<即应匹配信息发布平台>

测试报告

版本 <2.0>

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| <17日/7月/2019年> | <2.0> | 迭代二测试报告 | Summer855小组 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

1. 简介 4

1.1 目的 4

1.2 范围 4

1.3 定义、首字母缩写词和缩略语 4

1.4 参考资料 4

1.5 概述 4

2. 测试概要 4

3. 测试环境 4

4. 测试结果及分析 5

4.1 需求覆盖率及缺陷分布 5

4.2 缺陷严重程度 5

5. 缺陷清单 6

5.1 功能性缺陷 6

5.2 非功能性缺陷 7

6. 测试结论与建议 7

测试报告

# 简介

## 目的

本测试报告是 用来记录“即应匹配信息发布平台”开发过程中的功能及非功能需求的完成情况，主要目的是发现设计和实现中的缺陷和不足，然后进行修复和改善。

## 范围

此测试报告撰写于迭代二功能完成之后，测试范围包含迭代二的功能测试，包括活动的发起、删除、修改、查询服务。此测试报告基于测试完成之后，对测试中使用的方法、测试的有效程度进行总结。

## 定义、首字母缩写词和缩略语

即应——即应匹配信息发布平台

Auth service —— 即应微服务架构中的用于登录、验证的服务

User service —— 即应微服务架构中用于用户信息的添加、修改、查询、删除的服务

Api gateway —— 即应微服务架构中用于禁停http请求，将请求转发，调用其他服务的服务

Activity service —— 即应微服务架构中用于活动的发起、删除、修改、查询的服务

## 参考资料

《即应项目软件需求规约》 v1.2

《即应第二次迭代计划》 v1.0

《即应匹配信息发布平台 第二次迭代 系统测试用例》 v1.0

## 概述

本测试报告包含测试概要、测试环境、测试结果及分析、测试缺陷报告等，由于第一次迭代尚未进行非功能性需求的分析设计，此次测试将不会对性能等非功能性需求进行测试，重点将会放在Activity service的测试上。

# 测试概要

测试时间：16/07/2019

地点：项目开发教室（软件学院3101教室）

人员：Activity service开发者、api gateway开发者

测试方法：单元测试，人工测试

测试内容：对涉及数据库操作和关联查询等逻辑功能进行单元测试，确保这些逻辑符合我们的预期，对于项目迭代2所要求的功能进行人工测试，用postman发送请求、微信小程序点击等方法测试系统反馈的内容是否符合预期

# 测试环境

系统后台发布在ubuntu16.04上，CPU4核，内存8G，用kubernetes进行容器管理。Kubernetes版本是v1.15.0

微信小程序尚在开发模式，使用微信开发者工具与后台的交互，微信开发者工具运行在windows10家庭版上，版本是稳定版 Stable Build (1.02.1904090)

Postman是运行在linux Ubuntu 18.04 TLS上，版本是v7.2.2

利用go的官方test包进行单元测试。Go运行在Windows 10 上，版本是1.12

# 测试结果及分析

测试功能点数：11， 分别是 发布活动，删除活动，修改活动，查询单条活动，查询我的活动，查询所有活动，添加评论，参与活动，查看活动参与申请，接受活动参与申请，查看活动详情。

测试用例数：33

缺陷数：3

## 需求覆盖率及缺陷分布

**表1 需求覆盖率及缺陷分布**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试内容** | | **测试**  **用例数** | **需求**  **覆盖率** | **缺陷数** | **缺陷率** | **备注** |
| **功能项** | **发布活动** | 5 | 100% |  |  |  |
| **删除活动** | 5 | 100% |  |  |  |
| **修改活动** | 5 | 100% |  |  |  |
| **查询单条活动** | 5 | 100% |  |  |  |
| **查询我的活动** | 3 | 100% | 1 | 33% |  |
| **查询所有活动** | 1 | 100% |  |  |  |
| **添加评论** | 3 | 100% |  |  |  |
| **参与活动** | 1 | 100% |  |  |  |
| **查看活动参与申请** | 2 | 100% | 1 | 50% |  |
| **接受活动参与申请** | 1 | 100% |  |  |  |
| **查看活动详情** | 2 | 100% | 1 | 50% |  |
| **功能项小计** | 33 | 100% |  |  |  |
| **非功**  **能项** | **性能** |  |  |  |  |  |
| **可靠性** |  |  |  |  |  |
| **…** |  |  |  |  |  |
|  | **非功能项小计** |  |  |  |  |  |
|  | **总计** |  |  |  |  |  |

## 缺陷严重程度

**表2 缺陷严重程度**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **严重程度** | **致命** | **严重** | **一般** | **细微** | **总数** |
| **缺陷个数** | 0 | 2 | 1 | 0 | 3 |
| **占缺陷百分比** | 0 | 67% | 33% | 0 | 100% |

# 缺陷清单

## 功能性缺陷

**表3 功能性缺陷列表**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序**  **号** | **缺陷**  **编号** | **严重程度** | **功能模块** | **缺陷标题** | **缺陷描述** | **测试用例编号** |
| 1 | 1 | 一般 | 查询我的活动 | 无法查到活动，且返回“找不到活动”错误 | 修改了微服务server的名称但没有修改client调用server的名称，导致找不到server发生错误。 | jing-testcase-021 |
| 2 | 2 | 一般 | 查看活动详情 | 活动展示失败，界面为空界面 | 该活动没有传入图片，但后端没检查无图片的情况，导致空指针引用错误 | jing-testcase-025 |
| 3 | 3 | 一般 | 查看活动参与申请 | 查不到新的活动申请 | 测试前修改了后端代码对数据库的操作以及Entity结构，但并没有修改数据库的字段，导致数据库修改失败。 | jing-testcase-031 |

表中相关项说明：

1. 缺陷编号：为每个缺陷赋予一个唯一的编号，可以通过此编号对缺陷进行跟踪。例如：Bug001。
2. 缺陷严重程度：缺陷可以根据严重程度分为以下几种情况。
3. 致命（fatal）：致命的错误，测试执行直接导致系统死机、蓝屏、挂起、或是程序非法退出；系统的主要功能或需求没有实现。
4. 严重（critical）：严重错误，系统的次要功能点或需求点没有实现；数据丢失或损坏。执行软件主要功能的测试用例导致系统出错，程序无法正常继续执行；程序执行过于缓慢或是占用过大的系统资源。
5. 一般（major）：不太严重的错误，这样的缺陷虽然不影响系统的基本使用，但没有很好地实现功能，没有达到预期的效果。如次要功能丧失，界面错误，打印内容、格式错误，提示信息不太正确，或用户界面太差，简单的输入限制未放在前台进行控制，删除操作未给出提示，操作时间长等。
6. 细微（minor）：一些小问题，对功能几乎没有影响，产品及属性仍可使用。如软件的实际执行过程与需求有较小的差异；程序的提示信息描述容易使用户产生混淆；有个别错别字、 文字排列不整齐；辅助说明描述不清楚，显示格式不规范，长时间操作未给用户进度提示，提示窗口文字未采用行业术语，可输入区域和只读区域没有明显的区分标志，系统处理未优化等。
7. 功能模块：所测试并出现该缺陷的功能模块名称。
8. 缺陷标题：描述缺陷的标题。
9. 缺陷描述：对缺陷的详细描述。描述缺陷的重现步骤和问题所在，描述需清晰明了，使研发人员可以按照描述将缺陷重现出来。必要时可以放上相应的截图，以便于研发人员理解问题所在。

## 非功能性缺陷

**表4 非功能性缺陷列表**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序**  **号** | **缺陷**  **编号** | **严重程度** | **测试类型** | **缺陷标题** | **缺陷描述** | **测试用例编号** |
| 1 |  |  | 性能 |  |  |  |
| 2 |  |  | 性能 |  |  |  |
| … |  |  | … |  |  |  |
| n |  |  | 可靠性 |  |  |  |

# 测试结论与建议

本次测试的集成测试在API gateway之上进行，而单元测试则是直接测试的微服务部分。API gateway和微服务之间的交互是使用的人工测试完成的，即通过postman或者手动点击前端界面的方式进行测试。这样的方式没有办法保证覆盖率且无法定量地控制覆盖率，而且如果发生了错误，需要手动去后台去寻找。在今后的测试中，对前端、后端部分单独的测试应该多使用自动花测试工具，涉及到前后端交互的部分再使用人工测试。