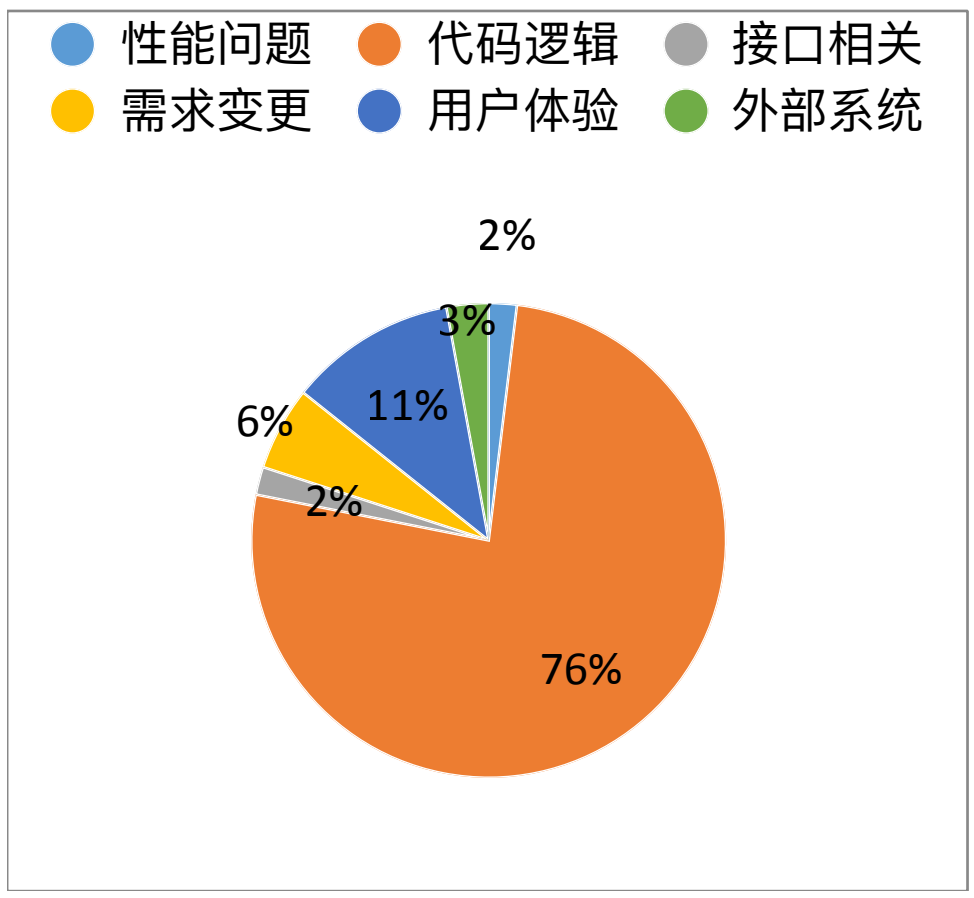
表格 1

bug类型		bug类型细分				
性能问题	2	数组越界				
		数据类型				
		逻辑问题				
		版本兼容	1			
		未知原因	1			
代码逻辑	80	逻辑错误	80			
		网络机制				
接口相关	2	接口问题	2			
		环境问题				
		请求参数问题				
完整性		考虑补全（数据联动）				
		考虑补全（网络差）				
		考虑补全（多用户）				
		考虑补全（版本兼容）				
		边界测试				
需求变更	6	需求文档没明确				
		需求文档变更	4			
		新增需求	2			
用户体验	12	交互处理	5			
		样式	6			
		兼容性	1			
其他		未知原因				
		数据调整				
历史遗留						
外部系统	3					



日均产生bug率

工作人日	Android开发的bug总数	bug数/工作人日		
455	105	0.23076923076923		

组员日均bug产生率

组员	工作人日	开发的bug总数	bug数/工作人日	
苏蓓蓓	65	27	0.415384615384615	
潘思嘉	65	27	0.415384615384615	
罗宝恩	65	23	0.353846153846154	
林俊炳	65	24	0.369230769230769	
任鹏		4		

Bug激活情况

BugID	激活次数	类型	所属	
99960	1		俊炳	
100223	1		俊炳	
100264	1		俊炳	
100536	1		俊炳	
99556	1		罗宝恩	
100110	1		罗宝恩	
100447	2		罗宝恩	
99202	1		潘思嘉	
99670	1		潘思嘉	
100071	1		潘思嘉	
100304	1		潘思嘉	
100570	1		潘思嘉	
100106	1		苏蓓蓓	
100350	1		苏蓓蓓	
100381	1		苏蓓蓓	
100579	1		苏蓓蓓	
100601	1		苏蓓蓓	

激活率

姓名	bug总数	激活总次数	激活总次数/bug总数	
苏蓓蓓	27	5	0.185185185185185	
潘思嘉	27	5	0.185185185185185	
罗宝恩	23	3	0.130434782608696	
林俊炳	24	4	0.166666666666667	

表格 1

苏蓓蓓	口袋蜜蜂1.0.1版本，主要解决1.0遗留的截图问题和分享问题。用原生webview截图，同时限制图片高度和大小。分享模块获取活跃度后面会改为点击分享图片时就去向接口请求。为了保证分享图片的清晰度，图片大小限制在10M以内，应用不对图片压缩，只经过第三方sdk压缩。			
罗宝恩	在口袋蜜蜂1.0.0版本中收到QA指派bug37个，处理结果为不值得修复7个，主要是系统特性和机型引起，修复成本较高或者无法修复，因而不作处理； 无法重现、特意设计和重复提交各1个，有效bug27个，3个为需求变动或新增需求，在剩下的24个bug中，多为在集成测试难以发现的由于设备兼容问题 或者一些细节问题引起。由于这次是开发一个新的app，1.0版本多为对口袋车微信版的复制，开发过程中没有使用微信版与app在功能上做比较，在消息 模块产生比较多问题，需求文档也没很好说明情况，特别是消息模板，需要反思下，在以后的需求评审过程中需要多考虑微信版与pc版的关联。在技术上， 该版本由于是新的app，除框架外的技术选型都偏向了比较新的方案，难免出了一些坑，首次在项目中用到RN相关的东西，在公测后发现有几个目前还没定位 到原因以及还没处理的crash，后续要跟进下；由于测试时间的调整打乱了一些原来内存优化的计划，后续版本补回，腾讯X5内核sdk目前版本有个比较明显 的内存泄漏问题，待官方更新SDK。			
潘思嘉	现在的框架组合下，react native 已经能够承担了大部分的“边角”工作，用户使用起来感觉不到和原生的区别。 整个ios app 包27.4M，ios 图片zip 后3.7M，RN jsbundle 包才1.3M，RN 的图片zip 后4M， 整个android app包18.4M，anroid res文件 zip 后1.4M，RN jsbundle 包才1.3M，RN 的图片zip 后4M 处理了这么多的边角功能，体积占比这么少，我觉得性价比高啊。 项目前期花了很多的时间在两个方面，一个是去沟通两个平台提供给RN 的功能接口，例如侧窗，拍照，打电话，跳转传值，网络请求等。另一个是框架没定型，导致做了的工作又删掉，例如本地是RN 有自己的导航的，后来改为全部使用原生的导航。还好，这些大多数工作都能复制到其他项目上，加上已经迭代了3次的项目架构和其他RN 基础组件，都能够沉淀下来作为其他项目的公共功能。估计之后很少会出现这样的时间浪费。 和林俊炳相比，我的速度慢很多。一个是缺经验，会把bugfree 作为参考补充到测试用例模板； 另一个是我缺少原生开发能力。所以在不断学习RN 的同时，我选择补习ios 方向。现在已经一本可以区分原生问题还是RN 问题，和极简单的原生修补工作。而对于自己本来对web 端的了解，接下来会尝试用在终端页的优化上。（更新：测试可以通过拦截并改写webview 的请求让webview 使用app 缓存的文件和解决图片跨域问题） RN 的一些坑： - 输入框切换明文导致光标移位。 - 远程调试和普通调试的js 方法不一样 - 键盘遮挡输入框问题，现在都只能绕开问题而不能真正解决			