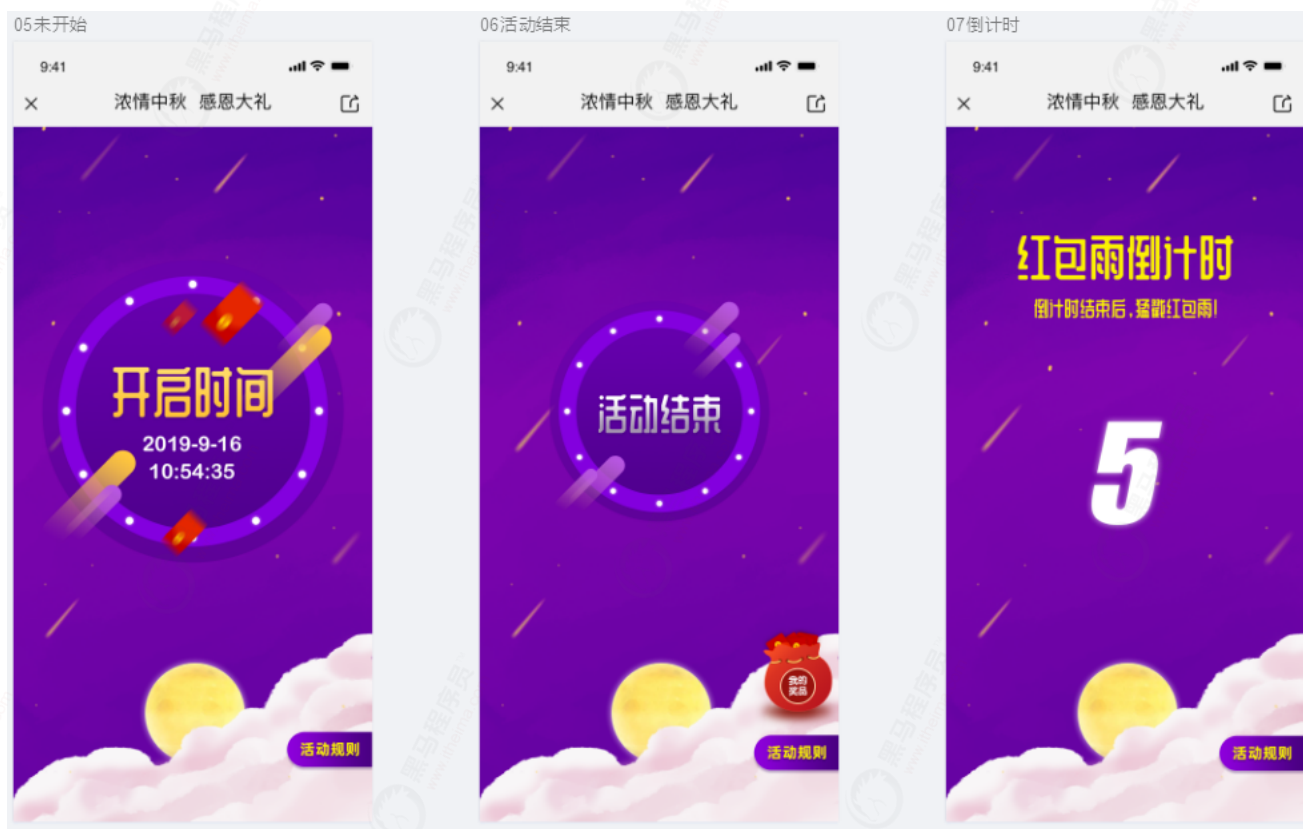
1) 活动列表



2) 活动详情



3) 抽奖展示



4) 个人中心

11我的



11我的_未中奖



14我的奖品_有中奖



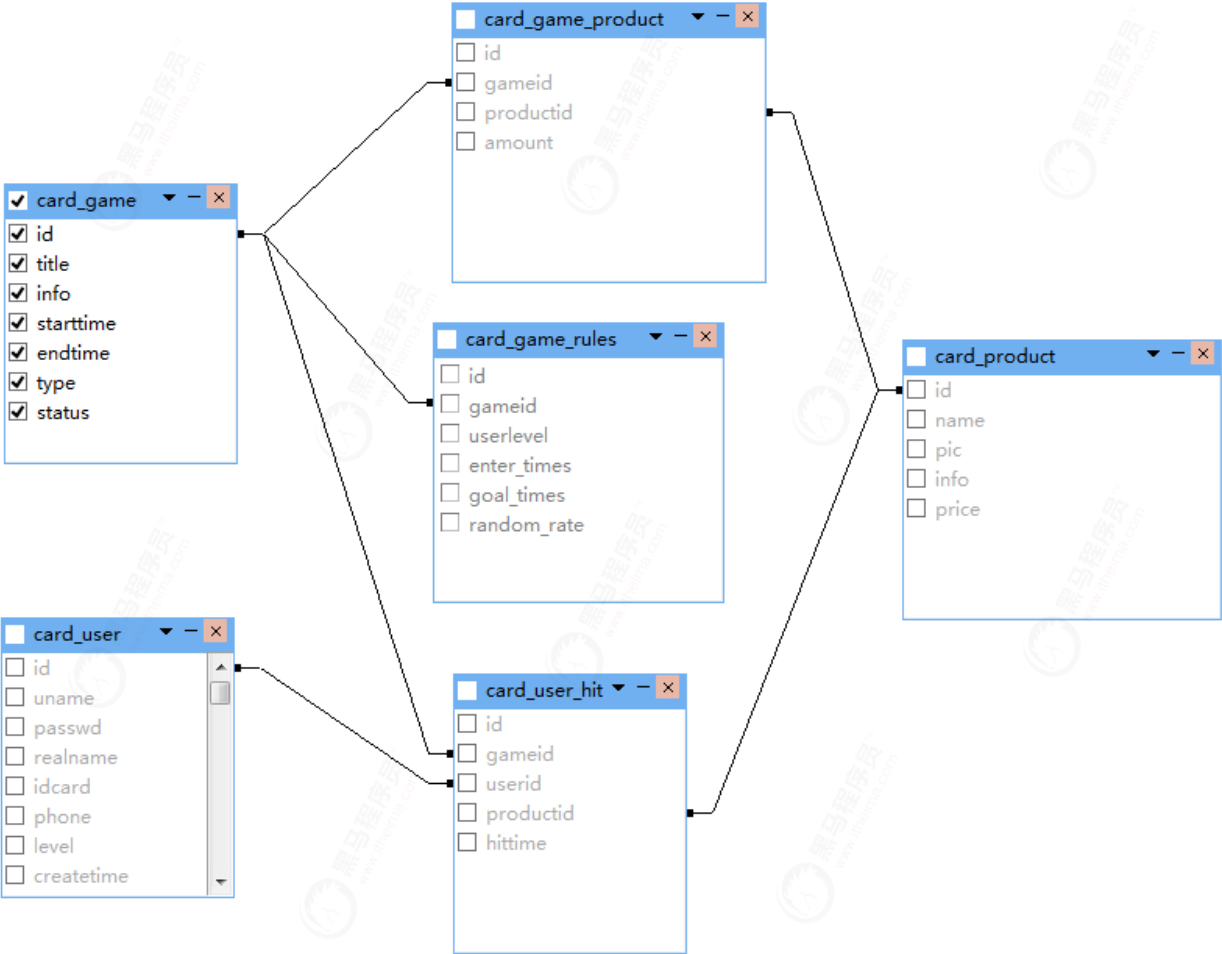
15我的奖品_没中奖



2.系统设计

2.1建模

2.1.1 ER图



2.1.2 数据表

1) 奖品表

card_product

字段	类型	备注
id	int(10) unsigned	
name	varchar(255)	奖品名称
pic	varchar(255)	图片
info	varchar(1000)	简介
price	decimal(10,2)	市场价

2) 活动表

card_game

字段	类型	备注
id	int(10) unsigned	
title	varchar(255)	活动主题
info	varchar(1000)	活动简介
starttime	datetime	开始时间
endtime	datetime	结束时间
type	tinyint(2)	类型（1=时间随机，2=瞬间秒杀，3=机会随机）
status	tinyint(1)	状态（0=新建，1=已加载）

3) 会员表

card_user

字段	类型	备注
id	int(11) unsigned	
uname	varchar(20)	用户名
passwd	varchar(50)	密码
realname	varchar(10)	姓名
idcard	varchar(18)	身份证号
phone	varchar(15)	手机号码
level	smallint(6)	等级
createtime	datetime	注册时间
updatetime	datetime	更新时间

4) 策略表

card_game_rules

字段	类型	备注
id	int(11) unsigned	
gameid	int(11) unsigned	活动id
userlevel	smallint(6)	会员等级
enter_times	smallint(6)	可抽奖次数（0为不限）
goal_times	smallint(6)	最大中奖次数（0为不限）
random_rate	tinyint(4)	用户中奖概率

5) 中奖记录

card_user_hit

字段	类型	备注
id	int(10) unsigned	
gameid	int(10) unsigned	活动
userid	int(10) unsigned	用户
productid	int(10) unsigned	奖品
hittime	datetime	中奖时间

6) 奖品活动关联

card_game_product

字段	类型	备注
id	int(10) unsigned	
gameid	int(11) unsigned	活动id
productid	int(11)	奖品id
amount	smallint(6)	数量

2.1.3 视图

1) 中奖信息

view_card_user_hit

字段	类型	备注
id	int(10) unsigned	
title	varchar(255)	活动主题
type	varchar(100)	值
uname	varchar(20)	用户名
realname	varchar(10)	姓名
idcard	varchar(18)	身份证号
phone	varchar(15)	手机号码
level	varchar(100)	值
name	varchar(255)	奖品名称
price	decimal(10,2)	市场价
gameid	int(10) unsigned	活动
userid	int(10) unsigned	用户
productid	int(10) unsigned	奖品
hittime	datetime	中奖时间

2) 奖品数统计

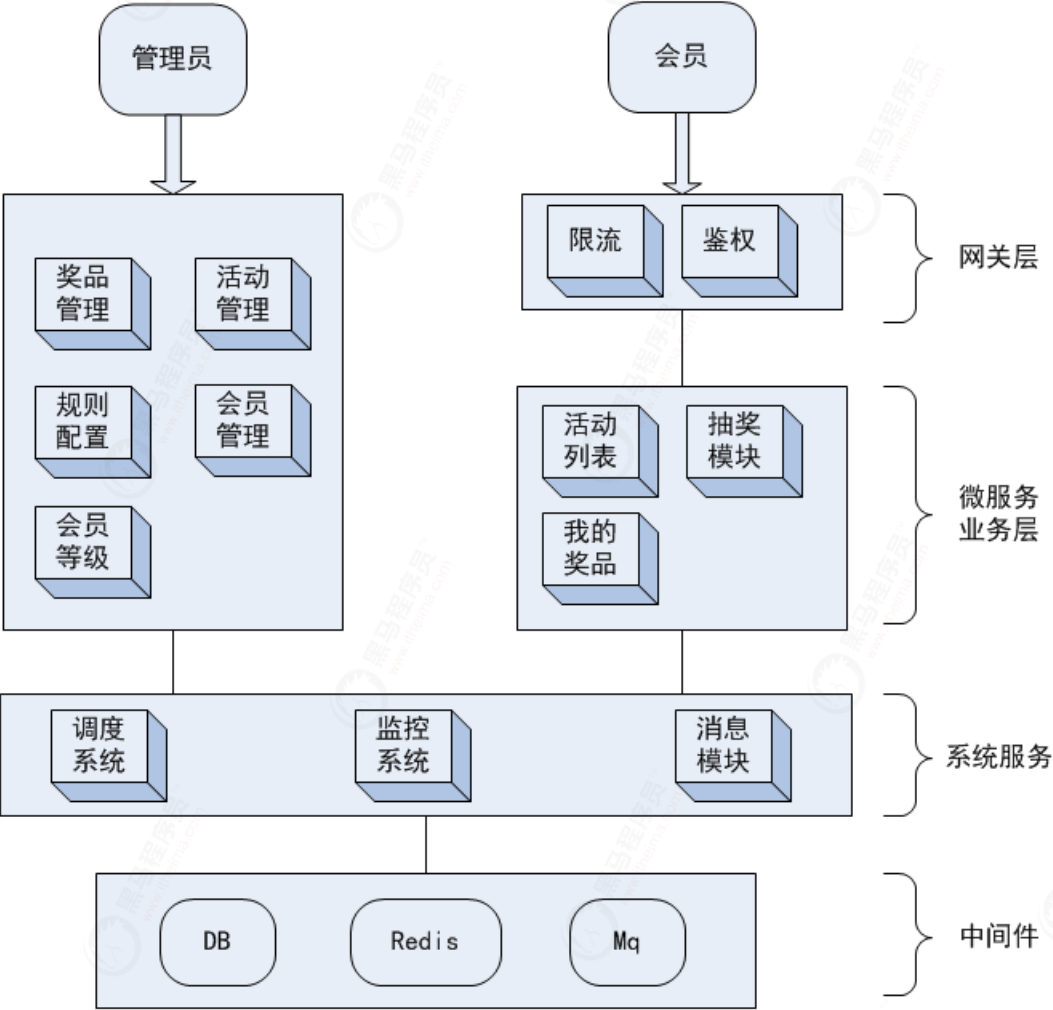
view_game_curinfo

字段	类型	备注
id	int(10) unsigned	
title	varchar(255)	活动主题
starttime	datetime	开始时间
endtime	datetime	结束时间
type	varchar(100)	值
total	decimal(27,0)	
hit	bigint(21)	

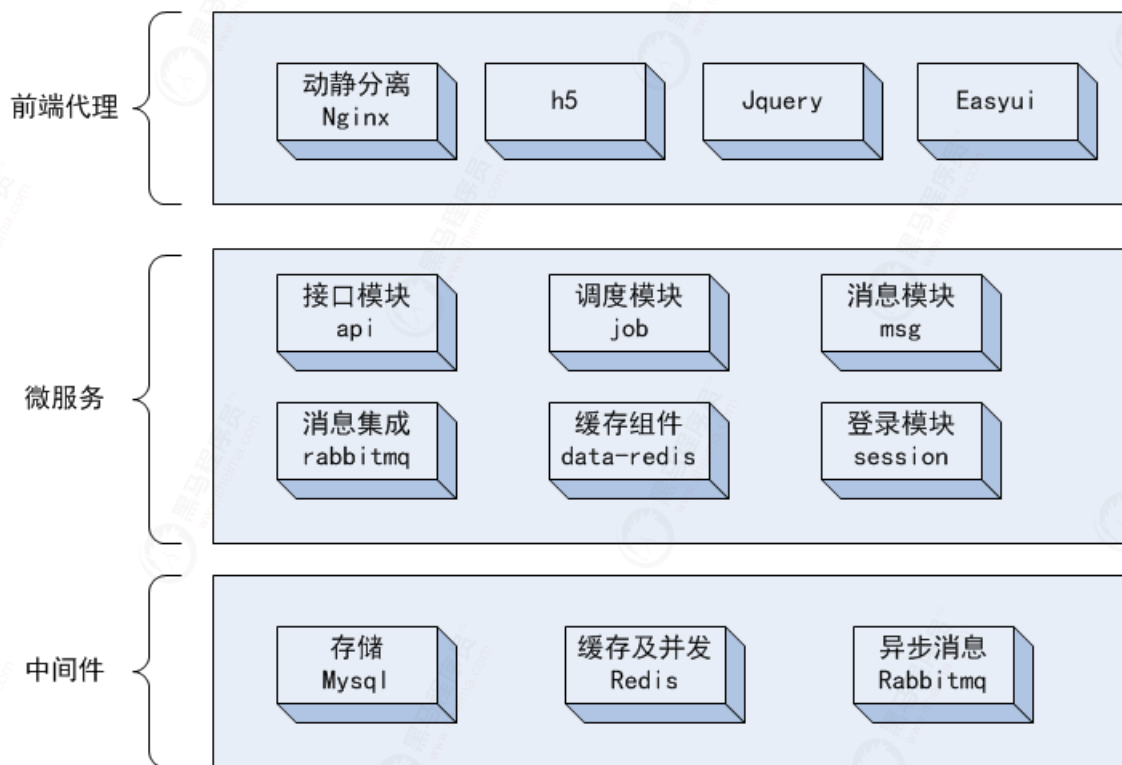
2.2概要设计

2.2.1系统拓扑

1) 业务架构



2) 软件架构



2.2.2 设计原则

开工前，可能想到的一些思想和规划.....

1) 动静分离

- 1·后台springboot启动微服务模块
- 2.静态文件分离，nginx直接响应，不要再绕后台应用机器

2) 微服务化

- 1·将模块细粒度拆分，微服务化
- 2·借助docker swarm的容器管理功能，实现不同服务的副本部署，滚动更新
- 3·在本项目中，api模块就部署了3份，以适应前端的高并发

3) 负载均衡

- 1·多个实例之间通过nginx做负载均衡，提升并发性能
- 2·本项目为大家展示的模块均部署在1台节点。生产环境涉及多台机器，用upstream实现。

4) 异步消息

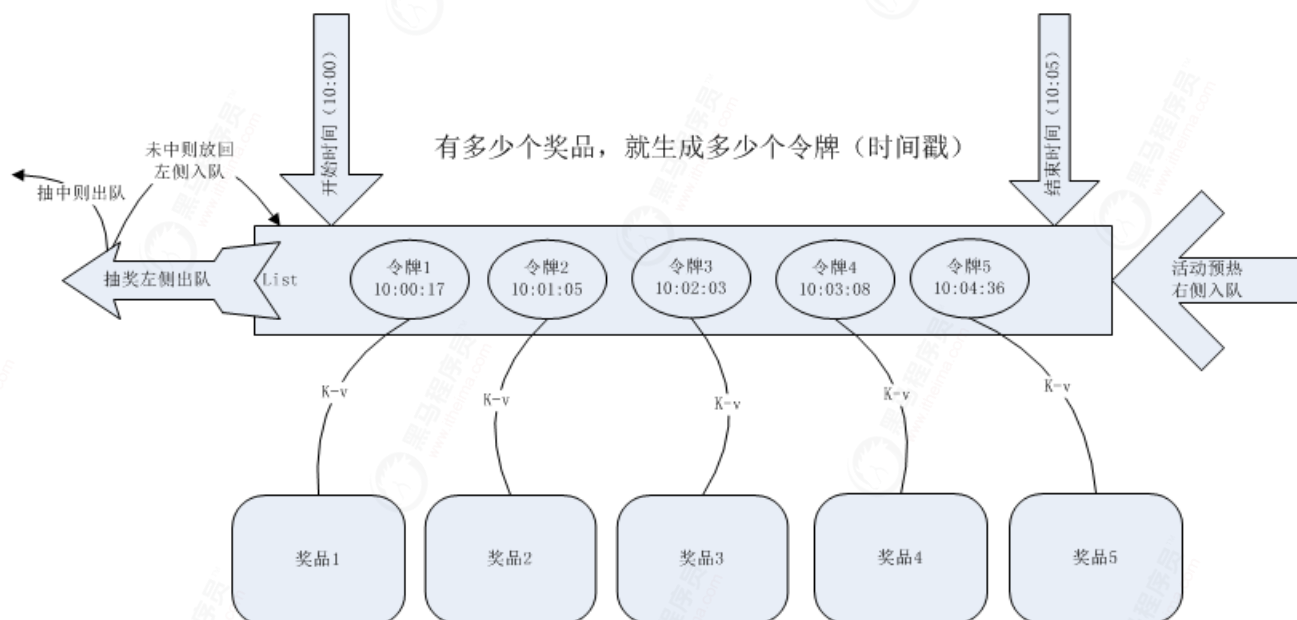
- 1·中奖后，中奖人及奖品信息要持久化到数据库。引入rabbitmq，将抽奖操作与数据库操作异步隔离。
- 2·抽奖中奖后，只需要将中奖信息放入rabbitmq，并立即返回中奖信息给前端用户。
- 3·后端msg模块消费rabbitmq消息，缓慢处理。

5) 缓存预热

- 1.每隔1分钟扫描一次活动表，查询未来1分钟内将要开始的活动。
- 2.将扫到的活动加载进redis，包括活动详细信息，中奖策略信息，奖品信息，抽奖令牌。
- 3.活动正式开始后，基于redis数据做查询，不必再与数据库打交道。

2.2.3缓存体系

开始我们的设计.....



1) 活动基本信息

k-v, 以活动id为key, 活动对象为value, 永不超时

```
redisUtil.set(RedisKeys.INFO+game.getId(),game,-1);
```

2) 活动策略信息

hset, 以活动id为group, 用户等级为key, 策略值为value

```
redisUtil.hset(RedisKeys.MAXGOAL + game.getId(),r.getUserlevel()+"",r.getGoalTimes());
redisUtil.hset(RedisKeys.MAXENTER + game.getId(),r.getUserlevel()+"",r.getEnterTimes());
```

3) 抽奖令牌桶

双端队列, 以活动id为key, 在活动时间段内, 随机生成时间戳做令牌, 有多少个奖品就生成多少个令牌。令牌即奖品发放的时间点。从小到大排序后从右侧入队。

```
redisUtil.rightPushAll(RedisKeys.TOKENS + game.getId(),tokenList);
```

4) 奖品映射信息

k-v, 以活动id_令牌为key, 奖品信息为value, 会员获取到令牌后, 如果令牌有效, 则用令牌token值, 来这里获取奖品详细信息

```
redisUtil.set(RedisKeys.TOKEN + game.getId()  
+"_"+token,productMap.get(cgp.getProductid()),expire);
```

5) 令牌设计技巧

假设活动时间间隔太短，奖品数量太多。那么极有可能产生的时间戳发生重复。

解决技巧：额外再附加一个随机因子。将（时间戳 * 1000 + 3位随机数）作为令牌。抽奖时，将抽中的令牌/1000，还原真实的时间戳。

```
//活动持续时间（ms）  
long duration = end - start;  
long rnd = start + new Random().nextInt((int)duration);  
//为什么乘1000，再额外加一个随机数呢？ - 防止时间段奖品多时重复  
long token = rnd * 1000 + new Random().nextInt(999);
```

6) 中奖计数

k-v，以活动id_用户id作为key，中奖数为value，利用redis原子性，中奖后incr增加计数。

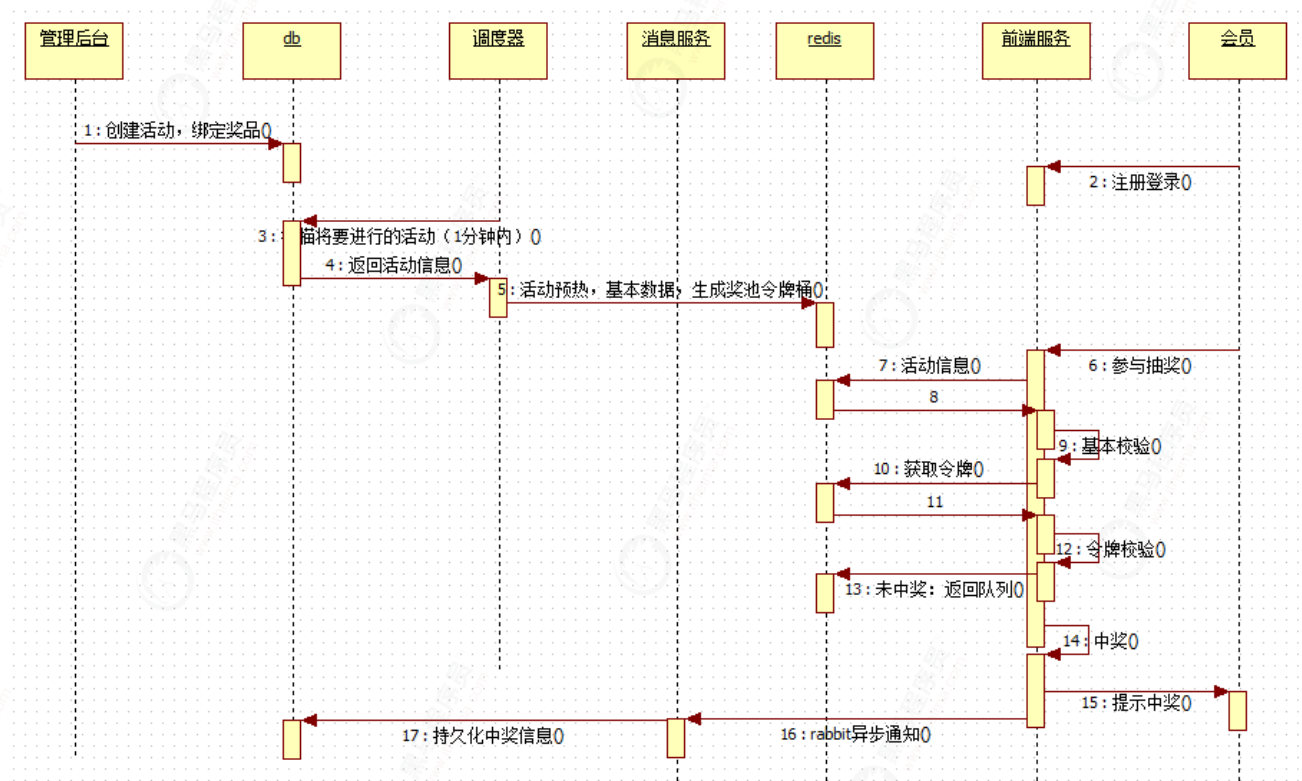
抽奖次数计数也是同样的道理

```
redisUtil.incr(RedisKeys.USERHIT+gameid+"_"+user.getId(),1);
```

7) 中奖逻辑判断

抽奖时，从令牌桶左侧出队和当前时间比较，如果令牌时间戳小于等于当前时间，令牌有效，表示中奖。大于当前时间，则令牌无效，将令牌还回，从左侧压入队列。

2.2.4业务时序图



2.3 框架选型

2.3.1 管理后台

借助开源快速开发平台，完成后台基本的增删改查。（课外拓展）

2.3.2 微服务模块

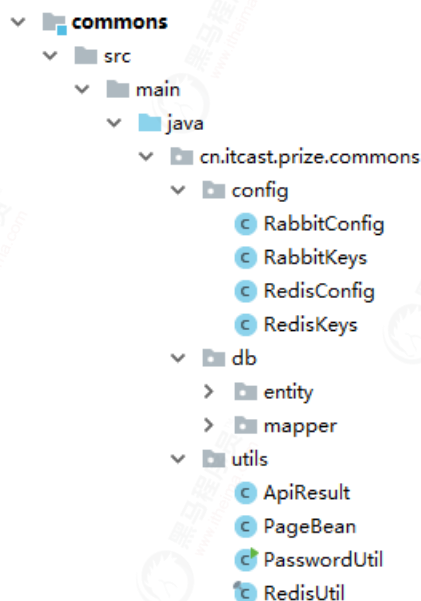
1) 公共pom

所有项目共用的依赖及版本定义。继承自springboot-2.2.4.RELEASE

- 文档: swagger2
- 数据源: mysql, mybatis, pagehelper
- 缓存: redis
- 消息队列: rabbitmq
- 工具包: fastjson, comcos-lang3, commons-collections

2) 公共模块commons

分页、密码、统一结果dto等工具类，mybatis生成的实体，mapper，redis及rabbit的配置bean



3) 接口模块api

提供给前端页面的rest接口，如：抽奖接口，活动查询接口，中奖信息接口等

注意！需要比父pom多出 spring-web, session-redis

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
  <groupId>org.springframework.session</groupId>
  <artifactId>spring-session-data-redis</artifactId>
</dependency>
```

4) 后台任务msg

- 调度任务：执行缓存预热
- 异步消息：rabbitmq消费端，接收中奖消息入库

备注（拓展）：

- 可以将job与msg剥离为两个独立微服务，利于分开部署与扩容。
- job可以集成分布式job服务，比如elastic-job

2.3.3 微服务框架集成

```
#服务名
spring.application.name=api
server.port=8001

#分布式session
spring.session.store-type=redis
server.servlet.session.timeout=36000

#以下部分，msg与api通用
spring.datasource.type=com.alibaba.druid.pool.DruidDataSource
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.datasource.url=jdbc:mysql://1.1.1.1/prize?useUnicode=true&characterEncoding=utf-8
spring.datasource.username=prize
spring.datasource.password=prize

mybatis.mapper-locations=classpath:mapper/*.xml

spring.jackson.date-format=yyyy-MM-dd HH:mm:ss
spring.jackson.time-zone=GMT+8

spring.rabbitmq.host=1.1.1.1
spring.rabbitmq.port=5672
spring.rabbitmq.username=guest
spring.rabbitmq.password=guest
spring.rabbitmq.virtual-host=/

spring.redis.host=1.1.1.1
spring.redis.port=9010
```

2.3.4 工具

2) mybatis-generator

引入坐标：

```
<dependency>
    <groupId>org.mybatis.generator</groupId>
    <artifactId>mybatis-generator-core</artifactId>
    <version>1.3.7</version>
</dependency>
```

配置pom插件：

```
<plugin>
  <groupId>org.mybatis.generator</groupId>
  <artifactId>mybatis-generator-maven-plugin</artifactId>
  <version>1.3.7</version>
  <dependencies>
    <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
      <version>5.1.31</version>
    </dependency>
    <dependency>
      <groupId>com.itheima.prize</groupId>
      <artifactId>coder</artifactId>
      <version>0.0.1-SNAPSHOT</version>
    </dependency>
  </dependencies>
  <configuration>
    <overwrite>true</overwrite>
    <outputDirectory>${project.build.directory}</outputDirectory>
    <verbose>true</verbose>
    <tableNames>%</tableNames>
  </configuration>
</plugin>
```

配置表映射关系:

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE generatorConfiguration
    PUBLIC "-//mybatis.org//DTD MyBatis Generator Configuration 1.0//EN"
    "http://mybatis.org/dtd/mybatis-generator-config_1_0.dtd">

<generatorConfiguration>
    <context id="MysqlContext" targetRuntime="MyBatis3" defaultModelType="flat">
        <property name="beginningDelimiter" value="`"/>
        <property name="endingDelimiter" value="`"/>

        <!-- 指定生成的java文件的编码,没有直接生成到项目时中文可能会乱码 -->
        <property name="javaFileEncoding" value="UTF-8"/>

        <!-- 生成的pojo, 将implements Serializable -->
        <plugin type="org.mybatis.generator.plugins.SerializablePlugin"></plugin>
        <!-- 这里的type里写的是你的实现类的类全路径 -->
        <!--<commentGenerator type="com.MyCommentGenerator">-->
            <!--<property name="suppressDate" value="true"/>-->
        <!--</commentGenerator>-->

        <commentGenerator>
            <property name="suppressDate" value="true"/>
        </commentGenerator>

        <jdbcConnection driverClass="com.mysql.jdbc.Driver"
            connectionURL="jdbc:mysql://1.1.1.1/prize?tinyInt1isBit=false"
            userId="root"
            password="root">
        </jdbcConnection>

        <javaTypeResolver type="com.JavaTypeResolverDefaultImpl">
        </javaTypeResolver>

        <!-- entity 包路径 -->
        <javaModelGenerator targetPackage="com.itheima.prize.common.db.entity"
targetProject="MAVEN">
            <property name="trimStrings" value="true"/>
        </javaModelGenerator>

        <!-- xml 路径-->
        <sqlMapGenerator targetPackage="com.itheima.prize.common.db.xml.mapper"
targetProject="MAVEN"/>

        <javaClientGenerator type="MIXEDMAPPER"
targetPackage="com.itheima.prize.common.db.mapper" targetProject="MAVEN"/>

        <table tableName="%">
            <generatedKey column="id" sqlStatement="Mysql" identity="true"/>
        </table>
    </context>
</generatorConfiguration>

```

tinyint会映射为bit，习惯上使用Integer更方便。自己定义一个type映射类并配置到上面的config xml中

```
public class JavaTypeResolverDefaultImpl implements JavaTypeResolver {
    ....

    public JavaTypeResolverDefaultImpl() {
        ....
        this.typeMap.put(-6, new JavaTypeResolverDefaultImpl.JdbcTypeInfo("TINYINT", new
        FullyQualifiedJavaType(Integer.class.getName())));
        ....
    }

    ....
}
```

3) 分页PageHelper

坐标:

```
<!--mybatis分页插件-->
<dependency>
    <groupId>com.github.pagehelper</groupId>
    <artifactId>pagehelper</artifactId>
</dependency>
```

定义一个pagebean:

```
@ApiModel("分页信息")
public class PageBean<T> {
    @ApiModelProperty(value = "当前页, 1开始")
    private Integer currentPage = 1;
    @ApiModelProperty(value = "每页条数, 默认10")
    private Integer pageSize = 10;
    @ApiModelProperty(value = "总条数")
    private Long totalNum;
    @ApiModelProperty(value = "是否有下一页")
    private Integer isMore;
    @ApiModelProperty(value = "总页数")
    private Integer totalPage;
    @ApiModelProperty(value = "开始索引")
    private Integer startIndex;
    @ApiModelProperty(value = "本页数据")
    private List<T> items;

    ...
}
```

使用:

```
// 在查询之前，调用startPage，设置页码及条数
```

```
PageHelper.startPage(curpage, limit);
```

```
List<ViewCardUserHit> all = hitMapper.selectByExample(example);
```

查询的sql会被自动拼接limit

```
2019-10-07 16:39:58.988 DEBUG 1 --- [io-8080-exec-10] c.i.p.c.d.m.V.selectByExample_COUNT : ==> Preparing: SELECT count(0) FROM view_card_user_hit WHERE (gameid = ?)
2019-10-07 16:39:58.988 DEBUG 1 --- [io-8080-exec-10] c.i.p.c.d.m.V.selectByExample_COUNT : ==> Parameters: 1(Integer)
2019-10-07 16:39:58.989 DEBUG 1 --- [io-8080-exec-10] c.i.p.c.d.m.V.selectByExample_COUNT : <== Total: 1
2019-10-07 16:39:58.989 DEBUG 1 --- [io-8080-exec-10] c.i.p.c.d.m.V.selectByExample : ==> Preparing: select id, title, type, unname, realname, idcard, phone, lev
hittime from view_card_user_hit WHERE ( gameid = ? ) limit ?,?
2019-10-07 16:39:58.989 DEBUG 1 --- [io-8080-exec-10] c.i.p.c.d.m.V.selectByExample : ==> Parameters: 1(Integer), 0(Integer), 2(Integer)
2019-10-07 16:39:58.992 DEBUG 1 --- [io-8080-exec-10] c.i.p.c.d.m.V.selectByExample : <== Total: 2
```

返回的数据体:

```

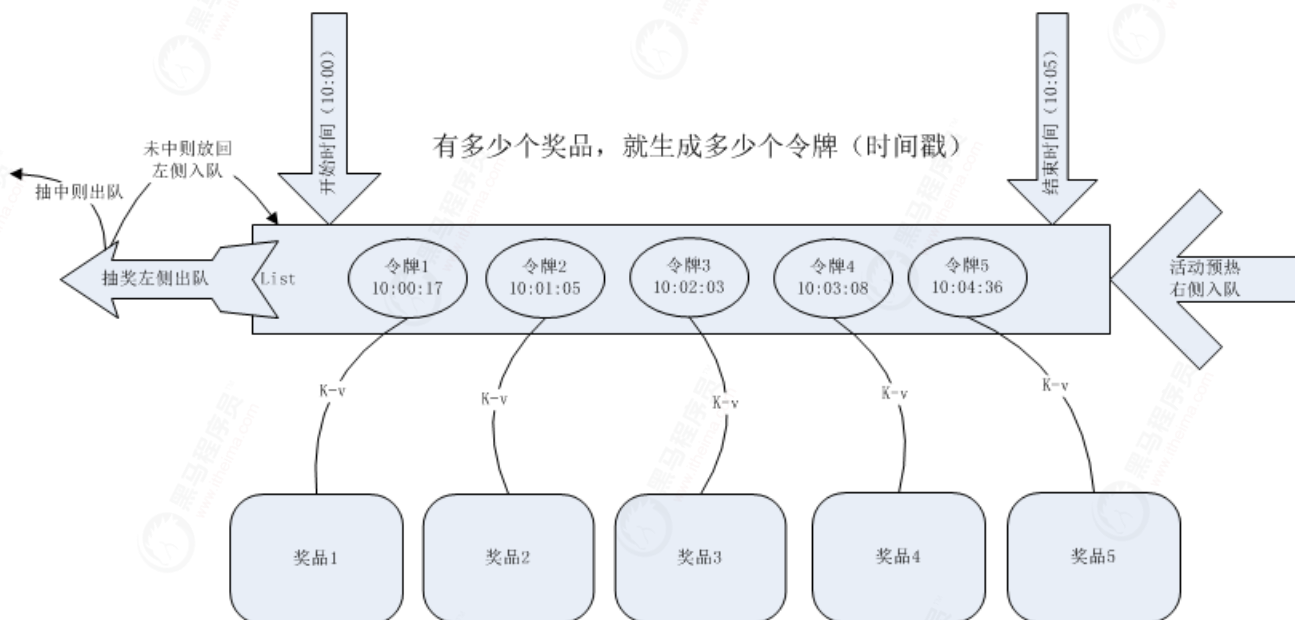
{
  "code": 1,
  "msg": "成功",
  "data": {
    "currentPage": 1,
    "pageSize": 2,
    "totalNum": 5,
    "isMore": 1,
    "totalPage": 3,
    "startIndex": 0,
    "items": [
      {
        "id": 1,
        "title": "10周年庆典",
        "type": "随机类",
        "uname": "2",
        "realname": "2",
        "idcard": "2",
        "phone": "2",
        "level": "二级会员",
        "name": "iPhoneX一部",
        "price": 5000,
        "gameid": 1,
        "userid": 8,
        "productid": 1,
        "hittime": "2019-09-24 15:26:45"
      },
      {
        "id": 2,
        "title": "10周年庆典",
        "type": "随机类",
        "uname": "2",
        "realname": "2",
        "idcard": "2",
        "phone": "2",
        "level": "二级会员",
        "name": "iPhoneX一部",
        "price": 5000,
        "gameid": 1,
        "userid": 8,
        "productid": 1,
        "hittime": "2019-09-24 15:26:46"
      }
    ]
  }
}

```

2.3.2 前端页面

h5+jquery实现抽奖界面（前端开发，非重点，了解即可）

3.代码实现



缓存结构设计回顾：

- 活动开始前1分钟扫描将要开始的活动
- 将活动信息加载进redis
- 将活动策略信息加载进redis
- 按活动奖品信息，生成对应个数的时间戳做令牌，从小到大排好序，右侧入队
- 以令牌为key，对应的奖品为value，建立映射关系，为中奖后获取奖品做准备
- 抽奖开始时，从令牌队列左侧获取令牌
- 如果令牌小于当前时间，说明中奖，找到令牌对应的奖品，抽走
- 如果令牌大于当前时间，说明未中奖，从左侧将令牌放回队列

3.1 活动预热

3.1.1 源码实现

根据缓存结构设计，初始化redis活动相关信息

3.1.2 过期时间

活动结束时间 - 当前时间 = 有效时长， 活动结束则缓存自动失效

3.1.3 调度策略

每隔1分钟扫描一遍活动表，查询未来1分钟内要开始的活动进行预热。

3.2 抽奖业务

3.2.1 源码实现

按时序图设计实现业务流程

3.2.2 并发与原子性

业务下沉到lua脚本:

```
local token = redis.call('lpop',KEYS[1])
local curtime = tonumber(KEYS[2])

if token ~= false then
    if ( tonumber(token)/1000 > tonumber(KEYS[2]) ) then
        redis.call('lpush',KEYS[1],token)
        return 1
    else
        return tonumber(token)
    end
else
    return 0
end
```

调用部分:

```
Long token = luaScript.tokenCheck("game_"+gameid,String.valueOf(new Date().getTime()));
if(token == 0){
    return new ApiResult(-1,"奖品已抽光",null);
}else if(token == 1){
    return new ApiResult(0,"未中奖",null);
}
```

3.3 中奖处理

3.3.1 rabbit配置

RabbitConfig实现rabbit的队列与消费消息配置

3.3.2 异步处理

中奖主流程中, 将中奖信息放入消息队列, 立刻返回结果响应前台。

```
//投放消息给队列, 中奖后的耗时业务, 交给消息模块处理
Map map = new HashMap(2);
map.put("gameid",gameid);
map.put("userid",user.getId());
map.put("productid",product.getId());
map.put("hittime",now.getTime());
rabbitTemplate.convertAndSend(RabbitKeys.EXCHANGE_DIRECT,RabbitKeys.QUEUE_HIT, map);
```

3.3.3 msg模块消费

```
@RabbitHandler
public void processMessage3(Map message) {
    logger.info("user hit : " + message);
    CardUserHit hit = new CardUserHit();
    hit.setGameid(MapUtils.getIntValue(message, "gameid"));
    hit.setUserid(MapUtils.getIntValue(message, "userid"));
    hit.setProductid(MapUtils.getIntValue(message, "productid"));
    hit.setHittime(new Date(MapUtils.getLongValue(message, "hittime")));
    // 入库
    hitMapper.insert(hit);
}
```

3.4缓存监控

3.4.1 客户端工具

查看缓存信息，可以借助redis-manager客户端工具，便捷，但日期格式不够直观

3.4.2 编码查看

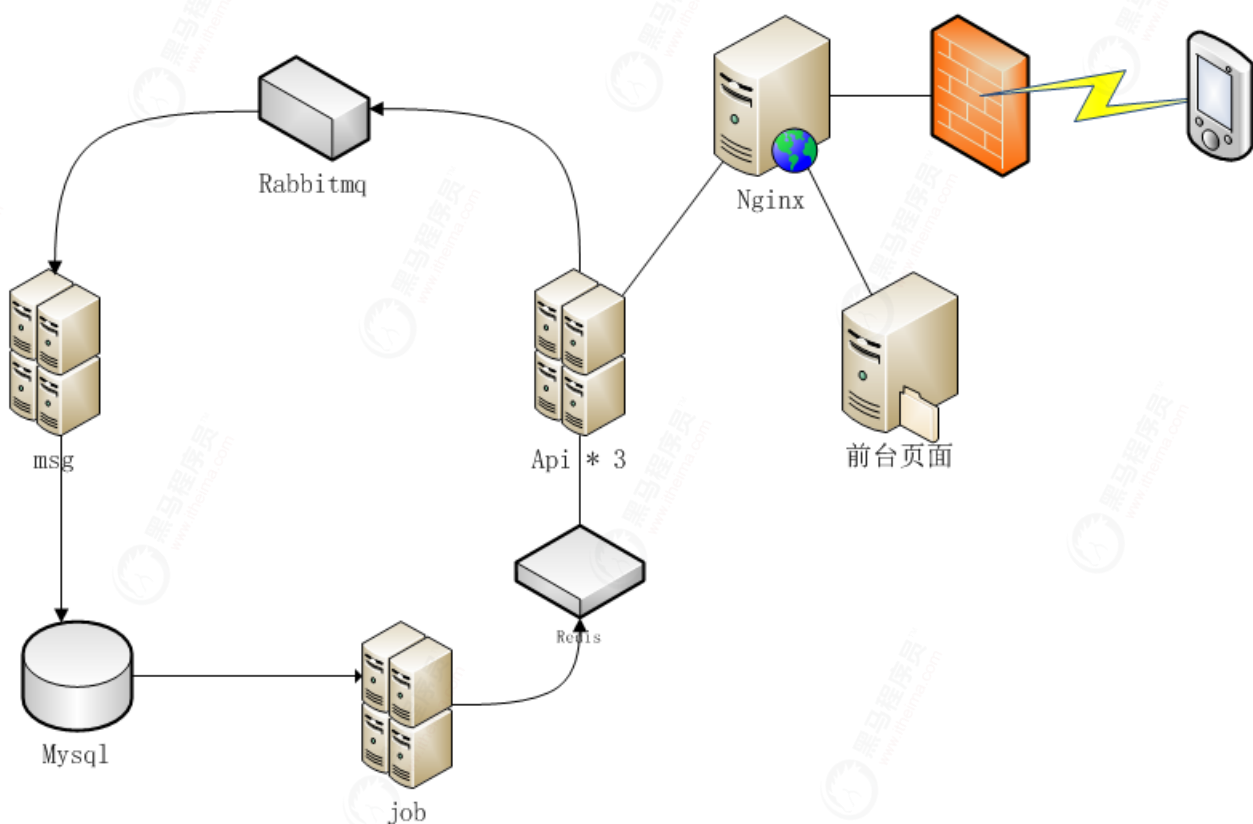
写rest接口，通过swagger查看，需要编码，但是可以自由格式化输出json

3.5代码启动

- 启动中间件：mysql，redis，rabbitmq
- 启动msg / job
- 启动api
- 启动管理后台，创建活动
- 查看job日志，是否在开始前的1分钟，将活动预热成功
- 请求抽奖接口，查看是否提示活动未开始
- 开始时间后，请求抽奖接口，调试api，看是否可以正常抽奖
- 中奖后，看msg日志，是否有中奖消息发送到消息模块
- 查看数据库，确认中奖信息是否正常写入db

3.6上线部署

3.6.1 部署拓扑



3.6.2 镜像

1) 脚本

```
docker build -t $app:$BUILD_NUMBER .
```

2) Dockerfile

build阶段需要的基础镜像及模板。

```
FROM daocloud.io/library/java:openjdk-8u40-jdk
#语言字符，解决乱码问题
ENV LC_ALL C.UTF-8
ENV LANG C.UTF-8
ENV LANGUAGE C.UTF-8
#时区及时间，不设置会影响活动的开始结束
ENV TZ=Asia/Shanghai
RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ > /etc/timezone
#将springboot的jar包打入镜像
COPY ROOT.jar /opt/ROOT.jar
WORKDIR /opt
#启动命令，注意环境配置 prod
CMD ["java", "-jar", "ROOT.jar", "--spring.profiles.active=prod"]
```

```
docker service rm $app
```

```
docker service create --name $app -p $port:8080 --hostname localhost $app:$version
```

3.6.3 中间件部署

redis, mysql, nginx 均采用docker方式部署

#获取镜像

```
docker pull daocloud.io/library/redis:latest
docker pull daocloud.io/library/rabbitmq:3.6.10-management
docker pull daocloud.io/mysql:5.7.4
docker pull daocloud.io/nginx:latest
```

#启动镜像

```
docker run --name redis -p 6379:6379 -d daocloud.io/library/redis

docker run --name mysql -v /opt/data/mysql:/var/lib/mysql -p3306:3306 -e
MYSQL_ROOT_PASSWORD=password -d daocloud.io/mysql:5.7.4

docker run -d --hostname my-rabbit --name rabbit -p 15672:15672 -p 5672:5672
daocloud.io/library/rabbitmq:3.6.10-management

docker run --name nginx -v /opt/data/nginx/html:/usr/share/nginx/html:ro -v
/opt/app/back/upload:/usr/share/nginx/upload:ro -v
/opt/data/nginx/nginx.conf:/etc/nginx/nginx.conf:ro -p 80:80 -d daocloud.io/nginx
```

3.6.4弹性扩容

采用微服务架构的初衷之一就是实现服务单元的快速独立扩容，借助docker-swarm的功能，变得很容易实现。以本项目为例，api模块在部署完成后，扩容到3个实例，以负载前端的并发请求：

```
docker service scale api=3
```

```
[root@iZ8vb3a9qxfwannyw16zZ app]# docker service ls
```

ID	NAME	MODE	REPLICAS	IMAGE	PORTS
v71cn5elmny4	api	replicated	3/3	api:10	*:9004->8080/tcp
ihzcd9oiigvj	eureka	replicated	1/1	eureka:23	*:9001->8080/tcp
p5xme0uf9czb	job	replicated	1/1	job:12	*:9002->8080/tcp
6zyc6nbu9mcn	msg	replicated	1/1	msg:10	*:9003->8080/tcp

```
[root@iZ8vb3a9qxfwannyw16zZ app]# docker service ps api
```

ID	NAME	IMAGE	NODE	DESIRED STATE
k880w0s5bn7z	api.1	api:10	iZ8vb3a9qxfwannyw16zZ	Running
nw8kejxhvkqk	api.2	api:10	iZ8vb3a9qxfwannyw16zZ	Running
q32v40vpkxso	api.3	api:10	iZ8vb3a9qxfwannyw16zZ	Running

3.6.5 管理后台部署

因为是开发平台，打出的war包，这里使用jetty启动即可

```
java -jar /opt/app/jetty-runner.jar --port 7070 /opt/app/back/
```

3.6.6 静态页面与nginx

/api的请求代理到微服务api模块，其他静态文件由nginx本地访问，配置如下：

```
http {  
  
    include        mime.types;  
    default_type   application/octet-stream;  
    sendfile        on;  
    keepalive_timeout 65;  
    server {  
        listen      80;  
        server_name localhost;  
        location ^~ /api/ {  
            proxy_pass http://172.17.0.1:9004;  
        }  
        location ^~ /upload/ {  
            alias /usr/share/nginx/upload/;  
        }  
        location / {  
            root /usr/share/nginx/html;  
        }  
    }  
}
```

3.7 发散思维与总结

3.7.1 发散思维（拓展）

1) lua脚本的运用

使用lua脚本，将抽奖的逻辑从java端移入redis服务器端，作为一个整体函数暴露给java调用，减少了java服务器与redis服务器之间的通信次数，性能会得到提升。

2) 怎么实现活动暂停功能？

要实现活动随时暂停，可以新增一个接口，该接口修改redis缓存中的活动状态。

抽奖接口逻辑中增加暂停状态判断。如果是暂停，返回给前台以提示。

3) 怎么实现多种投放策略？

投放策略即令牌的生成策略不同。可以修改令牌生成部分代码。按递增，递减，正态分布等多种函数生成时间戳。

3.7.2 总结

通过本项目我们接触到了以下知识点：

- 如何分析需求做软件架构设计
- 中间件的运用，rabbitmq，尤其是redis的数据结构
- 一些工具：分页PageHelper，swagger2，mybatis-generator
- 微服务思想及持续集成，动态扩容
- 基于docker swarm的发布与部署