華中科技大學

本科生毕业设计[论文]

基于 SSM 框架“报名助手”后台管理系统的设计与实现

院　　系　　软件学院

专业班级　数字媒体技术 201501 班

姓　　名　　刘长江

学　　号　　U201517167

指导教师　　方少红

2019 年 6 月 8 日

学位论文原创性声明

本人郑重声明：所呈交的论文是本人在导师的指导下独立进行研究所取得的研究成果。除了文中特别加以标注引用的内容外，本论文不包括任何其他个人或集体已经发表或撰写的成果作品。本人完全意识到本声明的法律后果由本人承担。

作者签名：　　年　　月　　日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保障、使用学位论文的规定，同意学校保留并向有关学位论文管理部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权省级优秀学士论文评选机构将本学位论文的全部或部分内容编入有关数据进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本学位论文属于 1、保密口，在　　年解密后适用本授权书

2、不保密口　。

（请在以上相应方框内打“√”）

作者签名：　　年　　月　　日

导师签名：　　年　　月　　日

基于 SSM 框架“报名助手”后台管理系统的设计与实现

摘要

随着信息化时代的到来，传统的信息管理方式带来越来越多的问题，大数据更是对信息管理提出了更高的要求。本文通过对目前流行的 SSM 框架的研究，结合软件工程和数据库系统原理中学到的知识，梳理业务需求，设计了一个基于 B/S 架构的后台管理系统，实现了用户参与活动，关注活动，管理活动，发起活动，参与组织，管理组织和创建组织等基础功能，同时完成了推荐系统和敏感词检查等高级功能。

论文首先介绍了项目背景和开发中使用到的 SSM 框架，DFA 以及基于 item 的协同过滤。接着介绍了系统的总体需求，在此基础上展开系统的功能性需求和非功能性需求。需求明确后，描述了系统的概要设计，主要是系统结构图和数据流向图。然后介绍了系统的详细设计，从设计思路，业务流程，模块描述，接口说明，类图以及数据库设计几个方面展开详尽的描述。紧接着开始系统的实现，介绍了后台代码的逻辑，同时展示了一些小程序端界面和后台管理系统界面。

系统开发过程中，小程序端采用了 mpvue，后台服务器采用 SSM 开发框架，后台管理系统采用 vue。

关键词：B/S 架构，SSM 框架，DFA, 基于 item 的协同过滤

Design and Implementation of "Registering Assistant" Background Management System Based on SSM Framework

Abstract

With the advent of the information age, traditional information management methods have brought more and more problems, and big data has put forward higher requirements for information management. This paper, through the research of the current popular SSM framework, combined with the knowledge learned from the principles of software engineering and database system, combing the business needs, designed a background management system based on B/S architecture, realized user participation activities, attention activities, management Activities, initiating activities, participating in organizations, managing organizations and creating organizations and other basic functions, while completing advanced functions such as recommendation systems and sensitive word checking.

The paper first introduces the project background and the SSM framework used in development, DFA and item-based collaborative filtering. Then the overall requirements of the system are introduced, and on this basis, the functional and non-functional requirements of the system are developed. After the requirements are clear, the outline design of the system is described, mainly the system structure diagram and the data flow diagram. Then introduced the detailed design of the system, from the design ideas, business processes, module description, interface description, class diagram and database design, detailed description. Then started the implementation of the system, introduced the logic of the background code, and showed some small program interface and background management system interface.

In the system development process, the small program side adopts mpvue, the background server adopts SSM development framework, and the background

management system adopts vue.

Keywords: B/S architecture, SSM framework, DFA, item-based collaborative filtering

第 1 章 绪论

1.1 项目背景

随着人们对精神生活要求越来越高，涌现出很多形式各异的活动让人们体验参与。这就需要一个平台对众多的活动进行管理，人们可以在这个平台上发布活动，寻找同伴；也可以参与自己感兴趣的活动，对参与的活动进行评论打分。这就是现在报名助手设计的初衷。

微信小程序这几年在大陆非常流行，有着不亚于 App 的使用体验。目前市场上有很多微信小程序的报名助手，他们都有自己的优点，可以给我们开发新的报名助手提供借鉴。众多报名助手小程序中，小立报名上手容易，界面简洁美观，功能齐全。所以在项目的实现过程中，参考了小立报名这个小程序，在完成小立报名的基本功能后，添加了一些新的功能，比如推荐系统，敏感词过滤等等。

从系统安全性，可靠性和可移植性出发，本文选择基于 J2EE 的 SSM（Spring, SpringMVC, MyBatis）集成开发框架进行系统的实现。SSM 框架是现阶段流行的 web 开发框架，该框架中的数据持久层技术是 MyBatis。MyBatis 是一种“半自动”式的 ORM 实现方案，它在 SQL 开发的工作量和数据库移植性方面的让步，为系统设计提供了更大的自由空间[1]。

1.2 研究意义和研究目的

现在市场上的报名助手对用户上传的信息监管力度不够大，很多非法信息以及违法活动都可以上传，同时缺少推荐模块，没有实现对用户进行活动的精准推荐。我们设计实现的报名助手实现了这 2 个功能，如果用户发起非法活动或者组织都会被系统拒绝，并且被警告。如果用户参与过活动，那么用户每次进入系统时，系统都会给用户推荐 3 个该用户最可能参与的活动，节省用户搜寻活动的时间。

1.3 国内外关于这个论题的研究现状和发展趋势

目前国内关于报名助手相关的实现多以微信小程序的方式存在，且参与者相对较少，发起报名的活动多以公共活动和广告宣传为主。或在一些社交软件中作为辅助功能，专用于针对当前群的发起活动或投票。所以当前缺少功能更加完备且使用更加方便的报名类程序。微信小程序借由微信平台，可以很方便的获取用户，成为当前应用程序发展的热门方向，小程序开发也成为备受关注的技术。

自 21 世纪以来，Java 发展迅速，在计算机语言中常年排到前 5，使用 Java 的设备数以亿计。当然，这与 Java 语言拥有的众多优点是密不可分的，Java 除了是一门结构严谨、面向对象的编程语言之外，还摆脱了硬件平台的束缚，真正做到了“一次编 写，到处运行”[2]；实现了热点代码检测和运行时编译及优化，这使得 Java 应用能随着运 行时间的增加而获得更高的性能[3]；此外，Java 还提供了一个相对安全的内存管理和访问 机制，避免了绝大部分的内存泄漏和指针越界问题[4]。

时至今日，Java Web 框架已经很成熟，在这些技术的帮助下，程序的开发效率得到了极大的提高。近几年移动无线技术发展迅猛，传统的 Web 应用开始更多地支持移动设备，并衍生出系统需要进行多终端化改造的问题[5]。因此基于 B/S 架构、支持移动端的 Web 系统相关的技术与开发值得深入研究。本课题设计开发的报名助手微信小程序，服务端开源框架的技术优势使得本系统能很好满足用户的需求。

1.4 相关技术介绍

1.4.1 Spring 技术

Spring 是 2003 年兴起的一个轻量级 Java 开发框架，源于 Rod Johnson 在其著作 Expert One-On-One J2EE Development and Design 中阐述的部分理念[6]。它在很大程度上降低了企业应用开发的复杂性，提供了 Java EE 应用的一站式解决方案，向上可以与 MVC 框架无缝整合，向下可以与各种持久层框架整合，

因此具有很强的框架整合能力，是业务逻辑层首选的实现框架[7]。

简单地说，Spring 是一个轻量级的控制反转(Inverse of Control, IOC)和面向切面编程(Aspect-Oriented Programming, AOP)的容器框架，控制反转也称依赖注入，是一种降低对象之间耦合关系的设计思想[8]。面向切面编程是对面向对象思想的一种补充，原理是利用代理的设计模式，主要用于在不改变原来业务逻辑模型的基础上动态地增加日志、安全或异常处理等功能。可以实现对事务的管理，以及负责事务的回滚机制，隔离业务逻辑的各个部分，降低耦合度，提高程序的可重用性和开发效率[9]。

1.4.2 SpringMVC 技术

Spring MVC 分离了控制器、模型对象、分派器以及处理程序对象的角色，这种分离让它们更容易进行定制[10]。SpringMVC 实现了 MVC 的核心概念，它为 Controller 和处理程序提供了大量与此模式相关的功能，当向 MVC 添加反转控制时，它使应用程序高度解耦，通过配置修改可以动态更改组件，增强了系统的可扩展性和可维护性，如图 1 所示为 SpringMVC 设计模式的请求和响应过程。

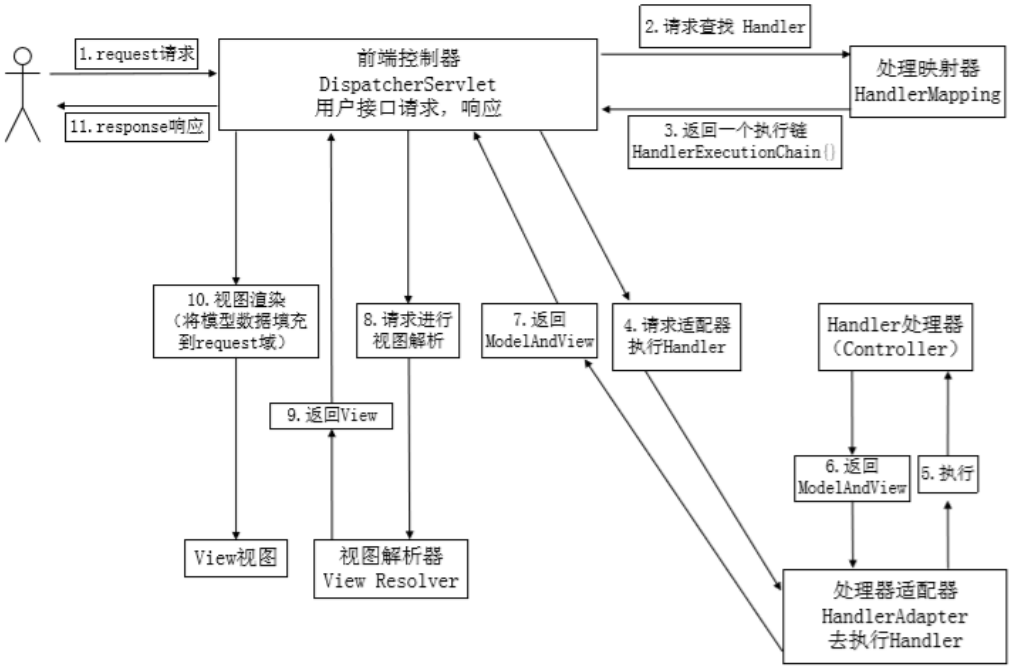


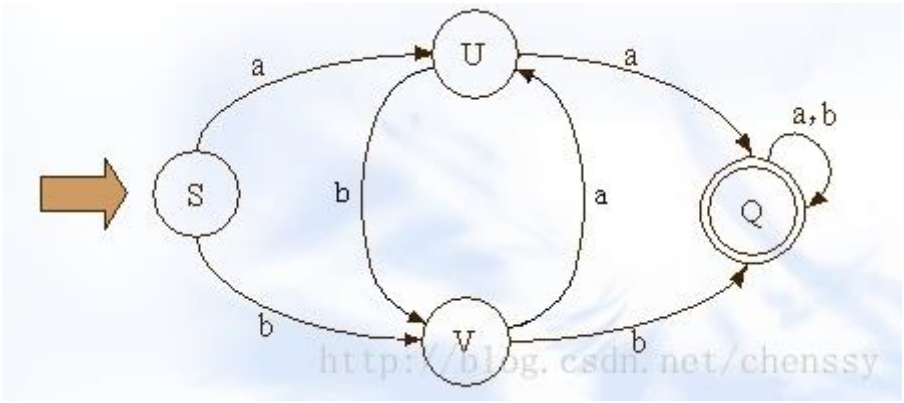
图 1. SpringMVC 设计模式的请求和响应过程

1.4.3 MyBatis 技术

MyBatis 来源于 Apache 的一个开源项目 iBatis，2010 年这个项目从 foundation 迁移到了 google code，并改名为 MyBatis。相对于全自动化的 Hibernate，MyBatis 虽为“半自动化”的 ORM 框架，但是其具有可以对 SQL 语句进行自由优化的优势，而且不需要花费精力去处理例如注册驱动、创建 connection、创建 statement、手动设置参数、结果集检索等 JDBC 繁琐的代码。[11]使用简单的 XML 或注解方法将要执行的各种 Statement 配置起来，通过 Java 的 POJOs 对象和 Statement 中的 SQL 进行映射生成最终的 SQL 语句，最后由 MyBatis 框架执行 SQL 并将结果映射成 Java 对象并返回[7]。

1.4.4 DFA 简介

在实现文字过滤的算法中，DFA 是唯一比较好的实现算法。DFA 即 Deterministic Finite Automaton，也就是确定有穷自动机，它是是通过 event 和当前的 state 得到下一个 state，即 event+state=nextstate。图 2 展示了其状态的转换：



在这幅图中大写字母（S、U、V、Q）都是状态，小写字母 a、b 为动作。通过上图我们可以看到如下关系

a b b

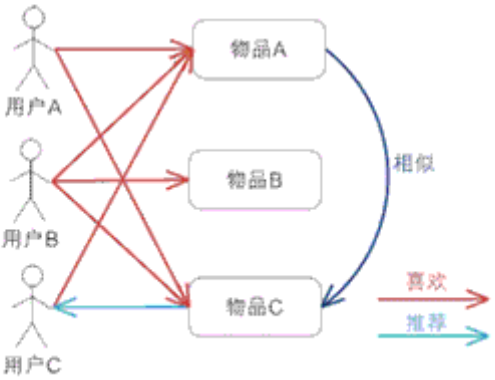
S -----> U S -----> V U -----> V

在实现敏感词过滤的算法中，我们必须要减少运算，而 DFA 在 DFA 算法中几乎没有什么计算，有的只是状态的转换。

1.4.5 基于 item 的协同过滤

基于 item 的协同过滤，通过用户对不同 item 的评分来评测 item 之间的相似性，基于 item 之间的相似性做出推荐。简单来讲就是：给用户推荐和他之前喜欢的物品相似的物品。

用户/物品	物品A	物品B	物品C
用户A	√		√
用户B	√	√	√
用户C	√		推荐



Iterm-based 的基本思想是预先根据所有用户的历史偏好数据计算物品之间的相似性，然后把与用户喜欢的物品相类似的物品推荐给用户。

1.5 本文组织结构

本文根据软件开发的流程，对报名助手小程序后台的设计与实现进行展开。每个章节的内容如下：

第一章 绪论。介绍了项目背景，研究意义和研究目的，国内外关于这个论题的研究和发展趋势，最后对系统实现过程中涉及的技术（SSM 框架，DFA 算法实现敏感词检查，基于物品的推荐系统和情感分析）进行了简要介绍。

第二章 需求分析。首先介绍了系统的总体需求，然后按照这个分析描述了系统的功能性需求和性能需求，确定将系统划分为五个模块：活动，组织，用户，后台管理，推荐系统。

第三章 概要设计。介绍了系统的设计思路，关键技术，系统结构图和数据

流向图，接着介绍了系统的开发环境和硬件环境。

第四章 详细设计。这一章是整篇论文的核心，对每个业务流程画了时序图并且进行了详尽的描述，接着对系统的五大模块要实现的功能分别进行详细的描述，在此基础上设计了后台要实现的接口。最后根据数据库的设计原则和系统需求设计了数据库，包括数据库的逻辑设计和物理设计。

第五章 系统实现。首先介绍了项目的搭建和配置，然后从五大模块入手，介绍了前端调用后台后代码的执行流程。同时，也展示了部分页面的设计结果。

第六章 总结和展望。在整个系统设计和实现的过程中，自己的学习以及体会，对本文进行了一个总结，同时提出了系统中的漏洞有待进一步完善。

第 2 章 需求分析

2.1 系统总体需求

2.1.1、Software perspective 软件概述

报名助手是基于微信小程序的一款社交工具，是人们管理和参与活动的一个平台。在这个平台中，有超级管理员和普通用户，超级管理员是平台的负责人，维护平台的正常运行，超级管理员拥有最高权限，可以查看发布到平台上的所有活动和组织，以及所有登录过的用户信息，同时也可以删除一些过期的活动，解散一些组织。普通用户是平台的使用者，可以发起活动，参与活动，关注活动，管理活动，创建组织，管理组织和参与组织。用户发起活动时需要填写相关的信息，并且后台会检测信息中是否含有非法内容，若用户发布的信息中含有非法内容，小程序端会发出警告。用户可以在探索中寻找自己感兴趣的活动，同时在地图上推荐 3 个用户可能感兴趣的活动。具有管理和发起活动权限的用户可以删除或者修改活动。对于组织，情况和活动类似，组织存在的目的是管理活动和用户。

2.1.2、About the Project 项目介绍

项目主要分为 3 部分，小程序端，后台服务器，后台管理系统。小程序端是用户界面，向用户显示各种信息和服务，同时和后台服务器进行数据的交互。后台服务器主要完成和数据库的交互，同时完成敏感词检查，推荐系统，情感分析等一系列其他的要求。后台管理系统完成数据库所有数据的显示，同时要显示他们之间的联系。

2.1.3、Environment of Product 产品环境介绍

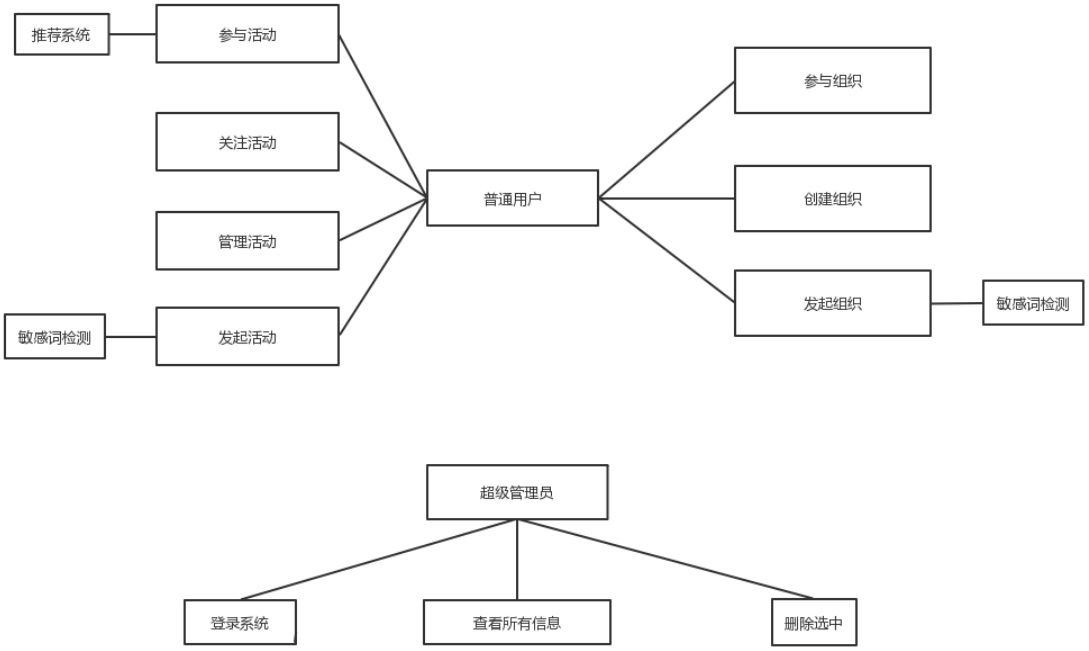
1. 硬件环境：

Debain 云服务器（用于远程数据库的部署），BosonNLP 远程服务器， 个人 PC。

2. 软件环境：

Tomcat , jdk , Spring+SpringMVC+Mybatis，MySQL，Node.js。

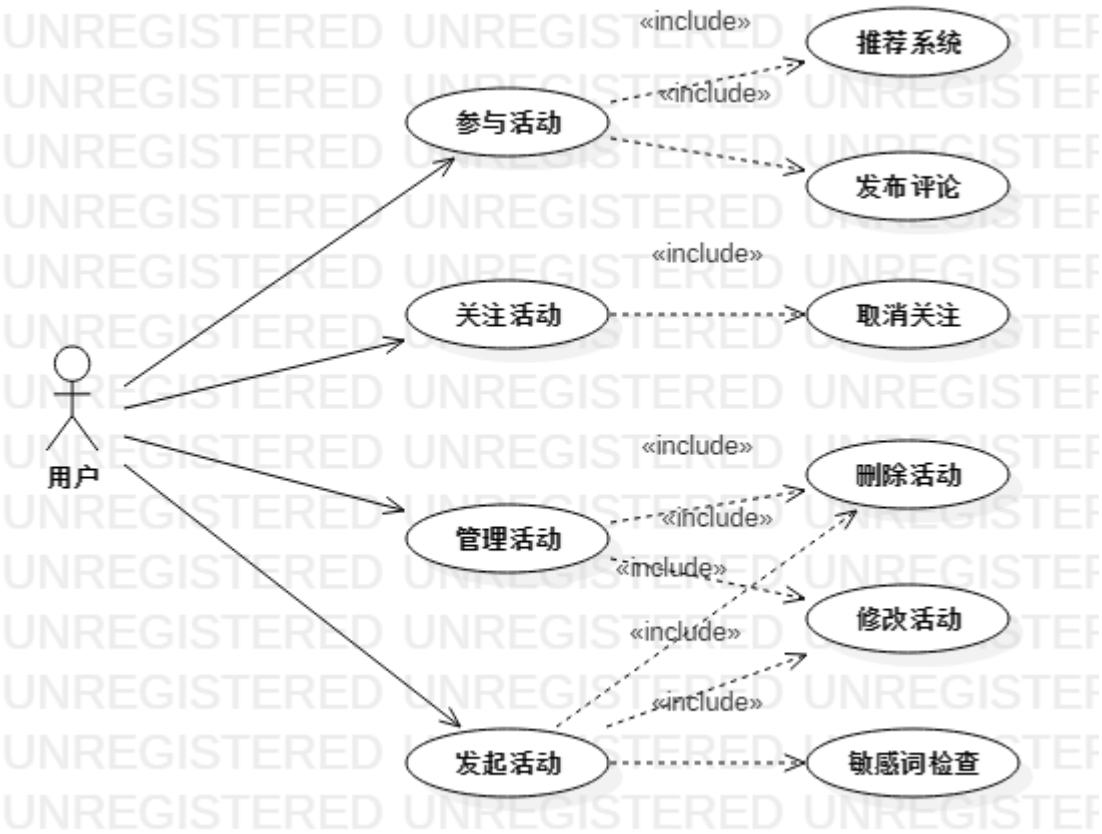
2.1.4、Software function 软件功能



软件功能图

2.2 系统功能性需求

2.2.1 用例图



2.2.2 用例说明

Use Case 1 用例 1 参与活动

Goal in Context 简要说明

用户通过探索，在地图上发现自己感兴趣的活动，点击对应的图标和小简介进入活动详情页，选择报名活动即可报名该活动，系统会自动获取用户信息。

Preconditions 前置条件

用户通过地图上或者其他的活动简介页面点击进入活动详情页面;用户已经填写了自己的信息；活动的最后报名截止时间还没有到；报名人数尚未达到上限，用户尚未报名该活动。

End Condition 后置条件

Success End Condition 成功后置条件

界面显示用户报名成功，在“我参与的活动”列表中可以查看到该活动。

Failed End Condition 失败后置条件

界面显示报名失败，并且给出可能的原因，用户满足要求后可以重新报名。

Actors

用户

Trigger 触发条件

用户选择报名活动。

Description 基本事件流描述

1、 用户通过简介进入活动详情页。

2、 用户在活动详情页可以看到活动的详细介绍，如果确认报名，点击最下面的报名活动。否则返回即可。

3、 用户点击报名后，若符合要求系统给出提示，显示报名成功。

Use Case 2 用例 2 关注活动

Goal in Context 简要说明

用户进入活动详情页后，可以关注该活动。对于已经关注的活动，用户也可以取消关注。

Preconditions 前置条件

用户已经填写自己的信息；用户还没有关注该活动。

End Condition 后置条件

Success End Condition 成功后置条件

界面显示用户已关注该活动，在“我关注的活动”列表中可以查看到该活动。

Failed End Condition 失败后置条件

活动关注报名失败，界面给出提示，用户可以重新尝试关注。

Actors

用户

Trigger 触发条件

用户选择关注活动。

Description 基本事件流描述

1、 用户点击“关注”直接关注该活动，活动详情页将显示用户已经关注该活动。

2、 我关注的活动列表可以查看该活动。

3、 用户选择取消关注，活动详情页显示用户未关注，同时取消我关注的活动中的显示。

Use Case 3 用例 3 管理活动

Goal in Context 简要说明

活动的管理员可以在活动的详情页面进入活动的统计信息页面，统计页面显示活动的参与者信息。管理员具有删除活动或者修改活动的权限。

Preconditions 前置条件

用户是活动的管理员。

End Condition 后置条件

Success End Condition 成功后置条件

活动信息被用户管理员修改或删除，非法用户也会被管理员剔除。

Failed End Condition 失败后置条件

系统显示用户没有权限，用户无法进行相关操作。

Actors

用户

Trigger 触发条件

管理员用户进入活动详情页。

Description 基本事件流描述

1、 用户点击“管理”。

2、 界面显示可以管理的项目“报名数据，修改活动，删除活动”。

3、 用户点击报名数据，系统显示报名名单信息，用户可查看，统计，删除或导出名单数据。

4、 当用户属于活动的创建者时，用户点击“管理员”，系统提供当前管理员列表，用户可以点击某个管理员进行删除，或点击添加管理员，系统生成二维码用于其他用户扫描成为活动管理员，创建者同样可以删除该活动，则系统同时删除

活动相关数据。

5、 用户可以点击修改活动，系统进入修改活动页面，用户可以对活动信息进行修改，点击提交，活动即被修改，系统将返回活动详情页面.

6、 用户可以选择一项留言并删除。

7、 用户点击删除活动，与该活动相关的所有的数据都会被删除。

Use Case 4 用例 4 发起活动

Goal in Context 简要说明

用户可以发起自己的活动。

Preconditions 前置条件

用户已经向系统提交自己的信息；在发起活动界面填写了相关信息。

End Condition 后置条件

Success End Condition 成功后置条件

活动显示用户已创建该活动，在“我创建的活动”列表中可以查看到该活动。

Failed End Condition 失败后置条件

系统显示发起活动需要填写的必要信息不足，尝试重新发起活动。

Actors

用户

Trigger 触发条件

用户点击“发起活动”

Description 基本事件流描述

1、 用户点击“发起活动”进入发起活动界面。

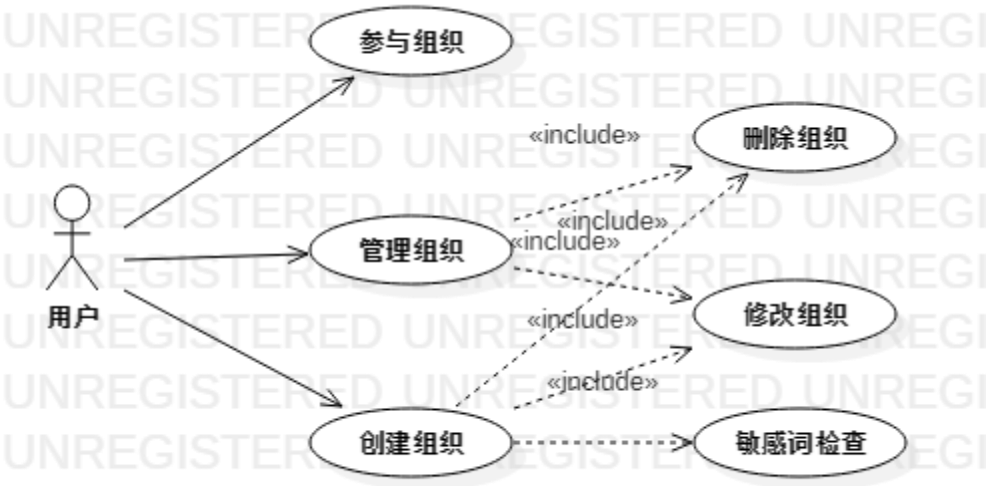
2、 用户按照要求填写活动的详细信息，上传相关图片。

3、用户点击“发布”，则该活动将发布出去，如果用户设置了相关的地理位置，则可以在探索栏目中查看地图中自己发起的活动。

4、 用户点击“我发起的活动”可以查看到自己创建的活动简介列表。

5、 用户进入自己创建的活动的详情页时，将获得创建者和管理员权限。

2.2.3 用例图



2.2.4 用例说明

Use Case 5 用例 5　创建组织

Goal in Context 简要说明

用户在创建组织界面填写相关信息后，即可创建组织。

Preconditions 前置条件

用户填写相关个人信息；在创建组织界面填写组织的详细信息。

End Condition 后置条件

Success End Condition 成功后置条件

成功创建的组织在“我创建的组织”页面中以列表的方式显示出来。

Failed End Condition 失败后置条件

系统提示创建组织必须填写相关信息，用户填写完毕后重新发布。

Actors

用户

Trigger 触发条件

用户点击“创建组织”进入创建组织界面。

Description 基本事件流描述

1、 用户进入组织创建界面。

2、 用户填写该组织的相关信息，比如组织名称，地址，电话等。

3、 用户点击提交创建该组织。

4、 系统显示创建成功，查看我创建的组织可以看到我创建的组织列表。

Use Case 6 用例 6　管理组织

Goal in Context 简要说明

组织的管理者可以管理组织发起的活动，组织中的用户，组织本身的信息，包括修改和删除。

Preconditions 前置条件

用户是组织的管理者。

End Condition 后置条件

Success End Condition 成功后置条件

组织信息，组织发起的活动信息，组织中的用户信息被管理者修改或者删除。

Failed End Condition 失败后置条件

系统提示用户没有组织的管理权限。

Actors

用户

Trigger 触发条件

用户点击“管理”进行组织的管理。

Description 基本事件流描述

1、 用户在管理的组织中点击“管理”，界面显示组织信息修改或者删除，组织中用户信息删除或修改，组织发起的活动信息删除或修改。

2、 用户选择任意一项即可进行相关信息的修改和删除。

Use Case 7 用例 7　参与组织

Goal in Context 简要说明

用户查看所有组织的简介，点击自己感兴趣的组织，进入组织详情页报名即可。

Preconditions 前置条件

用户个人信息系统已经获取；用户尚未参与该组织。

End Condition 后置条件

Success End Condition 成功后置条件

成功参与的组织在“我参与的组织”页面中以列表的方式显示出来。

Failed End Condition 失败后置条件

系统提示参与组织失败，用户重新报名参与。

Actors

用户

Trigger 触发条件

用户点击组织简介列表进入组织详情页面。

Description 基本事件流描述

1、 用户点击组织简介列表。

2、 进入组织详情页，点击最下面的参与组织。

3、 系统显示参与组织成功，查看我参与的组织可以看到刚刚参与成功的组织。

2.2.5 用例图

2.2.6 用例说明

Use Case 8 用例 8 后台管理员登录

Goal in Context 简要说明

超级管理员进入后台管理系统时先要进行登录，只有正确的账号密码才能登录进入后台管理系统，查看信息，删除信息。

Preconditions 前置条件

用户具有超级管理员权限。

End Condition 后置条件

Success End Condition 成功后置条件

用户进入后台管理系统。

Failed End Condition 失败后置条件

系统提示账号密码错误。

Actors

后台管理员。

Trigger 触发条件

用户登录后台管理系统。

Description 基本事件流描述

1、 用户填写账号密码

2、 账号密码正确进入后台管理系统，否则拒绝。

Use Case 9 用例 9 查看所有信息

Goal in Context 简要说明

超级管理员进入后台管理系统后，可以查看组织，活动，用户的所有信息，以及他们之间的联系。

Preconditions 前置条件

用户进入后台管理系统。

End Condition 后置条件

Success End Condition 成功后置条件

无。

Failed End Condition 失败后置条件

无。

Actors

后台管理员。

Trigger 触发条件

用户进入后台管理系统，并且点击相关选项。

Description 基本事件流描述

1、 用户进入后台管理系统。

2、 点击不同选项，查看不同的信息。

Use Case 10 用例 10 删除选中

Goal in Context 简要说明

超级管理员进入后台管理后，可以删除任何信息。

Preconditions 前置条件

用户进入后台管理系统。

End Condition 后置条件

Success End Condition 成功后置条件

相关信息被删除。

Failed End Condition 失败后置条件

无。

Actors

后台管理员。

Trigger 触发条件

用户选择一些数据项，点击“删除选中”

Description 基本事件流描述

1、 用户进入后台管理系统。

2、 查看相关信息时，可以选择任意信息，点击删除选中，即可删除。

2.3 系统性能需求

除了满足基本的功能性需求，还需要考虑下面的非功能性需求：

（1）安全性：首先考虑的是系统软件的安全性，应该选择合理的开发模式，搭建安全可靠的结构。也应该注意接口的安全，防止 SQL 注入或者页面篡改或者其他的非法操作。最后应该确保用户自身信息的安全性，防止用户数据泄露。

（2）健壮性：小程序端如果传给后台非法数据，后台应该有若错能力，可以通过设置参数的默认值来增加系统的健壮性。在测试环节，针对每一个模块，进行详细的测试，考虑尽可能多的边界值，保证系统的健壮性。

（3）稳定性：当多个用户同时进入系统，应该保持系统的响应速度。系统应该具有并发处理能力。开发过程中尽量优化代码，采用高效算法，从而提高系统的稳定性。

（4）及时性：用户提交或者获取数据时，系统应该对用户数据及时处理。在处理过程中，耗时的是网络传输数据，所以尽量减少网络数据传送，用内存来换取速度。

（5）可操作性：小程序端界面设计应该简单美观，后台服务器算法设计高效，后台管理系统界面也应该简单合理。

2.4 System Moudle 系统模块

“活动”模块

功能名称	子功能
活动	插入活动
	删除活动
	更新活动
	查找活动

“插入活动”子模块

a. 描述：

在主界面上，用户选择发起活动，填写规定的信息后，点击确定即可。

b. 输入：活动必要的信息。

c. 加工：将用户输入的信息写入数据库。

d. 输出：无。

“删除活动”子模块

a. 描述：

在我管理或关注的活动中，选择一个删除即可。

b. 输入：无。

c. 加工：删除数据库中所有与该活动相关的数据。

d. 输出：界面显示发生变化。

“更新活动”模块

a. 描述：

活动的管理员或者发起者点击管理，选择更新活动，进入活动详情页，修改部分信息提交即可。

b. 输入：需要修改的数据项。

c. 加工：将新的数据写入数据库。

d. 输出：该活动的详情页发生更新。

“查找活动”模块

a. 描述：

在主界面上，用户选择“搜索”，进入地图中，就可以查找活动。

b. 输入：无。

c. 加工：返回数据库中活动的详细信息。

d. 输出：无。

“组织”模块

模块名称	子模块
组织管理	插入组织
	删除组织

	更新组织
	查找组织

“插入组织”子模块

a. 描述：

用户填写相关的组织信息，提交即可。

b. 输入：介绍组织的相关信息。

c. 加工：将组织信息写入数据库。

d. 输出：显示创建成功并进入组织详情页。

“删除组织”子模块

a. 描述：

组织的管理者或者创建者可以删除组织。

b. 输入：无

c. 加工：删除数据库中所有与该组织相关的信息。

d. 输出：与该组织有关的所有界面发生更新。

“更新组织”子模块

a. 描述：

组织的管理员或者创建者选择待更新的活动，进入组织详情页，填写想修改的信息，提交即可。

b. 输入：需要修改的组织信息。

c. 加工：更新数据库中关于组织的信息。

d. 输出：组织详情页发生变化。

“用户”模块

“分享活动”模块

模块名称	子模块

用户	个人信息
	用户-活动
	用户-组织

“个人信息”子模块

a. 描述：

进入系统后，用户可以修改个人信息。

b. 输入：用户填写的个人信息。

c. 加工：将用户的相关信息写入数据库。

d. 输出：无。

“用户-活动”子模块

a. 描述：

这部分包括用户参与的活动，关注的活动，管理的活动和发起的活动。

b. 输入：用户 id，活动 id。

c. 加工：对用户参与的活动，用户关注的活动，用户管理的活动，用户发起的活动和活动表进行增删改查。

d. 输出：界面发生更新。

“用户-组织”子模块

a. 描述：

这部分包括用户参与的组织，管理的组织和创建的组织。

b. 输入：用户 id，活动 id。

c. 加工：对用户参与的组织，管理的组织，创建的组织和组织表进行增删改查。

d. 输出：界面发生更新。

“后台管理”子模块

模块名称	子模块
后台管理	登录系统

	查看主要信息
	删除选中

“登录系统”子模块

a. 描述：

超级管理员进入后台管理系统时，需要利用账号密码进行登陆。

b. 输入：无。

c. 加工：将超级管理员的账号和密码与系统预设的进行比对，一致则进入系统。

d. 输出：无。

“查看主要信息”子模块

a. 描述：

在后台管理系统中，超级管理员可以查看活动，组织，用户以及他们之间的关系。

b. 输入：无。

c. 加工：将数据库中的所有数据表显示。

d. 输出：界面刷新。

“删除选中”子模块

a. 描述：

超级管理员选择不需要的数据项，点击删除选中即可删除。

b. 输入：无。

c. 加工：数据库中相关信息被删除。

d. 输出：界面刷新。

“推荐系统”模块

模块名称	子模块
推荐系统	基于 item 的协同过滤

“基于 item 的协同过滤”子模块

a. 描述：

参加过一些活动用户，再次登录系统，进入地图选择活动时，系统会根据用户以往选择的活动向用户推荐 3 个用户可能感兴趣的活动。

b. 输入：用户 id。

c. 加工：调用情感分析远程接口，再利用已有的 Java 库实现。

d. 输出：推荐的活动。

2.5 Interface Requirements 接口需求

报名助手软件环境：windows 操作系统， JAVA 环境 JDK1.8+Tomcat 8.5， 数据库 Mysql5.5， IDEA，百度地图。在用需要探索活动时使用百度地图的 API，当用户需要导航至活动地址时，系统跳转至百度地图 APP。

2.6 本章小结

本章首先讲了报名助手这个系统的总体需求，然后介绍了系统的功能性需求和非功能性需求，在此基础上对系统进行了模块划分，介绍了每个模块需要实现的功能，最后提到了系统的接口设计。

第 3 章 报名助手小程序服务器端概要设计

3.1 总体概述

3.1.1 Software perspective 软件概述

系统采用了 B/S 模式的体系结构，数据库采用的是 MySQL。小程序端采用的是 mpvue；服务器端采用的是 spring + springMVC + MyBatis，这是 Javaweb 端成熟的框架，采用 SSM 框架简化了编码难度，加快了开发进度；后台管理系统采用的是 vue.js，与后台交互采用的是 vue 中 axios。

3.1.2 Design Considerations 设计思路

松耦合原则：小程序端，后台，后台管理界面，都具有很强的独立性，界面编码过程中都是面向组件的，后台对数据的操作是分层操作的，每个模块对系统的其他模块的影响都比较小。

安全性原则：为了用户信息的安全，小程序端无法获取后台的用户 openid 数据；同时后台利用 spring 拦截非法请求，小程序端的每个数据请求都会经过后台的 controller 分发到不同的控制映射器；后台管理系统需要超级管理员的账号密码权限，非法用户无法登录进行操作。所有的这些操作都保证了数据传送，加载过程以及用户信息的安全性。

可扩展性原则：后台采用 SSM 框架进行开发，当所有的配置文件写好，实体类设计完整，需要新的数据访问时，只需很简单的代码编写即可，整个后台具有很强的可扩展性。

3.1.3 Key Technologies 关键技术

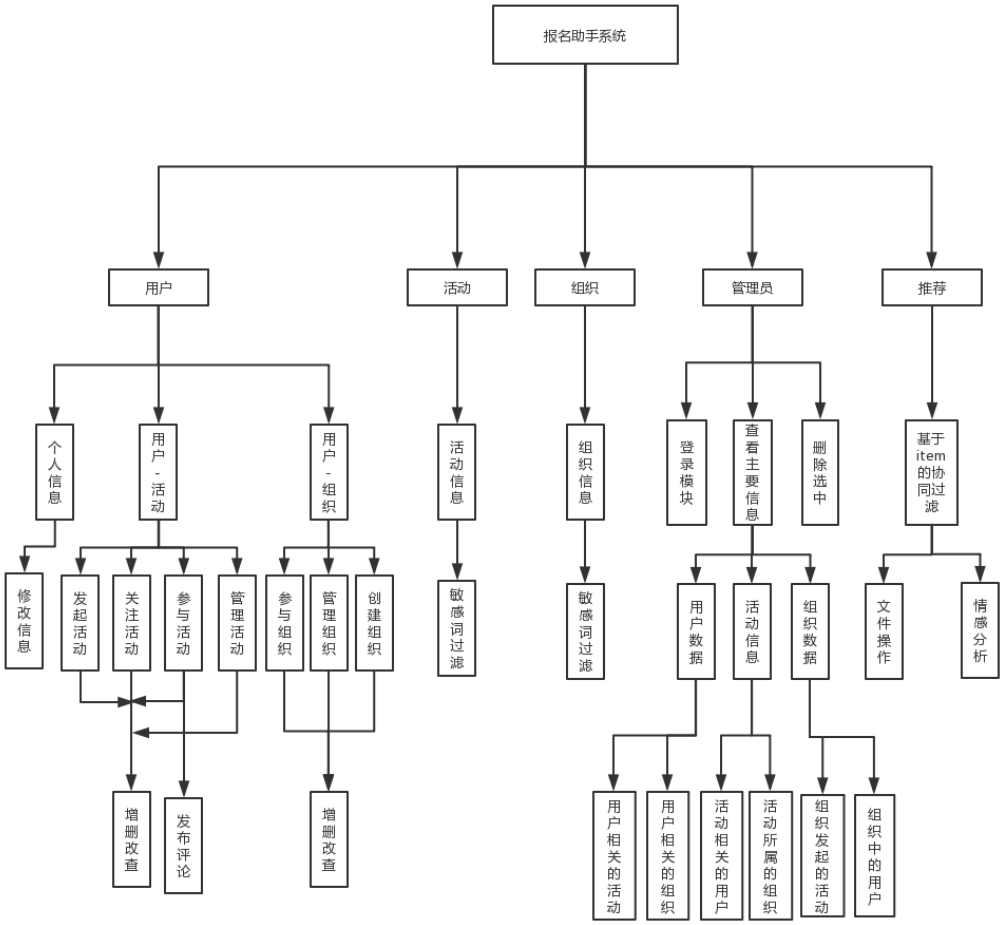
Spring:（1）IOC 控制反转和 DI 依赖注入：对象的创建，控制，销毁全部交由 spring 来管理，增加了编码的效率，实现解耦合（2）AOP 面向切面编程 （3）对其他框架的支持

SpringMVC：实现 web 层的分层设计，实现解耦合。

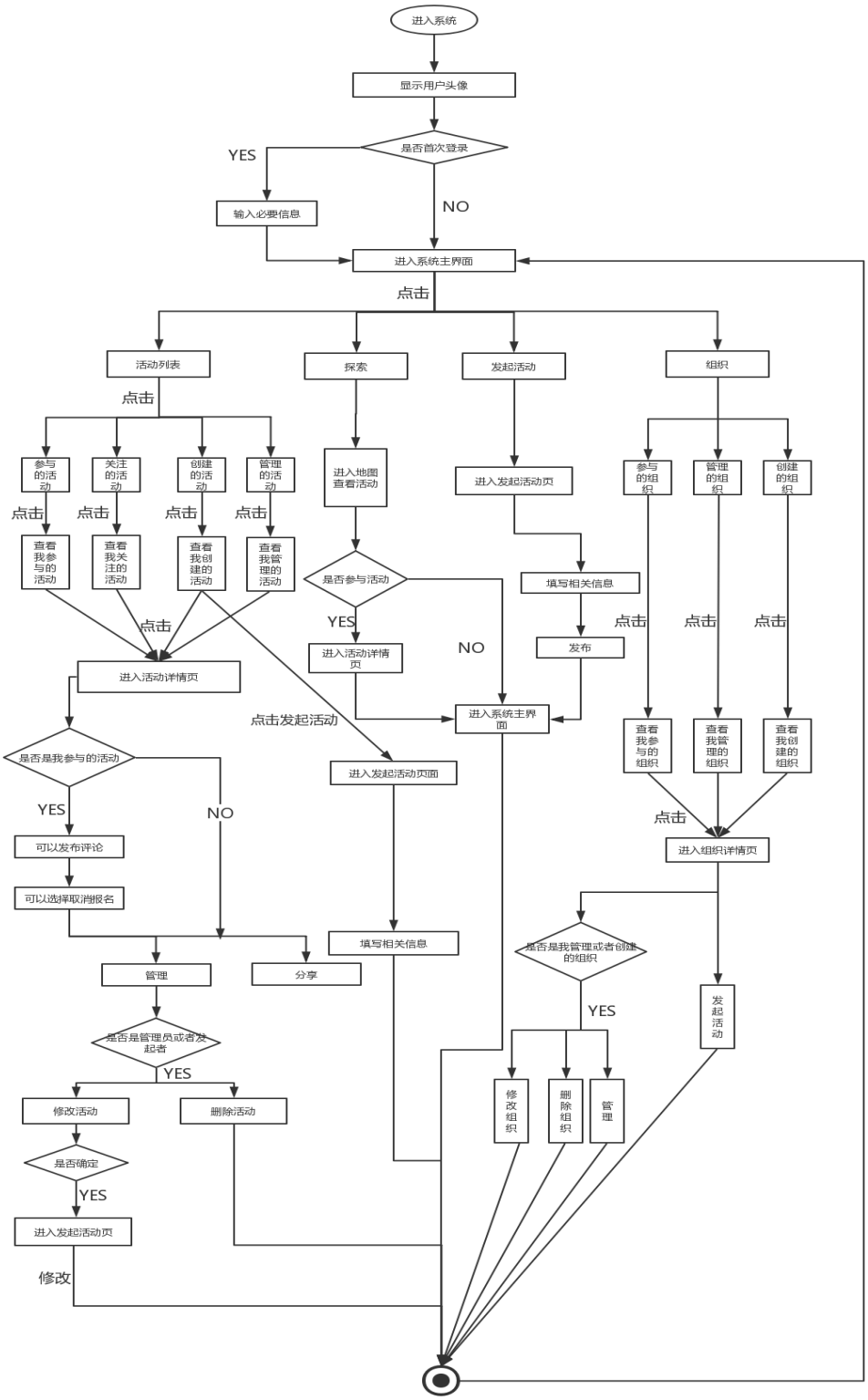
MyBatis: （1）写好配置文件后，所有对数据库的操作只需在 mapper.xml 文件中书写 sql 语句即可，避免在 Java 中对数据库进行硬编码。（2）利用 ResultMap 可以实现数据库的格式化数据与 beans 对象中的属性对应，方便数据库的操作。（3）对动态 sql 的支持 SSM 框架可以实现完美的整合，对 web 层解耦合，降低编码的难度。

3.1.4 System structure diagram 系统结构图

根据设计，系统分为用户模块，管理员模块，活动模块，组织模块，推荐模块。系统结构图如下所示：



3.1.5 Data flow diagram 数据流向图



用户进入系统，如果是首次登入，那么用户需要填写相关信息，比如年龄，姓名，性别等等，这样用户才可以参与活动。如果不是首次登录，用户直接进入系统主界面。在系统的主界面，有活动列表，探索，发起活动，组织四个选项，用户可以根据自己的需求进行选择。点击活动列表，用户可以看到参与活动，关注活动，管理活动，发起活动这 4 个按钮，点击参与活动，用户可以看到自己参与的活动，点击图标进入活动详情。关注活动和管理活动类似。用户可以在参与的活动中发起评论。点击发起活动进入发起活动页面，用户填写相关信息即可发起活动。提交数据的时候，后台会进行敏感词检查，如果用户上传非法信息，会被发起警告。如果用户是活动的管理员和发起者，可以对活动进行修改和删除。在主界面点击探索，进入地图，用户选择自己感兴趣的活动图标，可以显示活动的关键信息，点击进入活动详情页，可以参与报名。在主界面点击发起活动，用户进入发起活动页，填写相关信息可以发起活动。在主界面点击组织，可以查看自己参与的组织，管理的组织，创建的组织。点击参与的组织，用户进入组织详情页，可以查看组织信息。管理的组织类似。点击创建的组织，用户进入发起组织页，填写相关信息后发起活动，同时后台也会进行敏感词检查。如果用户是组织的管理者或者创建者，用户可以修改组织信息和删除组织。推荐系统调用文件操作和情感分析模块，每次用户进入系统，系统推荐 3 个用户可能感兴趣的活动。

3.2 Development Environment Design 开发环境设计

开发语言：JAVA

操作系统：windows10

数据库: MySQL5.5

数据库管理软件 Navicat

应用软件 JDK1.8 Tomcat8.5

开发软件 IDEA

3.3 Hardware Equipment 硬件设备

普通安卓手机，PC 一台，远程服务器。

3.4 接口描述

1. 后台服务器接口

http://localhost：8086/ EnrollAcvt

小程序端通过该接口与数据库进行数据交换。

2.BosonNLP 商业接口

http://api.bosonnlp.com/sentiment/analysis

向该接口传入待分析的文本，服务器端返回积极和消极的得分。

3.mahout 基于 item 的协同过滤推荐算法接口

这是 Apache 的一个开源库，利用这个库可以很容易实现基于物品的推荐系统。

3.5 本章小结

本章根据第二章的需求分析，对报名助手这个系统进行了概要设计。首先进行了总体概述，包括软件概述，设计思路，关键技术，系统结构图和数据流向图。接着介绍了系统的开发环境和硬件设备，最后进行了接口描述。本章为下一章的详细设计做铺垫。

第 4 章 报名助手小程序服务器端详细设计

4.1 Design Considerations 设计思路

4.1.1 Design Alternatives 设计可选方案

系统主要设计用户类，活动类，组织类等实体类，采用 SSM 框架，数据库采用 MySQL 存放活动数据，用户数据，组织数据。用户图片存放在静态资源文件夹中。系统额外功能如推荐系统，情感分析，敏感词检查将作为工具类来完成复杂逻辑。系统前端采用微信小程序+mpvue 设计界面，样式采用 css3 设计实现。后台管理系统采用 Vue 开发。对于部分用户数据，小程序端会使用小程序接口获取用户部分微信数据。

4.1.2 Design Constraints 设计约束

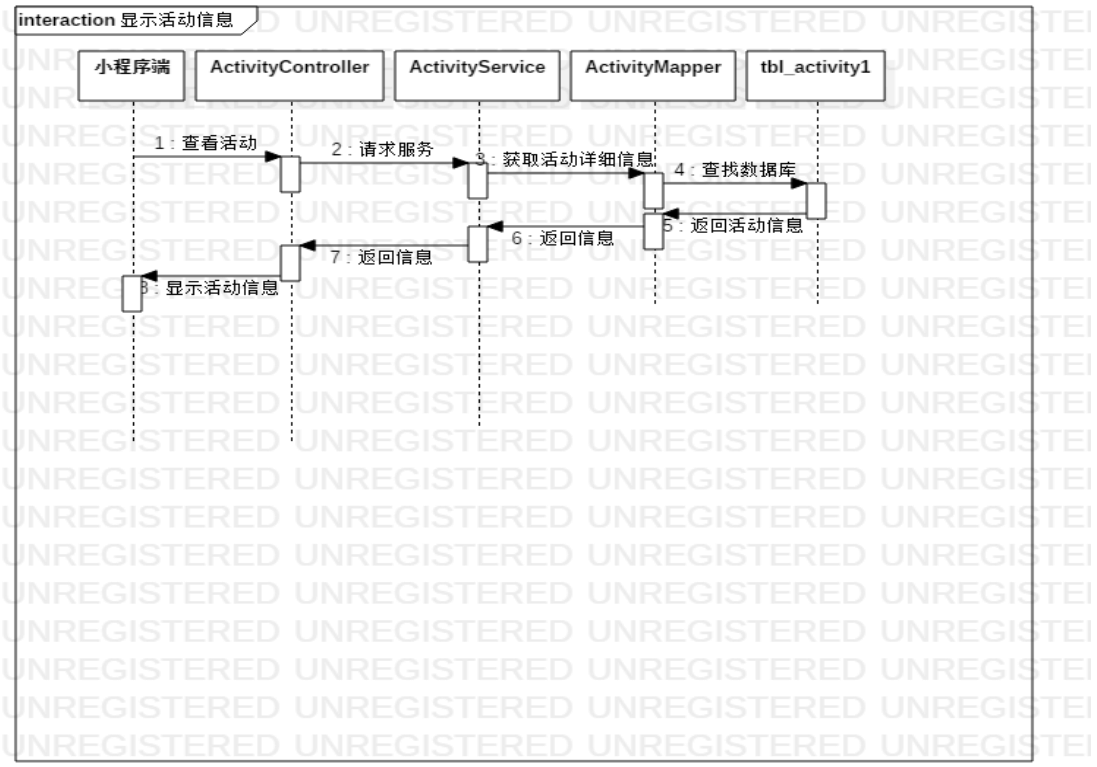
数据库之间的约束：数据库设计过程中会有五大约束：PRIMARY KEY 主键约束，UNIQUE 唯一性约束，DEFAUL 默认值约束，NOT NULL 非空约束，FOREIGN KEY 外键约束。在 tbl_activity1 中，orgId 必须是 tbl_org 中的 id，activityKey 满足默认值约束，默认情况下是 null；tbl_activity2 中的 id 必须和 tbl_activity2 中的 id 对应；tbl_user 中的 openid 满足非空约束；其余表中的 activityId，orgId，userId 必须满足外键约束。

遵循标准：后台开发遵循 SSM 框架，小程序端遵循微信小程序的接口标准和 mpvue 的标准，后台管理系统遵循 vue 的接口标准。

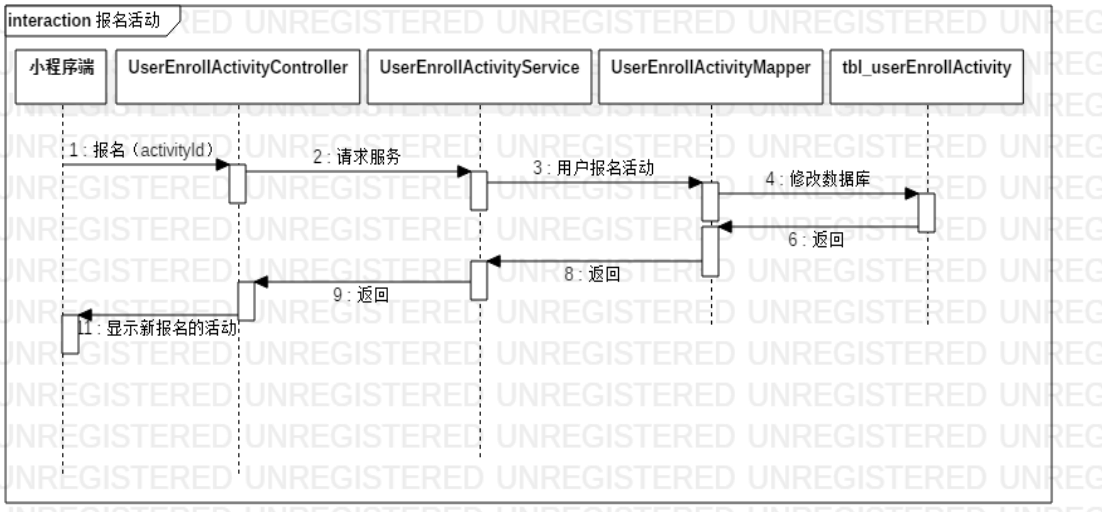
硬件限制：PC，安卓手机

技术限制：情感分析，推荐系统涉及到 NLP，概率论等，敏感词分析涉及离散数学，这些技术上的困难都是很大的。

4.2 Representation of the Business Flow 业务流程说明

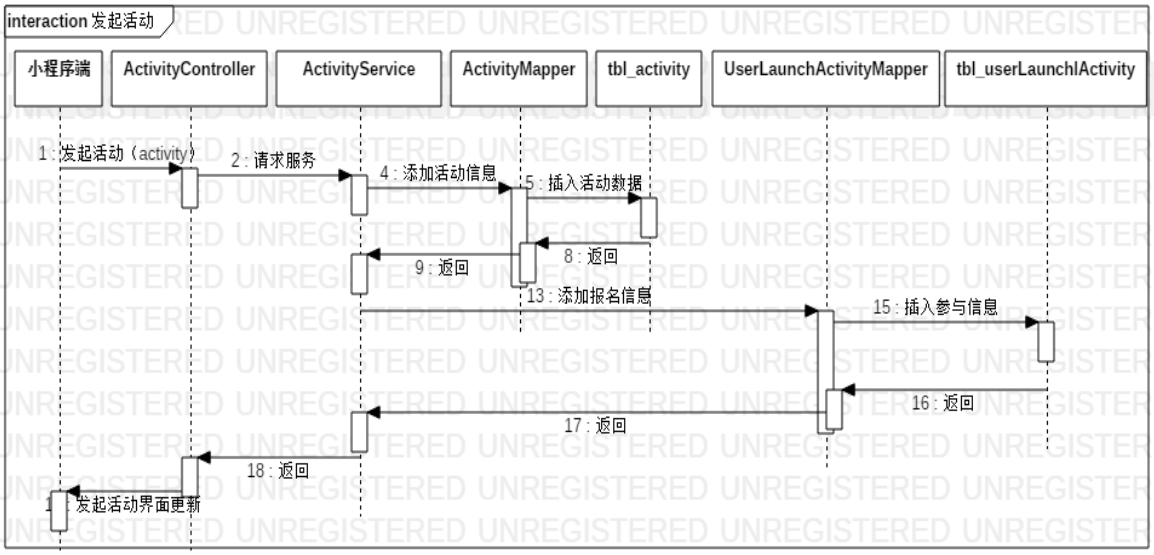


显示活动信息的时序图：用户点击自己参与/关注/管理/发起的活动，小程序端调用后台 controller 层中的显示活动信息的接口，controller 获取小程序端传来的活动 id 参数，再调用 service 层，最后 service 层调用 mapper 层，mapper 完成数据库操作返回活动数据。

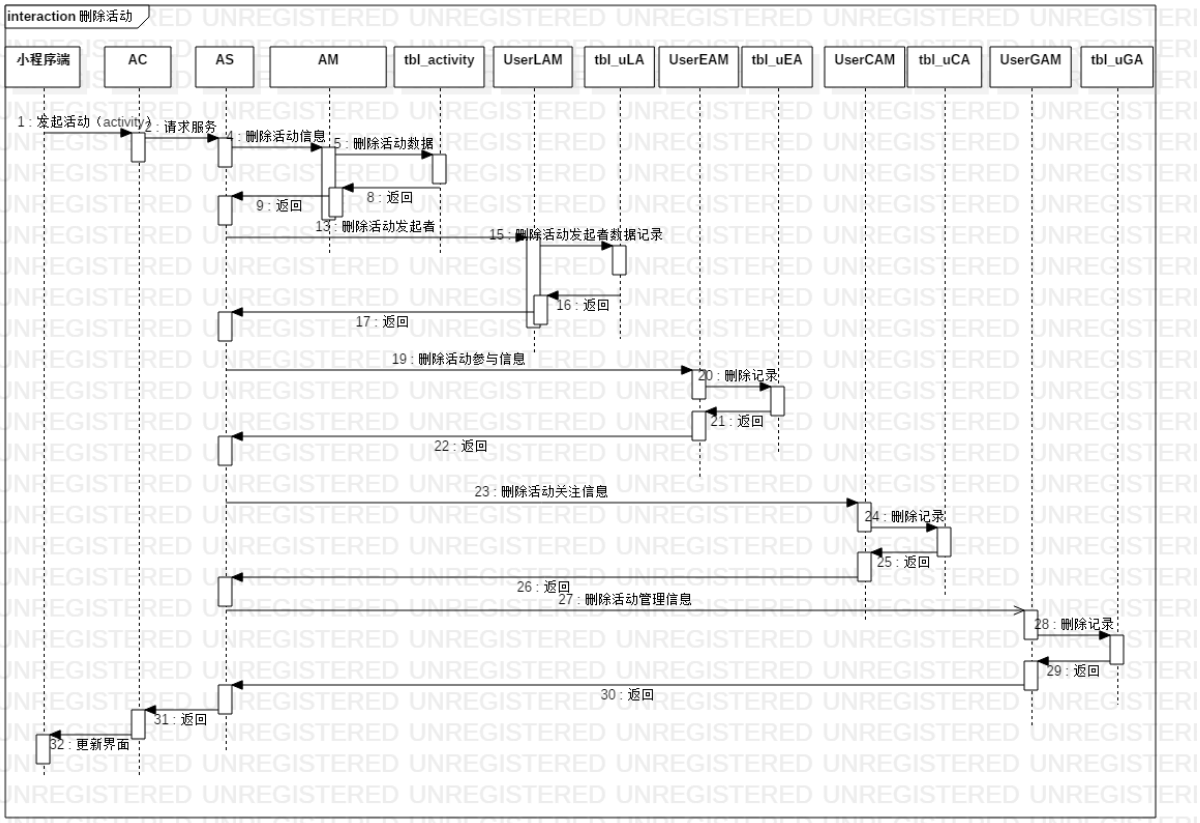


用户在小程序端点击报名活动，controller 拦截并分发请求，service 调用 mapper

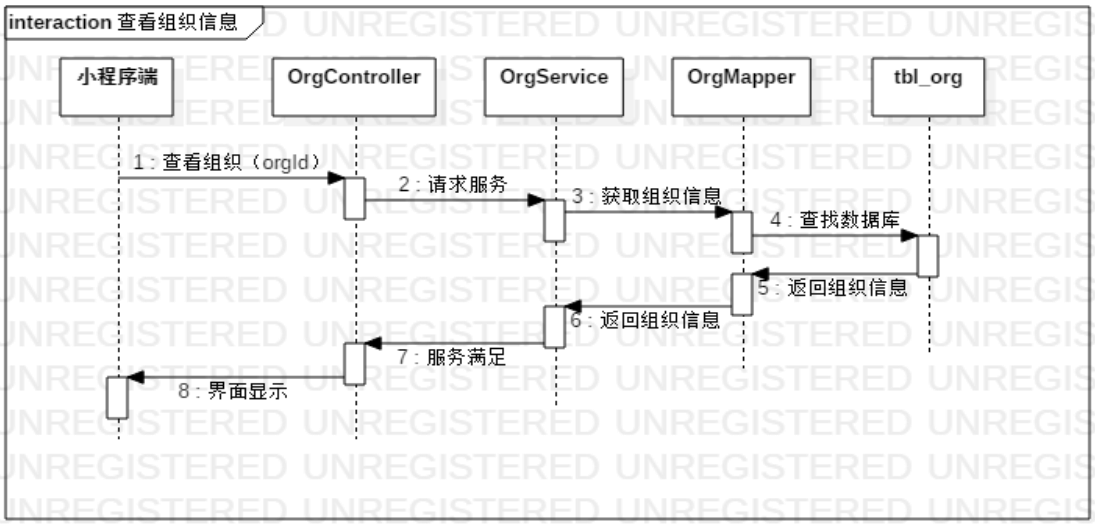
层，mapper 层在用户参与的活动数据表中插入一条新记录，同时在小程序端显示用户新报名的活动。



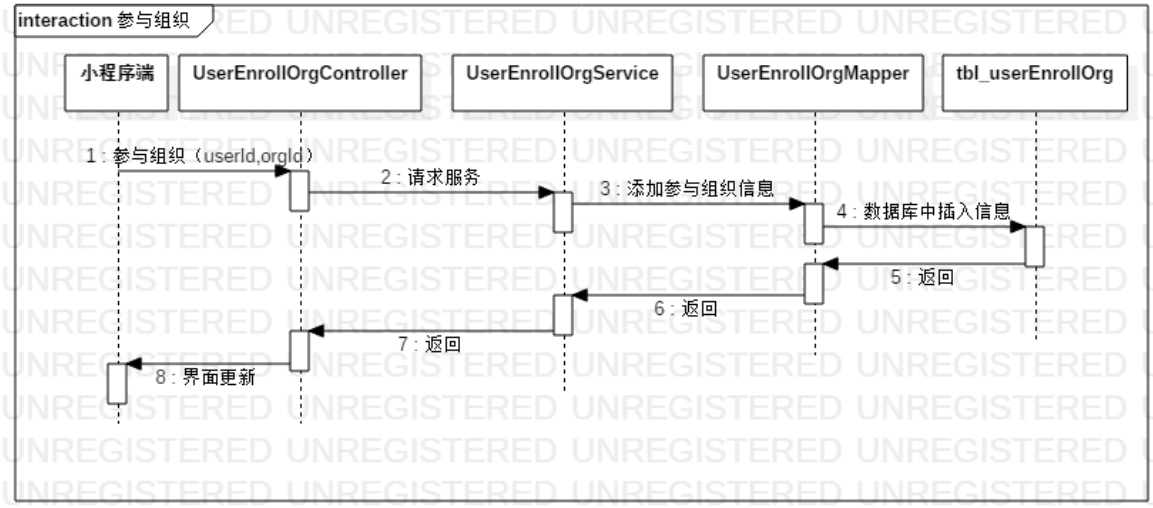
用户选择发起活动，小程序端将用户填写的信息发送到 controller 层，controller 封装信息，调用 service 层，service 需要调用 2 次数据库操作，先将活动信息写入活动表，再将用户 id 和活动 id 写入用户发起的活动数据表中，最后小程序端界面更新用户发起的活动。



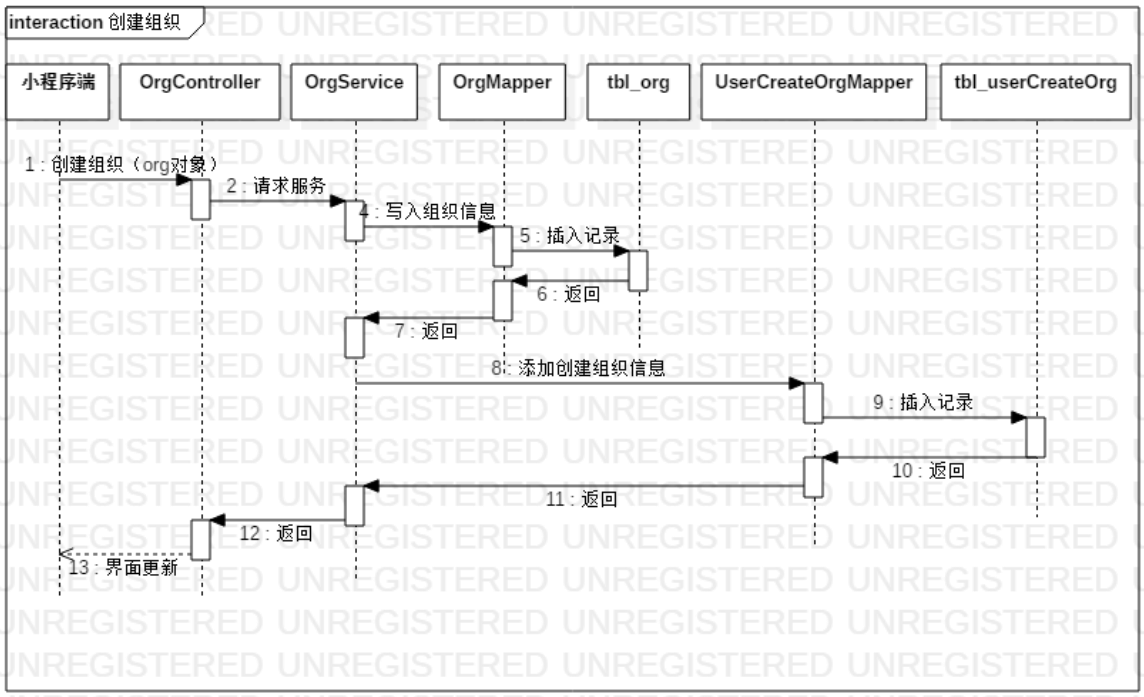
管理员或者发起者选择删除活动，小程序端向 controller 传送活动 id 参数，controller 调用 service 层，service 请求 5 次数据库的操作，分别删除活动表，用户参与活动表，用户关注活动表，用户管理活动表，用户发起活动表中的对应信息。最后，小程序端界面更新，已经删除的活动在界面上不再显示。



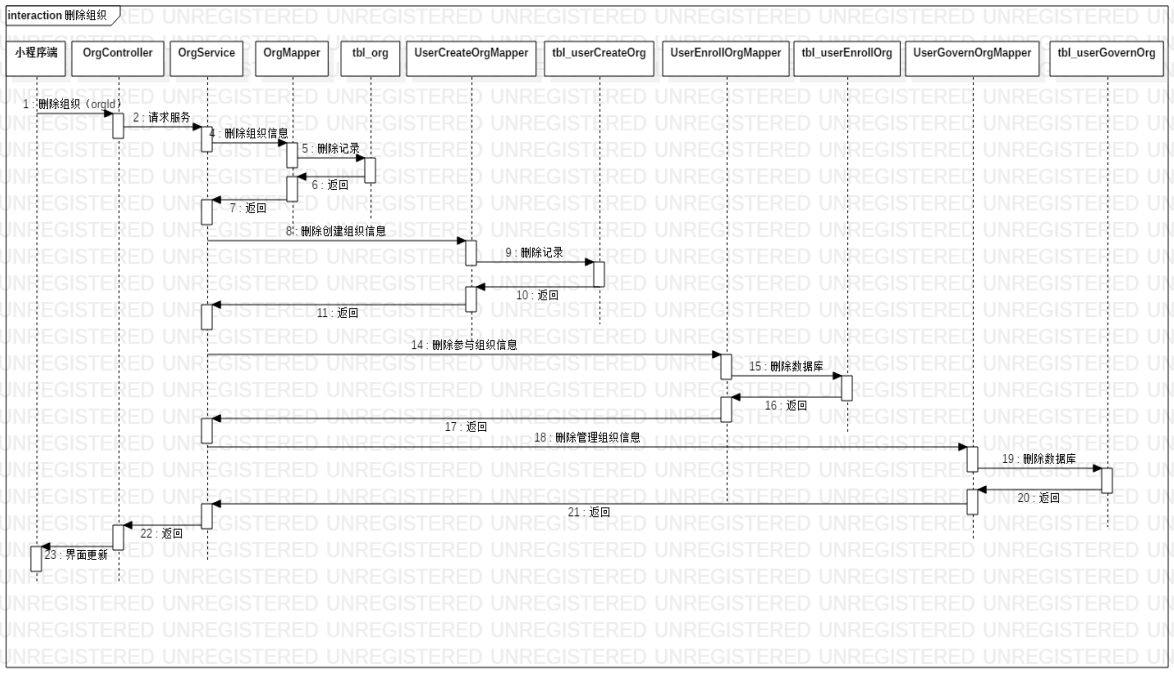
用户在小程序端查看我参与的组织，或者我管理的组织，或者我发起的组织，小程序端向 controller 传送组织 id，controller 调用 service 相应接口请求服务，service 层调用 mapper 层查找数据库，返回组织的详细信息，显示在小程序端的界面上。



用户在小程序端报名参与组织，controller 层接受 userId 和 orgId 这 2 个参数，向 service 层请求服务，service 层调用 mapper 层进行数据库的插入操作，同时界面更新。



用户在创建组织界面填写相关信息后，点击创建组织，小程序端将用户填写的数据传送到 controller 层，controller 将数据封装为 org 对象，调用 service 层的服务，service 层调用 2 次数据库操作，一次是将用户填写的组织信息插入到组织表中，另一次数据库操作是将用户 id 和组织 id 插入到用户创建的组织表中。最后，在界面更新用户创建的组织信息。



用户在小程序端删除组织，小程序端向 controller 层传送组织 id 参数，controller 调用 service 层的服务，service 调用 4 次 mapper 层的数据库操作，分别删除组织表，用户参与组织表，用户管理组织表，用户创建组织表中的信息，最后在小程序端进行界面更新。

4.3 Decomposition Description 分解描述

4.3.1 模块“活动”描述

1. Overview 简介

在活动这个模块中，主要涉及对 tbl_activity1 和 tbl_activity2 这两张表的操作，比如插入活动信息，删除活动信息，更新活动数据，查找活动等。

2. Functions 功能列表

模块名称	子模块	概述	处理
活动	插入活动	活动发起者填写活动的说明信息，发起活动	数据库中插入记录
	删除活动	管理员或者发起者可以删除过期的活动。	数据库中删除记录
	更新活动	管理员或者发起者可以更新活动信息，重新发布到地图上。	数据库中更新记录
	查找活动	在地图上可以寻找自己感兴趣的活动，点开标识进入活动详情页。	数据库中查找记录

4.3.2 模块“组织”描述

1. Overview 简介

在组织模块，负责组织信息的增删改查，主要操作的是数据表 tbl_org.

2. Functions 功能列表

模块名称	子模块	概述	处理
组织管理	插入组织	任何用户都可以发起组织，进行活动的统一管理。	插入数据库
	删除组织	组织的管理员或者创建者可以解散没有必要的组织。	删除数据库
	更新组织	组织的管理员或者创建者可以对组织信息进行更新，重新发布	修改数据库
	查找组织	用户可以查找自己感兴趣的组织	查找数据库

4.3.3 模块“用户”描述

1. Overview 简介

用户模块中，主要负责用户的个人信息，以及用户与活动，组织的关系，操作的数据表有 tbl_user，tbl_userEnrollActivity，tbl_userConcernActivity，tbl_userGovernActivity，tbl_userLaunchActivity，tbl_userEnrollOrg，tbl_userGovernOrg，tbl_userCreateOrg。

2. Functions 功能列表

模块名称	子模块	概述	处理
用户	个人信息	个人信息的添加或者更新	数据库插入，更新操作
	用户－活动	用户和活动的关系主要是用户参与活动，关注活动，管理活动，发起活动。这 4 种	数据库的增删改查操作

		关系涉及到增删改查。	
	用户－组织	用户和组织的关系主要是用户参与组织，管理组织，创建组织，这 3 种关系也涉及到增删改查。	数据库的增删改查操作

4.3.4 模块“后台管理”描述

1. Overview 简介

在后台管理中，超级管理员有权限查看发布在平台上的所有活动，组织，以及登录过系统的用户的信息。这个模块比较复杂，需要友好的显示数据库中的所有数据，并且体现他们之间的联系。这个模块又细分问登录子模块，查看所有信息子模块，删除选中子模块，其中查看所有信息子模块是重中之重。后台管理涉及对所有的数据表的操作。

2. Functions 功能列表

模块名称	子模块	概述	处理
后台管理	登录系统	后台管理系统的超级管理员利用账号密码进行登录	进行数据比对，防止非法用户进入后台进行恶意操作。
	查看主要信息	超级管理员查看用户，活动，组织的详细信息	在用户界面可以查看该用户参与，管理，关注和发起的活动；参与，管理和创建的组织。 在活动界面可以查看活动的发起者，管理者，关注者和参与者；隶属的组织

			在组织界面可以查看组织的参与者，管理者，创建者，发起的活动。
	删除选中	超级管理员选择删除的条目，进行删除	删除信息

4.3.5 模块“推荐系统”描述

1. Overview 简介

在推荐系统中，要完成的功能很单一，就是向用户推荐 3 个可能感兴趣的活动，节省用户的时间。在推荐的过程中，会调用文件操作子模块和情感分析子模块。操作的数据库主要是 tbl_userEnrollActivity 中用户对参与的活动所发布的评论 comment 这一数据项。

2. Functions 功能列表

模块名称	子模块	概述	处理
推荐系统	基于 item 的协同过滤	以文件操作和情感分析为基础，进行基于 item 的协同过滤，给用户推荐 3 个可能感兴趣的活动	活动推荐

4.4 Interface Description 接口描述

4.4.1“活动”的接口描述

Name 名称：selectEnrollActivity

Description 说明：根据 activityId 选择一条活动记录

Definition 定义: public Map<String, Object> selectEnrollActivity(int activityId)

Name 名称：insertUserCreateActivity

Description 说明：向活动表中插入一条记录

Definition 定义: public int insertUserCreateActivity(Activity activity)

Name 名称： updateActivity

Description 说明：更新活动记录

Definition 定义: public void updateActivity(Activity activity)

Name 名称: selectActivityListByIdList

Description 说明： 根据 activityIdList 选择活动

Definition 定义: public List<Activity1> selectActivityListByIdList(List<Integer> idList)

Name 名称：deleteActivity

Description 说明：根据 activityId 删除活动

Definition 定义: public void deleteActivity(int activityId)

Name 名称：deleteActivityByIdList

Description 说明：根据 activityIdList 删除活动表中的活动

Definition 定义： public List<Activity1> deleteActivityByIdList(List<Integer> activityIdList)

Name 名称：selectActivityByOrgId

Description 说明：根据 orgId 选择组织创建的活动

Definition 定义: public List<Activity1> selectActivityByOrgId(int orgId)

4.4.2“组织”的接口描述

Name 名称：selectOrgById

Description 说明：根据 orgId 选择组织

Definition 定义: public Organization selectOrgById(int orgId)

Name 名称：updateOrg

Description 说明：更新组织。

Definition 定义: public int updateOrg(Org org)

Name 名称：insertOrg

Description 说明：向组织中插入一条记录

Definition 定义: public int insertOrg(Org org)

Name 名称：deleteOrgById

Description 说明：根据 orgId 删除组织

Definition 定义: public void deleteOrgById(int orgId)

4.4.3“用户”的接口描述

Name 名称：insertUser

Description 说明：根据 code 插入一个用户，返回插入的用户信息

Definition 定义: public User insertUser(String code)

Name 名称：updateUser

Description 说明：更新用户

Definition 定义: public void updateUser(User user)

Name 名称：selectUserById

Description 说明：根据 userIdList 选择一批用户

Definition 定义: public Map<String, Object> selectUserById(List<Integer> idList)

Name 名称：selectActivityByUserId

Description 说明：根据 userId 选择用户参与，关注，管理，创建的活动

Definition 定义: public Map<String, Object> selectActivityByUserId(int userId)

Name 名称：isCreator

Description 说明：根据 orgId 和 userId 判断用户是不是组织的管理者

Definition 定义: public int isCreator(int orgId, int userId)

Name 名称：updateCreator

Description 说明：更新组织的管理者

Definition 定义: public void updateCreator(int orgId, int userId)

Name 名称：selectOrgByCreatorId

Description 说明：根据组织的创建者 userId 选择组织

Definition 定义: public List<Organization> selectOrgByCreatorId(int userId)

Name 名称：updateCreator

Description 说明：更新组织的管理者

Definition 定义: public void updateCreator(int orgId, int userId)

Name 名称：updateConcern

Description 说明：根据 userId 和 activityId 更新用户关注的活动

Definition 定义: public void updateConcern(int userId, int activityId)

Name 名称：selectUserConcernActivity

Description 说明：根据用户 userId 选择用户关注的活动

Definition 定义: public List<Activity1> selectUserConcernActivity(int userId)

Name 名称：deleteUserCreateOrgByOrgId

Description 说明：根据 orgId 删除用户创建的组织

Definition 定义: public void deleteUserCreateOrgByOrgId(int orgId)

Name 名称：selectUserCreateOrgByUserId

Description 说明：根据 userId 选择用户创建的组织

Definition 定义: public List<Organization> selectUserCreateOrgByUserId(int userId)

Name 名称：isOrgCreator

Description 说明：判断用户是否是组织的管理员，如果是，返回 1；否则返回 0

Definition 定义: public int isOrgCreator(int userId, int orgId)

Name 名称：InsertUserEnrollActivity

Description 说明：插入用户参与的活动，注意在 tbl_activity1 中要做相应的修改

Definition 定义: public void InsertUserEnrollActivity(int userId, int activityId)

Name 名称：deleteUserEnrollActivity

Description 说明：删除用户参与的活动，注意在 tbl_activity1 中要做相应的修改

Definition 定义: public void deleteUserEnrollActivity (int userId, int activityId)

Name 名称：selectUserEnrollActivityByUserId

Description 说明：根据 userId 选择用户参与的活动

Definition 定义: public List<Activity1> selectUserEnrollActivityByUserId (int userId)

Name 名称：deleteCommentById

Description 说明：根据 userId 和 activityId 删除评论

Definition 定义: public void deleteCommentById (int userId, int activityId)

Name 名称：insertCommentById

Description 说明：根据 userId 和 activityId 插入评论以及与评论相关的图片

Definition 定义: public void insertCommentById (int userId, int activityId)

Name 名称：selectOrgByUserId

Description 说明：根据 userId 选择用户参与的组织

Definition 定义: public List<Organization> selectOrgByUserId (int orgId)

Name 名称：selectUserEnrollOrgByOrgId

Description 说明：根据 orgId 选择参与该组织的用户

Definition 定义: public List<User> selectUserEnrollOrgByOrgId (int orgId)

Name 名称：deleteUserEnrollOrg

Description 说明：根据 orgId 和 userId 删除用户参与的组织

Definition 定义: public void deleteUserEnrollOrg(int orgId, int userId)

Name 名称：selectUserGovernActivityByUserId

Description 说明：根据 userId 选择用户管理的活动

Definition 定义: public List<Activity1> selectUserGovernActivityByUserId(int userId)

Name 名称：selectUserGovernActivityByActivityId

Description 说明：根据 activityId 选择该活动的管理者

Definition 定义: public List<User> selectUserGovernActivityByActivityId(int activityId)

Name 名称：selectUserGovernOrgByUserId

Description 说明：根据 userId 选择用户关注的组织

Definition 定义: public List<Organization> selectUserGovernOrgByUserId (int userId)

Name 名称：selectUserGovernOrgByOrgId

Description 说明：根据 orgId 选择该组织的管理者

Definition 定义: public List<User> selectUserGovernOrgByOrgId (int orgId)

Name 名称：selectUserLaunchActivityById

Description 说明：根据 userId 选择该用户创建的活动

Definition 定义: public List<Activity1> selectUserLaunchActivityById (int userId)

4.4.4 “后台管理”的接口描述

Name 名称：selectAllOrg

Description 说明：无条件选择所有的组织

Definition 定义: public List<Organization> selectAllOrg()

Name 名称：deleteOrgByIdList

Description 说明：根据组织的 orgIdList 删除组织

Definition 定义: public List<Organization> deleteOrgByIdList(List<Integer> idList)

Name 名称：selectAllOrgByUserId

Description 说明：根据用户 userId 选择用户参与，管理，创建的组织

Definition 定义: public Map<String, Object> selectAllOrgByUserId(int userId)

Name 名称：selectAllUser

Description 说明：无条件选择所有的用户

Definition 定义: public Map<String, Object> selectAllUser()

Name 名称：deleteSelect

Description 说明：删除选中的用户

Definition 定义: public List<User> deleteSelect(List<Integer> idList)

Name 名称：selectUserByOrgId

Description 说明：根据 orgId 选择组织的参与者，创建者和管理者

Definition 定义: public Map<String, Object> selectUserByOrgId (int orgId)

Name 名称：selectAllUserByActivityId

Description 说明：根据 activityId 选择活动的参与者，关注者，管理者和创建者

Definition 定义: public Map<String, Object> selectAllUserByActivityId (int activityId)

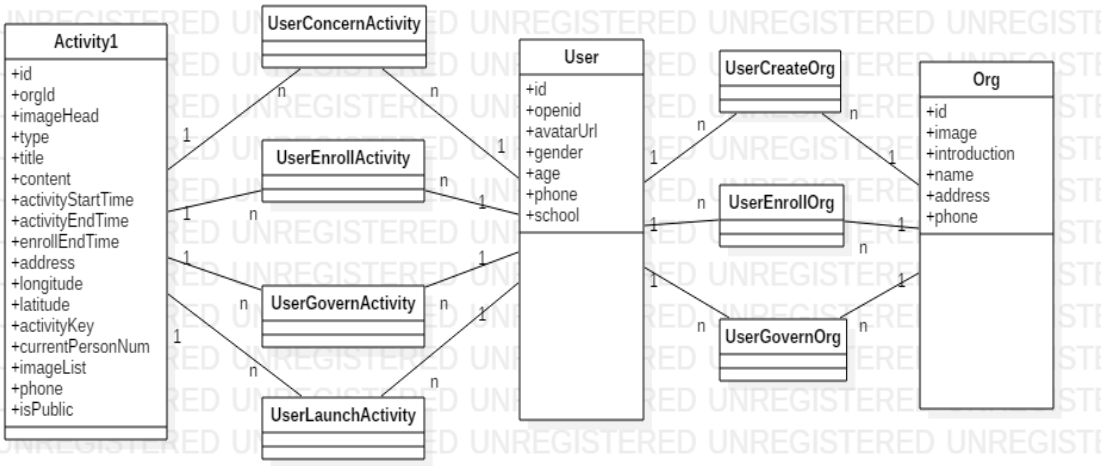
4.4.5 “推荐系统”的接口描述

Name 名称：recommendation

Description 说明：根据 userId 向用户推荐 3 个可能感兴趣的活动。

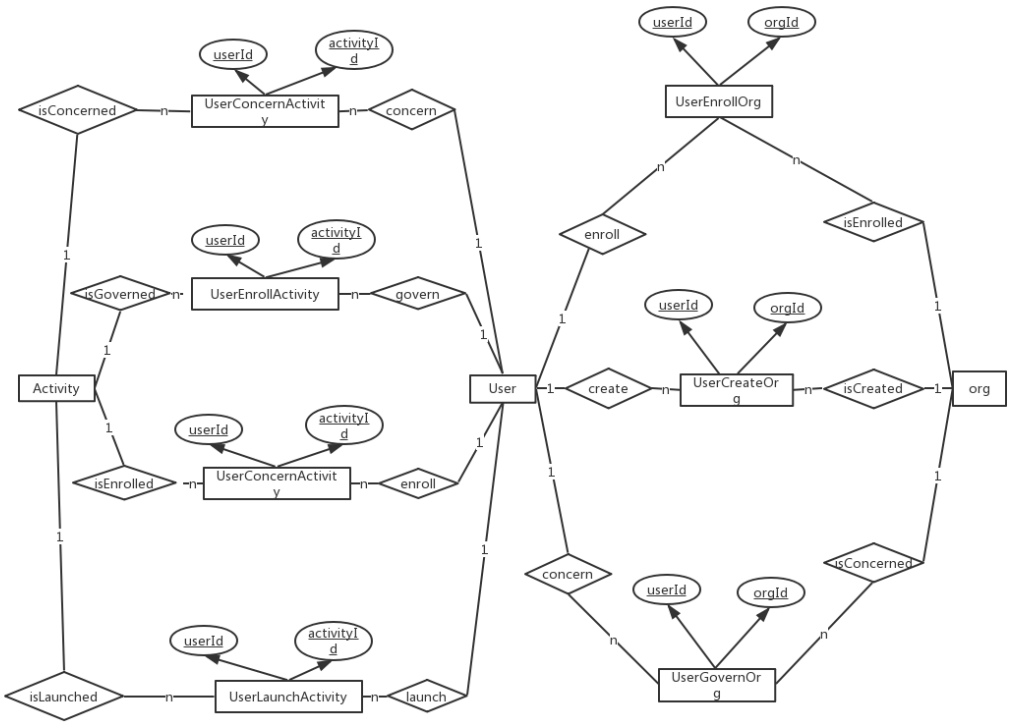
Definition 定义: public List<Activity1> recommend(int userId)

4.5 类之间的关系



4.6 Database Design 数据库设计

4.6.1 逻辑设计



4.6.2 物理设计

活动表 1：tbl_activity1

活动标识 id, 活动所属的组织 orgId，活动介绍的置顶图片 imageHead, 活动类型 type，活动标题 title，发布活动的内容 content，活动开始时间 activityStartTime, 活动结束时间 activityEndTime，报名结束时间 enrollEndTime，活动地址 address， 活动地址的经纬度 longitude，latitude，活动关键字 activityKey,参加该活动的当前人数 currentPersonNum，关于活动的展示图片列表 imageList，活动的联系电话 phone，活动是否公开 isPublic.

字段	类型	长度	默认值	备注
id	int	11	Primary key not null	活动 id，非空，主键，自动增长
orgId	int	11	0	活动所属组织的标识
imageHead	varchar	255	null	介绍活动的置顶图片，指示活动所属的类型
type	varchar	255	null	
title	varchar	255	null	
content	varchar	255	null	
avtivityStartTime	datetime	6	null	MySQL 会实现 String 到 datatime 的自动转换
activityEndTime	datetime	6	null	
enrollEndTime	datetime	6	null	
address	varchar	255	null	
longitude	varchar	255	null	
latitude	varchar	255	null	
activityKey	varchar	255	null	
currentPersonNum	varchar	255	null	
imageList	varchar	255	null	展示活动内容的图片列

				表，与置顶图片不同
phone	varchar	255	null	
isPublic	int	11	0	0 表示不公开，1 表示该活动的发起者希望活动公开

活动表 2：tbl_activity2

活动的唯一标识 id，参与该活动是否需要填写用户的姓名 isName，参与该活动是否需要填写用户的性别 isGender，参与该活动是否需要填写用户的年龄 isAge，参与该活动是否需要填写用户电话 isPhone，参与该活动是否需要填写用户的学校 isCorporation，该活动的报名人数上限 personLimit。

字段	类型	长度	默认值	备注
id	int	11	Primary key not null	活动 id，非空，主键，和 activity1 中的活动主键对应
isName	int	11	0	0 表示不需要填写，1 表示需要填写并且是必填项，2 表示需要填写但是可选项
isGender	int	11	0	0 表示不需要填写，1 表示需要填写并且是必填项，2 表示需要填写但是可选项
isAge	int	11	0	0 表示不需要填写，1 表示需要填写并且是必填项，2 表示需要填写但是可选项

isPhone	int	11	0	0 表示不需要填写，1 表示需要填写并且是必填项，2 表示需要填写但是可选项
isCorporation	int	11	0	0 表示不需要填写，1 表示需要填写并且是必填项，2 表示需要填写但是可选项
personLimit	int	11	0	

用户表：tbl_user

用户的唯一标识 id，微信服务器返回的用户标识 openid，用户的姓名 name，用户的头像 avatarUrl，用户的性别 gender，用户的年龄 age，用户的电话 phone，用户的学校 school。

字段	类型	长度	默认值	备注
id	int	11	Primary key not null	用户 id，非空，主键，自动递增
openid	varchar	255	null	用户从微信服务器中获取的标识
name	varchar	255	null	
avatarUrl	varchar	255	null	
gender	varchar	255	null	F 表示女生，M 表示男生
age	varchar	11	0	
phone	varchar	255	null	
school	varchar	255	null	

组织表：tbl_org

组织的唯一标识 id，组织的置顶图片 image，组织的介绍 introduction，组织的名称 name，组织的地址 address，组织的电话 phone。

字段	类型	长度	默认值	备注
id	int	11	Primary key not null	组织 id，非空，主键，自动递增
image	varchar	255	null	
introduction	varchar	255	null	
name	varchar	255	null	
address	varchar	255	null	
phone	varchar	255	null	

用户参与的活动：tbl_userEnrollActivity

用户标识 userId，表示活动的参与者；活动标识 enrollActivityId，表示一个活动；用户对活动的评论 comment，用户评论活动的工程中发布的图片 commentImage。

字段	类型	长度	默认值	备注
userId	int	11	Primary key not null	用户 id，非空，主键
enrollActivityId	int	11	Primary key not null	活动 id，非空，主键，和 userId 组成联合主键
comment	varchar	255	null	用户对参与的活动的评论，这一部分是进行情感分析的原数据
commentImage	varchar	255	null	用户在发布评论的过程中，有时候为了凸显感情，会发布表达自己情感的图片

用户关注的活动：tbl_userConcernActivity

用户标识 userId，表示活动的关注者；活动标识 concernActityId，表示被关注的活动。

字段	类型	长度	默认值	备注
userId	int	11	Primary key not null	用户 id，非空，主键
concernActityId	int	11	Primary key not null	活动 id，非空，主键，和 userId 组成联合主键

用户管理的活动：tbl_userGovernActivity

用户标识 userId，表示活动的管理者；活动标识 concernActityId，表示被管理的活动。

字段	类型	长度	默认值	备注
userId	int	11	Primary key not null	用户 id，非空，主键
governActivityId	int	11	Primary key not null	活动 id，非空，主键，和 userId 组成联合主键

用户发起的活动：tbl_userLaunchActivity

用户标识 userId，表示活动的发起者；活动标识 launchActivityId，表示被发起的活动。

字段	类型	长度	默认值	备注
userId	int	11	Primary key not null	用户 id，非空，主键
launchActivityId	int	11	Primary key not null	活动 id，非空，主键，和 userId 组成联合主键

用户参与的组织：tbl_userEnrollOrg

用户标识 userId，表示组织的参与者；组织标识 enrollOrgId，表示被参与的组织。

字段	类型	长度	默认值	备注

userId	int	11	Primary key not null	用户 id，非空，主键
enrollOrgId	int	11	Primary key not null	组织 id，非空，主键，和 userId 组成联合主键

用户管理的组织：tbl_userGovernOrg

用户标识 userId，表示组织的管理者；组织标识 governOrgId，表示被管理的组织。

字段	类型	长度	默认值	备注
userId	int	11	Primary key not null	用户 id，非空，主键
governOrgId	int	11	Primary key not null	组织 id，非空，主键，和 userId 组成联合主键

用户创建的组织：tbl_userCreateOrg

用户标识 userId，表示组织的创建者；组织标识 createOrgId，表示被创建的组织。

字段	类型	长度	默认值	备注
userId	int	11	Primary key not null	用户 id，非空，主键
createOrgId	int	11	Primary key not null	组织 id，非空，主键，和 userId 组成联合主键

4.7 本章小结

本章主要介绍了系统的详细设计，包括系统的设计思路，系统每个功能的时序图，每个模块的分解描述，小程序端需要后台服务的每个接口的描述，类之间的关系。最后，介绍了数据库的设计，包括逻辑设计和详细设计。

第 5 章 报名助手小程序服务器端系统实现

上一章介绍了系统的详细设计，本章将介绍如何实现上一章节中各模块的功能。

5.1 服务端框架实现

在 IDEA 中建立一个 maven 项目，需要用到的 jar 包在 pom 文件中导入依赖即可。需要在 applicationContext.xml，Mybatis-config.xml，web.xml，mapper.xml，db.properties，log4j.properties 等文件中进行相应的配置，使得后台服务器的架构满足 SSM 框架。

（1）web.xml 文件的配置

项目启动时，首先会读取 WEB-INF 目录下的 web.xml 文件，因此要在 web.xml 中添加对 Spring 的支持，通过 context-param 配置 Spring 和日志 log4j 的上下文位置，通过 filter 配置启动、编码、shiro 登录拦截等相关过滤器，通过 listener 配置 Spring 监听器，以及使用 servlet 配置 SpringMVC 前端控制器。

（2）spring 文件的配置

通过读取 web.xml 文件，项目加载时会查找 Spring 的具体配置信息，因此，需要在 resources 源目录下新建 ApplicationContext.xml，这是 Spring 的配置文件。 ApplicationContext.xml 文件配置开启组件扫描，一次性扫描指定目录下的所有文件，spring 会把类(类被打上@Component("name")标签)当做组件扫描，配置 mybatis sqlSessionFactory，同时使用 mapper 的动态扫描开发。

（3）mybatis 文件的配置

因为在 ApplicationContext.xml 文件中配置了 MyBatis 文件的位置、数据库连接池、Shiro 安全管理器等相关信息，将业务逻辑层和数据持久层分离并联系起来，所以还需要在 resources 源目录新建 mybatis-config.xml 文件，配置 mapper 的加载方式，将包内的映射器接口实现全部注册为映射器。*Mapper.xml 文件是开发过程中编写 SQL 语句的 XML 文件，通过 mybatis-config.xml 文件的配置在启动时加载。

（4）数据源和日志的配置

最后还需要resources源目录下建立db.properties 和log4j.properties 两个文件，配置连接数据库时的相关参数以及日志信息等。

到这里完成了 SSM 框架的搭建，可以开始正式的编码工作了。

5.2 “活动”模块

根据系统的模块分解图，活动这个模块主要负责活动信息的增删改查，这些子模块将会被用户模块调用。对活动表的插入记录，删除记录都需要配合其他的操作，而修改记录和查找记录可以单独被小程序端调用。下面以查找活动来举例说明后台服务端的实现过程。

用户在地图上点击活动简介，后台的 controller 层拦截请求，找到 URL 的路径为 EnrollAcvt/selectEnrollActivity，控制层 ActivityController 的对应代码为：

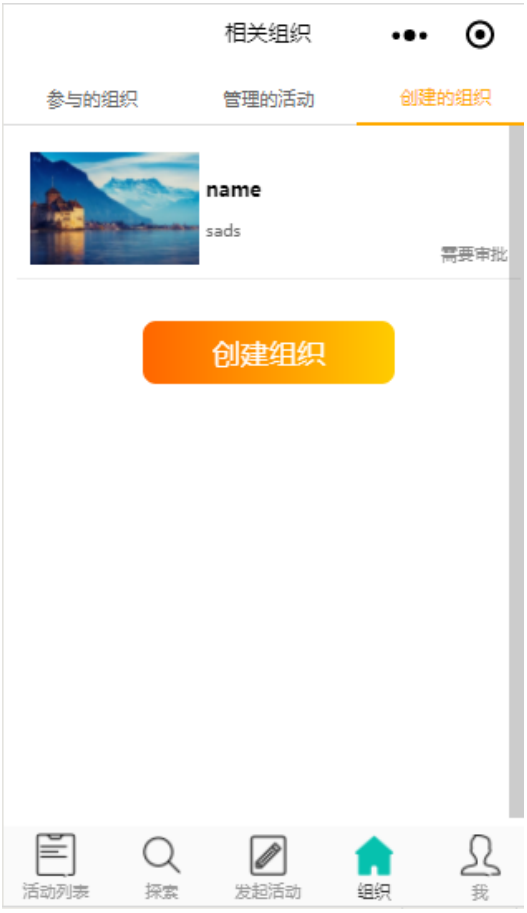
@RequestMapping("/selectEnrollActivity")

```
@ResponseBody
public Map<String, Object> selectEnrollActivity(HttpServletRequest req, HttpServletResponse response) {
    wac= WebApplicationContextUtils.getRequiredWebApplicationContext(req.getServletContext());
    Activity1Service activity1Service = wac.getBean(Activity1Service.class);
    int activityId = Integer.valueOf(req.getParameter("activityId"));
    Map<String, Object> map = activity1Service.selectEnrollActivity(activityId);
    System.out.println(map.toString());
    return map;
}
```

selectEnrollActivity 的实现是调用业务逻辑层 Activity1Service，Activity1Service 调用 mapper 来执行 MyBatis 数据持久层的 SQL 语句，然后返回结果集，传给小程序端，小程序端获取数据后实现界面跳转。

5.3 “组织”模块

组织模块负责组织信息的增删改查，可以提供给用户模块来调用。其中组织信息的修改和查找可以被单独调用，下面以查找组织举例。



用户点击组织简介，就会向后台传送组织 id 这个参数，后台 controller 拦截请求，找到 URL 的路径为 EnrollAcvt/ selectOrgById，控制层 OrgController 的对应代码为：

```
@RequestMapping("/selectOrgById")
@ResponseBody
Public Organization selectOrgById(HttpServletRequest req, HttpServletResponse response) {

    wac = WebApplicationContextUtils.getRequiredWebApplicationContext(req.getServletContext());
```

```
        OrganizationService organizationService =
wac.getBean(OrganizationService.class);

        int orgId = Integer.valueOf(req.getParameter("orgId"));
        return organizationService.selectOrgById(orgId);
    }
```

selectOrgById 的实现是调用业务逻辑层 OrgService，OrgService 调用 mapper 来执行 MyBatis 数据持久层的 SQL 语句，然后返回结果集，传给小程序端，小程序端获取数据后实现界面跳转。

5.4 “用户”模块

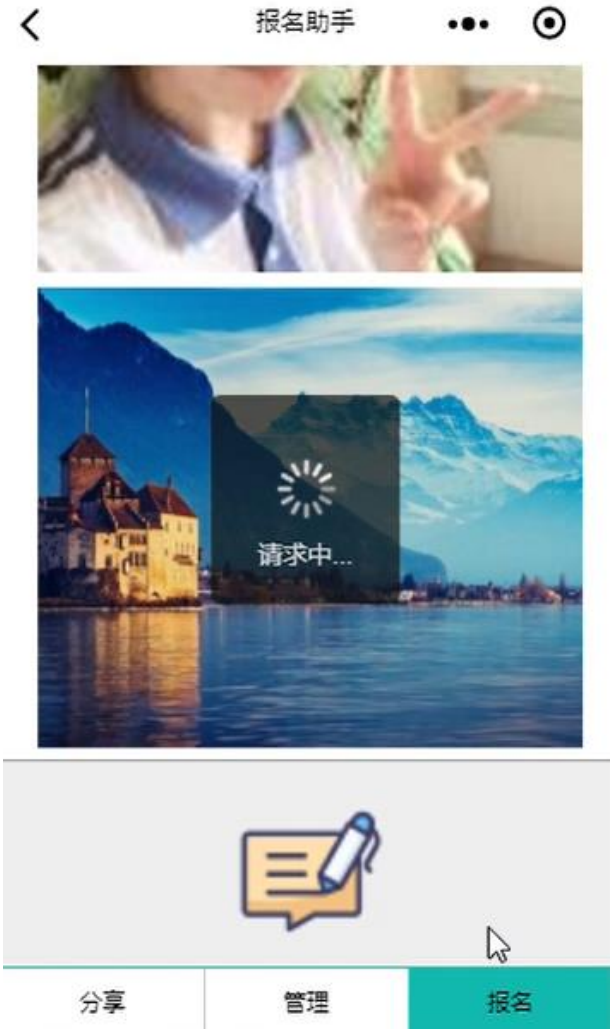
用户模块包括用户信息，用户参与活动，用户关注活动，用户管理活动，用户发起活动，用户参与组织，用户管理组织，用户创建组织。用户信息子模块涉及用户个人信息的增删改查，其他子模块都会调用活动和组织中的子模块。下面以用户参与活动举例，其业务流程如下：

1.点击活动详情页下面的“参与活动”，找到 URL，通过 springMVC 注解找到 UserEnrollActivityController 中的 InsertUserEnrollActivity()方法。

2.UserEnrollActivityController 将获取的参数传递给 UserEnrollActivityservice 层中的相应方法。

3.调用 mapper 接口中的相关方法，找到 UserEnrollActivity.xml 文件，执行 insert 语句，再找到 Activity1.xml 文件，执行 update 语句，更新活动的参与人数。

4.将该活动封装为对象返回给小程序端，小程序端在界面显示。



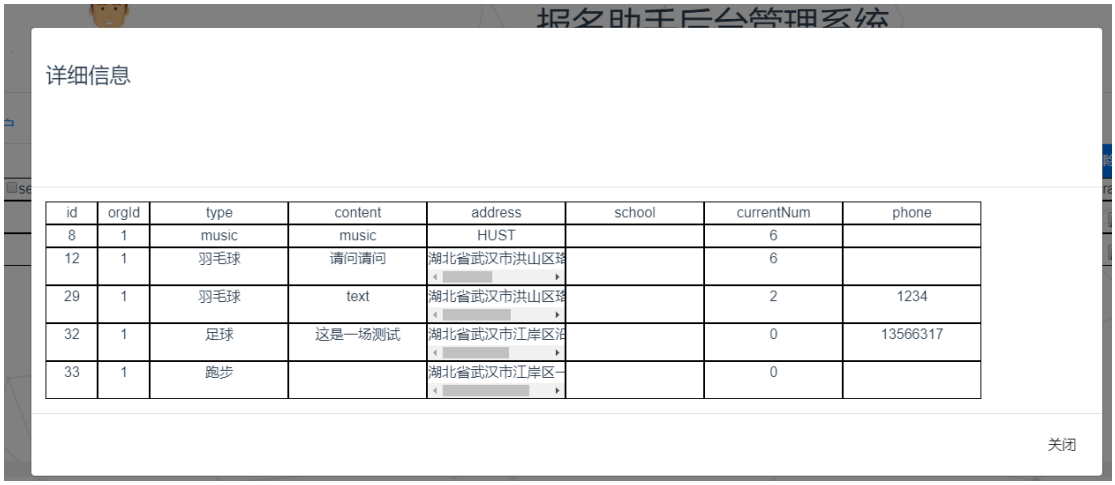
5.5 “后台管理”模块

后台管理主要是展示数据库中的信息，分为登录系统，查看所有信息，删除选中这 3 个子模块。从 web 界面封装数据到后台操作数据，执行流程都是一样的。这里不再赘述，展示几个界面：

登录界面

信息详情页面

查看图片



查看相关信息

5.6 “推荐系统”模块

推荐系统的功能是根据用户的历史喜好向用户推荐 3 个用户可能感兴趣的活动，设计情感分析子模块。推荐系统的处理流程为：

1.用户对参与的活动发起评论时，后台将评论写到数据库后，执行 BosonNLP 的情感分析接口，返回 2 个极性的比例，选取第一个 positive 的比例，再乘以 10 当成用户对活动的评分，写入 csv 文件。

2.用户再次进入系统的探索界面时，后台读取 data.csv 文件，调用 Apache 的 mahout 的基于物品的协同过滤接口，返回 3 个活动 id.

3.根据 idList 读取数据库，获取 3 个活动详情返回到微信小程序端。

下面给出代码：

```
@RequestMapping("/recommendation")
    @ResponseBody
    public List<Activity1> recommend(HttpServletRequest req, HttpServletResponse response) throws Exception{
        wac = WebApplicationContextUtils.getRequiredWebApplicationContext(req.getServletContext());
        Activity1Service activity1Service = wac.getBean(Activity1Service.class);
        int userId = Integer.valueOf(req.getParameter("userId"));
```

```
        ItemBasedCF itemBasedCF = new ItemBasedCF();
        List<Integer> activityIdList = itemBasedCF.itemCF(userId);
        List<Activity1> list = activity1Service.selectActivityListByIdList(activityIdList);
        return list;
    }
}
```

基于物品的协同过滤工具类：

```
    public class ItemBasedCF {

        public final static int RECOMMENDER_NUM = 3;

        public List<Integer> itemCF (int userId) throws Exception{
            RandomUtils.useTestSeed();
            String file = "c://data.csv";


            DataModel model = new FileDataModel(new File(file));
            ItemSimilarity item = new EuclideanDistanceSimilarity(model);

            Recommender r = new GenericItemBasedRecommender(model,item);
            List<RecommendedItem> list = r.recommend(userId,RECOMMENDER_NUM);  //获取推荐结果
            List<Integer> activityIdList = new ArrayList<Integer>();
            for(RecommendedItem item1 : list) {
                activityIdList.add((int)item1.getItemID());
            }

            return activityIdList;
```

```
        }
    }
```

情感分析工具类：

```
    public class SegmentAnalyst {

        public static final String SENTIMENT_URL =
                "http://api.bosonnlp.com/sentiment/analysis";


        public double segmentAnalyst(String comment) throws Exception{

            String[] str =new String[1];
            str[0] = comment;
            String body = new JSONArray(str).toString();
            HttpResponse<JsonNode> jsonResponse =
Unirest.post(SENTIMENT_URL)
                    .header("Accept", "application/json")
                    .header("X-Token", "3KG9-TbW.34284.o5-_DHtSlGdf")
                    .body(body)
                    .asJson();


            JSONArray json =
jsonResponse.getBody().getArray().getJSONArray(0);
            double score = Double.valueOf(json.get(0).toString());
            System.out.println(score);
            Unirest.shutdown();
            return score * 10;
        }
    }
```

第 6 章 总结与展望

6.1 总结

报名助手小程序实现了对活动，组织和用户的管理，该系统保留了现在流行的报名助手小程序的基本功能，添加了敏感词检查和推荐系统，改变了原有的报名助手非法活动和组织泛滥，用户无法快速查找自己感兴趣的活动一系列弊端。该系统的实现为类似基于 SSM 框架的后台系统的设计与实现提供了一个很好的范例，本文的主要工作有：

1. 介绍了 SSM 框架的核心要点和集成原理，DFA 算法实现敏感词检查，基于物品的推荐系统以及情感分析的原理。

2. 按照软件开发的流程对报名助手小程序后台进行了具体的实现。主要分为 4 部分进行了阐述：

1）对报名助手后台系统的需求进行了比较详细的描述，本文从报名助手后台系统的功能需求和性能需求两个方面进行分析，划分了 5 个模块：活动，组织，用户，后台管理，推荐系统。

2）对报名助手后台系统进行了概要设计，首先进行了总体概述，包括软件概述，设计思路，关键技术，系统结构图和数据流向图。接着介绍了系统的开发环境和硬件环境，最后对接口进行了描述。

3）详细设计了报名助手后台系统，对系统的设计思路，业务流程，模块分解描述，接口描述，类之间的关系进行了详尽的描述，最后介绍了数据库的设计，包括逻辑设计和物理设计。

4）对系统的编码实现进行了解释，首先对项目的搭建和配置进行了描述，然后分模块介绍了系统的具体实现，从小程序端到后台服务器，代码的执行流程。

3. 完成编码实现各个模块的功能后，进行测试，检查系统性能。最后完成了论文。

6.2 展望

本文设计和实现的报名助手小程序已经满足所有的需求，但是在小程序的实

现技术和功能上，还是有很多需要继续完善的地方。

1. 代码冗余比较严重，sql 语句书写不简练，导致数据库访问延时比较严重，页面加载出现延迟，需要进一步重构代码。

2. 系统的部署问题，由于一系列问题，系统没有部署到服务器上，所以小程序没有打包上线。

3. 小程序端的界面设计有些硬朗，棱角比较严重。后台管理系统界面设计简陋，有时间对界面元素应该重新设计规划。

4. 设计与实现更多的功能。

参考文献

[1] 曾令祝. MyBatis 用户指南中文版: [EB/OL]. [2010-6-15]. http://wenku.baidu.com/view/77f069160b4e767f5acfcebb.html?from=rec&pos=0&weight=349&lastweight=71&count=5.

[2] S.HORSTMANN C. JAVA 核心技术[M]. 机械工业出版社, 2014.

[3] 吕永会. 基于 Java 的软件保护技术研究[D]. 北京邮电大学, 2014.

[4] 周志明. 深入理解 Java 虚拟机:JVM 高级特性与最佳实践(第 2 版)[M]. 机械工业出版社, 2013.

[5] 许令波. 深入分析 Java Web 技术内幕[M]. 电子工业出版社, 2014.

[6] Fisher M, Partner J, Bogoevici M, et al. Spring Integration in Action[J]. Annals of the Rheumatic Diseases, 2015, 74(Suppl 2):229-229.

[7] 洪植林. 基于 SSM 框架的高校实验室信息管理系统的设计与实现[D].浙江工业大学,2016.

[8] 沃尔斯. Spring 实战[M]. 人民邮电出版社, 2013.

[9] 徐雯, 高建华. 基于 Spring MVC 及 MyBatis 的 Web 应用框架研究[M]. 微型电脑应用. 2012: 1-4

[10] 刘军, 戴金山. 基于 Spring MVC 与 iBATIS 的轻量级 Web 应用研究[J]. 计算机应用, 2006, 26(04): 840-3.

[11] 刘昆. 基于 J2EE 平台的轻量级框架的应用研究[D]. 武汉理工大学, 2008.