说明：普通红包是指金额每份金额固定的红包包括群普通红包和个人普通红包，个人普通红包也就是红包个数为1的群普通红包。

　　1 需求分析

　　一个字：钱;两个字：消遣

　　1.1用户为什么要发红包?

　　(1)逗别人玩自己开心

　　有些人发一些1分钱的红包，看到大家哄抢，自己觉得很爽;有些人自己发1个0.01的自己抢和别人比拼速度，这些无聊的人追求的是娱乐性，如同黑白快、2048等，满足无聊的人消耗时间就可以了。

　　(2)成为焦点人物

　　当你经常在群里发红包的时候，你就会成为「群明星」，让更多人认识你，和你说话，你有一种自己朋友遍天下的错觉，然而并没有什么卵用，人家是冲着你的钱来的。所以就有了【我发的红包总数】

　　【红包被抢提醒】

　　(3)获得关注卖广告

　　单纯发一个广告不但没有人看，而且会引起反感。但是你发个大红包，群里面的人会喜闻乐见，而且很亲切的问你项目的相关内容或者帮你填写调查问卷。不过后来小红包广告渐渐失去吸引力，因为大家的兴奋阈值提高了，而且重复的东西是不可能让用户持续高潮的，如同你xxoo的时候不能总是使用一个体位一样。因此对于服务号的摇一摇红包和关注红包，最少也有2元，而且还能裂变给你好友。2块钱是什么?官方的说法是一张彩票的价格，一份希望，2元彩票长期已经在广大群众心理建立了一个阈值，一份希望的阈值，『万一实现呢?』这种心理。而且这种希望还能传递给别人(裂变红包)，何乐而不为呢?

　　(4)纯粹是一种祝福

　　有时候是我们产品经理想太多了，人家可能仅仅是把传统的红包以微信红包这种新颖的方式发出来而已。以前只有结婚的人才能发红包，现在可以全民发，我也可以给朋友带去一份祝福。不过包多少呢?这个很让人纠结，太少不体面;给『上层』的人发红包，人家阈值高，太多支付不起。于是就有了【随机红包这个东西了】

　　1.2 用户为什么要抢红包?

　　(1)好玩刺激

　　这个理由还是留给那些无聊的人，不解释，跟『为什么要发红包?』的第一点差不多。那个【最佳手气】，就刺激了人们玩红包接龙(不是某宝那个坑爹的红包接龙…)

　　(2)贪婪——人类原始的欲望

　　在法律和道德重重的压制下，深深地掩盖了人的本性，当有这种合法而且又光明正大的”抢钱”，即使是0.01，也足以让人们压制的贪欲井喷，造就了微信红包的繁荣。虽然官方说，摇红包是为了让老一辈了解我们的世界，了解我们的生活，让我们过年回家能一起摇红包。可是从朋友圈，从新闻，大家在群里的反应，我一点都没感觉到红包让家人团聚在一起，不知道那是否是公关说辞，这里不作评价，而对人性的激发确是彻彻底底的。正如所有的自然科学最终都会回归到哲学问题上，所有的产品也终究要回归到对人性的思考上。(废话说太多了~~~)

　　(3)炫耀

　　证明自己单身20年，哦!不对，应该是证明自己手速快(不都是一个意思嘛，废话真多!)有的人无聊到自己发自己抢，以此在炫耀自己的4G网络、光纤还有…麒麟臂。

　　(4)减少损失

　　很多人发了拼手气红包觉得自己钱包大出血，于是自己又抢了一把，希望自己抢到大一点的金额，相当于发少一点红包。

　　1.3 为什么要晒红包?

　　(1)炫富心理——我发出的红包统计页面

　　(2)攀比心理——红包结果页面

　　这里就不多作解释了，想想你为什么喜欢在朋友圈发东西就懂了。

　　2 入口



　　入口主要分为两大类：聊天窗口和微信钱包。



　　2.1钱包

　　在钱包添加红包入口，是因为用户首次使用时，一般是先收到别人的红包，自己要取钱，那么去哪里取呢?肯定是钱包，钱包需要集合所有跟钱有关的概念，给用户一个深刻的印象，类似于OmniFocis的透视功能(不知道就算了…)，需要把相关内容聚合到一个入口。所以钱包聚合了和钱相关的内容，比如信用卡还款、手机充值、理财通等，用户第一次进来收钱的时候，自然就看到微信红包并进一步引导用户绑定信用卡和发红包。另外如果用户第一次使用红包不是收到别人红包，而是听说有红包这一功能或者看到别人发红包，他首先会想到钱相关的东西应该在红包里面。



　　2.2 聊天窗口

　　而这聊天窗口中加入红包入口则是更加简单粗暴，用户在过年或者平常使用会经常点开”+”发送图片，这就很容易会见到红包入口了，而且微信在过年的时候特意把红包按钮用红色高亮显示了，这就更加容易被用户发现，从而提高入口转换率。这里还有个逻辑，在单聊和群聊所进入的红包页面的不同的，如下图所示：



　　群红包默认为拼手气群红包而不是定额红包，为什么要酱紫呢?先看看拼手气群红包的优势：

　　1、金额随机，时大时小的金额能给用户惊喜;

　　2、可以看到其他用户抢了多少，引起攀比心理(这次抢得不爽，下次一定要抢个大大哒);

　　3、产生很多新奇玩法，比如手气最佳的发3倍金额;

　　4、由于可以看到谁抢了红包，所以群里面总是抢第一又不参与游戏的人基本上是使用外挂软件，群成员自发的要求群主踢掉这个人，这种众包式的”反外挂”比微信自己使用技术手段去解决更加节省成本和更加有效。

　　至于拼手气群红包相对于普通红包的劣势就是需要大量的计算资源去计算红包的随机金额，不过对于腾讯那么厉害的架构师和财力，这些计算资源算不了什么，反而消耗这些资源去拉起微信群的活跃度和拉高同时在线人数让财报好看点更加划算。

　　2.3 摇一摇

　　微信在春节前还额外的增加了摇一摇的红包入口，给一直被认为”约炮神器”的摇一摇洗白(哈哈，开玩笑啦!)。加速度传感器也很具有互动性，却一直隐藏在手机里面，使用率不如摄像头和话筒。摇一摇红包很好的利用了每个人手机里面”雪藏”的硬件。在”红包肯定是要发的”和”摇一摇功能的代码本来就有”的两大前提下，增加摇一摇红包这功能并不会增加多少开发量和成本，因为既然红包要发，无论用什么形式发，后台的负载均衡和高并发分流是肯定要做的，摇一摇的代码在约pao的时代就已经很完善了基本上不用改，接入红包的逻辑就可以了，所以机会成本很低。那为什么一定要发红包呢?因为上一年已经带坏头了，示范效应导致支付宝也来分一杯羹，微信能不发吗?

　　3 界面

　　3.1发红包页面

　　在单人聊天窗口进入的普通(定向)红包的页面只需要输入红包金额和祝福语，点击【塞钱进红包】，如果已经绑定银行卡，则调起对话框浮层【输入密码】;如果未绑定银行卡则跳转到零钱支付页面，点击按钮【使用零钱支付】即可，无需输入密码，在这个过程中，如果零钱不足，则会跳转到输入银行卡号的页面，点击【下一步】之后需要接着输入姓名、银行预留手机号和短信验证码，填写完成后即可用银行卡支付。



　　这里不得不提的是一个非常人性化的设计，当点击”改完普通红包”，从群手气红包切换到普通红包的过程中，已经输入的内容不会丢失，红包个数不变，此时的单个金额EditView(安卓UI控件)中的值会由  
总金额/红包个数  
得出并自动填充;当点击”改为群手气红包”，从普通红包切换到群手气红包的过程中，已经输入的内容不会丢失，红包个数不变，此时的总金额EditView中的值会由  
单个金额\*红包个数 计算出并自动填充，，不用用户重新输入，非常贴心。这也是微信”将用户体验做到极致”的地方之一。

　　3.2红包『抢』页面

　　聊天窗口会显示出红包样式的聊天消息，点击红包后会出现拆的页面。



　　3.3红包『拆』页面

　　点击按钮【拆】之后，那坨黄色的东西会转(用几帧图片切换形成的动画，在IOS上比Android上运行起来更加流畅)，那坨东西转完之后页面会跳转到【红包结果页面】。值得一提的是安卓最新版本中将Html版本的红包换成了安卓原生红包界面，为什么这么做呢?

　　一是微信惯用的牺牲客户端资源(CPU、内存、储存卡容量)去换取服务器端的稳定和减少资源投入的策略，页面资源放在本地，这样子web前端服务器容量就可以减少投入，同时也可以减少客户端对资源服务器的访问量。类似的，微信的聊天记录是默认不存储在服务器端的，而是将各种图片语音小视频全部塞到你手机的内存里面，微信表情在6.0版本之前也是不保存到服务器的。

　　二是以往基于web的红包页面经常会出现”妈的页面还在loading红包就没了””红包来了却连不了网是怎样一种体验”等等的用户抱怨，而原生的页面因为放在本地不需要远程加载，只需要传输简单的红包ID，发送者等少量信息即可通知客户端显示红包页面，可以减少联网时间和降低网络状况对抢红包的体验流畅度，让用户抢不到红包都不会觉得是因为微信没优化好，而是自己太幸福  
(没单身的手速慢，哈哈)。下图为几种红包”拆”页面(大家来玩找不同，嘻嘻)：



　　那这四个页面分别会在什么时候出现呢?在5.2中会做详细的介绍。

　　3.4红包结果页面

　　红包结果页面会显示抢到红包的人的列表，其中金额最大的为手气最佳。当有两个或者以上金额相同的时候，以时间最早的一个为最佳手气。页面还会显示发红包的人极其昵称、你自己领到的金额(如果没领到就不会显示)，零钱入口和转发该红包的入口、我的红包记录入口。红包结果页面也有很多种，详见本文的5.3部分.



　　3.5摇一摇红包



　　摇一摇红包和企业红包的随机方法和群手气红包大同小异，由于没有接触过企业红包的发放流程，这里不多说。

　　为什么要有剩余红包个数呢?

　　引用鹏飞在人人都是产品经理举办的产品经理大会广州站上说的一句话”给用户一个预期，现在还有没有红包，还有多少，而且这个数字必须准确，不能忽悠用户。有些朋友和我说，他们就是在最后几秒摇到的。所以，要让用户为希望而摇，为了希望，把手摇断，又算什么!”。没错，这个数字是”准确”的，但是他并不是实时的。因为过于频繁刷新的数字少量减少，不仅用户没有感知，不停的访问数据库剩余红包数对于服务器也是极大压力，所以推测微信是采用这种策略：每减少1个单位(比如说50W)的红包数量，自动将这个值写入缓存服务器，用户摇红包的时候都直接访问缓存，而且不是每次摇都访问剩余数，而是摇n次之后(比如摇了5次)才去请求一次剩余红包数，这样就把传递到服务器的压力减少n倍。

　　上图最后那个页面你没见过?

　　微信官方说，当服务器压力过大的时候，唤起让用户休息一下这个页面。这里我提出另外一种策略，也许微信也采用了这种策略：当用户摇一摇请求红包时，服务器压力过大，网络阻塞或者队列已满等异常情况下，会直接通知客户端”你没有抢到”，也就是直接返回那个摇红包的页面进行下一次的摇一摇动作，这样子永远也不会显示那个”休息一下”的页面。

　　4后台

　　4.1数据库

　　以下关系型数据库设计的字段是基于少量请求下，我们模拟红包系统的可行方案，并没有考虑高并发、分库分表以及缓存的情况，关于这部分内容可以查看本文4.4部分整理一些大神的回答作为了解。

　　(1)用户信息数据表user\_info

　　userID、红包ID、祝福语、红包类型、红包个数、红包金额、超时

　　(2)用户钱包数据表user\_wallet

　　userID、money、银行卡ID等其他字段

　　(3)发送红包数据表red\_send

　　红包ID、senderID、红包个数、红包金额、祝福语、最佳手气、发出时间

　　(4)接收红包数据表red\_receive

　　红包ID、receiver、接收时间、接收金额

　　4.2随机算法

　　很多人说红包序列是预先在手机发出去的时候已经产生好随机序列，其实这样会产生大量的数据库读写操作，内存读的速度以DDR3-2400为例，能达到17G/s，写的速度达到18G/s(参考文献：http://m.it168.com/article\_1410707\_p5.html)。而硬盘数据库的读写速度最多达到133MB/s。可见大量的从硬盘读写数据不但容易使硬盘损坏，更达不到高并发的读写需求。所以预先生成随机序列写入数据库，用户抢的时候再读出红包金额并将用户信息写入数据库并不科学。所以采用内存实时计算随机序列并异步写入硬盘数据库储存的方法。基于内存的随机序列是伪随机序列，他并不是真正的随机，而是根据种子通过一定的算法计算出来的值，只要种子不变，每次计算出来的值的序列是一致的。也就是说当红包指纹(ID或者ID+时间戳或者其他算法生成)一定时，计算出来的序列是一致的，这样子就不用储存在数据库，而是实时计算，第一次取序列的第一个值，第二次取序列的第二个值，如此类推。(更详细的说明可以参考http://www.open-open.com/lib/view/open1430473257443.html)。具体步骤如下(代码以python举例子，没办法知道人家后台用什么语音写的)：

　　以红包ID为种子

　　>>>red\_ID = 1775509988475009

　　>>>random.seed(red\_ID)

　　群手气红包的最小值为0.01，摇一摇红包的最小值为2.00

　　>>>min = 1.00

　　>>>if (红包为群手气红包):

　　min = 0.01

　　else(红包为摇一摇红包):

　　min = 2.00

　　群手气红包的最大值为剩余红包总额和个数的商的2倍(你可以在群里不停地发红包做回归，记得叫上我去拿红包，哈哈)。

　　>>>max = (remain\_money/remain\_num)\*2

　　而摇一摇红包官方给出的计算公式是剩余金额/剩余红包数\*n

　　n主观猜测也是等于2，在这公司基础上再人为控制概率。

　　方案一：

　　人为干扰概率的，有人拿到京东618元的红包，动脑子想想，京东店庆是618，这个金额绝对不是随机出来的，而是设定好金额，然后每个金额范围都有一定的概率。

　　比如说2元—5元概率为85%;5元—20元概率为10%，20元—50元概率为4.99%，618元概率为0.01%。(概率仅作参考，因为样本量太大，官方也没提供数据，这里只是提供其中一种可行的方案，以下代码也只是提供思路，与实际可运行的代码略有差别)

　　>>>a = random.uniform(0,1)

　　>>>b,\_max,\_min = 0

　　>>>if a < 0.85:

　　\_min = 2.00

　　\_max = 5.00

　　>>>elif a < 0.95 & a >= 0.85:

　　\_min = 5.00

　　\_max = 20.00

　　>>>elif a < 0.9999 & a >= 0.95:

　　\_min = 20.00

　　\_max = 50.00

　　>>>elif a > 0.9999:

　　\_min = 618.00

　　\_max = 618.00

　　>>>random.uniform(min,max)

　　方案二：

　　\_min = 2.00

　　\_max = 剩余金额/剩余红包数\*n

　　人为放出618元的彩蛋红包，并且用上述方法设置概率为0.0001%

　　4.3红包发出去那一刻发生了什么?

　　这一部分由于个人的水平限制，未能给出有深度的简介，这里为了文章的完整性，借用胖胖的文章作为说明(胖胖的博客为www.phppan.com)

　　(1)发红包后台操作：

　　在数据库中增加一条红包记录，存储到CKV，设置过期时间;

　　在Cache(可能是腾讯内部kv数据库，基于内存，有落地，有内核态网络处理模块，以内核模块形式提供服务))中增加一条记录，存储抢红包的人数N

　　(2)抢红包后台操作：

　　抢红包分为抢和拆，抢操作在Cache层完成，通过原子减操作进行红包数递减，到0就说明抢光了，最终实际进入后台拆操作的量不大，通过操作的分离将无效请求直接挡在Cache层外面。这里的原子减操作并不是真正意义上的原子减操作，是其Cache层提供的CAS，通过比较版本号不断尝试，存在一定程度上的冲突，冲突的用户会放行，让其进入下一步拆的操作，这也解释了为啥有用户抢到了拆开发现领完了的情况。

　　拆红包在数据库完成，通过数据库的事务操作累加已经领取的个数和金额，插入一条领取流水，入账为异步操作，这也解释了为啥在春节期间红包领取后在余额中看不到。拆的时候会实时计算金额，其金额为1分到剩余平均值2倍之间随机数，一个总金额为M元的红包，最大的红包为  
M \* 2 /N(且不会超过M)，当拆了红包后会更新剩余金额和个数。财付通按20万笔每秒入账准备，实际只到8万每秒。

　　4.4 Q&A若干整理(这一部分是网上整理的，不知道如何分类比较好就放在一起了)

　　①既然在抢的时候有原子减了就不应该出现抢到了拆开没有的情况?

　　这里的原子减并不是真正意义上的原子操作，是Cache层提供的CAS，通过比较版本号不断尝试。

　　②cache和db挂了怎么办?

　　主备 +对账

　　③有没有红包个数没了，但余额还有情况?

　　没有，程序最后会有一个take all操作以及一个异步对账保障。

　　④为什么要分离抢和拆?

　　总思路是设置多层过滤网，层层筛选，层层减少流量和压力。这个设计最初是因为抢操作是业务层，拆是入账操作，一个操作太重了，而且中断率高。  
从接口层面看，第一个接口纯缓存操作，搞压能力强，一个简单查询Cache挡住了绝大部分用户，做了第一道筛选，所以大部分人会看到已经抢完了的提示。

　　⑤抢到红包后再发红包或者提现，这里有什么策略吗?

　　大额优先入账策略

　　⑥有没有从数据上证明每个红包的概率是不是均等?

　　不是绝对均等，就是一个简单的拍脑袋算法。官方已经在产品经理大会上说明这是个拍脑袋的算法了。

　　⑦发红包人的钱会不会冻结?

　　是直接实时扣掉，不是冻结。

　　⑧采用实时算出金额是出于什么考虑?

　　实时效率更高，预算才效率低下。预算还要占额外存储。因为红包只占一条记录而且有效期就几天，所以不需要多大空间。就算压力大时，水平扩展机器是。详见本文4.2的说明。

　　⑨实时性：为什么明明抢到红包，点开后发现没有?

　　答：2014年的红包一点开就知道金额，分两次操作，先抢到金额，然后再转账。

　　2015年的红包的拆和抢是分离的，需要点两次，因此会出现抢到红包了，但点开后告知红包已经被领完的状况。进入到第一个页面不代表抢到，只表示当时红包还有。详见本文Jinkey在第五部分的说明。

　　⑩红包的设计

　　答：微信从财付通拉取金额数据过来，生成个数/红包类型/金额放到redis集群里，app端将红包ID的请求放入请求队列中，如果发现超过红包的个数，直接返回。根据红包的逻辑处理成功得到令牌请求，则由财付通进行一致性调用，通过像比特币一样，两边保存交易记录，交易后交给第三方服务审计，如果交易过程中出现不一致就强制回归。

　　⑪并发性处理：红包如何计算被抢完?

　　答：cache会抵抗无效请求，将无效的请求过滤掉，实际进入到后台的量不大。cache记录红包个数，原子操作进行个数递减，到0表示被抢光。财付通按照20万笔每秒入账准备，但实际还不到8万每秒。

　　⑫如何保持8w每秒的写入?

　　答：多主sharding，水平扩展机器。

　　⑬查询红包分配，压力大不?

　　答：抢到红包的人数和红包都在一条cache记录上，没有太大的查询压力。

　　⑭一个红包一个队列?

　　答：没有队列，一个红包一条数据，数据上有一个计数器字段。

　　⑮每领一个红包就更新数据么?

　　答：每抢到一个红包，就cas更新剩余金额和红包个数。

　　⑯红包如何入库入账?

　　数据库会累加已经领取的个数与金额，插入一条领取记录。入账则是后台异步操作。

　　⑰入帐出错怎么办?比如红包个数没了，但余额还有?

　　答：最后会有一个take all操作。另外还有一个对账来保障。

　　5交互

　　5.1前后端交互时序

　　(1)绑定银行卡



　　(2)收发群手气红包



　　① 发起红包操作

　　② 银行扣款逻辑，不成功则返回，成功则进行下一步

　　③ 请求将红包写入数据库某个set，并获取红包ID返回客户端

　　④ 长连接通知客户端成功

　　⑤  
其他用户接收到红包消息，点开，拆。由于用户操作的速度远远低于计算机处理速度，所以这打开和拆开的分离，相当于设置了一道缓冲。另外，点开之后，不直接获取金额，而是先读取红包是否领完的缓存，如果没领完则显示【拆】的按钮。点击【拆】之后再次访问缓存看红包是否领完，如果没领完，则请求服务器内存计算随机金额并返回客户端，然后异步写入数据库。

　　⑥ 红包结果会写入LIstView(安卓的UI控件名称，ios也有类似的控件)中，用户可以马上看到

　　⑦ 当用户再次打开红包结果页面时，会从数据库读取最新的结果列表并更新结果列表。

　　(3)收发普通红包



　　① 发起红包操作

　　② 银行扣款逻辑，不成功则返回，成功则进行下一步

　　③ 选择发送对象(若在聊天窗口中发起着跳过这一步)

　　④ 计算红包均值(总额/个数)，将红包个数和均值写入数据库，返回红包ID到客户端

　　⑤ 其他用户点开红包，拆，访问红包个数判断是否大于0，若为TRUE，则个数减1;若为FALSE则通知客户端显示【已领完】样式。

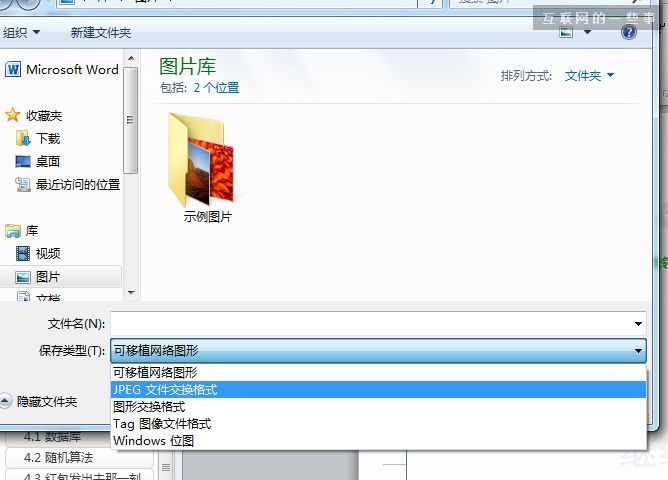
　　5.2 界面交互

　　5.2.1 基本流程



　　5.2.2 拆红包页面显示逻辑

　　对群手气红包、群普通红包、普通红包(其实就是红包个数为1的群普通红包)和是否领到和是否领完做3×3×3的交叉分析之后，归纳出以下结论：



　　5.2.3 红包结果页面显示逻辑



　　说明：

　　1 代表有出现该项

　　“字样”代表下图所示区域的文字内容：



　　“按钮”代表蓝色文字链接，如下图所示：



　　金额是指自己拿到的金额



　　抢到的人是指一个列表：



　　绿色格子代表没有这种逻辑，可能是不出现该页面或者其他原因。

　　对上表的数据进行挖掘，我们可以发现以下规则集：

　　(1)当领到红包的时候，会显示按钮”已存入零钱，可用于发红包”、”已存入零钱，可用于消费”、”已存入零钱，可用于转账”、”已存入零钱，可用于提现”的其中一个，顺序或随机出现;并显示自己所获得的红包金额。

　　(2)当自己发的红包没被领完，会显示按钮”继续发送此红包”;

　　(3)领到别人发的红包时，会显示按钮”查看我的红包记录”;

　　(4)对于群手气红包被领完时，如果红包是自己发的会显示字样”n个红包共n元，n秒被抢光”;如果是被人发的红包则会显示字样”n个红包，n秒被抢光”;对于(群)普通红包被领完时，会显示字样”n个红包共n元”;

　　(5)对于红包(个数大于1)没被领完，自己的红包会显示字样”已领取x/y个，共x/y元”;别人发的红包字样”领取x/y个”;

　　(6)对于红包(个数等于1)没领完时，会显示字样”红包金额n元，等待对方领取”;

　　(7)对于群手气红包和自己发的普通红包都会显示抢到红包的人的列表;

　　(8)已经被领完的群手气红包才会显示”最佳手气”的标识;

　　从(4)-(6)的规则我们可以看出，微信做到为什么是一个优秀的产品而不仅仅是一个及格的产品。自己发的红包会显示出总金额，自己发了多少钱自己心里有数，却不希望别人看到总的金额(虽然可以根据列表算出来，但是大部分人不会去计算每一个别人红包的总金额)，避免发红包的用户还要承受”面子问题”挫伤用户发红包的积极性。这样去营造一种无分贵贱贫富，人人都可以发红包的氛围，间接提高发红包的人数和整个平台的活跃度。

　　5.2.4摇一摇红包

　　这一部分因为写文章的时候摇一摇红包活动已经下线了，所以只能从网上找来截图，简略地说明一下流程。如下图：

