.﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽有ou﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽ .4﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽可以﷽述、元数据等的是，他们**基于Web的房产信息系统的设计与实现**

摘 要

随着我国国民经济水平的不断提高，以及大量人口涌入城市，产生了大量的住房需求，导致了房地产行业的飞速发展。然而行业的快速发展也带来了不少新问题，房源信息体量越来越大，广大用户的需求也越来越多变。传统的线下实体店通过张贴海报图片等宣传方式已经不能满足如今的需求。与此同时，互联网技术也在不断发展和完善，互联网使得人们获取信息不再有空间和时间局限性，人们能够随时随地通过互联网获取信息。目前互联网技术已经应用到了各种生活场景中，极大地满足了人们对所需信息的需求。本设计立足于解决人们对房产信息的实际需求，通过运用互联网技术解决传统房产行业信息化不足的问题。

本文主要阐述了一个B/S架构的房产信息系统的设计与实现。对Java Spring 技术栈，MySQL数据库，Solr搜索引擎，协同过滤推荐算法，WebSocket以及React.js等技术进行了详细的论述。同时论述了房产信息系统的需求分析，架构设计，数据库表结构,详细模块的划分与设计，以及具体的功能实现方法。

**关键词：**互联网+，房产信息，Web，Java web，React.js，搜索，推荐

**待翻译**

**ABSTRACT**

**Key words：**

目 录(待完善)

[1 引 言 1](#_Toc483592149)

[1.1 课题研究的背景 1](#_Toc483592150)

[1.2 课题研究的现状 1](#_Toc483592151)

[1.3 课题的主要工作 1](#_Toc483592152)

[1.4 课题的组织结构 2](#_Toc483592153)

[1.5 本次设计的工作 2](#_Toc483592154)

[2 系统相关知识 3](#_Toc483592155)

3 系统需求分析

4 系统实现设计

5 系统实现

6 关键技术

7 展望与总结

[参考文献 38](#_Toc483592201)

[谢 辞 40](#_Toc483592202)

# 1 引 言

## 课题研究的背景和意义

## (1)传统房产信息行业存在的问题

## 随着人们生活水平不断提高，大众对于“衣食住行”中的住的需求日益增加，使我国房地产行业的发展蒸蒸日上：不断涌现的规格不一的房地产公司，不断扩大的业务范围，不断增多的获取房地产信息的需求。然而由于传统行业的局限性，使得房产信息不对称，虽然房地产公司的数量多了，但并没很好的解决人们找房难的问题。目前部分的传统房地产公司的信息化智能化能力还比较弱，通常只有线下实体店，以致于客户无法方便快捷的获取到楼盘及房源信息。公司的产品推广只能局限于某个片区，从而无法进一步发展业务。传统的房产信息管理软件大多是桌面程序，无法通过互联网获取房产交易信息，房产数据的录入大多采用人工录入和手工导入的数段，数据具有很大的滞后性，大量的房产交易数据无法及时更新，由于数据的重复性、出错率大大升高，并且数据的时效性降低，不同网点之间的信息沟通不畅通，使得公司的工作人员的工作效率低下，给客户和公司的经营管理带来了不少麻烦。另一方面是对客户的交易行为及偏好没有很好的分析手段，难以满足客户个性化的找房买房需求。

## (2)互联网+

## 近年“ 互联网+” 已经成为了一种全民现象，互联网与制造业交通运输等传统行业的结合越来越紧密，并且促进了传统行业效率的提升。网上售票、网上点餐、在线约车、共享单车等“互联网+”项目极大的方便了人们的生活。“ 互联网+” 即计算机信息技术以及互联网平台，实现互联网与传统行业紧密结合，充分发挥两者各自的优势实现互补合作，创造新的发展形态。 “ 互联网+” 将互联网技术成果运用于经济、社会、文化各领域之中，提高了社会的生产力和创新力，让人们享受到信息革命带来的好处，形成更泛的以互联网为基础设施和实现工具的经济发展新形态，推动社会发展和进步，提升整个社会资源的配置效率。

## (3)互联网+房产

## 计算机软件产业日益成熟，云计算、大数据、人工智能等新技术的出现于快速发展，加强了与房地产行业的“ 互联 +” 模式的联系。随着“互联网+”的发展，房地产行业也迎来了向信息化智能化转变的转折点。通过运用“互联网+”思维，把房产信息搬到互联网上，利用互联网信息传播的优势，解决房产行业信息不对称的问题，实现行业服务水平的优化与提升。同时，利用数据分析智能推荐等技术，提升产信息系统的信息化与智能化水平。

## (4)结论

## 由于房产交易活跃、市场规模不断扩大，产生了大量的房产信息，并且信息具有更新速度快的特点。对于房产公司来说，及时地收集房产交易信息至关重要，并且能准确地对信息进行分类管理，同时能及时地找到满足客户需要的房源。客户是公司在激烈的行业竞争中，能够取胜的唯一法宝，而想要留住用户就必须提供优质的内容，并且为用户推荐个性化的楼盘房源信息，尽可能的满足用户找房需求。房产信息行业属于信息密集型产业，房产信息的及时、准确以及信息沟通的顺畅是保障业务开展的重要基础。所以行业需要进行信息化智能化，利用先进的计算机互联网技术帮助行业来管理和展示信息。虽然市面上存在安居客、58同城等功能齐全的房产信息平台，但这些网站只是一个平台，难以满足个性化的需求，且信息的真实性难以保证。

## 在上述背景下，需要建设基于 Web 的房产信息系统，满足房地产公司未来的经营发展的需要。按照实际业务的需求，实现楼盘房源信息的管理及展示。实现用户对大量房源数据的分类搜索，并且分析用户行为并且为用户推荐合适的信息。实现实时的客服交流，快速满足用户的咨询。

## 1.2 课题研究的现状(待完成)

目前房产中介的模式

安居客

链家

## 1.3课题的主要工作

本文在调查及分析了房产行业现状后，得出了房产信息系统的设计及实现构思。使用Java EE这类比较成熟的开发技术，以及MySQL, Solr, React.js等广受欢迎的开源技术，实现了一套可行的房产信息管理系统，为传统中小型房地产公司实现信息化提供了参考。

该系统是基于Web的解决方案，前后端分类的设计，B/S架构，具有可扩展性并发性及跨平台的便利性。系统主要包括房源及楼讯这两大部分，有房源的上传、展示及修改删除功能模块，楼讯亦是如此，还包括房源的关键词模糊搜索，相似房源的推荐，为注册用户提供符合偏好的房源推荐已经客服咨询功能。有了这些贴近用户需求的功能，可以为用户快速定位到所需的房源信息，为用户节约成本，同时也促进了交易，帮助房产公司实现更多的销售业绩。

本设计的主要内容有以下几点：

1. 研究前端技术React.js，采用面向组件模块的开发方式，降低耦合度，提高可维护性。
2. 研究后端技术Java EE Spring 技术栈，Spring框架对Java对象的管理，Spring MVC对程序层次结构的控制以及API路由的管理，Spring Boot管理整个程序代码的组织结构及相关配置，Java ORM框架Mybatis 与MySQL数据库进行数据层的交互。
3. 研究Solr搜索引擎的使用与配置，实现房源信息的关键词搜索。
4. 研究推荐算法，结合系统选择出适合的推荐算法，实现相似房源的展示及房源推荐。
5. 研究WebSocket通信协议，实现实时的客服咨询。
6. 登录注册、房源管理、楼讯管理、搜索、推荐、咨询这几个模块的具体设计与实现。

## 1.4 课题的组织结构

论文总共分为六章，各章节的主要内容如下：

第一章：引言，主要说明了本设计研究的内容和背景，实现房产信息化的目的，互联网+的模式对行业带来的革新。并且给出了本文的组织结构。

第二章：系统相关技术及其基本概念，主要介绍实现本房产信息管理系统涉及到的技术，以及选择这些技术的优点和原因。同时也介绍了系统设计实现所采用的架构模式。

第三章：系统需求分析，采用软件工程需求分析的方式，分析了房产信息系统的相关需求，设计了各类用例，得出了相关的功能设计模块。良好的系统需求分析，使设计不偏离实现目的的轨道，保证完成的系统满足用户需求。

第四章：系统设计，根据系统需求分析得出的结果，设计了系统的总体架构，并且对总体架构的各个功能模块进行了详细设计，以及数据库表结构ER图的设计。

第五章：系统功能的实现，在系统需求分析和设计的基础上，利用所选的技术对各个模块进行具体的实现，登录注册、房源发布展示、楼讯发布展示、房源搜索及房源推荐等模块的关键代码逻辑。

第六章：关键技术，

第七章：总结与展望，总结了本设计所做的工作，一些不足之处和可以扩展的方向。

# 2 系统相关技术及其基本概念

## 2.1 B/S结构

(1)什么是B/S结构

B/S结构（Browser/Server，浏览器/服务器模式），是互联网技术兴起后的一种网络结构模式，Web浏览器是客户端最主要的应用软件，如IE、Chrome、Safari、Firefox等。这种模式统一了客户端，浏览器通过渲染HTML显示页面。将系统功能实现的核心部分集中到服务器上，简化了系统的开发、维护和使用。客户机上只要安装一个浏览器客户端，通过互联网协议与服务端程序进行交互，在服务端可以使用独立的应用服务器和数据库服务器。目前主要的网站，都采用的是B/S结构。结构图如下：

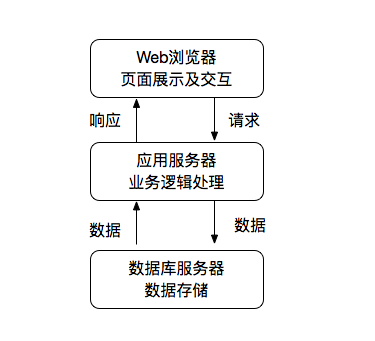


图 2.1 B/S结构

(2)B/S结构的优点

系统维护和升级简单，周期短，几乎可以随时更新。当前，软件系统的改进和升级越发频繁，B/S架构的产品明显体现着更为方便的特性。从可维护性方面，所有的客户端都是浏览器，不需要做多少维护工作。从更新升级方面，所有的操作只需要针对服务器进行，如果是异地，只需要把服务器连接专网即可，实现远程维护、升级和共享。客户端只是进行简单的请求，系统的部署发布是很容易的事情，不再像传统的C/S架构，需要重新发布客户端软件或者补丁，用户需要再次下载安装，软件的更新升级过程繁琐，周期长。

成本降低，服务器操作系统有更多的选择。如可以选择开源的Linux系统，当前的趋势是凡使用B/S架构的应用管理软件，只需安装在Linux服务器上即可，而且安全性高。

可扩展性高，可以通过增加服务器资源，负载均衡等方式应对大量并发的请求。

## 2.2 MVC模式

(1)什么是MVC模式

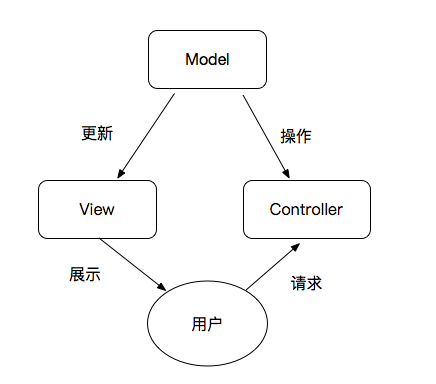
MVC模式（Model–view–controller）是软件工程中的一种软件架构模式，把软件系统分为三个基本部分：模型（Model）、视图（View）和控制器（Controller）。其目的是实现一种动态的程式设计，使后续对程序的修改和扩展简化，并且使程序某一部分的重复利用成为可能。除此之外，此模式通过对复杂度的简化，使程序结构更加直观。软件系统通过对自身基本部分分离的同时也赋予了各个基本部分应有的功能。

MVC将应用程序划分为三种组件，模型 - 视图 - 控制器设计定义它们之间的相互作用。

模型（Model） 用于封装与应用程序的业务逻辑相关的数据以及对数据的处理方法。“Model”有对数据直接访问的权力，例如对数据库的访问。“Model”不依赖“View”和“Controller”，也就是说， Model 不关心它会被如何显示或是如何被操作。但是 Model 中数据的变化一般会通过一种刷新机制被公布。为了实现这种机制，那些用于监视此 Model 的 View 必须事先在此 Model 上注册，从而，View 可以了解在数据 Model 上发生的改变。

视图（View）能够实现数据有目的的显示（理论上不是必需的）。在 View 中一般没有程序上的逻辑。为了实现 View 上的刷新功能，View 需要访问它监视的数据模型（Model），因此应该事先在被它监视的数据那里注册。

控制器（Controller）起到不同层面间的组织作用，用于控制应用程序的流程。它处理事件并作出响应。“事件”包括用户的行为和数据 Model 上的改变。



2.2 MVC模式图

(2)MVC模式的优点

低耦合：视图层和业务层分离，允许更改视图层代码而不用重新编译模型和控制器代码。一个应用的业务流程或者业务规则的改变只需要改动MVC的模型层即可，模型与控制器和视图相分离，容易改变应用程序的数据层和业务规则。

高重用性和可适用性：MVC模式允许使用各种不同样式的视图来访问同一个服务端的

代码，例如，很多数据可能用HTML来表示，但是也有可能用WAP来表示，而这些表示所需要的仅令是改变视图层的实现方式，而控制层和模型层无需做任何改变。

较低的生命周期成本: 使降低开发和维护用户接口的技术含量成为可能。

快速的部署:使用MVC模式使开发时间得到相当大的缩减，它使业务程序员集中精力于业务逻辑，界面程序员集中精力于界面展示上。

可维护性：分离视图层和业务层也使得WEB应用更便于维护和修改。

有利于软件工程化管理：由于不同的层各司其职，每一层不同的应用具有某些相同的特征，有利于通过工程化、工具化管理程序代码。

## 2.3 HTTP通信协议

HTTP是客户端浏览器或其他程序与Web服务器之间的应用层通信协议。在Internet上的Web服务器上存放的都是超文本信息，客户机需要通过HTTP协议传输所要访问的超文本信息。

HTTP协议的主要特点可概括如下：

(1)支持客户/服务器模式。

(2)简单快速：客户向服务器请求服务时，只需传送请求方法和路径。请求方法常用的有GET、HEAD、POST。每种方法规定了客户与服务器联系的类型不同。由于HTTP协议简单，使得HTTP服务器的程序规模小，因而通信速度很快。

(3)灵活：HTTP允许传输任意类型的数据对象。正在传输的类型由Content-Type加以标记。 (4)无连接：无连接的含义是限制每次连接只处理一个请求。服务器处理完客户的请求，并收到客户的应答后，即断开连接。采用这种方式可以节省传输时间。

(5)无状态：HTTP协议是无状态协议。无状态是指协议对于事务处理没有记忆能力。缺少状态意味着如果后续处理需要前面的信息，则它必须重传。另一方面，在服务器不需要先前信息时它的应答就较快。

## 2.4 前端框架React.js

ReactJS是Facebook开源的，为了解决构建随着时间数据不断变化的大规模应用程序而设计的用来构建用户界面的JavaScript库。目前社区有很多可复用的组件库，如本文使用的Ant Design UI组件库，使用组件库大大地缩短了前端页面的开发周期。

ReactJS有如下特点：

1. 采用单向数据流，易追踪数据。
2. 将HTML，JS结合起来构建组件，组件状态和结果意义对应起来，实现模块化开发。
3. 创建了高性能的虚拟DOM，避免直接操作DOM。即组件不是真实的DOM节点，而是存在于内存中的一种数据结构。所有的DOM变动都先在虚拟DOM上发生，然后再将实际发生变动的部分反映在真实DOM上。

对于组件化的具体说明：

React推荐以组件的方式去重新思考UI构成，将UI上每一个功能相对独立的模块定义成组件，然后将小的组件通过组合或者嵌套的方式构成大的组件，最终完成整体UI的构建。MVC让我们实现了表现，数据，控制的分离，开发中从技术的角度对UI进行了拆分；React启发开发者从功能的角度出发，将UI分成不同的组件，每个组件都独立封装。

组件的特征：

可组合，一个组件可以和其它组件一起使用，或者嵌套在另一个组件内部。

可重用，每个组件都有独立功能，可被使用在多个UI场景。

可维护，每个小的组件只包含自身的逻辑，更容易被理解和维护。

React 的问世把前端开发人员从复杂的数据交互和 UI 渲染不可预测的问题中解放出来，让的开发人员可以将精力专注于数据的变化上。于此同时，React 运用虚拟 DOM 技术，极大的提升了页面的性能。并且它通过对模块的划分，让应用的逻辑和数据流更加清晰明了。

### 2.5 后端技术Java及Spring 技术栈

2.5.1 Java

Java是一种广泛使用的计算机编程语言，拥有跨平台、面向对象、泛型编程的特性，广泛应用于企业级Web应用开发和移动应用开发。

Java编程语言的风格十分接近C++语言。继承了C++语言面向对象技术的核心，Java舍弃了C++语言中容易引起错误的指针，改以引用取代，同时移除原C++与原来运算符重载，也移除多重继承特性，改用接口取代，增加垃圾回收器功能。Java不同于一般的编译语言或直译语言。它首先将源代码编译成字节码，然后依赖各种不同平台上的虚拟机来解释执行字节码，从而实现了“一次编写，到处运行”的跨平台特性。

Java用于web开发的优势有以下几点：

(1)面向对象编程：面向对象的方式使代码具有很好的可维护性和可扩展性。

(2)广泛性：不但有成熟的设计模式，而且还有成熟的框架，可以用很多表达式以及标签来展示我们需要的内容。

(3)稳定性：拥有较高的安全性，使用广泛，目前Java已被许多大型网站使用，业界已经有丰富的实践，Java语言本身也在不断更新和进步。

2.5.2 Spring框架

Spring是一个开源的轻量级的控制反转(IoC)和面向切面(AOP)的容器框架。它是为了解决企业应用开发的复杂性而创建的。Spring使用基本的JavaBean来完成以前只可能由EJB完成的事情。Spring的用途不仅限于服务器端的开发。从简单性、可测试性和松耦合的角度而言，任何Java应用都可以从Spring中受益。

Spring框架的主要特点有：

(1) 轻量：从大小与开销两方面而言Spring都是轻量的。完整的Spring框架可以在一个大小只有1MB多的JAR文件里发布。并且Spring所需的处理开销也是微不足道的。此外，Spring是非侵入式的：Spring应用中的对象不依赖于Spring的特定类。

(2) 控制反转：Spring通过一种称作控制反转（IoC）的技术促进了松耦合。当应用了IoC，一个对象依赖的其它对象会通过被动的方式传递进来，而不是这个对象自己创建或者查找依赖对象。你可以认为IoC与JNDI相反——不是对象从容器中查找依赖，而是容器在对象初始化时不等对象请求就主动将依赖传递给它。

(3) 面向切面：Spring提供了面向切面编程的丰富支持，允许通过分离应用的业务逻辑与系统级服务（例如审计（auditing）和事务（transaction）管理）进行内聚性的开发。应用对象只实现它们应该完成的业务逻辑。它们并不负责其它的系统级关注点，例如日志、事务支持以及安全性验证。

2.5.3 Spring MVC

Spring MVC是一种基于Java实现MVC设计模式的请求驱动类型的轻量级Web框架，即使用了MVC架构模式的思想，将web层进行解耦，基于请求-响应模型帮助我们简化日常web系统的开发。其前端控制器是DispatcherServlet主要用于控制流程；应用控制器为Handler Mapping-处理器映射器进行处理器管理和View Resolver-视图解析器进行视图的解析；页面控制器/动作跳转处理器为Controller接口；支持本地化解析、文件上传等；提供了十分灵活的数据校验、格式化和数据绑定机制；采用约定优于配置的契约式编程方式。

Spring MVC典型的执行流程大致如下：

1、首先浏览器发送请求给前端控制器DispatcherServlet，DispatcherSerlvet根据请求信息，基于一定的原则选择合适的控制器进行处理并把 请求委托给它。

2、页面控制器接收到请求之后进行功能处理，首先需要收集、绑定请求参数到一个对象(命令对象)，并进行验证，然后将该对象委托给业务对象进行处理(service层)；业务对象处理之后控制器将返回一个ModelAndView(模型数据和逻辑视图名)；

3、DispatcherServlet根据返回的逻辑视图名，选择合适的视图进行渲染(界面展示、资源加载)，并把模型数据传入以便视图渲染。

4、前端控制器将响应返回个客户端浏览器。

2.5.4 Spring Boot

Spring Boot是由Pivotal团队提供的全新框架，其设计目的是用来简化新Spring应用的初始搭建以及开发过程。该框架使用了特定的方式来进行配置，从而使开发人员不再需要定义样板化的配置。

使用Spring Boot构建Spring应用有如下优点：

1、为所有Spring开发提供一个从根本上更快，且随处可得的入门体验。

2、开箱即用，但通过不采用默认设置可以快速摆脱这种方式。

3、提供一系列大型项目常用的非功能性特征，比如：内嵌服务器，安全，指标，健康检测，外部化配置。

4、没有代码生成，也不需要XML配置。

2.6 MySQL及 Java ORM框架MyBatis

2.6.1 MySQL

MySQL是一款开源的快速易用的 RDBMS，很多企业（不分规模大小）都在使用它来构建自己的数据库。它之所以非常流行，原因在于具备以下这些优点：

1、基于开源许可发布，无需付费即可使用。

2、自身的功能非常强大，足以匹敌绝大多数商业数据库。

3、使用业内所熟悉的标准SQL数据库语言。

4、可运行于多个操作系统，支持多种编程语言。

2.6.2 MyBatis

MyBatis 是支持定制化 SQL、存储过程以及高级映射的优秀的持久层框架。MyBatis 避免了几乎所有的 JDBC 代码和手动设置参数以及获取结果集。MyBatis 可以对配置和原生Map使用简单的 XML 或注解，将接口和 Java 的 POJOs(Plain Old Java Objects,普通的 Java对象)映射成数据库中的记录。

MyBatis简单易学，没有任何第三方依赖，最简单安装只要两个jar文件和配置几个sql映射文件易于学习，易于使用，通过文档和源代码，可以比较完全的掌握它的设计思路和实现。其次该框架十分的灵活，mybatis不会对应用程序或者数据库的现有设计强加任何影响。 sql写在xml里，便于统一管理和优化。通过sql基本上可以实现我们不使用数据访问框架可以实现的所有功能。

2.7 Solr搜索引擎

Solr是一个高性能，采用Java5开发，基于Lucene的全文搜索服务器。同时对其进行了扩展，提供了比Lucene更为丰富的查询语言，同时实现了可配置、可扩展并对查询性能进行了优化，并且提供了一个完善的功能管理界面，是一款非常优秀的全文搜索引擎。

其文档通过Http利用XML 加到一个搜索集合中。查询该集合也是通过http收到一个XML/JSON响应来实现。它的主要特性包括：高效、灵活的缓存功能，垂直搜索功能，高亮显示搜索结果，通过索引复制来提高可用性，提供一套强大Data Schema来定义字段，类型和设置文本分析，提供基于Web的管理界面等。

2.8 协同过滤

协同过滤（英语：Collaborative Filtering），利用某兴趣相投、拥有共同经验之群体的喜好来推荐用户感兴趣的信息，个人通过合作的机制给予信息相当程度的回应（如评分）并记录下来以达到过滤的目的进而帮助别人筛选信息，回应不一定局限于特别感兴趣的，特别不感兴趣信息的纪录也相当重要。

协同过滤推荐算法通过在用户活动中寻找特定模式来为用户产生有效推荐。它依赖于系统中用户的惯用数据，例如通过用户对其阅读过书籍的评价可以推断出用户的阅读偏好。这种算法的核心思想就是：如果两个用户对于一些项的评分相似程度较高，那么一个用户对于一个新项的评分很有可能类似于另一个用户。值得注意的是，协同过滤推荐的时候不依赖于项的任何附加信息，例如描述、元数据等，或者用户的任何附加信息，例如喜好、人口统计相关数据等。

2.9 Websocket

WebSocket一种在单个 TCP 连接上进行全双工通讯的协议。它使得客户端和服务器之间的数据交换变得更加简单，允许服务端主动向客户端推送数据。在 WebSocket API 中，浏览器和服务器只需要完成一次握手，两者之间就直接可以建立持久性的连接，并进行双向数据传输。

这种技术要达到的目的是让用户不需要刷新浏览器就可以获得实时更新。它有着广泛的应用场景，比如在线聊天室、在线客服系统、评论系统、WebIM等。

# 系统需求分析

本设计的房产信息系统主要需要处理的需求包括房源、楼讯等业务。在各项需求中，有着不

同的具体业务操作。需要通过对需求的详细分析，设计出系统功能用例，明确系统应当实现的具体业务功能。

## 3.1 总体业务分析

房产信息系统的主要功能为信息的展示及发布，具体的描述如下图所示

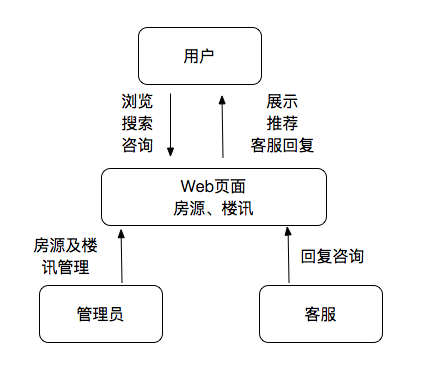


图3.1

根据图3.1得出总体业务逻辑中包括以下几个部分

(1) Web页面与用户的交互：用户可以通过Web页面浏览房源及楼讯，同时可以根据关键字或者既定房源条件进行搜索房源。根据用户的浏览习惯，为用户推荐合适的房源。在房源展示中提供类似的房源供用户浏览。在房源详情页，用户可以通过Web页面与客服进行实时资讯交流。

(2) Web页面与管理员的交互：管理员可以通过Web页面进行房源及楼讯的管理，包括房源发布、更新、删除、设置推荐，楼讯的发布、更新、删除。

(3) Web页面与客服的交互：客服人员可通过Web页面回答用户关于房源的实时咨询。

3.2 系统用例分析

根据总体业务逻辑的分析，再细分各个业务的具体用例，得出更加细粒度的需求分析结果。

3.2.1 参与者划分

把系统使用者划分为角色不同的参与者，分析不同参与者的功能用例，明确不同类别的使用者对系统功能的需求。

根据总体业务分析的结果，把系统的使用者细分为以下几类参与者

(1) 用户

这里用户的定义是有住房需求的，能与房产公司达成交易的潜在客户。用户通过房产信息系统获取自己想要的房源信息及楼讯。

(2) 管理员

管理员属于房产公司员工，通过房产信息系统管理公司的房源及市场相关的楼讯。

(3) 客服

客服可以是房产公司客服或者是外包客服，通过房产信息系统满足用户的实时咨询。

3.2.2 参与者用例分析

基于上节参与者的划分，给出各类参与者的具体用例，每一个用例代表了一个具体的系统应该实现的功能。

1. 用户用例分析

根据用户对总体业务的需求，分析得出图3.2的用例分析结果

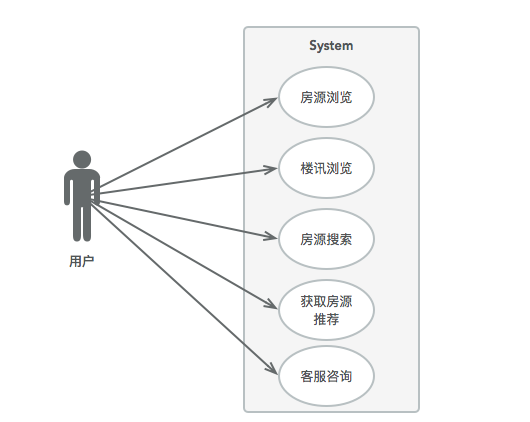


图3.2

房源浏览用例：

用户能够浏览到房源列表，房源详情页面。房源列表展示多个房源，每一项包含该房源的关键信息。房源详情页面包含所有关于该房源的信息，并展示与该房源相似的房源链接。

楼讯浏览用例：

用户能够浏览楼讯列表及楼讯详情页面。楼讯列表展示多条楼讯，每一项包含该楼讯的关键信息。楼讯详情页面显示该楼讯的详细信息。

房源搜索用例：

用户能够对房源进行搜索，搜索条件为模糊关键词匹配。同时用户还能根据具有房源特征相关的具体选项进行搜索过滤。

获取房源推荐用例：

在用户进行过一些浏览操作之后，系统能根据用户的浏览特点，给出符合用户浏览偏好的房源推荐列表。

客服咨询用例：

用户能够在房源详情页面，通过在线聊天的方式进行针对该房源的客服咨询。

1. 根据管理员对总体业务的需求，分析得出图3.3的用例分析结果

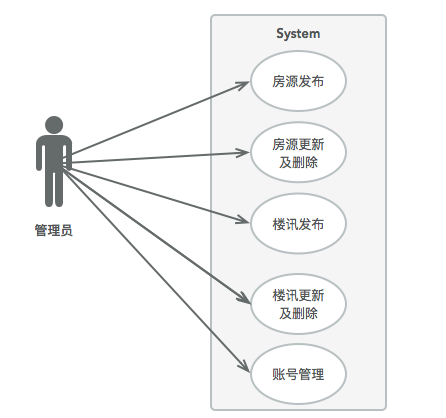


图3.3

房源发布用例：

管理员可以通过系统页面进行房源发布，发布的房源包含展示图片、小区名、位置、户型、价格、面积等必要的房源信息。

房源更新及删除用例：

管理员可以获取自己已发布的房源列表，对这些房源进行更新及删除下架操作。

楼讯发布用例：

管理员可以发布楼讯，发布的楼讯采用富文本编辑器进行编辑，可以插入图片，设置图片布局，设置标题段落等。

楼讯更新及删除用例：

管理员能够获取已发布的楼讯列表，对该列表中的楼讯进行更新及删除操作。

账号管理用例：

管理员可以对系统中的管理员及客服账号进行管理，如账号申请、权限修改等。

(3) 根据客服对总体业务的需求，分析得出图3.5的用例分析结果

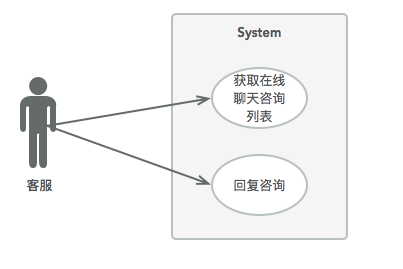


图3.5

获取在线聊天咨询列表用例：

客服登录后可以获取在线聊天的列表，通过该列表可以选择咨询的用户进行回复。

回复咨询用例：

客服可以与用户进行实时的在线聊天咨询沟通，使得用户对房源进行进一步的了解。

1. 对于所有参与者的登录注册用例图如下

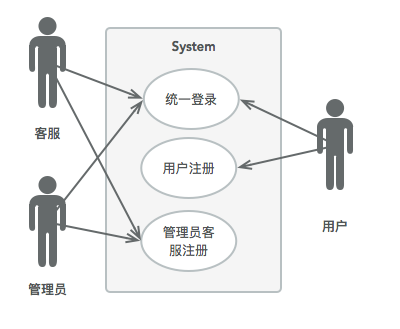


图3.6

统一登录用例：

系统所有的参与都通过一个统一的模块进行登录。

用户注册用例：

用户可以在注册界面直接进行注册。

客服及管理员注册用例：

客服管理员通过管理员的账号管理用例中的账号注册功能进行注册。

4 系统设计

4.1系统总体架构设计

在系统需求明确之后，需要进一步分析，把系统整体分解为更小的子系统和组件，从而形成

不同的逻辑层或服务。同时需要对划分出来的各个功能模块进行设计，如数据库设计，API设计，界面设计，搜索逻辑设计，推荐模型设计等。在设计这些软件模块时，需要满足一些软件设计原则：分离基础服务、业务规则、业务流程，选择合适的工具外化业务规则和业务流程，分离业务组件和技术组件，高类聚，低耦合，业务信息的执行可以分散，但业务信息的管理要尽量集中。

利用好分层的思想，分层的思想，就是每一个层次专注做一件事情。每一个层次都为上层提供服务。每一个层次对于其上层来说，都是可以复用的。分层设计的软件，结构清晰，代码各司其职，能够最大限度地重用代码。

本文设计的房产信息系统，所有者为房地产信息公司，使用者大多数为有住房需求的广大互联网用户，系统最终的目的是促成双方交易，让用户能快速方便地找到需要的住房资源，房产公司也能提高自身的营业效率。围绕这一点，系统架构设计以房源为中心，衍生出房源搜索、房源推荐、房源咨询、相关楼讯浏览等模块。

系统的整体架构如下图4.1所示

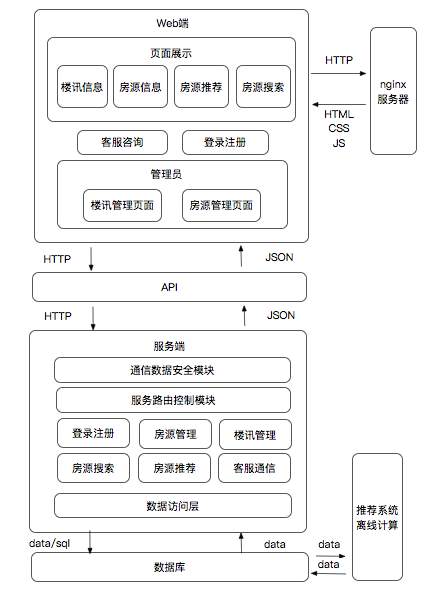


图4.1

前后端交互的过程如下：

用户通过Web端访问前端界面，前端HTML,CSS,JS等静态资源文件使用nginx服务器进行分发管理，Web浏览器首先访问nginx服务器获取静态的页面资源。在静态页面中，由JS控制向服务端发出AJAX请求，获取动态JSON数据。前后端通信均采用HTTP协议。

对总体架构图中的模块说明：

(1) nginx服务器

nginx 是一款支持高并发低延迟的高性能服务器，很合适做静态资源分发处理。前端资源打

包后托管到服务器目录下，web端向nginx服务器请求页面静态资源。划分出nginx服务器层主要有web端能快速显示出界面，可对前端资源进行缓存处理，面对业务的扩大可以选择使用多个nginx服务器实例，具有可扩展性等优点。

(2) Web端

Web端负责与使用者的交互，主要有房源展示、房源搜索、房源推荐、楼讯展示等页面，对

于管理员来说，还具有房源发布、楼讯发布、已发布的房源及楼讯列表等页面模块。Web端通过AJAX向服务端发出HTTP请求获取相关数据。

(3) API层

API层约定了Web端和服务端的通信格式，如通信方法、参数等。通信使用HTTP协议，对于数据的获取一般采用GET方法，对于资源上传及更新一般采用POST方法。通过API层，服务端只需向Web端提供约定的接口，Web端不在意服务端的内部实现方法，实现了服务端与Web端的逻辑解耦，使得系统便于扩展和维护。

1. 服务端及数据库

服务端负责整个系统的数据管理和业务逻辑处理功能，服务端主要可以分为安全检查、服

务路由控制、服务层及数据访问层。

通信数据安全检查：验证访问token，防止恶意调用，提供系统的安全性。

服务路由控制：根据访问API的规则，选择合适的服务进行操作，处理完毕后向API返回JSON数据。

服务层：负责处理具体的业务逻辑，登录注册，房源管理，楼讯管理，客服咨询等系统业务逻辑。

数据访问层：

与数据库进行交互，向服务层提供统一的数据操作接口。

1. 推荐系统离线计算

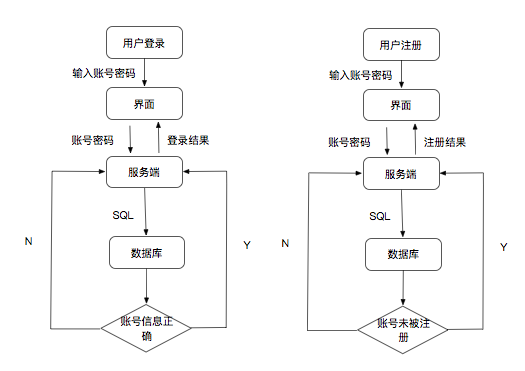
采用离线计算的方式，直接从数据库中提取相关业务写入的信息进行推荐计算，计算的结果

再而保存到数据库供服务层业务使用。

## 

## 4.2 功能模块设计

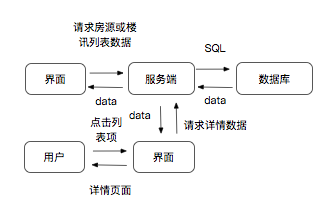
用户登录注册，流程图如下所示：



用户登录时通过Web界面输入账号密码，前端向后端请求登录验证，用户输入的账号密码作为参数通过HTTP请求验证结果。服务端执行向数据库发出SQL，若账号存在且账号和密码相匹配，返回登录成功，否则登录失败。

用户注册时通过Web页面输入账号密码，并且确认密码，前端需验证两次输入的密码是否相同，不同则提示错误，验证通过过向服务端端请求账号注册，服务端查询数据库该账号是否已经存在，不存在则插入新的账号数据，返回注册成功。

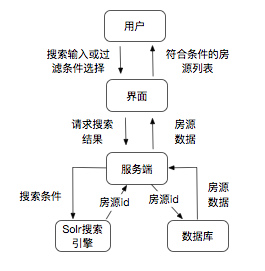
房源及楼讯浏览，流程图如下：



在用户点开系统Web页面时，前端便会分页请求房源及楼讯列表数据，服务端从数据库中取出房源楼讯数据返回前端，界面以列表的方式把这些数据显示出来供用户浏览点击。

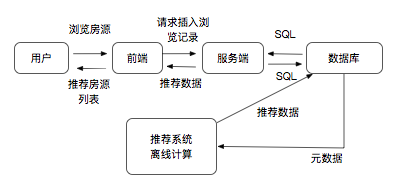
用户点击列表项后，前端根据点击的列表项请求相关的详细数据，服务端从数据库中查出后返回，前端获取数据后渲染出房源或列表详情页面。

房源搜索，处理流程图如下所示：



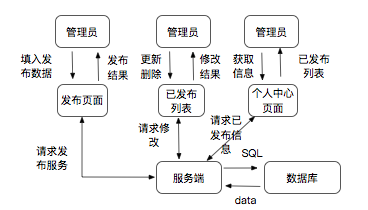
用户搜索通过在前端页面输入关键字或者选择房源特征过滤选项如选择户型、价格区间等进行房源搜索，前端请求搜索服务，搜索条件作为参数，服务端接到请求后请求Solr搜索引擎服务，得到搜索引擎返回的符合搜索条件的id，再通过数据库查询房源id对应的数据返回到前端页面，页面随即渲染出搜索到的房源列表。

房源推荐，处理流程图如下：



房源推荐的过程为：用户浏览房源时，前端便发出AJAX请求服务端新增或更新浏览数据，服务端把这些元数据插入数据库。在推荐系统进行离线计算时，从数据库中获取进行推荐需要的元数据如用户浏览记录填充推荐算法数据集，计算出相关的推荐结果之后把数据存入数据库。在用户登录之后，前端请求针对当前用户的推荐信息，服务端从数据库中查出相关的推荐信息返回到前端页面。

房源及楼讯管理，处理流程如下图所示：



房源及楼讯的管理，主要有发布、更新及删除操作。管理员登录后，在发布页面选择发布房源或者楼讯，随后填写相关信息，发布房源需填写关于房源的业务选项，房源展示图片，户型，价格，交通情况等，发布楼讯则直接在富文本编辑器中编写楼讯文章即可。管理员在个人中心中点击已发的房源或者楼讯，在获取的列表中选择相应的房源或楼讯进行更新或删除操作。操作提交之后前端请求服务端进行房源或楼盘数据的插入、更新及删除。

客服咨询模块设计(待完成)

4.3 数据库设计

根据需求分析、总体架构设计及各功能模块的设计，本文的房产信息系统数据库表有以下几个分类：

1. 房源相关

房源基本信息表、房源图片表、房源标签表、房源推荐表、相似房源表、房源浏览记录表、

标签信息表。

1. 楼讯相关

楼讯表、楼讯内容表。

1. 用户相关

用户信息表。

房源基本信息表包含房源的一些基本信息字段，建表SQL语句如下

CREATE TABLE `house` (

`house\_id` int(11) NOT NULL AUTO\_INCREMENT COMMENT '房源主键',

`name` varchar(45) NOT NULL DEFAULT '' COMMENT '房源小区名称',

`creator\_name` varchar(45) NOT NULL DEFAULT '' COMMENT '房源上传者名称',

`create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP COMMENT '创建 时间',

`price` int(11) NOT NULL DEFAULT '0' COMMENT '价格',

`city\_id` int(11) NOT NULL DEFAULT '0' COMMENT '城市id',

`region\_id` int(11) NOT NULL DEFAULT '0' COMMENT '区域id',

`position` varchar(45) NOT NULL DEFAULT '' COMMENT '门牌位置',

`address` varchar(60) DEFAULT '' COMMENT '房源地理位置',

`area` int(11) NOT NULL DEFAULT '0' COMMENT '面积',

`status` tinyint(4) NOT NULL DEFAULT '1' COMMENT '上架状态',

`des` varchar(200) NOT NULL DEFAULT '' COMMENT '房源相关描述',

`phone` varchar(15) NOT NULL DEFAULT '' COMMENT '联系人电话',

`station\_tag` int(11) NOT NULL DEFAULT '0' COMMENT '地铁站标签',

`route\_tag` int(11) NOT NULL DEFAULT '0' COMMENT '地铁路线标签',

`dec\_tag` int(11) NOT NULL DEFAULT '0' COMMENT '装修类型标签',

`type\_tag` int(11) NOT NULL DEFAULT '0' COMMENT '户型标签',

`region\_tag` int(11) NOT NULL DEFAULT '0' COMMENT '区域标签',

`contact` varchar(15) NOT NULL DEFAULT '' COMMENT '联系人名称',

PRIMARY KEY (`house\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='房源信息表';

房源图片表，存储房源相对应的图片url，一个房源id可对应多个图片url，建表SQL语句如下：

CREATE TABLE `house\_imgs` (

`house\_id` int(11) NOT NULL DEFAULT '0' COMMENT '房源id',

`img` varchar(100) NOT NULL DEFAULT '' COMMENT '图片url',

PRIMARY KEY (`house\_id`,`img`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='房源图片';

标签信息表，存储房源标签基本信息，建表SQL语句如下：

CREATE TABLE `tag` (

`tag\_id` int(11) NOT NULL AUTO\_INCREMENT COMMENT '标签主键',

`name` varchar(20) NOT NULL DEFAULT '' COMMENT '标签名',

`status` tinyint(4) NOT NULL DEFAULT '1' COMMENT '标签使用状态',

PRIMARY KEY (`tag\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='标签信息表';

房源标签表，该表为房源表和标签表的一对多的关系表，一个房源可以有多个标签，建表SQL语句如下：

CREATE TABLE `house\_tags` (

`house\_id` int(11) NOT NULL DEFAULT '0' COMMENT '房源id',

`tag\_id` int(11) NOT NULL DEFAULT '0' COMMENT '标签id',

PRIMARY KEY (`house\_id`,`tag\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='房源标签关系表';

房源推荐表，存储推荐系统为用户离线计算出的推荐房源结果，一个用户id对应多个推荐房源id，

建表SQL语句如下：

CREATE TABLE `recommend\_house` (

`id` int(11) NOT NULL AUTO\_INCREMENT COMMENT '主键',

`user\_id` int(11) NOT NULL DEFAULT '0' COMMENT '用户id',

`house\_id` int(11) NOT NULL DEFAULT '0' COMMENT '房源id',

`create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP COMMENT '记录生成时间',

PRIMARY KEY (`id`),

KEY `recommend\_house\_user\_id\_index` (`user\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='房源推荐';

相似房源表，存储推荐系统根据用户浏览情况离线计算出的相似房源，每个房源id对应多个相似房源id，建表SQL语句如下：

CREATE TABLE `similar\_house` (

`id` int(11) NOT NULL AUTO\_INCREMENT COMMENT '主键',

`house\_id` int(11) NOT NULL DEFAULT '0' COMMENT '房源id',

`similar\_id` int(11) NOT NULL DEFAULT '0' COMMENT '相似房源id',

`create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP COMMENT '记录生成时间',

PRIMARY KEY (`id`),

KEY `similar\_house\_house\_id\_index` (`house\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='相似房源表';

用户浏览记录表，存储用户浏览相关房源的次数，提供给推荐系统作为推荐元数据。每个用户对应多个房源，并且通过组合主键约束用户对应的某个房源浏览记录只有一条，节约存储空间。建表SQL语句如下：

CREATE TABLE `browse\_count` (

`user\_id` int(11) NOT NULL DEFAULT '0' COMMENT '用户id',

`house\_id` int(11) NOT NULL DEFAULT '0' COMMENT '房源id',

`count` int(11) NOT NULL DEFAULT '1' COMMENT '浏览次数',

PRIMARY KEY (`user\_id`,`house\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='用户浏览记录表';

楼讯基本信息表，存储楼讯的基本信息，建表SQL语句如下：

CREATE TABLE `news` (

`id` int(11) NOT NULL AUTO\_INCREMENT COMMENT '主键',

`create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP COMMENT '创建时间',

`title` varchar(30) NOT NULL DEFAULT '' COMMENT '标题',

`abstract` varchar(100) NOT NULL DEFAULT '' COMMENT '楼讯摘要',

`title\_pic` varchar(100) NOT NULL DEFAULT '' COMMENT '展示图片',

`creator\_name` varchar(20) NOT NULL DEFAULT '' COMMENT '创建者名称',

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='楼讯基本信息表';

楼讯内容表，由于楼讯的内容所占存储空间比较大，且只有用户点击楼讯详情时才使用，所以单独建一张内容表使用大字段存储楼讯内容。建表SQL语句如下：

CREATE TABLE `news\_content` (

`news\_id` int(11) NOT NULL DEFAULT '0' COMMENT '楼讯id',

`content` longtext COMMENT '楼讯内容',

PRIMARY KEY (`news\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='楼讯内容';

用户基本信息表，存储用户账号密码角色权限等基本信息，建表SQL语句如下：

CREATE TABLE `user` (

`id` int(11) NOT NULL AUTO\_INCREMENT COMMENT '用户id',

`name` varchar(15) NOT NULL DEFAULT '' COMMENT '用户名 唯一',

`password` varchar(45) NOT NULL DEFAULT '' COMMENT '密码',

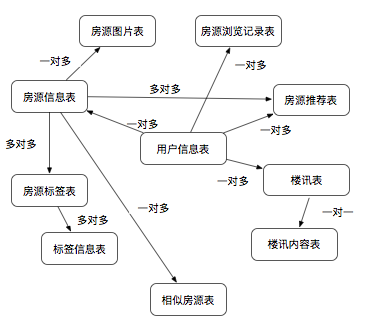
`role\_id` int(11) NOT NULL DEFAULT '1' COMMENT '角色id',

PRIMARY KEY (`id`),

UNIQUE KEY `name\_UNIQUE` (`name`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='用户基本信息表';

各表的对应关系如下图所示：



4.4 UI设计

借助于Reactjs组件化的可复用性，系统界面UI使用基于Reactjs的UI组件库Ant Design进行实现，采用了Ant Design的布局，卡片，轮播图，列表等组件。

4.5 搜索设计

在上传房源时，房源数据插入数据库之后，将房源id及房源数据中的可以用于检索的条件作为索引插入Solr搜索引擎中，为用户提供搜索功能。同理，在更新或删除房源数据时，也同步更新Solr索引。

4.5 推荐系统设计

推荐系统采用离线计算的方式执行，由于使用初期数据体量不大，可把数据载入内存中使用Python脚本进行推荐计算，待数据体量变得庞大时，可考虑使用分布式计算框架进行推荐计算(详见总结与展望一章)。

推荐算法采用基于SVD的协同过滤算法，用户浏览数据作为推荐计算的元数据，包括用户id，浏览的房源id，浏览次数，由此建立推荐数据集矩阵。

业界已经有许多优秀的Python推荐算法实现类库，本设计采用了python-recsys，使用其SVD算法API进行推荐计算。

# 

# 5 系统实现

## 5.1 Web端首页

待完善

5.2 登录注册模块

用户登录模块如下图所示，登录时填入账号和密码，UI界面以弹窗的方式显示

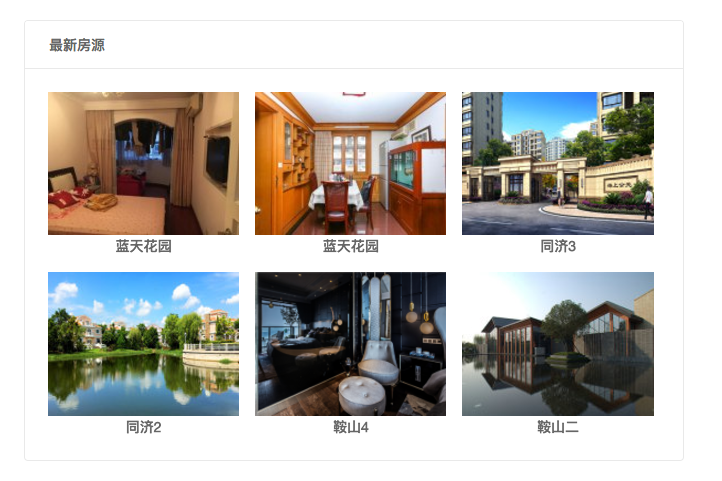


用户注册模块如下图所示，注册时需要输入两次相同的密码，以防止用户误输入



5.3 房源浏览模块(待完善)

系统最新房源页面如下图所示，以轮播图的方式展示最新发布的18个房源的首页图片及房源名称，每一轮中有6个房源显示，点击房源图片可进入该房源详情页面



房源推荐页面(待完善)

房源详情页面(待完善)

5.4 楼讯浏览模块(待完善)

系统发布的楼讯列表如下图所示，分页显示，点击加载更多可以查看下一页的楼讯，点击楼讯项可进入该楼讯的详情页面





楼讯详情页面(待完善)

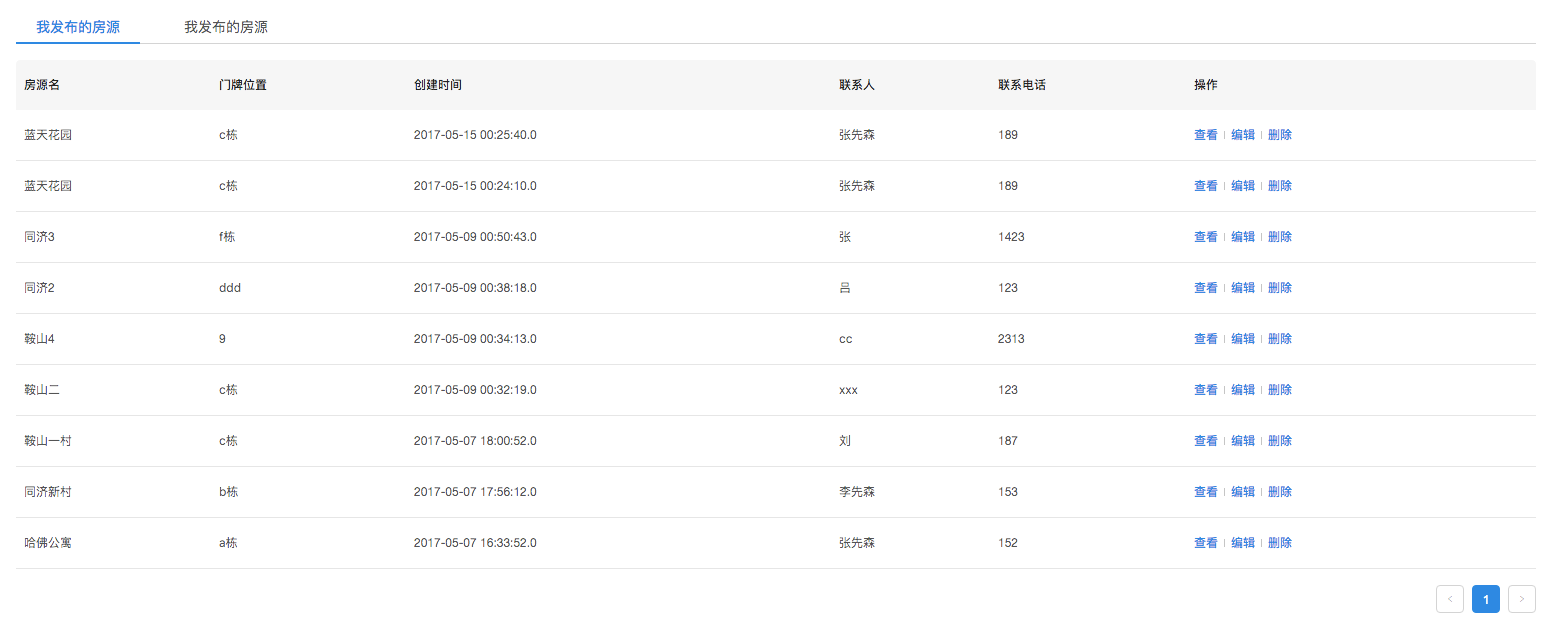
5.5房源管理模块

管理员登录个人中心后，点击发布房源，填写相关房源选项(选项待丰富),点击提交即可发布房源，发布页面如下图所示



图5.5

已发布房源列表，管理员登录个人中心后可以看到自己发布的房源信息，数据分页显示，点击相应操作即可进行房源的编辑及删除，页面如下图所示

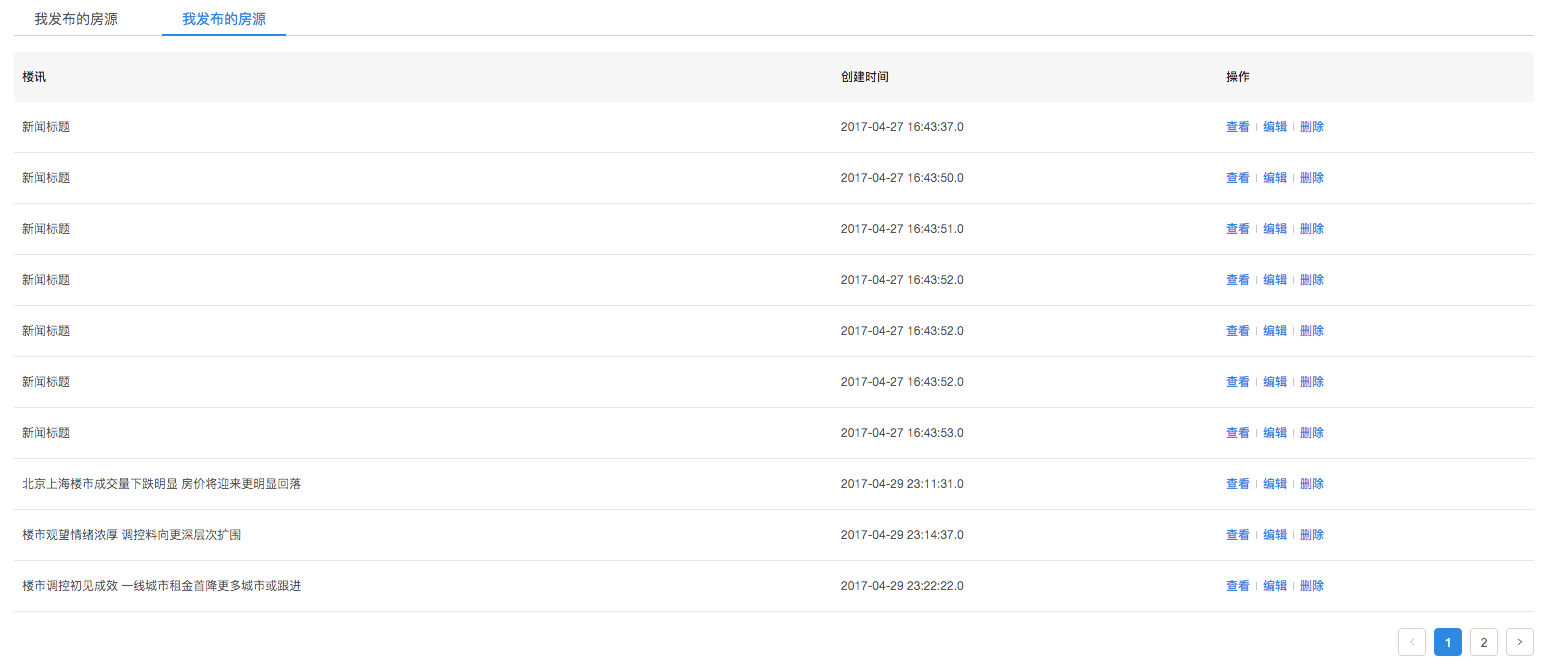


5.6 楼讯管理

楼讯上传，管理员登录个人中心后可点击发布楼讯，在富文本框中可以进行控制内容格式、插入图片等操作，编辑楼讯后点击提交即可发布，页面如下图所示

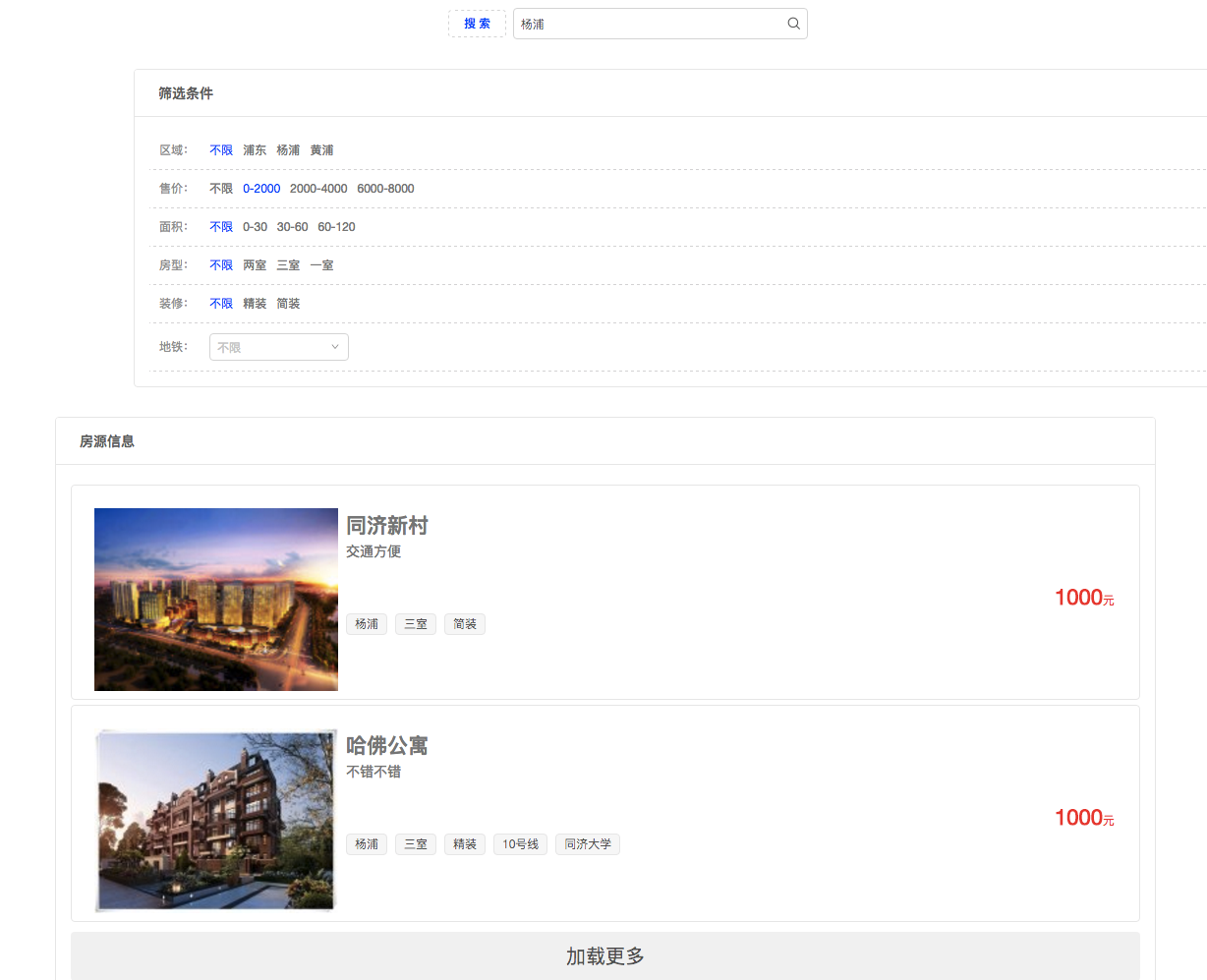


已发布楼讯列表，管理员登录用户中心后点开发布的楼讯选项卡，楼讯数据分页显示，点击操作选项进行更新删除操作，页面如下图所示



5.7 房源搜索

房源搜索可以通过输入关键字或选项过滤进行搜索，搜索结果由Solr搜索引擎得出，以房源列表的方式进行分页显示，点击房源项可进入该房源的详情页面，搜索页面如下图所示



5.8客服咨询实现(待完成)

6 系统关键技术

6.1 Solr搜索引擎（排序结果待完善）

(1)索引配置：

在使用搜索引擎时，我们需要根据业务逻辑配置索引，配置代码如下



配置中<field>为索引项

area项为面积，price项为房源价格，只建立索引但不存储，为int型，搜索引擎允许我们对int型的数据进行区间搜索。

dec项为装修类型，region项为区域，route项为地铁线路，station项为地铁站名称，type项为房源户型，只建立索引但不存储，为string型。

des项为房源描述，name项为房源名称，只建立索引但不存储，为text\_smartcn类型，对于该类型，搜索引擎会对该项进行中文分词检索。

id项为房源id，相当于索引主键，必须且唯一，建立索引并对其值进行存储，id需要作为搜索结果返回。

<solrQueryParser>项defaultOperator属性设置为AND，表示搜索条件使用AND进行过滤。

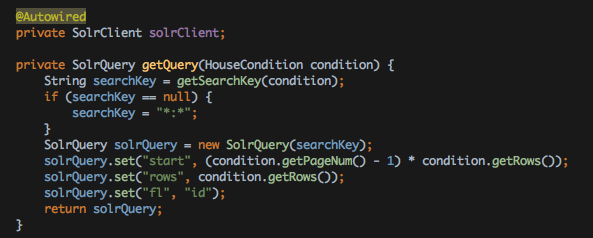
(2)在Java代码中建立索引及搜索：

Solr提供了SolrJ这个Java类库进行搜索引擎相关的操作，下图的代码向搜索引擎中插入一个搜索文档。



document为solr文档对象，向文档中添加上述配置的索引项数据，随后使用solrClient对象向solr实例进行提交即可。

下图代码演示了获取搜索结果对象的过程



searchKey是一个满足搜索格式的String对象，把这个String作为搜索条件，并设置分页数量，只返回id项，构建出SolrQuery对象，之后便可使用该对象作为参数，调用solr搜索服务获取搜索结果对象。

6.2 推荐引擎

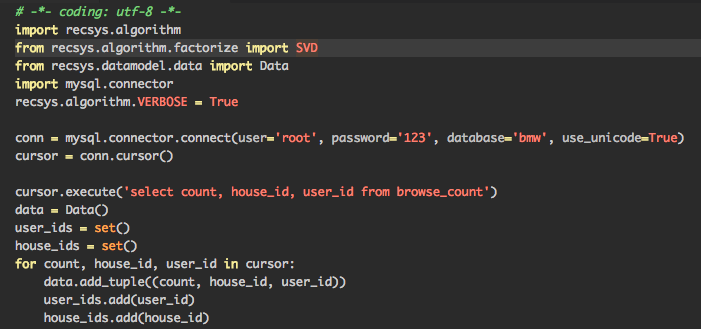
(1)基于SVD的协同过滤算法介绍(待完善)

(2)基于Python的推荐引擎脚本

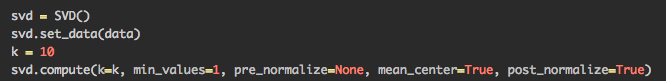
Python版本为2.7

使用python-recsys类库执行推荐算法

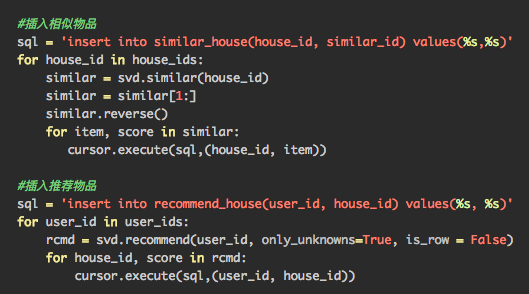
使用mysql-connector作为mysql连接



上述代码从数据库中获取到了用户浏览记录数据，并把house\_id作为row，user\_id作为column，count作为value添加到data数据集中构建数据矩阵。



上述代码创建了一个SVD对象，同时填入数据集，执行SVD算法



上述代码的关键是svd.similar() 及svd.recommend() 这个两个函数。

svd.similar(house\_id)函数返回一个与该house\_id相似的house\_id列表，获取到列表之后便把其中数据插入到相似房源表中。

svd.recommend(user\_id)函数返回一个位该user\_id推荐的house\_id列表，获取到列表后把其中的数据插入到推荐房源表中。

7 总结与展望(待完成)

# 参考文献(待完成)

# 

# 谢 辞(待完成)