高考志愿填报与录取系统详细设计说明书

**1 引言**

**1.1 编写目的**

本文档为高考志愿填报与录取系统的概要设计说明。文档中明确说明了系统的设计思想、基本功能、模块划分以及后期维。以便于各模块开发人员能更好地了解该系统的基本情况及各模块详细功能。本文档可以为程序设计人员和系统管理人员提供清晰的设计思路，是详细设计的依据和基础，同时在开发后期的维护阶段也起到至关重要的作用。

**本文档的预期读者为：**

1）项目的责任人和管理者；

2）参与概要设计和需求确认的有关业务人员；

3）项目的开发人员、测试人员；

**1.2 背景**

高考一直是社会各界高度关注的焦点话题。在高校招生录取这个庞大系统工程中,志愿填报和录取机制属于其中一个关键而复杂的子系统,同时也是高考制度改革的重点和难点。高考志愿填报与录取系统是在高考招生制度的基础上,为考生提供志愿参考的服务系统。

**系统名称：**高考志愿填报与录取系统

**项目提出者及负责人：**指导教师赵逢禹

**项目开发者：**软件协同设计第五小组

**系统用户：**参与高考的毕业生及招生管理部门的工作人员

**软件运行中心：**考生的个人电脑及招生管理部门内部系统

**1.3 定义**

列出本文件中用到的专门术语和外文首字母的组词的原词组。

**1.4 参考资料**

《软件工程导论》张海藩 清华大学出版社 2013.8

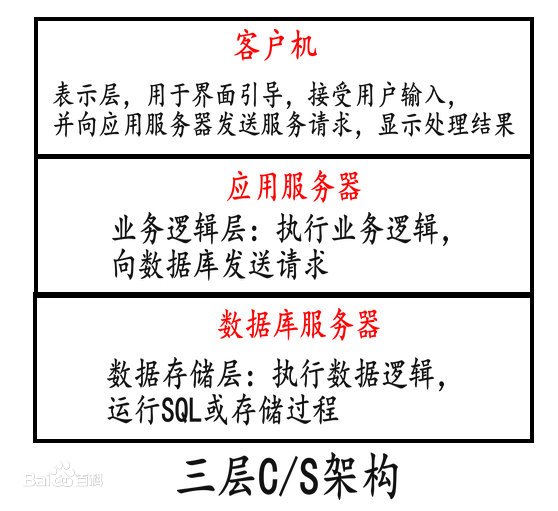
**2 总体设计**

**2.1 设计概述**

本系统采用 B/S 即 Brower/Server 架构。

B/S 结构（Browser/Server，浏览器/服务器模式），是 WEB 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。这种模式统一了客户端，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，如Netscape Navigator 或 Internet Explorer，服务器 安装SQL Server、Oracle、MYSQL 等数据库。浏览器 通过 Web Server 同数据库进行数据交互。

随着 Internet 和 WWW 的流行，以往的主机/终端和 C/S 都无法满足当前的全球网络开放、互连、信息随处可见和信息共享的新要求，于是就出现了 B/S 型模式，即浏览器/服务器结构。它是 C/S 架构的一种改进，可以说属于三层 C/S 架构，如图3所示。主要是利用了不断成熟的 WWW 浏览器技术，用通用浏览器就实现了原来需要复杂专用软件才能实现的强大功能，并节约了开发成本，是一种全新的软件系统构造技术。



**图 1 三层 C/S 架构**

第一层是浏览器，即客户端，只有简单的输入输出功能，处理极少部分的事务逻辑。由于客户不需要安装客户端，只要有浏览器就能上网浏览，所以它面向的是大范围的用户，所以界面设计得比较简单，通用。

第二层是 WEB 服务器，扮演着信息传送的角色。当用户想要访问数据库时，就会首先向 WEB 服务器发送请求，WEB 服务器统一请求后会向数据库服务器发送访问数据库的请求，这个请求是以 SQL 语句实现的。

第三层是数据库服务器，他扮演着重要的角色，因为它存放着大量的数据。当数据库服务器收到了 WEB 服务器的请求后，会对 SQL 语句进行处理，并将返回的结果发送给 WEB 服务器，接下来，WEB 服务器将收到的数据结果转换为 HTML 文本形式发送给浏览器，也就是我们打开浏览器看到的界面。

本系统比较大的特点就是前后端的完全分离，引入了 Express.js 作为 Node.js 中间件，使我们能够使用 JavaScript 在前端实现 Controller 层，避免了使用 Java Servlet 来控制浏览器的页面跳转。

**2.2 运行环境**

浏览器：火狐、谷歌、QQ 浏览器，Internet Explorer 等桌面版浏览器

操作系统：Windows 7 及以上

支持环境：IIS 5.0

内存：512 M 以上

硬盘空间：40G 以上

CPU: 233MHZ 以上

内存：256M 以上

硬盘空间：1.5G 以上

**2.3 系统的结构设计**

我们采用了完全的前后端分离架构，如图 2 所示，前端由用户浏览器和运行在前端服务器中的 Node.js 服务程序组成，后端由运行在后端服务器的 Java Servlet 和数据库 Mysql 组成。

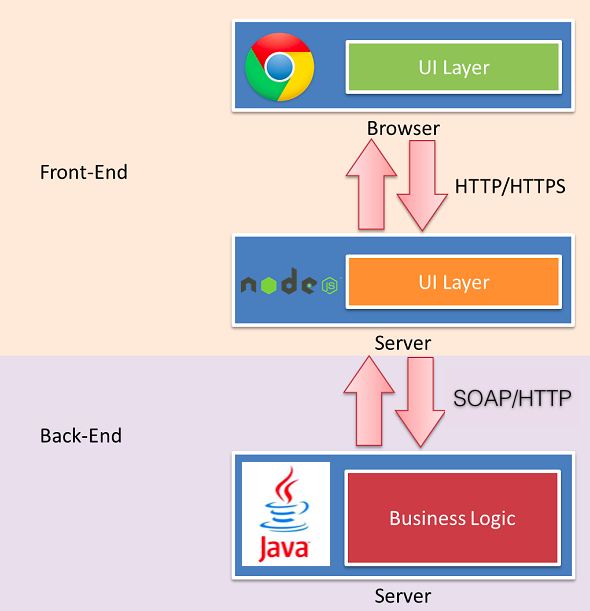
为什么我们要加入 Node.js 层呢？现有的前后端分离解决方案将 Controller 层和 Model 层都放在了后端进行开发，前端只负责 View 层展示，这种模式严重阻碍了前端开发效率，也让后端不能专注于业务开发。

更好的解决方案是让前端能控制 Controller 层，但是如果在现有技术体系下很难做到，因为不可能让所有前端都学 Java，安装后端的开发环境。

Node.js 就能很好的解决这个问题，我们无需学习一门新的语言，就能做到以前开发帮我们做的事情，一切都显得那么自然。

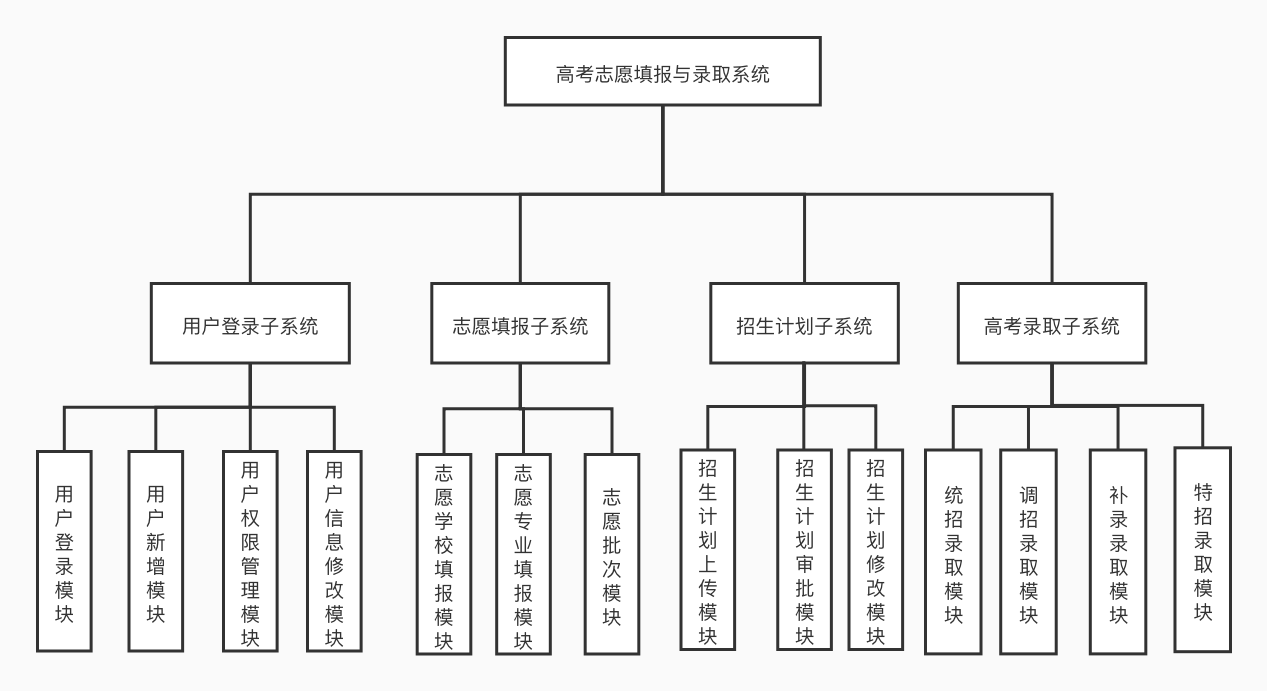
所以，我们引入了 Node.js 中间层，用户通过 URL 访问网页时，浏览器通过 HTTP 协议向前端服务器发送请求，运行在前端服务器中的 Node.js 服务程序将网页内容返回给用户浏览器，最后在浏览器中进行显示。当用户点击页面中的链接进行跳转时，也是同样向 Node.js 服务程序发送请求，这样就实现了让前端控制 Controller 层。

当用户请求的页面包含数据时，Node.js 服务程序会向运行在后端服务器的 Java 服务程序发送请求，从而从数据库中获取到需要的数据，并在 Node.js 服务程序中进行一些初步的渲染，最后返回用户浏览器进行显示。



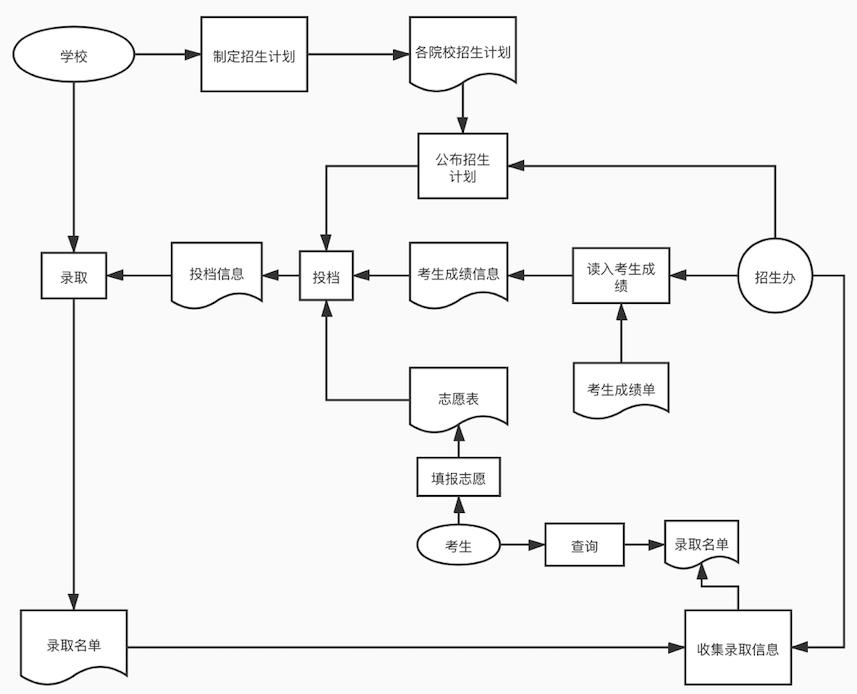
**图 2 基于 Node.js 的前后端分离架构**

本系统主要分为四个子系统，一个子系统由用户方使用，主要实现用户登录、增改等功能，一个是志愿填报子系统，完成考生填报学校专业的操作，一个为招生计划子系统，由学校使用，另一个为高考录取子系统，完成招生办的录取功能。划分的子系统结构图如下所示：



**图 3 功能模块图**

各个系统业务流程图如下所示：



**图 4 业务流程图**

**2.4 问题域类模型**

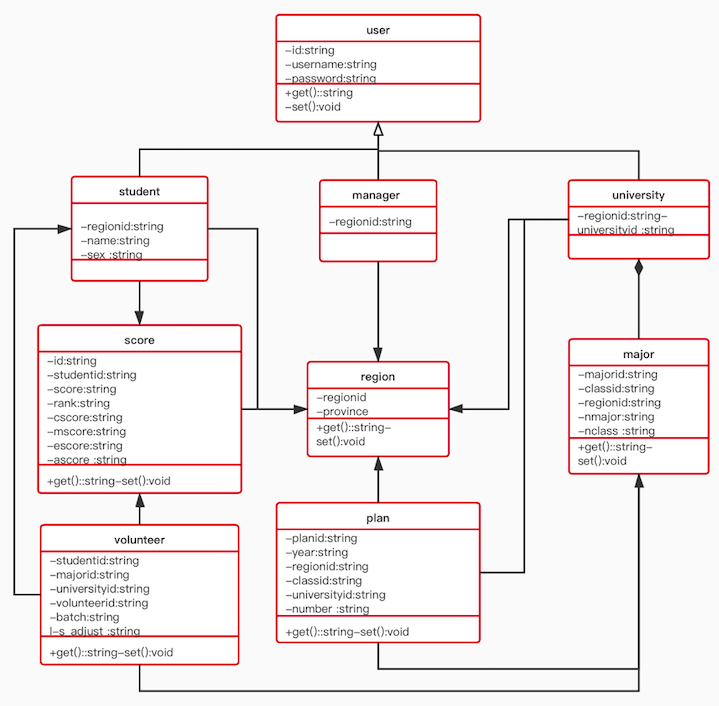
通过分析需求，共筛选出 8 个问题域对象，分别是系统管理员、学生、高校招生办、高考志愿、招生计划、高考成绩、地区、专业对象及三个持久化实现类，每个类的描述如表 1 所示：

各问题域类之间的关系描述如下：

**表 1 实体类描述**

|  |  |
| --- | --- |
| **问题域类** | **描述** |
| manager | 存储管理员信息 |
| student | 存储学生信息 |
| score | 存储学生高考成绩信息 |
| plan | 存储招生计划信息 |
| region | 存储地区信息 |
| major | 存储学校专业信息 |
| volunteer | 存储学生高考志愿信息 |
| university | 存储高校招生办信息 |
| user | 存储用户账号密码 |

实体类之间的关系如图 5 所示。



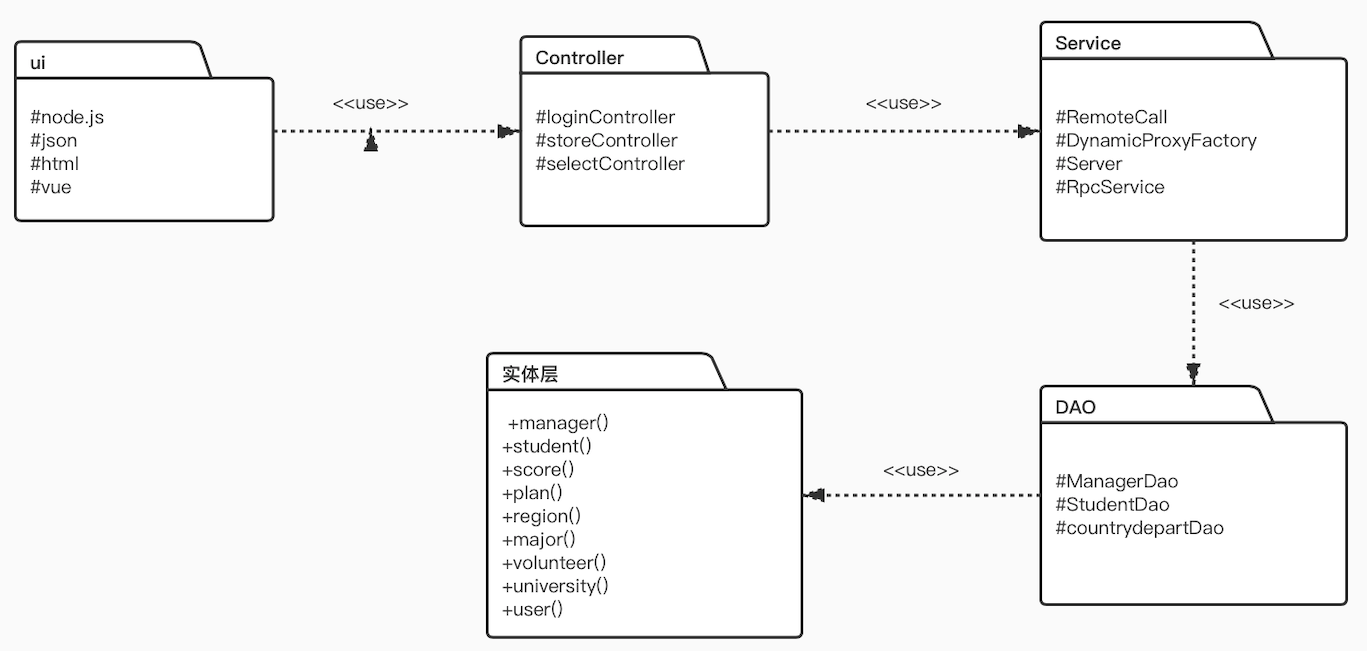
**图 5 实体类之间的关系**

持久化类描述如表 2 所示。

**表 2 持久化类描述**

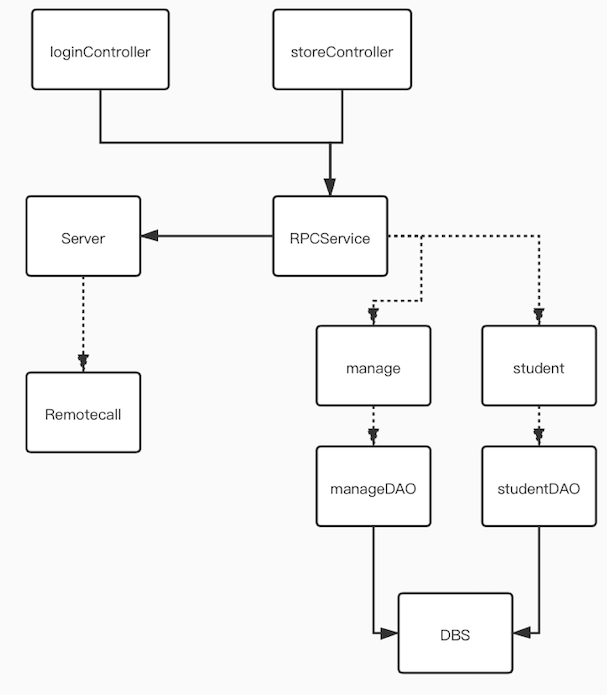
|  |  |
| --- | --- |
| ManagerDao | 管理员与数据库的连接类 |
| StudentDao | 学生与数据库的连接类 |
| countrydepartDao | 国家招生管理部门与数据库的连接类 |

系统顶层结构设计：通过包图说明了各设计模块之间的关系，包括数据库设计，功能服务设计，控制类设计以及 UI 设计。ui 负责前端页面数据传输及界面跳转的实现，controller 层为控制层，主要处理外部请求，调用 service 服务层，将数据封装返回给调用方。系统包图如下所示：



**图 6 包图**

控制类之间的关系如图 7 所示。

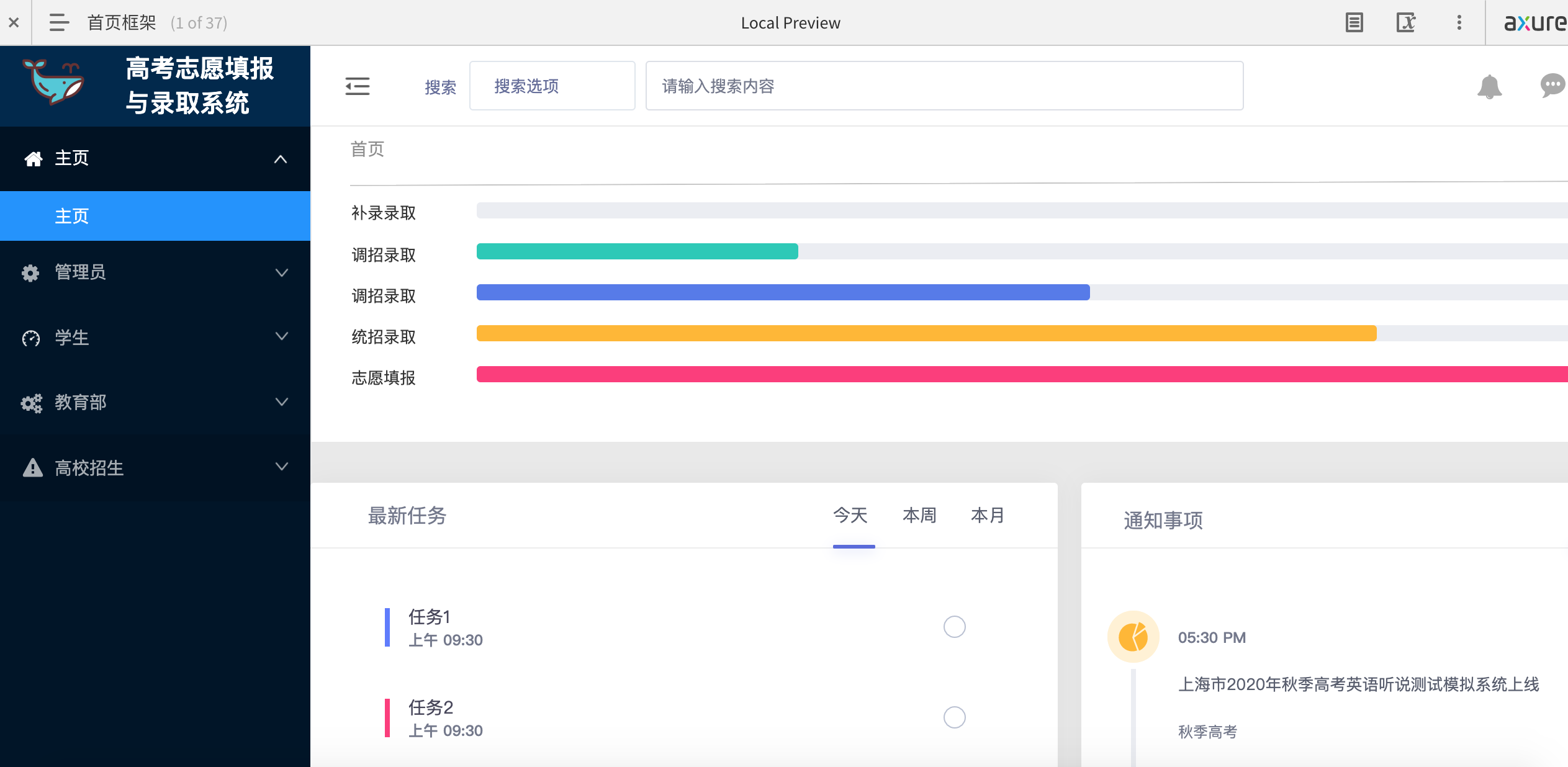


**图 7  控制类之间的关系**

**2.5 系统的界面设计**

**Axure原型链接：**<https://fj072c.axshare.com>

图8为首页的整体框架，我们将左侧的导航栏根据使用角色的不同分为了5个大类。分别为主页、管理员、学生、教育部、高校招生。



**图 8 首页框架**

图9为首页页面，页面的最上方有一个搜索框，可以用来搜索整个页面的内容。

在搜索框的下方，是录取工作的进度条。页面左下方是登陆用户需要完成的任务，比如学生需要填报志愿，高校需要提交招生计划等。

为了美观考虑，我们在页面右下角放入了通知事项一栏，通知并不存入数据库，而是将从上海市教育考试院摘取的一些信息直接编码成链接放在页面中。



**图 9 首页**

图10为账号管理页面，管理员有权对账号进行新增、修改和删除。



**图 10 账号管理**

图11为高考成绩页面。



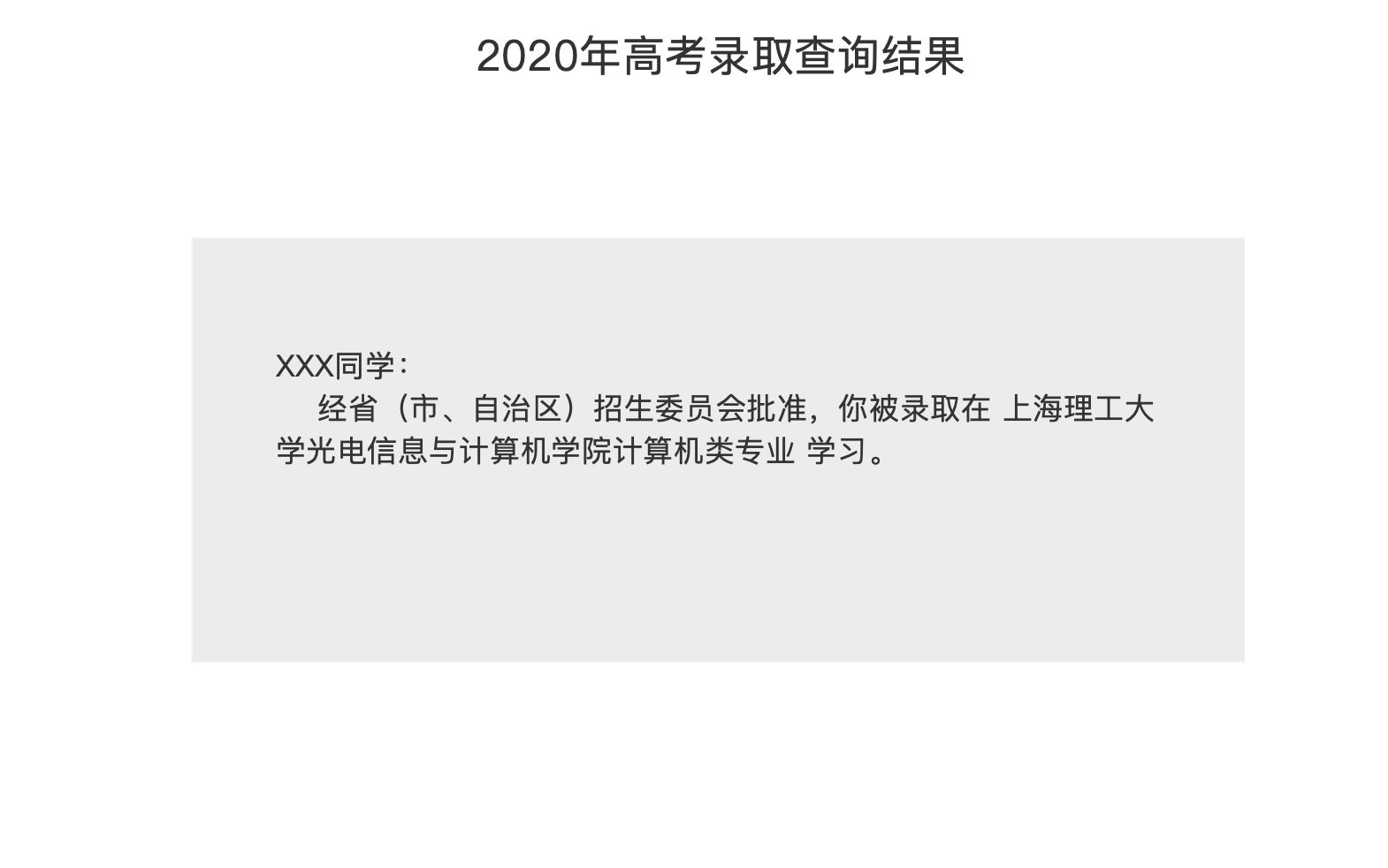
**图 11 结果查询**

图12为统招/补录/特招志愿填报页面。



**图 12 志愿填报**

图13为录取结果页面。



**图 13 录取结果**

图14为招生计划审批页面。



**图 14 招生计划审批**

图15为招生进度管理页面。



**图 15 招生进度管理**

图16为招生计划上传页面。



**图 16 招生计划上传**

图17为统招/补录/特招提档名单页面。



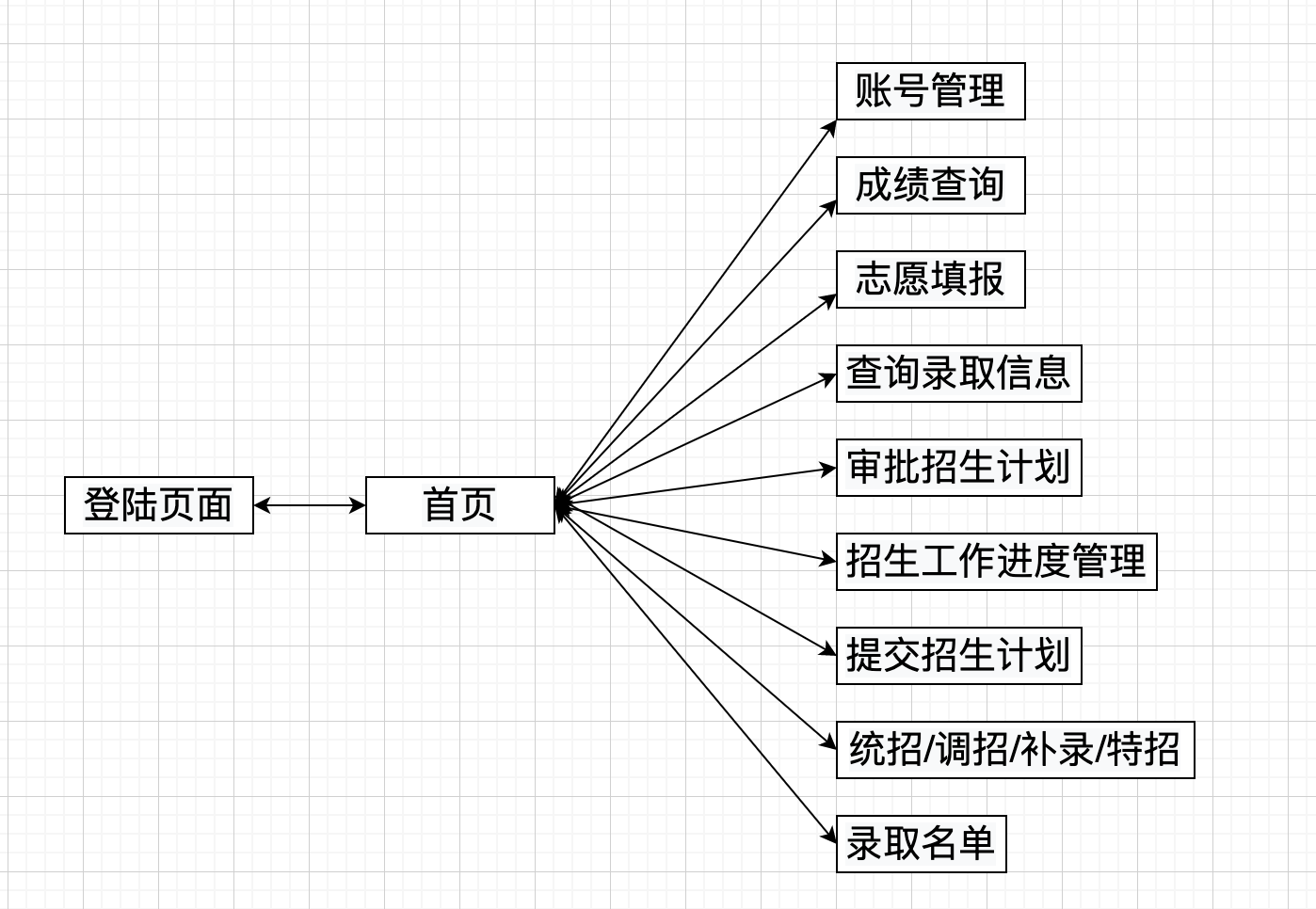
**图 17 提档名单**

图18为录取名单页面。



**图 18 录取名单**

图19为页面跳转逻辑图，因为首页中有侧边栏目录，对应着每一个分页面的入口，所以每一个分页面都能够通过首页进入。



**图 19 页面跳转逻辑**

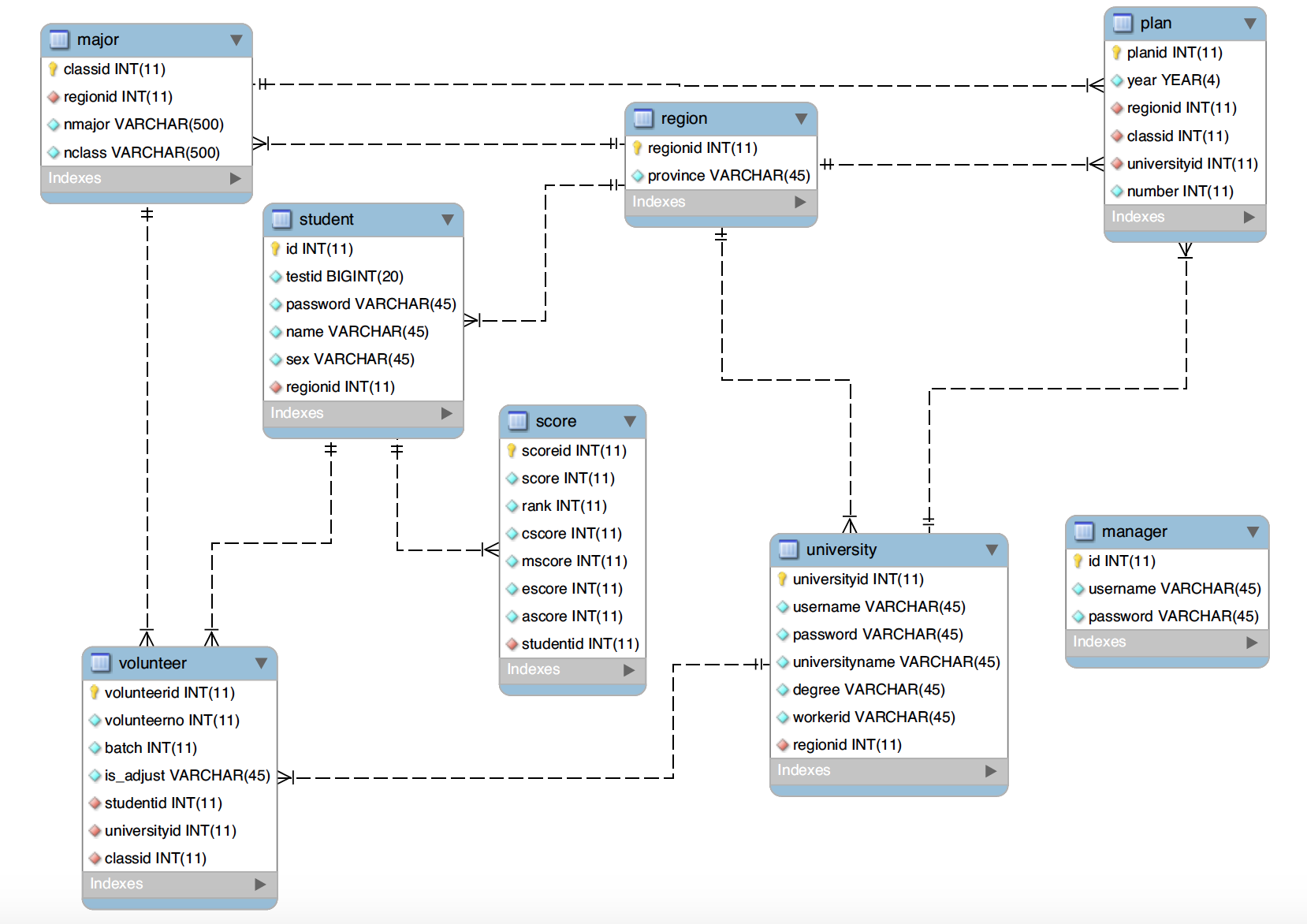
**2.6 数据存储设计**

根据分析得到的问题域对象，设计出每个对象对应的数据库类，以实现各对象涉及信息的增删查改操作，具体的类如表 3。

**表 3 数据库类描述**

|  |  |
| --- | --- |
| 数据库类 | 描述 |
| ManagerDao | 管理员与数据库的连接类 |
| StudentDao | 学生与数据库的连接类 |
| CountrydepartDao | 国家招生管理部门与数据库的连接类 |
| UniversityDao | 高校招生办与数据库的连接类 |

数据库的 ER 图如图 20 所示，一共有 8 张数据表。



**图 20 数据库 ER 图**

数据库表的设计如下。表 4 为系统管理员表，数据库中的表名为 manager。

**表 4 系统管理员表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 允许为 NULL | Key | 字段含义 |
| id | int | N | 主键 | 管理员编号 |
| username | varchar | N |  | 用户名 |
| password | varchar | N |  | 密码 |

表 5 为学生表，数据库中的表名为 student。

**表 5 学生表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 允许为 NULL | Key | 字段含义 |
| id | int | N | 主键 | 学生身份证号 |
| testid | int | N |  | 学生准考证号 |
| password | varchar | N |  | 密码 |
| regionid | int | N | 外键 | 学生所在地 |
| name | varchar | N |  | 姓名 |
| sex | varchar | N |  | 性别 |

表 6 为学生高考成绩表，数据库中的表名为 score。

**表 6 学生高考成绩表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 允许为 NULL | Key | 字段含义 |
| scoreid | int | N | 主键 | 编号 |
| studentid | int | N | 外键 | 学生身份证号 |
| score | int | N |  | 总分 |
| rank | int | N |  | 排名 |
| cscore | int | N |  | 语文成绩 |
| mscore | int | N |  | 数学成绩 |
| escore | int | N |  | 英语成绩 |
| ascore | int | N |  | 综合成绩 |

表 7 为招生计划表，数据库中的表名为 plan。

**表 7 招生计划表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 允许为 NULL | Key | 字段含义 |
| planid | int | N | 主键 | 编号 |
| year | int | N |  | 年份 |
| regionid | int | N | 外键 | 地区编号 |
| classid | int | N | 外键 | 专业大类编号 |
| universityid | int | N | 外键 | 大学编号 |
| number | int | N |  | 招生人数 |

表 8 为地区表，数据库中的表名为 region。

**表 8 地区表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 允许为 NULL | Key | 字段含义 |
| regionid | int | N | 主键 | 地区编号 |
| province | varchar | N |  | 省份/市 |

表 9 为专业表，数据库中的表名为 major。

**表 9 专业表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 允许为 NULL | Key | 字段含义 |
| classid | int | N | 主键 | 专业大类编号 |
| regionid | int | N | 外键 | 地区编号 |
| nmajor | varchar | N |  | 专业名称 |
| nclass | varchar | N |  | 大类名称 |

表 10 为高考志愿表，数据库中的表名为 volunteer。

**表 10 高考志愿表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 允许为 NULL | Key | 字段含义 |
| volunteerid | int | N | 主键 | 志愿编号 |
| studentid | int | N | 外键 | 学生身份证号 |
| classid | int | N | 外键 | 专业大类编号 |
| universityid | int | N | 外键 | 大学编号 |
| volunteerno | int | N |  | 第 x 志愿 |
| batch | int | N |  | 批次 |
| Is\_adjust | varchar | N |  | 是否允许调剂 |

表 11 为高校招生办表，数据库中的表名为 university。

**表 11 高校招生办表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 允许为 NULL | Key | 字段含义 |
| universityid | int | N | 主键 | 大学编号 |
| username | varchar | N |  | 招生办管理人员用户名 |
| password | varchar | N |  | 招生办管理人员密码 |
| workerid | varchar | N |  | 招生办管理人员编号 |
| regionid | int | N | 外键 | 地区编号 |
| universityname | varchar | N |  | 大学名称 |
| degree | varchar | N |  | 一本/二本 |

**2.7 尚未解决的问题**

软件对于并发的要求尚不清晰。

**3 运行设计**

**3.1 运行模块组合**

表12为运行模块组合表。

**表 12 运行模块组合表**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 所施控制        模块组合 | 用户密码修改模块 | 用户信息修改模块 | 权限管理模块 | 志愿填报模块 | 高考录取模块 | 招生计划模块 |
| 学生权限操作 | √ |  |  | √ |  |  |
| 教育部权限操作 | √ | √ |  |  | √ | √ |
| 高校权限操作 | √ | √ |  |  | √ | √ |
| 管理员权限操作 | √ | √ | √ |  |  |  |

**3.2 运行控制**

表13为运行控制表。

**表 13 运行控制表**

|  |  |
| --- | --- |
| 运行控制 | 控制方法 |
| 学生权限运行控制 | 用户登录后检查用户身份，若用户身份为学生，则系统对该用户施加学生权限控制 |
| 教育部权限运行控制 | 用户登录后检查用户身份，若用户身份为教育部，则系统对该用户施加教育部权限控制 |
| 高校权限运行控制 | 用户登录后检查用户身份，若用户身份为高校，则系统对该用户施加高校权限控制 |
| 管理员权限运行控制 | 用户登录后检查用户身份，若用户身份为管理员，则系统对该用户施加管理员权限控制 |

**3.3 运行时间**

表14为运行时间表。

**表 14 运行时间表**

|  |  |
| --- | --- |
| 运行控制 | 运行时间 |
| 学生权限运行控制 | <3s |
| 教育部权限运行控制 | <3s |
| 高校权限运行控制 | <3s |
| 管理员权限运行控制 | <3s |

**4 系统出错处理设计**

**4.1 出错信息**

本系统运行过程中可能遇到的错误及解决方法如表15所示。

**表 15 出错信息表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 错误类型 | 错误码 | 错误原因 | 显示内容 | 解决办法 |
| 数据库连接错误 | 401 | 数据库连接设置不正确，或数据库异常 | 数据库连接错误，请检查数据库连接 | 取消本次操作，提醒系统管理员检查数据库连接 |
| 用户操作错误或输入错误 | 402 | 输入不规范 | 参数格式错误，清重新输入 | 通过对话框，提醒用户操作错误，然后再次操作 |
| 数据丢失错误 | 403 | 输入的数据未能成功录入数据库 | 数据未成功保存，请重新操作 | 输入数据后检查是否成功保存至数据库，并发出提示 |
| 找不到页面 | 404 | 链接或网络错误 | Not Found | 检查链接是否正确，检查网络 |
| 计算机病毒 | 405 | 计算机遭到病毒、蠕虫或木马侵扰，导致程序无法正常运行 | 程序无法正常运行 | 安装查杀病毒软件并定期升级，必要时对系统数据备份，重装系统 |
| 系统程序错误 | 406 | 程序本身有错误而造成的功能不正常、死机、数据丢失、非正常中断 | 程序发生错误 | 定期检查和维护系统，修复程序错误 |
| 系统异常错误 | 407 | 因开发与运行平台的固有原因而产生的异常错误 | 系统异常，点击确定终止当前程序 | 输出错误报警信息，并终止当前程序的运行 |

**4.2 补救措施**

我们对本程序的几种可能的错误进行了分析，决定不同的错误采取以下不同的处理方法。

主要的处理方法有：

1）数据库连接错误：这类错误主要是数据库设置不正确，或数据库异常引起的，系统会提示用户取消本次操作，若问题不能解决则提醒用户可通过邮件方式通知系统管理员检查数据库设置。并且对于数据库中的信息，要定期进行数据备份， 每天进行增量转储，每月一次进行海量转储，可以使用静态转储也可使用动态转储。

2）用户操作错误或输入错误：造成问题的原因是用户输入不规范造成，系统将提示用户合法的输入信息，尽量避免用户的错误输入。

3）数据丢失错误：可能由于网络错误导致用户输入的数据不能成功保存到数据库中，解决该错误可向用户发出警告。

4）计算机病毒：计算机病毒为恶意植入计算机的程序。通过提示安装与更新的杀毒软件来预防此类原因造成的系统错误，必要是提示用户进行重要数据备份并重装系统。

5）系统程序错误：程序本身有未发现的逻辑错误而造成的功能不正常、死机、数据丢失、非正常中断，解决此类错误可以定期检查和维护系统，修复程序错误。

6）系统异常错误：该错误由于开发与运行平台的固有原因而产生的异常错误，解决手段为输出错误报警信息，并终止当前程序的运行。

**4.3 系统维护设计**

软件的维护主要包括系统程序的维护和数据库的维护。

对系统程序的维护，分为两方面，一方面系统维护人员定期对系统进行操作来发现问题或漏洞，并解决问题和漏洞，另一方面用户在使用系统时发现的自己不可解决的问题或严重的错误，可通过系统提出并通过邮件通知该系统维护人员。

对于数据库的维护，需要提供数据库的自动备份与恢复功能，方便实现数据库的维护和管理。

**5 接口设计**

我们使用了swaggerhub来进行接口文档的设计

链接：<https://app.swaggerhub.com/apis-docs/CurrenWong/CourseDesign/>