《巧兼职》开发文档

# 项目概况

## 项目名称

巧兼职

## 项目背景

“巧兼职”的主要定位是为所有高校的大学生提供一个兼职平台，此兼职平台秉着专一的精神一心一意只为对在校大学生提供服务。“巧兼职”寓意小巧玲珑的兼职平台、小巧而精致是这一平台的特色。通过融入目前大热的大数据的应用理念，以提升消费者需求品质为导向，提炼出适宜大学生的优质兼职信息为主要亮点。平台致力于通过最少的步骤、最简单的操作、最体贴的服务来打动用户的心。通过与各大高校合作的方式，以帮助学校打造兼职平台为特色，共同打造提供智能服务的新型大学生兼职平台。“巧兼职”最关键的是为学生提供兼职，增加兼职的数量，提供适合学生的兼职工作，巧兼职有智能推送兼职的功能，满足学生的要求是当务之急。也有根据学生的要求，列出符合要求的兼职信息让学生挑选。

## 项目目标

巧兼职的主要定位是为所有高校的大学生提供一个兼职平台，此兼职平台秉着专一的精神一心一意只为对在校大学生提供服务。平台的目的是帮助或者减轻校方做兼职发布的工作，尽量将校方发布兼职信息的工作变成校方审核兼职信息的工作，减轻相关校方部门的工作压力，使得兼职发布更加正规有效。

巧兼职，是自主研发的领先的兼职管理平台，拥有“巧兼职”商家端、大学生移动端还有平台、校方的后台管理端。商家提供的兼职信息经过“巧兼职”平台，直接发布到相对应的各大高校的后台管理端，经过校方管理进行审核后成功进行发布。大学生手持“巧兼职”APP，通过系统智能推荐或者自由游览等方式，获取到心目中的最理想的兼职，最后点击报名即可成功预约。整个流程中如果发生了任何的矛盾，都可通过平台的投诉服务尽心投诉，平台在工作时间会立刻处理，杜绝各种不良事件发生。

## 项目组成员

|  |  |  |
| --- | --- | --- |
| 学号 | 姓名 | 职责 |
| 201606110062 | 李利光 | 组长 |
| 201606110064 | 李志成 | 队员 |
| 201606110079 | 黄扬帆 | 队员 |
| 201606120025 | 周志勇 | 队员 |
| 201608030040 | 陈婷婷 | 队员 |

## 使用说明

项目基于Spring Boot 2.1开发，以jar方式打包。

在提供源代码的情况下，可以通过Intellj IDEA打开项目，并运行DemoApplication，然后打开浏览器访问：<http://localhost>即可。

如果没有源代码的情况下，可以通过javac命令运行jar包，在程序启动成功后，在浏览器输入：<http://localhost>即可访问。

# 项目任务安排

李利光：需求分析，数据库设计，原型设计，登录注册功能设计，首 页交互设计，模块整合，发布兼职功能设计，学生信息导入。

李志成：首页页面设计，注册页面设计，注册前端交互设计，兼职审 核前后端功能实现。

黄扬帆：原型设计，兼职发布页面设计，编写项目文档，个人资料设 置。

周志勇：兼职详情前后端功能，申诉前后台功能。

陈婷婷：登录页面设计，ppt制作，密码修改。

## 模块需求

### 用户管理模块

#### 概述

用户管理包括管理员对校方用户管理，校方对商家用户管理。实现对用户基本信息、扩展信息、角色信息、权限信息的基本管理。管理员可以批量导入校方用户信息，可以批量导入商家用户信息，校方可以批量导入商家信息，商家、学生可以管理自己的个人信息。

### 用户功能模块

### 1、兼职审核管理

#### 概述

兼职审核是校方用户对商家商家发布的兼职信息的各类分析审核，包括兼职安全问题、兼职时间、兼职地点等方面信息的审核。相邻学校之间可以确定合作关系达到兼职信息共享、审核共审的目的。

### 2、兼职发布管理

#### 概述

兼职发布模块是实现商家在平台上发布兼职信息，然后这些兼职信息会由校方用户进行审核，审核通过之后这些信息会发布到兼职平台上，根据一些兼职需要推送给适合需要这类兼职的学生用户。

### 兼职查找

**概述**

兼职查找是针对学生用户的需求开发的，在学生用户在平台首页根据自己要查找的兼职而去显示学生需要的兼职，而不是一页一页的去寻找适合自己的兼职信息，浪费时间在查找兼职的事情上。

### 4、申诉管理

#### 概述

申诉管理是实现了针对兼职过程中的发生各种不符合兼职的事物的后台管理，例如兼职时间、兼职工作量等不符合兼职信息上的信息。也是类似于售后功能，专门处理兼职工作上发生的问题。

## 非功能性需求

### 扩展性需求

1、增加了课表导入功能，学生用户输入学号和密码，就会自动导入学生课表，这样子什么时候有课什么时候没有课系统一目了然，这样子更能优先推送兼职信息给有空闲时间的学生用户。

2、校方用户审核兼职，这样子的兼职信息更加可靠。

### 界面需求

非管理类功能实现同时支持PC类、智能手机类终端进行操作。所有操作必须有明确、及时的结果反馈给用户，比如等待、操作成功、操作失败等。

### 交互性

我们的“巧兼职”对于学生有兼职信息的推送功能，能让学生登录就能看到自己需要的兼职，能让学生用户找到最适合自己的兼职。对于商家用户来说有学生信息推荐功能，推荐一些符合兼职要求的的学生给商家，商家用户也能最快找到合适自己的兼职员工，实现了商家用户和学生用户能节省时间去做更多的事情。

# 设计方案

## 用户管理模块

### 用户注册功能

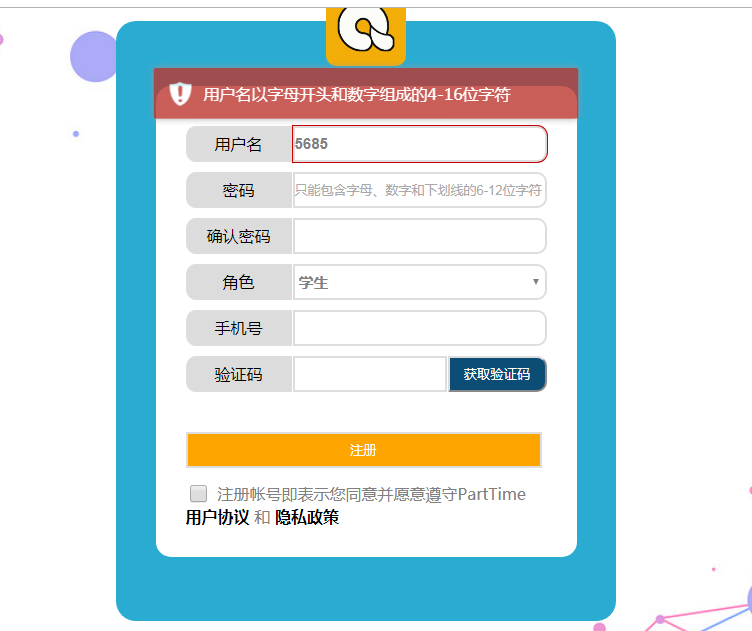
**输入：**用户名、密码、手机号、验证码和选择角色

**处理：**验证用户输入的信息是否正确

**输出：**注册成功跳转到首页面

**界面**

**注册页面**



#### 关键代码

Shiro关键代码

*/\*\*  
 \* Shiro的过滤器链  
 \*/*@Bean  
**public** ShiroFilterFactoryBean shiroFilter(DefaultWebSecurityManager securityManager) {  
 ShiroFilterFactoryBean shiroFilter = **new** ShiroFilterFactoryBean();  
 shiroFilter.setSecurityManager(securityManager);  
 */\*\*  
 \* 默认的登陆访问url  
 \*/* shiroFilter.setLoginUrl(**"/spark/login"**);  
 */\*\*  
 \* 登陆成功后跳转的url  
 \*/* shiroFilter.setSuccessUrl(**"/spark/index"**);  
 */\*\*  
 \* 没有权限跳转的url  
 \*/* shiroFilter.setUnauthorizedUrl(**"/global/error"**);  
  
 */\*\*  
 \* 覆盖默认的user拦截器(默认拦截器解决不了ajax请求 session超时的问题,若有更好的办法请及时反馈作者)  
 \*/* HashMap<String, Filter> myFilters = **new** HashMap<>();  
 myFilters.put(**"user"**, **new** GunsUserFilter());  
 shiroFilter.setFilters(myFilters);  
  
 */\*\*  
 \* 配置shiro拦截器链  
 \*  
 \* anon 不需要认证  
 \* authc 需要认证  
 \* user 验证通过或RememberMe登录的都可以  
 \*  
 \* 当应用开启了rememberMe时,用户下次访问时可以是一个user,但不会是authc,因为authc是需要重新认证的  
 \*  
 \* 顺序从上到下,优先级依次降低  
 \*  
 \* api开头的接口，走rest api鉴权，不走shiro鉴权  
 \*  
 \*/* Map<String, String> hashMap = **new** LinkedHashMap<>();  
 hashMap.put(**"/static/\*\*"**, **"anon"**);  
 hashMap.put(**"/spark/index"**, **"anon"**);  
 hashMap.put(**"/spark/login"**, **"anon"**);  
 hashMap.put(**"/spark/register"**, **"anon"**);  
 hashMap.put(**"/spark/mgr/add"**, **"anon"**);  
 hashMap.put(**"/spark/partTime/page"**, **"anon"**);  
 hashMap.put(**"/gunsApi/\*\*"**, **"anon"**);  
 hashMap.put(**"/login"**, **"anon"**);  
 hashMap.put(**"/global/sessionError"**, **"anon"**);  
 hashMap.put(**"/kaptcha/\*\*"**, **"anon"**);  
 hashMap.put(**"/\*\*"**, **"user"**);  
 shiroFilter.setFilterChainDefinitionMap(hashMap);  
 **return** shiroFilter;  
}

FormValation核心代码

**validateFields**: {  
 **partTimeType**: {  
 **validators**: {  
 **notEmpty**: {  
 **message**: **'兼职类型不能为空'** }  
 }  
 },  
 **partTimeTitle**: {  
 **validators**: {  
 **notEmpty**: {  
 **message**: **'兼职标题不能为空'** }  
 }  
 },  
 **recruiterNumber**: {  
 **validators**: {  
 **notEmpty**: {  
 **message**: **'兼职人数不能为空'** },  
 **integer**: {  
 **message**: **'兼职人数只能为整数且必須大于0'**,  
 },  
 *// callback: {  
 // message:'',  
 // callback: function(input) {  
 // return parseInt(input)>0;  
 // }  
 // },* }  
 }

*/\*\*  
 \* 验证数据是否为空  
 \*/****ReleaseInfo***.validate = **function** () {  
 **$**(**'#releaseInfoForm'**).data(**"bootstrapValidator"**).resetForm();  
 **$**(**'#releaseInfoForm'**).bootstrapValidator(**'validate'**);  
 **return $**(**"#releaseInfoForm"**).data(**'bootstrapValidator'**).isValid();  
};

**$**(**function** () {  
 ***Feng***.initValidator(**"releaseInfoForm"**, ***ReleaseInfo***.**validateFields**);  
  
  
});

initValidator: **function** (formId, fields) {  
 **$**(**'#'** + formId).bootstrapValidator({  
 **feedbackIcons**: {  
 **valid**: **'glyphicon glyphicon-ok'**,  
 **invalid**: **'glyphicon glyphicon-remove'**,  
 **validating**: **'glyphicon glyphicon-refresh'** },  
 **fields**: fields,  
 **live**: **'enabled'**,  
 **message**: **'该字段不能为空'** });  
},

**亮点：**

权限配置，通过shiro实现rabc（基于角色的权限访问控制）可以更加细粒的把控角色的权限。

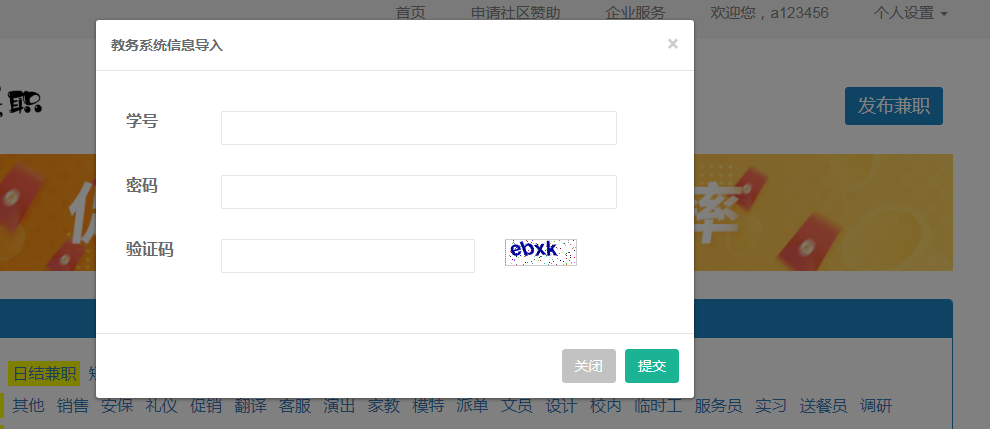
### 课表导入功能

**输入：**用户名学号、用户密码进行验证

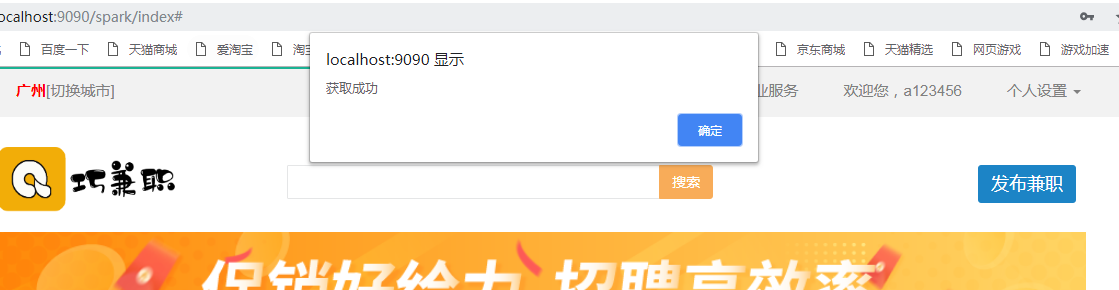
**输出：**导入成功跳转到首页面

**界面**

**课表导入页面**



**导入成功**



**关键代码**

*/\*\*  
 \* 登陆  
 \** ***@throws*** *IOException  
 \*/***public void** load() **throws** IOException {  
 System.***out***.println(**"check = "** + **check**);  
 *//拼接请求字符串* String str = **"\_\_VIEWSTATE="** + URLEncoder.*encode*(**viewState**, **"gb2312"**) + **"&txtUserName="** + **studentID** + **"&TextBox2="** + **studentPassword** + **"&txtSecretCode="** + **check** + **"&RadioButtonList1="** + URLEncoder.*encode*(**"学生"**, **"gb2312"**)  
 + **"&Button1=&lbLanguage=&hidPdrs=&hidsc="**;  
 ;  
 System.***out***.println(**"参数列表:"** + str);  
 *//登录提交的网址* URL url = **new** URL(**submitURL**);  
 HttpURLConnection conn = (HttpURLConnection) url.openConnection();  
 *//Post提交* conn.setRequestMethod(**"POST"**);  
 conn.setReadTimeout(5000);  
 conn.setUseCaches(**false**);  
 *//禁止程序自己跳转到目标网址，必须设置，不然程序会自己响应  
 //302返回码，自己请求跳转后的网址，出现Object Had Moved!错误* conn.setInstanceFollowRedirects(**false**);  
 *// 写入cookie* conn.setRequestProperty(**"Cookie"**, **cookie**);  
 conn.setDoOutput(**true**);  
 OutputStream out = conn.getOutputStream();  
 *//写入参数* out.write(str.getBytes());  
 out.close();  
 *//打印返回码* System.***out***.println(**"返回码:"** + conn.getResponseCode());  
 *//打印服务器返回的跳转网址* **Location** = conn.getHeaderField(**"Location"**);  
 System.***out***.println(**"Location :"** + **Location**);  
 BufferedReader read = **new** BufferedReader(**new** InputStreamReader(conn.getInputStream()));  
 String temp;  
 *//读取页面源码* StringBuffer ab = **new** StringBuffer();  
 **while** ((temp = read.readLine()) != **null**) {  
 ab.append(temp);  
 }  
 **if** (conn.getResponseCode() != 302) {  
 *//getError为自定义函数，提取出错误信息* System.***out***.println(**"错误"**);  
 **return**;  
 }  
}

### 用户登陆功能

#### 输入：用户名和用户密码

#### 处理：验证用户输入的信息是否正确

#### 输出：登录成功跳转到首页

#### 界面

**错误提示页面**





#### 关键代码







## 3.2兼职功能模块

### （1）发布兼职功能

#### 输入：输入相关的兼职信息，例如工作时间、工作地点等等

#### 处理：验证用户输入的信息是否正确

#### 输出：显示发布成功

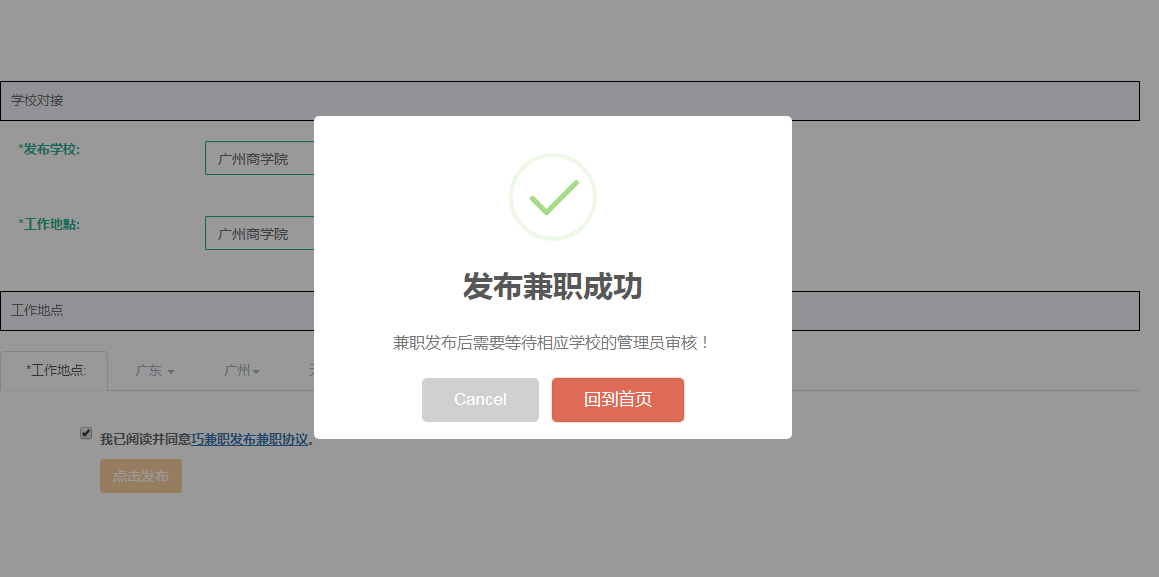
#### 界面

**兼职发布页面**





**发布兼职成功页面**



**关键代码**

**算法（可选）**

#### 亮点（可选）

### 兼职审核功能

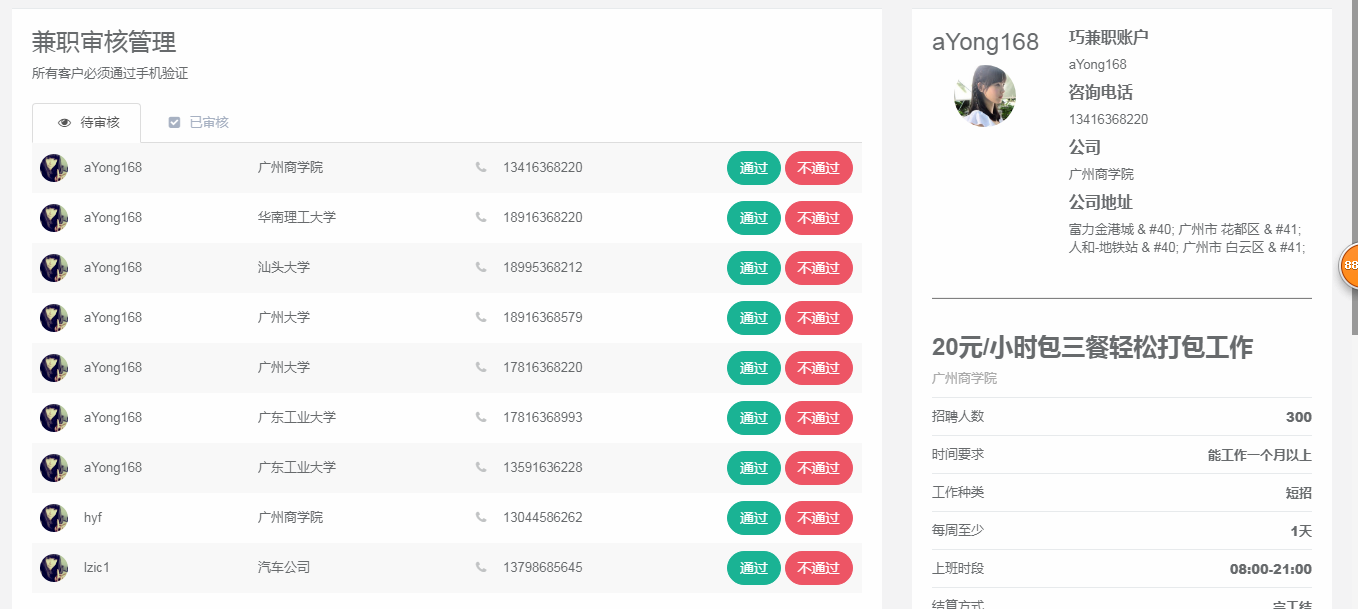
**输入：**输入校方用户账号密码，登录到审核页面。

#### 处理：审核兼职的信息可靠性、安全性等方面的问题。

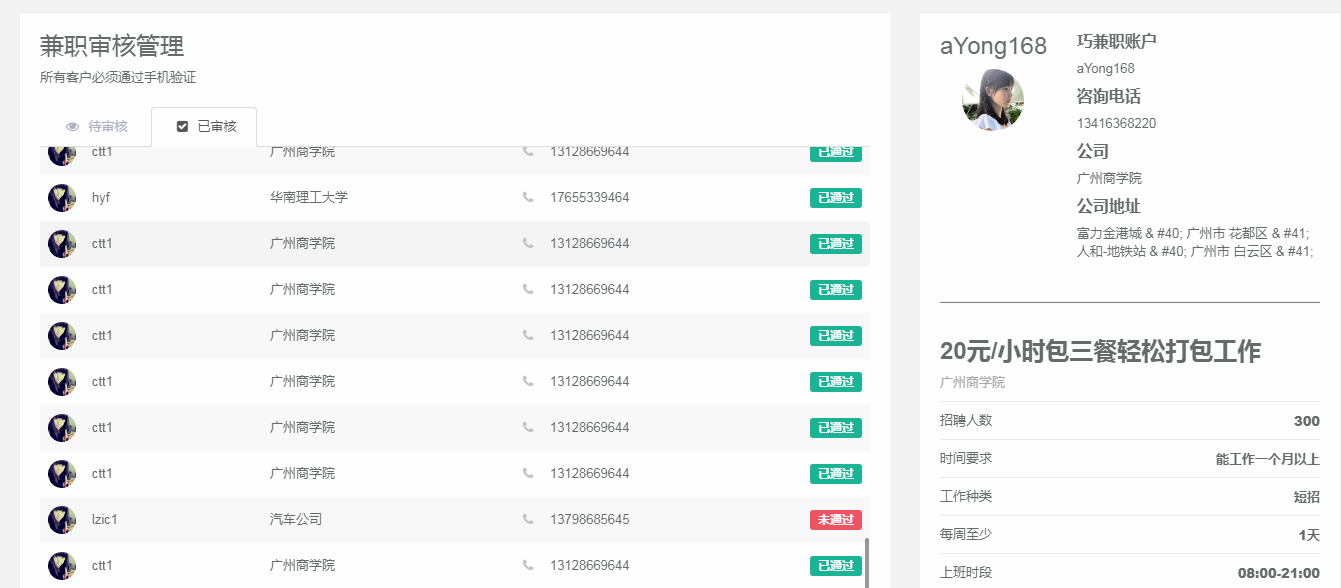
#### 输出：可以同意或者不同意兼职信息的发布。

#### 界面

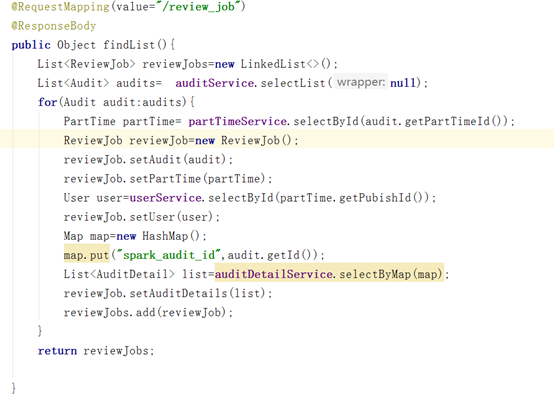
**待审核兼职页面**



**已审核的兼职页面**



**关键代码**

****

****

**算法（可选）**

#### 亮点（可选）

### 兼职查找功能

**输入：**在平台首页可以查看到各种各样的兼职信息，学生用户以根据自己的要求去搜索兼职信息。

#### 输出：根据学生用户的搜索要求显示出兼职信息。

#### 界面

**首页兼职信息**





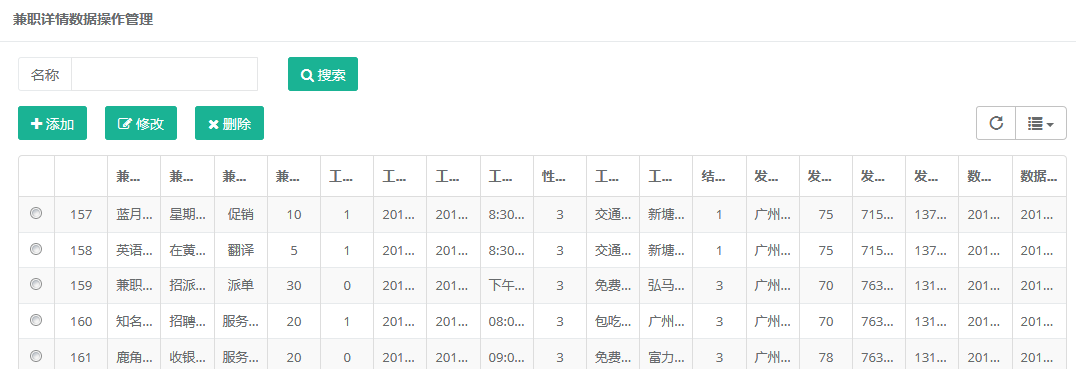
**关键代码**

### 兼职详情功能

#### 处理：点击主页的兼职信息可进入此兼职的详细内容，该数据是从数据库取出渲染。兼职信息可以在后台进行增删改查。

#### 界面





**关键代码**

兼职详情管理（增删改查）

*/\*\*  
 \* 获取（和查找）兼职详情数据操作列表  
 \*/* @RequestMapping(value = **"/list"**)  
 @ResponseBody  
 **public** Object list(String condition) {  
 **if**(ToolUtil.*isNotEmpty*(condition)){  
 EntityWrapper<PartTime> partTimeDetailEntityWrapper = **new** EntityWrapper<>();  
 Wrapper<PartTime> type = partTimeDetailEntityWrapper.like(**"part\_time\_type"**,**"%"**+condition+**"%"**);  
 List<PartTime> partTime = **partTimeService**.selectList(type);  
  
  
 **return** partTime;  
 }**else** {  
 List<PartTime> partTimeDetails = **partTimeService**.selectList(**null**);  
 **return** partTimeDetails;  
 }  
 }  
  
 */\*\*  
 \* 分页获取（和查找）兼职详情数据操作列表  
 \*/* @RequestMapping(value = **"/page"**)  
 @ResponseBody  
 **public** Object page() {  
 Page<PartTime> page = **new** PageFactory<PartTime>().defaultPage();  
 **return** page.setRecords(**partTimeService**.getAuditPartTime(page));  
 }  
  
 */\*\*  
 \* 新增兼职详情数据操作  
 \*/* @RequestMapping(value = **"/add"**)  
 @ResponseBody  
 **public** Object add(PartTime partTime) {  
 **if**(partTime != **null**){  
 partTime.setGmtCreate(**new** Date());  
 }  
 **partTimeService**.insert(partTime);  
 **return** *SUCCESS\_TIP*;  
 }  
  
 */\*\*  
 \* 新增兼职详情数据操作  
 \*/* @RequestMapping(value = **"/publish"**)  
 @Transactional  
 @ResponseBody  
 **public** String publish(PartTime partTime,String[] workWelfares) {  
 *//填充剩余数据* StringBuffer workWelf=**new** StringBuffer();  
 **for**(**int** i=0;i<workWelfares.**length**;i++){  
 workWelf.append(workWelfares[i]);  
 **if**(i<workWelfares.**length**-1) workWelf.append(**","**);  
 }  
 partTime.setWorkWelfare(workWelf.toString());  
 partTime.setGmtCreate(**new** Date());  
 partTime.setGmtModified(**new** Date());  
 partTime.setPubishId(ShiroKit.*getUser*().**id**);  
 **partTimeService**.insert(partTime);  
 Audit audit=**new** Audit();  
 audit.setStatus(1);  
 audit.setPartTimeId(partTime.getId());  
 audit.setGmtCreate(**new** Date());  
 audit.setGmtModified(**new** Date());  
 **iAuditService**.insert(audit);  
 **return "/spark/index.html"**;  
 }  
  
 */\*\*  
 \* 删除兼职详情数据操作  
 \*/* @RequestMapping(value = **"/delete"**)  
 @ResponseBody  
 **public** Object delete(@RequestParam Integer partTimeId) {  
 **partTimeService**.deleteById(partTimeId.longValue());  
 **return** *SUCCESS\_TIP*;  
 }  
  
 */\*\*  
 \* 修改兼职详情数据操作  
 \*/* @RequestMapping(value = **"/update"**)  
 @ResponseBody  
 **public** Object update(PartTime partTime) {  
 **partTimeService**.updateById(partTime);  
 **return** *SUCCESS\_TIP*;  
 }

兼职详情model

@TableName(**"spark\_part\_time"**)  
**public class** PartTime **extends** Model<PartTime> {  
  
 **private static final long *serialVersionUID*** = 1L;  
  
 @TableId(value = **"id"**, type = IdType.***AUTO***)  
 **private** Long **id**;  
 */\*\*  
 \* 兼职标题  
 \*/* @TableField(**"part\_time\_title"**)  
 **private** String **partTimeTitle**;  
 */\*\*  
 \* 兼职内容  
 \*/* @TableField(**"part\_time\_content"**)  
 **private** String **partTimeContent**;  
 */\*\*  
 \* 兼职类型  
 \*/* @TableField(**"part\_time\_type"**)  
 **private** String **partTimeType**;  
 */\*\*  
 \* 兼职招聘人数  
 \*/* @TableField(**"recruiter\_number"**)  
 **private** Integer **recruiterNumber**;  
 */\*\*  
 \* 工作种类 0 短招 1长招  
 \*/* @TableField(**"work\_type"**)  
 **private** Integer **workType**;  
 */\*\*  
 \* 工作开始时间  
 \*/* @TableField(**"work\_start\_time"**)  
 @DateTimeFormat(pattern = **"yyyy-MM-dd"**)  
 **private** Date **workStartTime**;  
 */\*\*  
 \* 工作结束时间  
 \*/* @TableField(**"work\_end\_time"**)  
 @DateTimeFormat(pattern = **"yyyy-MM-dd"**)  
 **private** Date **workEndTime**;  
 */\*\*  
 \* 工作时间范围，一天可能有多个非连续时间段，用‘ ，’分割  
 \*/* @TableField(**"work\_time\_range"**)  
 **private** String **workTimeRange**;  
 */\*\*  
 \* 性别要求（1：男 2：女 3: 男+女）  
 \*/* @TableField(**"gender\_requirement"**)  
 **private** Integer **genderRequirement**;  
 */\*\*  
 \* 工资  
 \*/* @TableField(**"work\_salary"**)  
 **private** String **workSalary**;  
 */\*\*  
 \* 工作福利，每项福利用逗号分隔  
 \*/* @TableField(**"work\_welfare"**)  
 **private** String **workWelfare**;  
 */\*\*  
 \* 工作地点  
 \*/* @TableField(**"work\_place"**)  
 **private** String **workPlace**;  
 */\*\*  
 \* 结算周期 1.日结 2.周结 3.月结 4.完工结  
 \*/* @TableField(**"settlement\_cycle"**)  
 **private** Integer **settlementCycle**;  
 */\*\*  
 \* 发布的学校对象  
 \*/* @TableField(**"publish\_school"**)  
 **private** String **publishSchool**;  
 */\*\*  
 \* 发布者id  
 \*/* @TableField(**"pubish\_id"**)  
 **private** Integer **pubishId**;  
 */\*\*  
 \* 发布兼职者的个人邮箱  
 \*/* @TableField(**"publish\_email"**)  
 **private** String **publishEmail**;  
 */\*\*  
 \* 发布兼职者的电话号码  
 \*/* @TableField(**"publish\_phone"**)  
 **private** String **publishPhone**;  
 */\*\*  
 \* 数据创建时间  
 \*/* @TableField(**"gmt\_create"**)  
 @DateTimeFormat(pattern = **"yyyy-MM-dd"**)  
 **private** Date **gmtCreate**;  
 */\*\*  
 \* 数据修改时间  
 \*/* @TableField(**"gmt\_modified"**)  
 @DateTimeFormat(pattern = **"yyyy-MM-dd"**)  
 **private** Date **gmtModified**;

### 兼职申诉功能

**输入：**在兼职申诉页面填写需要申诉的兼职。

#### 处理：申诉内容传带后台，会有专门处理这类事情的人工去审核。

#### 输出：经过人工核实之后进行处理申诉要求。向诉求人发送申诉 结果

#### 界面

**申诉页面**



**申诉列表页面**



**关键代码**

上传显示凭证js

<**div class="form-group"**>  
 <**label class="col-sm-2 control-label"**><**em**>\*</**em**>上传凭证</**label**>  
 <**div class="col-sm-10"**>  
 <**input type="file" onchange="***show*(**this**)**" accept="image/gif,image/jpeg,image/png,image/jpg" id="upload" multiple="multiple"**>  
 <**script**>  
 **function** *show*(img) {  
 **imgFile** = img;  
 **var** allFiles = **imgFile**.files;  
 **for** (**var** i = 0; i < allFiles.length; i++) {  
 **var** img = allFiles[i];  
 **$**(**"img"**).remove();  
 **var** readImg = **new *FileReader***();  
 readImg.readAsDataURL(img);  
 readImg.onload = **function** (readEvent) {  
 **var** base64 = readEvent.target.result;  
 **var** appd = **"<img src=\""** + base64 + **"\" width=\"200px\" height=\"200px\">"**;  
 **$**(**"#image"**).append(appd);  
 };  
 }  
 }  
 </**script**>  
 </**div**>  
</**div**>

申诉管理（增删改查）

*/\*\*  
 \* 查看(获取)申诉列表  
 \*/* @RequestMapping(value = **"/list"**)  
 @ResponseBody  
 **public** Object list(String condition) {  
 **if**(ToolUtil.*isNotEmpty*(condition)){  
 EntityWrapper<Complaint> complaintEntityWrapper = **new** EntityWrapper<>();  
 Wrapper<Complaint> type = complaintEntityWrapper.like(**"status"**,**"%"**+condition+**"%"**);  
 List<Complaint> complaints = **complaintService**.selectList(type);  
 **return** complaints;  
 }**else** {  
 List<Complaint> complaints = **complaintService**.selectList(**null**);  
 **return** complaints;  
 }  
*// return complaintService.selectList(null);* }  
  
 */\*\*  
 \* 新增申诉  
 \*/* @RequestMapping(value = **"/add"**)  
 @ResponseBody  
 **public** Object add(Complaint complaint) {  
 **if**(complaint != **null**){  
 complaint.setGmtCreate(**new** Date());  
 }  
 **complaintService**.insert(complaint);  
 **return** *SUCCESS\_TIP*;  
 }  
  
 */\*\*  
 \* 删除申诉  
 \*/* @RequestMapping(value = **"/delete"**)  
 @ResponseBody  
 **public** Object delete(@RequestParam Integer complaintId) {  
 **complaintService**.deleteById(complaintId.longValue());  
 **return** *SUCCESS\_TIP*;  
 }  
  
 */\*\*  
 \* 修改申诉  
 \*/* @RequestMapping(value = **"/update"**)  
 @ResponseBody  
 **public** Object update(Complaint complaint) {  
 **complaintService**.updateById(complaint);  
 **return** *SUCCESS\_TIP*;  
 }

申诉管理model

@TableName(**"spark\_complaint"**)  
**public class** Complaint **extends** Model<Complaint> {  
  
 **private static final long *serialVersionUID*** = 1L;  
  
 @TableId(value = **"id"**, type = IdType.***AUTO***)  
 **private** Long **id**;  
 */\*\*  
 \* 申诉标题  
 \*/* @TableField(**"title"**)  
 **private** String **title**;  
 */\*\*  
 \* 申诉内容  
 \*/* @TableField(**"content"**)  
 **private** String **content**;  
 */\*\*  
 \* 申诉凭证（图片），最多9张，保存的地址用逗号隔开  
 \*/* @TableField(**"voucher"**)  
 **private** String **voucher**;  
 */\*\*  
 \* 1 未处理 2 正在处理 3已处理  
 \*/* @TableField(**"status"**)  
 **private** Integer **status**;  
 */\*\*  
 \* 申诉发起者的id  
 \*/* @TableField(**"complainants\_id"**)  
 **private** Integer **complainantsId**;  
 */\*\*  
 \* 数据创建时间  
 \*/* @TableField(**"gmt\_create"**)  
 @DateTimeFormat(pattern = **"yyyy-MM-dd"**)  
 **private** Date **gmtCreate**;  
 */\*\*  
 \* 数据修改时间  
 \*/* @TableField(**"gmt\_modified"**)  
 @DateTimeFormat(pattern = **"yyyy-MM-dd"**)  
 **private** Date **gmtModified**;