**《城市房价数据分析系统》**

**项目概要设计**

**V1.0**

**版 本 历 史**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本/状态 | 作者 | 参与者 | 日期 | 备注 |
|  |  |  |  |  |
|  |  |  |  |  |

**目 录**

**第一部分 引言**

**第二部分 项目概述**

**第三部分 设计约束**

**第四部分 城市房价数据分析系统方案设计**

**第五部分 E-R实体设计**

**第六部分 总体设计**

**第七部分 用户界面设计**

**第八部分 运行环境和部署**

**第一部分 引言**

**一、编写目的**

编写本文的主要目的是把需求分析得到的用例模型转换为软件结构和数据结构。设计软件结构的具体任务是：将一个复杂系统按功能进行模块划分、建立模块的层次结构及调用关系、确定模块间的接口及人机界面等。数据结构设计包括数据特征的描述、确定数据的结构特性、以及数据库的设计。

本设计是指导详细设计和项目实施的重要指导性文件，也是进行系统集成测试和重要依据。

**二、读者对象**

该文档的读者为用户代表、软件分析人员、开发管理人员和测试人员。

**三、术语与缩写解释**

**1、网络爬虫**

网络爬虫，是一种按照一定的规则，自动地抓取万维网信息的程序或者脚本。又称为网页蜘蛛，网络机器人，在FOAF社区中间，更经常被称为网页追逐者，另外一些不常使用的名字还有蚂蚁、自动索引、模拟程序或者蠕虫。

网络爬虫按照系统结构和实现技术，大致可以分为以下几种类型：通用网络爬虫（General Purpose Web Crawler）、聚焦网络爬虫（Focused Web Crawler）、增量式网络爬虫（Incremental Web Crawler）、深层网络爬虫（Deep Web Crawler）。 实际的网络爬虫系统通常是几种爬虫技术相结合实现的。

重要步骤有：

（1）  网页搜索策略：可以分为深度优先、广度优先和最佳优先三种。深度优先在很多情况下会导致爬虫的陷入(trapped)问题，目前常见的是广度优先和最佳优先方法。

（2）  抓取目标分类：对抓取目标的描述可分为基于目标网页特征、基于目标数据模式和基于领域概念3种。

（3）  网页分析算法：可以归纳为基于网络拓扑、基于网页内容和基于用户访问行为三种类型。

**2、大数据**

大数据，是指无法在一定时间范围内用常规软件工具进行捕捉、管理和处理的数据集合；是需要新处理模式才能具有更强的决策力、洞察发现力和流程优化能力的海量、高增长率和多样化的信息资产；也是一种规模大到在获取、存储、管理、分析方面大大超出了传统数据库软件工具能力范围的数据集合，具有海量的数据规模、快速的数据流转、多样的数据类型和价值密度低四大特征。

**3、人工智能**

人工智能是计算机科学的一个分支，它企图了解智能的实质，并生产出一种新的能以人类智能相似的方式做出反应的智能机器，该领域的研究包括机器人、语言识别、图像识别、自然语言处理和专家系统等。人工智能从诞生以来，理论和技术日益成熟，应用领域也不断扩大，可以设想，未来人工智能带来的科技产品，将会是人类智慧的“容器”。人工智能可以对人的意识、思维的信息过程的模拟。人工智能不是人的智能，但能像人那样思考、也可能超过人的智能。

**4、中国城市房价**

**房价**（房地产价格）是指建筑物连同其占用土地在特定时间段内房产的市场价值，即房价：土地价格+建筑物价格，是房地产经济运行和资源配置最重要的调节机制。房价的价格定位由多种因素构成。

房价有三种存在形式：要约价格、成交价格、申报价格。要约价格随市场行情随时波动，成交价格是供需双方博弈后的结果，申报价格是向政府管理部门登记的结果。而广泛使用的房产成交价格往往是申报价格，而申报价格质量非常差。

**5、房价数据分析系统**

**HPDAS**是房价数据分析系统(Housing price data analysis system)的缩写，房价数据分析系统是由信息采集、数据存储、信息分析、可视化和房价预测等模块组成，通过大数据处理系统，构建数据仓库，实现房价大数据分析的管理系统。从网页上抓取数据，清洗储存之后，通过可视化技术展示出来，最后通过合适的人工智能算法和数据预处理与特征处理方法，预测房价趋势。用户可以通过页面查看所需要的房价信息以及房价走势。

**四、参考资料**

《系统需求分析》

**第二部分 项目概述**

**一、项目描述**

中国城市房价的变化与变迁都会勾动亿万居民的心弦，而掌握城市房价的走向与趋势，也成为了诸多用户的需求与需要。城市房价数据分析系统基于各大城市的房价，提供了房价可视化，房价的查询功能，房价的横向对比和纵向对比，预测房价的走向和趋势。城市房价数据分析系统能够挖掘获取网络上有关经济、政策的信息，并通过这些信息的倾向实现房价预测功能。

房价数据分析系统是由信息采集、数据存储、信息分析、可视化和房价预测等模块组成，通过大数据处理系统，构建数据仓库，实现房价大数据分析的管理系统。从网页上抓取数据，清洗储存之后，通过可视化技术展示出来，最后通过合适的人工智能算法和数据预处理与特征处理方法，预测房价趋势。用户可以通过页面查看所需要的房价信息以及房价走势。

**二、项目功能描述**

用户业务主要包含数据部分和展示部分。

数据部分包含五个主要业务：房价获取、房价处理、房价存储、房价政策、房价分析。

展示部分包含五个主要业务：用户系统、管理员系统、房价展示、房价对比、房价预测。

**1、房价获取**

房价获取主要是获取以往几年与以后的实时房价信息。

任务信息包括：

1)    具体日期

2)    平均房价

3)    行政区域

**2、房价处理**

       房价处理是将获取的数据进行数据处理，由于数据量极为庞大，而且爬取的信息相当杂乱无章，所以需要良好的预处理，才能再将房价数据存储和利用，理想的状态是只保留所需要的几种信息。

**3、房价存储**

房价存储是是将处理好的数据存入房价数据库。

数据库包括如下信息：

1)具体日期

2)平均房价

3)行政区域

然后根据实际情况建立合适的数据库。

**4、房价政策**

房价政策是获取行政区域所实行的房价政策，用于辅助房价分析。

房价政策主要包括如下信息

1)      政策实行时间

2)      行政区域

3)      政策内容

**5、房价分析**

房价分析是按数据库内的房价数据与对应房价政策，经过人工智能的处理之后得出相对合理的分析结果：

包括如下分析结果：

1)    行政区域

2)    预测房价

3)    合理建议

**6、用户系统**

用户系统包括用户数据库、用户注册、用户登录功能。

用户数据库用以存储用户账号信息，并在登录时交互确认登录信息，注册时交互新增用户信息。

用户注册用于注册新账号。

用户登录用于登录账号。

**7、管理员系统**

管理员系统包括管理员登录、系统管理。

管理员登录是登录管理员系统。

系统管理是给予管理员一定权限，对房价数据分析系统进行一定的管理。

**8、房价展示**

房价展示是在索引之后，展示所需的房价信息

包括如下展示信息：

1)    行政区域

2)    房价折线图，随时间分布

3)    房价饼状图，随房价段位分布

**9、房价对比**

房价对比是按照时间段对比两个行政区域的房价：

包括如下对比信息：

1)行政区域

2)时间段

3)平均房价

点击对比后结果以表格形式呈现，最好以拖动的表格样式，并能支持多种条件的展示。

**10、房价预测**

房价预测是按照房价分析结果展示：

包括如下展示信息：

1)    行政区域

2)    房价现行政策

3)    预测房价

4)    合理建议

点击预测后结果以表格形式呈现。

**第三部分 设计约束**

**一、需求约束**

**1、本系统应当遵循的技术标准**

数据命名的规则遵循驼峰式命名法；

**2、软、硬件环境标准**

本系统采用Vue+Flask，数据库采用MySQL。

**3、接口/协议标准**

本系统通过SOAP协议实现智能移动终端和服务器之间的数据通信。

**4、用户界面标准**

按网页设计标准，并进行界面全部美工优化。

**5、软件质量**

1）正确性

系统必须交易能够被正确处理；

2）健壮性

系统应能够7\*24小时无故障运行；

3）效率性

系统可以支持100个终端同时发起业务，处理业务的时间不超过10秒钟；

4）易用性

界面应采用图形化操作方式，便于业务人员操作；

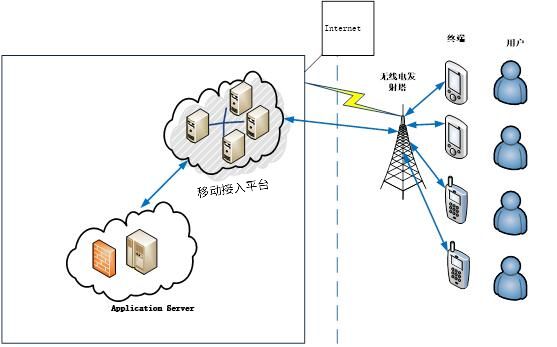
5）安全性

报文中的关键数据域以密文的方式传输；

6）可扩展性

应该充分考虑到将来交易的修改或增加，避免需求变更时大规模修改程序。

7）网络体系结构



注：网络结构说明

A）终端

表示各类能够使用浏览器的智能终端。

B）用户

操作终端的使用用户。

C）无线电发射塔

指各类网络。

D）移动接入平台

通过开放对外的Web Service端口，可以来访问内网中的各种应用。

E）Application Server

各类连入移动接入平台的内部应用。

**二、隐含约束**

1）用户具有基本的业务技能和基本的电脑或手机知识，对我们提供的操作界面应保证他们经过简单培训后无障碍的操作；

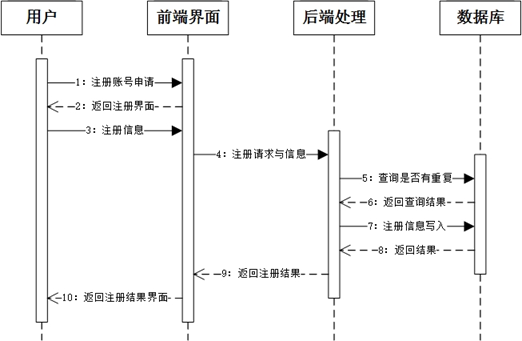
2）网页可以流畅运行在HTML4.0版本以上的浏览器中；

3）应该把有可能变动的参数存放到配置文件或数据库中，保证修改参数的灵活性。

**第四部分 城市房价数据分析系统方案设计**

**一、城市房价数据分析系统相关业务流程**

**1、用户注册模块**



用户注册模块包括：

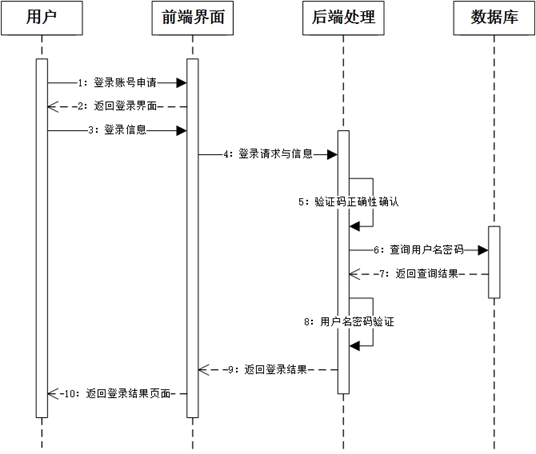
l  用户输入注册信息

l  后端调取数据库中信息进行验证是否合法

l  后端进行注册处理并将用户信息写入数据库

l  成功后创建账号并将信息返回给用户

**2、用户登录模块**



用户登录模块包括：

l  用户输入登录信息

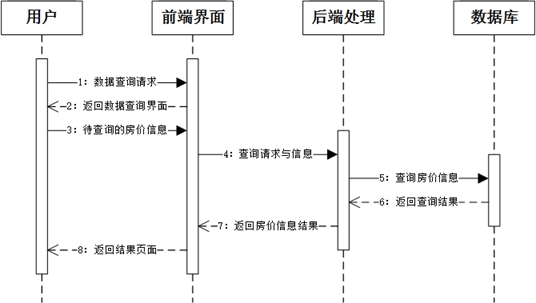
l  后端进行验证码正确性确认

l  验证成功后后端向数据库查询用户名和密码

l  后端进行用户名和密码验证

l  验证成功后返回登录结果给用户

**3、房价数据查询模块**



房价数据查询模块包括：

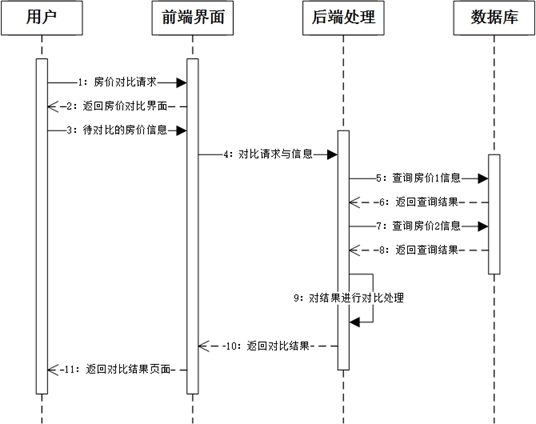
l  用户输入查询信息

l  后端得到查询信息后向数据库发送查询请求

l  数据库返回查询结果给后端

l  后端将查询结果返回给前端并展示给用户

**4、房价对比模块**



房价对比模块包括：

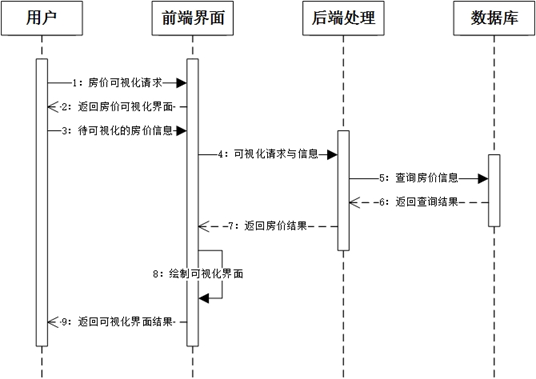
l  用户发送房价对比请求

l  后端根据请求向数据库两次查询房价信息

l  后端根据查询结果进行结果对比处理

l  后端将结果返回给前端并展示给用户

**5、房价可视化模块**



房价可视化模块包括：

l  用户发送房价可视化请求

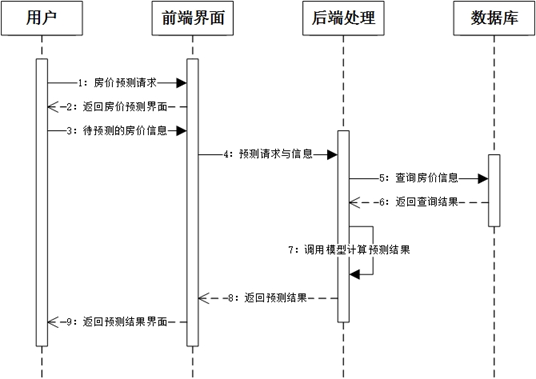
l  后端根据信息向数据库中查询房价信息

l  后端将查询结果发送给前端

l  前端根据查询结果绘制房价可视化界面

l  前端将绘制完毕的可视化界面展示给用户

**6、房价预测模块**



房价预测模块包括：

l  用户发送房价预测请求

l  后端得到请求后向数据库中查询对应房价的数据

l  后端根据查询结果并调用模型进行房价预测

l  后端将预测结果发送给前端并展示给用户

**二、业务功能概要结构**

**1.注册登录模块**



**1.1 用户注册**

1、初始化用户注册界面

2、用户向其中输入注册信息

3、用户根据界面验证码输入验证码

4、用户点击注册按钮

5、系统检查信息合法性以及验证码正确性

6、系统调取数据库中账号信息检测是否重复

7、系统检测成功后向数据库中写入新账号信息

8、系统显示注册成功界面

**1.2 用户登录**

1、初始化用户登录界面

2、用户向其中输入登录信息

3、用户根据界面验证码输入验证码

4、用户点击登录按钮

5、系统检验验证码正确性

6、系统根据账号查询数据库中对应密码

7、系统将对应密码和输入密码进行比对

8、系统检测成功后显示登录成功界面

**1.3 验证码验证**

1、系统生成验证码图片和对应图片的字符

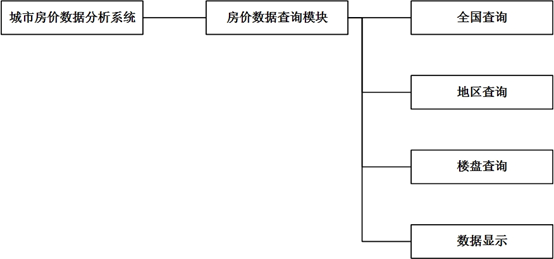
2、系统将图片发送给前端界面显示

3、系统接收到用户所填写的验证码

4、系统将用户填写的验证码和生成的字符进行比对

5、系统检测后返回验证结果

**2.房价数据查询模块**



**2.1 全国查询**

1、初始化查询界面

2、用户点击全国查看按钮

3、系统查询全国的各省份房价数据

4、数据库返回房价数据

5、前端界面显示

**2.2 地区查询**

1、初始化查询界面

2、用户选择地区

3、用户点击查询按钮

4、系统查询选定地区的房价数据

5、数据库返回房价数据

6、前端界面显示

**2.3 楼盘查询**

1、初始化查询界面

2、用户选择楼盘

3、用户点击查询按钮

4、系统查询选定楼盘的房价数据

5、数据库返回房价数据

6、前端界面显示

**2.4 数据显示**

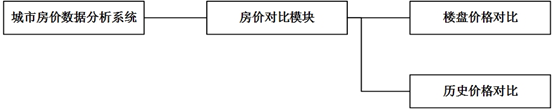
1、接收到从数据库中传送的数据

2、将数据发送给前端

3、前端接受数据

4、前端显示数据

**3.房价对比模块**



**3.1 楼盘价格对比**

1、初始化对比界面

2、用户选择楼盘1

3、用户选择楼盘2

4、系统向数据库查询楼盘1和楼盘2的房价数据

5、系统处理两个楼盘的房价数据

6、系统显示两个楼盘的对比结果

**3.2 历史价格对比**

1、初始化对比界面

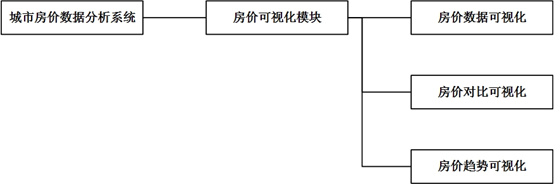
2、用户选择楼盘

3、系统向数据库查询楼盘的历史房价数据

4、系统处理历史房价数据

5、系统显示历史价格的对比结果

**4.房价可视化模块**



**4.1 房价数据可视化**

1、初始化可视化界面

2、用户选择房价数据

3、系统向数据库查询楼盘的房价数据

4、系统根据数据绘制可视化界面

5、系统将房价数据可视化界面进行显示

**4.2 房价对比可视化**

1、初始化可视化界面

2、用户选择对比的两个楼盘

3、系统向数据库查询两个楼盘的房价数据

4、系统根据数据绘制对比可视化界面

5、系统将房价对比可视化界面进行显示

**4.3 房价趋势可视化**

1、初始化可视化界面

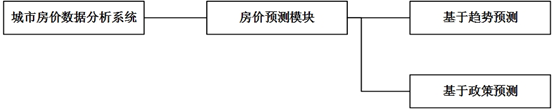
2、用户选择房价趋势

3、系统向数据库查询楼盘的历史房价数据

4、系统根据历史数据绘制可视化界面

5、系统将房价趋势可视化界面进行显示

**5.房价预测模块**



**5.1 基于趋势预测**

1、初始化房价预测界面

2、用户选择将要进行房价预测的地区

3、用户点击房价预测按钮

4、系统调用基于趋势的房价预测模型

5、系统根据模型和预测范围进行房价预测

6、系统显示房价预测的结果

**5.2 基于政策预测**

1、初始化房价预测界面

2、用户选择将要进行房价预测的地区

3、用户点击房价预测按钮

4、系统调用基于政策的房价预测模型

5、系统根据模型和预测范围进行房价预测

6、系统显示房价预测的结果

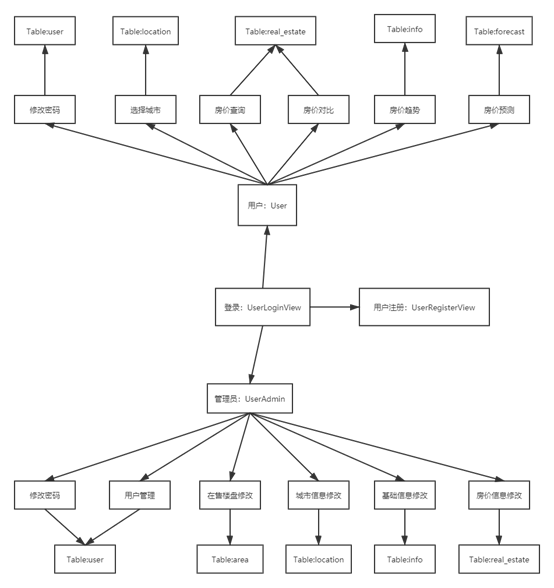
**三、模块定义**

|  |  |  |
| --- | --- | --- |
| 模块 | 组件 | 详细信息 |
| 注册登录 | 用户注册 | 用户在系统内进行注册 |
| 用户登录 | 根据用户账户密码进行登录 |
| 验证码验证 | 用户输入验证码进行人机验证 |
| 房价数据查询 | 全国查询 | 用户查询全国各省份的房价信息 |
| 地区查询 | 用户查询各地区的房价信息 |
| 楼盘查询 | 用户查询各楼盘的房价信息 |
| 数据显示 | 系统对用户所查询的数据进行显示 |
| 房价对比 | 楼盘价格对比 | 系统对不同楼盘的房价进行对比 |
| 历史价格对比 | 系统对楼盘的历史价格进行对比 |
| 房价可视化 | 房价数据可视化 | 系统对房价数据进行可视化 |
| 房价对比可视化 | 系统对房价对比进行可视化 |
| 房价趋势可视化 | 系统对房价趋势进行可视化 |
| 房价预测 | 基于趋势预测 | 系统对房价进行基于房价趋势的预测 |
| 基于政策预测 | 系统对房价进行基于经济政策的预测 |

**第五部分 E-R实体设计**

一、E-R实体结构图

**对象图：**



二、实体描述

1、user实体描述

|  |  |  |  |
| --- | --- | --- | --- |
| **编号** | **英文名** | **中文名** | **数据类型** |
| 1 | id | 用户id | int |
| 2 | password | 密码 | varchar |
| 3 | last\_login | 上次登录时间 | datetime |
| 4 | is\_superuser | 是否为管理员 | tinyint |
| 5 | username | 用户名 | varchar |
| 6 | first\_name | 名 | varchar |
| 7 | last\_name | 姓 | varchar |
| 8 | sex | 性别 | tinyint |
| 9 | address | 地址 | varchar |
| 10 | date\_joined | 注册日期 | datetime |

2、location实体描述

|  |  |  |  |
| --- | --- | --- | --- |
| **编号** | **英文名** | **中文名** | **数据类型** |
| 1 | id | 城市ID | int |
| 2 | province | 省份名 | varchar |
| 3 | city | 城市名 | varchar |
| 4 | is\_ready | 是否展示 | tinyint |
| 5 | domain | 城市名拼音或缩写 | varchar |

3、real\_estate实体描述

|  |  |  |  |
| --- | --- | --- | --- |
| **编号** | **英文名** | **中文名** | **数据类型** |
| 1 | id | 楼盘id | int |
| 2 | province | 省份名 | varchar |
| 3 | city | 城市名 | varchar |
| 4 | category | 新楼盘或二手房 | varchar |
| 5 | name | 楼盘名称 | varchar |
| 6 | price | 价格 | int |
| 7 | total | 总价(万元) | float |
| 8 | area | 所在区县 | varchar |

4、area实体描述

|  |  |  |  |
| --- | --- | --- | --- |
| **编号** | **英文名** | **中文名** | **数据类型** |
| 1 | id | 区县id | int |
| 2 | area | 区县名 | varchar |
| 3 | num | 楼盘个数 | int |
| 4 | city | 城市名 | varchar |
| 5 | province | 省份名 | varchar |
| 6 | domain | 城市名拼音或缩写 | varchar |

5、info实体描述

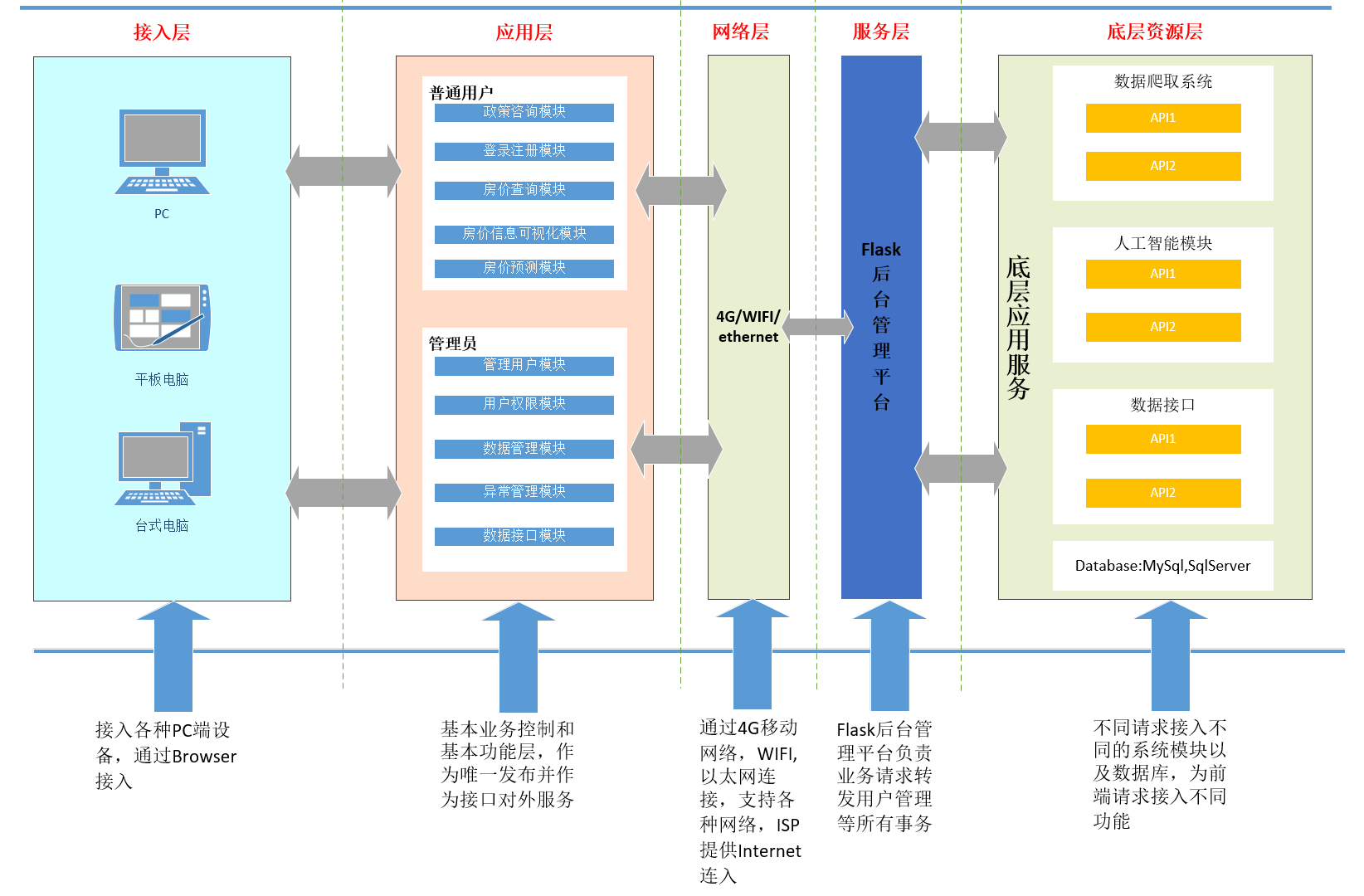
|  |  |  |  |
| --- | --- | --- | --- |
| **编号** | **英文名** | **中文名** | **数据类型** |
| 1 | id | Info id | int |
| 2 | xfPrice | 近12个月新房价格 | varchar |
| 3 | esfareaPrice | 近12个月二手房价格 | varchar |
| 4 | compareMonth | 近12个月(横坐标的值，间隔为一个月) | varchar |
| 5 | compareYear | 横坐标的值所属年份 | varchar |
| 6 | minPrice | 近12个月最高价 | float |
| 7 | maxPrice | 近12个月最低价 | float |
| 8 | threeAreaPrice | 近3年新房价格 | varchar |
| 9 | threeEsfareaPrice | 近3年二手房价格 | varchar |
| 10 | threeCompareDate | 近3年(横坐标的值，间隔为三个月) | varchar |
| 11 | threeCompareYear | 横坐标的值所属年份 | varchar |
| 12 | threeminPrice | 近3年最高价 | float |
| 13 | threemaxPrice | 近3年最低价 | float |
| 14 | buildPriceNameList | 每平米价格区间（6千以下,6-8千,8千-1万,1-1.2万,1.2-1.5万,1.5-2万,2万以上） | varchar |
| 15 | buildPriceCount | 每平米价格区间对应的数量 | varchar |
| 16 | esfPriceNameList | 二手房价格（70万以下,70-100万,100-150万,150-200万,200-250万,250-300万,300-400万,500万以上） | varchar |
| 17 | esfPriceCount | 二手房价格对应的数量 | varchar |
| 18 | domain | 城市名拼音或缩写 | varchar |
| 19 | province | 省份名 | varchar |
| 20 | city | 城市名 | varchar |

6、forecast实体描述

|  |  |  |  |
| --- | --- | --- | --- |
| **编号** | **英文名** | **中文名** | **数据类型** |
| 1 | province | 省份名 | varchar |
| 2 | city | 城市名拼音或缩写 | varchar |
| 3 | interval | 预测的时间间隔(月、年) | varchar |
| 4 | num | 横坐标的值 | int |
| 5 | xf\_avg | 新房价格 | float |
| 6 | esf\_avg | 二手房价格 | float |
| 7 | avg | 总体平均价格 | float |
| 8 | xf\_percent | 新房上涨或下降百分比 | varchar |
| 9 | esf\_ percent | 二手房上涨或下降百分比 | varchar |
| 10 | percent | 总体百分比 | varchar |
| 11 | domain | 城市名拼音或缩写 | varchar |

第六部分 总体设计

一、Flask逻辑架构设计



1、轻量级的Flask-web架构

从构件化与可扩展化的角度出发，采用Flask-web架构的技术方案、可扩展化的开发模式，应用功能的实现在支撑扩展件的基础上完成。内置开发服务器和调试器，自带的开发服务器使开发者在调试程序时无须再安装其他任何网络服务器就能调试，Flask默认处于调试状态，使得运行中的任何错误会同时向两个目标发送信息，能与Python单元测试功能无缝衔接。

2、专业的B&C/S结构

将开发系统的目标用户群进行细分，划分为管理员和普通用户。考虑到B/S在信息采集和信息展示方面的灵活性的特点，C/S在安全性、事务处理和交互性方面的特点。本工程采用B&C/S结构，充分满足客户的业务习惯和需求。

3、定制化的逻辑架构

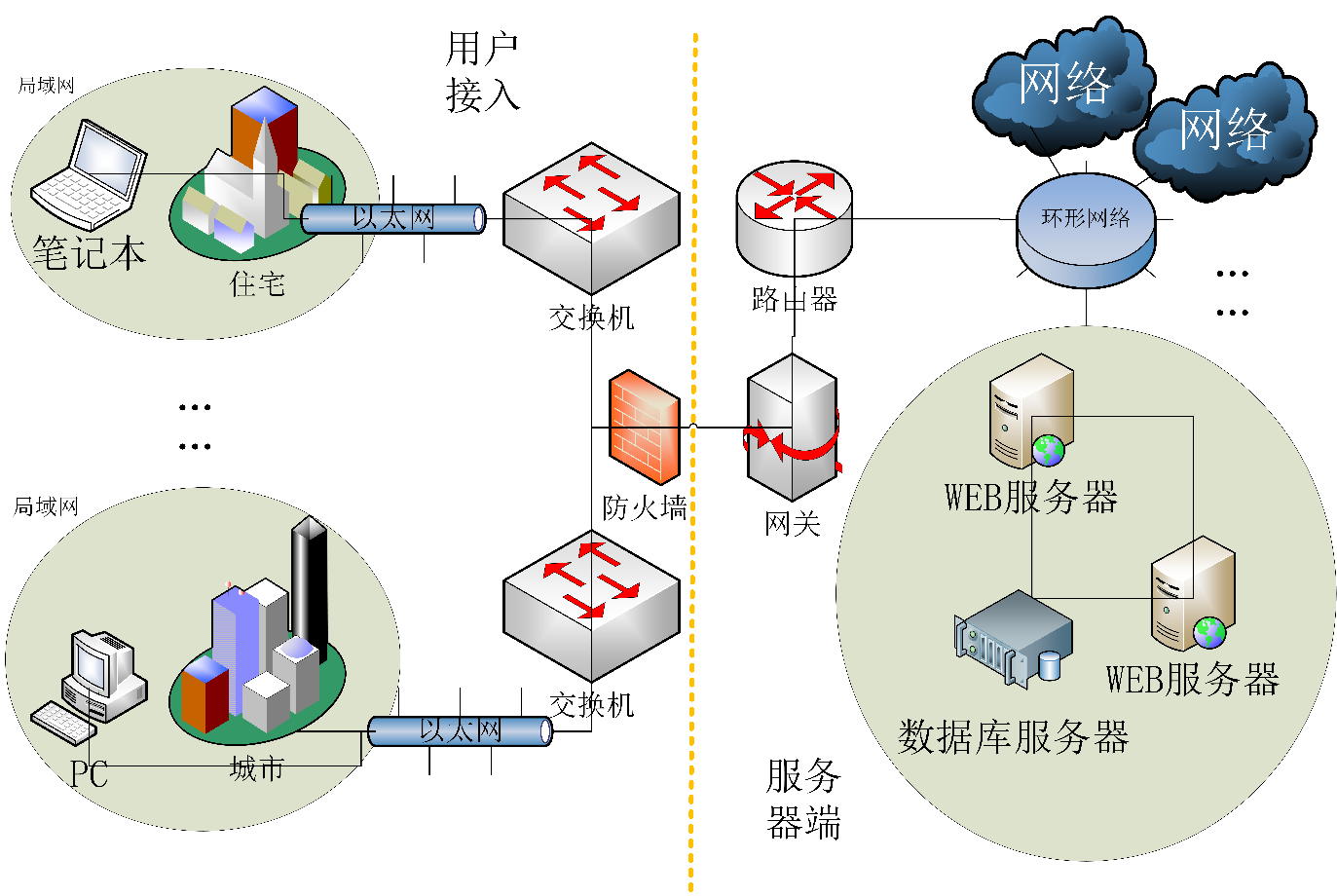
把每个功能都分成三部分，用户视图层，逻辑接口层，数据处理层，逻辑清晰。如果用户更换不同的用户界面或不同的数据存储机制都不会影响接口层的核心逻辑代码，扩展性强。

可以在接口层，准确的记录日志与流水。所有的业务单元都是通过json描述反应业务流程的，所有外部数据交换都是通过json接口描述来实现的。

4、终端web-application

在Flask框架上扩展业务处理应用，通过WIFI/以太网等联网技术，便于管理员方便、快捷、实时处理业务。

**二.物理架构设计**



1）服务端

系统服务端可以部署到WEB服务器上，管理员通过浏览器实现业务管理操作；

2）客户端

因为系统采用B&C/S模式，所以PC客户端最低只要求IE浏览器支持，最好使用火狐，谷歌等兼容性更好的浏览器；

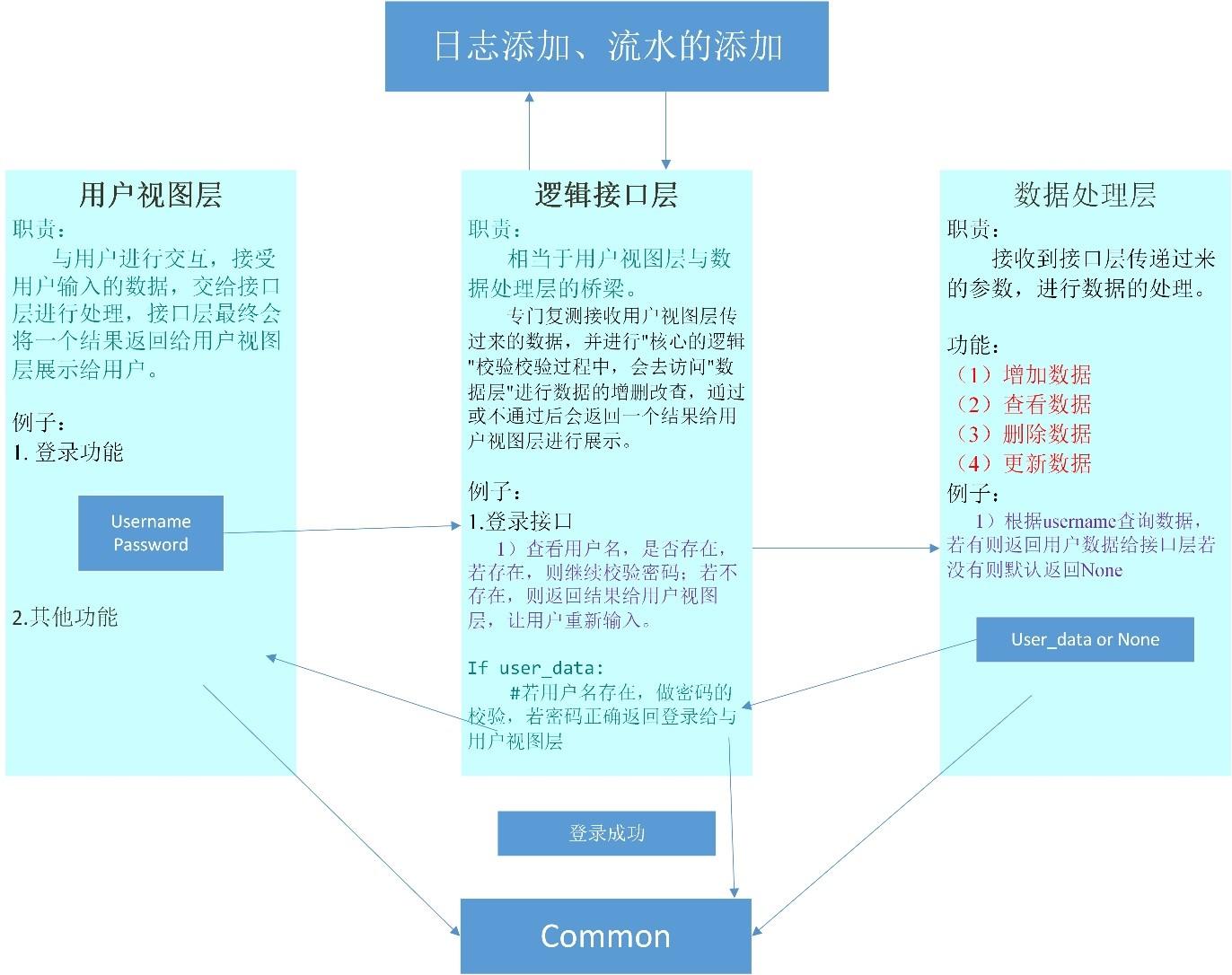
3）接口端

Python-Flask自带实现管理端口，并实现数据接口，管理员在浏览器上实现对数据操作。

三.技术架构设计

1、Python三层模型设计

系统实现Python的三层模型架构，使用Python3.7.3标准库，Flask的架构库，virtualenv负责创建虚拟环境，python-dotenv管理虚拟环境，pysql和SQLAlchemy作为模型层进行数据的持久化。



1）数据处理层

每个业务功能的主操作场景采用一个核心数据处理类，利用goolge或其他浏览器对PC端场景切换设计，由数据处理类控制不同业务处理用户视图的切换。

接收逻辑接口层传递过来的参数，做数据的增删改查。

2）用户视图层

用于与用户交互的，可以接收用户的输入，打印接口返回的数据。用户视图层负责数据显示与数据输入。数据的输入通过事件交给逻辑接口层处理，数据显示通过逻辑接口层的托管实现刷新。

3）逻辑接口层

逻辑接口层服务数据处理，负责向服务器请求数据，并把数据处理好传递给托管。交给用户视图层完成。接收用户视图层传递过来的参数，根据逻辑判断调用数据层加以处理，并返回一个结果给用户视图层

2、三层架构设计的好处

1）把每个功能都分成三部分，逻辑清晰

2）如果用户更换不同的用户界面或不同的数据存储机制都不会影响接口层的核心逻辑代码，扩展性强

3）可以在接口层，准确的记录日志与流水

3、Web Service技术

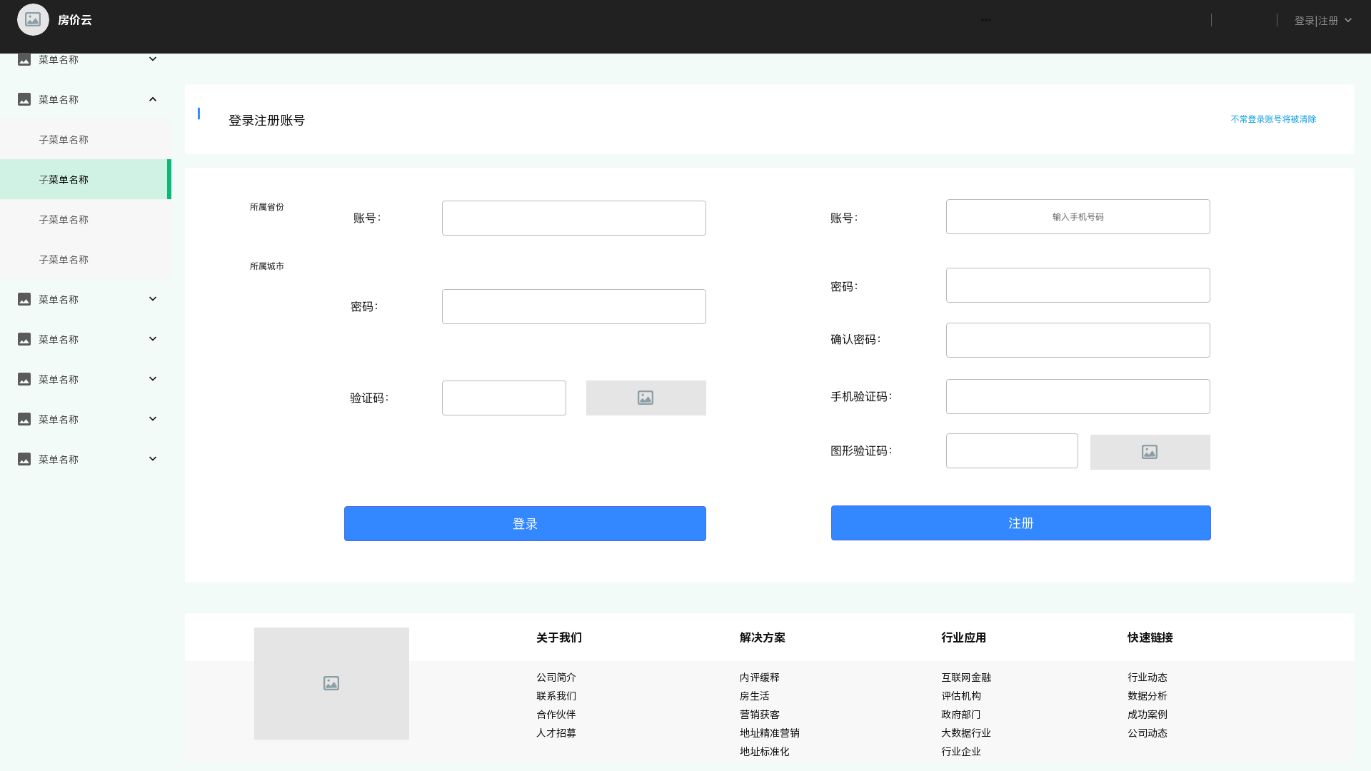
PC端的所有数据请求采用Web Service技术，在Flask的后台应用设计一个WebService服务器服务提供，通信的数据格式采用json。

在PC的浏览器端大量处理json的解析处理。

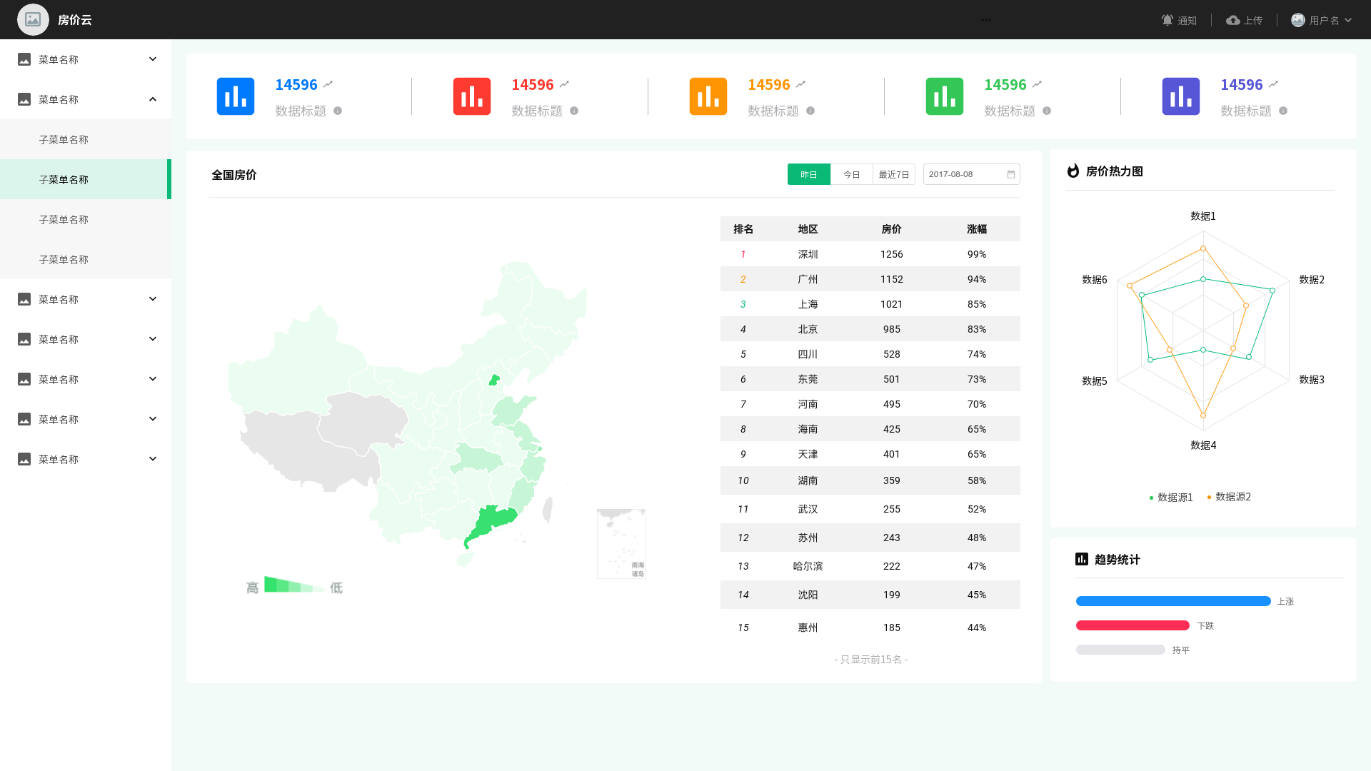
**第七部分 用户界面设计**

一、桌面布局设计

1、登录注册界面风格



2、主桌面风格



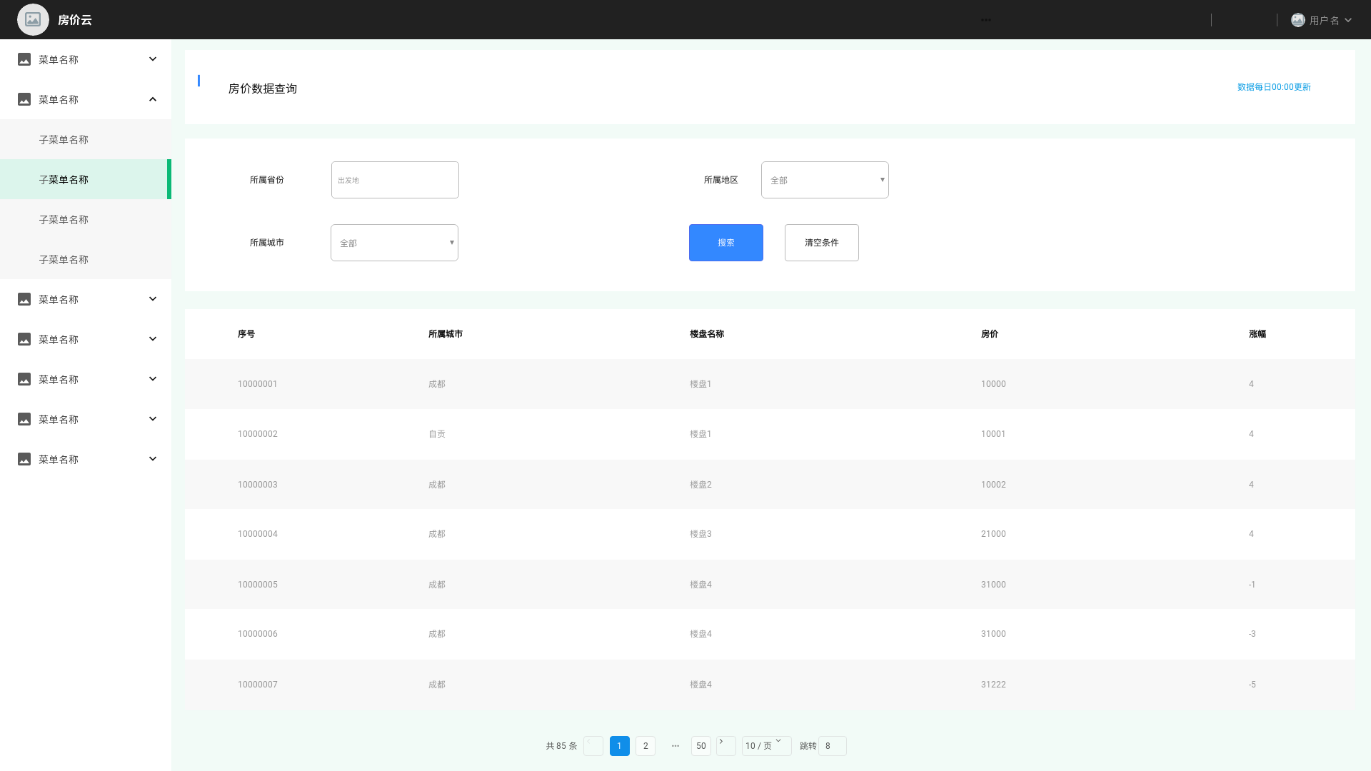
1、  点击每个导航按钮后进入相应的内容页面

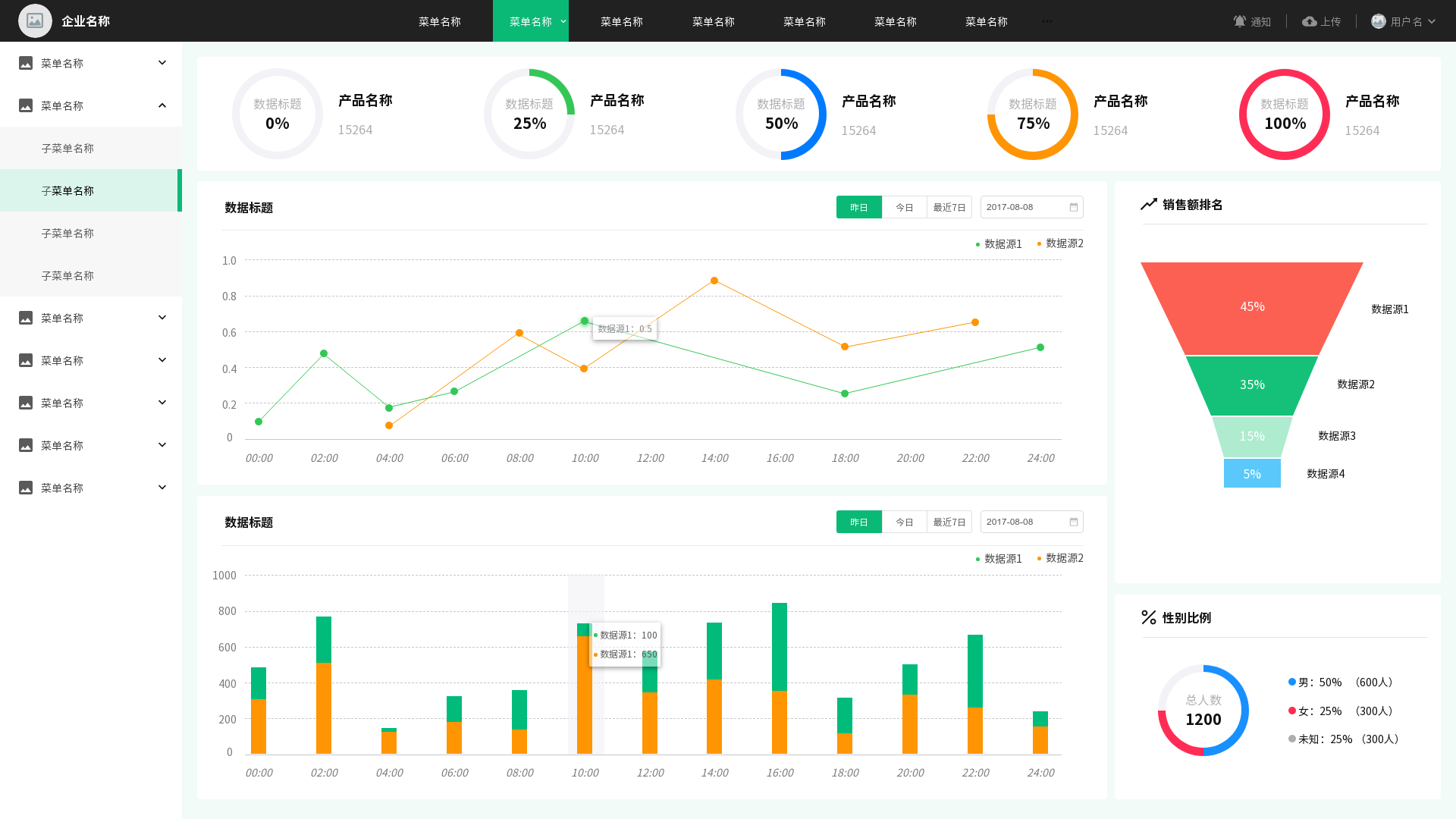
2、  点击具体的城市进入城市详细房价中心，点击用户可进入个人中心。

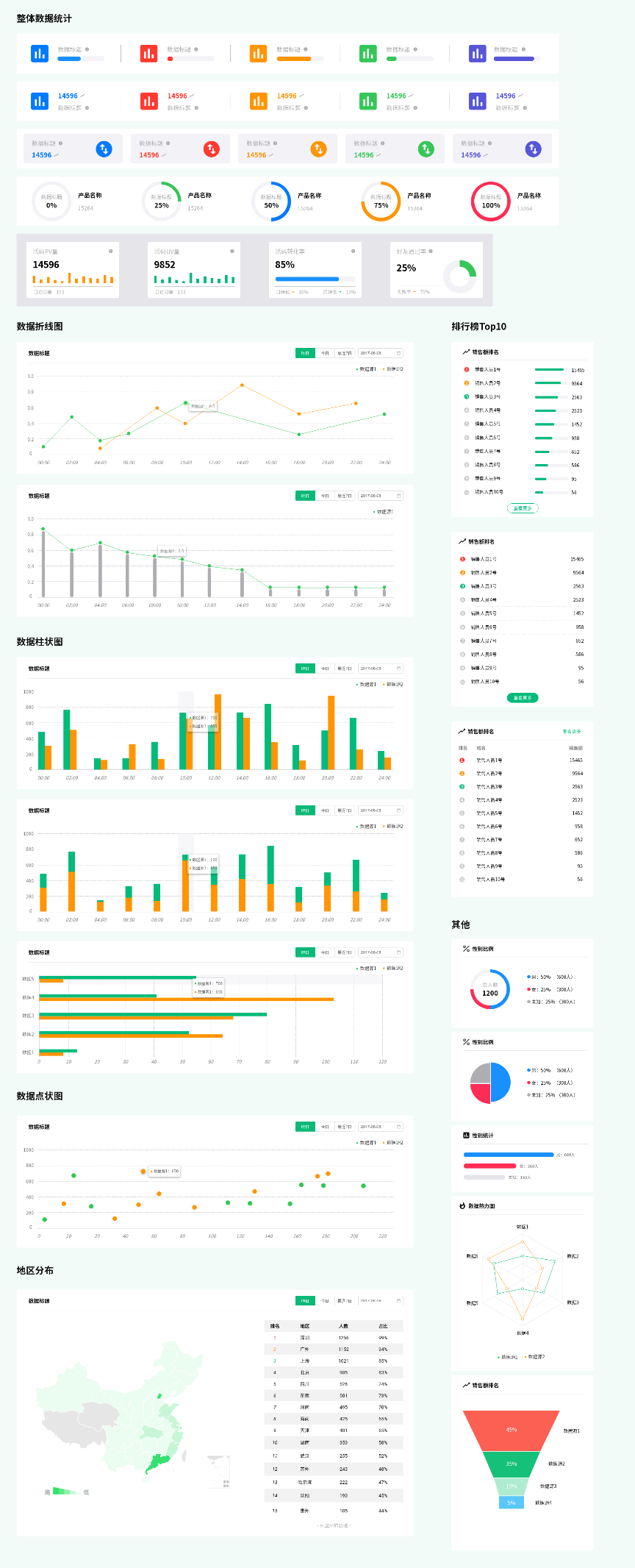
3、  菜单图标中，会有房价预测，房价查询，政策信息等功能项.

二、业务界面风格展示

1、风格展示







统一风格采用顶部和侧边的导航条操作风格。数据的展示使用同一风格的颜色和格式图表。

根据业务不同，每个区域根据需要可以不要。

**第八部分 运行环境和部署**

一、运行环境

1、终端浏览器环境

1)    IE 11版本以上

2)    Google Browser

3)    Firefox Browser等兼容性好的浏览器

2、客户机器环境

1)    Windows XP和以上

2)    Linux

3)    MacOS

3、开发环境要求

|  |  |  |
| --- | --- | --- |
| 项目 | 名称 | 版本 |
| 开发平台 | Windows 10 | 2019 |
| 开发工具 | Visual Studio Code/PyCharm | Visual Studio Code1.47/PyCharm2019 |
| 代码管理工具 | GIT | git version 2.18.0.windows.1 |
| 开发环境 | Python | 3.7.3  3.7.6  3.8 |

二、系统性能要求

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| # | 项目 | 模块 | 级别 | 技术参数 |
| 1 | 设计实现技术指标 | 系统架构 | A | 采用C/S模式三层架构。 |
| 2 | 面向对象开发语言与框架 | A | 采用Python开发语言和Javascript等前端脚本语言。采用Viso图表设计软件绘制框架 |
| 3 | 注释和文档 | A | 符合CMMI软件开发过程标准文档（至少提供：需求、概要、详细设计、测试报告、部署和环境、用户手册），代码注释量>=30%。 |
| 4 | 模块化和适合实训 | A | SOA设计、模块化，保证系统各模块单元较强的独立性适合实训。 |
| 5 | 测试覆盖率 | A | 功能覆盖率>=100%，业务覆盖率>=100%，语言覆盖率>=100%，逻辑覆盖率>=80%。 |
| 6 | 资源利用率要求 | CPU占用率 | B | <=70%利用率（附近标准配置） |
| 7 | 内存使用率 | B | <=85%利用率（附近标准配置） |
| 8 | 响应时间要求 | 服务器 | B | <=100ms（附近标准配置） |
| 9 | 网络 | B | <=100ms（附近标准配置） |
| 10 | 客户端 | B | <=5s（附近标准配置） |
| 11 | 系统稳定性要求 | 成熟性 | A | 真实的用户，成功使用本系统 |
| 12 | 稳定性 | B | 无故障运行时间>=365天，系统恢复时间<=2小时。 |
| 13 | 先进性 | A | 采用目前体验最好、最流行、兼容性好的谷歌浏览器与火狐浏览器。 |
| 14 | 典型意义 | A | 案例项目要有典型意义，有推广价值。 |
| 15 | Web服务接口要求 | WebService服务器端 | A | 支持协议定制，支持安全过滤，支持消息队列，多种模式重发，支持成功检测。消息延迟<=2S，丢包<=0.001%。 |
| 16 | FLASK平台框架 | A | 消息延迟<=2S，丢包<=0.001%。 |
| 17 | 集成部署环境 | 服务器 | A | Linux Server |
| 18 | 数据库 | A | MySQL 8+及以上 |
| 19 | PC电脑 | A | Linux，MacOs，Microsoft Windows XP 及以上 |
| 说明：级别（A:表示非常重要必须达到的技术性能要求,B:表示重要推荐达到的技术性能要求,C：表示非重要可以弱化的技术性能要求.） | | | | |