项目名称：

通用后台权限管理系统

项目目标：

1. 掌握软件开发的基本流程
2. 掌握springboot框架
3. 掌握mybatis框架各知识点
4. 掌握shiro安全框架的认证和授权
5. 掌握html和javascript各知识点
6. 掌握mysql各知识点

项目大纲：

1. 背景知识
2. 需求分析
3. 功能分析
4. 架构设计
5. 代码开发

## 1、背景知识

在当前风起云涌的电商领域和广大企业信息化管理领域，软件系统的后台管理都是不可或缺的组成部分，而后台管理中的权限管理模块是衡量一个后台管理是否安全高效的重要指标。

## 2、需求分析

由于不同的后台管理系统需求多样化，本次开发的是通用型，对于大多数的后台管理系统逻辑都已足够使用，主要应用于WEB应用程序，如：网站管理后台、CMS、CRM、OA、商城后台等等。

目前常见的权限控制分为四大类：

1. **用户管理**

对于后台产品而言，是需要公司内部人员去创建账号的。而这个账号就是一把钥匙，我们通过控制账号所具备的权限，进而控制这个员工的所操作范围。

1. **角色管理**

角色往往是基于业务管理需求而预先在系统中设定好的固定标签，每个角色对应明确的系统权限，他是一个集合的概念，是众多最小权限颗粒的组成。我们通过把权限给这个角色，再把角色给账号，从而实现账号的权限，因此它承担了一个桥梁的作用。引入角色这个概念，可以帮助我们灵活的扩展，使一个账号可以具备多种角色。

1. **功能权限控制**

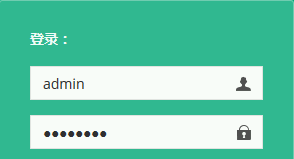
功能权限定义：为可见、可以操作的功能范围。在目录、菜单上加权限控制，有权限的就可以访问对应模块，没有的连菜单名都看不到。在业务模块的功能按钮上加权限控制，最小粒度的控制用户行为。

1. **数据权限控制**

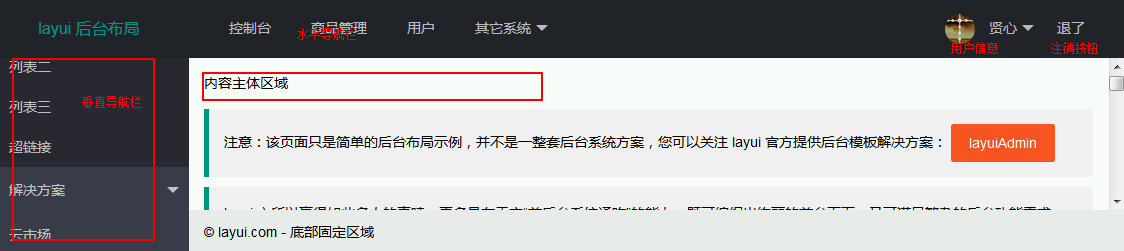
数据权限控制主要控制某条数据记录对用户是否可见，结合功能权限可以更灵活的配置业务过程中每一位员工的功能操作权限及数据可见范围，全面保障企业数据的安全性。

## 3、功能分析

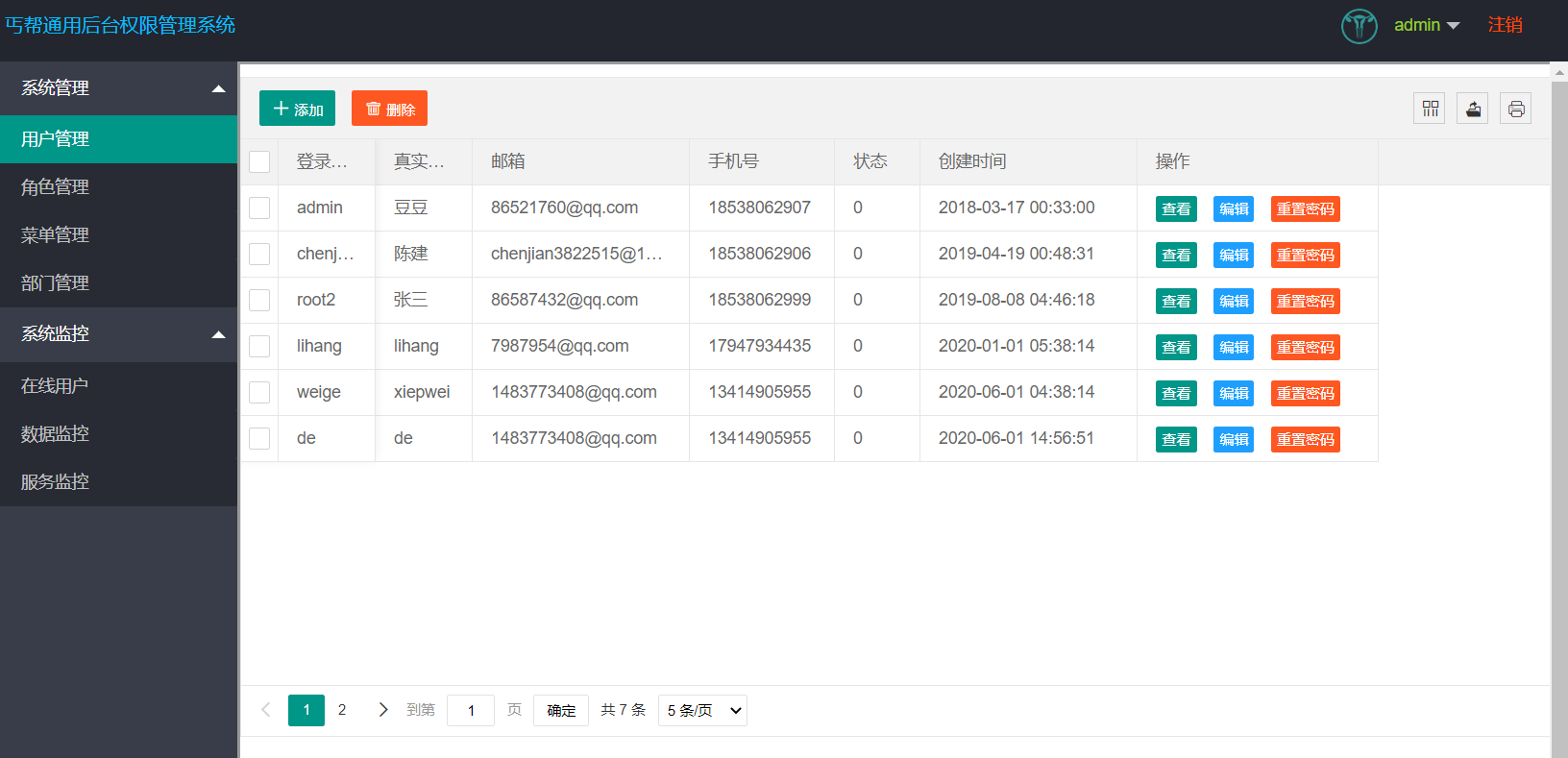
1. 登录功能
   1. 登录页面



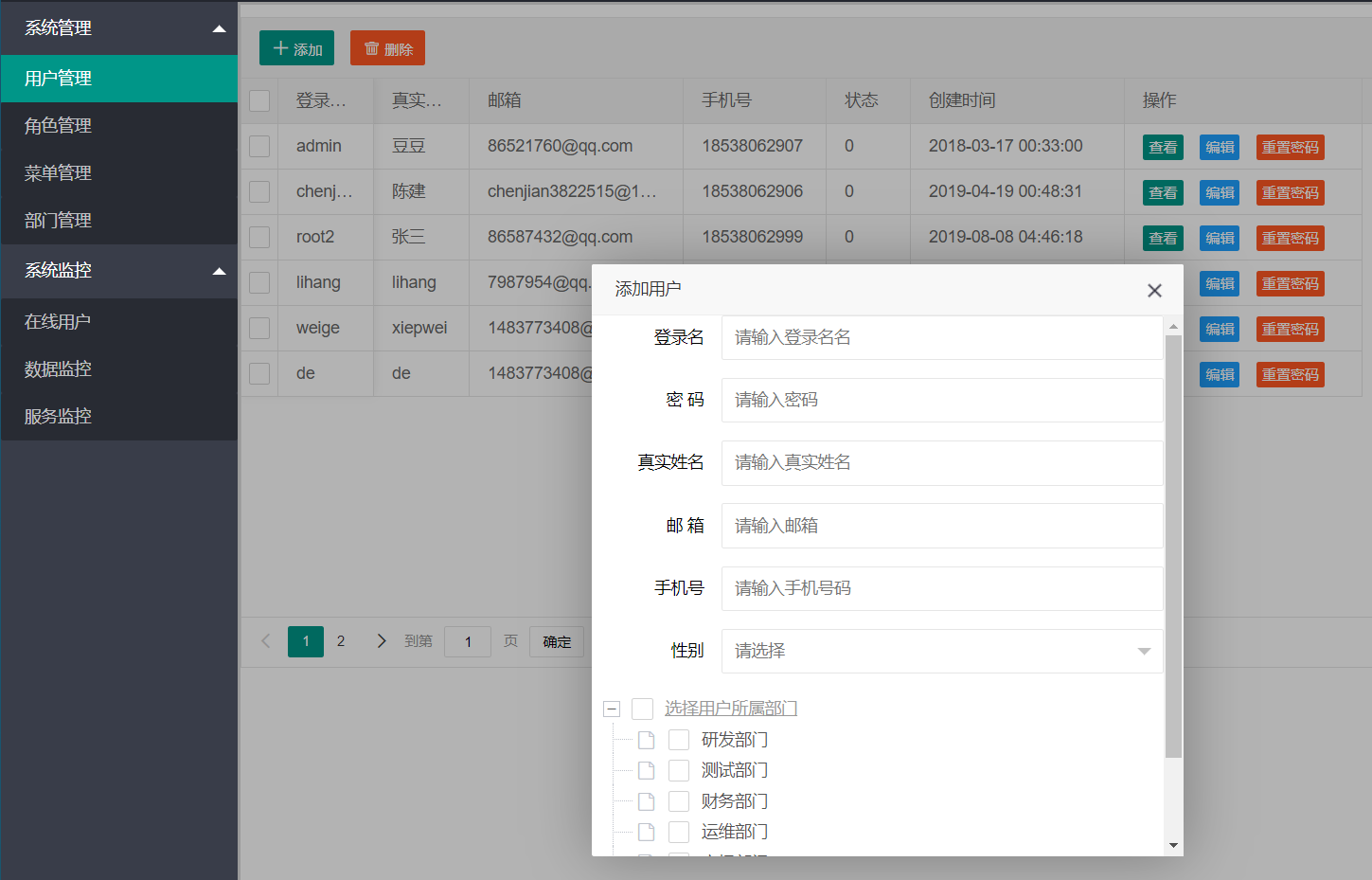
* 1. 异常登录提示错误信息



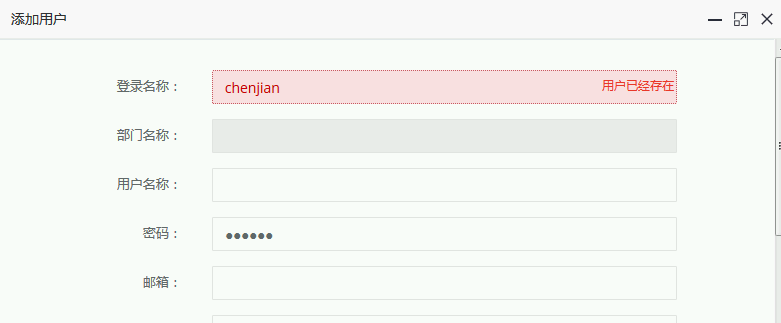
1. 用户管理
   1. 查询用户



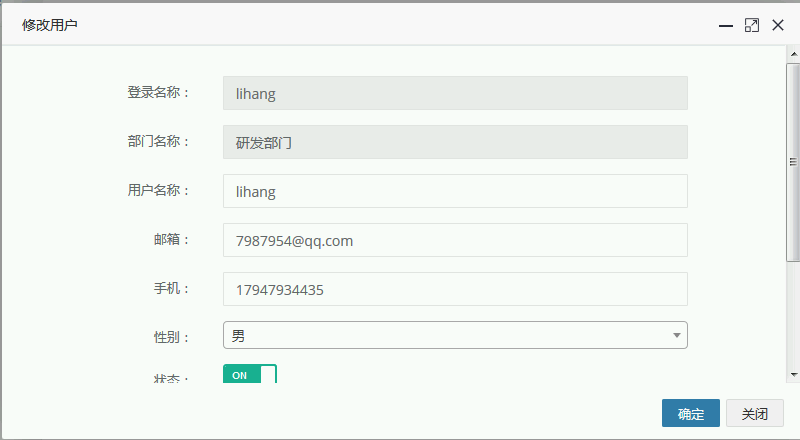
* 1. 添加用户，设置角色



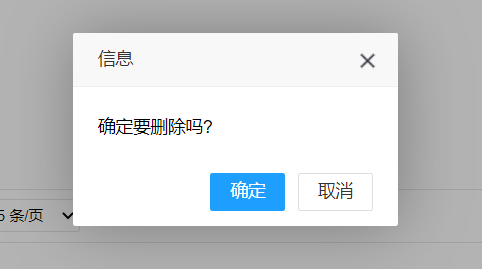
校验登录用户名是否存在onblur，layui的input校验



* 1. 修改用户，禁用用户



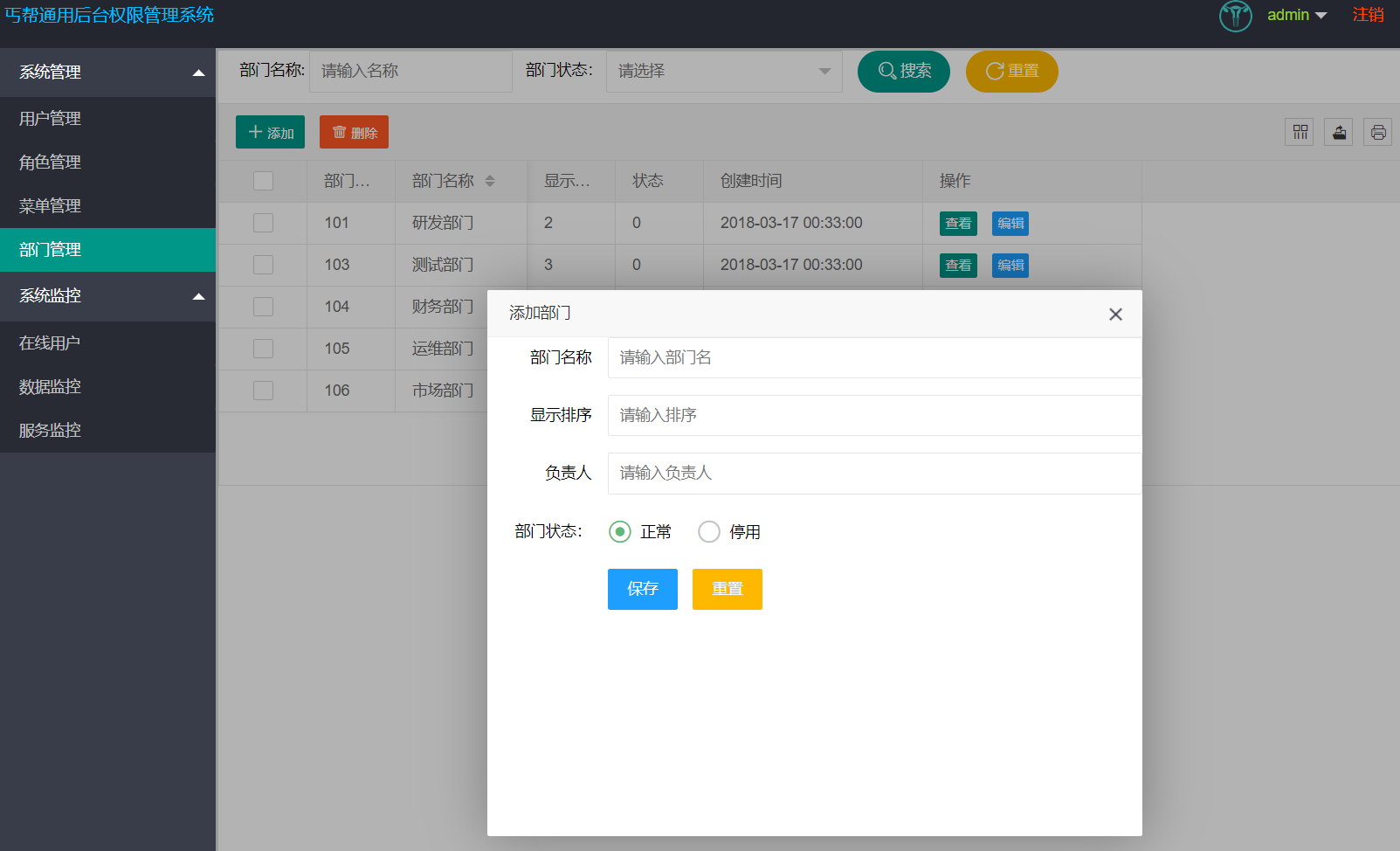
* 1. 删除用户



1. 部门管理
   1. 查询部门



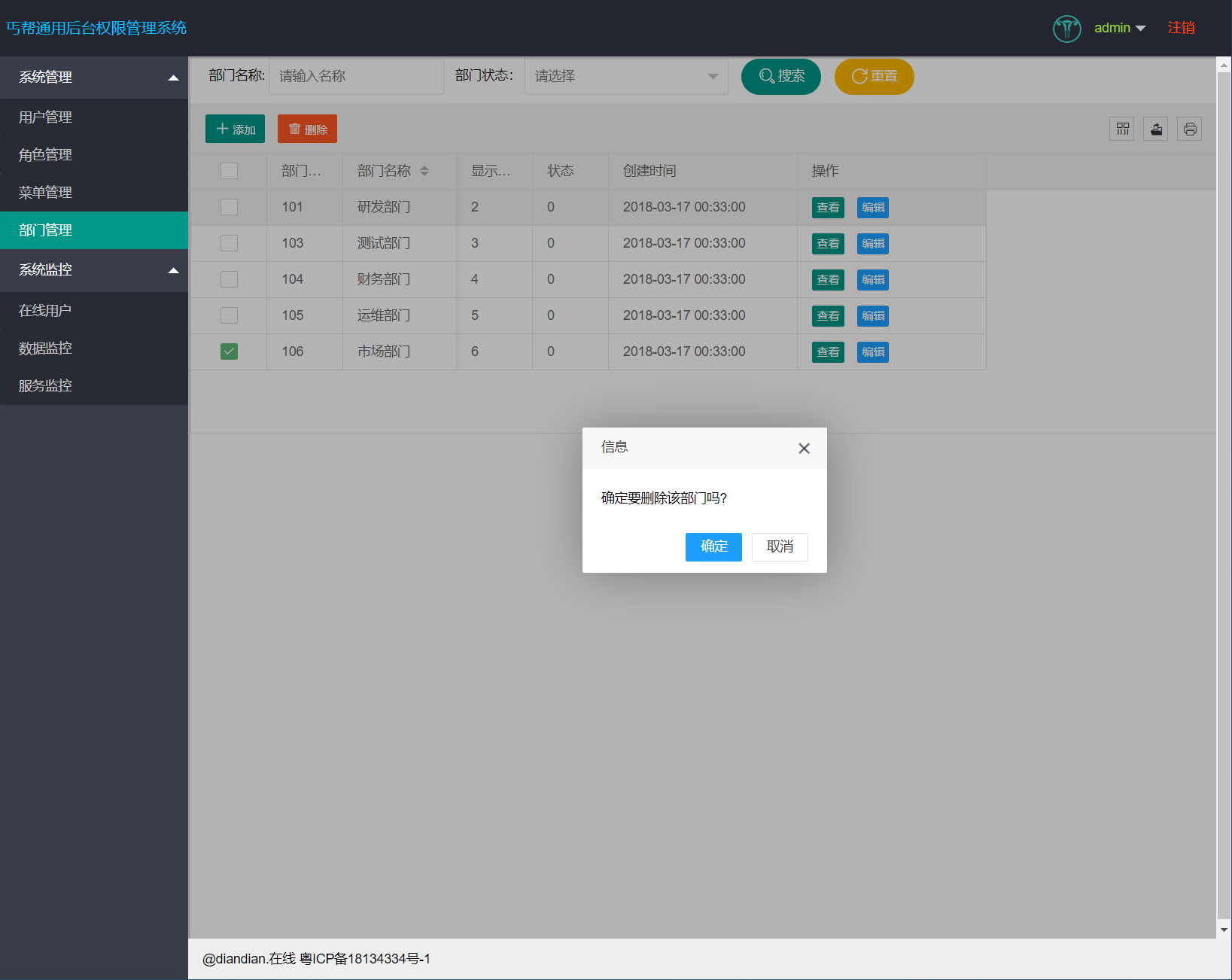
* 1. 添加部门



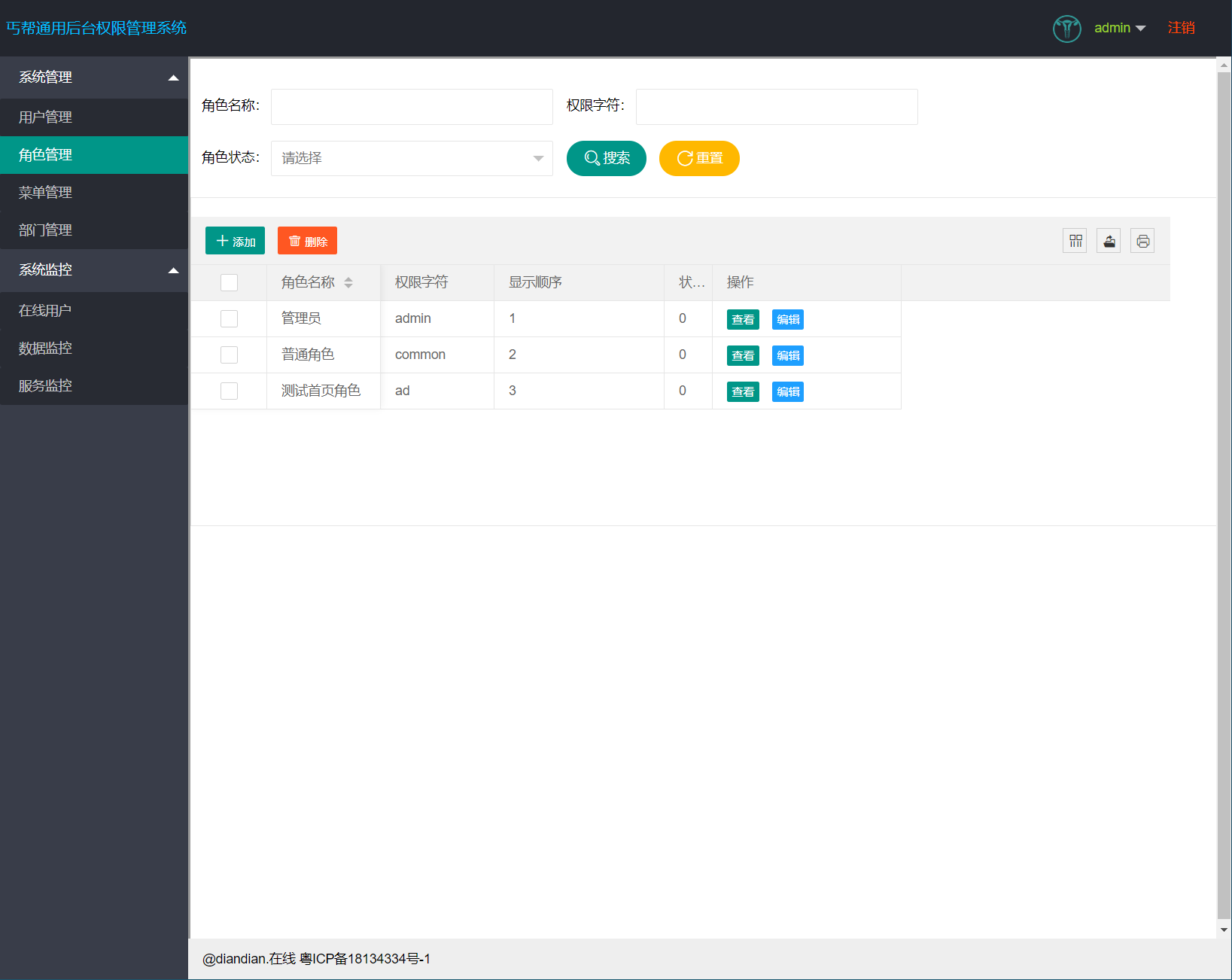
* 1. 修改部门，禁用部门



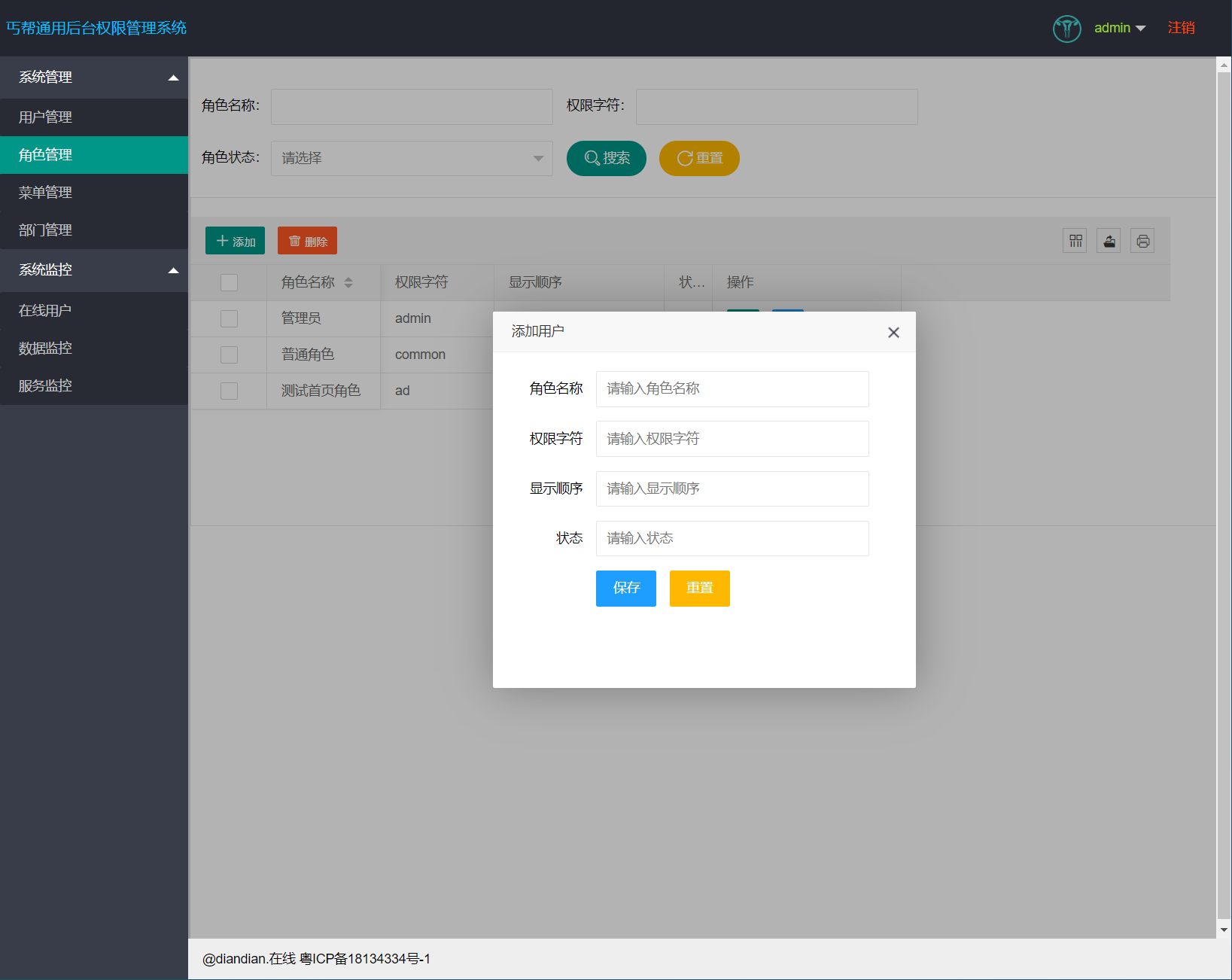
* 1. 删除部门



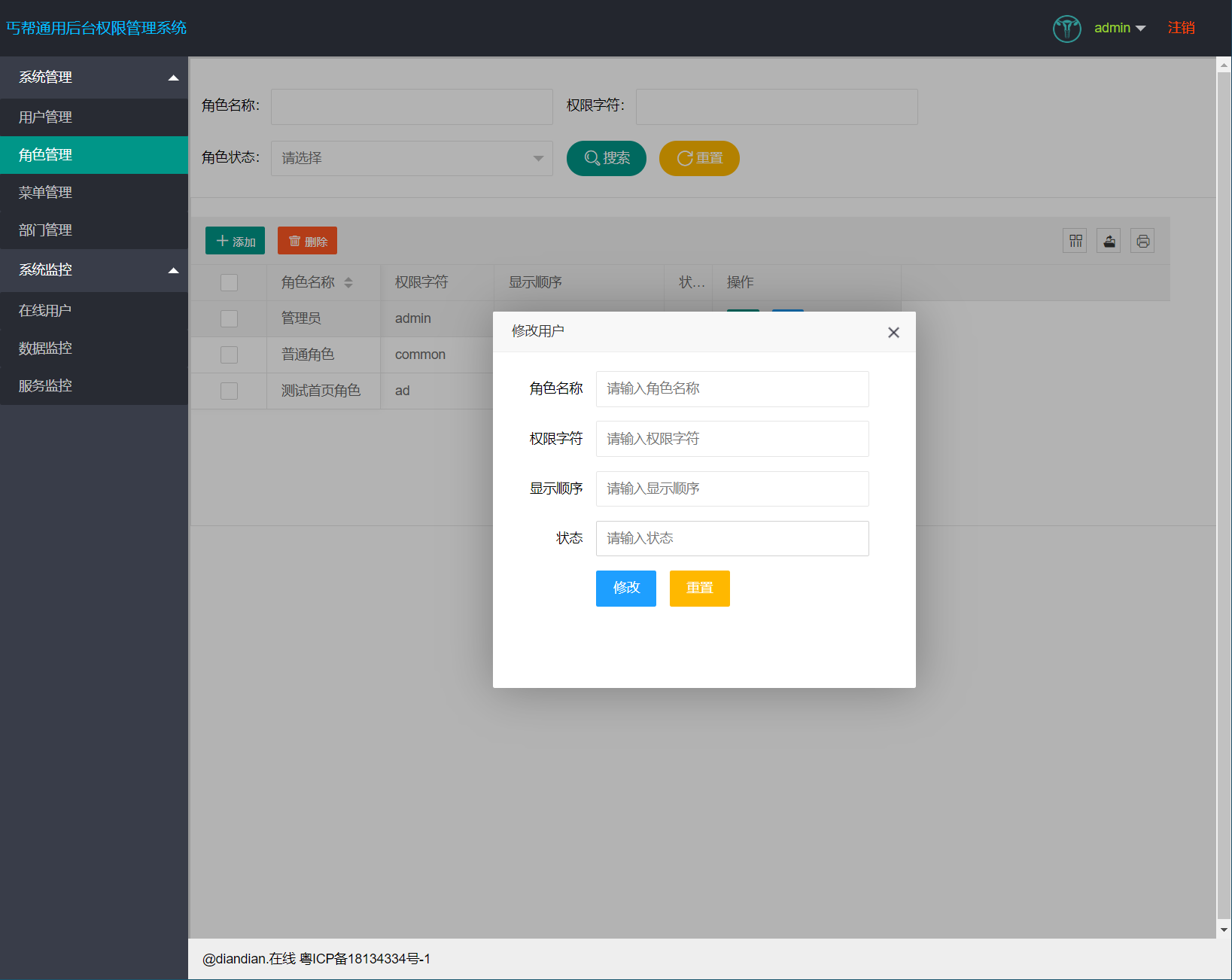
1. 角色管理
   1. 查询角色



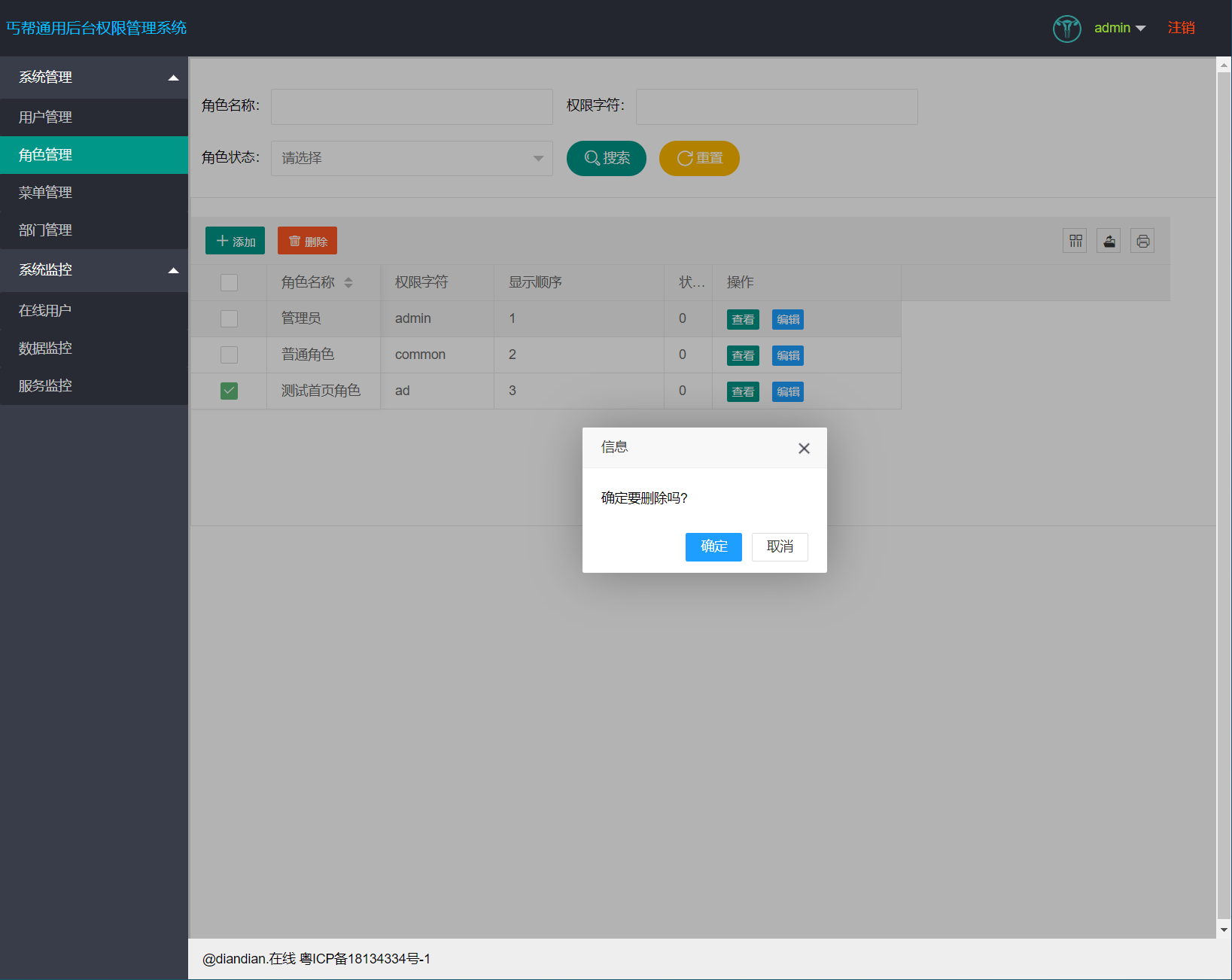
* 1. 添加角色



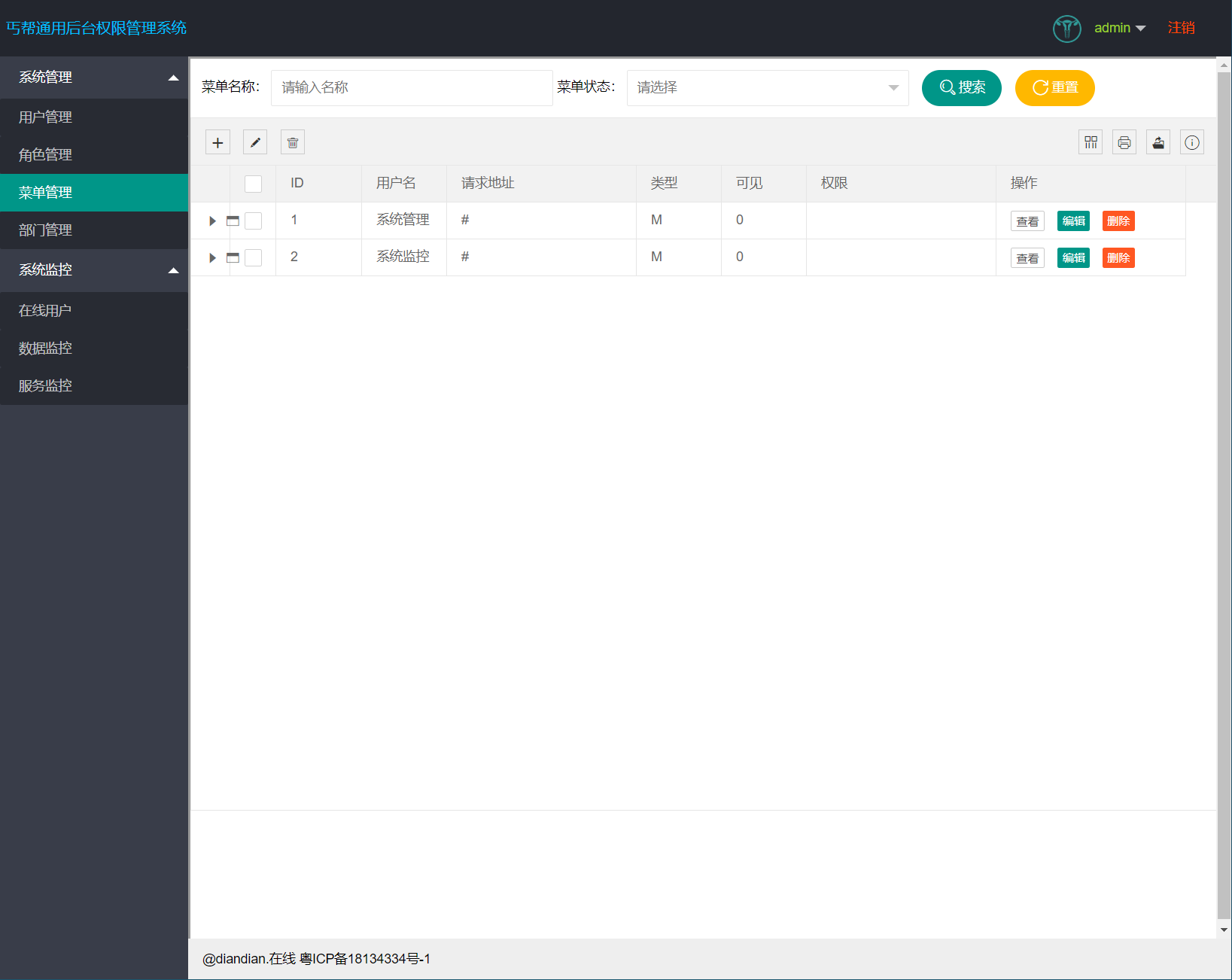
* 1. 修改角色



* 1. 删除角色



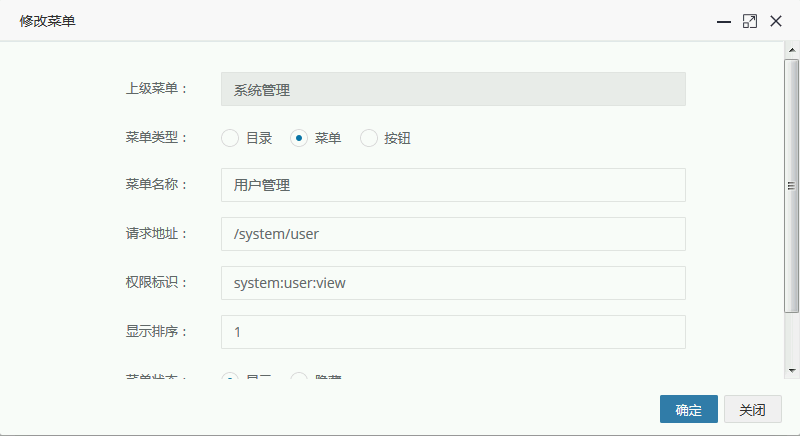
1. 菜单管理
   1. 查询菜单



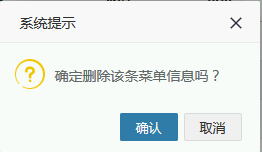
* 1. 添加菜单



* 1. 修改菜单



* 1. 删除菜单

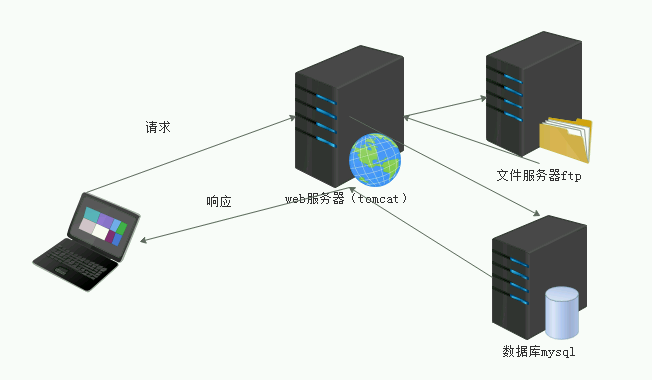


## 4、原型设计

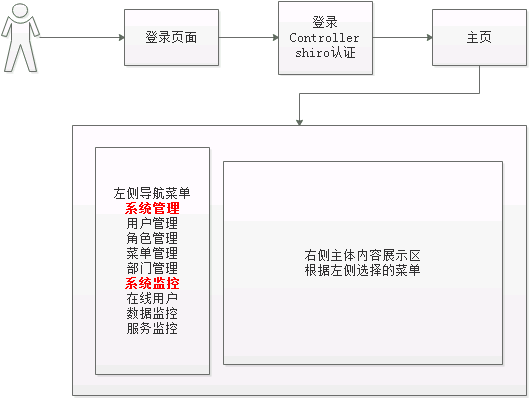
产品经理设计产品原形

## 5、架构设计

### 5.1、整体架构设计

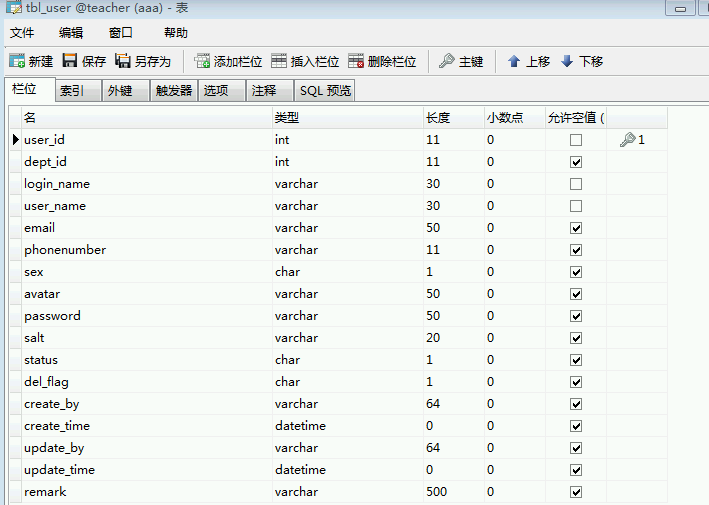


项目地址：<http://www.diandian.xn--3ds443g:8080/login>

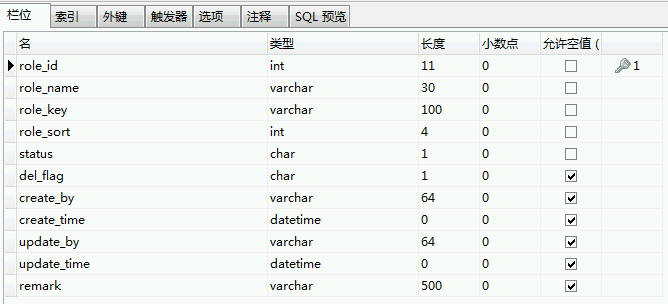


### 5.7、数据模型设计

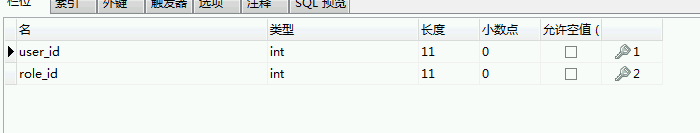
#### 5.7.1、用户表



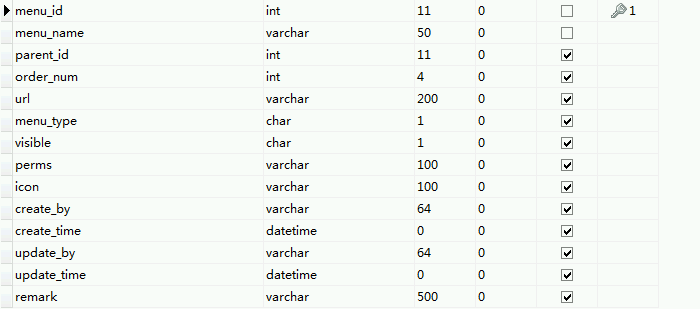
#### 5.7.2、角色表



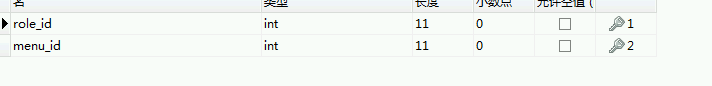
#### 5.7.3、用户角色表（中间表）



#### 5.7.4、菜单表



#### 5.7.5、角色菜单表



#### 5.7.6、部门表



# 表的清单

|  |  |
| --- | --- |
| 名称 | 代码 |
| tbl\_dept | tbl\_dept |
| tbl\_menu | tbl\_menu |
| tbl\_role | tbl\_role |
| tbl\_role\_menu | tbl\_role\_menu |
| tbl\_user | tbl\_user |
| tbl\_user\_role | tbl\_user\_role |

# 表栏的清单

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 代码 | 注释 | 数据类型 | 主要的 |
| user\_id | user\_id |  | int | TRUE |
| dept\_id | dept\_id |  | int | FALSE |
| login\_name | login\_name |  | varchar(30) | FALSE |
| user\_name | user\_name |  | varchar(30) | FALSE |
| email | email |  | varchar(50) | FALSE |
| phonenumber | phonenumber |  | varchar(11) | FALSE |
| sex | sex |  | char(1) | FALSE |
| user\_id | user\_id |  | int | TRUE |
| role\_id | role\_id |  | int | TRUE |
| menu\_id | menu\_id |  | int | TRUE |
| menu\_name | menu\_name |  | varchar(50) | FALSE |
| parent\_id | parent\_id |  | int | FALSE |
| order\_num | order\_num |  | int | FALSE |
| url | url |  | varchar(200) | FALSE |
| menu\_type | menu\_type |  | char(1) | FALSE |
| visible | visible |  | char(1) | FALSE |
| dept\_id | dept\_id |  | int | TRUE |
| dept\_name | dept\_name |  | varchar(30) | FALSE |
| order\_num | order\_num |  | int | FALSE |
| status | status |  | char(1) | FALSE |
| del\_flag | del\_flag |  | char(1) | FALSE |
| create\_by | create\_by |  | varchar(64) | FALSE |
| create\_time | create\_time |  | datetime | FALSE |
| role\_id | role\_id |  | int | TRUE |
| menu\_id | menu\_id |  | int | TRUE |
| role\_id | role\_id |  | int | TRUE |
| role\_name | role\_name |  | varchar(30) | FALSE |
| role\_key | role\_key |  | varchar(100) | FALSE |
| role\_sort | role\_sort |  | int | FALSE |
| status | status |  | char(1) | FALSE |
| del\_flag | del\_flag |  | char(1) | FALSE |
| create\_by | create\_by |  | varchar(64) | FALSE |

## 代码开发

开发难点：

1. 菜单管理

https://fly.layui.com/jie/30625/



权限菜单



<https://www.layui.com/demo/tree.html>

步骤一：根据json字符串封装java对象（gsonformat插件）

|  |
| --- |
| public class LayUiTree {   private String title;  private int id;  private String field;  private boolean checked;  private boolean spread;  private List<LayUiTree> children;   } |

步骤二：查询所有的菜单，并且组装成LayUiTree 对象

|  |
| --- |
| package com.aaa.util;   import com.aaa.entity.LayUiTree; import com.aaa.entity.Menu;  import java.util.ArrayList; import java.util.Iterator; import java.util.List;  */\*\*  \* 权限数据处理  \*   \** ***@author*** *teacherChen  \*/* public class TreeUtils {  */\*\*  \* 根据父节点的ID获取所有子节点  \*   \** ***@param*** *list 分类表  \** ***@param*** *parentId 传入的父节点ID  \** ***@return*** *String  \*/* public static List<LayUiTree> getChildPerms(List<Menu> list, int parentId)  {  List<LayUiTree> returnList = new ArrayList<LayUiTree>();  for (Menu menu : list) {  // 一、根据传入的某个父节点ID,遍历该父节点的所有子节点  if(menu.getParentId()==parentId){  //构造tree对象  LayUiTree tree= new LayUiTree();  tree.setId(menu.getMenuId());  tree.setTitle(menu.getMenuName());  tree.setChecked(false);  //开始递归，把所有菜单和当前菜单放入  *recursionFn*(list, tree);  returnList.add(tree);  }  }  return returnList;  }   */\*\*  \* 递归列表  \*   \** ***@param*** *list  \** ***@param*** *t  \*/* private static void recursionFn(List<Menu> list, LayUiTree t)  {  // 得到t的子节点列表  List<LayUiTree> childList = *getChildList*(list, t);  t.setChildren(childList);  for (LayUiTree tChild : childList)  {  if (*hasChild*(list, tChild))  {  // 判断是否有子节点  Iterator<LayUiTree> it = childList.iterator();  while (it.hasNext())  {  LayUiTree n = it.next();  *recursionFn*(list, n);  }  }  }  }   */\*\*  \* 得到子节点列表  \*/* private static List<LayUiTree> getChildList(List<Menu> list, LayUiTree t)  {   List<LayUiTree> tlist = new ArrayList<LayUiTree>();  Iterator<Menu> it = list.iterator();  while (it.hasNext())  {  Menu n = (Menu) it.next();  if (n.getParentId() == t.getId())  {  LayUiTree tree= new LayUiTree();  tree.setId(n.getMenuId());  tree.setTitle(n.getMenuName());  tree.setChecked(false);  tlist.add(tree);  }  }  return tlist;  }   */\*\*  \* 判断是否有子节点  \*/* private static boolean hasChild(List<Menu> list, LayUiTree t)  {  int size = *getChildList*(list, t).size();  if(size>0){  return true;  }else  {  return false;  }  }  } |

2、部门管理控制层

Controller  
@RequestMapping("/dept")  
public class DeptController {  
 @Autowired  
 private DeptBiz deptBizImpl;  
  
 @RequestMapping("/toShowDept")  
 public String toShowDept(){  
  
 return "dept/showDept";  
 }  
 @RequestMapping("/toShowsearchDept")  
 public String toShowsearchDept(){  
 return "dept/showDept";  
 }  
  
  
 @RequestMapping("/searchDept")  
 @ResponseBody  
 public Object searchDept(String deptName,String status){  
 //开始查询  
 List<Dept> deptList = deptBizImpl.search(deptName,status);  
 LayUiTable layUiTable = new LayUiTable();  
 layUiTable.setCode(0);  
 layUiTable.setMsg("返回消息");  
 layUiTable.setCount(deptList.size());  
 layUiTable.setData(deptList);  
 return layUiTable;  
 }  
  
 @RequestMapping("/showDept")  
 @ResponseBody  
 public LayUiTable showDept() {  
 //开始查询  
 List<Dept> deptList = deptBizImpl.selectAllDept();  
 LayUiTable layUiTable = new LayUiTable();  
 layUiTable.setCode(0);  
 layUiTable.setMsg("返回消息");  
 layUiTable.setData(deptList);  
 return layUiTable;  
 }  
  
 @RequestMapping("/saveDept")  
 @ResponseBody  
 public Object saveUser(Dept tbl\_dept){  
 int i=deptBizImpl.insertSelective(tbl\_dept);  
 Map map=new HashMap<>();  
 if(i>0){  
 map.put("code", MyConstants.*successCode*);  
 map.put("message",MyConstants.*saveSuccessMsg*);  
 }else {  
 map.put("code",MyConstants.*failCode*);  
 map.put("message",MyConstants.*saveFailMsg*);  
 }  
 return map;  
  
 }  
 @RequestMapping("/editDept")  
 @ResponseBody  
 public Object editUser(Dept tbl\_dept){  
 int i = deptBizImpl.updateByPrimaryKeySelective(tbl\_dept);  
 Map map= new HashMap<>();  
 if(i>0){  
 map.put("code", MyConstants.*successCode*);  
 map.put("message",MyConstants.*editSuccessMsg*);  
 }else {  
 map.put("code",MyConstants.*failCode*);  
 map.put("message",MyConstants.*editFailMsg*);  
 }  
 return map;  
 }  
  
  
 @RequestMapping("/delDept")  
 @ResponseBody  
 public Object delUser( @RequestParam(value = "ids") String ids){  
 //将json字符串转换成list对象  
 List<String> list= (List<String>) JSON.*parse*(ids);  
 int i = deptBizImpl.delUserByID(list);  
 Map map= new HashMap<>();  
 if(i>0){  
 map.put("code",MyConstants.*successCode*);  
 map.put("message",MyConstants.*delSuccessMsg*);  
 }else {  
 map.put("code",MyConstants.*failCode*);  
 map.put("message",MyConstants.*delFailMsg*);  
 }  
 return map;  
 }

1. 登录
2. @Controller  
   public class LoginController {  
    @Autowired  
    private MenuBiz menuBiz;  
    */\*\*  
    \* 将请求toLogin转发到登录页面login.html  
    \** ***@return*** *\*/* @RequestMapping("/toLogin")  
    public String toLogin(){  
    return "login";  
    }  
    @RequestMapping("/login")  
    public String login(String username, String password, Model model){  
    //登录验证  
    System.*out*.println(username+password);  
    //获取shiro的主体  
    Subject subject = SecurityUtils.*getSubject*();  
    //构建用户登录令牌  
    UsernamePasswordToken token= new UsernamePasswordToken(username,password);  
    //AuthenticationToken token  
    //subject.login(token);  
    try {  
    subject.login(token);  
    }catch (UnknownAccountException e){  
    model.addAttribute("message","用户名错误");  
    return "login";  
    }catch (IncorrectCredentialsException e){  
    model.addAttribute("message","密码错误");  
    return "login";  
    }  
    //将要去index页面之前，保存部分数据到model(在首页的头边栏使用)  
    model.addAttribute("loginName",username);  
    //放入所有的菜单，根据当前登录的用户  
    List<LayUiTree> menus = menuBiz.selectAllMenuByName(username);  
    model.addAttribute("menus",menus);  
    return "index";  
    }  
     
    */\*\*  
    \* 注销  
    \** ***@return*** *\*/* @RequestMapping("/logout")  
    public String logout(){  
    Subject subject = SecurityUtils.*getSubject*();  
    subject.logout();  
    return "login";  
    }  
   }
3. 角色控制层
4. @Controller  
   @RequestMapping("/role")  
   public class RoleController {  
    @Autowired  
    private RoleBiz roleBizImpl;  
    @RequestMapping("/toShowRole")  
    public String toShowRole(){  
    return "role/showRole";  
    }  
     
    @RequestMapping("/showRole")  
    @ResponseBody  
    public LayUiTable showRoleLayui() {  
    //开始查询  
    List<Role> roleList = roleBizImpl.selectAllRole();  
    LayUiTable layUiTable = new LayUiTable();  
    layUiTable.setCode(0);  
    layUiTable.setMsg("返回消息");  
    layUiTable.setData(roleList);  
    return layUiTable;  
    }  
     
    // 根据关键字查询数据  
    @RequestMapping("/searchRole")  
    @ResponseBody  
    public LayUiTable showRoleInfo(  
    @RequestParam(value="roleName",defaultValue="" ) String roleName,  
    @RequestParam(value="roleKey",defaultValue="" ) String roleKey,  
    @RequestParam (value="status",defaultValue="" ) String status) {  
    System.*out*.println("查询中...");  
    //开始查询  
    List<Role> role = roleBizImpl.showRoleInfo(roleName,roleKey,status);  
    LayUiTable layUiTable = new LayUiTable();  
    layUiTable.setCode(0);  
    layUiTable.setMsg("返回消息");  
    layUiTable.setData(role);  
    return layUiTable;  
    }  
     
    @RequestMapping("/saveRole")  
    @ResponseBody  
    public Object saveRole(Role roleInfo){  
    int i = roleBizImpl.insertSelective(roleInfo);  
    Map map= new HashMap<>();  
    if(i>0){  
    map.put("code", MyConstants.*successCode*);  
    map.put("message",MyConstants.*saveSuccessMsg*);  
    }else {  
    map.put("code",MyConstants.*failCode*);  
    map.put("message",MyConstants.*saveFailMsg*);  
    }  
    return map;  
    }  
     
    */\*\*  
    \* 修改用户信息  
    \** ***@param*** *roleInfo  
    \** ***@return*** *\*/* @RequestMapping("/editRole")  
    @ResponseBody  
    public Object editRole(Role roleInfo){  
    int i = roleBizImpl.updateByPrimaryKeySelective(roleInfo);  
    Map map= new HashMap<>();  
    if(i>0){  
    map.put("code",MyConstants.*successCode*);  
    map.put("message",MyConstants.*editSuccessMsg*);  
    }else {  
    map.put("code",MyConstants.*failCode*);  
    map.put("message",MyConstants.*editFailMsg*);  
    }  
    return map;  
    }  
    @RequestMapping("/delRole")  
    @ResponseBody  
    public Object delRole( @RequestParam(value = "ids") String ids){  
    //将json字符串转换成list对象  
    List<String> list= (List<String>) JSON.*parse*(ids);  
    int i = roleBizImpl.delRoleByID(list);  
    Map map= new HashMap<>();  
    if(i>0){  
    map.put("code",MyConstants.*successCode*);  
    map.put("message",MyConstants.*delSuccessMsg*);  
    }else {  
    map.put("code",MyConstants.*failCode*);  
    map.put("message",MyConstants.*delFailMsg*);  
    }  
    return map;  
    }  
     
   }
5. 用户控制层
6. @Controller  
   //此处的RequestMapping是窄化请求，1年级  
   @RequestMapping("/user")  
   public class UserController {  
    @Autowired  
    private UserBiz userBizImpl;  
     
    @RequestMapping("/toShowUser")  
    public String toShowUserLayui() {  
    return "user/showUser";  
    }  
     
    @RequestMapping("/showUser")  
    @ResponseBody  
    public LayUiTable showUserLayui(int page, int limit) {  
    //开始查询  
    PageInfo<User> pageInfo = userBizImpl.selectAllUser(page, limit);  
    LayUiTable layUiTable = new LayUiTable();  
    layUiTable.setCode(0);  
    layUiTable.setMsg("返回消息");  
    //设置分页之后的返回值  
    layUiTable.setCount(pageInfo.getTotal());  
    layUiTable.setData(pageInfo.getList());  
    return layUiTable;  
    }  
     
    @RequestMapping("/saveUser")  
    @ResponseBody  
    public Object saveUser(User userInfo){  
    int i = userBizImpl.insertSelective(userInfo);  
    Map map= new HashMap<>();  
    if(i>0){  
    map.put("code",MyConstants.*successCode*);  
    map.put("message",MyConstants.*saveSuccessMsg*);  
    }else {  
    map.put("code",MyConstants.*failCode*);  
    map.put("message",MyConstants.*saveFailMsg*);  
    }  
    return map;  
    }  
     
    */\*\*  
    \* 修改用户信息  
    \** ***@param*** *userInfo  
    \** ***@return*** *\*/* @RequestMapping("/editUser")  
    @ResponseBody  
    public Object editUser(User userInfo){  
    int i = userBizImpl.updateByPrimaryKeySelective(userInfo);  
    Map map= new HashMap<>();  
    if(i>0){  
    map.put("code",MyConstants.*successCode*);  
    map.put("message",MyConstants.*editSuccessMsg*);  
    }else {  
    map.put("code",MyConstants.*failCode*);  
    map.put("message",MyConstants.*editFailMsg*);  
    }  
    return map;  
    }  
    @RequestMapping("/delUser")  
    @ResponseBody  
    public Object delUser( @RequestParam(value = "ids") String ids){  
    //将json字符串转换成list对象  
    List<String> list= (List<String>) JSON.*parse*(ids);  
    int i = userBizImpl.delUserByID(list);  
    Map map= new HashMap<>();  
    if(i>0){  
    map.put("code",MyConstants.*successCode*);  
    map.put("message",MyConstants.*delSuccessMsg*);  
    }else {  
    map.put("code",MyConstants.*failCode*);  
    map.put("message",MyConstants.*delFailMsg*);  
    }  
    return map;  
    }  
     
   }