|  |  |
| --- | --- |
| **学 号：** | **密 级：** |

**（封面、封底用120克白色铜版纸打印，无须彩打。所有括号内提示性内容及括号在打印时必须删除，下同。论文所有数字一律采用Times New Roman字体）**



**本科毕业设计（论文）**

**UNDERGRADUATE THESIS**



**类 型：** 毕业论文

**题 目：**基于Java的一种网络存储系统的设计与实现

**专业名称：** 数据科学与大数据技术

**入校年份：** 2023级

**学生姓名：** 梁燕

**指导教师：** 姓名 吴瀛 职称 副教授

**院(系)名称：** 机械与电气工程学院院

**完成时间：** XXXX年XX月

合肥城市学院

**本科毕业设计（论文）**

**基于Java的一种网络存储系统的设计与实现**

学生姓名： 梁燕

学生学号： 23203160225

指导教师： 姓名 吴瀛 职称 副教授

专业名称： 数据科学与大数据技术

院(系)名称： 机械与电气工程学院

年 月**（Times New Roman，小三号，年月要居中）**

**A Dissertation Submitted for the Degree of Bachelor**

**（上述英文需加粗，本文英文字母及数字一律Times New Roman字体；本页如无特殊说明字体大小均为三号不加粗；行间距均为1.25倍行距，段前段后均为0，均居中）**

**English Title（英文标题小二号，加粗居中，下方空一行三号Times New Roman）**

By

Author name**（按姓前名后书写，首字母大写）**

CITY UNIVERSITY OF HEFEI

Hefei, Anhui, P.R.China

Month, Year**（按完成时间答辩的年月时间书写，如**May,2019**）**

**（本页除签名和日期填写外，其他内容不能改动）**

**毕业设计（论文）独创性声明**

本人郑重声明：所呈交的毕业设计（论文）是本人在指导教师指导下进行独立研究工作所取得的成果。据我所知，除了文中特别加以标注和致谢的内容外，设计（论文）中不包含其他人已经发表或撰写过的研究成果，也不包含为获得 合肥城市学院 或其他教育机构的学位或证书而使用过的材料。对本文成果做出贡献的个人和集体，本人已在设计（论文）中作了明确的说明，并表示谢意。

毕业设计（论文）中表达的观点纯属作者本人观点，与合肥城市学院无关。

毕业设计（论文）作者签名： 签名日期： 年 月 日

**毕业设计（论文）版权使用授权书**

本学位论文作者完全了解 合肥城市学院 有关保留、使用毕业设计（论文）的规定，即：除保密期内的涉密设计（论文）外，学校有权保存并向国家有关部门或机构送交设计（论文）的复印件和电子光盘，允许设计（论文）被查阅或借阅。本人授权 合肥城市学院 可以将本毕业设计（论文）的全部或部分内容编入有关数据库，允许采用影印、缩印或扫描等复制手段保存、汇编毕业设计（论文）。

（保密的毕业设计（论文）在解密后适用本授权书）

|  |  |
| --- | --- |
| 学位论文作者签名： | 指导教师签名： |
| 签名日期： 年 月 日 | 签名日期： 年 月 日 |

**摘** **要**

在数据量爆发式增长的当下，单机存储的局限性日益凸显。为了高效利用有限的存储资源，提升数据存储与管理效率，降低中小企业的数据存储成本，顺应数字化转型的发展趋势，本课题对网络存储系统在国内外的研究与发展状况展开深入剖析。经研究发现，当前网络存储系统存在性能瓶颈、数据安全性不足等问题，亟待改进。为解决这些问题，本课题设计了一种基于Java的网络存储系统。该系统针对网络存储场景的特定需求，提供了创新性的解决方案。通过优化开发框架和存储结构，充分挖掘网络存储潜力，提升了数据存储性能和安全性，有效降低了中小企业的数据存储成本。

**关键词：**网络存储；云安全；Java；软件工程

# ABSTRACT

In the current period of explosive data growth, the limitations of single - machine storage are becoming increasingly prominent. As the amount of data generated by enterprises and digital applications surges exponentially, single - machine storage, with its fixed capacity and limited processing power, struggles to handle the ever - expanding data load.

To efficiently utilize scarce storage resources, enhance data storage and management efficiency, reduce data storage costs for small and medium - sized enterprises (SMEs), and adapt to the digital transformation trend, this research conducts an in - depth analysis of the research and development of network storage systems both at home and abroad. The study reveals that current network storage systems are plagued by performance bottlenecks, such as slow data access under high - load conditions, and insufficient data security, like vulnerability to cyber - attacks. These issues urgently need to be addressed.

To solve these problems, this project has designed a Java - based network storage system. Catering to the specific requirements of network storage scenarios, it provides innovative solutions. By optimizing the development framework to streamline data processing and the storage structure to improve data organization, it fully unlocks the potential of network storage. This not only boosts data storage performance and security but also effectively cuts down SMEs' data storage costs.

**KEYWORDS:** Network storage; Cloud security; Java; Software Engineering

**目 录**

[1 绪 论 8](#_Toc193403318)

[1.1 课题背景 8](#_Toc193403319)

[1.2 国内外研究现状 8](#_Toc193403320)

[1.2.1 国内研究现状 8](#_Toc193403321)

[1.2.2 国外研究现状 9](#_Toc193403322)

[1.3 课题目的及意义 9](#_Toc193403323)

[1.4 研究内容 10](#_Toc193403324)

[1.5 本章小结 10](#_Toc193403325)

[2 开发环境部署和开发技术 11](#_Toc193403326)

[2.1 开发环境综述 11](#_Toc193403327)

[2.2 开发环境部署 11](#_Toc193403328)

[2.2.1 JDK部署 11](#_Toc193403329)

[2.2.2 Postgresql数据库部署 11](#_Toc193403330)

[2.3 开发技术 12](#_Toc193403331)

[2.3.1 Solon框架 12](#_Toc193403332)

[2.3.2 SqlToy框架 13](#_Toc193403333)

[2.3.3 Postgresql数据库 14](#_Toc193403334)

[3 需求分析和系统总体结构 15](#_Toc193403335)

[3.1 可行性分析 15](#_Toc193403336)

[3.1.1 技术可行性分析 15](#_Toc193403337)

[3.1.2 经济可行性分析 15](#_Toc193403338)

[3.2 需求分析 15](#_Toc193403339)

[3.2.1 功能性需求 15](#_Toc193403340)

[3.2.2 非功能性需求 16](#_Toc193403341)

[3.3 本章小结 16](#_Toc193403342)

[4 数据库设计 17](#_Toc193403343)

[4.1 数据库逻辑设计 17](#_Toc193403344)

[4.2 数据库表的创建 18](#_Toc193403345)

[4.2.1 SQL语句实现 18](#_Toc193403346)

[4.2.2 数据库访问 20](#_Toc193403347)

[4.3 本章小结 20](#_Toc193403348)

[5 系统功能的设计与实现 22](#_Toc193403349)

[5.1 后端接口的设计 22](#_Toc193403350)

[5.1.1 登录接口 22](#_Toc193403351)

[5.1.2 文件接口 22](#_Toc193403352)

[5.1.3 文件夹接口 25](#_Toc193403353)

[5.1.4 安全验证接口 27](#_Toc193403354)

[5.2 前端页面的设计 28](#_Toc193403355)

[5.2.1 登录注册页面 28](#_Toc193403356)

[5.2.2 文件管理页面 28](#_Toc193403357)

[5.2.3 分享文件页面 28](#_Toc193403358)

[5.3 本章小结 28](#_Toc193403359)

[6 测试及运行 28](#_Toc193403360)

[6.1 系统功能性测试 28](#_Toc193403361)

[6.1.1 登录模块功能测试 28](#_Toc193403362)

[6.1.2 文件操作模块功能测试 29](#_Toc193403363)

[6.1.3 文件夹操作模块功能测试 29](#_Toc193403364)

[6.2 系统性能测试 30](#_Toc193403365)

[6.3 系统安全性测试 30](#_Toc193403366)

[7 总结与展望 31](#_Toc193403367)

[7.1 课题总结 31](#_Toc193403368)

[7.2 课题展望 31](#_Toc193403369)

[参考文献 32](#_Toc193403370)

[致 谢 33](#_Toc193403371)

1 绪 论

* 1. 课题背景

在现如今信息技术和经济水平的带动下，计算机在日常生活中已非常普遍，近年来网络速度大幅提升后，人们的存储习惯正由物理存储向云存储转变。各大网络厂商也在不断地推出自家的云存储系统软件，像百度网盘，阿里云盘等产品，虽然也推出了企业版供企业使用，但在数据安全、使用投入等方面依然无法满足企业需求。大部分企业往往会选择自建云存储系统以便进行需求定制。依托云计算技术的成熟，云存储系统得以获得更低的成本但拥有更好性能表现，如近几年流行的开源云计算平台OpenStack。其以资源的形式整合了软件定义网络技术、存储服务、计算服务、认证等功能，同时也通过扩展具备了监控、裸设备等功能。总的来说，云存储技术对整个互联网和社会发展都具有重要意义。

但目前的云存储系统大多服务于拥有大规模集群的用户，对于中小型企业的服务器资源利用不够充分。基于此，本课题首先分析了云存储系统的现状和需求，明确了系统的功能；设计系统功能模块，包括数据库设计、后端处理设计、前端页面设计；然后进行功能实现，包括开发前端界面、用户登录功能、文件操作功能、资源调度功能等；最后进行功能测试，确保系统可用性和用户体验；在设计的过程中，高度关注安全问题，对常见的Web漏洞，如：越权、目录穿梭、Sql注入等进行防护，提高了系统的安全性。

* 1. 国内外研究现状

1.2.1 国内研究现状

国内许多学者就云计算架构进行了大量研究，如在数据一致性验证上李帅、陈越等人提出了一种基于ZSS（Zhang，Safavi-Naini，Susilo)短签名轻量级云存储动态数据完整性验证方法。云平台往往是以软硬件资源为依据，提供相应的存储、网络、计算平台，因而也称之为云计算平台。如今，国内外许多公司都提供云平台服务。如百度、阿里、腾讯等企业均推出了云计算平台和服务，并将云平台当作企业重要的发展方向。

阿里云是目前国内最大的云计算平台，也在国际上处于领先队伍，对国内云计算和云存储的发展有领军意义，有重要的参考价值。虽然阿里云的创立时间相对较晚，但阿里云发展迅速，将其业务扩展至海外，为许多国家提供服务。

对于阿里云的云存储系统而言，以阿里云盘为主要产品，使云服务器 ECS 拥有了相应的数据块级随机存储服务，且具有高可靠、持久性、低时延的特点。阿里云盘按 I/O 质量和底层存储集群的硬盘材质划分，分别为普通云盘、SSD 云盘。其中，对于普通云盘而言，其本身具有较低的价格，但是在访问时延、吞吐量、IOPS 等指标方面并不具备较高的优势，因而主要以低 I/O 负载、不需要被经常访问的应用场景为主；而对于 SSD 云盘而言，其在访问时延、吞吐量、IPOS 方面都具有一定的优势，可以在 NoSQL 数据库、中大型关系数据库、IVO 密集型应用中使用。

1.2.2 国外研究现状

早在 2006 年，亚马逊公司就专门研究了亚马逊云（AWS），并依据 AWS 向为全球 190 个国家/地区的数十万家企业提供计算能力、数据库存储、内容交付和其他功能。到了 2010 年，AWS 更是得到了全世界都关注，美国不少创业公司直接将自己的应用和服务构建在 AWS 上。其中，被广泛使用的 Amazon S3 便是亚马逊公司的简单存储业务，该服务开发了统一的以 RESTful 接口，构建了 API 接口，提供各类存储服务。企业用户基于 S3 接口，可以突破本地存储服务器的限制，方便的构建基于亚马逊云的 Web 应用，存储应用数据到云端，并快速的进行数据检索。

1.3 课题目的及意义

在云原生时代，网络存储方式越来越成为企业和个人存储数据的首要选择。但是现在网络存储系统往往内存占用高，功能繁杂，安全性较低，不符合国家“绿色计算”的号召。基于此，本课题开发了一个基于国产Java框架Solon的轻量级网络存储系统，旨在为用户提供高效、便捷、安全的存储服务。该系统追求轻量和快速响应，以满足小型团队或个人对存储资源的需求。通过采用Solon框架，利用其轻量级和高性能的特点，实现系统的简洁构建和部署。同时，将机器学习应用融入其中，尤其是在请求处理阶段引入恶意 URL 识别模型，这一创新点旨在提高系统的安全性，能够在请求进入时对其进行恶意 URL 的检测，为用户存储的数据提供更安全的保障，避免因恶意请求而导致的数据泄露或系统破坏。

本课题凭借轻量级和低成本特性，为预算和技术资源有限的小型团队提供经济实用的存储方案，助力其将更多资源投入业务发展；简洁易用界面与强大安全防护功能，也让个人用户能安心存储重要数据，提升使用体验和安全感。在技术创新上，采用国产 Java 框架 Solon，为网络存储系统构建与部署提供新思路，推动国产技术在该领域的应用与发展；引入机器学习进行恶意 URL 检测，为其他存储系统安全建设提供借鉴，丰富网络安全防护手段。从行业发展来看，系统填补针对小型团队和个人用户的轻量级、高安全性存储系统市场空白，促使行业关注细分市场，推动产品服务多元化，提升行业服务水平与竞争力，同时符合 “绿色计算” 号召，树立绿色存储典范，推动行业注重资源合理利用与环境友好。

1.4 研究内容

本课题对国内外云存储系统的研究与发展进行了深入分析。在研究中，发现了当前云存储系统存在的一些问题，并提出了改进的方案。为了解决这些问题，设计了一种基于国产开源框架Solon的轻量级云存储系统。该系统专注于满足小规模集群的云存储需求，并提供了全新的解决方案。通过整合多台云服务器的零散存储资源，该系统充分利用了每一寸“土地”，从而降低了企业的运营成本。

1.5 本章小结

在信息技术飞速发展、经济持续进步的当下，云存储日益普及。各大网络厂商推出云存储产品，但其企业版在数据安全与使用成本上无法满足企业需求，许多企业选择自建云存储系统。云计算技术成熟，如 OpenStack 整合多种功能，推动云存储发展，不过当前云存储系统多服务大规模集群用户，中小企业服务器资源利用不足。

国内学者深入研究云计算架构，李帅、陈越等人提出基于 ZSS 短签名的验证方法。百度、阿里、腾讯等企业推出云计算平台，阿里云是国内云计算的佼佼者，业务拓展至海外。其云存储系统以阿里云盘为核心，分普通云盘和 SSD 云盘，适用于不同应用场景。国外亚马逊公司 2006 年研究亚马逊云（AWS），为全球企业提供多样功能，2010 年备受关注，其 Amazon S3 简单存储业务提供统一接口，方便企业构建云端应用。

本课题针对现有云存储系统问题，深入分析国内外研究与发展情况，提出基于国产开源框架 Solon 的轻量级云存储系统设计方案。该系统专注满足小规模云存储需求，整合零散存储资源，提升资源利用率，降低企业运营成本，为云存储领域提供新思路。后续将围绕系统设计、实现与测试展开研究，验证其可行性与实用性。

2 开发环境部署和开发技术

2.1 开发环境综述

本系统部署在一台i5-12450H的windows PC上，JDK版本为21，Postgresql版本为16.6。开发IDE为IntelliJ IDEA Ultimate 2024.1。

2.2 开发环境部署

2.2.1 JDK部署

本系统开发基于Java语言实现，需通过JDK（Java Development Kit）提供编译与运行环境。JDK（Java Development Kit） 是 Java 程序开发的核心工具包，包含了开发 Java 应用程序所需的所有资源。它为开发者提供了编写、编译、调试和运行 Java 程序的工具和类库。JDK的部署过程如下：

（一）JDK版本选择

选择稳定且兼容性强的JDK21版本，本课题需要用到虚拟线程，其支持虚拟特性可保障开发与运行环境的稳定性。

（二）安装步骤

首先下载JDK安装包，访问Oracle官网获取对应操作系统的安装。接下来执行安装程序，选择默认路径安装“Add Java to PATH”选项以简化环境变量配置。然后验证安装，打开命令行工具，输入命令“java --version”验证版本信息，成功显示版本号（java 21.0.4）表明基础环境已就绪。

（三）环境变量配置

在Windows系统中进行配置，第一步右键点击此“此电脑”，选择属性 → 高级系统设置 → 环境变量。第二步新建系统变量JAVA\_HOME，值为JDK安装路径。第三步编辑path变量，追加%JAVA\_HOME%\bin。

2.2.2 Postgresql数据库部署

在本系统的开发中，PostgreSQL 数据库作为核心的数据存储和管理组件，其正确部署对于系统的稳定运行和数据的有效管理至关重要。本次部署的目标是搭建一个具备高可用性、高性能和轻量化的 PostgreSQL 数据库环境，以满足系统在数据存储和处理方面的需求。

（一）安装包获取

在postgresql官网获取windows系统的安装包，为更好的兼容，选择最新的版本进行安装（postgresql16.6）。

（二）postgresql安装

首先双击下载的.exe文件来运行安装程序，以管理员身份启动安装向导。接下来选择安装目录（C:\PostgreSQL\16），需要选择勾选核心组PostgreSQL Server来更好的支撑系统的数据运行。

（三）postgresql配置

数据库初始化配置：首先设置超级用户（postgres）密码，需符合安全规范，选择默认端口，选择数据库集群的本地化设置（字符集UTF-8）。接下来进行服务配置，设置Windows服务名称，配置服务启动账户后便安装成功。

环境变量配置：将PostgreSQL的bin目录（C:\PostgreSQL\16\bin）添加到系统环境变量PATH中，以便命令行直接调用psql等工具。

2.3 开发技术

2.3.1 Solon框架

Solon是一个Java “生态型”应用开发框架。从零开始构建，有自主的标准规范与开放生态。目前已近14万行代码。其追求更快、更小、更简单。同时支持 java8、java11、java17、java21；组合不同的插件应对不同需求；方便定制；快速开发。

特点：

1.HTTP、WebSocket、Socket三种信号统一的开发体验（俗称：三源合一）

2.支持“注解”与“手动”两种模式并重，按需自由操控

3.Not Servlet，可以适配任何 Http 通讯框架（所以：最小 0.3m 运行rpc架构）

4.独特的 IOC/AOP 容器设计。不会因为依赖变多而启动很慢

5.适合 Web、Scheduling、FaaS、Remoting、Cloud 等任何开发场景

6.兼顾 Handler + Context 和 Listener + Message 两种架构模式

7.强调插件式扩展，可扩展可切换；适应不同的应用场景

8.支持 GraalVm Native Image 打包

9.允许业务插件“热插”、“热拔”、“热管理”

Solon的生态架构图如图2.3.1所示图示例：

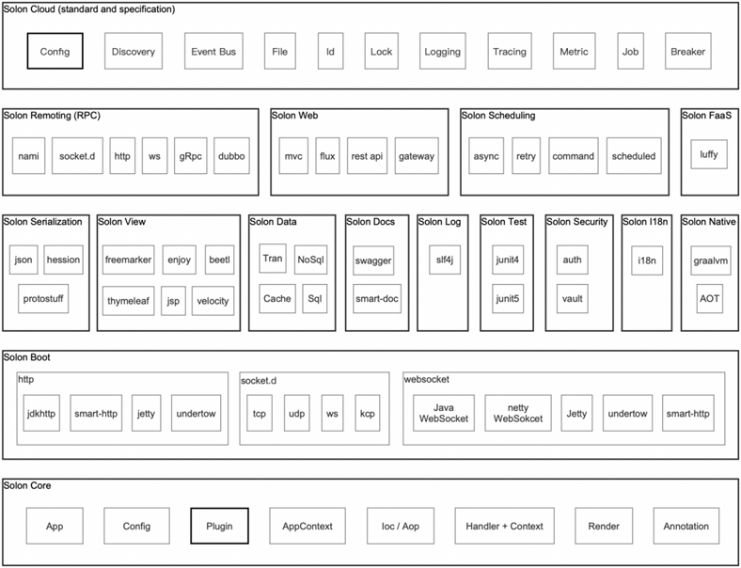


图2.3.1 Solon生态架构图

2.3.2 SqlToy框架

sqltoy-orm是比JPA+MyBatis更加贴合项目的orm框架，具有jpa式的对象CRUD的同时具有比myBatis(plus)更直观简洁性能强大的查询功能。

特点：

1.类似JPA的对象化CRUD、对象级联加载和新增、更新

支持通过POJO生成DDL以及直接向数据库创建表

2.强化update操作，提供弹性字段修改能力，不同于hibernate先load后修改，而是一次数据库交互完成修改，确保了高并发场景下数据的准确性

3.改进了级联修改，提供了先删除或者先置无效，再覆盖的操作选项

4.增加了updateFetch、updateSaveFetch功能，强化针对强事务高并发场景的处理，类似库存台账、资金台账，实现一次数据库交互，完成锁查询、不存在则插入、存在则修改，并返回修改后的结果

5.增加了树结构封装，便于统一不同数据库树型结构数据的递归查询

6.支持分库分表、支持多种主键策略(额外支持基于redis的产生特定规则的业务主键)、加密存储、数据版本校验

7.提供了公共属性赋值(创建人、修改人、创建时间、修改时间、租户)、扩展类型处理等

8.提供了多租户统一过滤和赋值、提供了数据权限参数带入和越权校验

9.极为直观的sql编写方式，便于从客户端<-->代码，双向快速迁移，便于后期变更维护

10.支持缓存翻译、反向缓存匹配key代替like模糊查询

11.提供了跨数据库支持能力：不同数据库的函数自动转换适配，多方言sql根据实际环境自动匹配、多数据库同步测试，大幅提升了产品化能力

12.提供了取top记录、随机记录等特殊场景的查询功能

13.提供了最强大的分页查询机制:1)自动优化count语句;2)提供基于缓存的分页优化，避免每次都执行count查询;3)提供了独具特色的快速分页；4)提供了并行分页

14.所有基于sql和jdbc 各类数据库查询

2.3.3 Postgresql数据库

PostgreSQL，也称为 Postgres，是一种开源关系数据库，以其可靠性、灵活性和对开放技术标准的支持而享有盛誉。PostgreSQL 支持非关系和关系数据类型。它被称为当今可用的最兼容、最稳定和最成熟的关系数据库之一，并且可以轻松处理复杂的查询。PostgreSQL 是一个“一刀切”的解决方案，适用于许多寻求经济高效的方法来改进其数据库管理系统 (DBMS) 的企业。它具有足够的可扩展性和通用性，可以通过强大的扩展生态系统快速支持各种专业用例，涵盖时间序列数据类型和地理空间分析等工作。作为开源数据库解决方案构建的 PostgreSQL 完全没有许可限制、供应商锁定的可能性或过度部署的风险。PostgreSQL 通过对象关系数据库管理系统 (ORDBMS) 进行管理。

PostgreSQL 为负责管理业务活动的在线事务处理 (OLTP)协议的企业数据库管理员提供了理想的解决方案，包括电子商务、客户关系管理系统 (CRM) 和财务分类帐。它也是管理接收、创建和生成的数据分析的理想选择。

主要优点：

1.性能和可扩展性——包括地理空间支持和无限制的并发性——以及跨多种数据类型的深入、广泛的数据分析。

2.通过使用多版本并发控制 (MVCC)支持并发，这使得写操作和读操作同时发生。

3.由于它的兼容性和对多种编程语言的支持，包括Python、Java、JavaScript、C/C++ 和 Ruby，因此支持深度语言。

4.业务连续性，通过跨服务器的异步或同步复制方法提供高可用性服务。

5.通过开源数据库管理技术实现更大的灵活性和成本效益的创新。

3 需求分析和系统总体结构

3.1 可行性分析

3.1.1 技术可行性分析

技术可行性是对客户需求以及现有技术的综合分析评估。通过对客户提出的功能以及性能需求进行分析，结合现有开发技术，确定完成功能以及性能需求所需要解决的技术难题，并着重解决这些技术难题。

本系统基于当今流行的开发技术，使用PostgreSQL数据库存储数据，不仅能够存储、处理海量数据，还可以充分保持数据一致性、安全性、可靠性和高性能。与此同时，采用国产企业级开发框架Solon + SqlToy + Hutool 进行开发，模块化程度高，易于功能维护与扩展。并且开发语言Java在21版本加入虚拟线程后，并发处理能力也是首屈一指。可以说，本系统开发在技术上也是完全能够满足开发需求的。

3.1.2 经济可行性分析

经济可行性是评估系统开发、运行、维护的成本以及经济效益。通过对学生信息管理系统开发所投入的人力成本、服务器成本、后期维护成本等进行评估。确保系统能够顺利开发。并为管理决策提供科学依据，作为进一步开展工作的基础，提高了工作效率。

通过对国内企业的抽样调查，发现企业对系统开发的成本是在企业承受范围之内的，本云存储系统在开发并使用后，可以提高企业的云存储能力和服务器利用效率。所以，开发轻量级云存储系统在经济上是可行的。

3.2 需求分析

3.2.1 功能性需求

一个云存储系统应该给用户提供包括登录和注册功能的用户权限功能、包括文件上传、下载、删除、分享、重命名等功能的文件操作功能、包括文件夹创建、删除、重命名等功能的文件夹操作功能，以及识别恶意请求的安全验证功能。本课题的系统总体方案框图如图3.1所示：

图示例

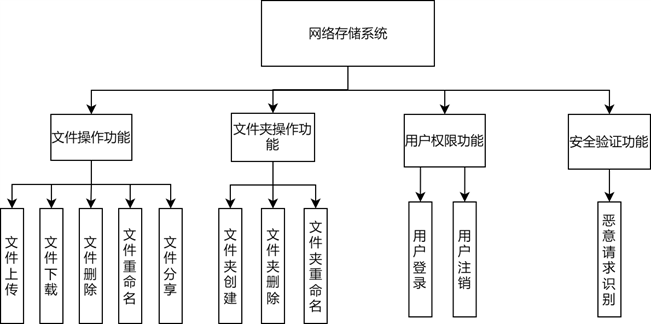


图3.1 系统总体方案框图

3.2.2 非功能性需求

本课题设计初衷是为了多用户提供网络存储服务，因而需要合理分析各功能用户接受的等待时间。并发场景下，每个用户显示文件列表的等待时间应该在1秒以内，文件上传、文件下载、文件分享、文件重命名等操作应该在3秒内获得响应。并且在响应错误率也要控制在1%以下。

云存储系统上存储着众多用户的不同文件，为了避免可能存在的系统安全隐患，应做好数据隔离和访问控制，让用户数据不会互相干扰，同时严格控制权限，保护用户的数据隐私。在保护数据的同时，针对常见的Web应用漏洞进行防护，防止恶意攻击者利用如：SQL注入、文件上传、目录穿梭等手段影响系统稳定性。

3.3 本章小结

本章围绕云存储系统展开了需求分析与系统总体结构探讨。在可行性分析中，技术层面，基于流行开发技术，采用 PostgreSQL 数据库、国产企业级开发框架 Solon + SqlToy + Hutool 以及 Java 21 版本，从数据存储、功能维护扩展到并发处理能力，均能满足开发需求；经济层面，经成本评估和对国内企业抽样调查，开发成本在企业承受范围，且系统投入使用可提升云存储能力与服务器利用效率，开发具备经济可行性。

需求分析部分，功能性需求涵盖用户权限、文件操作、文件夹操作及安全验证功能；非功能性需求则聚焦性能与安全，性能上要求并发场景下用户操作响应时间短，错误率控制在 1% 以下，安全方面强调数据隔离、访问控制及 Web 应用漏洞防护，确保系统稳定、安全运行，为后续系统设计与开发奠定了坚实基础。

4 数据库设计

4.1 数据库逻辑设计

数据操作是一个系统的核心，主要包括添加、删除、修改、查询。数据库设计是一个系统开发的基础，在设计时要保证数据的一致性、完整性、可靠性和数据库的高性能。基于此，本系统使用PostgreSQL数据库存储数据，并设计了包括用户、文件两个实体，本系统的E.R图如下图4.1所示：

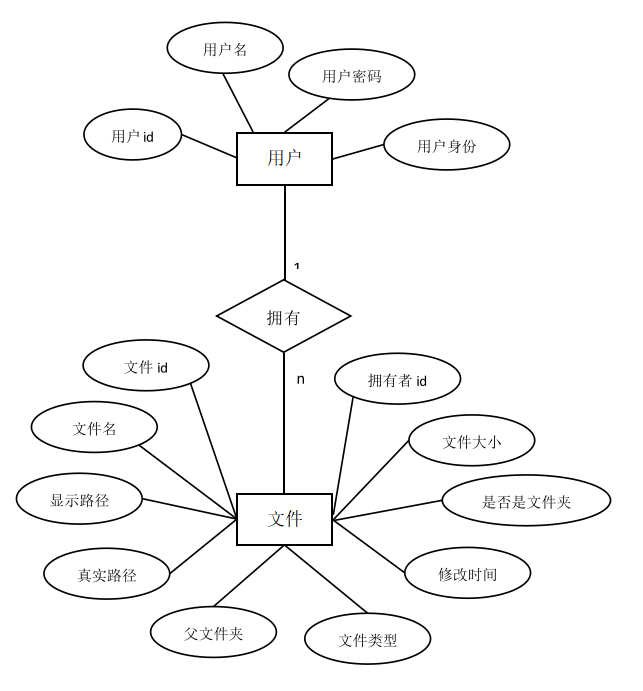


图4.1 系统E.R图

从E.R图中可以得出本系统的逻辑结构，其中用户表包含用户id、用户名、用户密码、用户身份四个字段，用户id为主键，用户身份用0或1区分管理员和普通用户。文件表包含文件id、文件名、显示路径、真实路径、父文件夹、文件类型、修改时间、是否是文件夹、文件大小以及拥有者id十个字段，其中文件id为主键，拥有者id为用户表中用户id映射的外键。由于系统中包含大量以文件名或父文件夹为查询条件的操作，所以为文件名和父文件夹分别添加了基于b+tree的索引。

完整的数据表结构如下：

1. 用户表

表4.1 用户表

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段类型 | 字段说明 |
| uid | varchar | 管理员id |
| uname | varchar | 用户名 |
| passwd | varchar | 密码 |
| identity | smallint | 用户身份 |

1. 用户表

表4.2 文件表

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段类型 | 字段说明 |
| fid | varchar | 管理员id |
| fname | varchar | 文件名 |
| view\_path | varchar | 现实路径 |
| real\_path | varchar | 真实路径 |
| parent\_path | varchar | 父文件夹 |
| update\_time | date | 修改时间 |
| dir | boolean | 是否是文件夹 |
| ftype | varchar | 文件类型 |
| uid | varchar | 拥有者id |
| size | bigint | 文件大小（Kb） |

4.2 数据库表的创建

4.2.1 SQL语句实现

首先创建名为fastfs的数据库，并设置字符编码为UTF-8，建库语句如下：

CREATE DATABASE `fastfs` WITH TEMPLATE = template0 ENCODING = 'UTF8' LOCALE\_PROVIDER = libc LOCALE = 'en\_US.utf8';

再建立user表作为存储用户信息的数据库，并创建user.id序列作为用户表主键的自增序列，建表语句如下：

CREATE TABLE public."user" (

uid character varying NOT NULL,

uname character varying(255) NOT NULL,

passwd character varying(255) NOT NULL,

identity smallint NOT NULL

);

ALTER TABLE public."user" OWNER TO fastfs;

CREATE SEQUENCE public.user\_id

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

ALTER SEQUENCE public.user\_id OWNER TO fastfs;

最后建立file表作为存储文件信息的数据库，并创建file.id序列作为文件表主键的自增序列，建表语句如下：

CREATE TABLE public.file (

fid character varying NOT NULL,

fname character varying NOT NULL,

view\_path character varying NOT NULL,

real\_path character varying NOT NULL,

parent\_path character varying NOT NULL,

update\_time date NOT NULL,

dir boolean NOT NULL,

ftype character varying NOT NULL,

uid character varying NOT NULL,

size bigint NOT NULL

);

ALTER TABLE public.file OWNER TO fastfs;

CREATE SEQUENCE public.file\_id

START WITH 0

INCREMENT BY 1

MINVALUE 0

NO MAXVALUE

CACHE 1;

ALTER SEQUENCE public.file\_id OWNER TO fastfs;

4.2.2 数据库访问

传统的Java应用访问数据库往往采用JDBC方式，繁琐且灵活度不高，本课题为贯彻轻量级开发的理念，采用国产开源ORM框架SQLToy，仅需在application.yml配置文件中添加数据源即可完成数据库访问配置，配置如下：

datasource:

driver-class-name: org.postgresql.Driver

username: root

password: root

jdbcUrl: jdbc:postgresql://192.168.1.100:5432/fastfs?useSSL=false&useUnicode=true&characterEncoding=UTF-8

配置完成后，即可在项目任意位置注入LightDao对象，通过LightDao进行数据库的增删改查。

4.3 本章小结

本章聚焦于数据库设计，着重阐述了数据库逻辑设计以及数据库表的创建相关内容。

在数据库逻辑设计环节，鉴于数据操作在系统中的核心地位以及对数据一致性、完整性、可靠性和高性能的要求，本系统选用 PostgreSQL 数据库存储数据。设计涉及用户和文件两个实体，依据 E.R 图构建出系统逻辑结构。用户表涵盖用户 id、用户名、用户密码和用户身份字段，其中用户 id 作为主键，以 0 或 1 区分管理员和普通用户身份；文件表包含文件 id、文件名等十个字段，文件 id 为主键，拥有者 id 作为外键关联用户表，且针对文件名和父文件夹添加基于 b+tree 的索引，以优化查询操作。

在数据库表创建方面，通过 SQL 语句实现建库与建表。首先创建名为 fastfs 的数据库，设置字符编码为 UTF-8；接着分别创建 user 表和 file 表存储用户与文件信息，并为两张表的主键创建自增序列。在数据库访问上，摒弃传统繁琐且灵活度低的 JDBC 方式，采用国产开源 ORM 框架 SQLToy，仅在 app.yml 配置文件中添加数据源，完成数据库访问配置后，便可在项目中通过注入 LightDao 对象进行数据库的增删改查操作，贯彻了轻量级开发理念，为系统的数据管理与操作奠定了坚实基础。

5 系统功能的设计与实现

5.1 后端接口的设计

5.1.1 登录接口

用户登录模块作为云存储系统的入口，需要对用户输入的用户名和密码进行识别校验，只有校验成功的用户才能进行文件上传、文件下载、文件删除、文件重命名、文件分享等操作。用户登录模块的流程图如下图 5.1 所示：

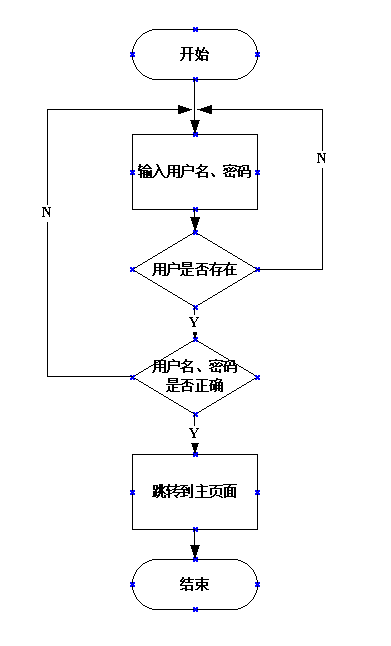
****

图5.1 用户登录流程图

5.1.2 文件接口

文件接口主要包括文件上传、文件下载、文件删除、文件重命名、文件分享和分享文件下载六大功能。

文件上传操作需要用户处于登录状态，并且上传的文件名不能带有路径分隔符“/”，否则会返回上传失败，上传成功后，系统会将文件的存储信息保存到数据库中，最后返回上传成功的json数据包给用户。文件是上传的流程图如图5.2所示：

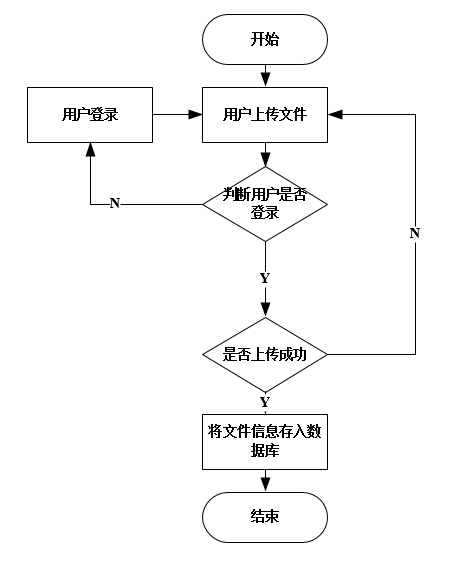


图5.2 文件上传流程图

文件下载操作同样需要先验证用户登录状态，通过用户传入的文件名在数据库中寻找对应文件的真实路径，并通过真实路径在对应的目录下载文件，因为隐藏了具体的目录，无法通过目录穿梭等漏洞非法访问其他文件，安全性较高。文件下载的流程图如图5.3所示：

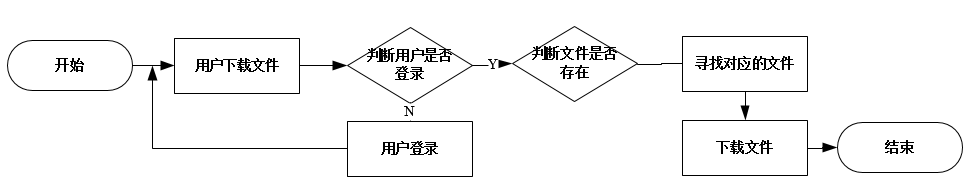


图5.3 文件下载流程图

文件删除操作之前也需要验证用户是否登录，若登录则根据文件名在数据库中寻找对应的文件真实路径，并调用对应从节点的文件删除操作，当删除成功后，返回删除成功的json数据包到主节点，主节点接收到成功的数据包后将数据库中对应文件的数据删除，最后返回删除成功的json数据包给用户，文件删除的流程图如图5.4所示：

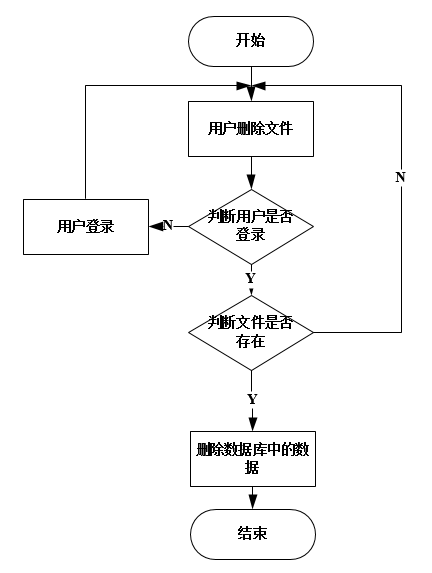


图5.4 文件删除流程图

文件重命名同样需要用户登录后才能进行操作，由于数据库设计中将文件实际名称和显示名称分离，所以文件重命名只需将数据库中对应文件的显示路径字段修改即可，由于只涉及数据库操作，响应速度较快。文件重命名流程图如图5.5所示：

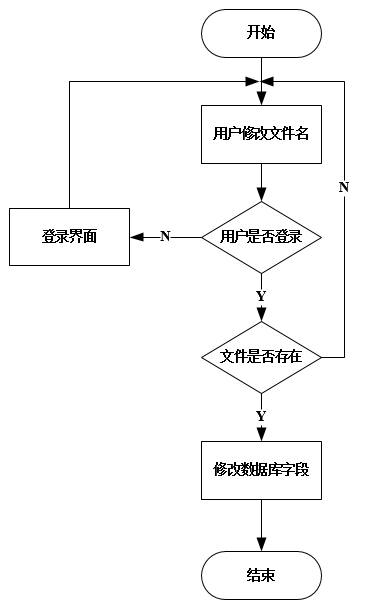


图5.5 文件重命名流程图

文件分享操作仅限登录用户使用，在用户选择好要分享的文件后，系统会在缓存中寻找文件是否已分享，若已分享，判断文件是否过期，若未过期，将缓存中的下载分享文件URL返回给用户。系统每3秒会清理一次过期缓存。文件分享操作流程图如图5.6所示：

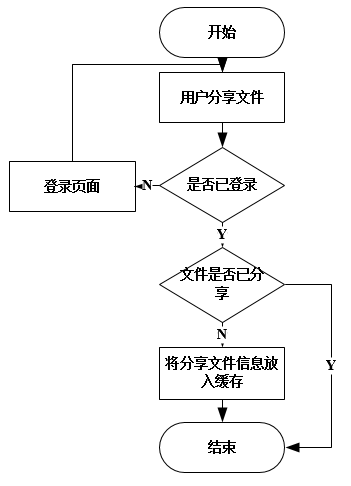


图5.6 文件分享流程图

分享文件下载主要用于从分享文件获得的下载URL进行下载，直接通过调用从节点的文件下载操作进行，由于没有进过主节点中转，响应效率较高。分享文件下载的流程图如图5.7所示：

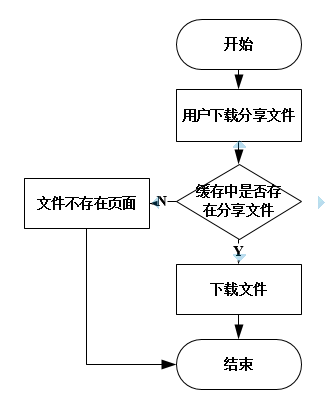


图5.7 分享文件下载流程图

5.1.3 文件夹接口

新建文件夹需要用户已登录，并且创建的文件夹名不能带有特殊字符，否则会返回创建失败。文件夹创建不涉及文件IO操作，IO操作只在文件上传时产生，由于只涉及数据操作，响应速度较快。文件夹创建的流程图如图5.8所示：

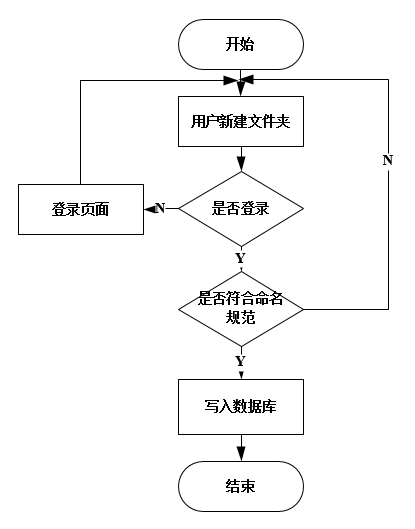


图5.8 文件夹创建流程图

文件夹重命名同样需要用户已登录，新文件夹名需要满足文件夹命名规范，不能含有特殊字符。文件夹重命名同样不涉及文件IO，只涉及数据库操作，效率较高。文件夹重命名流程图如图5.9所示：

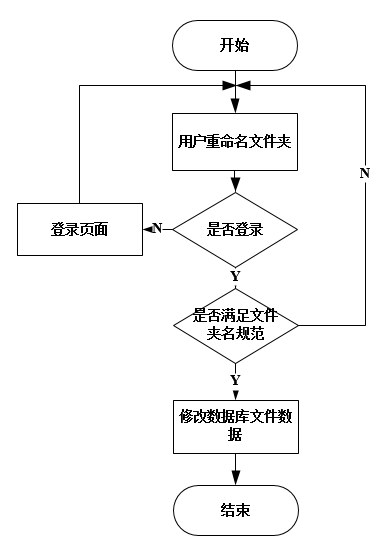


图5.9 文件夹重命名操作

文件夹删除功能与其他文件夹操作相似，都需要用户登录后操作。当用户进行文件夹删除操作时，系统会以要删除的文件夹为父文件夹为条件搜索数据库中所有满足条件的文件，当所有文件都删除后，返回删除成功给用户。由于涉及网络IO，效率会稍低。文件夹删除的流程图如图5.10所示：

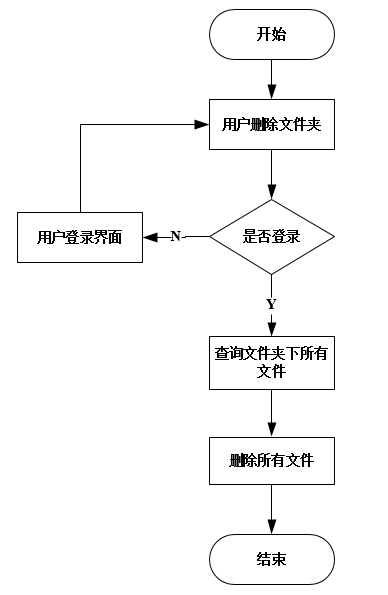


图5.10 文件夹删除流程图

5.1.4 安全验证接口

系统在接收用户请求时，有时会受到恶意请求的攻击，如执行恶意脚本，反序列化攻击，目录穿梭漏洞攻击等。一个健康的软件系统需要对这些恶意请求进行拦截处理。当一个请求进入系统时，系统会将请求url转给请求识别器，当识别器返回正常请求时才会放行，若识别为恶意请求，则将请求重定向到拦截结果页面。安全验证接口的流程图如图5.11所示：

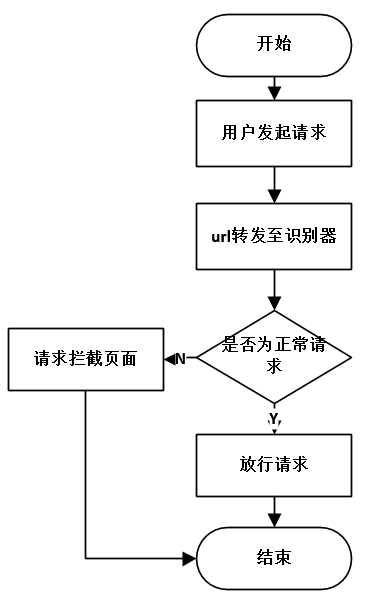


图5.11 安全验证流程图

5.2 前端页面的设计

5.2.1 登录注册页面

登录页面包含一个输入表单和一个注册页面的提示，需要输入的内容包括用户的账户和密码，当点击登录按钮后，系统会将账户和密码发给后端的验证接口。根据后端返回的结果，给出对应的提示，如果登录成功则跳转到首页。页面效果展示如图5.12所示：



图5.12 登录页面展示

注册页面也包含一个表单，包含用户的用户名和密码，当点击注册按钮时，系统将输入的信息发送给后端接口，再根据结果给用户相应的提示。注册页面展示如图5.13所示：



图5.13 注册页面展示

5.2.2 文件管理页面

文件管理页面包含当前位置和登录用户信息，以及返回上级、刷新、上传文件和新建文件夹按钮。在页面中心的文件列表部分，左键点击文件图标会出现下载、删除、分享和重命名四个功能按钮。

点击上传文件会弹出上传文件的iframe页面

5.2.3 分享文件页面

5.3 本章小结

6 测试及运行

6.1 系统功能性测试

本文在测试过程中针对具体需求，采用黑盒测试技术来完成功能测试，对系统各个功能点设计全面的测试用例来检验。测试覆盖登录功能、文件操作功能、文件夹操作功能。

6.1.1 登录模块功能测试

根据上文的分析，将登录模块划分成如下等价类：

表6.1 登录模块等价类

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 有效等价类 | 编号 | 无效等价类 | 编号 |
| 普通用户 | 正确输入账号密码 | 1 | 正确输入账号密码 | 2 |
| 管理员用户 | 正确输入账号密码 | 3 | 正确输入账号密码 | 4 |

根据划分的等价类，编写的测试用例如下：

表6.2 登录模块

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 输入 | 预期输出 | 实际输出 | 覆盖的等价类编号 |
| 1 | uanme=user&passwd=123 | 登录成功 | 登录成功 | 1 |
| 2 | uanme=user&passwd=111 | 登录失败 | 登录失败 | 2 |
| 3 | uanme=admin&passwd=123 | 登录成功 | 登录成功 | 3 |
| 4 | uanme=admin&passwd=111 | 登录失败 | 登录失败 | 4 |

6.1.2 文件操作模块功能测试

根据上文的分析，将文件操作模块划分成如下等价类：

表6.3 文件操作等价类

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 有效等价类 | 编号 | 无效等价类 | 编号 |
| 用户 | 登录后进行文件上传 | 1 | 未登录进行文件上传 | 2 |
|  | 登录后进行文件下载 | 3 | 未登录进行文件下载 | 4 |
|  | 登录后进行文件删除 | 5 | 未登录进行文件删除 | 6 |
|  | 登录后进行文件分享 | 7 | 未登录进行文件分享 | 8 |

表6.4 文件操作模块测试用例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 输入 | 预期输出 | 实际输出 | 覆盖的等价类编号 |
| 1 | 未登录访问系统 | 登录页面 | 登录页面 | 2、4、6、8 |
| 2 | 登录状态上传文件 | 上传成功 | 上传成功 | 1 |
| 3 | 登录状态下载文件 | 下载成功 | 下载成功 | 3 |
| 4 | 登录状态删除文件 | 删除成功 | 删除成功 | 5 |
| 5 | 登录状态分享文件 | 分享成功 | 分享成功 | 7 |

6.1.3 文件夹操作模块功能测试

根据上文的分析，将文件夹操作模块划分成如下等价类：

表6.5 文件夹操作等价类

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 有效等价类 | 编号 | 无效等价类 | 编号 |
| 用户 | 登录后进行文件夹创建 | 1 | 未登录进行文件夹创建 | 2 |
|  | 登录后进行文件夹重命名 | 3 | 未登录进行文件夹重命名 | 4 |
|  | 登录后进行文件夹删除 | 5 | 未登录进行文件夹删除 | 6 |

根据划分的等价类，编写的测试用例如下：

表6.6 文件夹操作模块测试用例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 输入 | 预期输出 | 实际输出 | 覆盖的等价类编号 |
| 1 | 未登录访问系统 | 登录页面 | 登录页面 | 2、4、6 |
| 2 | 登录状态文件夹创建 | 创建成功 | 创建成功 | 1 |
| 3 | 登录状态文件夹重命名 | 重命名成功 | 重命名成功 | 3 |
| 4 | 登录状态文件夹删除 | 删除成功 | 删除成功 | 5 |

6.2 系统性能测试

6.3 系统安全性测试

7 总结与展望

7.1 课题总结

7.2 课题展望

参考文献

致 谢