**基于WEB的疫情期间社区出入管理系统的设计与实现**

**摘要：** 随着互联网的发展，“线上化”已经成为常态，如何方便快捷的为用户提供服务的问题越来越突出。针对疫情期间“封城、封路、封社区”的情况，文章设计并实现了基于web的疫情期间社区人员出入管理系统。此系统致力于为社区住户提供安全、便捷、高效、透明的出入程序办理平台。

**关键词：**社区管理系统、Web、Java

**一、前言**

2020年，突如其来的疫情，一场没有硝烟的战争，让人猝不及防。我们度过了有史以来最安静的一个春节，感受到了中国对于突发灾难反应速度之快、力度之大，也见证了中国全国人民关于最“宅”比拼。

在疫情最为紧张的时候，封城封路封小区，需要大量的工作人员在各个出入口登记排查，是一个非常复杂且冗余的工作。当进行这些工作时，人员大量聚集，增加了病毒传播的机会。特别地，像社区这样人员比较集中之地，减少人员聚集、减少办理程序时间是十分有必要的，所以基于Web的疫情期间社区出入人员管理系统应运而生。

该系统主要解决社区的四个问题：（1）将线下事务转移到线上，减少人员聚集，提高社区安全性；（2）简化复杂的办事流程，提高办事效率；（3）将社区的人员情况通过软件实时展示，提高信息的透明度，增加住户的安全感；（4）积累数据，进行数据分析，供管理者决策。

客户端是比较流行的软件模式，但是它的问题也非常明显，用户机上安装了许多的软件，每天常用的却极少，这是用户不太喜欢的一个模式。所以，为了给用户良好的体验，本该系统基于web，用户只需要打开网页就可以办理相关事务，查询和自己息息相关的社区情况。本设计采用了B/S模式软件常用的Java语言进行开发，前端用Jsp进行展示，服务端选择Servlet技术，服务器软件使用开源的Tomcat，数据库使用MySQL，数据库连接选择了JDBC，同时使用了阿里巴巴的Druid连接池技术。

**二、需求分析**

**2.1产品适用范围**

疫情期间社区出入管理系统是针对像2020年这样疫情突发、社区人员出入需要严格管控的情景开发，适用于社区特殊时期的管理或者特殊小区的管理（社区人员外出需申请，外来人员不能随意进入）。

**2.2功能需求**

经过调研，我们了解到用户的需求是：当用户使用我们的系统时，需要对用户进行分类选择不同的角色登录至不同的界面。当审核者登录时，系统将跳转至审核者界面，审核者对用户信息进行增删改查；住户登录时，系统跳转到住户界面，住户可以申请外出、确认外出；当门卫登录时，系统跳转到进出人员授权大屏，门卫可以对其授权。

构建用例图如下：

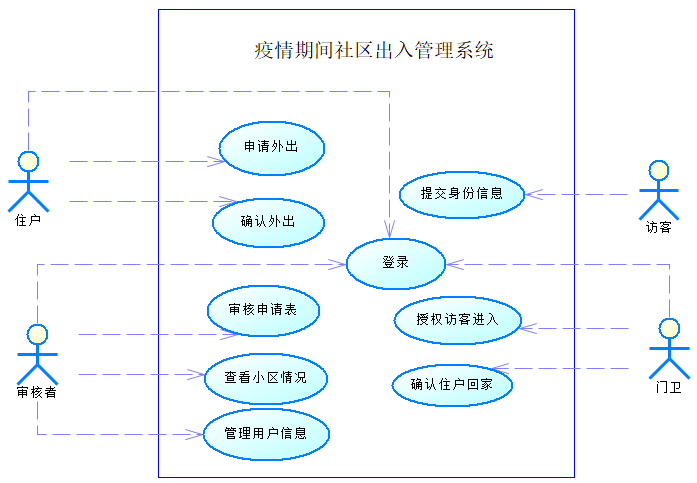


图2-1 疫情期间社区出入管理系统

该系统设计四个角色：住户、访客、审核者和门卫。其中，住户、审核者和门卫需要登录才能进行操作，而访客则不需要登陆。对各个用例的具体描述如下：

用况：登录

【前置条件：系统成功发布，成功通过浏览器进入】

系统呈现用户名输入框及密码输入框，输入用户名及密码，点击登录就把用户账号及密码与数据库中的内容进行匹配，匹配成功进入主界面，匹配失败提示密码或账号错误。

用况：管理住户信息

【前置条件：审核者成功登录系统】

审核者选择住户信息管理

系统呈现出当前已存在住户的信息，并提供增加，删除，修改和按条件查询的界面。

管理员对信息完成处理（增删改查）后，提示保存，并通知是否操作成功。

用况：审核申请表

【前置条件：审核者成功登录系统】

审核者选择审核申

系统呈现住户当前申请且未处理的申请表，并提供通过和驳回操作。审核者处理完成后，通知操作是否成功。

用况：申请外出

【前置条件：住户成功登录系统】

住户选择申请外出

系统呈现外出申请表单，用户填写表单并提交后，通知是否操作成功。

用况：确认外出

【前置条件：住户成功登录系统】

住户选择外出

系统呈现外出成功，并提示注意安全，早点回家。

用况：授权访客进入

【前置条件：门卫成功登录系统】

门卫选择对应访客离开

系统呈现社区内访客的列表，门卫授权离开后，该访客从列表中消失。

用况：确认住户回家

【前置条件：门卫成功登录系统】

门卫选择对应住户回家

系统呈现外出住户的列表，门卫点击回家按钮后，该住户从列表中消失。

用况：提交身份信息

【前置条件：系统成功发布，成功通过浏览器进入】

访客进入访客身份界面

系统呈现访客信息表，访客填写并提交后，系统通知是否成功进入。

**2.3非功能需求**

**用户界面需求**

|  |  |
| --- | --- |
| **需求名称** | **详细要求** |
| 界面简洁 | 保留所必须的文本框，不设置多余菜单及无用文本框及按钮。 |
| 界面设计 | 采用bootstrap |
| 命令交互 | 功能执行后可以得到反馈结果，提示成功还是失败 |
| 信息显示 | 因为人员出入严格控制，出入人员比较少，所以采用一个列表进行显示，不进行分页。 |

表2-1用户界面需求表

**软件环境需求**

|  |  |
| --- | --- |
| **需求名称** | **详细要求** |
| IntelliJ IDEA | IntelliJ IDEA 2020.1.2 x64 |
| JDK | jdk1.8.0\_251 |
| Tomcat | apache-tomcat-8.5.54 |
| MySQL | MySQL5.5.28 |
| Chrome浏览器 | 发布后可以在任意浏览器中运行 |

表2-2软件环境需求表

**三、总体设计**

**3.1结构设计**

疫情期间社区出入管理系统的用户，主要包括住户、审核者、访客和门卫。审核者模块包含：住户信息管理，申请表管理，查看社区情况；住户模块包含：申请外出，确认外出；门卫模块包含：授权访客离开，授权住户回家；访客模块仅有填写身份信息。系统总体结构如下图所示：

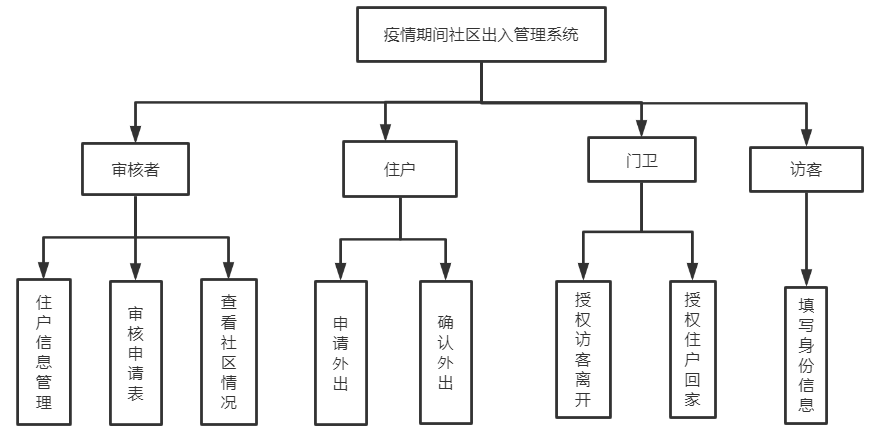


图3-1 系统总体结构图

**3.2审核者各模块的分析说明：**

**3.2.1住户信息管理**

该模块实现添加住户或审核者，根据姓名或住址查询用户，修改住户或审核者信息，这些信息包括姓名、性别、年龄、住址、电话、身份（住户或审核者）。

**3.2.2 审核申请表**

该模块实现审核者查看未处理的外出申请表，如果申请信息正确且合理，予以审核通过，否则驳回申请。

**3.2.3 查看社区情况**

该模块实现实时显示社区内访客人员信息和外出住户信息，帮助审核者了解社区情况。

**3.3住户各模块的分析说明：**

**3.3.1申请外出**

该模块实现住户填写外出申请表（内容包括外出时间、回家时间、申请理由），提交给审核者审核。

**3.3.2 确认外出**

该模块实现住户外出前点击确认外出，系统将住户的状态修改为外出状态。

**3.4门卫各模块的分析说明：**

**3.4.1 授权访客离开**

该模块实现访客离开时，门卫点击该访客对应的离开按钮，系统将其离开时间写入数据库。

**3.4.2 授权住户回家**

该模块实现住户回家进入社区时，门卫点击住户回家，系统修改住户状态外在家。

**3.5访客各模块的分析说明：**

**3.5.1 填写身份信息**

该模块实现访客进入社区前填写身份信息（包括电话、姓名、单位），并提交，系统将其身份信息和进入时间写入数据库。

**四、数据库设计**

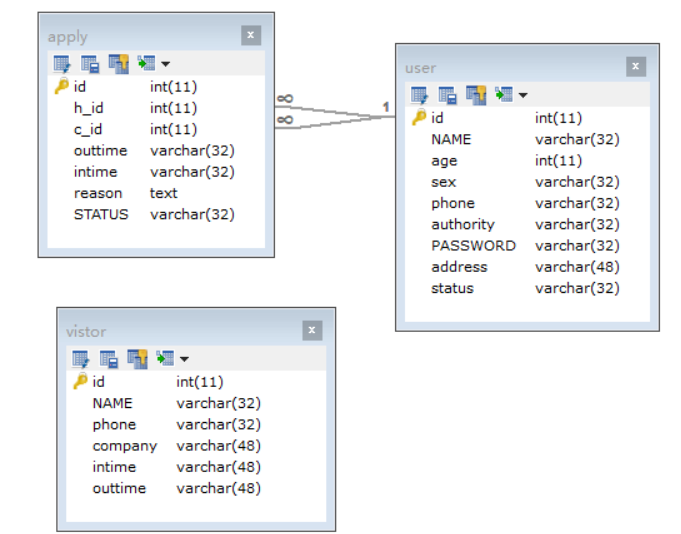
**4.1全局E-R图（如下图）**

图4-1 全局E-R图

E-R图描述：数据库中共需要三个数据表，“用户”存储住户、审核者和门卫的个人信息和状态，供登录、申请和管理时使用；“访客”存储进出社区访客的个人信息和时间；“申请”存储用户的申请表，包括出入时间。

**4.2关系模式**

（1）用户（ID，姓名，年龄，性别，电话，权限，住址）

（2）申请表（ID,H\_ID，C\_ID，外出时间，回家时间，申请原因，状态）

（3）访客（ID,姓名，电话，单位，进入时间，离开时间）

**4.3数据表**

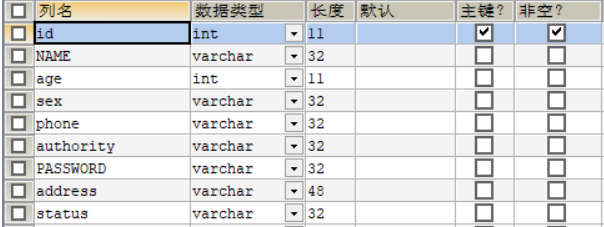
**4.1.1用户信息表**

图3-1 用户信息表

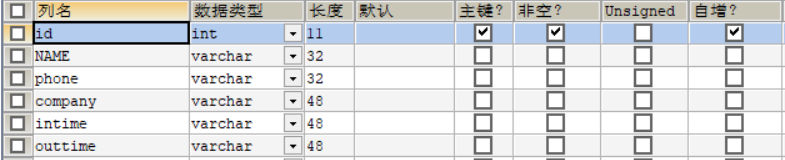
**4.1.2访客信息表**

图3-2 访客信息表

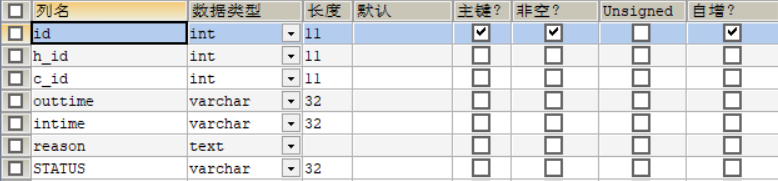
**4.1.3申请信息表**

图3-3 申请信息表

**五、界面设计**

**5.1系统首页**

图5-1 系统首页

**5.2登录界面**

图5-2 登录界面

**5.3 用户信息管理界面**

审核者登录并选择住户信息管理后，跳转到住户信息管理界面。如下图：



图5-3 用户信息管理界面

**5.4 用户信息修改界面**



图5-3 用户信息修改界面

**5.5 添加用户界面**



图5-5 添加用户界面

**六、软件模块设计**

该项目采用了经典的MVC设计模式。

* Model（模型）表示应用程序核心（比如数据库记录列表）。
* View（视图）显示数据（数据库记录）。
* Controller（控制器）处理输入（写入数据库记录）。

**6.1登录模块的设计**

住户、审核者、门卫三个角色在进入首页时，需要选择对应的身份进行登录，用户数据用户名和密码进行登录，登录成功则跳转到对应角色的操作界面，否则保持登录界面，并输出“用户名或密码错误”的提示信息。

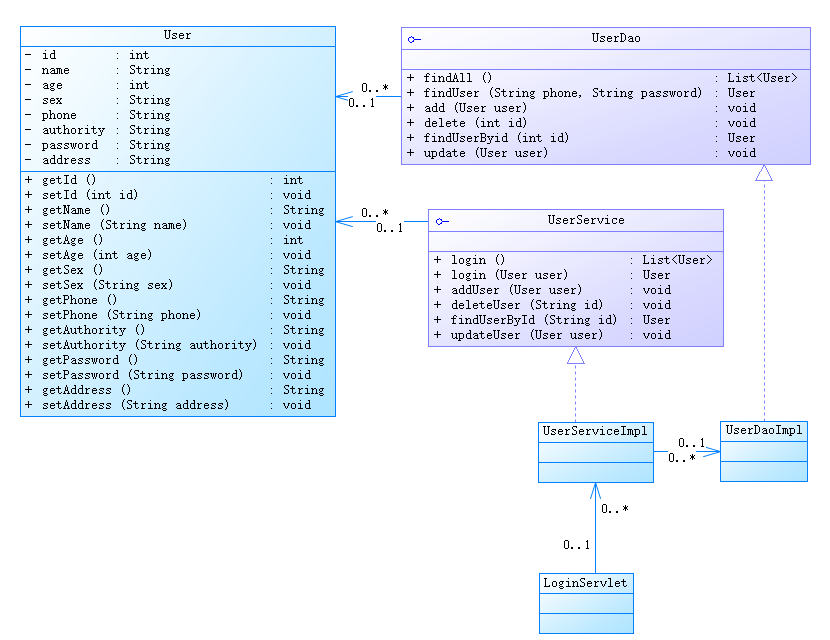


图6-1 登录模块类图

详细描述：类LoginServlet从浏览器登录界面中获取用户名和密码，封装成JaveBean传递给类UserviceImpl；类UserviceImpl将其传入类UserDaoImpl；类UserDaoImpl将接收到的JavaBean与数据库进行对比，若对比成功，返回用户对象，否则返回null，逆向传递给LoginServlet；LoginServlet如果接收到用户对象，则跳转到用户操作界面，否则发送错误信息到浏览器。

**6.2审核者模块设计**

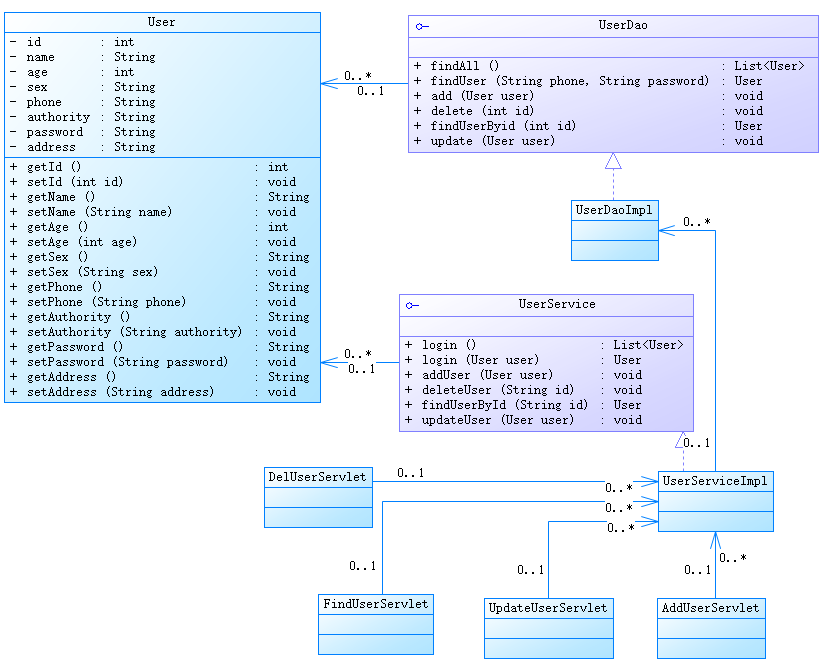


图6-2 审核者模块类图

**6.3申请表模块设计**

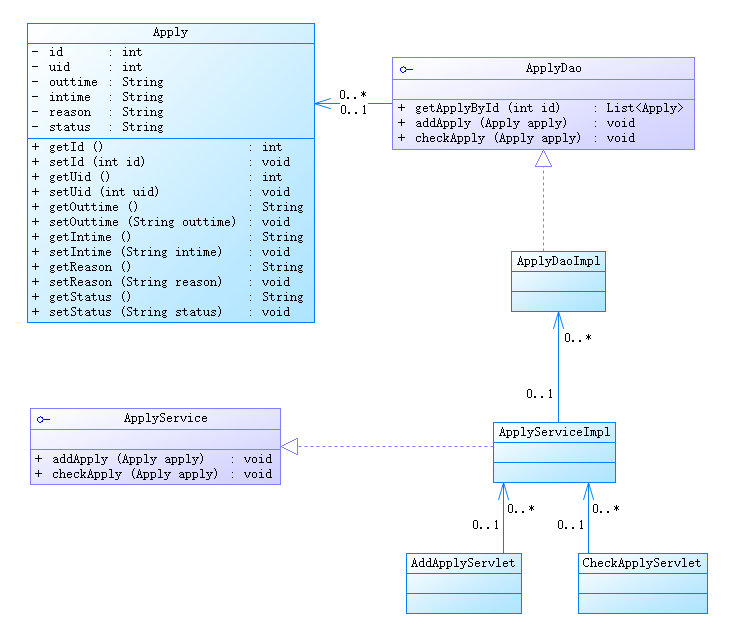


图6-3 申请表模块类图

**6.4访客模块设计**

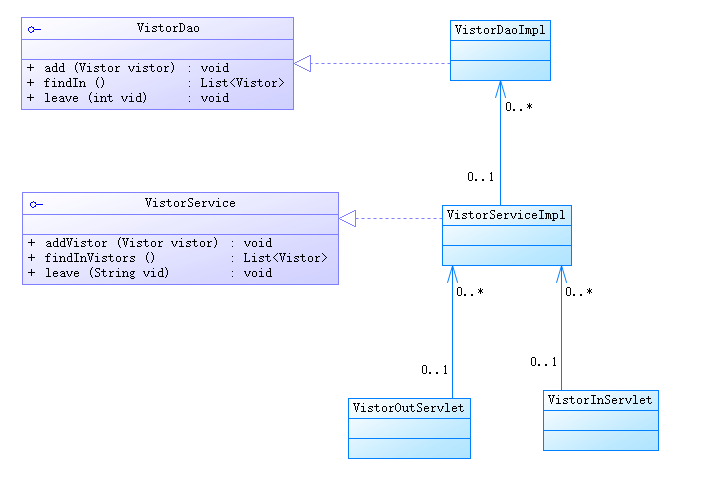


图6-4 访客模块类图

**七、系统的实现关键代码**

**7.1审核者管理用户界面实现**

|  |
| --- |
| @WebServlet("/userListServlet") public class UserListServlet extends HttpServlet {  protected void doPost(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {  UserService service = new UserServiceImpl();  List<User> users = service.findAll();  req.setAttribute("users",users);  req.getRequestDispatcher("/list.jsp").forward(req,res);  }   protected void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {  this.doPost(req, res);  } } |

**7.2 数据库连接**

|  |
| --- |
| Druid.propties  driverClassName = com.mysql.jdbc.Driver url = jdbc:mysql://127.0.0.1/cmsde username = root password = root initialSize = 5 maxActive = 10 maxWait = 30000  public class JDBCUtil {  private static DataSource *ds*;  static {  try {  *// 加载配置文件* Properties pro = new Properties();  InputStream is = JDBCUtil.class.getClassLoader().getResourceAsStream("druid.properties");  pro.load(is);  *// 初始化连接池对象  ds* = DruidDataSourceFactory.*createDataSource*(pro);  } catch(IOException e){  e.printStackTrace();  } catch(Exception e){  e.printStackTrace();  }  }  */\*\*  \* 获取连接池对象  \*/* public static DataSource getDataSource(){  return *ds*;  }  */\*\*  \* 获取Connection对象  \*/* public static Connection getConnection() throws SQLException {  return *ds*.getConnection();  } } |

**7.3用户登录获取权限**

|  |
| --- |
| @WebServlet("/loginServlet") public class LoginServlet extends HttpServlet {  protected void doPost(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {  req.setCharacterEncoding("utf-8");  User user = new User();  Map<String,String[]> map = req.getParameterMap();  try {  BeanUtils.*populate*(user,map);  } catch (IllegalAccessException e) {  e.printStackTrace();  } catch (InvocationTargetException e) {  e.printStackTrace();  }  UserService service = new UserServiceImpl();  User loginUser = service.login(user);  if (loginUser != null){  HttpSession session = req.getSession();  String welcome = loginUser.getName()+"欢迎您！";  session.setAttribute("welcome",welcome);   if (loginUser.getAuthority().equals("管理员"))  res.sendRedirect(req.getContextPath()+"/checkerHome.jsp");  else if (loginUser.getAuthority().equals("门卫"))  res.sendRedirect(req.getContextPath()+"/doorkeeperWorkServlet");  else res.sendRedirect(req.getContextPath()+"/householdHome.jsp");   }else {  req.setAttribute("login\_msg","用户名或密码错误！");  req.getRequestDispatcher("/login.jsp").forward(req,res);  }   }   protected void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {  this.doPost(req, res);  } } |

**7.4审核者对用户信息进行操作**

**7.4.1 修改用户信息**

|  |
| --- |
| @WebServlet("/updateUserServlet") public class UpdateUserServlet extends HttpServlet {  protected void doPost(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {  req.setCharacterEncoding("utf-8");  Map<String,String[]> map = req.getParameterMap();  User user = new User();  try {  BeanUtils.*populate*(user,map);  } catch (IllegalAccessException e) {  e.printStackTrace();  } catch (InvocationTargetException e) {  e.printStackTrace();  }  System.*out*.println(user.toString());  UserService service = new UserServiceImpl();  service.updateUser(user);  res.sendRedirect(req.getContextPath()+"userListServlet");  }   protected void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {  this.doPost(req, res);  } } |

**7.4.2 添加用户**

|  |
| --- |
| @WebServlet("/addUserServlet") public class AddUserServlet extends HttpServlet {  protected void doPost(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {  req.setCharacterEncoding("utf-8");  Map<String,String[]> map = req.getParameterMap();  User user = new User();  try {  BeanUtils.*populate*(user,map);  } catch (IllegalAccessException e) {  e.printStackTrace();  } catch (InvocationTargetException e) {  e.printStackTrace();  }  UserService service = new UserServiceImpl();  service.addUser(user);  res.sendRedirect(req.getContextPath()+"/userListServlet");  }   protected void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {  this.doPost(req, res);  } } |

**7.4.3 删除用户**

|  |
| --- |
| @WebServlet("/delUserServlet") public class DelUserServlet extends HttpServlet {  protected void doPost(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {  String id = req.getParameter("id");  UserService service = new UserServiceImpl();  service.deleteUser(id);  res.sendRedirect(req.getContextPath()+"/userListServlet");  }   protected void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {  this.doPost(req, res);  } } |

**八、系统测试**

**8.1测试计划**

针对审核者对系统的需求，对审核者在该系统的登录、新用户的添加、老用户的信息修改和删除进行测试，通过用户信息列表来显示。采用黑盒测试方法进行测试。同时，对非功能需求进行测试，界面中无多余文本框及按钮，操作完成后具有提示，操作结果是否正常显示，提示信息是否正确且友好。

**8.2功能测试用例**

|  |  |  |  |
| --- | --- | --- | --- |
| 功能描述 | 审核者登录，添加一个新用户，并对其进行修改、删除等操作。 | | |
| 用例目的 | 测试审核者管理用户信息功能是否正常 | | |
| 前提条件 | Web应用项目发布至服务器上 | | |
| 输入/动作 | | 期望的输出/相应 | 实际情况 |
| 错误登录----审核者登录输入密码错误。 | | 提示“用户名或密码错误”。 | 提示“用户名或密码错误”。 |
| 添加----添加一个新用户。 | | 添加后在用户列表中显示。 | 添加后的用户列表。 |
| 修改----对添加用户的信息进行修改。 | | 对页面内已有信息进行修改，修改后页面上显示修改后的信息。 | 显示修改后的信息。 |
| 删除----选择用户，删除。 | | 删除后，显示成功删除提示，并显示用户列表。 | 选中用户成功删除 |

表8-1 功能测试用例表

**8.3非功能测试用例**

|  |  |  |  |
| --- | --- | --- | --- |
| 功能描述 | 使用户获得良好的使用体验 | | |
| 用例目的 | 测试用户界面设计及反馈 | | |
| 前提条件 | Web应用项目发布至服务器上并成功登录 | | |
| 输入/动作 | | 期望的输出/相应 | 实际情况 |
| 点击文本框及按钮 | | 每个文本框及按钮提供对应的功能，无多余及无反应按钮 | 找不到多余文本框及按钮 |
| 进行“增删改查”操作 | | 操作成功时进行提示 | 具有提示 |
| 点击用户信息管理 | | 显示所有用户信息 | 正常显示 |

表8-2 非功能测试用例表

**8.4功能测试结果**

经过测试，社区管理系统满足审核者对用户的添加、修改、删除、查询操作。

存在的问题：该系统存在一定的安全问题，同一用户名可以在不同浏览器上同时登录，这可能会导致数据库的管理出现问题。其次，该系统的一些界面提示不够友好，需要进一步维护和改善。

错误登录：



图8-1 错误登录

添加新用户：



图8-2 添加界面



图8-3 添加后用户列表

修改用户信息：



图8-4 修改界面

删除用户：



图8-5 删除界面

**8.5非功能测试结果**

经过观测可以看到，无多余按键及文本框。

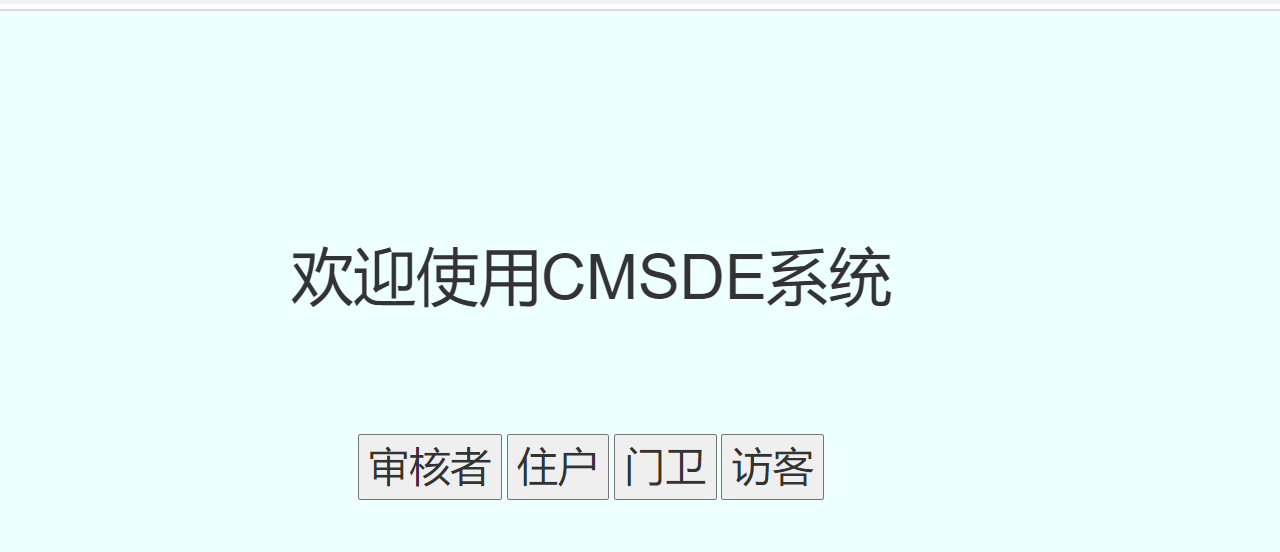


图8-6界面测试

用户列表显示：



图8-7用户列表显示测试

操作提示：



图8-8删除操作提示测试

**参考文献**

[1]蒋少莉,贾艳波.Java Web程序设计课程教学改革——基于项目驱动的互动式教学模式探索[J].教育现代化,2020,7(07):44-45+52.

[2]朱秀梅,林晓玥,王天东.基层社区数字化应急管理系统构建研究[J/OL].软科学:1-16[2020-06-15].http://kns.cnki.net/kcms/detail/51.1268.G3.20200612.1511.034.html.

[3]Booch,G,Object-Oriented Analysis and Design with Applications(Third Edition).北京电子工业出版社，2016.5

[4]Cay S.Horstmann,Core Java Volume I-Fundamentals(10th Edition),机械工业出版社，2016.8