**2016年全国高校移动互联网**

**应用开发创新大赛**

**[作品名称]**

**设计文档**

所属组别： 本科组☑ 高职高专组□

所属赛区： 华中赛区

所属学校： 武汉大学

团队名称： 一起嗨皮

指导老师： 李晨亮

队 长： 查道琛

队 员： 龚银超 段誉

目　　录

[摘要 1](#_Toc465500431)

[第一章 概述 2](#_Toc465500432)

[1.1 选题背景 2](#_Toc465500433)

[1.2 设计思想 3](#_Toc465500434)

[1.3 作品特色 3](#_Toc465500435)

[1.4 作品创意与市场前景 4](#_Toc465500436)

[第二章 可行性分析与目标群体 5](#_Toc465500437)

[2.1 可行性分析 5](#_Toc465500438)

[2.1.1 经济可行性 5](#_Toc465500439)

[2.1.2 技术可行性 5](#_Toc465500440)

[2.1.3 人员可行性 6](#_Toc465500441)

[2.2 目标群体 6](#_Toc465500442)

[第三章 作品功能描述与设计 7](#_Toc465500443)

[3.1 功能描述与总体架构设计 7](#_Toc465500444)

[3.1.1 作品功能描述 7](#_Toc465500445)

[3.1.2 作品总体设计图 7](#_Toc465500446)

[3.1.3 后台架构设计 8](#_Toc465500447)

[3.1.4 Android端架构设计 9](#_Toc465500448)

[3.2 核心算法及后台逻辑设计 12](#_Toc465500449)

[3.2.1 核心算法设计 12](#_Toc465500450)

[3.2.2 后台逻辑设计 19](#_Toc465500451)

[第四章 作品实现 33](#_Toc465500452)

[4.1 核心算法实现 33](#_Toc465500453)

[3.2.1 爬虫模块实现 33](#_Toc465500454)

[3.2.2 Lucene索引与检索模块实现 34](#_Toc465500455)

[3.2.1 个性化推荐模块实现 38](#_Toc465500456)

[4.2 后台逻辑实现 39](#_Toc465500457)

[4.2.1 Django逻辑实现步骤 39](#_Toc465500458)

[4.2.2 Spring MVC接口实现 42](#_Toc465500459)

[4.2.3 REST API 44](#_Toc465500460)

[4.3 Android端实现 45](#_Toc465500461)

[第五章 作品展示与测试 46](#_Toc465500462)

[5.1 作品UI展示 46](#_Toc465500463)

[5.2 作品测试 46](#_Toc465500464)

[5.2.1 测试方案 46](#_Toc465500465)

[5.2.2 测试环境 46](#_Toc465500466)

[5.2.3 测试及分析 47](#_Toc465500467)

[第六章 团队介绍及人员分工 51](#_Toc465500468)

[第七章 其它 52](#_Toc465500469)

[7.1 开发结束后的感触 52](#_Toc465500470)

[7.2 不足之处和今后的设想 52](#_Toc465500471)

[7.3 用到的第三方代码或组件 53](#_Toc465500472)

[参考资料 54](#_Toc465500473)

摘要

随着移动互联网的不断发展，基于位置的服务（Location Based Service, LBS）已成为移动互联网的热门方向。LBS在产业界已得到广泛应用，比如我们熟知的美团、大众点评、滴滴打车、百度地图等。另一方面，面向校园的移动互联网产品也层出不穷，比如超级课程表、武大助手、ofo校园单车等。从面向校园的方面入手，我们发现校园类移动互联网产品在基于位置这个特性方面普遍缺失或者不够细粒化，没有充分挖掘移动互联网基于位置的特性。

鉴于此，团队开发了EasyCampus，一个基于位置的个性化校园服务APP。作品的主要特色功能有：讲座模糊搜索、基于位置的讲座个性化推荐、帮拿快递平台。此外，作品还集成导航、地图搜索，定位、附近娱乐、附近ATM机、附近车站、天气预报、附近自习室等一系列方便校园生活的功能。

框架方面，作品后台采用流行的Django框架以及Spring MVC，对象关系映射的使用使得数据库便于操作、维护；Android开发使用最新的MVP框架，模型与视图完全分离，方便维护与测试。

核心功能方面，我们利用爬虫技术爬取了学校相关网站的讲座信息，用正则技术对这些自然文本进行处理，提取关键信息（标题、作者、时间、地点等）。我们将这些爬取的讲座信息以及平台上发布的讲座信息统一建库索引。基于索引，我们实现了基于同义词扩展的讲座模糊搜索功能。此外，我们基于用户自定义标签、搜索历史纪录、用户位置数据等用户使用数据构建用户画像，实现基于位置的个性化推荐功能。团队开发出了帮拿快递平台，允许实名认证用户发布自己的快递信息、获取周边人发布的快递信息，帮助解决“没空拿快递”的普遍问题。

作品的UI设计流畅、美观，在View测试和压力测试下，产品都有较好的表现。

团队开发该APP的宗旨是为大学生提供更加优质、更加细粒化的基于位置的个性化服务，为校园生活、社交互动等提供平台。虽然因人力、时间所限，一些方面做的还是有所欠缺，与能够真正上线的产品还有一些距离，但是团队希望以这个demo为雏形，最终开发出能够真正投入使用的产品。

第一章 概述

1.1 选题背景

近年来，移动互联网产业发展迅猛，其产值已有超过传统互联网之势。一方面是由于移动互联网具有移动性、便携性的特点，用户可以随时随地接入互联网获取信息；另一方面，移动互联网可以为用户提供精准的位置定位，为用户提供多种多样的个性化服务。目前，移动互联网已在包括电子商务、游戏、流媒体在内的各个领域得到广泛应用，《2015中国移动互联发展指数数据报告》数据显示，2015年中国智能终端规模已达12.4亿台，4G用户较前一年增长近100%，移动互联网产业发展潜力巨大[1]。

基于位置的服务（Location Based Service，LBS）在20世纪70年代被提出，近年来随着移动互联网的发展被再次受到重视[2]。在移动互联网时代，通过对用户的精准定位为用户提供各种个性化的服务已成为移动互联网发展的一个趋势。LBS已在工业界得到广泛应用，例如我们熟知的美团、大众点评基于位置进行个性化推荐、嘀嘀打车基于位置联结起乘客与司机、百度地图根据位置进行智能导航等等。在未来，随着人们生活品质的上升，人们对于基于位置的服务的需求也将越来越高。

移动互联网产业的一个重要特点是青年的主力地位。据《2015中国移动互联发展指数数据报告》显示，2015年7月移动网民中25岁及以下占36.6%，26-35岁占37.3%[1]。大学生是移动互联网的重要用户群体，近年来针对大学生的移动互联网产品层出不穷，许多已取得一定的成功。例如，超级课程表、课程格子聚焦于对课表信息的优化，以方便课业生活；以武大助手公众号为代表的一系列校园服务微信公众号为学生提供课程给分评价、成绩计算等服务；最近高调进入校园的ofo校园单车，以分享经济为理念，也得到了一笔不小的融资。基于校园的移动互联网产品的优势在于其针对性，其致力于方便学生的生活，为学生提供咨询、社交等方面的服务。

但是，这些基于校园的移动互联网产品有其局限性，很多需求得不到满足。例如，许多在学校内开展的讲座纵使富有营养，也常常因为信息的不对称而不为人所知，许多优质的讲座常常只有寥寥数人参加。虽然许多校园公众号提供对讲座推广的服务，但是这个过程对于用户来说却是被动的，用户只是被动地接受讲座信息，很难去主动寻找自己感兴趣的讲座。其次，拿快递是大学生活的日常，但是学生常常因为有事而不能及时取得快递，甚至最终出现快递丢失的情况。在有事时难以找到伙伴帮忙取快递是大学生的家常便饭。总的来说，如今面向校园的APP在基于位置的服务方面做得不够细粒化，移动互联网精准定位的优势没有被充分挖掘。

基于以上的不足，我们开发了EasyCampus，一个基于位置的个性化服务校园APP。EasyCampus抓取学校网站发布的各种讲座信息，结合相关人员在平台上发布的其它讲座信息，统一建库索引，为用户提供个性化检索服务。同时EasyCampus基于用户的个性标签、位置信息、搜索记录信息等，构建用户画像，进行个性化讲座推送。一方面，用户可以主动地去寻找自己感兴趣的讲座，打破信息的不对称；另一方面，用户可以及时得到自己感兴趣的讲座推送。EasyCampus提供校内帮取快递平台，经实名认证后的用户可以随时随地发布自己的快递信息，或者帮助他人取快递，解决取快递难的问题。EasyCampus作为一个综合的移动互联网产品，还提供校园资讯、天气预报、社交、导航、车站定位等服务，致力于让校园生活变得更加舒适、方便。

1.2 设计思想

EasyCampus的主要设计思想是基于位置的搜索与个性化推荐，以及一个综合的校园服务平台。

基于位置的搜索与个性化推荐。团队利用网络爬虫技术爬取校园相关网站所发布的讲座信息，利用索引引擎对爬取的讲座进行全文索引。基于该索引，向用户开放模糊检索功能，用户不必再被动地接受学校内的讲座信息，而是可以主动地去搜寻自己感兴趣的讲座。同样基于该索引，团队综合考虑用户地理位置信息、用户搜索历史记录、用户自己定义的个性化标签，构建用户画像，在索引中过滤出用户最可能感兴趣的讲座，并由服务器向用户推送讲座，如此用户可以及时地了解到自己感兴趣的讲座，或者自己附近的讲座。团队充分发挥移动互联网基于位置的特性，结合用户使用数据，致力于提供优质的基于位置的个性化搜索与推荐服务。

一个综合的校园服务平台。学生的需求是多方面的，综合性的服务平台更容易受到学生的青睐。团队通过做需求分析，尽可能地列举出了大学生在学校内的需求，从帮取快递，到社交互动，到天气预报，从复杂的功能到小的模块，我们都尽量在产品中呈现。我们花大量的时间进行UI设计，尽力将我们设想的功能以良好的用户体验展现给用户。

1.3 作品特色

1.3.1 讲座个性化搜索

作品不仅提供发布讲座信息的平台，而且利用爬虫抓取讲座信息，将讲座信息统一索引，向用户提供模糊搜索的接口。用户可以主动地对讲座进行检索，第一时间定位到自己最感兴趣的讲座。

1.3.2 基于位置的个性化推送

作品根据用户的地理位置信息、搜索记录、自定义标签构建用户画像，定时向用户推送用户最有可能感兴趣的讲座。用户在忙碌之余依然可以接受到自己感兴趣的讲座信息。

1.3.3 基于位置的帮取快递平台

作品允许身份认证的用户发布自己的快递领取信息，同时也能搜索到附近同学发布的快递信息及具体的地理位置，通过与发布者进行沟通，帮忙领取快递。该平台可以让忙碌的同学依然可以按时取到快递，也可以帮助同学扩展交际圈、人脉。

1.3.4 综合的校园服务平台

作品功能丰富，除讲座搜索及推送、帮取快递平台、社交互动等核心功能以外，还集成导航、地图搜索，定位、附近娱乐、附近ATM机、附近车站、天气预报、附近自习室等一系列方便校园生活的功能。

1.3.5 流畅、友好的UI设计

作品采用健壮的Volley框架实现网络异步通信，避免阻塞主进程，使得应用稳定流畅；主菜单的栏目采用流行的滑动切换方式，有较好用户体验；作品充分考虑到不同分辨率的安卓终端，自适应不同分辨率的屏幕。

1.4 作品创意与市场前景

作品创意点在于面向校园的基于位置的细粒化服务与个性化推荐。作品中的主要体现是个性化讲座搜索、基于位置的个性化推荐、基于位置帮取快递平台等等。作品充分考虑移动互联网的特性，特别是基于位置的特性，为大学生提供服务。在现有的技术条件下，Google、百度、Apache等技术型公司以及一些开源组织为开发者提供了大量的框架、工具集、API等，作品涉及的定位、爬虫、全文索引、构建用户画像等功能均可以在现有的技术条件下实现。

目前，已有许多基于校园的移动互联网产品取得了成功，基于校园的移动互联网开发具有前景。作品的讲座检索与基于位置的个性化推荐，以及帮拿快递平台，是现有校园APP所缺乏的，作品在校园APP中具备竞争力。作品功能丰富，比现有的校园APP考虑得更加周全，相比而言，具备一定的竞争优势。

第二章 可行性分析与目标群体

2.1 可行性分析

2.1.1 经济可行性

作品开发基于安卓平台，成本较低，开发投入方面主要需要安卓手机、计算机、网络、服务器等，均在团队的承受范围之内。产品开发在经济上是可行的。

2.1.2 技术可行性

产品的关键技术在现有的技术条件下均可实现。

网络爬虫技术可使用Python的BeautifulSoup库结合正则表达式实现。Beautifulsoup是Python第三方库，可用来帮助分析html源码，大大提高分析html源码的效率。

讲座索引及检索可结合全文检索引擎工具包Lucene实现。Lucene是apache软件基金会4 jakarta项目组的一个子项目，是一个开放源代码的全文检索引擎工具包，但它不是一个完整的全文检索引擎，而是一个全文检索引擎的架构，提供了完整的查询引擎和索引引擎，部分文本分析引擎[3]。结合Lucene，团队可以实现索引、检索、个性推荐的功能。

服务器后台及接口方面，作品可采用基于Python的框架Django以及基于Java的Spring MVC框架实现。Django是一个开放源代码的Web应用框架，由Python写成，采用了MVC的框架模式，即模型M，视图V和控制器C[4]。Django采用对象关系映射（object-relational mapping，ORM），可方便的进行数据定义和数据库操作。Spring 框架同样提供了构建 Web 应用程序的全功能 MVC 模块[5]。使用MVC架构和对象关系映射可显著提高后台开发效率，且开发出来的产品结构优美、便于维护。

Android开发方面，作品可采用当今流行的MVP架构，View负责显示，Presenter负责逻辑处理，Model提供数据。模型与视图的完全分离，可让作品易于开发、维护。采用Volley框架可实现网络通信。Volley是在2013年Google I/O大会上推出的新的网络通信框架，它操作简单便捷，其健壮性有利于正确更新UI和获取数据。Volley既可以进行简单的Http通信，又可以轻松加载网络上的图片，在性能上，Volley比较适合去进行数据量不大，但通信频繁的网络操作，非常符合本APP的网络通信特点，而且Volley在不同的线程上异步执行所有请求而避免了阻塞主线程，可使得APP的运行稳定流畅。使用百度公司开发百度地图SDK可轻松地实现定位功能；使用百度云推送的SDK可方便实现服务器向终端的个性化推送。Android的UI设计也已经相当成熟，相关文档也较为齐全。因此作品的构想在Android端是可行的。

综上，团队的设想在现有的技术条件下是可以实现的，作品具有技术可行性。

2.1.3 人员可行性

团队三人各有所长。一人擅长Android开发，一人擅长数据挖掘、文字处理技术，一人擅长后台服务器的搭建，分工明确。团队指导老师的研究方向包含数据挖掘、自然语言处理，可为相关技术的实现提供指导。因此，作品具备人员可行性。

2.2 目标群体

作品主要面向在校大学生，为其提供基于位置的信息推送、信息查询、互动交流等功能。

第三章 作品功能描述与设计

3.1 功能描述与总体架构设计

3.1.1 作品功能描述

作品是基于移动互联网的校园综合服务平台，主要特色有基于位置的个性化搜索与推荐、基于位置的帮取快递平台等，同时提供诸如校园资讯、天气预报、社交、导航、车站定位等服务。

用例图（User Case）是被称为参与者的外部用户所能观察到的系统功能的模型图，呈现了一些参与者和一些用例，以及它们之间的关系，主要用于对系统、子系统或类的功能行为进行建模[6]。团队成员对大学生的用户需求进行分析、归纳，对作品的功能进行建模，用例图如图3-1所示。



1. 作品用例图

3.1.2 作品总体设计图

作品设计分为三大部分，分别为信息抓取及后台逻辑部分、基于Lucene的个性化搜索与推荐部分、Android终端部分。各部分之间采用数据耦合。设计通过REST（Representational State Transfer）风格API（Application Programming Interface）进行数据交换。另外作品调用了一部分第三方API（如百度地图API、百度云推送API等）辅助实现作品的功能。作品的总体设计图如图3-2所示。



1. 作品总体设计图

3.1.3 后台架构设计

后台设计基于MVC架构，即Model（模型）、View（视图）及Controller（控制器）。Model负责数据有关的操作，View负责决定向用户展现什么样的内容（作品中的具体形式是REST风格的接口），Controller负责集中控制，决定用户请求的走向及函数的调用。后台设计架构图如图3-3所示。

为均衡负载，作品将基于Lucene的检索与个性化推荐与其它业务逻辑分开，基于Spring MVC进行设计；其它业务逻辑使用基于Django框架进行设计。两者分别搭载到两个不同的服务器上，之间通过REST风格的接口进行通信。



1. 后台设计框架图

3.1.4 Android端架构设计

Android端基于MVP框架，并结合业务需求的特点做了一些适当的调整。Android端的架构图如图3-4所示。

MVP框架由3部分组成：View负责显示，Presenter负责逻辑处理，Model提供数据。MVP框架有许多优点，可总结如下：

1. 模型与视图完全分离，我们可以修改视图而不影响模型；
2. 可以更高效地使用模型，因为所有的交互都发生在一个地方——Presenter内部；
3. 我们可以将一个Presenter用于多个视图，而不需要改变Presenter的逻辑。这个特性很有用，因为视图的变化总是比模型的变化频繁；
4. 如果我们把逻辑放在Presenter中，那么我们就可以脱离用户接口来测试这些逻辑（单元测试）。



1. Android端的MVP架构图

（1）framework层

framework层主要包括三个功能模块：网络请求（Net）、离线地图（OfflineMap）、地图和定位（Location）。

Net模块是对底层网络框架Volley的一个良好的封装，使上层需要发送网络请求时调用的API更加人性化。

OfflineMap模块是对离线地图的功能的一个封装，关于请求查询是否存在可用的离线地图包、下载的进度、是否有更新等API全部封装在这个模块。

Location模块是包含API最多的一个功能模块，除离线地图外与地图SDK有关的一切功能全部封装在这个模块。关于这个模块，有一些有趣的话题。从大的功能上来分类，Location模块的功能可分为定位与地图查询两大部分。其中定位的功能在Location模块里包含了一个有用的设计。不同于Net模块，对于上层的业务模块在通过Net模块进行网络请求时需要传参相应的回调方法，帮助Net模块在进行网络请求得到结果以后，无论是成功还是失败可以调用相应的回调方法。但是在Location模块的定位功能里，由于任何一个地图SDK都存在一定概率的定位失败的风险，但是本系统高度依赖定位的结果，为了改善用户体验，在Location模块里做了一个类似于缓存的机制：DataCenter。

DataCenter的设计基于观察者模式，这个DataCenter扮演消息源的角色。当定位SDK的定位结果回来以后，只会告诉DataCenter，进行数据的更新。如果数据有效，直接更新位置信息，并通知所有监听这个消息源的观察者。如果定位返回的数据无效，DataCenter会自动请求定位SDK再次定位，连续三次定位返回的结果都是无效的，才认为应用环境有问题，通知上层定位失败。这个设计可以有效的容纳定位sdk出错的概率，当偶尔发生定位失败的时候，上层业务模块无须关心是否需要再次请求定位，只要监听了DataCenter消息源，就安静的等待DataCenter的通知就可以了。

（2）common层

common层又可以称之为代理层，其作用就是对framework层的各个功能模块做了一个统一的代理。对于上层的业务模块，屏蔽了下层framework层的具体模块和具体逻辑。当framework层发生变化时，上层无须关心。此外，common层中的database对数据库的相关操作进行封装，避免了在业务层重复创建数据库对象。

（3）业务模块层

业务层使用的设计就是典型的MVP架构，View和Presenter进行严格的接口隔离，大大地降低了View的代码臃肿情况。同时Presenter作为业务处理的执行者直接与下层打交道，用户通过View发生交互式，View通知Presenter包括发生网络请求，发送定位请求，查询地图热点等繁重的网络任务。在等到结果返回时，Presenter层负责进行数据的转换、过滤、判断等操作，并把相关数据送到View进行显示或操作，使得用户在操作APP时非常流畅。

（4）BaseActivity

这个设计是因为业务本身的需求，从MVP架构中延伸出来的。DataCenter中定位数据发生变化时需要通知上层业务模块进行界面上的改变。但是业务模块中有多个具体的模块，包括nearby，indoor，find，mine，每个模块中可能有多个activity，每个activity中又有多个View和ViewGroup。他们都希望在定位信息发生变化的时候，及时得到通知，并做出相应的变化。

在Android中，每个界面都是一个Activity，Activity有一个rootView，rootView里包含了一个名叫content的ViewGroup，这个content其实就是一个activity所有的View和ViewGroup的根节点。既然如此，只需要将activity在DataCenter中注册，然后activity实例在收到定位信息变化的通知时，只需通过找到这个名为content的根节点，然后遍历整个View树，在树中，每一个ViewGroup(其实就是View树中的非叶子节点)都需要负责将这个定位信息的通知分发下去，每一个View(其实就是View树中的叶子节点)自己处理是否需要定位信息即可。对于ViewGroup，Android开发中常用的布局包括，LinearLayout（线性布局），RelativeLayout（相对布局），FrameLayout（帧布局）等。在这个APP中，只用到了LinearLayout和RelativeLayout，另写两个类SSLinearLayout和SSRelativeLayout分别继承LinearLayout和RelativeLayout，这样布局本身的功能不会变，只需给这两个新的布局、Activity以及需要依赖定位信息的View实现一个共同的接口就可以完成定位信息通知的分发。为了将代码更好的抽象，抽取了一个BaseActivity，从根节点分发定位信息的逻辑放在这里，每个Activity都去继承这个BaseActivity，如此开发者就不用再关心定位信息如何传递给具体的Activity，View或ViewGroup。

3.2 核心算法及后台逻辑设计

本节首先介绍几个核心算法模块的设计，即爬虫模块、Lucene索引与检索模块、个性化推荐相关模块，然后逐一介绍后台逻辑的设计。

3.2.1 核心算法设计

3.2.1.1 爬虫模块

（1）相关技术及原理介绍

编程语言为Python,主要运用requests，BeautifulSoup，re这些Python库来完成爬虫任务，每个库的作用如下：

requests库用来根据URL地址来获取html信息；

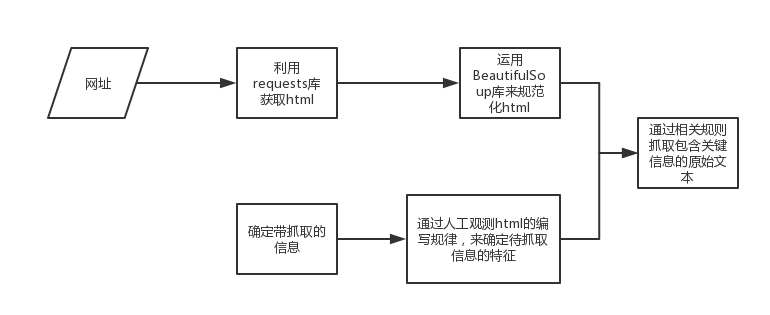
Beautifulsoup库用来规则化html，让我们能够以一定的规则提取html中特定位置的信息；

re库用来运用正则表达式来从不规则的文本中提取信息，例如通过正则表达式，可以从2016年10月28号上午9：00这样的文本中提取出年、月、日、时、分的信息。

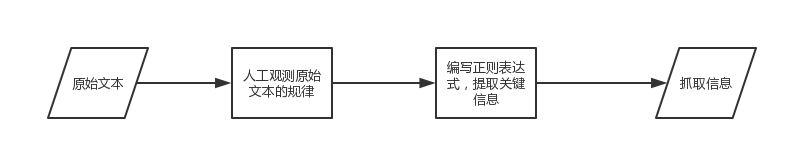
（2）爬虫基本框架

1. 抓取原始html数据。通过requests库来根据url获取html数据；
2. 解析html数据，提取有效文本。BeautifulSoup可用来规则化html，通过观察网页的组成方式，我们可以总结出一部分规律，即确定哪一部分的信息以何种形式出现。通过运用BeautifulSoup中的相关技术来找出包含关键信息的句子，但是此时得到的文本是自然语言形式的，不能够直接存入数据库中使用；
3. 运用正则进行自然语言处理。通过人工总结规律，使用re库提供的正则表达式来从自然文本中提取出规范的信息。

相关流程见如图3-5、图3-6所示。



1. 爬虫流程图



1. 自然文本处理流程图

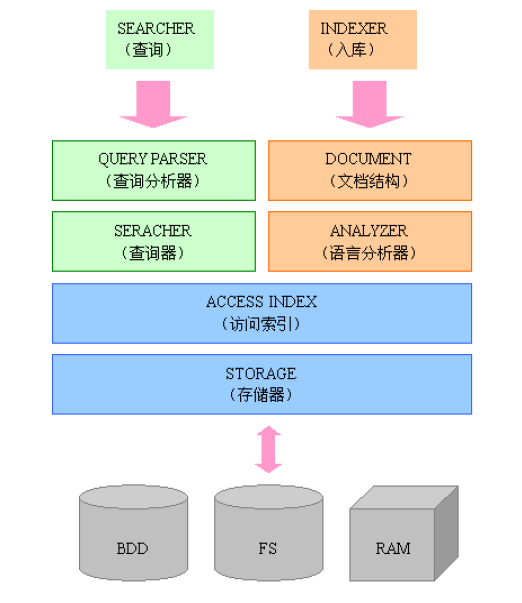
（3）算法流程

1. 选定爬虫入口。通常每个学校的每个学院都会有一个专门的讲座主页，这些主页中间就包含众多的讲座地址，一般我们选取第一个讲座的URL为当前URL，作为爬虫的入口。
2. 爬取讲座。应用爬虫基本框架抓取信息；
3. 验证获取信息的完整性。若完整则存入数据库，否则丢弃；
4. 顺序遍历。在讲座主页寻找下一入口点。若寻到下一入口点，以下一入口点为当前URL，转(b)，否则，算法结束。

3.2.1.2 Lucene索引及检索模块

（1）相关技术介绍

Lucene是一个优秀的开源全文检索工具包，能为我们提供包括分词、索引、搜索在内的功能，其主要逻辑图3-7所示。



1. Lucene技术逻辑图

（2）Lucene核心部分介绍

分词器：我们运用了CJKAnalyzer分词器，该分词器原理简单，对于中文文本abc，该分词会将之分成ab，bc两个词组。虽然原理简单，但是效果可观；

QueryParser：我们运用了MultifieldQueryPaser，该技术能够保证搜索的时候能同时搜索文档的多个域，即除了讲座的题目以外，也能搜索作者信息等；

BoostingQuery：该功能用来设置query的权重，这样保证在一次索引中，更重要的query能够在最后的得分统计有更大的影响力；

BooleanQuery：该技术用来合并两个query。

（3）同义词扩展功能介绍

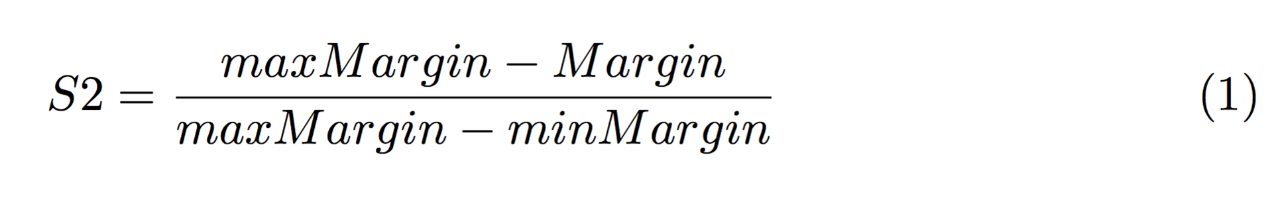
考虑到中文中常出现同义词的情况，而这种情况往往会使得搜索的召回率不高，我们引入同义词扩充功能。我们建立同义词表，表中每一行为同义词序列，称之为TYC，其数据结构为二维数组，TYC[0]表示第0行同义词序列。TYC\_Size表示TYC的行数。S为一个set集合，不允许重复，用来记录原始query包含哪些同义词。伪代码如算法1所示。

|  |  |
| --- | --- |
| **算法1 同义词扩展算法** | |
| **输入：** 原始关键字original\_query, 同义词表TYC, 同义词表行数TYC\_Size | |
| **输出：** 关键字扩展关键字expanded\_query | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | S  line = 0  while line < TYC\_Size do  for word in TYC[line] do  if word in original\_query then  S.add(line)  else  continue  End if  End for  line <-line+1  end while  for \_line in S  for word in TYC[\_line]  if word not in original\_query then  expanded\_query += word  else  continue  end if  end for  end for |

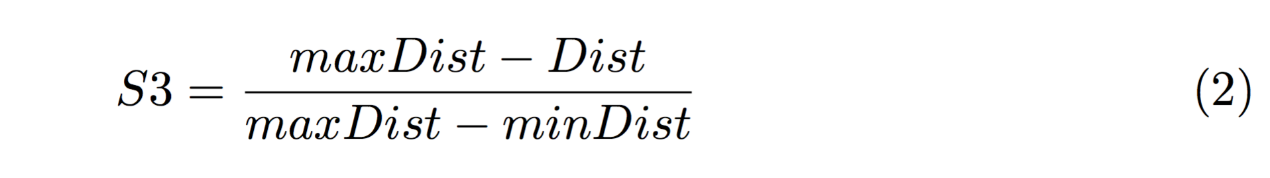
通过以上算法得到expanded\_query之后，将原始query(权重较高)和扩展query通过BooleanQuery结合为新的query，再带入Lucene中进行索引，即起到支持模糊搜索的目的。

（4）扩展得分功能介绍

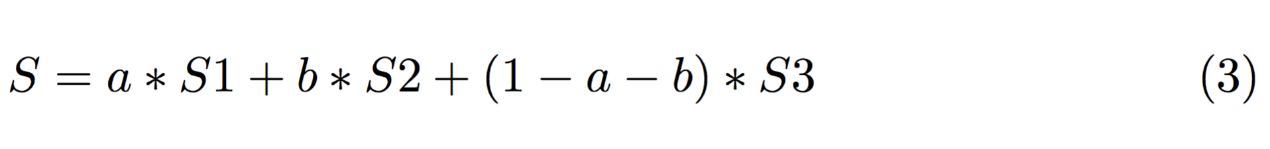
考虑到讲座时间、讲座地点、用户地点多维度的用户数据，我们改写Lucene所带的排序机制，使其支持多维度的扩展得分计算。具体来说，我们将时间和位置两个维度考虑在计算得分机制之内。在介绍扩展得分的算法之前，首先对几个公式说明。



公式（1）中，设一个列表list\_t，该列表每一个元素表示被索引到、还未到时间的讲座的时间与当前时间（用户搜索时的时间）的差值（ms为单位）。maxMargin为list\_t中最大的一个值，minMargin为list\_t中最小的一个值，Margin为每个讲座的值，S2表示某一讲座的一项得分。可以看到该公式会对更靠近当前时间的讲座更大的S2值。



公式（2），设一个列表list\_d，该列表每一个元素表示被索引到、还未到时间的讲座的位置（以经纬度表示）与当前用户位置经纬度的平方差。同S2中一样，maxDist为list\_d中最大的一个值，minDist为list\_d中最小的一个值，Dist为待评分的讲座离用户的距离S2表示某一讲座的一项得分。。可以看到该公式同样会对更靠近当前位置的讲座更大的S3值



公式（3），该公式采用线性组合的方式，综合三种得分方式的权重来得到最终得分，其中S1代表文本程度上的匹配得分，S2、S3分别由公式（2）、公式（3）计算得出。a，b的取值可以通过调节参数来优化，目前取0.6，0.2。

扩展得分计算方法如下：

a. 首先得到原始的Lucene得分，我们记为S1，S1仅仅是文本程度上的匹配得分；

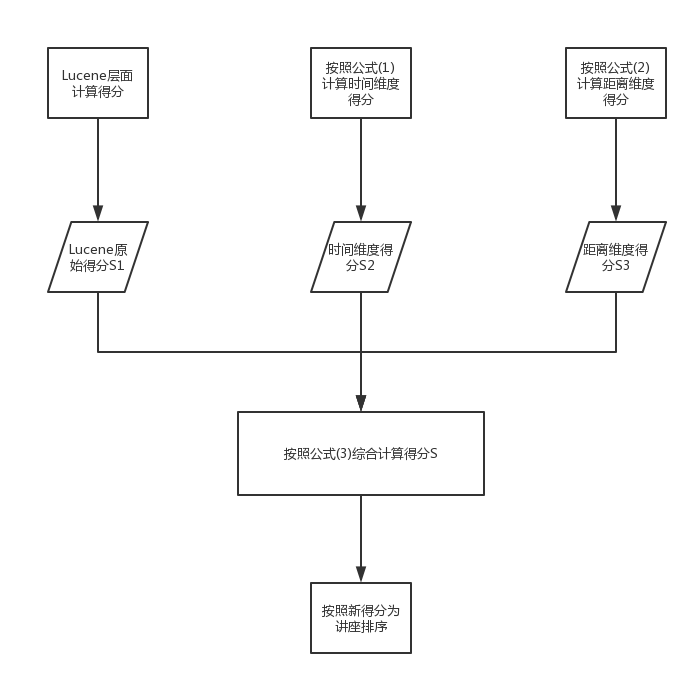
b. 时间越靠近今天的得分应越高，我们用公式（1）来计算其得分。记为S2；

c. 位置越靠近用户的讲座得分应越高，我们用公式（2）来表示其得分，记为S3;

d. 综合考虑三个得分，我们用公式（3）来表示最终的得分；

e. 对讲座重新进行排序。

该部分流程图如图3-8所示。



1. 扩展得分流程图

3.2.1.3 基于用户画像数据的个性化推荐模块

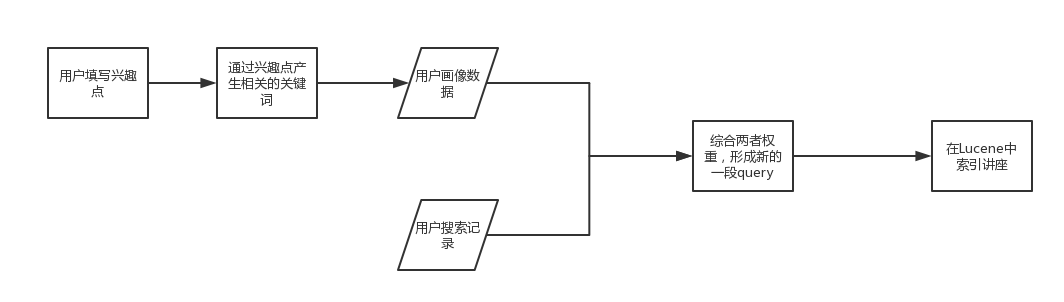
（1）相关原理介绍

为了实现个性化推荐，首先我们需要充分了解用户，我们通过在注册阶段让用户选取兴趣点这样的方式，获取到一部分关键字，我们称之为用户画像数据（例如用户表示对“数据挖掘”感兴趣，那么其可能也对“大数据”、“神经网络”相关的讲座感兴趣），同时存储用户的搜索记录，因为用户的搜索记录也能表达用户对于某种类别的讲座的倾向性。

这样我们就有了与某个用户相关的部分文本数据，这些数据从某种程度上蕴含着用户的倾向性，这些也是我们用来推荐的关键信息。

（2）算法介绍

我们分别将用户画像数据以及搜索记录数据视为两个query，通过赋予不同的权重，将两个query合并之后带入到Lucene的索引框架中，将索引到的讲座信息作为被推荐的讲座。流程如图3-9所示。



1. 扩展得分流程图

3.2.1.4 基于位置与关键字的精选讲座推荐模块

（1）相关原理介绍

上节所介绍的基于用户画像数据的个性化推荐是以用户为中心，筛选相关讲座。但是，实际中经常会出现后台需要将优质讲座推荐给用户群体的需求，即相对于上一节“逆向的”推荐过程。本节从另一个角度进行个性化推荐，即选定优质讲座，筛选相关用户组。

首先，本节介绍两个概念——组播和基于标签推送。

组播：即成组播报，将一条信息同时推送给两个或两个以上的移动设备；

基于标签推送：为每个用户打上标签，被打上相同标签的用户属于同一个以标签命名的用户组。相同用户可被打上多个不同标签，即一个用户可以属于多个不同的用户组。基于标签推送，即对以某个标签命名的用户组进行组播。

基于位置的精选讲座推荐，即依据用户历史活动记录（坐标）相对于讲座地理位置的远近进行分类，将讲座优先推荐给经常在讲座位置附近活动的用户，对这些用户进行组播。

基于关键字的精选讲座推荐，即从讲座的标题中抽取关键字，反向映射到蕴含关键字的标签，进行基于标签推送。

（2）算法介绍

对于后台人员选定的精选讲座，我们首先获取讲座的地理位置字段、标题字段；之后我们基于位置或者关键字对于数据库存取的记录进行比对，最后返回用户组或标签名并进行组播或基于标签推送。流程如图3-10所示。



1. 精选讲座推荐流程图

3.2.2 后台逻辑设计

3.2.2.1 用户模块

（1）数据库设计

用户模块设计三个数据模型，即用户权限模型user\_power，用户在线状况记录模型user\_online，用户资料模型user\_profile，按照对象关系映射的设计思想，这三个模型均为主用户模型（user模型）的扩展模型，呈现“一对一”的关系。

用户权限模型user\_power各字段定义如表3-1所示。

1. 用户权限模型各字段定义

|  |  |
| --- | --- |
| 字段名 | 说明 |
| user | 与user模型呈现“一对一”关系 |
| publisher | 布尔型，标志社团、组织用户权限 |
| express | 布尔型，标志快递相关权限（实名认证用户） |

用户在线状况记录模型user\_online各字段定义如表3-2所示。

1. 用户在线状况记录模型各字段定义

|  |  |
| --- | --- |
| 字段名 | 说明 |
| user | 与user模型呈现“一对一”关系 |
| channel\_id | 字符型，标志在线用户设备号，不在线时为空 |

用户资料模型user\_profile各字段定义如表3-3所示。

1. 用户资料模型各字段定义

|  |  |
| --- | --- |
| 字段名 | 说明 |
| user | 与user模型呈现“一对一”关系 |
| user\_identity | 字符型，记录用户身份号 |
| nick\_name | 字符型，用户昵称 |
| gender | 字符型，性别 |
| head\_icon | Image类型，用户头像 |

（2）逻辑设计

用户模块实现注册、登陆、注销、修改密码、上传与获取个人资料、用户精确搜索、获取用户权限的功能。

1. 注册、登陆、注销、修改密码功能设计

注册时，密码存储以pbkdf\_sha256方式加密，只可单向校验密码，以保障用户密码的安全；登陆采用Token认证方式，即每次登陆时，根据用户的个人信息与session信息计算相应的Token，Token失效期为7天，在7天之内用户可不必再次输入密码，用户登录时上传用户设备ID号，标识用户在线状态；注销时，清除服务器端保存的设备ID号；修改秘密时，首先根据pbkdf\_sha256校验用户输入的原密码，原密码正确时，用pbkdf\_sha256算法将新密码加密后存入数据库中。该部分的程序流程图如图3-11所示。



1. 注册、登陆、注销、修改密码程序流程图
2. 上传与获取个人资料、用户精确搜索、获取用户权限功能设计

首先校验Token与用户设备ID号，两者均正确时功能才能被执行。上传用户资料时，通过接口上传用户基本信息和图片，后台保存图片并且将用户基本信息和图片路径保存至数据库中。获取用户资料时，查询数据库并返回用户基本信息，若为图片，则返回图片的数据流。用户精确搜索部分，输入用户名，在数据库中精确查询用户名字段，返回查询结果。获取用户权限部分，以用户名为query在数据库中查询用户权限，返回查询结果。以上功能的程序流程图3-12所示。（校验Token与用户设备号部分在程序流程图中省略）



1. 上传与获取个人资料、用户精确搜索、获取用户权限程序流程图

3.2.2.2 好友模块

好友模块实现好友的添加、删除等好友管理逻辑。

（1）数据库设计

好友模块设计两个数据模型，即好友关系模型user\_friend、好友请求记录模型friend\_request。前者用来记录好友关系，后者用来记录请求添加好友、回应好友请求的记录。

好友关系模型user\_friend各字段定义如表3-4所示。

1. 好友关系模型各字段定义

|  |  |
| --- | --- |
| 字段名 | 说明 |
| user | 与user模型呈现“一对一”关系 |
| friends | 与user模型呈现“多对多”关系 |

好友请求记录模型user\_request各字段定义如表3-4所示。

1. 好友请求记录模型各字段定义

|  |  |
| --- | --- |
| 字段名 | 说明 |
| user | 字符型，发起请求用户 |
| request\_friends | 字符型，接收请求用户 |
| valid | 布尔型，请求是否得到响应 |
| request\_time | 时间类型，请求时间 |

（2）逻辑设计

好友模块实现发起好友请求、回应好友请求、获取历史好友请求、获取好友列表、删除好友功能。

1. 发起好友请求、回应好友请求、获取历史好友请求功能设计

首先校验Token与用户设备ID号，两者均正确时功能才能被执行。发起好友请求时，输入目标用户用户名，后台向好友请求模型中增添请求记录。回应好友请求时，首先修改请求记录中的valid字段，若同意好友请求，则在好友关系模型中增加新的好友对。获取历史好友请求部分，根据当前用户名，返回好友请求记录模型中与当前用户有关的记录。以上功能的程序流程图3-13所示。（校验Token与用户设备号部分在程序流程图中省略）



1. 发起好友请求、回应好友请求、获取历史好友请求程序流程图
2. 获取好友列表、删除好友功能设计

首先校验Token与用户设备ID号，两者均正确时功能才能被执行。获取好友列表部分，输入用户名，后台查询好友关系模型，返回好友列表。删除好友部分，输入要删除的用户名，后台在好友关系模型中删除相应的好友对。以上功能的程序流程图3-14所示。（校验Token与用户设备号部分在程序流程图中省略）



1. 获取好友列表、删除好友功能设计程序流程图

3.2.2.3 快递模块

快递模块实现快递的发布、删除、查询等功能。

（1）数据库设计

快递模块设计两个数据模型，发布快递模型published\_express、发布者模型express\_publisher，前者用来记录发布的快递信息，后者用来记录快递发布者信息。

发布快递模型published\_express各字段定义如表3-6所示。

1. 发布快递模型各字段定义

|  |  |
| --- | --- |
| 字段名 | 说明 |
| express\_id | 字符型，快递ID号 |
| published\_time | 时间类型，发布时间 |

发布者模型express\_publisher各字段定义如表3-7所示。

1. 快递发布者模型各字段定义

|  |  |
| --- | --- |
| 字段名 | 说明 |
| user | 与user模型呈现“一对一”关系 |
| published\_expresses | 与published\_express模型呈现“多对多”关系 |

（2）百度地图API数据表设计

为方便基于位置的快递检索，我们将快递信息同时存放至百度地图API提供的数据中心中。我们设计了数据表express，该表的字段如图3-15所示



1. 百度地图API数据中心express表字段设计

（3）逻辑设计

快递模块实现发布快递、删除快递、获取发布的快递、获取好友快递信息的功能。

首先校验Token与用户设备ID号，两者均正确时功能才能被执行。之后校验是否为认证用户，只有认证用户才能使用快递功能。发布快递部分，输入快递的经纬度、地点名、时间，后台将快递存入发布快递模型published\_express、发布者模型express\_publisher，并且将地点信息存入百度地图API数据中心express表中。删除快递部分，输入快递ID号，后台删除发布快递模型published\_express中相应数据，同时删除百度地图API数据中心express表中的相应数据。获取发布的快递部分，输入用户名，查询快递模型published\_express，返回该用户发布的所有快递ID号。获取好友快递信息部分，输入用户名，查询发布快递模型published\_express，返回该用户所有好友发布的所有快递ID号。以上功能的程序流程图3-16所示。（校验Token、校验用户设备号、校验认证用户部分在程序流程图中省略）



1. 发布快递、删除快递、获取发布的快递、获取好友快递信息程序流程图

3.2.2.4 聊天模块

聊天模块实现聊天记录的上传与下载功能（后台主要实现保存聊天记录，点对点的聊天采用第三方接口配合Android端实现）。

（1）数据库设计

聊天模块设计一个数据模型，即聊天记录模型chat\_log，用来记录用户的聊天记录，其各字段定义如表2-8所示。

1. 聊天记录模型各字段定义

|  |  |
| --- | --- |
| 字段名 | 说明 |
| sender | 字符型，聊天发起者 |
| receiver | 字符型，聊天接收者 |
| content | 字符型，聊天内容 |
| send\_time | 时间类型，发送时间 |

（2）逻辑设计

聊天模块实现上传、下载聊天记录的功能。

首先校验Token与用户设备ID号，两者均正确时功能才能被执行。聊天记录上传，输入发起用户、目标用户、内容，后台将信息存入数据库。聊天记录下载，输入发起用户、目标用户、初时间、末时间，后台在数据库中查询并返回指定时间范围内的记录。以上功能的程序流程图3-17所示。（校验Token与用户设备号部分在程序流程图中省略）



1. 上传、下载聊天记录程序流程图

3.2.2.5 讲座模块

讲座模块实现快递的发布、删除、查询等功能。

（1）数据库设计

讲座模块设计两个数据模型，发布讲座模型published\_lecture、发布者模型publisher，前者用来记录发布的讲座信息，后者用来记录讲座发布者信息。

发布讲座模型published\_lecture各字段定义如表3-9所示。

1. 发布讲座模型各字段定义

|  |  |
| --- | --- |
| 字段名 | 说明 |
| lecture\_id | 字符型，讲座ID号 |

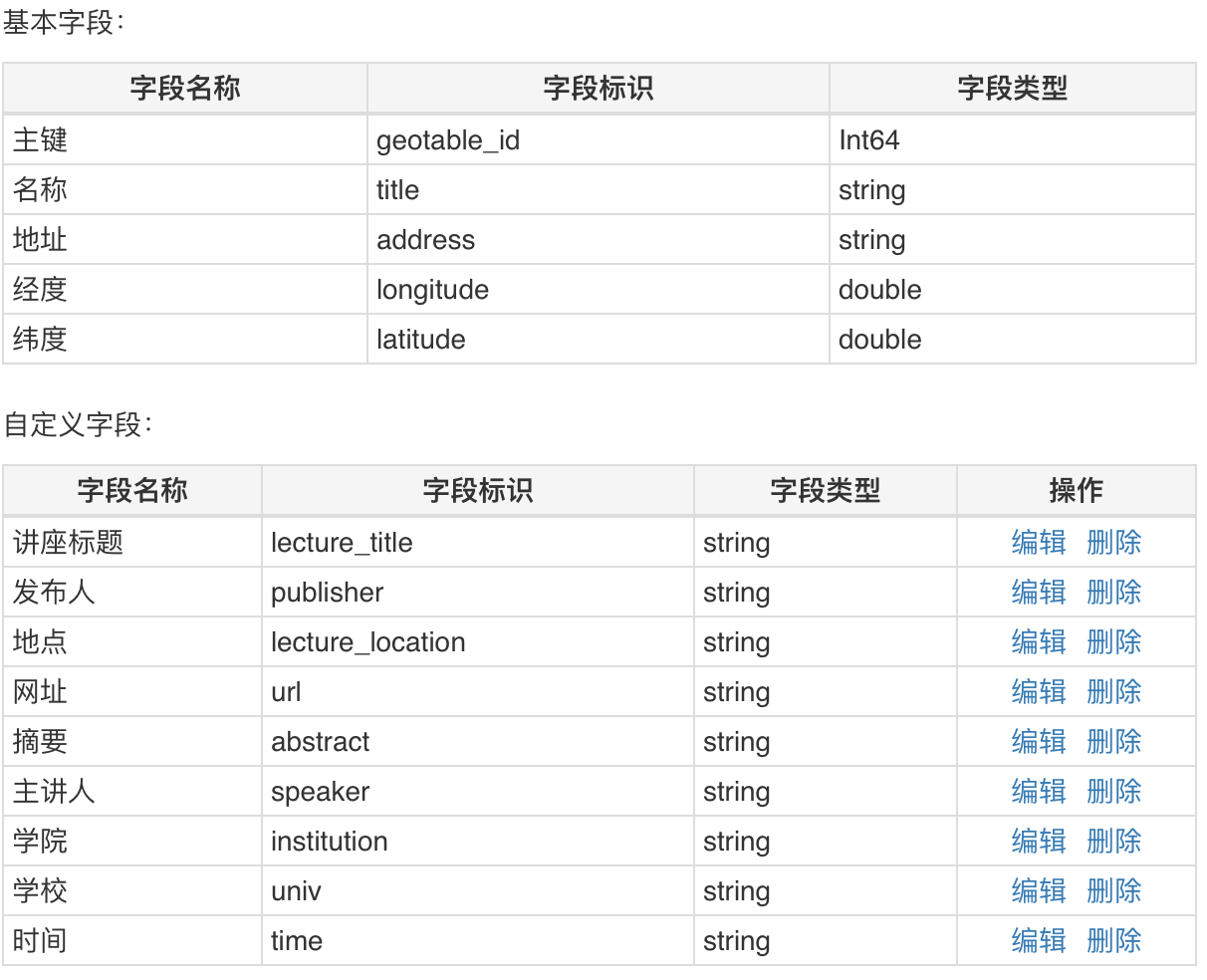
发布者模型publisher各字段定义如表3-7所示。

1. 讲座发布者模型各字段定义

|  |  |
| --- | --- |
| 字段名 | 说明 |
| user | 与user模型呈现“一对一”关系 |
| published\_expresses | 与published\_lecture模型呈现“多对多”关系 |

（2）百度地图API数据表设计

为方便基于位置的讲座检索，我们将讲座信息同时存放至百度地图API提供的数据中心中。我们设计了数据表lecture，该表的字段如图3-18所示



1. 百度地图API数据中心lecture表字段设计

（3）逻辑设计

讲座模块实现发布讲座、删除讲座、获取发布的讲座的功能。

首先校验Token与用户设备ID号，两者均正确时功能才能被执行。之后校验是否为社团或组织用户，只有经确认的社团或者组织用户才能发布讲座。发布讲座部分，输入讲座的经纬度、地点名、时间等基本信息，后台将讲座信息存入发布讲座模型published\_lecture、发布者模型publisher，并且将地点信息存入百度地图API数据中心lecture表中。删除讲座部分，输入讲座ID号，后台删除发布讲座模型published\_lecture中相应数据，同时删除百度地图API数据中心lecture表中的相应数据。获取发布的讲座部分，输入用户名，查询发布讲座模型published\_lecture，返回该用户发布的所有讲座ID号。以上功能的程序流程图3-19所示。（校验Token、校验用户设备号、校验社团或组织用户部分在程序流程图中省略）



1. 发布讲座、删除讲座、获取发布的讲座程序流程图

3.2.2.6 好友动态模块

好友动态模块主要实现动态的发布、删除、获取、点赞等功能。

（1）数据库设计

好友动态设计两个数据模型，即动态模型moment、动态发布者模型moment\_publisher。前者用来存储好友动态，后者用来存储好友动态的发布者。

发布动态模型moment各字段定义如表3-11所示。

1. 发布动态模型各字段定义

|  |  |
| --- | --- |
| 字段名 | 说明 |
| moment\_content | 字符型，动态文字部分 |
| published\_time | 时间类型，动态发布时间 |
| thumb\_up | 点赞，与user模型呈现“多对多”关系 |
| picture | Image类型，动态图片 |

动态发布者模型moment\_publisher各字段定义如表3-12所示。

1. 动态发布者模型各字段定义

|  |  |
| --- | --- |
| 字段名 | 说明 |
| user | 与user模型呈现“一对一”关系 |
| published\_moments | 与published\_moment模型呈现“多对多”关系 |

（2）逻辑设计

好友动态模块实现发布好友动态、删除好友动态、获取好友发布的动态、点赞的功能。

首先校验Token与用户设备ID号，两者均正确时功能才能被执行。发布好友动态，输入用户名、动态内容、图片，后台在数据库中增加相应动态。删除好友动态，输入用户名和动态ID（模型自动生成ID号），后台在数据库中删除相应记录。获取好友发布的动态，输入用户名，后台首先获取其所有的好友，然后在数据库中查询并返回好友发布的所有动态。点赞，输入用户名、动态ID、操作（赞/取消），数据库在动态模型中查找相应动态，若操作为“赞”则增加thumb\_up字段相应多对多关系；若为“取消”则删除thumb\_up字段相应多对多关系。以上功能的程序流程图3-20所示。（校验Token与用户设备号部分在程序流程图中省略）



1. 发布好友动态、删除好友动态、获取好友发布的动态、点赞程序流程图

3.2.2.7 数据模块

数据模块用来搜集用户使用数据，设计搜集三类数据，用户位置数据、用户搜索历史数据、用户自定义标签数据。

（1）数据库设计

数据模块设计四个数据模型。用户位置模型user\_place用来存放用户的位置信息；标签模型tag，用来存放标签名及其对应的关键字；用户个性模型user\_personality，用来存放用户自定义的个性化标签信息；搜索历史模型search\_history用来存放用户的搜索记录。

用户位置模型user\_place各字段定义如表3-13所示。

1. 用户位置模型各字段定义

|  |  |
| --- | --- |
| 字段名 | 说明 |
| username | 字符型，用户名 |
| lat | 字符型，维度 |
| lng | 字符型，经度 |
| time | 时间类型，位置信息的时间 |

标签模型tag各字段定义如表3-14所示。

1. 标签模型各字段定义

|  |  |
| --- | --- |
| 字段名 | 说明 |
| tag\_name | 字符型，标签模型 |
| published\_time | 时间类型，动态发布时间 |

用户个性模型user\_personality各字段定义如表3-15所示。

1. 用户个性模型各字段定义

|  |  |
| --- | --- |
| 字段名 | 说明 |
| user | 与user模型呈现“一对一”关系 |
| tags | 与tag模型呈现“多对多”关系 |

搜索历史纪录模型search\_history各字段定义如表3-16所示。

1. 搜索历史数据模型各字段定义

|  |  |
| --- | --- |
| 字段名 | 说明 |
| username | 字符型，用户名 |
| content | 字符型，搜索内容 |
| time | 时间类型，搜索时间 |

（2）逻辑设计

数据模块实现上传位置数据、上传自定义标签、上传搜索数据的功能。

首先校验Token与用户设备ID号，两者均正确时功能才能被执行。上传位置数据，输入用户名、经度、维度，后台在用户位置模型中添加相应记录；上传自定义标签，输入用户名、标签组、操作（增/删），首先分离标签组，若操作“增”，则在用户个性模型tags字段增加相应多对多关系，若操作为“删”，则在用户个性模型tags字段删除相应多对多关系；上传搜索数据，输入用户名、搜索内容，后台在在搜索历史纪录模型中添加相应记录。以上功能的程序流程图3-21所示。（校验Token与用户设备号部分在程序流程图中省略）



1. 上传位置数据、上传自定义标签、上传搜索数据流程图

第四章 作品实现

本章介绍团队如何实现上述设计方案。本章首先介绍几个核心算法的实现，然后介绍后台逻辑的实现，最后介绍Android端的实现。

4.1 核心算法实现

3.2.1 爬虫模块实现

首先，需要抓取待抽取信息的讲座的URL，我们通过爬虫的基本框架来抓取。其部分核心代码如下：

|  |
| --- |
| if(University=="武汉大学")and(Constitution=="计算机学院"):  html = requests.get(url)  html.encoding = 'utf-8'  html = html.text  soup = BeautifulSoup(html,'html.parser')  while(len(soup.find\_all("a",text="下一页"))!=0):  lectures\_url\_t = soup.find\_all("dd")  for l in lectures\_url\_t:  lectures\_urls.append("http://cs.whu.edu.cn" + l.a["href"])    \_url = url + soup.find\_all("a",text="下一页")[0]["href"]  html = requests.get(\_url)  html.encoding = 'utf-8'  html = html.text  soup = BeautifulSoup(html,'html.parser')  lectures\_url\_t = soup.find\_all("dd")  for l in lectures\_url\_t:  lectures\_urls.append("http://cs.whu.edu.cn" + l.a["href"]) |

得到了讲座的URL之后，抓取相关信息（讲座名称、讲座人、讲座时间、讲座地点等）的相关代码如下：

|  |
| --- |
| for item in items:  item=item.text  if"题目" in item or "主题" in item or "授课内容" in item:  try:  lec\_title = re.split(":|：",item,maxsplit=1)[1].strip()  except:  continue  elif "主讲人" in item and "简介" not in item:  lec\_author = re.split("：|:",item)[1].strip()  elif "报告人" in item and "简介" not in item:  lec\_author = re.split("：|:",item,maxsplit=1)[1].strip()  elif "授课人" in item:  lec\_author = re.split("：|:",item)[1].strip()  elif "讲座地点"in item or "地点" in item:  try:  lec\_location = re.split("：|:",item)[1]  except:  continue  if "经管院" in lec\_location:  lec\_location = lec\_location[3:]  elif "经济与管理学院" in lec\_location :  if "武汉大学" not in lec\_location:  lec\_location = lec\_location[7:]  else:  lec\_location = lec\_location[11:]  elif"讲座时间" in item or "时间" in item:  if"年" in item and "月" in item and "日" in item:  \_data=re.split(":|：",item,maxsplit=1)[1].strip()  \_data1 = \_data.split("日")[0]  \_data2 = \_data.split("日")[1]  lec\_month = \_data1.split("年")[1].split("月")[0].strip()  lec\_day = \_data1.split("月")[1]  if "点" in \_data2:  lec\_hour = \_data2.split("点")[0].strip()  if"半" not in \_data2 and "分" not in \_data2:  lec\_min = "00";  elif "分" in \_data2:  lec\_min = \_data2.split("点")[1].strip().split("分")[0].strip()  elif "～" in \_data2:  re\_hour = \_data2.split("～")[0].strip()  sp\_hour = re.split(":|：",re\_hour)  lec\_hour = sp\_hour[0].strip()  lec\_min = sp\_hour[1].strip() |

3.2.2 Lucene索引与检索模块实现

对于运用Lucene框架的模块，我们贴出我们使用Lucene的一些核心技术，而不贴出Lucene源代码。我们对于Lucene的相关运用见如下代码：

|  |
| --- |
| public static List<Lecture> NormalSearchLectures(String queryString ) throws Exception{  String index = "lectures\_index";  String[] field = new String[]{"title","author"};    IndexReader reader = DirectoryReader.open(FSDirectory.open(Paths.get(index)));  IndexSearcher searcher = new IndexSearcher(reader);  Analyzer analyzer = new CJKAnalyzer();    MultiFieldQueryParser parser = new MultiFieldQueryParser(field, analyzer);  // Query query = parser.parse(queryString);  Query query = queryExpansion\_search(queryString);  // System.out.println("Searching for: " + query);  // TopDocs results = searcher.search(query, 50,new Sort(SortField.FIELD\_SCORE, new SortField("time", SortField.Type.STRING)));  TopDocs results = searcher.search(query, 50);  int numTotalHits = results.totalHits;  // System.out.println("Results num:" + numTotalHits);  ScoreDoc[] hits = results.scoreDocs;  Document doc = null;  List<Lecture> lectures = new ArrayList<Lecture>();  Lecture lecture = null;  float score = (float)0.0;  for (int i=0; i<hits.length; i++) {  doc = searcher.doc(hits[i].doc);  score = hits[i].score;  lecture = new Lecture(doc.get("id"), doc.get("title"), doc.get("author"), doc.get("time"), score, doc.get("lat"), doc.get("lng"));  lectures.add(lecture);  }    return lectures;    } |

在扩展同义词方面，我们运用人工编辑同义词表的形式实现，相关代码如下：

|  |
| --- |
| public static String QueryExpansionbyTYC(String old\_queryString){  String new\_queryString = "";  ArrayList<Integer> index = new ArrayList<Integer>() ;  for(int i = 0; i < TYC.size(); i++){  ArrayList<String> tycs = TYC.get(i);  for(String tyc : tycs){  if (old\_queryString.contains(tyc)){  index.add(i);  continue;  }  else{  //do nothing  }  }  }  if(index.size()>0){  for(int \_index : index){  ArrayList<String> words = TYC.get(\_index);  for (String w : words){  if(!old\_queryString.contains(w)){  new\_queryString = new\_queryString.concat(" "+w);  }  }  }  }  else{  //we do nothing  }  return new\_queryString;  } |

在扩展得分方面，相关代码如下：

|  |
| --- |
| public static List<Lecture> SearchLectures(String queryString, String url ) throws Exception{  List<Lecture> lecs = new ArrayList<Lecture>();  List<Lecture> normal\_lecs = NormalSearchLectures(queryString);  Map<Integer, Double> scoreMap = new HashMap<Integer, Double>();  SimpleDateFormat dataFormat = new SimpleDateFormat("yyyy-MM-dd-HH-mm");  Date data\_user = new Date();  Date data\_lec;  int lec\_size = normal\_lecs.size();  ArrayList<Double> timeList = new ArrayList<Double>();  ArrayList<Double> distList = new ArrayList<Double>();  double a = 0.6;  double b = 0.2;  double c = 0.2;  double margin\_time;  double margin\_dist;    for(int i = 0; i < lec\_size; i++){  Lecture lec = normal\_lecs.get(i);  data\_lec = dataFormat.parse(lec.time);  margin\_time = data\_lec.getTime() - data\_user.getTime();  double[] loc = getUserLocation(url);  margin\_dist = Math.sqrt(Math.pow((loc[0]-Double.parseDouble(lec.lng)), 2)+Math.pow((loc[1]-Double.parseDouble(lec.lat)), 2));  if(margin\_time<0){  lec\_size--;  normal\_lecs.remove(i);  }  else{  timeList.add(margin\_time);  distList.add(margin\_dist);  }  }    double max\_t = 0;  double min\_t = Double.MAX\_VALUE;  double max\_d = 0;  double min\_d = Double.MAX\_VALUE;    for(int i = 0; i < lec\_size; i++){  double t = timeList.get(i);  double d = distList.get(i);  if(Double.compare(t, max\_t)>=0){  max\_t = t;  }  if(Double.compare(min\_t,t)>=0){  min\_t = t;  }  if(Double.compare(d, max\_d)>=0){  max\_d = d;  }  if(Double.compare(min\_d,d)>=0){  min\_d = d;  }  }    for(int i = 0; i < lec\_size; i++){  Lecture lec = normal\_lecs.get(i);  double s1 = lec.score;  double s2 = (max\_t - timeList.get(i))/(max\_t - min\_t);  double s3 = (max\_d - distList.get(i))/(max\_d - min\_d);  double S = a\*s1+b\*s2+c\*s3;  scoreMap.put(i, S);  }    ArrayList<Entry<Integer, Double>> l = new ArrayList<Entry<Integer, Double>>(scoreMap.entrySet());  Collections.sort(l, new Comparator<Map.Entry<Integer, Double>>() {  public int compare(Map.Entry<Integer, Double> o1, Map.Entry<Integer, Double> o2) {  return (int) ((o2.getValue() - o1.getValue()) \* 100000000000000000.0);  }  });    Lecture lec;  for(int i = 0; i < lec\_size; i++){  int index = l.get(i).getKey();  lec = normal\_lecs.get(index);  lec.SetScore(scoreMap.get(index));  lecs.add(lec);  }  return lecs;  } |

3.2.1 个性化推荐模块实现

本模块的实现依托于对用户画像数据、用户搜索数据的抓取，进行推荐的核心代码如下：

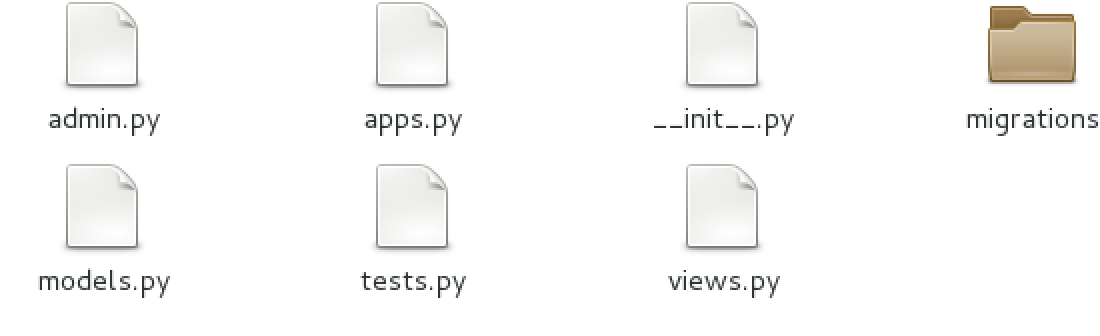
|  |
| --- |
| public static List<Lecture> Recommend\_Lectures(String sh\_url, String kw\_url) throws IOException{  List<Lecture> lecs = new ArrayList<Lecture>();  ArrayList<String> sh = new ArrayList<String>();  ArrayList<String> kw = new ArrayList<String>();  String kw\_string = "";  String sh\_string = "";  String index = "lectures\_index";    String[] field = new String[]{"title","author"};  Analyzer analyzer = new CJKAnalyzer();  MultiFieldQueryParser parser = new MultiFieldQueryParser(field, analyzer);  BooleanQuery.Builder result = new Builder();  float boost\_sh = (float) 0.2;  float boost\_kw = (float) 1.0;  Query sh\_query = null;  Query result\_query = null;  Query kw\_query = null;    try {  sh = GetSearchHistory(sh\_url);  kw = GetKeyword(kw\_url);    } catch (IOException e) {  e.printStackTrace();  }    for(int i = 0; i < sh.size(); i++){  sh\_string = sh\_string.concat(" "+sh.get(i));  }  for(int j = 0; j < kw.size(); j++){  kw\_string = kw\_string.concat(" "+kw.get(j));  }    try {  sh\_query = parser.parse(sh\_string);  kw\_query = parser.parse(kw\_string);    BoostQuery new\_boostquery = new BoostQuery(sh\_query,boost\_sh);  BoostQuery old\_boostquery = new BoostQuery(kw\_query,boost\_kw);  result.add(old\_boostquery.getQuery(), BooleanClause.Occur.SHOULD);  result.add(new\_boostquery.getQuery(), BooleanClause.Occur.SHOULD);    result\_query = result.build();  }  catch (ParseException e) {  e.printStackTrace();  }    IndexReader reader = DirectoryReader.open(FSDirectory.open(Paths.get(index)));  IndexSearcher searcher = new IndexSearcher(reader);  Query query = result\_query;  TopDocs results = searcher.search(query, 50,new Sort(SortField.FIELD\_SCORE, new SortField("time", SortField.Type.STRING)));  int numTotalHits = results.totalHits;  ScoreDoc[] hits = results.scoreDocs;  Document doc = null;  Lecture lecture = null;  for (int i=0; i<hits.length; i++) {  doc = searcher.doc(hits[i].doc);  float score = hits[i].score;  lecture = new Lecture(doc.get("id"), doc.get("title"), doc.get("author"), doc.get("time"), score, doc.get("lat"), doc.get("lng"));  lecs.add(lecture);  }  return lecs;  } |

4.2 后台逻辑实现

4.2.1 Django逻辑实现步骤

后台逻辑主要基于Django，后台按照MVC的设计思想逐个实现各模块功能。因各模块的接口类型、数据库操作方面大同小异，且具体逻辑在上一章已详细展示，本节仅以用户模块为例说明实现步骤，其它模块不再赘述。

步骤1：新建模块。例如新建用户模块，在项目终端中输入命令“python manage.py startapp user”，输入命令后user模块被创建，user目录如图4-1所示。



1. user模块创建后目录

步骤2：定义模型。例如对于用户模块，编辑目录下的“models.py”文件，定义模型“UserPower”、“UserOnline”、“UserProfile”，代码如下：

|  |
| --- |
| from django.db import models  from django.contrib.auth.models import User  # Create your models here.  class UserPower(models.Model):  user = models.OneToOneField(User, blank=True, null=True, on\_delete=models.CASCADE, related\_name= 'user\_power')  publisher = models.BooleanField(default=False)  express = models.BooleanField(default=False)  def \_\_str\_\_(self):  return self.user.username  class UserOnline(models.Model):  user = models.OneToOneField(User, blank=True, null=True, on\_delete=models.CASCADE, related\_name= 'user\_online')  channel\_id = models.CharField(max\_length=150, blank=True, null=True, default=None)  def \_\_str\_\_(self):  return self.user.username  def create\_user(user):  UserOnline.objects.get\_or\_create(user=user)  class UserProfile(models.Model):  user = models.OneToOneField(User, blank=True, null=True, on\_delete=models.CASCADE, related\_name= 'user\_profile')  user\_identity = models.CharField(max\_length=150, blank=True, null=True, default=None)  nick\_name = models.CharField(max\_length=150, blank=True, null=True, default=None)  gender = models.CharField(max\_length=10, default='secret')  head\_icon = models.ImageField(upload\_to='img')  def \_\_str\_\_(self):  return self.user.username  class Meta:  db\_table = 'UserProfile' |

步骤3：定义接口URL。例如对于用户模块，编写目录中“url.py”，定义相关接口的URL，代码如下：

|  |
| --- |
| from django.conf.urls import url  from user import views as user\_views  urlpatterns = [  url(r'^log\_in/$', user\_views.LogInView.as\_view(), name='log\_in-view'),  url(r'^log\_out/$', user\_views.LogOutView.as\_view(), name='log\_out-view'),  url(r'^change\_password/$', user\_views.ChangePasswordView.as\_view(), name='change\_password-view'),  url(r'^register/$', user\_views.RegisterView.as\_view(), name='register-view'),  url(r'^upload\_profile/$', user\_views.UploadProfileView.as\_view(), name='upload\_profile-view'),  url(r'^get\_profile/$', user\_views.GetProfileView.as\_view(), name='get\_profile-view'),  url(r'^get\_image/$', user\_views.GetImageView.as\_view(), name='get\_image-view'),  url(r'^search\_user/$', user\_views.SearchUserView.as\_view(), name='search\_user-view'),  url(r'^get\_power/$', user\_views.GetPowerView.as\_view(), name='get\_power-view'),    ] |

步骤4：实现视图。例如对于用户模块，编写“views.py”，实现每一个功能。以修改密码为例，首先定义ChangePasswordView类，该类继承APIVIew，首先声明该接口进行Token校验，然后定义函数post，request为提交post请求数据。逻辑部分首先校验原密码，若校验成功则以方式存入新密码。根据不同错误返回不同的status值。代码如下：

|  |
| --- |
| class ChangePasswordView(APIView):  """  Authentication is needed for this methods  """  authentication\_classes = (TokenAuthentication,)  permission\_classes = (IsAuthenticated,)    def post(self, request, format=None):  token = re.split(r"(\s)\*",request.META["HTTP\_AUTHORIZATION"],maxsplit=1)[2]  try:  data = request.data  except ParseError as error:  return Response(  {'status':1, 'detail': 'Invalid JSON - {0}'.format(error.detail)},  status=status.HTTP\_400\_BAD\_REQUEST  )  if "old\_passwd" not in data or "new\_passwd" not in data or "channel\_id" not in data:  return Response(  {'status':2, 'detail': 'Wrong credentials'},  status=status.HTTP\_401\_UNAUTHORIZED  )  old\_passwd = data.\_\_getitem\_\_("old\_passwd")  new\_passwd = data.\_\_getitem\_\_("new\_passwd")  channel\_id = data.\_\_getitem\_\_("channel\_id")  user = User.objects.get(auth\_token=token)  online\_user, success = UserOnline.objects.get\_or\_create(user=user)  if online\_user.channel\_id == None:  return Response({'status':3, 'detail': "user is not online"})  if not online\_user.channel\_id==channel\_id:  return Response({'status':4, 'detail': "Channel id is wrong"})  try:  if not user.check\_password(old\_passwd):  return Response(  {'status':5, 'detail': 'Old password is incorrect'},  status=status.HTTP\_401\_UNAUTHORIZED  )  user.set\_password(new\_passwd)  user.save()    except:  return Response(  {'status':6, 'detail': 'Change password failed'},  status=status.HTTP\_401\_UNAUTHORIZED  )  return Response({'status':0, 'detail': "Change password succeeded"}) |

步骤5：迁移。以用户模块为例，在项目根目录依次输入命令“python manage.py makemigrations”、“python manage.py migrate”将数据模型以表的形式迁入MySQL数据库。

其它各模块的实现与用户模块的实现基本类似，为节省篇幅不再赘述。

4.2.2 Spring MVC接口实现

Spring MVC支持REST风格的API，使用Jackson模块解析或者导出json数据串。作品个性化推荐与检索相关接口基于Spring MVC实现，接口最关键的部分在于控制器的定义，控制器中通过类的方法与Request系列标注定义相关接口，并且CrossOrigin标注可轻松解决跨域传送问题。作品控制器定义的代码如下所示：

|  |
| --- |
| package search;  import org.springframework.web.bind.annotation.CrossOrigin;  import org.springframework.web.bind.annotation.RequestBody;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.bind.annotation.RequestParam;  import org.springframework.web.bind.annotation.ResponseBody;  import org.springframework.web.bind.annotation.RestController;  import def.AddLecture;  import org.springframework.web.bind.annotation.RequestMethod;  @RestController  public class SearchController {  @CrossOrigin  @RequestMapping(value ="/search", method = RequestMethod.GET)  public @ResponseBody Search search(@RequestParam(value="query", defaultValue="test") String query) throws Exception {  return new Search(query);  }  @CrossOrigin  @RequestMapping(value = "/index",  method = RequestMethod.POST,  headers = {"Content-type=application/json"})  public Index index(@RequestBody AddLecture lecturesJSON ) {  return new Index(lecturesJSON);  }  @CrossOrigin  @RequestMapping(value = "/single\_push", method = RequestMethod.GET)  public AndroidPushMsgToSingleDevice msg(@RequestParam(value="username", defaultValue="test") String username,  @RequestParam(value="token", defaultValue="test") String token )throws Exception {  return new AndroidPushMsgToSingleDevice(username, token);  }    @CrossOrigin  @RequestMapping(value = "/tag\_push", method = RequestMethod.GET)  public TagPush tagPush(@RequestParam(value="id", defaultValue="test") String id,  @RequestParam(value="token", defaultValue="test") String token )throws Exception {  return new TagPush(id, token);  }    @CrossOrigin  @RequestMapping(value = "/location\_push", method = RequestMethod.GET)  public LocationPush locationPush(@RequestParam(value="id", defaultValue="test") String id,  @RequestParam(value="token", defaultValue="test") String token )throws Exception {  return new LocationPush(id, token);  }    } |

4.2.3 REST API

后台对每个功能都实现了相应的接口，均为REST风格的API。API分为两种，一种为提供给Android终端的API，另一种为后台实现需要的API，后者需要接口需要专门的Token校验，仅后台开发人员可以使用。提供给Android终端的REST API如表4-1所示。

1. 提供给Android终端的REST API

|  |  |  |
| --- | --- | --- |
| URL | 类型 | 说明 |
| /user/log\_in/ | POST | 用户登录 |
| /user/change\_password/ | POST | 用户修改密码 |
| /user/register/ | POST | 用户注册 |
| /user/log\_out/ | POST | 用户注销 |
| /user/upload\_profile/ | POST | 用户上传个人资料 |
| /user/get\_profile/ | POST | 用户获取个人资料 |
| /user/get\_image/ | POST | 用户获取头像 |
| /user/search\_user/ | POST | 用户精确搜索 |
| /user/get\_power/ | POST | 获取用户权限信息 |
| /data/upload\_place/ | POST | 上传用户位置数据 |
| /data/upload\_search\_history/ | POST | 上传用户搜索数据 |
| /data/upload\_personality/ | POST | 上传用户标签 |
| /friend/request\_friend/ | POST | 请求添加好友 |
| /friend/respond\_request/ | POST | 回应好友请求 |
| /friend/get\_friends/ | POST | 获取好友列表 |
| /friend/delete\_friend/ | POST | 删除好友 |
| /friend/get\_requests/ | POST | 获取好友请求信息 |
| /chat/get\_channel/ | POST | 获取设备ID号 |
| /chat/upload\_chat/ | POST | 上传聊天记录 |
| /chat/download\_chat/ | POST | 下载聊天记录 |
| /moment/create\_moment/ | POST | 发布动态 |
| /moment/moments\_of\_publisher/ | POST | 查询发布的动态 |
| /moment/delete\_moment/ | POST | 删除动态 |
| /moment/moments\_of\_friends/ | POST | 获取好友请求 |
| /express/create\_express/ | POST | 发布快递信息 |
| /express/delete\_express/ | POST | 删除快递信息 |
| /express/expresses\_of\_publisher/ | POST | 获取发布的快递信息 |
| /express/expresses\_of\_friends/ | POST | 获取好友发布的快递信息 |
| /lecture/create\_lecture/ | POST | 发布讲座 |
| /lecture/delete\_lecture/ | POST | 删除讲座 |
| /lecture/lectures\_of\_publisher/ | POST | 获取发布的讲座 |
| /search/query | GET | 讲座查询 |

提供给后台开发人员的REST API如表4-2所示。

1. 提供给开发人员的REST API

|  |  |  |
| --- | --- | --- |
| URL | 类型 | 说明 |
| /data/get\_keywords/ | GET | 获取用户标签对应关键字 |
| /data/get\_search\_history/ | GET | 获取用户搜索记录 |
| /data/fetch\_channel\_id/ | GET | 获取用户设备号 |
| /data/get\_tag/ | GET | 获取新讲座对应标签 |
| /data/get\_current\_location/ | GET | 获取用户当前位置 |
| /data/get\_location\_user/ | GET | 获取用户历史位置信息 |

4.3 Android端实现

第五章 作品展示与测试

5.1 作品UI展示

5.2 作品测试

5.2.1 测试方案

5.2.1.1 后台接口测试方案

后台接口使用基于Django的REST Framework框架中APITestCase类进行测试。APITestCase继承Django的TestCase类，可方便地设计测试用例，并用断言语句检测返回结果是否与预期相符。

后台接口测试采用白盒测试法，即针对每个逻辑功能在不同状况下的返回值设计测试用例，检验测试用例返回结果是否与预期相符。具体测试步骤将在5.2.3.2节举例说明。

5.2.1.2 性能测试方案

移动端开发中，由于移动设备自身的软硬件特点，性能是必须要考虑的因素。性能评估上，我们进行View测试和压力测试。

View测试使用过度绘制（overdraw）进行测试与评估。过度绘制描述的是屏幕上某个像素在同一帧时间内被绘制了多少次。在多层次的UI结构里面，如果不可见的UI也在做绘制的操作，这就会导致某些像素区域被绘制了多次。这就浪费大量的CPU以及GPU资源。对于过度绘制的测试方法，Google为我们提供了及其便利的工具。打开手机开发者选项，打开调试过度绘制的开关，即可看到结果。结果分别用颜色去标记相应的图层，紫色为最佳，绿色次之，然后粉色，最后红色，红色已经产生较严重的性能问题。

压力测试使用Monkey生成大量随机事件进行测试。Monkey是运行在模拟器上和真机设备上的一段程序，它会产生用户事件的一系列伪随机流，比如点击、触摸、手势，还有一些系统级别的事件。Monkey通过一种随机且大量重复的方式来做应用的压力测试。

5.2.2 测试环境

（1）服务器

CPU：1核，1Gbhz

内存：1GB

系统盘：20GB

操作系统：CentOS 7.0 64位

公网宽带：1Mbps

（2）Android终端

5.2.3 测试及分析

5.2.3.1 后台接口

首先以用户模块的登陆接口测试为例进行说明。测试采用白盒测试法，登陆部分代码如下：

|  |
| --- |
| class LogInView(APIView):  """  """    def get(self, request, format=None):  return Response({'detail': "GET Response"})    def post(self, request, format=None):  try:  data = request.data  except ParseError as error:  return Response(  {'status':1, 'detail': 'Invalid JSON - {0}'.format(error.detail)},  status=status.HTTP\_400\_BAD\_REQUEST  )  if "user" not in data or "password" not in data or "channel\_id" not in data:  return Response(  {'status':2, 'detail': 'Wrong credentials'},  status=status.HTTP\_401\_UNAUTHORIZED  )  username = data.\_\_getitem\_\_("user")  password = data.\_\_getitem\_\_("password")  channel\_id = data.\_\_getitem\_\_("channel\_id")  try:  user = User.objects.get(username=username)  except:  return Response(  {'status':3, 'detail': 'No such user'},  status=status.HTTP\_401\_UNAUTHORIZED  )  if not user.check\_password(password):  return Response(  {'status':4, 'detail': 'Password is incorrect'},  status=status.HTTP\_401\_UNAUTHORIZED  )  online\_user, success = UserOnline.objects.get\_or\_create(user=user)  if not online\_user.channel\_id == None and not online\_user.channel\_id == channel\_id:  try:  old\_channel\_id = online\_user.channel\_id  online\_user.channel\_id = channel\_id  online\_user.save()  #push offline notification  except:  return Response(  {'status':5, 'detail': 'Switch user error'},  status=status.HTTP\_401\_UNAUTHORIZED  )  elif online\_user.channel\_id == None:  try:  online\_user.channel\_id = channel\_id  online\_user.save()  except:  return Response(  {'status':6, 'detail': 'Login in the first device error'},  status=status.HTTP\_401\_UNAUTHORIZED  )  token = Token.objects.get\_or\_create(user=user)    return Response({'status':0,'detail': 'POST answer', 'token': token[0].key}) |

可发现，程序对于不同的错误类型返回不同的“status”值。我们着重对逻辑进行测试，对于由于异常产生的“status”值不进行测试。测试之前首先创建用户“john”，密码为“johnpassword”设计测试样例如表5-1所示。

1. 登陆功能测试样例

|  |  |  |
| --- | --- | --- |
| 序号 | 参数 | 预期结果 |
| 1 | {'user': 'john', 'password': 'johnpassword', 'channel\_id':'1'} | 成功，status=0 |
| 2 | {'password': 'johnpassword', 'channel\_id':'1'} | 缺少参数，status=2 |
| 3 | {'user': 'Kim', 'password': 'johnpassword', 'channel\_id':'1'} | 用户不存在，status=3 |
| 4 | {'user': 'john', 'password': '12345', 'channel\_id':'1'} | 密码错误，status=4 |

编写用户模块的“tests.py”文件。首先定义测试类如下：

|  |
| --- |
| class LoginTest(APITestCase): |

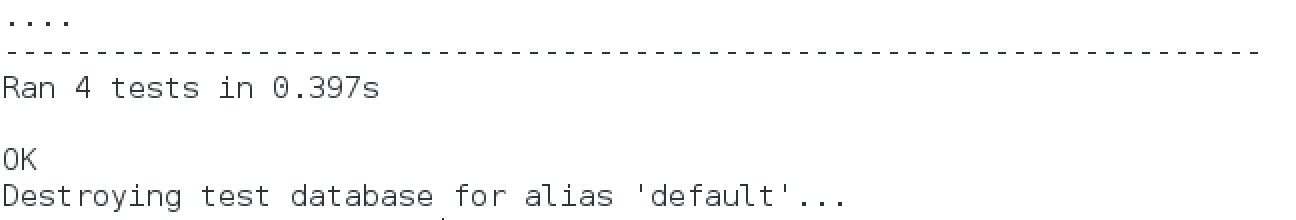
接着，定义初始化函数，初始化用户“john”以及测试用例，代码如下：

|  |
| --- |
| def setUp(self):  self.superuser = User.objects.create\_superuser('john', 'john@snow.com', 'johnpassword')  self.data0 = {'user': 'john', 'password': 'johnpassword', 'channel\_id':'1'}  self.data2 = {'password': 'johnpassword', 'channel\_id':'1'}  self.data3 = {'user': 'Kim', 'password': 'johnpassword', 'channel\_id':'1'}  self.data4 = {'user': 'john', 'password': '12345', 'channel\_id':'1'} |

最后编写每一个状态的测试代码，对于每一个测试用例，使用断言与预期结果进行比对，代码如下：

|  |
| --- |
| def test\_status\_0(self):  response = self.client.post(reverse('user:log\_in-view'), self.data0)  s = json.loads(response.content.decode("utf-8"))  self.assertEqual(s['status'], 0)  def test\_status\_2(self):  response = self.client.post(reverse('user:log\_in-view'), self.data2)  s = json.loads(response.content.decode("utf-8"))  self.assertEqual(s['status'], 2)  def test\_status\_3(self):  response = self.client.post(reverse('user:log\_in-view'), self.data3)  s = json.loads(response.content.decode("utf-8"))  self.assertEqual(s['status'], 3)  def test\_status\_4(self):  response = self.client.post(reverse('user:log\_in-view'), self.data4)  s = json.loads(response.content.decode("utf-8"))  self.assertEqual(s['status'], 4) |

编写完成测试代码后，在项目根目录输入命令“python manage.py test”，Django自动对每个测试用例进行测试、比对，结果如图5-1所示。



1. 登陆功能测试结果

可发现四个测试用例的返回结果与预期均相同，测试通过。其它模块各功能测试步骤与此相同，不再赘述。经过团队的测试与调试，确认了后台每个接口均能正常工作。

5.2.3.2 性能测试

第六章 团队介绍及人员分工

团队成员均来自武汉大学计算机学院，包括研究生一名，本科生两名。团队各成员介绍及分工如下：

查道琛，武汉大学计算机学院计算机科学与技术专业2014级本科生，擅长Java、Python、服务器后台的搭建及相关技术，曾获得过国家奖学金、全国大学生信息安全竞赛一等奖等，有较丰富的项目经验。本作品中主要负责统筹安排、数据库的设计、后台接口的开发，以及相关文档的撰写。

第七章 其它

7.1 开发结束后的感触

从最开始的一个想法，到一步步的需求分析、数据库设计、后台逻辑设计、检索与推荐、UI设计，到最后产品雏形的形成以及测试，团队中的每一个成员都在不断学习、不断产生新的想法、不断完善我们的作品。我们希望做出一个不一样的校园类APP，提供不一样的服务，方便更多的大学生。这个作品是团队共同努力的结果，充斥着团队每一个成员的汗水。

决定参加这个比赛的一开始，我们是没有什么头绪的，我们只有一个初步的想法——设计一个基于位置的校园个性化服务APP，却不知道从何入手，不知道怎样的产品能够吸引眼球。我们从试用同类型的其他面向校园的APP开始，一点点地整理我们的思路。我们逐渐发现，大多数校园类APP没有充分挖掘移动互联网基于位置的特性。以此为入口点，我们开始做需求分析，分析大学生需要什么样的服务。经过讨论，我们慢慢设想出了讲座发布与个性化检索、基于位置的讲座推荐、帮拿快递平台等零零散散的思路。又经过了很长时间的讨论，我们才真正将我们的思路理清楚，转入开发阶段。

开发过程也是漫长而充满挑战的，大到个性化推荐算法的设计、模糊搜索的实现，小到一个函数、一行代码的编写，常常让我们煞费苦心。好在团队成员是好学的，并且对这样一个产品的开发充满着兴趣。我们克服了一个又一个的难点、调通了一个又一个的bug，终于及时地完成了产品demo。其中，我们的做得最久的是UI设计。一个好的安卓APP必定需要一个流畅、美观的UI，我们花了大力气进行UI布局设计、图标设计、线程的优化，虽然最后的作品还没有完全达到我们预想的样子，但是作为一个demo还是可以胜任的。

这个作品充斥着我们的汗水与心血，也包含着我们团队成员的一种情怀——用技术让大学生的生活更加方便。我们真的希望我们的这些想法能被应用于商业化的产品中，让每一个大学生都能享受这样的基于位置的个性化服务。

7.2 不足之处和今后的设想

由于时间所限，我们的作品以演示demo为目的，确有一些地方还做的得不到位。首先，我们只爬取了300余条讲座进行建库索引，在实际环境下，这是远远不够的。只有让讲座覆盖面更广、更全，才能真正把讲座搜索、个性化推荐做好。其次，因人力有限，我们所有的服务都只针对了本校，对于其它学校没有予以考虑。再次，我们虽然做了不少测试，但是测试的依然不够全面，在少数极端情况下依然会出现卡顿、错误等情况。

今后，如果这样的一个APP能真正运营起来，有许多方面需要进一步完善。首先，覆盖面得够广，爬虫模块需要进一步扩充，以适应不同学校、不同网站的需求；其次，需要搜集更多样化的用户数据，分析得到更加全面的用户画像；再次，后台方面需要增加更多的扩展服务器，以适应实际的业务需求。总之，从这样的一个demo到一个能上线的产品还是有一大段距离的，需要更多的人力与时间。

7.3 用到的第三方代码或组件

后台：

1. Django框架：基于Python的后台开发框架；
2. Spring MVC：基于Java的后台开发框架；
3. Django Restframwork：基于Django的接口框架；
4. Lucene：Apache提供的全文索引工具包；
5. 百度云推送SDK：百度云推送服务器端的SDK；
6. 百度地图API：百度地图提供的位置服务相关接口；
7. re库：基于Python的正则表达式模块；
8. json库，基于Python的json处理模块；
9. requests库，基于Python的Http请求模块；
10. BeautifulSoup库：用来规则化html文本。

前端：

1. 百度地图SDK：百度地图开发基于位置服务的SDK；
2. 百度云推送SDK：百度运推送开发推送功能的SDK；

参考资料

1. 佚名. TalkingData发布2015年中国移动互联网发展指数数据报告[J]. 中国金融电脑, 2015(10):95-95.
2. 百度百科：LBS. <http://baike.baidu.com/link?url=fLHalJR9cGqRVuff_n6YkTg1zFsj3KXd8pcZ8QywxTEgMSmQg8NCaYMvMT__NtOkweYsRsQj7xDabsMlOCLvxK>
3. 百度百科：lucene. <http://baike.baidu.com/link?url=da6khfeStopFCZV_4D2qVBKo0ltw-hKl8oMnS5fy_6Mq3GVFFFByH2Nep8zjJ2MCR3iAOw_EOvI2P_2PNa3nO_>
4. 百度百科：django. <http://baike.baidu.com/subview/962167/9372788.htm#viewPageContent>
5. 百度百科：spring MVC. <http://baike.baidu.com/link?url=SEiwBOdnQVEpHNxgOrGKJbUWKCW82HklremwKe5sRoV72w-t40RWB4frq_HmTgBi_Y-x9gu5cg1Ln0MjVJ9CfaKkTCAmwPLC2YRITa3MJT_>
6. 百度百科：用例图. <http://baike.baidu.com/link?url=L28B63FfP301xW0Dabep_AKrcz0MYOZ0z6vxkZKnz-tEB0CkK0TEFQ5Wo1Gzd2dsm3hW9feqdPSuUFMQb4dwy09iaOULCwiCCJPecS66iaU-fosIfkQwF8b1k0B6uN72>