# 第一章引言

## 1.1项目背景

自2007年，苹果推出了第一代iPhone，智能手机开始真正走向市场。并凭借其人性化，功能强，速度快的特点飞速发展和普及。随之而来的是移动应用出现在我们生活的方方面面，从日常社交，新闻浏览，运动健身到购物付款，移动应用保罗万象，甚至南大宿舍洗衣机也需要U净这款app使用。现代人一机在手，行走天下，越来越离不开手机，离不开里面覆盖生活方法面面的应用。据工信部发布2017年上半年我国互联网业务运行情况报告显示国内移动应用程序市场持续活跃且移动互联网应用数量已超402万款，其中游戏达116万款。

然而在这高速发展的移动应用市场却也存在着诸多隐患，2016年CNCERT/CC捕获及通过厂商交换获得的移动互联网恶意程序样本数量为2,053,501个。而Android平台用户成为最主要的攻击对象。恶意程序针对Android平台的恶意程序共有2,053,450个，占99.9%以上，位居第一。应用安全正成为用户越来越关注的点。另外移动应用的使用场景比传统pc端应用或网页应用更加复杂，同一个移动应用的使用环境包括不同版本的不同操作系统，不同的硬件设备，不同的网络环境。

所以对于开发者而言，确保移动应用在复杂环境中有效运行无疑是一个艰巨的任务。而测试无疑是质量安全的必要保证。但移动应用环境的复杂带来了移动应用测试的复杂，对于开发者而已，保证测试设备完备，测试人员专业必然是困难的事情。面对这样的难题，一个提供大量只有多样手机用户的专业的测试平台则是帮助开发人员的法宝。这就是群智测试工具兴起的必然。

## 1.2群智测试工具以及现状

### 1.2.1bugtags

Bugtags是国内一个精干的移动开发团队为改善移动产品质量而专门打造的测试平台产品。Bugtags采用的是所见即所得的问题上报方式，得益于此有效提高了问题上报的效率和问题描述的准确度；同时Bugtags也提供了自动收集分析崩溃信息与bug生命周期管理功能。能够展示bug描述信息，设备信息，重现步骤，堆栈信息，用户自定义数据，控制台日志，网络请求，十分全面如下图1.1所示。该团队还提供了bugtags日志的专有服务。但如果是免费用户则只能是使用基本功能，得不到细致周到的服务。



图1.1 bugtags的bug信息

### 1.2.2班墨云测试

班墨云测试平台自称为是全球领先的人工智能云测试平台，其除了提供与bugtags类似的众测服务外，还为用户提供兼容性测、适配测试、稳定性测试、网络测试、功能测试、性能测试等服务。自动化测试，人工智能测试时其一大特色。该测试平台也不局限于移动应用，网站，H5，游戏开发者均可使用该平台，改善项目开发。该平台如下图1.2所示：



### 1.2.3群智测试现状

从测试平台现状来看，移动应用群智测试大多借助于平台广阔的用户群体，由用户下载装有平台开发SDK的应用，使用logcat和monkey进行测试，收集log日志和crash信息，用web页面进行展示。其中用户测试时缺乏反馈信息，大量用户重复提交相同bug，一定程度上提升了开发人员的负担。我们希望除了提供群智测试平台外，用户测试时可以实时得到平台反馈，减少bug的重复提交，提升bug描述的精确有效。

## 1.3项目功能概述

本人负责开发群智测试系统bughunter的后台服务器，该服务器应提供一下基本功能：

1.bug信息的提交。

2.bug信息的筛选反馈。

3.应用bug信息的统计分析数据。

4.用户，应用及其版本信息，bug信息的存储使用，主要是增删改查功能，其中需要记录bug的修改历史记录。

1.4论文的主要工作和组织结构

论文主要表述群智安卓测试系统BugHunter后端的设计与实现，还有其与web前端及Android测试端的交互。

本文第一章为论文的引文，第二章是bughunter所用到的技术概述，第三章描述了系统的需求分析和概要设计，第四章则表述了系统的详细设计与实现，第五章作为最后表达了本次毕业设计的总结与展望。

# 第二章技术概述

## 2.1后台框架Spring Boot

从应用方面而言，框架是整个或部分系统的可重用设计，表现为一组抽象构件及构件实例间交互的方法;从目的方面可以说框架是可被应用开发者定制的应用骨架。

软件系统发展到今天变得非常复杂，特别是服务器端软件，涉及到的知识，内容，问题极多。而使用别人成熟的框架，框架可以帮助完成一些基础工作，开发人员只需要集中精力完成系统的业务逻辑设计。并且框架一般是成熟，稳健的，可以处理系统很多细节问题，如事物处理、安全性、数据流控制等问题。框架一般都经过广泛使用和验证，结构性好，扩展性强，而且它是不断升级的。

为了便于bughunter系统的后台开发，我们也应用了框架来开发，采用了目前流行的java web框架Spring Boot。Spring Boot是由Pivotal团队提供的全新框架，其设计目的是用来简化新Spring应用的初始搭建以及开发过程。相比Spring和SpringMVC需要进行很多手动配置，不仅徒增工作量而且在跨平台部署时容易出问题，Spring-Boot遵循约定大于配置的软件设计范式，使用了特定的方式来进行配置，从而使开发人员从样板化的配置中解放出来。

依据Spring Boot的官网，其主要提供以下核心功能：

1.独立运行的Spring项目，Spring Boot可以以jar包的形式来运行，运行一个Spring Boot项目只需要通过java -jar xx.jar的命令。

2.内嵌Servlet容器，Spring Boot可以内嵌Tomcat，无需以war包的形式部署项目。

3.提供starter简化Maven配置，使用Spring或者SpringMVC需要添加大量的依赖，而这些依赖大都是固定的。Spring Boot 通过starter能够帮助我们简化Maven配置。

4.自动配置Spring。

5.准生产的应用监控。

6.无代码生成和xml配置。

关于使用Spring Boot进行开发的更详细的内容，将在第三章和第四章进行更详细的说明。

## 2.2依赖注入IoC和面向切面编程AOP

Spring的核心特性就是IoC和AOP，IoC（Inversion of Control），即“控制反转”；AOP（Aspect-Oriented Programming），即“面向切面编程”。

IoC也叫DI（Dependency Injection），即依赖注入。并非一种技术实现，而是一种设计思想。在任何一个有实际开发意义的程序项目中，会使用很多类来描述它们特有的功能，并且通过类与类之间的相互协作来完成特定的业务逻辑。这样，每个类都需要负责管理与自己有交互的类的引用和依赖，代码将会变的异常难以维护和极度的高耦合。而IoC正是用来解决这个问题，通过IoC将这些相互依赖对象的创建、协调工作交给Spring容器去处理，每个对象只需要关注其自身的业务逻辑关系。如此，获得依赖的对象的方式，进行了反转，变成了由spring容器控制对象如何获取外部资源。可以说IoC有利于实现松耦合，对象被动接收依赖类而不是自己主动寻找，实现了JNDI的反转：对象不是从容器中查找它的依赖类，而是容器在实例化对象时主动将依赖类注入给它。

AOP（Aspect Oriented Programming），面向切面编程，《Spring3.x企业应用开发实战》第6章写道：AOP是OOP的有益补充。

OOP引入封装、继承、多态等概念建立一种对象层次结构，模拟公共行为的一个集合。但OOP并不适合定义横向的关系，例如日志功能，日志代码往往横向地散布在所有对象层次中，而与它对应的对象的核心功能毫无关系。还有如安全性、异常处理和透明的持续性，这种散布在各处的无关的代码被称为横切，在OOP设计中导致了大量代码的重复，不利于模块重用。

AOP则不同，利用称为"横切"的技术，剖解开封装的对象内部，将那些影响了多个类的公共行为封装至一个可重用模块，命名为"Aspect"，即切面，减少系统的重复代码，降低模块之间的耦合度，并有利于未来的可操作性和可维护性。使用"横切"技术，AOP将软件系统分为两个部分：核心关注点和横切关注点。业务处理主要流程是核心关注点，与之关系不大的、经常发生在核心关注点的多处的部分则是横切关注点。AOP的作用就在于此，分离系统中的各种关注点，将核心关注点和横切关注点分离开来。将业务逻辑从系统服务中分离出来，有助于内聚开发，将服务模块化，并把它们声明式地应用在需要它们的地方，使这些组件更加专注于自身的业务，不需要涉及其它系统问题。

## 2.3 Spring Boot JPA

JPA是Java Persistence API的简称，中文名Java持久层API，是JDK 5.0注解或XML描述对象－关系表的映射关系，并将运行期的实体对象持久化到数据库中。JPA为java开发者提供了对象-关系的映射机制，用于管理java应用中的关系数据。如下图2.1就是对象关系映射的一个例子。

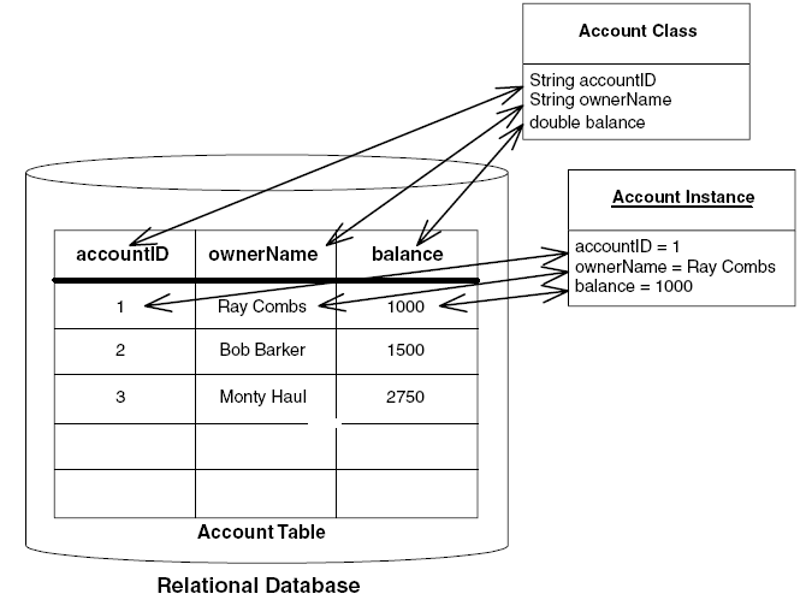


图2.1 对象关系映射示例

对象关系映射ORM通过编程的方式将对象类型转换成关系类型。主要特点是将object映射成数据库中的数据。在映射的过程中必须考虑数据，数据类型和数据之间的关系。ORM在三个阶段中将数据存入数据库中如下图2.2所示。

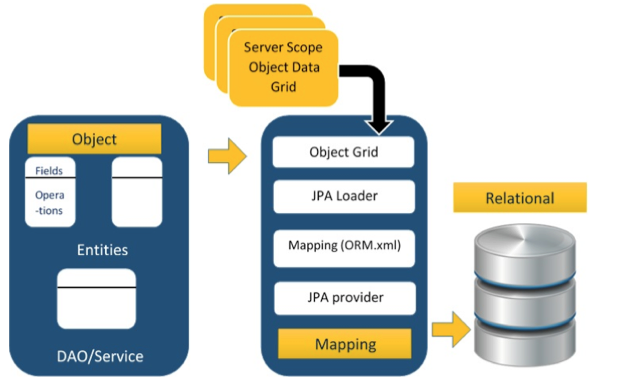


图2.2 ORM三个阶段

而在本次开发中，由于使用了Spring Boot的框架，使用JPA极其简单方便，只需要配置maven依赖，配置文件，创建实体以及数据访问接口即可。可以说本次开发得益于JPA的简单方便，查询能力，数据层开发十分快捷简练。

## 2.4关系型数据库管理系统MySQL

RDBMS，即传统的关系型数据库管理系统。典型的代表有MySQL、Oracle、SQL Server以及PostgreSQL等。

关系型数据库一般满足事务型数据库的要求ACID，即：Atomicity（原子性），Consistency（一致性），Isolation（隔离性），Durability（持久性）。除此之外，RDBMS的表设计还有几大设计范式，1NF、2NF、3NF、BCNF。

这些原则与范式强调数据必须在任何时刻都是完整的、可靠的、一致的。这种模型是很直观的、符合人们对于数据预期的一种模型，也是对于安全性来说最佳的数据模型。

考虑到关系型数据库存在锁（如MySQL存储引擎MyISAM的表锁、InnoDB的行锁），在高并发量下，锁竞争会异常激烈，导致响应时间下降的问题，还有关系型数据库对bughunter的优势并不明显，本次bughunter系统后台的开发中，思考过使用与JSON完美结合的MongoDB，或者当下很流行的Spring Boot和Key-Value数据库Redis组合，但鉴于时间紧迫，开发人员的知识储备等情况，仍选用开源关系型数据库MySQL作为数据持久化存储的解决方案。

MySQL是最流行的关系型数据库管理系统之一，在WEB应用方面，MySQL是最好的RDBMS应用软件。与其他的大型数据库例如Oracle、SQL Server等相比，MySQL有它的不足之处，但丝毫没有减少它受欢迎的程度。对于一般的个人使用者和中小型企业来说，MySQ提供的功能绰绰有余，而且MySQL是开源软件，可以大大降低成本。当下一般中小型网站的开发都选择 MySQL 作为网站数据库。

MySQL在技术社区中有大量支持，而且在课程的学习实践中，MySQL也是我们选择最多的数据层的解决方案。相信采用MySQL，能够更快更好地完成本次bughunter的后台的开发工作。

## 2.5AppKey和AppSecret

目前开放api接口平台会有AppId、AppKey、AppSecret，各有用途：

1. AppId是用来标记开发者账号的, 是用户id。

2. AppKey客户端调用API时的唯一标识，服务器通过App Key来鉴别应用的身份。调用API接口时必须传入的参数。

3.AppSecret是服务端给客户端分配的密钥，用来保证应用来源的可靠性，防止请求数据被伪造。

当下也有简化的场景如省去AppId, 默认每一个用户有且仅有一套权限配置， 所以直接将AppId= AppKey。或者省去AppId和AppKey，通常用于开放性接口的地方。

本次开发bughunter后台仅供web前端和Android端调用，所以为简便，只采用了AppKey和AppSecret，主要用于Android端请求，唯一标识App。

## 2.6跨域请求

本次开发集成测试时，发现由于前后端域名不一致，存在跨域请求问题，遂采用CORS（Cross-origin resource sharing）即跨域资源机制。CORS是一个W3C标准，允许浏览器向跨源服务器，发出XMLHttpRequest请求，从而克服了AJAX只能同源使用的限制。

CORS需要浏览器和服务器同时支持。目前所有浏览器基本都支持该功能，IE浏览器则不能低于IE10。

CORS通信过程都是浏览器自动完成，不需用户参与。对于开发者来说，CORS通信与同源的AJAX通信没有区别，代码一致。浏览器发现AJAX请求跨源，就会自动添加附加的头信息，有时还会多发出一次附加请求，但用户不会有所察觉。

实现CORS通信的关键在于服务器。只要服务器实现了CORS接口，就可以跨源通信了。本次于服务其中采用Servlet中的过滤器Filter来实现CORS机制。

过滤器Filter可以实现如下功能：

1.在HttpServletRequest 到达Servlet之前，拦截HttpServletRequest。根据需要检查HttpServletRequest，也可以修改HttpServletRequest头和数据。

2.在HttpServletResponse 到达客户端之前，拦截HttpServletResponse。根据需要检查HttpServletResponse，可以修改HttpServletResponse 头和数据。

而CORS实现正是利用了功能1，本次实现如下图2.3所示：



图2.3 CORS实现

# 第三章系统需求分析与概要设计

## 3.1BugHunter项目整体概述

群智安卓测试BugHunter系统是一个面向安卓测试的移动应用群智测试平台，它的主要目标是为安卓测试经理提供bug生命周期的管理功能，为测试人员提交bug高效准确的上报功能。

BugHunter系统通过web前端安卓测试经理不仅提供app和bug的管理功能，还提供了丰富的bug信息统计图表的展示，更便于分析app的开发情况。

通过开发安卓测试SDK，应用开发人员可以方便的将我们提供的静态库添加到其app中添加bug的反馈和提交功能，之后测试人员可以很方便的使用我们提供的悬浮按钮查看当前页面存在的bug和提交当前页面bug，当app崩溃时将自动上传bug信息。

本人则负责用spring-boot+mysql为基础开发server端，为app提供bug上报途径，为web端提供app和bug的信息筛选和管理功能。

## 3.2系统需求分析

### 3.2.1用例图

为了更形象的阐述需求，将系统的功能以用例图和用例描述的方式展示如下图3.1所示：

用例图由角色，系统边界和用例组成，它可以帮助我们更好地描述系统内外交互。系统的涉众有测试人员和测试经理两类。测试人员借助系统完成测试任务上报bug。测试经理则研究系统收集到的数据，分析应用开发状况。

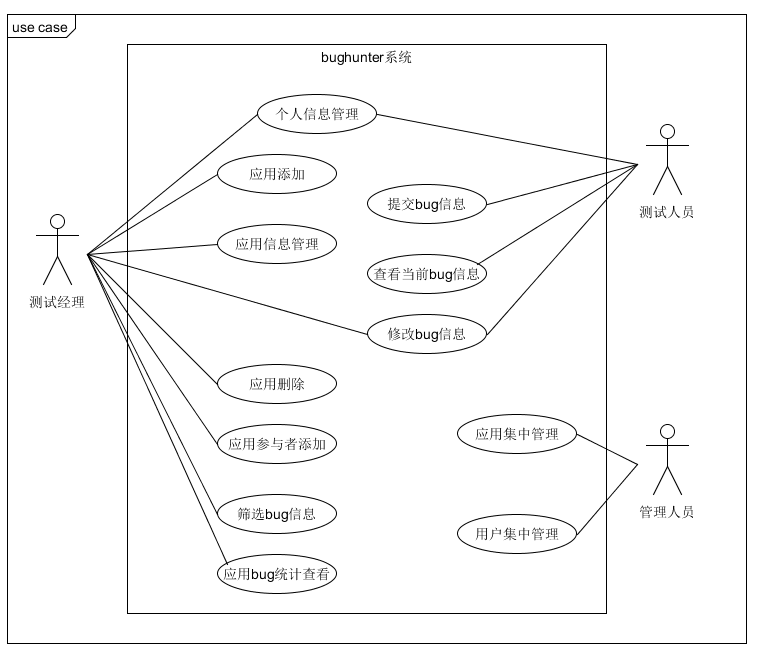


图3.1 用例图

### 3.2.2用例描述

基于以上用例图编写用例描述如下列表所示：

1）个人信息管理

|  |  |
| --- | --- |
| 项目 | 内容描述 |
| ID | uc01 |
| 名称 | 个人信息管理 |
| 参与者 | 用户 |
| 触发条件 | 用户需求管理个人信息 |
| 前置条件 | 用户已登陆成功 |
| 后置条件 | 用户信息按用户的需要做出修改 |
| 正常流程 | 1.用户进入个人信息页面  2.系统返回用户个人信息  3.用户修改个人信息，并点击确定  4.系统将用户个人信息修改情况反馈用户 |
| 扩展流程 | 无 |
| 特殊需求 | 无 |

表3.1 个人信息管理用例

2）应用添加

|  |  |
| --- | --- |
| 项目 | 内容描述 |
| ID | uc02 |
| 名称 | 应用添加 |
| 参与者 | 测试经理 |
| 触发条件 | 测试经理添加应用到账户 |
| 前置条件 | 测试经理已经登陆成功 |
| 后置条件 | 系统将应用信息添加到数据库 |
| 正常流程 | 1.测试经理进入应用添加页面，并且输入完整应用信息，点击添加  2.系统自动生成AppKey和AppSecret，将应用信息添加到数据库，返回AppKey和AppSecret |
| 扩展流程 | 无 |
| 特殊需求 | 1.AppKey应为app主键，确保唯一性  2.AppSecret为app验证信息，需难以破解，确保安全性 |

表3.2 应用添加用例

3）应用信息管理

|  |  |
| --- | --- |
| 项目 | 内容描述 |
| ID | uc03 |
| 名称 | 应用信息管理 |
| 参与者 | 测试经理 |
| 触发条件 | 应用信息需要被查看或修改 |
| 前置条件 | 测试经理已经登陆成功 |
| 后置条件 | 系统将应用信息修改提交到数据库 |
| 正常流程 | 1.测试经理进入应用信息页面  2.系统返回应用信息  3.测试经理修改应用信息，点击确定  4.系统按需要修改应用信息，返回修改结果 |
| 扩展流程 | 无 |
| 特殊需求 | 无 |

表3.3 应用信息修改用例

4）应用删除

|  |  |
| --- | --- |
| 项目 | 内容描述 |
| ID | uc04 |
| 名称 | 应用删除 |
| 参与者 | 测试经理 |
| 触发条件 | 应用已经被放弃，需要删除 |
| 前置条件 | 测试经理已经登陆 |
| 后置条件 | 应用被从数据库删除，包括应用所含bug |
| 正常流程 | 1.测试经理进入应用页面，点击删除  2.系统删除数据库中包含的应用及其bug信息，返回结果 |
| 扩展流程 | 无 |
| 特殊需求 | 无 |

表3.4 应用删除用例

5）应用参与者添加

|  |  |
| --- | --- |
| 项目 | 内容描述 |
| ID | uc05 |
| 名称 | 应用参与者添加 |
| 参与者 | 测试经理 |
| 触发条件 | 有人员需要被添加应用管理权限 |
| 前置条件 | 该应用测试经理已经登陆 |
| 后置条件 | 选择的用户被添加到应用中 |
| 正常流程 | 1.测试经理进入应用成员添加页面，输入添加人员邮箱，成员类型，点击确定  2.系统将该成员添加至应用，返回结果 |
| 扩展流程 | 2a.该成员邮箱不存在  2a1.系统返回该成员不存在  2b.该测试经理无添加成员权限  2b1.系统返回权限不足信息 |
| 特殊需求 | 1.仅应用创建者有权限添加成员 |

表3.5 应用参与者添加用例

6）筛选bug信息

|  |  |
| --- | --- |
| 项目 | 内容描述 |
| ID | uc06 |
| 名称 | 筛选bug信息 |
| 参与者 | 测试经理 |
| 触发条件 | 测试经理筛选bug信息，分析应用开发状况 |
| 前置条件 | 测试经理已经登陆 |
| 后置条件 | 无 |
| 正常流程 | 1.测试经理点击进入bug展示页面  2.系统返回该应用下所有bug用于展示  3.测试经理点选筛选条件  4.系统根据筛选条件返回所需bug信息 |
| 扩展流程 | 无 |
| 特殊需求 | 1.筛选条件为app版本和bug类型，状态，优先级 |

表3.6 筛选bug信息用例

7）修改bug信息

|  |  |
| --- | --- |
| 项目 | 内容描述 |
| ID | uc07 |
| 名称 | 修改bug信息 |
| 参与者 | 测试经理或测试人员 |
| 触发条件 | bug信息不够完善，或bug状态改变需要修改信息 |
| 前置条件 | 用户已登录 |
| 后置条件 | 系统按需要修改bug信息并记录修改历史 |
| 正常流程 | 1.用户已进入bug信息页面，做出修改，点击确定  2.系统按需要修改bug信息并记录修改历史后返回结果 |
| 扩展流程 | 无 |
| 特殊需求 | 1.bug信息用户仅可以修改描述、状态、优先级、类型 |

表3.7 修改bug信息用例

8）提交bug信息

|  |  |
| --- | --- |
| 项目 | 内容描述 |
| ID | uc08 |
| 名称 | 提价bug信息 |
| 参与者 | 测试人员 |
| 触发条件 | 测试人员发现bug或应用崩溃 |
| 前置条件 | 测试人员登陆成功或测试人员仅使用而未登陆均可 |
| 后置条件 | 系统将测试人员提交的bug保存到数据库 |
| 正常流程 | 1.测试人员发现bug并且填写提交信息完毕后点击确定，或应用在使用中崩溃自动提交bug。  2.系统记录提交的bug到数据库，返回结果 |
| 扩展流程 | 2a.该系统的版本未被记录  2a1.系统记录该系统版本 |
| 特殊需求 | 1.用户仅可填写bug描述、状态、优先级、类型，其他信息由SDK自动提取 |

表3.8 提交bug信息用例

9）查看当前bug信息

|  |  |
| --- | --- |
| 项目 | 内容描述 |
| ID | uc09 |
| 名称 | 查看当前bug信息 |
| 参与者 | 测试人员 |
| 触发条件 | 测试人员进入app行的activity |
| 前置条件 | 无 |
| 后置条件 | 无 |
| 正常流程 | 1.测试人员点击应用使应用进入新的页面  2.系统返回当前页面已存在的bug信息 |
| 扩展流程 | 无 |
| 特殊需求 | 无 |

表3.9 提交bug信息用例

10）应用bug统计查看

|  |  |
| --- | --- |
| 项目 | 内容描述 |
| ID | uc10 |
| 名称 | 应用bug统计查看 |
| 参与者 | 测试经理 |
| 触发条件 | 测试经理进行bug统计查看 |
| 前置条件 | 该应用测试经理已经登陆 |
| 后置条件 | 无 |
| 正常流程 | 1.测试经理点击进入bug统计信息展示页面  2.系统返回当前最新版本的bug统计信息  3.测试经理点选不同版本查看  4.系统返回不同版本bug统计信息 |
| 扩展流程 | 无 |
| 特殊需求 | 无 |

表3.10 应用bug统计查看用例

11）应用集中管理

|  |  |
| --- | --- |
| 项目 | 内容描述 |
| ID | uc10 |
| 名称 | 应用集中管理 |
| 参与者 | 测试经理 |
| 触发条件 | 测试经理进行bug统计查看 |
| 前置条件 | 该应用测试经理已经登陆 |
| 后置条件 | 无 |
| 正常流程 | 1.测试人员点击应用使应用进入新的页面  2.系统返回当前页面已存在的bug信息 |
| 扩展流程 | 无 |
| 特殊需求 | 无 |

表3.11 应用集中管理

12）用户集中管理

|  |  |
| --- | --- |
| 项目 | 内容描述 |
| ID | uc12 |
| 名称 | 用户集中管理 |
| 参与者 | 测试经理 |
| 触发条件 | 测试经理进行bug统计查看 |
| 前置条件 | 该应用测试经理已经登陆 |
| 后置条件 | 无 |
| 正常流程 | 1.测试人员点击应用使应用进入新的页面  2.系统返回当前页面已存在的bug信息 |
| 扩展流程 | 无 |
| 特殊需求 | 无 |

表3.12 用户集中管理

### 3.2.3非功能需求描述

系统的非功能需求如表3.10，要求系统具有良好的可靠性、易用性、可维护性和可移植性，同时，系统要求在较短时间内对用户操作做出响应。

|  |  |
| --- | --- |
| 需求类别 | 需求描述 |
| 效率 | 对于所有操作，系统应及时作出响应，响应时间应在0.5秒之内 |
| 可维护性 | 系统设计应尽可能合理，容易维护和扩展新模块 |
| 安全 | 系统需要验证操作是否合法，并拒绝非法操作 |
|  |  |

表3.13 非功能需求

系统所受的约束见表3.11，系统需要在有网络的环境下运行。

|  |  |
| --- | --- |
| 约束类别 | 约束描述 |
| 开发约束 | 系统服务器要求使用spring-boot+mysql的方式开发 |
| 部署环境 | 该系统服务器端将被部署到阿里云服务器ECS上，配置为1核，2GB，系统使用CentOS 7.3 64位 |
| 网络约束 | 系统必须在有网络环境下运行 |
| 用户约束 | 测试人员可不经过登陆使用系统，但须提供AppKey和APPSecret |

表3.14 约束

## 3.3系统概要设计

### 3.3.1E-R图

根据需求给出bughunter系统E-R图如下图3.2所示：

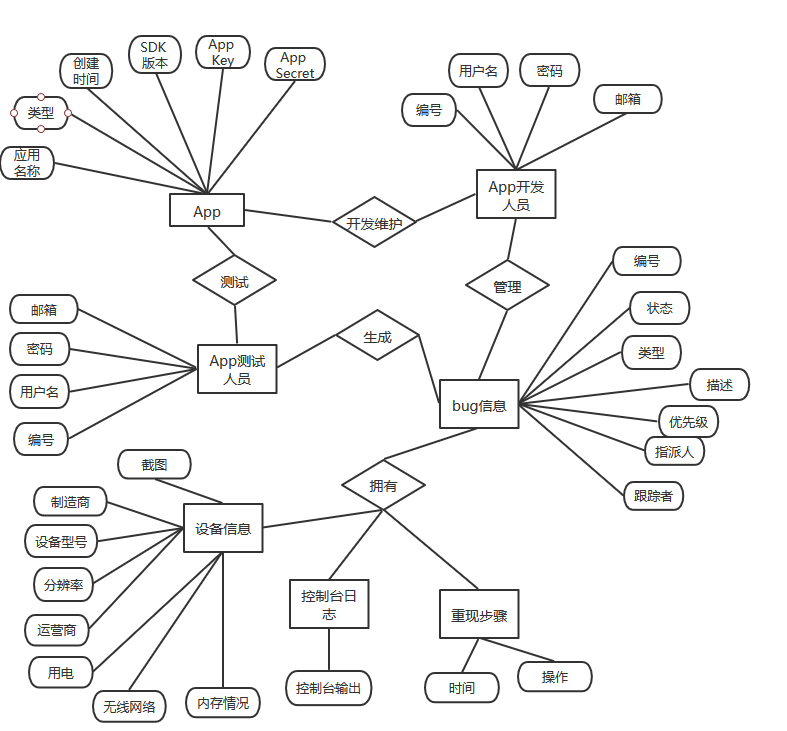
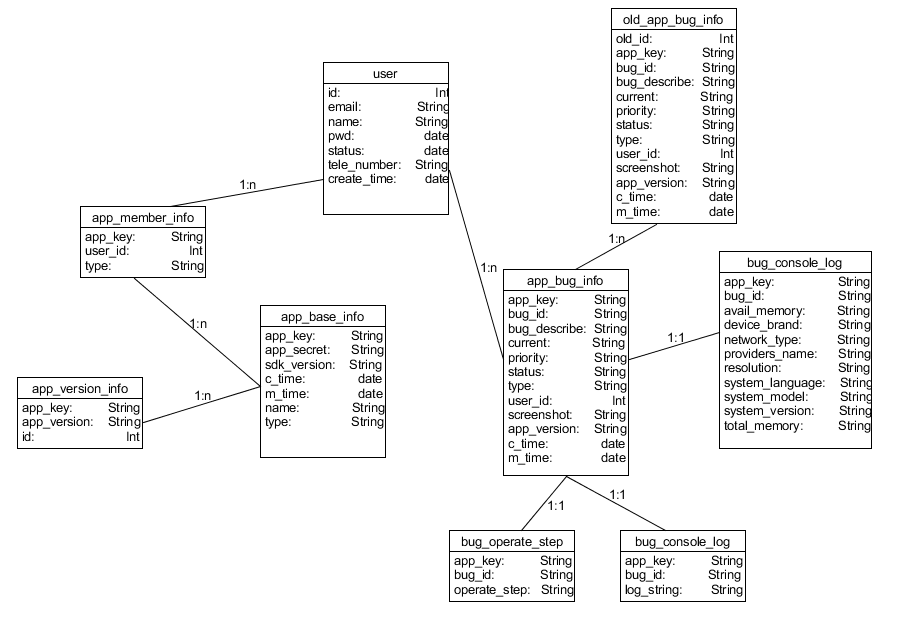


图3.2 系统E-R图

### 3.3.2数据库视图



### 3.3.3体系结构设计-物理视图（Physical View）

### 3.3.4体系结构设计-开发视图（Development View）

## 3.4本章小结

# 第四章BugHunter详细设计与实现

## 4.1项目子模块概述

本项目以Spring Boot框架为基础开发，主要分为Controller、Service、Entity三个模块，下面分别对这三个模块进行简要描述，从整体上描述各模块的功能职责。各模块的详细设计将在4.2做详细说明。

### 4.1.1Controller模块

Controller模块主要用来映射url路径及接收HTTP请求，返回响应。当请求被发送至服务器后，根据路径将被映射到Controller模块响应对象的响应方法中，由于我们采用JSON格式传输数据，将根据需要解析请求体中的JSON数据，后调用Service模块的方法以获得请求的对象，之后将数据打包返回响应。不合法的请求将被抛弃。Controller层仅涉及路径映射和返回响应，不涉及data层及其他复杂的业务逻辑。

### 4.1.2Service模块

Service模块介于Controller层和data层之间，为Controller层提供所需要的服务，从data层获取需要的数据，有时需要对数据进行再加工处理，之后打包为Controller层需要的格式，返回Controller层调用方。Service模块应该负责业务逻辑的处理，实际开发中，项目业务逻辑较少，使Service模块更多的成为Controller层和data层之间的中介，降低系统的耦合性。

### 4.1.3Data模块

Data模块是bughunter项目后台开发的重中之重，Data模块主要利用JPA为java开发者提供的对象-关系映射机制来管理关系数据。开发主要包括数据库配置，定义实体类，实现DAO层。其中DAO层主要利用了JpaRepository，之后会做详细说明。Data层为Service提供管理数据库数据的功能，帮助Service完成自身所提供的服务。

## 4.2Controller模块详细设计

## 4.3Service模块详细设计

## 4.4Data模块详细设计

# 第五章总结与展望