**新疆大学本科毕业论文(设计)**



|  |  |
| --- | --- |
| **论文题目:** | **基于数字孪生的机房管理原型** |
|  | **系统设计与实现** |
| **学生姓名:** | **赵泽天** |
| **学 号:** | **20192501228** |
| **所属院系:** | **软件学院** |
| **专 业:** | **软件工程** |
| **班 级:** | **软件2019-3** |
| **指导老师:** | **冷洪勇** |
| **日 期:** | **2023年 4月 22日** |

声 明

本人郑重声明，本论文是在导师的指导下独立完成，除加注和致谢外，文中不包含他人所发表或撰写的成果。本人拥有自主知识产权，没有抄袭、剽窃他人成果，对于参考的文献已经加注并表示感谢。若有不实之处，本人愿意承担相关法律责任。

作者签名：

签字日期： 年 月 日

本科学位论文（设计）版权使用授权书

本人完全了解学校有关保留、使用本科学位论文（设计）的规定，同意学校保留并向国家有关部门或机构送交论文（设计）的复印件和电子版，允许论文（设计）被查阅和借阅。本人授权新疆大学将本学位论文（设计）的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文（设计）。

本学位论文（设计）属于：

保 密🞎，在\_\_年解密后适用本授权书。

不保密🞎。

作者签名： 指导教师签名：

日期： 年 月 日 日 期： 年 月 日

摘 要

基于数字孪生的机房管理原型系统，作为新一代数据中心可视化管理平台，采用数字孪生技术，实现对机房的虚拟仿真。让管理人员可以清晰直观地掌握IT运营中的有效信息，实现透明化与可视化管理，进而有效提升资产管理与监控管理的效率，实现立体式、可视化的新一代机房运行管理。

本系统采用前后端分离的设计模式，前端使用React框架搭配Ant Design组件库构建系统界面，以及WebGL第三方库three.js，后端使用SpringBoot开发框架搭配Mybatis-Plus开发符合restfull规范的数据接口，数据库使用Mysql，又加入了Nginx作为负载均衡服务，最后部署到AWS云平台上提供持久的可访问服务。本系统实现了登录注册模块、机房模拟及监控模块、机房状态管理模块、报修消息管理模块、用户管理模块以及个人资料模块，根据用户角色不同而展示不同的界面和权限，实现了管理员可以清晰直观监控机房状态、普通用户可以及时提交报修信息、管理员与普通用户权限分离的目标。

毕业设计系统实现的效果，如，实际测试和运行的情况，系统的性能如何，带来了哪些优势，应用价值如何，应用前景如何等（约70-170字）。

**关 键 词：**数字孪生；管理系统；SpringBoot；React；Three.js

ABSTRACT

The digital twin-based server room management prototype system, as a new generation data center visualization management platform, adopts digital twin technology to realize virtual simulation of the server room. It allows managers to clearly and intuitively grasp effective information in IT operations, realize transparent and visualized management, and then effectively improve the efficiency of asset management and monitoring management, and realize a three-dimensional and visualized new generation of server room operation management.

The system adopts the design pattern of front-end and back-end separation, the front-end uses React framework with Ant Design component library to build the system interface and WebGL third-party library three.js, the back-end uses SpringBoot development framework to develop the data interface in line with the restfull specification, the database uses Mysql, and adds Nginx as a load The backend uses SpringBoot development framework to develop data interfaces that conform to the restfull specification, the database uses Mysql, and Nginx is added as a load balancing service, and finally deployed to the AWS cloud platform to provide persistent accessible services. The system implements the login and registration module, the server room simulation and monitoring module, the server room status management module, the initiation and processing of repair messages module, the user management module, and the personal information module, displaying different interfaces and permissions according to the user roles, achieving the goal that the administrator can clearly and intuitively monitor the server room status, regular users can submit repair information in a timely manner, and the administrator and regular user permissions are separated.

**KEY WORDS:** Digital twins; Management System; SpringBoot; React; Three.js

目 录

[1 绪论 1](#_Toc133511315)

[1.1 研究背景及意义 1](#_Toc133511316)

[1.2 研究手段 1](#_Toc133511317)

[1.3 论文组织结构 2](#_Toc133511318)

[2 相关技术与工具概论 3](#_Toc133511319)

[2.1 相关技术介绍 3](#_Toc133511320)

[2.1.1 SpringBoot 3](#_Toc133511321)

[2.1.2 React 3](#_Toc133511322)

[2.1.3 Three.js 3](#_Toc133511323)

[2.1.4 MySql 4](#_Toc133511324)

[2.1.5 RESTful API 4](#_Toc133511325)

[2.1.6 Git 4](#_Toc133511326)

[2.2 相关工具介绍 4](#_Toc133511327)

[2.2.1 IntelliJ IDEA 4](#_Toc133511328)

[2.2.2 Visual Studio Code 4](#_Toc133511329)

[2.2.3 Postman API Platform 5](#_Toc133511330)

[2.2.4 GitHub 5](#_Toc133511331)

[2.3 本章小结 5](#_Toc133511332)

[3 系统需求分析 6](#_Toc133511333)

[3.1 系统业务流程分析 6](#_Toc133511334)

[3.2 系统功能需求分析 7](#_Toc133511335)

[3.2.1 登录注册模块 7](#_Toc133511336)

[3.2.2 机房模拟及监控模块需求分析 7](#_Toc133511337)

[3.2.3 机房状态管理模块 8](#_Toc133511338)

[3.2.4 报修消息管理模块 8](#_Toc133511339)

[3.2.5 用户管理模块 8](#_Toc133511340)

[3.2.6 个人资料模块 8](#_Toc133511341)

[3.3 系统非功能需求分析 8](#_Toc133511342)

[3.4 本章小结 9](#_Toc133511343)

[4 系统设计 10](#_Toc133511344)

[4.1 系统功能设计 10](#_Toc133511345)

[4.1.1 注册 10](#_Toc133511346)

[4.1.2 登陆 10](#_Toc133511347)

[4.1.3 模拟机房环境 10](#_Toc133511348)

[4.1.4 发起错误请求和服务器状态管理 10](#_Toc133511349)

[4.1.5 报错记录管理 11](#_Toc133511350)

[4.1.6 人员管理 11](#_Toc133511351)

[4.1.7 个人资料 11](#_Toc133511352)

[4.2 系统数据库设计 11](#_Toc133511353)

[4.2.1 E-R图设计 11](#_Toc133511354)

[4.2.2 数据库表设计 13](#_Toc133511355)

[4.3 本章小结 14](#_Toc133511356)

[5 系统开发与实现 16](#_Toc133511357)

[5.1 注册登陆模块 16](#_Toc133511358)

[5.2 机房模拟及监控模块 19](#_Toc133511359)

[5.3 机房状态管理模块 21](#_Toc133511360)

[5.4 报修信息管理模块 23](#_Toc133511361)

[5.5 用户管理模块 24](#_Toc133511362)

[5.6 个人资料模块 25](#_Toc133511363)

[5.7 本章小结 26](#_Toc133511364)

[6 系统测试 27](#_Toc133511365)

[6.1 测试目标和原则 27](#_Toc133511366)

[6.2 系统测试环境 27](#_Toc133511367)

[6.3 系统功能测试 27](#_Toc133511368)

[6.3.1 注册功能 27](#_Toc133511369)

[6.3.2 登录功能 28](#_Toc133511370)

[6.3.3 修改个人信息功能 29](#_Toc133511371)

[6.3.4 普通用户发起报错请求 29](#_Toc133511372)

[6.3.5 管理员修改服务器状态 29](#_Toc133511373)

[6.3.6 机房模型拖拽缩放等功能 30](#_Toc133511374)

[6.4 测试结论 30](#_Toc133511375)

[6.5 本章小结 30](#_Toc133511376)

[7 总结与展望 31](#_Toc133511377)

[7.1 论文结论与总结 31](#_Toc133511378)

[7.2 未来研究展望 31](#_Toc133511379)

[参考文献 32](#_Toc133511380)

[致 谢 33](#_Toc133511381)

# 绪论

本章针对基于数字孪生的现代化的机房管理系统，研究了其背景和意义，分析了目前及以前的机房管理系统的发展状况，以服务器机房管理员的具体业务流程和工作内容为基础，分析了该系统设计时的研究手段，详细描述了论文的组织架构。

## 研究背景及意义

目前很多机房管理模式不能满足服务器机房的管理需要而暴露弊端，如机房服务器数量较大，管理接口不规范，造成了较大的管理和遇到某台服务器突发异常时的宕机服务器排查难度，以及服务器工作状态不正常并且消息更新不及时而造成主要业务受到影响的现象，机房管理人员的操作不能满足财务制度规范化的要求[1] 。

因此建立可视化、易观察、易管理，基于数字孪生技术模拟真实环境的机房管理系统，实现机房管理的信息化，透明化，可视化，自动化，让管理人员可以清晰直观地掌握服务器运行管理的有效信息，实现透明化与可视化管理，进而有效提升监控管理与资产管理的效率，对解决上述所暴露出来的问题是具有重要意义的。

## 研究手段

（1）查阅、收集、整理所需资料；

（2）根据设计任务制定工作进度；

（3）完成对数字孪生开发相关技术资料和交互可视化系统开发相关需求资料的查询和整理工作；

（4）选择合适的软件开发框架，完成系统的需求分析；

（5）完成系统的概要设计、详细设计，编码实现系统；

（6）完成相关技术文档的整理及论文的撰写；

## 论文组织结构

论文的结构为七个章节，其中各个章节的主要涉及的内容分别是：

第一章：绪论。提出论文的选题意义和相关背景，其中包括就机房管理系统的必要性研究以及投入使用后带来的优势，并阐述了研究过程和研究手段。

第二章：相关技术概述，详细说明基于多层架构，运行在浏览器中的机房管理系统的平台框架。列举并且简要介绍项目开发过程中涉及到的机房管理系统相关理论和技术，其中包括基于Java语言的SpringBoot开发框架技术，React 前端框架技术以及实现数字孪生3D模拟实际环境的WebGL第三方库three.js等。

第三章：系统需求分析，通过机房管理员对应的具体业务和对应的工作需求，来定义系统的具体功能和流程，完成了系统对应的过程建模，数据建模和需求建模。对于目标系统整体需求有一个完善的了解。

第四章：系统设计，设计系统的架构以及按照系统的功能需求分析划分功能模块，设计各个模块对应的功能业务。然后对应的数据库对象模型，根据对应的对象模型，在数据库中构造出对应的数据表。同时使用类图，序列图，流程图等配合文字，对本系统的主要模块进行详细的分析，最终集成各个模块构成系统。

第五章：系统开发与实现，根据已完成的系统架构，分模块逐个实现每个功能。通过系统各模块页面，搭配关键实现代码阐述功能或页面的实现思路。

第六章：系统测试，阐述了系统的运行环境，并构建单独的测试环境，对重点模块的类模型进行单独的单元测试，同时对系统主要功能点设计对应的测试用例，执行测试用例的同时梳理业务逻辑。最后就测试结果进行分析，并得出结论。

第七章：总结与展望，总结本论文的全部工作，分析工作内容的价值和必要性，总结目前系统的优点与长处，同时分析本系统的不足之处和需要改进的地方，并对本系统未来可能需要的功能、页面和性能方面的优化进行展望。

# 相关技术与工具概论

本章主要就机房管理系统开发过程中使用到的关键技术和主要工具进行介绍，相关技术介绍了系统后端数据接口实现技术：SpringBoot框架，前端页面实现技术React框架，实现在浏览器中3D模拟机房环境所用到的Three.js。数据存储方案使用了MySql数据库，数据接口遵循RESTful规范。开发过程的代码版本控制和同步使用到了Git。其中主要工具方案为，后端使用IntelliJ IDEA，前端使用Visual Studio Code，前后端接口调试工具使用Postman API，代码版本控制和同步平台GitHub。

## 相关技术介绍

### SpringBoot

SpringBoot概述和作用：即基于Spring框架的快速开发框架，用于简化Spring应用程序的配置和部署。

SpringBoot的优点和应用场景：具有快速启动、内嵌服务器、自动配置等，应用场景有Web应用、微服务等。

### React

React概述和作用：即用于构建用户界面的JavaScript库，遵循Flux单向数据流的应用架构。

React的优点和应用场景：可组合性、高效性、灵活性等，可以应用于单页应用、大型应用等。

### Three.js

Three.js概述和作用：即用于在Web上创建和呈现3D图形的JavaScript库，提供大量特性与API以绘制3D场景于浏览器。

Three.js的优点和应用场景：如易用性、跨平台、高性能等，应用场景有游戏、虚拟现实、实现基于数字孪生的数据可视化等。

### MySql

MySql概述和作用：即一个开源的关系型数据库管理系统，用于持久化存储和管理应用数据。

MySql的优点和应用场景：如可靠性、安全性、可扩展性等，另外也是最流行的关系型数据库之一，应用场景有Web应用、企业应用等。

### RESTful API

RESTful API概述和作用：REST即Representational State Transfer，用于设计和开发Web API的一种软件架构风格，REST原则是客户端和服务器之间的交互在请求之间是无状态的。

RESTful API的优点和应用场景：优点有行为和资源分离，接口开放性高，应用场景广泛，可应用于各式各样的的前后端服务请求。

### Git

Git概述和作用：即一个开源的分布式版本控制系统，用以有效、高速的处理从小到大的项目版本管理。

Git的优点和应用场景：是目前世界上最先进的分布式版本控制系统，没有之一，Git不但可以应用于应用代码的版本控制，并且可以用于文档的管理等。

## 相关工具介绍

### IntelliJ IDEA

IntelliJ IDEA 是一款由 JetBrains 公司开发的 Java 集成开发环境，提供了丰富的功能和插件，支持多种编程语言和框架，如 Java、Kotlin、Scala、Android、Spring 等等。它的主要特点是智能提示、代码重构、版本控制、调试等等。IntelliJ IDEA 适用于中大型项目的开发，具有良好的性能和稳定性。

### Visual Studio Code

Visual Studio Code（VSCode） 是由微软开发的一款跨平台开发工具，支持多种编程语言和框架，如 JavaScript、Python、C++、Vue、React 等等。VSCode 的特点是轻量、快速、可扩展，支持代码提示、调试、版本控制等等。它是一款适用于小型项目和日常编程的工具。

### Postman API Platform

Postman 是一款常用的 API 测试工具，支持多种 HTTP 请求方式，如 GET、POST、PUT、DELETE 等。它的特点是可视化的接口测试、请求的实时查看和响应的断言，可以帮助开发人员和测试人员更快速地调试和测试 API。

### GitHub

GitHub 是一个基于 Git 的代码托管平台，它为开发者提供了一个在线的代码托管服务，支持代码版本管理、代码审核、代码分享等等。在 GitHub 上可以找到很多开源项目和社区，方便开发者学习和交流。

## 本章小结

本章主要介绍了机房管理系统开发过程中所用到的关键技术和主要工具。在技术方面，介绍了系统后端数据接口实现技术SpringBoot框架、前端页面实现技术React框架和在浏览器中3D模拟机房环境所用到的Three.js等技术；数据存储方案使用了MySql数据库，数据接口遵循RESTful规范；在开发过程的代码版本控制和同步使用了Git。在工具方面，介绍了后端使用IntelliJ IDEA，前端使用Visual Studio Code，前后端接口调试工具使用Postman API，代码版本控制和同步平台GitHub等工具。同时，本章还对各个技术和工具的优点、应用场景进行了介绍。

# 系统需求分析

## 系统业务流程分析

首先确定业务目标和场景：业务目标是对机房服务器进行监测和管理维护，主要业务开始于普通用户在工作过程中发现服务器运行状态不正常，然后根据所用服务器的编号发起报错请求，并填写错误类型和具体错误内容。普通用户发起请求后会在系统上更新服务器管理界面中的相应编号的Fix Info状态为Show Fix Info，并展示错误类型和具体的错误内容，然后会管理员系统中在报修信息管理模块中生成一条报修记录，记录下本次报修相关信息如报修人，报修服务器编号，错误类型，错误具体内容等。

管理员在自己的系统上查看到某个状态不正常时，可以查看用户提交的错误类型和错误信息，然后去机房找到该编号的服务器进行检查，如果没有问题或者立即修复了错误则把状态修改为正常工作，如果检查到了错误并且没能当场修复，则在系统中将服务器状态修改为不工作。

以上流程完成之后会在报修信息管理模块中之前由于用户报修操作创建的报修记录中补全修复日期，修复人的用户名等。就此，本系统的主要业务流程结束。

根据上述对系统业务流程分析，可以设计出系统用例图，如图3-1所示：

图示

描述已自动生成

**图3-1 系统用例图**

## 系统功能需求分析

### 登录注册模块

登陆注册对于普通用户和管理员来说都是必要的功能需求，另外还需要对用户名、手机号、电子邮箱和密码等进行相应的数据校验，使得这些必要字段符合要求，如用户名不能重复，手机号和电子邮箱不能重复且必须符合相应的格式，密码应该符合包含大小写字母和数字以及长度要求。

另外，对于注册功能来说，用户需选择账户类型，注册为管理员还是普通用户。对于登录功能来说，需要根据账户类型的不同向服务器发送不同的请求来实现不同角色具有的权限不同，系统展示页面的不同。

### 机房模拟及监控模块需求分析

机房模拟及监控属于本系统的重点功能，需要模拟出实际的3D机房环境，将服务器管理模块的数据绑定到模拟界面，并展示出服务器当前状态，使得用户可以清晰的查看设备运行状态。

### 机房状态管理模块

机房状态管理模块根据用户角色权限的不同，将展示出不同的页面和操作选项。

对于普通用户来说，可以根据自己使用中的服务器的运行状态，在该模块进行发起服务器错误请求，包括错误类型以及具体的错误内容等。

对于管理员，该模块将展示服务器的所有详细信息，并且可以筛选条件来展示所有服务器、包含错误请求的服务器以及运行状态不正常的服务器，最重要的是可以根据实际检查服务器后及时更新服务器的运行状态，从而避免由于信息不及时而造成的损失和麻烦。

### 报修消息管理模块

本模块的主要功能是记录用户报修消息，在用户发起一个服务器错误请求时，就会在本模块中添加一条报修记录，包含报修人的用户名，报修时间，错误类型和错误具体内容等。

另外，在管理员处更新一台服务器的运行状态时，之前在本模块中新增的错误请求记录将自动完善处理错误请求的管理员用户名，以及处理时间等。

### 用户管理模块

用户管理模块的仅对管理员开放，其功能是可以让管理员清晰的查看到系统的所有用户的详细信息，包括用户名，创建时间，电子邮箱、手机号联系方式，用户类型，账号状态等。使得管理员在处理报错信息时可以直观的获取发起报错的用户的联系方式，便于确定错误的具体内容。

### 个人资料模块

个人资料模块对于管理员和普通用户同时开放，本模块主要展示当前用户的个人资料等内容，提供修改用户名、联系方式等功能。

## 系统非功能需求分析

本系统除了功能需求外，如下非功能性的需求同样非常重要。

1. 用户注册时的字段限制

如用户名不能重复，手机号和电子邮箱不能重复且必须符合相应的格式，密码应该符合包含大小写字母和数字以及长度要求。

（2）根据用户权限提供不同的页面和操作

当管理员登录系统时，机房模拟及监控模块、机房状态管理模块、报修消息管理模块、用户管理模块以及个人资料模块，所有五个模块将全部展示。并且在机房管理模块中提供修改服务器状态的操作。

当普通用户登陆系统时，仅展示模拟及监控模块、机房状态管理模块和个人资料模块。并且在机房状态管理模块中仅能发起服务器错误请求，而不能直接修改服务器运行状态。

（3）系统安全性

安全性对于机房管理系统来说尤为重要，服务器状态通常只能对内部用户可见，因此，在登陆之前，系统内所有内容均不可见，并且考虑到用户未退出系统可能将系统信息暴露给无关人员，所以在用户超过特定时间无操作的情况下，系统将自动删除登陆用户信息并退出系统。

另外，本系统应当在采用前后端分离模式时，加密用户信息，因此，本系统将采用JSON Web Token技术并进行通信加密，通过JSON对象在各方之间安全地传输信息，每个请求中都必须包含jwt数字签名。

## 本章小结

本章主要是关于系统需求分析的内容。首先，通过确定业务目标和场景，对系统的业务流程进行了分析，并设计出系统用例图。然后，对系统的功能需求进行了分析，包括登录注册模块、机房模拟及监控模块、机房状态管理模块、报修消息管理模块、用户管理模块和个人资料模块。其中，机房模拟及监控模块是系统的重点功能，需要模拟出实际的3D机房环境，将服务器管理模块的数据绑定到模拟界面，并展示出服务器当前状态。其他模块则主要是为了实现系统的业务流程而存在，包括记录用户报修消息、展示所有用户的详细信息等。

# 系统设计

## 系统功能设计

### 注册

注册功能的目的是提供用户注册账号以登陆系统。点击注册按钮后如果字段不符合要求则进行提示注册失败，注册成功则跳转登录页面。

注册时需填写用户名用于显示在系统中，手机号和电子邮箱用来联系用户，以及设置密码。这些字段都有相应的填写限制，如用户名需控制在3到20个字符，且由于使用用户名登陆，不能重复，手机号和电子邮箱必须符合相应格式，并不能重复。考虑到系统安全性，密码限制为至少含有一个大写字母一个小写字母和数字，并且长度至少为8位。为了之后通过账户类型是管理员还是普通用户来进行用户权限与页面展示的限制，需要在注册时选择注册管理员账户还是普通用户账户。

### 登陆

登陆的目的是使已经注册好账户的用户可以顺利登陆系统，并且在点击登录按钮之后提示是否登陆成功，成功则进入模拟机房模块，不成功则提示登录失败。

登陆时的请求中应该包含用户信息如用户名、密码和用户类型，成功则返回储存了登录用户信息的jwt密钥，并存入浏览器Cookies中用于接下来实现用户权限和展示页面的限制。

### 模拟机房环境

在机房模拟界面需要模拟出实际的3D机房环境，并实现缩放效果和拖拽旋转效果，还需要将服务器管理模块的数据绑定到模拟界面，通过鼠标悬停到服务器上即展示出当前选中服务器的工作状态等信息，使得用户可以清晰的查看设备运行状态。

### 发起错误请求和服务器状态管理

发起错误请求和服务器状态管理功能在同一界面，但是根据账户类型向不同的用户展示不同的效果和操作。

对于普通用户，只能进行查看服务器信息和发起报错请求，查看服务器信息可以通过限制条件展示所有服务器，含有错误请求的服务器以及工作状态错误的服务器列表。不能改变服务器的工作状态，在发起报错请求时需要用户填写错误类型和具体错误信息以供管理员排查服务器状态，点击发起按钮之后，如果未填写错误类型和具体错误信息，将不能成功，系统将提示请求失败，填写了错误类型和具体错误信息之后系统提示请求成功。

对于管理员，除了可以查看服务器信息、查看报错请求还可以更改服务器工作状态，同样更改成功和失败系统都会有提示反馈。

### 报错记录管理

报错记录管理的目的是向管理员展示之前所有的报错记录，所以仅对管理员可见，包含了发起人用户名、错误类型、错误详细信息，以及在管理员改变服务器状态后向记录中插入处理错误的管理员的用户名。

### 人员管理

人员管理的目的是向管理员展示系统中用户所有的信息，如用户名、账户创建时间、用户联系方式等，用于紧急情况下联系错误发起人进行错误排查，所以仅对管理员可见。

### 个人资料

个人资料主要展示展示当前登录用户的账户信息等内容，提供修改用户名、联系方式等功能。在修改个人资料时需要对各个字段进行和注册时相同的限制，以保证用户信息数据的有效性。

## 系统数据库设计

### E-R图设计

E-R图即实体-联系图，是数据库设计阶段不可缺少的产物。根据之前的系统设计和系统流程图等，设计出系统E-R图以及User、Server和FixInfo实体E-R图，本管理系统的E-R图如图4-1所示：

图示

描述已自动生成

**图4-1 系统E-R图**

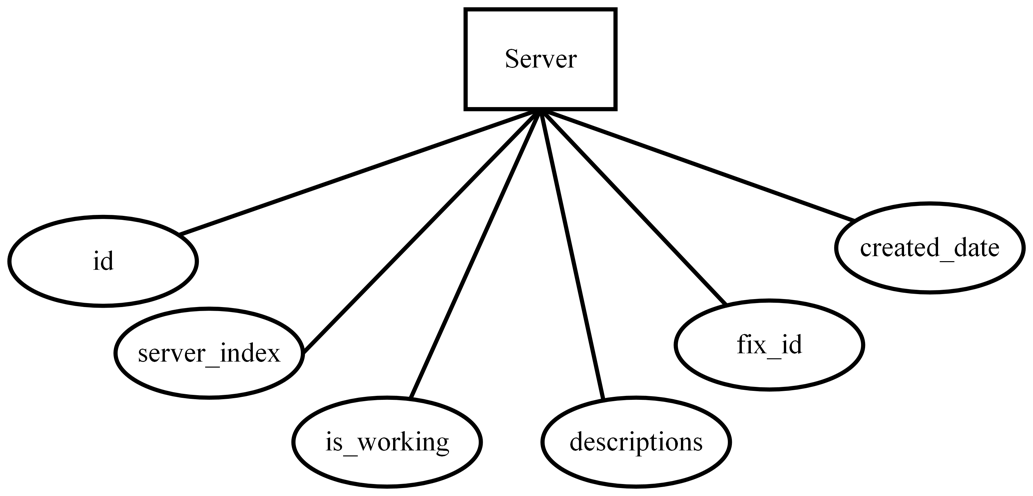
（1）User实体和属性如图4-2所示：

图形用户界面, 应用程序

描述已自动生成

**图4-2 User实体E-R图**

（2）Server实体和属性如图4-3所示：



**图4-3 Server实体E-R图**

（3）FixInfo实体和属性如图4-4所示：

图形用户界面, 应用程序

描述已自动生成

**图4-3 FixInfo实体E-R图**

### 数据库表设计

本系统选择使用MySQL数据库，因为MySql是一个成熟、稳定的关系型数据库系统，具有高可靠性、高性能和开源免费等优点，它具有严格的数据结构、数据一致性和数据完整性等优点，适合处理结构化数据。系统总共建立了3张表，分别是用于储存用户信息的User表，用于存储服务器信息的Server表以及用于存储用户发起的报错记录的FixInfo表，其中Server表中含有外键fix\_id，各表结构如下。

（1）User表主要存储用户信息，参与登录、注册、创建报错记录和权限控制等功能模块，具体字段如表4-1所示：

**表4-1 User表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 字段名 | 类型 | 长度 | 非空 |
| ID | id | varchar | 255 | True |
| 用户名 | username | Varchar | 255 | True |
| 密码 | password | varchar | 255 | True |
| 手机号 | phone | varchar | 11 | True |
| 电子邮箱 | email | varchar | 255 | True |
| 是否可用 | is\_active | boolean | 1 | True |
| 是否是管理员 | is\_admin | boolean | 1 | True |
| 简介 | short\_intro | varchar | 255 | False |
| 图片 | img | varchar | 255 | False |

（2）Server表主要存储服务器信息，参与服务器管理、报错记录和模拟机房等功能模块，具体字段如表4-2所示：

**表4-2 Server表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 字段名 | 类型 | 长度 | 非空 |
| ID | id | varchar | 255 | True |
| 服务器编号 | server\_index | varchar | 255 | True |
| 状态是否正常 | is\_working | boolean | 1 | True |
| 服务器描述 | descriptions | varchar | 255 | False |
| 报错记录ID | fix\_id | varchar | 255 | False |
| 创建日期 | created\_date | datetime | N/A | True |

（3）FixInfo表主要存储用户发起的报错请求内容，参与创建报错记录和管理员修改服务器状态等功能模块，具体字段如表4-3所示：

**表4-3 User表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 字段名 | 类型 | 长度 | 非空 |
| ID | id | varchar | 255 | True |
| 请求发起用户 | sender | varchar | 255 | True |
| 修复管理员 | fixer | varchar | 255 | False |
| 发起日期 | created\_date | datetime | N/A | False |
| 错误类型 | fix\_status | varchar | 255 | True |
| 错误信息 | info | varchar | 255 | False |
| 服务器编号 | server\_index | varchar | 255 | True |
| 修复日期 | fix\_date | datetime | N/A | False |

## 本章小结

本章主要介绍了基于数字孪生的机房管理原型系统的系统设计，包括系统功能设计和系统数据库设计两个方面。

在系统功能设计方面，首先介绍了系统的注册和登录功能，包括用户填写的字段和限制条件，以及注册时选择管理员账户或普通用户账户的区别；其次介绍了模拟机房环境的设计，包括3D机房环境的实现、缩放和拖拽旋转效果的实现、服务器管理模块数据的绑定和展示等；接着介绍了发起错误请求和服务器状态管理的功能，普通用户只能查看服务器信息和发起报错请求，管理员可以进行服务器状态管理和查看报错请求；最后介绍了报错记录管理、人员管理和个人资料管理等功能。

在系统数据库设计方面，介绍了使用E-R图设计实体和联系的过程，以及具体实现时User、Server和FixInfo实体的属性和关系。同时，系统选择使用MySQL数据库来实现数据的存储和管理。

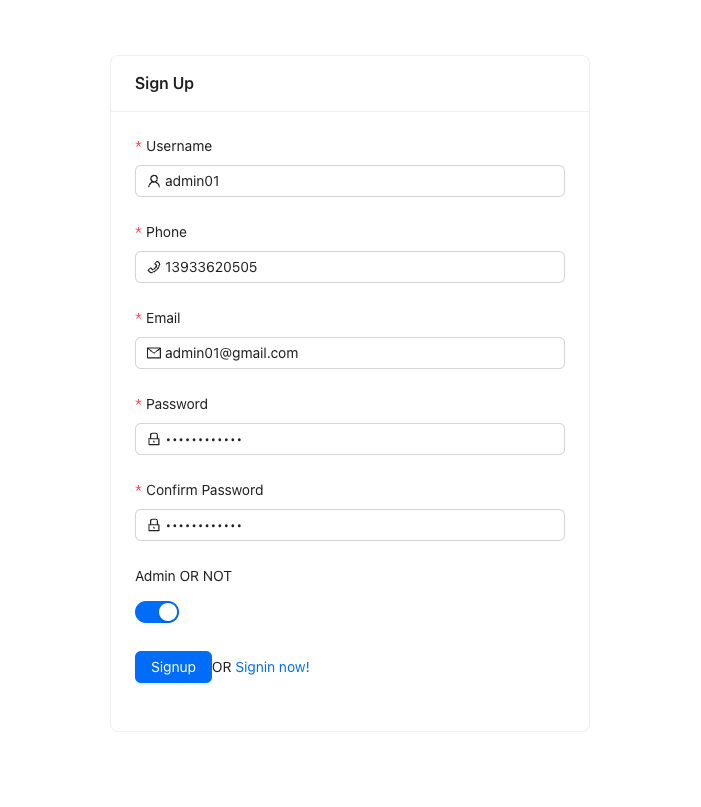
总的来说，本章介绍了一个完整的机房管理系统的设计方案，包括系统功能和数据库的设计，为后续实现提供了重要的参考和指导。

# 系统开发与实现

系统采用前后端分离开发模式，系统后端数据接口使用SpringBoot框架实现，前端页面使用React框架实现，使用Three.js搭配React框架实现在浏览器中3D模拟机房环境。数据存储方案使用了MySql数据库，数据接口遵循RESTful规范。

## 注册登陆模块

注册时用户需要填写用户名、手机号、电子邮件地址以密码，如图5-1所示：



**图5-1 注册界面**

填写完字段之后点击Signup按钮，前端即向后端发送包含用户注册信息的注册请求，后端收到请求后进行数据校验，数据符合要求则响应前端，并由前端进行跳转到登陆界面的操作。处理请求关键代码如下：

|  |
| --- |
| @PostMapping("/signup")  public Result<?> signup(@Validated @RequestBody User user, BindingResult bindingResult){  if(bindingResult.hasErrors()){  String failedMsg = bindingResult.getFieldError().getDefaultMessage();  return Result.fail(failedMsg);  }  boolean b = userService.addUser(user);  if (b){  return Result.success("sign up successfully");  }  return Result.fail("sign up failed");  } |

登陆界面需输入用户名和密码，界面如图5-2所示：

图形用户界面, 文本, 应用程序

描述已自动生成

**图5-2 登陆界面**

登陆界面需输入用户名和密码，前端即向后端发送包含用户登录字段的请求，后端收到请求后查看用户名是否存在，并对输入的密码和数据库中的加密过的密码进行匹配，成功则提示登录成功并跳转到模拟机房界面并携带登录用户的token用于之后的权限限制和控制展示页面，登录失败则提示登录失败。

请求处理关键代码如下：

|  |
| --- |
| @PostMapping("/signin")  public Result<Map<String, Object>> signin(@RequestBody User user) {  Map<String, Object> data = userService.signin(user);  if (data != null){  return Result.success(data,"login successful!");  }  return Result.fail("login failed!");  } |

登录成功之后，后端返回包含登录用户信息的token，也就是jwt，并需要在以后的所有请求中携带token，在请求进入Controller层处理之前进行验证，验证通过才能正常请求原本服务，否则返回请求失败，还用于之后的权限限制和控制展示页面。

后端配置jwt验证的关键代码如下：

|  |
| --- |
| @Component  @Slf4j  public class JwtValidateInterceptor implements HandlerInterceptor {  @Autowired  private Jwt jwt;  @Override  public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) throws Exception {  log.debug(request.getRequestURL()+" --------------need check");  String token = request.getHeader("token");  log.debug("-\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*token:"+token);  if (token != null){  try {  jwt.parseToken(token);  log.debug(request.getRequestURL()+" -----------check passed");  return true;  } catch (Exception e) {  throw new RuntimeException(e);  }  }  log.debug(request.getRequestURL()+" --------------check failed");  response.setContentType("application/json;charset=utf-8");  response.getWriter().write(JSON.toJSONString(Result.fail("jwt is invalid")));  return false;  }  } |

## 机房模拟及监控模块

机房模拟及监控模块实现了对机房的3D模拟并，在鼠标悬停到服务器上时将该服务器对应的详细信息和工作状态等展示出来，如图5-3所示：



**图5-3 机房模拟及监控模块界面**

本模块使用到了Three.js这个WebGL第三方库，将3D模型导入React框架创建的Web项目中，实现在浏览器中展示3D对象的效果。主要步骤是先建立场景，然后加载GLTF格式的3D模型，实现选中某个服务器并现实信息面板。

建立场景的关键代码如下：

|  |
| --- |
| constructor(canvas: HTMLCanvasElement, modelPath: string = './models/') {  this.renderer = new WebGLRenderer({ canvas })  this.renderer.setSize(window.innerWidth, window.innerHeight \* 0.941)  this.scene = new Scene()  this.scene.background = new Color("white")  this.camera = new PerspectiveCamera(  45, canvas.width / canvas.height, 0.1, 1000  )  this.camera.position.set(0, 10, 15)  this.camera.lookAt(0, 0, 0)  this.controls = new OrbitControls(  this.camera,  this.renderer.domElement  );  this.modelPath = modelPath  this.crtTexture('cabinet-hover.jpg')  } |

加载GLTF模型的关键代码如下：

|  |
| --- |
| loadGLTF(modelName: string) {  gltfLoader.load(this.modelPath + modelName, ({ scene: { children } }) => {  children.forEach((obj: any) => {  const { map, color } = obj.material as MeshStandardMaterial;  this.changeMat(obj, map, color);  // 如果模型包含cabinet字符的话，就将其添加到机柜集合  if (obj.name.includes("cabinet")) {  this.cabinets.push(obj);  }  });  this.scene.add(...children)  })  } |

选中服务器并展示信息面板的关键代码如下：

|  |
| --- |
| if (!canvasEle.current) { return }  canvasEle.current.width = window.innerWidth  canvasEle.current.height = window.innerHeight  room = new MachineRoom(canvasEle.current)  room.loadGLTF('machineRoom.gltf')  room.animate()  //当鼠标划入机柜，显示信息面板  room.onMouseOverCabinet = ({ name }: { name: string }) => {  setVisible('block')  axios.get('http://127.0.0.1:8080/server/info', {  params: { 'serverIndex': name }  })  .then(response => {  console.log(response.data)  let description: string = response.data.data.descriptions  let fixId: string = response.data.data.fixId  setState((prevState) => {  return Object.assign({}, prevState, { curCabinet: { name, description, fixId } })  })  })  .catch(error => {  console.log(error);  });  }  //当鼠标在机柜上移动，让信息面板随鼠标移动  // room.onMouseMoveCabinet = (left: number, top: number) => {  room.onMouseMoveCabinet = (left: any, top: any) => {  setState((prevState) => {  return Object.assign({}, prevState, { planePos: { left, top } })  })  }  //当鼠标划出机柜，隐藏信息面板  room.onMouseOutCabinet = () => {  setVisible('none')  } |

## 机房状态管理模块

机房状态管理模块根据用户角色展示不同的页面和操作选项，管理员权限登录时展示服务器对应的详细信息和工作状态，并提供修改服务器状态的按钮。修改状态时填写字段有相应的字段限制，点击update按钮后前端向后端发起包含修改操作的请求，后端根据请求修改数据库中保存的服务器状态等信息，然后返回相应结果，成功则提示update成功并更新服务器信息，否则提示update错误，页面效果如图5-4所示：

**图形用户界面, 应用程序

描述已自动生成**

**图5-4 机房状态管理模块管理员视角**

更新服务器状态的关键代码如下：

|  |
| --- |
| @PutMapping("/update")  public Result<?> updateServer(@RequestBody Map map) {  log.debug(map.toString());  Server server = serverService.getServerInfo(map.get("serverIndex").toString());  log.debug("$$$$$$$$$$$before update server0:" + server.toString());  server.setIsWorking(map.get("isWorking").toString() == "true"?true:false);  server.setDescriptions(map.get("descriptions").toString());  String username = map.get("username").toString();  log.debug("$$$$$$$$$$$before update server1:" + server.toString());  boolean updated = serverService.update(server, username);  if (updated) {  return Result.success("server details updated successfully");  }  return Result.fail("server details updated failed");  } |

普通用户权限登录时展示服务器对应的详细信息和工作状态，只提供发起报错请求的选项，不提供修改服务器状态的操作。有用户对服务器发起错误请求时该服务器对与所有用户都不能再提起请求，并在FixInfo字段中可以查看提交的错误类型和错误具体内容。页面效果如图5-5所示：

**图形用户界面, 应用程序, 网站

描述已自动生成**

**图5-5 机房状态管理模块普通用户视角**

本页面提供选择全部服务器、含有报错请求的服务器以及工作状态不正常的服务器筛选按钮，其主要实现代码如下：

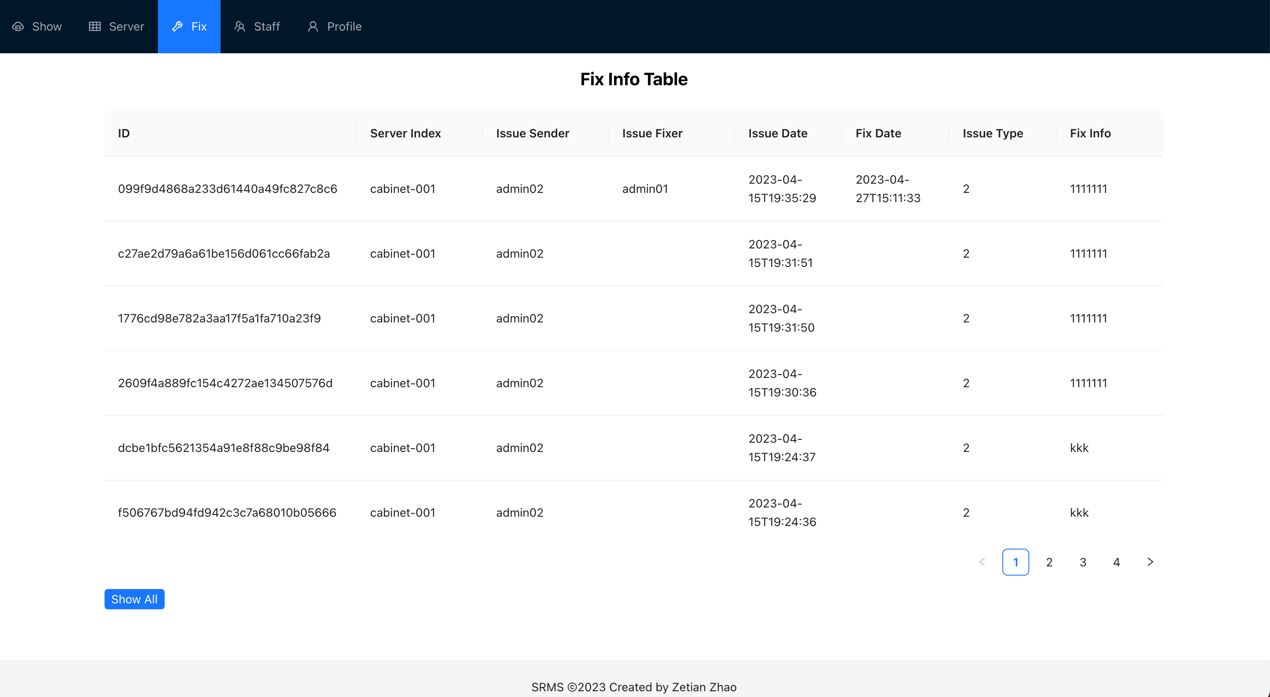
|  |
| --- |
| @GetMapping("/info")  public Result<Server> getServerInfo(@RequestParam String serverIndex) {  log.debug("-------------serverIndex:" + serverIndex);  Server server = serverService.getServerInfo(serverIndex);  return Result.success(server);  }  @GetMapping("/issued")  public Result<List<Server>> getIssued() {  List<Server> list = serverService.getIssuedServer();  log.debug("#####################issued serverList:" + list);  return Result.success(list);  }  @GetMapping("/failed")  public Result<List<Server>> getFailed() {  List<Server> list = serverService.getFailedServer();  log.debug("#####################failed serverList:" + list);  return Result.success(list);  } |

## 报修信息管理模块

报修信息管理页面仅对管理员可见，当有普通用户对某台服务器发起报修请求时，系统将自动在报修信息表中新增下一条报修信息，包含发送者、发起时间、错误类型以及错误详细内容的记录。

当管理员在系统中发现有服务器含有报错信息时，可以先更改服务器状态为False，处理完错误之后，重新修改为True，此时系统将在报错信息表的记录中补全修复管理员的用户名、修复日期等。

页面效果如图5-6所示：

****

**图5-6 报错信息管理界面**

系统自动新增报错记录的关键代码如下：

|  |
| --- |
| @PostMapping("/add")  public Result<FixInfo> createFixInfo(@Validated @RequestBody FixInfo fixInfo, BindingResult bindingResult){  if(bindingResult.hasErrors()){  String failedMsg = bindingResult.getFieldError().getDefaultMessage();  return Result.fail(failedMsg);  }  FixInfo addedFixInfo = fixInfoService.addFixInfo(fixInfo);  if (addedFixInfo != null){  return Result.success(addedFixInfo, "issue created successfully");  }  return Result.fail("issue created failed");  } |

管理员修复后系统自动补全报错记录的关键代码如下：

|  |
| --- |
| /\*\*  \*1. set fixId filed in server to null  \*2. set fixer filed in fixInfo to username  \*3. set fixDate filed in fixInfo to now  \*/  if (server.getIsWorking()){  FixInfo fixInfo = new FixInfo();  fixInfo.setId(server.getFixId());  server.setFixId(null);  fixInfo.setFixer(username);  fixInfo.setFixDate(LocalDateTime.now());  LambdaQueryWrapper<FixInfo> lambdaQueryWrapper = new LambdaQueryWrapper<>();  lambdaQueryWrapper.eq(FixInfo::getId, fixInfo.getId());  fixInfoService.getBaseMapper().update(fixInfo, lambdaQueryWrapper);  }  log.debug("$$$$$$$$$$$updated server2:"+server.toString());  LambdaQueryWrapper<Server> lambdaQueryWrapper = new LambdaQueryWrapper<>();  lambdaQueryWrapper.eq(Server::getServerIndex, server.getServerIndex());  int update = this.baseMapper.update(server, lambdaQueryWrapper); |

## 用户管理模块

用户管理页面仅对管理员可见，展示所有用户的用户信息，包括ID、用户名、手机号、电子邮件地址以及用户的角色。

页面效果如图5-7所示：

**图形用户界面, 应用程序, 网站

描述已自动生成**

**图5-7 用户管理界面**

查询所有用户信息的关键代码如下：

|  |
| --- |
| @GetMapping("/all")  public Result<List<User>> getAll() {  List<User> list = userService.list();  return Result.success(list);  } |

## 个人资料模块

个人资料页面展示当前用户的所有信息，同时提供修改个人资料的按钮，以及退出账号按钮。

页面效果如图5-7所示：

**图形用户界面, 应用程序, Teams

描述已自动生成**

**图5-8 个人资料界面**

个人资料展示、修改功能实现的关键代码如下：

|  |
| --- |
| @GetMapping("/info")  public Result<Map<String, Object>> getUserInfo(@RequestParam(name = "token") String token){  log.debug("in serviceImpl----------------"+token);  Map<String, Object> userInfo = userService.getUserInfo(token);  if (userInfo != null){  return Result.success(userInfo,"get user info successful!");  }  return Result.fail("fail to get user info ");  }  @PutMapping("/update")  public Result<Map<String, Object>> updateUserInfo(@Validated @RequestBody User user){  Map<String, Object> updatedUserInfo = userService.updateUserInfo(user);  if (updatedUserInfo != null){  return Result.success(updatedUserInfo, "update successfully");  }  return Result.success("update failed");  } |

## 本章小结

本章主要介绍了本系统各个模块的开发和实现，包括前端React框架和Three.js库的使用，后端SpringBoot框架的使用，数据存储方案使用MySql数据库，数据接口遵循RESTful规范。

在具体实现过程中，详细介绍了注册登录模块和机房模拟及监控模块的实现，包括关键代码的讲解。其中，注册登录模块实现了用户的注册、登录功能，并使用jwt验证token实现了用户权限限制和控制展示页面；机房模拟及监控模块实现了对机房的3D模拟，并在鼠标悬停到服务器上时将该服务器对应的详细信息和工作状态等展示出来。该模块使用到了Three.js这个WebGL第三方库，将3D模型导入React框架创建的Web项目中，实现在浏览器中展示3D对象的效果，关键代码包括建立场景、加载GLTF模型、选中服务器并展示信息面板等。

# 系统测试

## 测试目标和原则

测试目标：对系统的各个模块的各个功能进行测试，从而完善系统功能，检验系统安全性和稳定性。

测试原则：针对每个功能设计测试用例，采用黑盒测试的方法对系统功能性进行测试。

## 系统测试环境

系统的测试环境如下表所示：

**表6-1系统测试环境**

|  |  |  |
| --- | --- | --- |
| 系统名称 | 基于数字孪生的机房管理原型系统 | |
| 软件环境 | 操作系统 | macOS Ventura 13.3.1 |
| jdk版本 | jdk 17.0.6 |
| Node.js版本 | v18.15.0 |
| MySql版本 | Ver 8.0.32 for macos13 on arm64 |
| 浏览器版本 | Google Chrome Version 112.0.5615.137 (Official Build) (arm64) |
| 硬件环境 | CPU | Apple M2 Pro |
| 内存 | 16GB |
| 硬盘 | 512GB |
| 分辨率 | 3024\*1964 |

## 系统功能测试

### 注册功能

用户注册时，每个输入字段都有限制，如用户名应该在3-20个字符之间，密码必须包含至少一个大写字母一个小些字母和一个数字，且不少于8个字符。因此设计测试用例来观察系统的响应是否符合预期结果。

注册模块测试结果如表6-2所示：

**表6-2注册功能测试结果**

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 预期结果 | 实际结果 | 测试结论 |
| 所有字段不输入 | 注册失败 | 注册失败 | 正确 |
| 个别字段不输入 | 注册失败 | 注册失败 | 正确 |
| Username:mm | 注册失败 | 注册失败 | 正确 |
| Username:admin01 | 注册成功 | 注册成功 | 正确 |
| Phone:110 | 注册失败 | 注册失败 | 正确 |
| Phone:13633920505 | 注册成功 | 注册成功 | 正确 |
| Email:email.com | 注册失败 | 注册失败 | 正确 |
| Email: [admin01@gmail.com](mailto:admin01@gmail.com) | 注册成功 | 注册成功 | 正确 |
| Password:Admin123  RePassword:Admin124 | 注册失败 | 注册失败 | 正确 |
| Password:Admin123  RePassword:Admin123 | 注册成功 | 注册成功 | 正确 |

### 登录功能

用户登录需要用户名已存在并且密码正确。因此设计测试用例来观察系统的响应是否符合预期结果。

登录模块测试结果如表6-3所示：

**表6-3登录功能测试结果**

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 预期结果 | 实际结果 | 测试结论 |
| 所有字段不输入 | 登录失败 | 登录失败 | 正确 |
| 个别字段不输入 | 登录失败 | 登录失败 | 正确 |
| Username:user  Password:admin123 | 登录失败 | 登录失败 | 正确 |
| Username:admin01  Password:admin1234 | 登录失败 | 登录失败 | 正确 |
| Username:admin01  Password:admin123 | 登录成功 | 登录成功 | 正确 |

### 修改个人信息功能

用户修改个人信息时，应该遵循注册时字段的限制，否则将提示修改失败。因此设计测试用例来观察系统的响应是否符合预期结果。

用户修改个人信息模块测试结果如表6-4所示：

**表6-4修改个人信息功能测试结果**

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 预期结果 | 实际结果 | 测试结论 |
| 所有字段修改为空 | 修改失败 | 修改失败 | 正确 |
| 手机号字段改为空 | 修改失败 | 修改失败 | 正确 |
| Email修改为：admin.com | 修改失败 | 修改失败 |  |
| Username改为：dd | 修改失败 | 修改失败 | 正确 |
| Username改为：admin001 | 修改成功 | 修改成功 | 正确 |

### 普通用户发起报错请求

用户发起报错请求时，应该选择错误的类型，否则将提示请求失败。因此设计测试用例来观察系统的响应是否符合预期结果。

普通用户发起报错请求测试结果如表6-5所示：

**表6-5普通用户发起报错请求测试结果**

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 预期结果 | 实际结果 | 测试结论 |
| 不选错误类型  不填写错误信息 | 请求失败 | 请求失败 | 正确 |
| 选择错误类型  不填写错误信息 | 请求成功 | 请求成功 | 正确 |
| 选择错误类型  填写错误信息 | 请求成功 | 请求成功 | 正确 |

### 管理员修改服务器状态

管理员修改服务器状态的修改限制比较少。因此设计测试用例来观察系统的响应是否符合预期结果。

管理员修改服务器状态测试结果如表6-6所示：

**表6-6管理员修改服务器状态测试结果**

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 预期结果 | 实际结果 | 测试结论 |
| 状态为True  不填写服务器信息 | 修改成功 | 修改成功 | 正确 |
| 状态为False  不填写服务器信息 | 修改成功 | 修改成功 | 正确 |
| 状态为True  填写服务器信息 | 修改成功 | 修改成功 | 正确 |
| 状态为True  填写服务器信息 | 修改成功 | 修改成功 | 正确 |

### 机房模型拖拽缩放等功能

对于模拟机房及监控模块，信息清晰，易观察尤为重要，所以要对机房模型进行拖拽缩放，以及是否能展示鼠标悬停服务器的信息面板等功能进行测试。因此设计测试用例来观察系统的响应是否符合预期结果。

机房模型拖拽缩放等功能的测试结果如表6-7所示：

**表6-7机房模型拖拽缩放等功能测试结果**

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 预期结果 | 实际结果 | 测试结论 |
| 鼠标点按拖拽 | 机房模型旋转 | 机房模型旋转 | 正确 |
| 鼠标滑动滚轮 | 机房模型缩放 | 机房模型缩放 | 正确 |
| 鼠标悬停服务器 | 展示信息面板 | 展示信息面板 | 正确 |

### 长时间无操作清空信息

安全性对于机房管理系统来说尤为重要，服务器状态通常只能对内部用户可见，因此，在登陆之前，系统内所有内容均不可见，并且考虑到用户未退出系统可能将系统信息暴露给无关人员，所以在用户超过特定时间无操作的情况下，系统将自动删除登陆用户信息，所有操作均会失败。因此设计测试用例来观察系统的响应是否符合预期结果。

长时间无操作清空信息的测试结果如表6-7所示：

**表6-7长时间无操作清空信息的测试结果**

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 预期结果 | 实际结果 | 测试结论 |
| 未登录 | 所有信息为空 | 所有信息为空 | 正确 |
| 登录后清空浏览器Cookie | 所有信息为空，所有操作失败 | 所有信息为空，所有操作失败 | 正确 |
| 长时间无操作 | 所有信息为空，所有操作失败 | 所有信息为空，所有操作失败 | 正确 |

## 测试结论

在测试环境中，该系统测试对系统进行了各个模块的各个功能测试，系统的响应均符合预期结果。在每个功能测试中，设计了测试用例来观察系统的响应，全部功能都能符合预期结果。每个测试用例包括预期结果、实际结果和测试结论。根据测试结果可以判断本系统测试成功，系统相应结果符合预期结果。

## 本章小结

本章主要介绍了系统测试的目标和原则，以及系统测试环境和各模块的功能测试。测试目标是对系统的各个模块的各个功能进行测试，从而完善系统功能，检验系统安全性和稳定性。测试原则是针对每个功能设计测试用例，采用黑盒测试的方法对系统功能性进行测试。系统测试环境包括软件环境和硬件环境。各模块的功能测试包括注册功能、登录功能、修改个人信息功能、普通用户发起报错请求和管理员修改服务器状态。对每个功能分别设计测试用例，观察系统的响应是否符合预期结果，以确定系统功能是否正确。测试结果表明，系统各功能的测试结果符合预期，功能能够正常工作。

# 总结与展望

## 论文结论与总结

本论文通过分析分析传统机房管理系统的弊端，，直观清晰，有利于管理人员排查服务器错误

目前很多机房管理模式不能满足服务器机房的管理需要而暴露弊端，如机房服务器数量较大，管理接口不规范，造成了较大的管理和遇到某台服务器突发异常时的宕机服务器排查难度，以及服务器工作状态不正常并且消息更新不及时而造成主要业务受到影响的现象，机房管理人员的操作不能满足财务制度规范化的要求[1] 。

## 未来研究展望

# 参考文献

# 致 谢