深圳技术大学
SHENZHEN TECHNOLOGY UNIVERSITY

# 本科毕业论文（设计）

**题目：面向协同办公的文件管理系统设计与实现**


| | |
|---|---|
| 姓　　名 | 陈杰森 |
| 学　　院 | 大数据与互联网学院 |
| 专　　业 | 物联网工程 |
| 学　　号 | 202002010306 |
| 指导教师 | 史诗洁 |
| 职　　称 | 副教授 |
| 提交日期 | 2024 年 5 月 17 日 |

# 深圳技术大学本科毕业论文（设计）诚信声明

本人郑重声明：所呈交的毕业论文（设计），题目《面向协同办公的文件管理系统设计与实现》 是本人在指导教师的指导下，独立进行研究工作所取得的成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式注明。除此之外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。本人完全意识到本声明的法律结果。

毕业论文（设计）作者签名：陈杰森

日期：2024年5月17日

# 目录

# 面向协同办公的文件管理系统设计与实现

**【摘要】**协同办公作为信息时代崭新的工作模式，相对于传统的办公模式，在跨端协同办公和团队合作中具有优化信息管理机制、降低团队沟通成本、提高工作效率等优势。随着项目合作的开展，研究团队内往往会产生大量的文件材料，如研究文档、研究报告、办公文档（如 Word 文档、Excel 数据表格等），这些文件材料记录了项目研究的进程，记录了项目成员最新的工作成果，是项目的重要资产，需要妥善地存储管理和高效便捷地共享。

本论文从文件管理和文件版本同步两个角度出发，设计一套面向协同办公团队的文件管理系统。本文使用对象存储技术组织和管理文件数据，并根据不同的文件类型和业务场景，提供两种实现文件多版本同步的方案，分别是实时在线文档技术和文件版本同步算法。

实时在线文档技术能够实现多人同时编辑和查看文档，提高团队协作效率。在实时在线文档技术的支持下，团队成员可以即时看到彼此的修改，有助于快速反馈和调整，减少沟通成本。但是实时在线文档技术的同步效果会受网络延迟等因素的制约，导致编辑冲突、数据不一致的问题。对于大型文档或复杂格式，实时渲染和同步可能会消耗较多的计算资源，影响性能。

针对多实时在线文档技术存在的缺点，本论文设计文件版本同步算法——版本控制树算法，该算法基于对象存储技术管理文件元数据，使用树的数据结构组织文件的不同版本。版本控制树算法可以有效地解决离线文件版本冲突和同步问题，实现文件不同版本的追踪和记录，解决了系统的文件格式支持的问题，提高了团队协同办公的效率，丰富了系统的应用场景。

**【关键词】**协同办公；对象存储技术；实时在线文档；文件版本同步算法

# Design and Implementation of File Management System for Collaborative Office

【**Abstract**】As a new working mode in the information age, the collaborative office has the advantages of optimizing information management mechanisms and improving work efficiency in cross-end collaborative offices compared with traditional office modes. With the development of project cooperation, a large number of documents are often produced within the research team. These documents record the progress of project research and the latest work results of project members, which are important assets of the project and need to be properly stored, managed, and efficiently shared.

From the perspectives of file management and file version synchronization, this paper designs a file management system for collaborative office teams. This paper uses object storage technology to organize and manage file data, and according to different file types and scenarios, provides two schemes to achieve multi-version synchronization of files, which are real-time online document technology and file version synchronization algorithm.

Real-time online document technology can let multiple people edit and view documents at the same time. With the support of real-time online document technology, team members can see each other's modifications instantly. However, the synchronization effect of real-time online document technology is restricted by network delay and document format, resulting in edit conflicts and data inconsistency.

Given the shortcomings of multiple real-time online document technologies, this paper designs a file version synchronization algorithm "version control tree" algorithm. The algorithm organizes and manages different versions of files based on object storage technology and tree data structure, which can effectively solve the problems of offline file version conflicts and file format support, and realize the tracking and recording of different versions of files.

【**Key words**】Cooperative Office; Object Storage Technology; Real-time Online Documentation; File Version Synchronization Algorithm

# 1. 引言

## 1.1 研究背景及意义

随着信息技术的快速发展，团队办公模式正面临着深刻的变革。传统的办公方式，受限于时间、地点和设备，已经无法满足现代团队对高效、灵活办公的需求。在项目合作不断推进的过程中，研究团队通常会累积大量的文件资料，这些文件资料不仅详细记录了项目的研究进度，还记录了团队成员的最新工作成果，构成了项目不可或缺的重要资源。因此，团队的文件管理是一项重要的任务，它涉及到文件的存储、共享、更新和版本同步等多个环节。传统的文件管理方式往往依赖于本地存储、通讯软件、邮件等工具传递，这种方式在处理大量文件和多人协作时效率低下，且容易出错。因此需要一种更加高效、便捷的协同办公方式。

为了解决传统办公模式的不足，在互联网基础上衍生出来的文件管理系统得到了快速发展与广泛应用。团队成员可以通过不同的终端设备和操作系统，在不同的时间和地点进行协同办公。这种新型的办公模式不仅打破了时间和空间的限制，还大大提高了工作效率和团队协作能力[1]。

目前市场上的面向协同办公的文件管理系统还存在一些问题和挑战，如数据同步的准确性、管理效率低，以及不同设备和操作系统之间的兼容性差等。因此，本研究选题旨在设计并开发一个高效、稳定、便捷的文件管理系统，以解决上述问题，满足协同办公的需求。

## 1.2 国内外发展现状

随着云技术、大数据技术的快速发展以及跨端协同办公的需求增加，在互联网技术的基础上衍生出来的数据共享系统得到了快速发展与广泛应用[2]。

在互联网发展的初始阶段，互联网用户可以通过多种发送共享办公文件，如邮件、在线聊天工具等。随着团队成员数量和文件数量的增加，会出现文件管理混乱，沟通效率低的问题。

为了解决 Linux 内核开发中对版本控制的需求，Linus 用 C 语言开发了 Git，

随后 Git 成为软件开发中版本同步和多人协同的重要工具[3]。Git 允许团队成员协同工作，通过分布式版本控制有效追踪和管理代码的每一次修改，实现多版本同的步与合并。但 Git 存在学习难度高、容易误操作、跨平台性差的问题。

为了用户能够通过便捷的方式实现办公文件的传输和同步，国内外云服务厂商推出协同办公产品，如国外厂商 Google 的 Docs 及 Microsoft 的 Teams，国内厂商阿里的钉钉、腾讯的腾讯文档以及字节跳动的飞书等，这些产品集成了即时通讯、文件共享、任务管理等功能，致力于提升团队协作效率，使得团队成员可以跨越时空进行高效沟通。但在这些产品中，由于数据存储在云端，对网络环境依赖性强，网络不稳定、文档格式支持等会影响办公效率。

以上的方案一定程度上解决了协同办公中文件管理、权限管理、文件共享的需求，但仍存在以下问题[4-8]：

(1)系统兼容性问题：不同的企业可能使用不同的操作系统、数据库和应用软件，这导致协同办公文件管理系统在跨平台、跨系统运行时可能出现兼容性问题。这不仅影响了用户的使用体验，还可能导致数据丢失或损坏。

(2)维护成本高。对于中小型的研究团队，采用基于云存储的方案往往需要花费较多的人力物力投入到系统的维护和运维中，这会大大提高团队的研究成本，也往往会造成资源过剩。

(3)需要承担故障风险。基于云服务尝试提供的存储服务会对团队的数据造成一定的风险，如宕机、信息泄露等问题，都会对团队的数据安全带来威胁。

## 1.3 研究的主要内容

本论文旨在构建一个高效、安全、易用的文件管理系统，以满足团队协同在办公过程中，对各种格式文件的管理需求。本文将从需求分析、关键技术研究、系统设计与实现等方面详细介绍研究的主要内容。

本文将从协同办公的应用场景出发，调研协同办公中对文件管理的实际需求，包括文件的查看、编辑、共享、权限管理、版本控制等功能。同时，本文还将关注系统的性能、易用性等方面的要求，确保文件系统能够满足用户的期望。

系统设计是本研究的核心部分，主要包括系统架构设计、后端接口设计、数据库设计、文件同步模块的设计。本文采用分层架构的思想，将系统划分为多个相对独立的层次，核心业务按照服务对象分为多个模块，分别为身份管理模块、权限管理模块、文件管理模块和文件同步模块。这种设计方式不仅提高了系统的可扩展性和可维护性，有助于降低系统的复杂性。

数据库是文件系统的核心组成部分，本文将根据需求分析的结果，设计合理的数据库表结构、字段类型及约束条件，确保数据的完整性和安全性。同时兼顾数据的存储和访问效率，以提高系统的性能。

在协同办公场景中，团队成员需要对同一文件进行编辑和修改。为了解决文件多版本的同步问题，本文根据不同的文件类型和业务需求，研究并设计不同的解决方案，使系统能灵活地应用到不同的场景和办公文件，实现文件数据的共享和同步。

## 1.4 论文结构概述

本文的研究内容为协同办公场景下的文件管理系统设计与实现，分为五个章节进行论述，分别是引言、相关的理论知识及技术、系统需求分析与设计、文件同步管理系统的实现、总结与展望。

第一章：引言。首先论述系统的研究背景及意义，然后综述国内外面向协同办公场景的文件管理系统的发展现状，最后分析本文的研究主要内容。

第二章：相关的理论知识及技术。介绍系统设计中使用的相关技术，主要分为前端开发和后端开发的相应技术，为后续的系统实现提供参考和支持。

第三章：系统需求与分析。详细描述系统的需求背景和需求分析过程，包括用户需求、功能需求等方面。通过对需求的分析，确定系统设计的基本方向和功能模块，为系统的具体实现奠定基础。

第四章：文件同步管理系统的实现。介绍系统的整体架构设计、各功能模块的实现、关键技术实现细节等。通过具体的实现案例，读者可以了解系统设计理念的具体落地方式，以及系统在实际应用中的表现和效果。

第五章：总结与展望。总结本文针对在线文档、离线文件的文件同步方案的设计和实现，并展望未来的研究方向和发展趋势。

# 2. 理论知识及技术

## 2.1 前端开发技术

### 2.1.1 Vue.js 框架

Vue.js 框架是用于构建用户界面的前端开发框架，由尤雨溪在 2014 年创建。Vue.js 基于标准 HTML、CSS 和 JavaScript 构建，提供了一套声明式的、组件化的编程模型。Vue.js 框架现已成为一个强大的工具，使得开发用户界面变得高效且直观，广泛应用于生产环境中[9]。

Vue.js 的主要特点包括轻量级、双向数据绑定、组件化以及状态管理等。组件化是 Vue.js 的一个强大功能，组件可以扩展 HTML 元素，封装可重用的代码。这种组件化的开发方式提高了代码的可维护性，使得团队协作变得更加容易[10]。

Vue.js 的优点还包括易于学习和上手、灵活性高、性能优良以及支持 TypeScript 等。其模板语法简洁明了，可读性高，同时支持使用 TypeScript 开发，提高了代码的健壮性和可维护性。

### 2.1.2 Element Plus 组件库

Element Plus 是一个基于 Vue 3 的 UI 组件库，为开发者提供了丰富的组件和功能扩展，使开发者能够迅速构建出高质量的 Web 应用。

Element Plus 的设计理念是提供开箱即用的 UI 组件与扩展，保持了轻量级的体积，使开发者高效地构建应用程序。Element Plus 的设计风格简洁，遵循现代 UI 设计标准，使得 Element Plus 适用于各种规模的项目。此外，Element Plus 还提供了主题定制功能，开发者可以根据项目需求更换组件的样式，以满足不同的开发需求。

在实际开发中，开发者可以根据项目需求选择性地导入所需的组件，从而优化项目的体积和加载速度。此外，Element Plus 还支持多语言配置，开发者可以轻松地将应用适配到不同的语言环境。

### 2.1.3 HTML、CSS、JavaScript、Ajax 技术

随着互联网技术的迅速发展，网页设计与开发已成为信息技术领域的重要组成部分。其中，HTML、CSS、JavaScript 以及 Ajax 技术构成了现代网页开发重要技术。

HTML 全称为超文本标记语言，是构建网页内容的标准标记语言，构成网页基本框架[11]。CSS，即层叠样式表，是一种用来描述 HTML 或 XML（包括 SVG、XHTML 等格式）文档样式的语言。CSS 可以静态地修饰网页，配合各种脚本语言动态地对网页各元素进行格式化[12]。

JavaScript 是一种轻量级、解释型的编程语言。JavaScript 支持面向对象、函数式编程范式。在网页开发中，JavaScript 主要用于实现交互功能，如响应用户点击、动态修改页面内容等[13]。

Ajax 即异步 JavaScript 和 XML，是指一种创建交互式、快速动态网页应用的网页开发技术，无需重新加载整个网页的情况下，能够更新部分网页的技术，并且可以通过在后台与服务器进行少量数据交换，可以使网页实现异步更新[14]。

HTML、CSS、JavaScript 和 Ajax 技术是现代网页开发的强大工具，这四种技术相互协作共同构建了现代 Web 应用的丰富功能和良好用户体验。

## 2.2 Etherpad 技术

Etherpad 是一种开源的实时协作编辑平台，允许多个用户同时编辑同一个文档，其强大的实时同步功能使其成为协同办公环境中的理想选择[15]。

Etherpad 的优势在于实时协作能力。通过 WebSocket 技术，Etherpad 能够将每个用户的修改实时同步到所有其他参与者的界面上，确保所有用户始终看到最新的文档内容。不同用户的编辑内容会以不同的颜色高亮显示，便于区分和识别每个参与者的贡献，提升了团队协作的效率和透明度。

除了基本的实时编辑功能，Etherpad 还具有强大的版本控制功能。它会自动记录每一次修改，用户可以回顾历史版本，并在需要时恢复到任意一个之前的版本。

Etherpad 具备灵活和便捷的扩展性，通过各种插件，Etherpad 可以实现语法高亮、任务列表、实时聊天等功能，进一步增强其在不同应用场景中的适用性。此外， Etherpad 允许用户自行托管，确保了数据的隐私和安全。

## 2.3 后端开发技术

### 2.3.1 Go 语言与 Gin 框架

Go 语言是 Google 于 2009 年发布的一种强静态类型、编译型语言。其设计目标就是解决大规模并发处理和网络编程问题。Go 语言的特性包括内存安全，GC（垃圾回收），结构形态及 CSP-style 并发计算等，这使得 Go 语言在处理多处理器系统应用程序时具有显著优势[16]。

Gin 是一个用 Go 语言编写的 Web 框架，具备简洁、快速和安全的特性。作为一个轻量级的框架，Gin 具有高性能和低内存占用的特点，非常适合用于构建高性能的 Web 应用程序[17]。

在实际应用中，Gin 框架因其高效性能和易用性而得到广泛的应用。无论是构建小型 Web 应用还是大型分布式系统，Gin 框架都能发挥出良好的性能和开发效率。

### 2.3.2 MySQL 数据库

MySQL 数据库由瑞典 MySQL AB 公司开发，如今它已经成为了 Oracle 旗下的一款重要产品。经过数十年的发展，MySQL 已经成为了最流行的关系型数据库管理系统之一[18]。

MySQL 数据库采用了多种优化技术，如索引、缓存等，以提供高效的查询性能[19]。MySQL 数据库广泛应用于各种领域，如电子商务、社交网络、在线游戏等。它为企业和个人提供了稳定、可靠的数据存储和查询解决方案[20]。MySQL 数据库凭借其开源、跨平台、高性能和数据完整性保障等特点，在数据库管理领域占据了重要地位。

### 2.3.3 Docker 容器技术

随着云计算技术的快速发展，容器技术已成为当今软件开发和运维领域的重要创新[21]。与传统的虚拟机相比，容器共享宿主机的操作系统内核，因此具有更小的体积、更快的启动速度和更高的资源利用率。

Docker 是一个开源的容器化平台，为开发者提供了一个简单易用的容器运行时环境。Docker 通过将应用程序及其运行环境打包成一个 Docker 镜像，实现了应用程序的快速部署和管理。Docker 提供丰富的命令行工具和 API 接口，方便开发者进行容器的创建、部署和管理。

目前，Docker 已经成为云计算领域的重要技术之一，广泛应用于云原生应用的开发和部署[22]。Docker 也在推动容器技术的标准化和规范化方面做出了重要贡献。

### 2.3.4 WebSocket 技术

WebSocket 是一种网络通信协议，能够在单个 TCP 连接上进行全双工通信，即客户端和服务器可以同时发送和接收消息[23]。

WebSocket 的原理是基于 TCP 协议的，它首先通过 HTTP 协议进行握手，然后升级为一个持久的 WebSocket 连接[24]。这个连接在建立之后，客户端和服务器就可以通过这个连接自由地发送和接收数据，而不需要每次都建立新的连接。

WebSocket 技术具有广泛的应用场景，尤其是在需要实时交互和数据更新的场景中。如即时通讯、实时数据展示、多人游戏、实时协作、数据监控等。

# 3. 需求分析与数据库设计

## 3.1 需求分析

### 3.1.1 业务需求

在传统的办公环境中，常见的文件类型有 Word 和 PDF 等格式的文本文档，以及 Excel 等表格数据文档。随着业务领域的不断拓展和业务数据类型的多样化，现代办公环境已不仅仅局限于传统的文档类型。如今，视频、图片等新型文档也逐渐成为办公文件中不可或缺的一部分，系统需要支持对各类型文件的管理。

协同办公往往涉及多种操作系统和设备，系统需要具备良好的跨平台兼容性，确保用户在任何设备上都能访问和管理文件。协同办公的规模和业务需求会不断变化，系统需要具备良好的可扩展性，能够随着数据量的增加而灵活扩展。

为了团队能够高效地共享文件数据，文件管理系统需要实现使用户能够通过网络便捷地获取团队文件，并支持文件的查看、编辑、文件信息查询等基础功能，从而使文件在团队成员之间高效、安全地共享与传递。

为了保障团队文件数据的安全性，团队中不同成员对文件的访问权限应有所不同。系统需要建立完善的权限管理机制，确保文件只能被有对应权限的用户访问和修改，防止信息泄露和误操作。

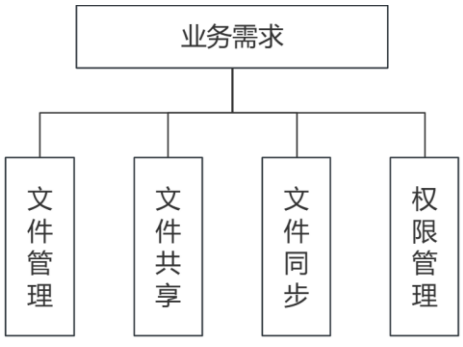


图 3-1 业务需求分析

针对团队协同办公的不同业务需求，本文设计了两种方案实现团队文件的同步，分别是实时在线文档技术和文件同步算法。

实时在线文档技术适用于常见格式的办公文件，如文本文档和表格数据文档，并且对于这类文档，团队成员需要进行频繁的修改，各成员需要实时地查看文档最新的内容，能够及时地反馈。

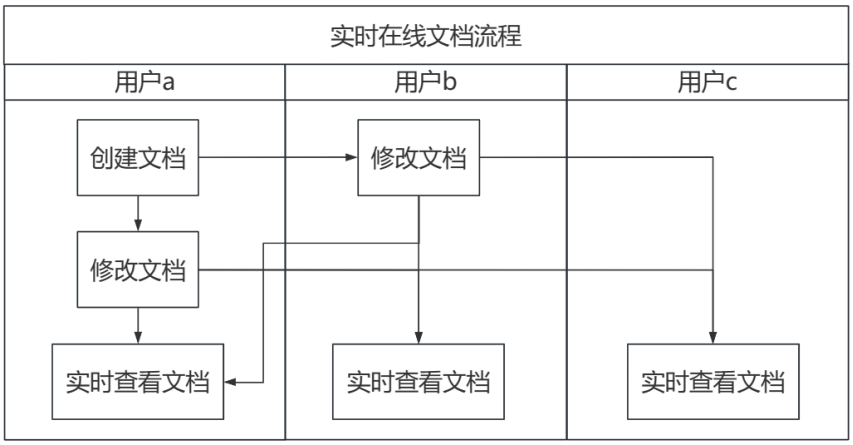


图 3-2 实时在线文档流程图

对于复杂的、大型的文件，由于文档格式复杂、网络延时等因素的制约，实时在线文档无法有效地实现文件的同步。因此，本文设计文件同步算法——版本控制树算法，解决离线文档的同步问题。

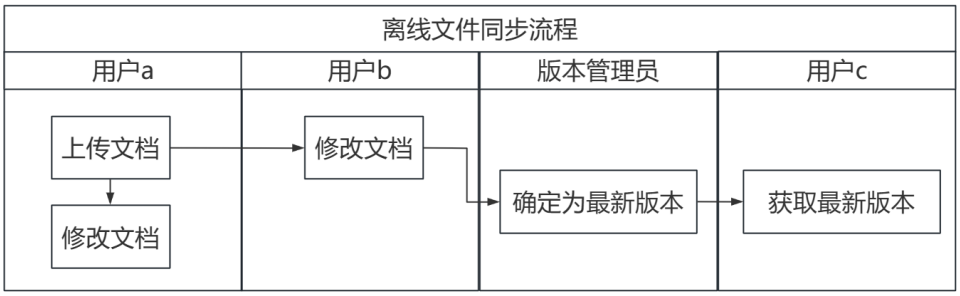


图 3-3 离线文件同步流程

## 3.1.2 功能需求

实时在线文档中，用户每次文件修改后系统需要实时展示最新的文件数据。同时系统需要记录文档的历史数据，包括文档内容、用户操作和操作时间等，用户可以轻松回滚到任意历史版本进行查看，必要时可以回退到特定的历史版本。此外，为了减少实时在线编辑的冲突，系统可以区分不同用户的身份，以直观的方式展示用操作用户。为了提高团队的沟通效率，系统应该支持实时聊天的功能，让办公成员实时在线沟通协商，提升协同工作的效率。

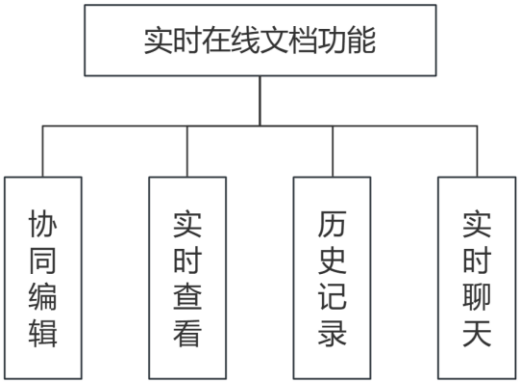


图 3-4 实时在线文档功能需求

为了支持丰富类型文件的管理和同步，以及减少网络延时、锁机制等因素对实时在线文档的制约，系统需要具备离线文件共享的功能，包括文件的传输、共享。同时，系统针对离线文件设计文件同步算法，实现离线文件的版本同步，包括解决版本冲突问题和追踪历史版本。

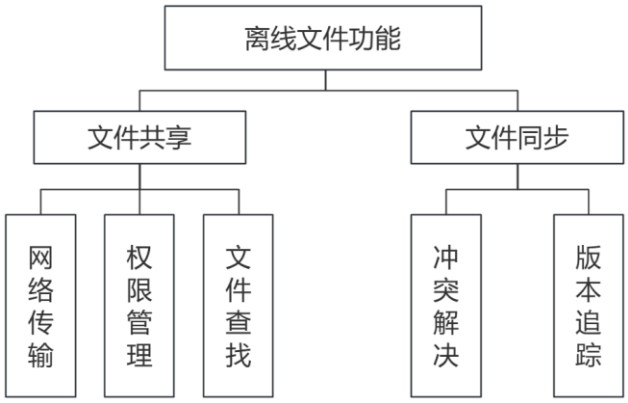


图 3-5 离线文件功能需求

此外，系统需要提供灵活的权限设置功能，允许管理员设置不同的用户组，并为不同用户组分配不同的文件访问和修改权限。

### 3.1.3 功能框架图

本系统采用了分层架构的设计思想，从上到下依次为终端设备层、展示层、数据传输层、后端接口层以及存储层。这种架构使得系统的各个部分可以相对独立地进行开发和维护，提高了系统的可伸缩性和可维护性。

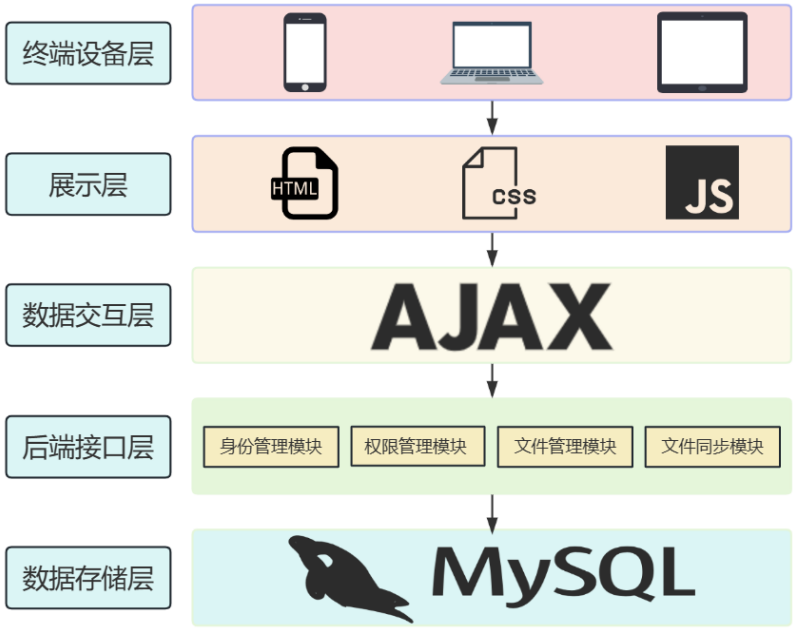


图 3-6 功能框架图

终端设备层主要负责与用户进行交互。前端设备可以是手机、平板电脑和桌面电脑，保证跨平台的兼容性。

展示层使用 HTML 和 CSS 技术构建用户界面的外观和布局，使用 JavaScript 构建页面的动态效果和交互逻辑。该层为用户提供直观、易用的界面，方便用户与系统进行交互。

数据交互层使用 Ajax 技术实现数据的异步传输，与后端接口进行数据的请求和响应。提高了用户体验，使得页面加载更加流畅和快速。

后端接口层包含多个功能模块，分别是身份管理模块、权限管理模块、文件管理模块和文件同步模块。这些模块负责处理前端的请求，执行相应的业务逻辑，并将结果返回给前端。其中，身份管理模块负责用户的身份认证；权限管理模块负责管理用户的对团队和文件的权限；文件管理模块负责文件的上传、下载和删除等操作；文件同步模块则负责实现文件在多个设备之间的同步。

存储层采用 MySQL 数据库作为数据的存储介质。该层负责存储系统的各类数据，包括团队信息、用户信息、文件信息等。通过优化数据库设计和查询，可以确保数据的快速读取和高效存储。

### 3.1.4 角色用例图

本文将共享文件分为在线文档和离线文件。在线文档通过实时在线文档进行协同编辑和信息共享，离线文件对象存储技术和文件同步算法进行版本进行管理和同步。

(1) 在线文档：常用的办公文档格式，如文本文档和表格数据文档，这类文档的格式简单，能够通过网页展示，用户能够通过网页对文档进行修改，并且用户对这些文档的修改频率较高，需要实时获取其他用户的最新修改内容，使文件数据能够实时地全局共享。

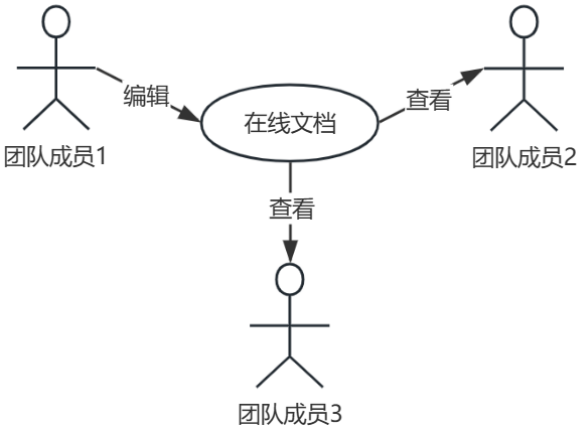


图 3-7 角色用例图 1

(2)离线文件：文件的格式丰富，如视频、图片等。系统支持对该类文档的管理和同步，用户可以并行地更新文件，用户之间的更新行为相互独立，同步过程不受网络延时等因素制约。

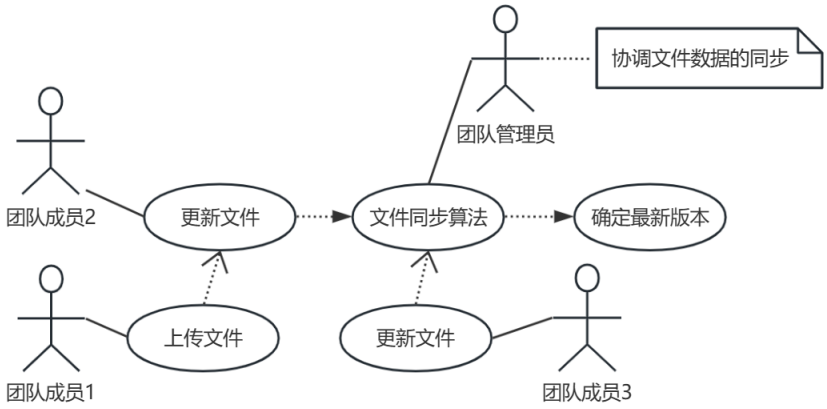


图 3-8 角色用例图 2

## 3.2 数据库设计

### 3.2.1 E-R 图设计

数据库系统需要完成团队信息、用户数据、文件元数据和文件版本信息的管理。系统涉及的对象实体有团队、用户、文件和文件版本，各对象实体之间存在较为复杂的关联关系，需要设计合适的数据表进行管理。同时，系统需要高效地查询对象实体的信息，需要设计数据索引等提高系统的查询效率。

E-R 图，即实体-联系图(Entity Relationship Diagram)，是数据库设计中的重要工具[25]。它通过图形化方式直观展示数据模型，主要由实体、属性和关系三个基本成分构成。E-R 图有助于设计者清晰地理解业务需求，简化了复杂数据结构的视觉表示。

系统的对象实体有团队、用户、文件元数据和文件版本信息，具体的 E-R 图设计如图 3-9 所示。

实体包含较多的属性，图 3-9 展示了实体部分关键属性。团队和用户实体之间为多对多的关系，团队可容纳多名成员，团队管理员通过邀请的方式将用户加入团队中，一个用户可以加入多个团队中。

用户加入团队后通过上传离线文件或者创建在线文档的方式添加团队的协同办公文件，后端系统将文件数据存入云服务器，并在数据库保存文件的元数据记录，文件的元数据包括文件名、文件哈希值、文件创建者等。

文件数据创建后，团队成员可以通过下载离线文档到本地，或者通过网页查看在线文档的方式对文件修改。每次修改后的文件作为原文件的一个新版本，后端系统保存新版本的文件数据到云服务器，并在数据库创建文件版本记录，文件版本记录包括新版本的创建时间、创建用户、版本状态等基本信息。

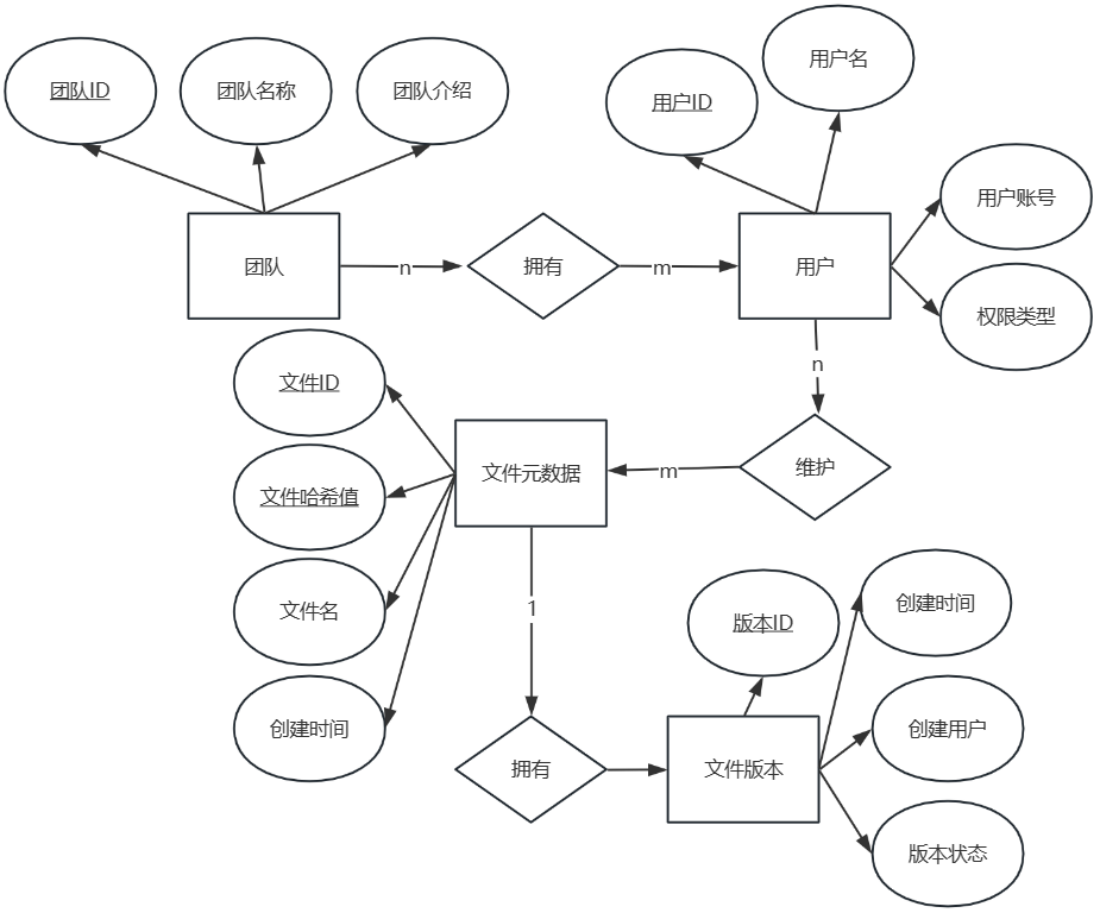


图 3-9 E-R 图

## 3.2.2 数据库表设计

（1） 团队信息数据表 team_detail，如表 3-1 所示。

表 3-1 团队信息表结构

| 字段名 | 数据类型 | 注释 |
|---|---|---|
| id | bigint(20) | 团队主键 |
| name | varchar(50) | 团队名称 |
| create_time | datetime | 团队的创建时间 |
| creator_id | bigtint(20) | 创建用户的主键 |
| description | varchar(150) | 团队的描述信息 |
| avatar_url | varchar(256) | 团队头像链接 |

为团队名称 name 列创建索引，提高用户查找目标团队的效率。当用户创建团队后，会被自动记为团队的管理员，团队的创建时间 create_time 为团队信息保存到数据库的时间。

（2） 用户信息数据表 user_detail，如表 3-2 所示。

表 3-2 用户信息数据表结构

| 字段名 | 数据类型 | 注释 |
|---|---|---|
| id | bigint(20) | 用户主键 |
| name | varchar(50) | 用户名称 |
| account | varchar(50) | 用户账号 |
| password | varchar(50) | 用户密码 |
| create_time | datetime | 用户注册时间 |
| avatar_url | varchar(256) | 用户头像链接 |

对用户名称 name 列和用户账号 account 列创建索引，团队管理员在邀请用户加入团队时，可以提高搜索用户的效率。

（3） 团队-用户关联表 team-user，如表 3-3 所示。

表 3-3 团队-用户关联表结构

| 字段名 | 数据类型 | 注释 |
|---|---|---|
| team_id | bigint(20) | 团队主键 |
| user_id | bigint (20) | 用户主键 |
| admin_type | enum | 用户账号 |

该表用于记录团队和用户的所属关系,有序对$< team_id,user_id >$表示主键为 user_id 的用户属于主键为 team_id 的团队，其中用户在团队中的权限为 admin_type，admin_type 的值可取“T”，“V”，“W”，“R”，分别对应团队管理员、版本管理组、读写组和只读组。

（4） 文件元数据表 file_metadata，如表 3-4 所示

表 3-4 文件元数据表结构

| 字段名 | 数据类型 | 注释 |
|---|---|---|
| id | bigint(20) | 元数据主键 |
| file_name | varchar(50) | 文件名 |
| md5 | varchar(32) | 文件 MD5 值 |
| team_id | bigint(20) | 所属团队的主键 |
| creator_id | bigint (20) | 创建用户主键 |
| create_time | datetime | 创建时间 |
| file_size | bigint(32) | 文件的大小(MB) |
| file_path | varchar(50) | 文件路径 |
| file_status | enum | 文件的状态 |

为文件名 file_name、文件 MD5 值 md5、文件路径 file_path、文件所属团队的主键 team_id 创建索引，后端的业务对该 4 列的查询操作较多，有利于提高查询效率。后端收到前端传送的文件数据后，会自动计算相关的字段字，然后保存到数据库。文件的状态 file_status 可取“L”、“U”,分别表示“最新(Latest)”，即文件处于最新的状态；“已更新(Updated)”，即该文件被更新到其它版本。

（5） 文件版本信息表 file_version，如表 3-5 所示。

表 3-5 文件版本数据表结构

| 字段名 | 数据类型 | 注释 |
|---|---|---|
| id | bigint(20) | 表主键 |
| file_id | bigint(20) | 对应文件的主键 |
| md5 | varchar(32) | 版本的 MD5 值 |
| updater_id | bigint (20) | 更新用户主键 |
| update_time | datetime | 更新时间 |
| file_size | bigint(20) | 文件的大小(MB) |
| file_path | varchar(50) | 文件路径 |

# 4. 文件同步管理系统的实现

## 4.1 身份管理

用户在前端页面上输入账号和密码，当用户点击登录按钮时，敏感信息便通过 https 协议安全发送到后端服务器[26]。后端服务器接收到用户的账号数据后，会查询数据库获取对应账号密码信息进行检验。

账号密码的合法性验证通过后，系统会生成一个代表用户身份的 JWT(Json Web Token)。JWT 作为用户身份验证的凭证颁发给前端，前端会将 token 存储到客户端本地[27]。在 JWT 的有效期内，每当前端发起请求时，都会携带这个 JWT，以验证请求者的身份。

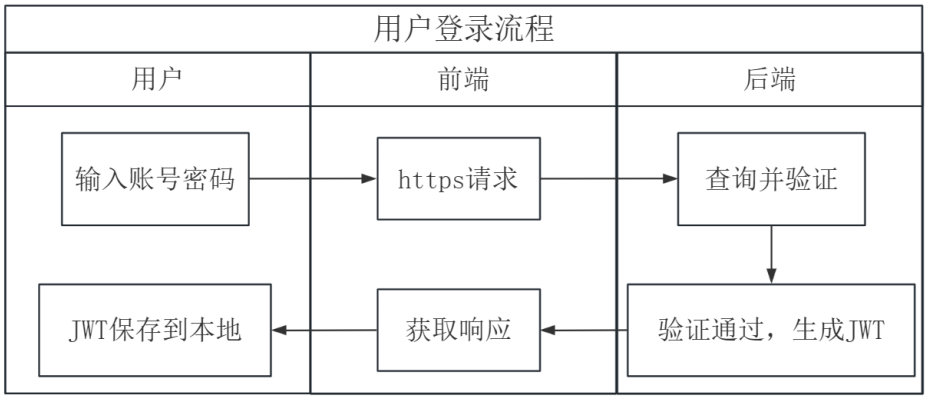


图 4-1 用户登录流程

基于 JWT 机制，用户在完成初始登录后，就可以在一定时间内无需再次输入账号密码进行身份验证。此外，JWT 机制还增强了系统的可扩展性和灵活性。如图 4-2 所示，后端可以根据需要在负载部分（Payload）设置 JWT 的有效期、添加用户信息和自定义数据等，便于后端确认用户身份，同时减少用户操作，提升用户体验。

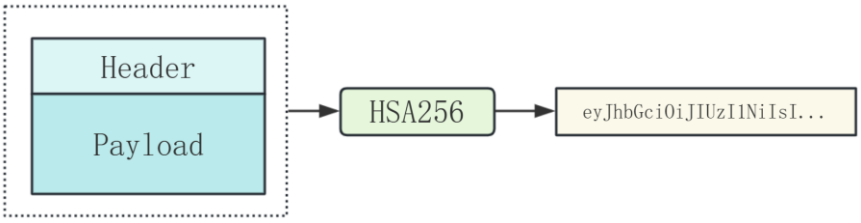


图 4-2 JWT 机制

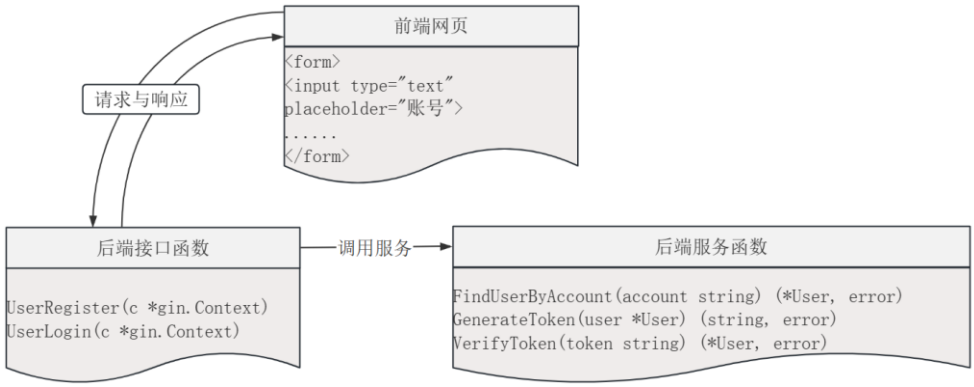


图 4-3 用户的注册和登录实现

## 4.2 权限管理

根据用户管理的对象，将管理权限分为团队管理员和文件管理员。其中文件管理员分成 3 个组，分别是版本管理组、读写组和只读组。各类管理员拥有的团队权限如图 4-4。

| 权限身份分类及其权限 | | | |
|---|---|---|---|
| 团队管理员 | 文件管理员 | | |
| | 版本管理组 | 读写组 | 只读组 |
| 1. 添加/删除团队成员<br>2. 管理用户权限<br>3. 查看文件<br>4. 编辑文件<br>5. 创建文件<br>6. 确定文件版本 | 1. 查看文件<br>2. 编辑文件<br>3. 创建文件<br>4. 确定文件版本 | 1. 查看文件<br>2. 编辑文件<br>3. 创建文件 | 1. 查看文件 |

图 4-4 权限身份分类及其权限

用户的权限采用分层设计理念，通过位图（bitmap）结合位运算来定义和判断用户的权限范围。系统内共划分为四类权限组，每组权限由一个四位二进制数标识，具体如下：团队管理员（0b1111）、版本管理组（0b0111）、读写组（0b0011）以及只读组（0b0001）。为了判断某一用户权限（以变量 admin_type 表示）是否包含特定权限组的权限，系统采用二进制位与（&）运算进行验证。如判断用户权限 admin_type 是否属于读写组，运算条件如下

$$if(admin\_type \ \& \ 0b0011 \ == \ 0b0001)$$

## 4.3 文件管理

### 4.3.1 对象存储技术

对象存储技术是一种用于存储大量非结构数据的解决方案，它以扁平的命名空间来存储数据，并将数据和元数据一起存储在一个对象中。

在系统中，文件被视为一个对象实体，每个对象都包含数据本身、文件元数据和自定义元数据。系统的文件对象结构如图 4-5 所示。


图 4-5 文件对象结构

系统用文件数据的 MD5 值作为文件的唯一标识。MD5 是一种广泛使用的加密哈希函数，它可以从任意长度的数据中生成一个 128 个字符的哈希值。系统通过直接对比文件的 MD5 值是否相等，从而判断文件的数据是否相同，达到文件数据对比的目的。MD5 的优点在于其高效性和唯一性，它能够迅速地为大量数据生成独特的哈希值[28]。此外，MD5 算法具有良好的抗碰撞性，进一步增强数据验证的准确性。

### 4.3.2 实时在线文档

实时在线文档允许多个用户在同一文档上进行实时的协同编辑，团队成员能够实时查看文档的创作过程，实现信息的即时共享和反馈。

为了提升协同办公和管理的效率，实时在线文档系统需要实现：编辑用户的身份识别，以确保不同用户的编辑操作可以被明确区分；实时聊天功能，允许团队成员在编辑过程中进行即时沟通；版本记录和历史版本回溯功能，使得团队能够追踪文档的修改历史，并在必要时恢复到先前的版本。

在多用户协作编辑的场景中，为了实现编辑过程中用户的身份可视化和区分，系统为不同用户的编辑文本进行不同颜色的编码。这种颜色编码机制有助于用户快速识别和定位自己的编辑内容，便于跟踪和理解其他用户所做的更改。如图 4-6 所示，当 user1 和 user2 共同参与文档编辑时，通过采用不同的前景色来区分各自的编辑区域。这种策略有效地降低了编辑过程中的冲突，提高了协作的效率和准确性。

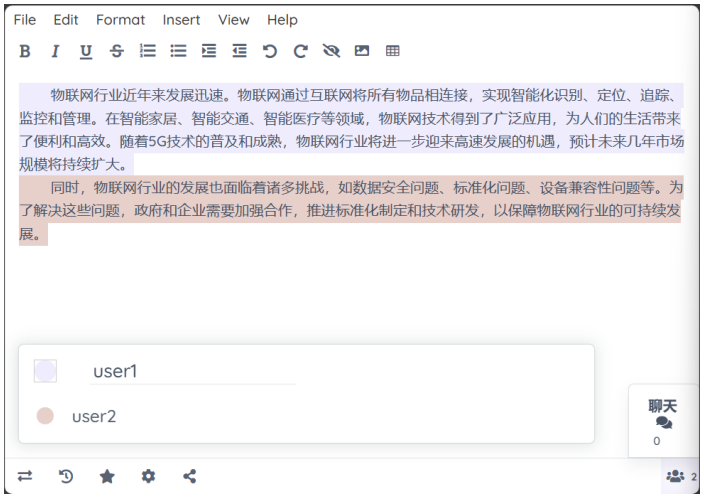


图 4-6 区分不同的用户

为了提高在线编辑的效率，系统集成在线聊天功能，允许用户在编辑文档的同时实时交流，降低了沟通的时间和成本，从而促进了用户之间的即时协作和信息共享，如图 4-7 所示。

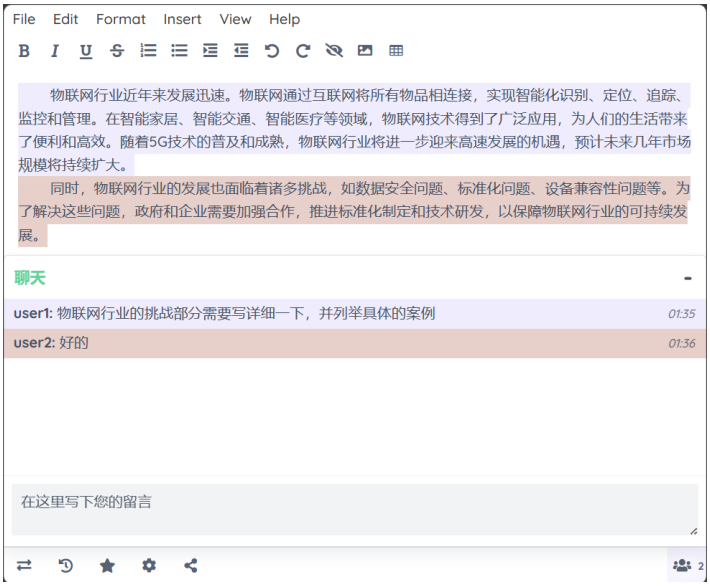


图 4-7 文档携带的实时聊天室

在协同办公过程中，系统会自动记录文档的各个版本，包括文档内容、保存时间、编辑的用户等，允许用户查看文档的历史版本，可以选择恢复到之前的特定版本。

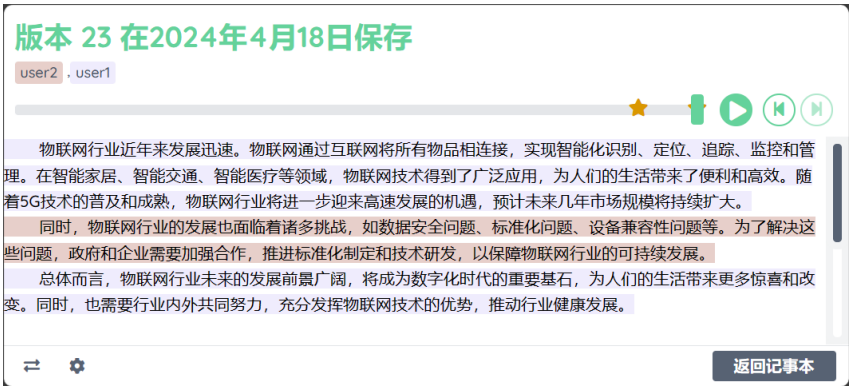


图 4-8 系统记录文件的历史版本

### 4.3.3 离线文件的上传与下载

对于离线文件，用户可以通过网络传输，下载团队的文件数据到本地，对文件编辑后上传到系统。

前端利用 Element-Plus 组件库中的“el-upload”组件，封装文件数据，并将数据传输到服务器。“el-upload”组件提供了一系列详尽的参数配置选项，使开发者能够高效地定制和实现符合特定业务需求的功能。

后端系统接收到前端发送的文件后，将文件数据解析为 Go 语言的字节切片([]byte)类型数据，并存到服务器磁盘，然后会计算文件数据表的相关字段，保存记录到数据库。

当用户请求文件下载时，前端发送包含文件标识的请求到后端。后端接收到请求后，根据文件标识找到对应的文件，设置正确的响应头部信息，包括文件类型、文件名等，并将文件内容作为响应体返回给前端。前端收到响应后触发浏览器的文件下载功能，从而完成文件下载过程。

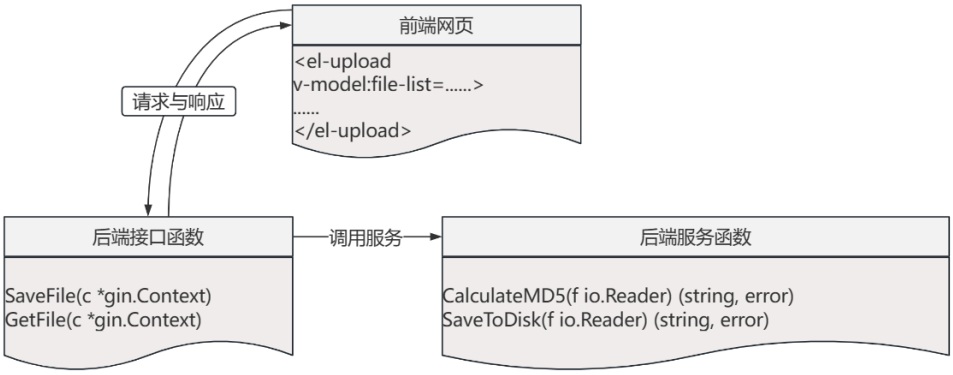


图 4-9 文件的上传和下载

### 4.3.4 版本控制树算法

网络延时会导致在线文档同步效果不佳，同时在线文档支持的文档格式有限，因此，在线文档不能被广泛地应用到丰富的业务场景之中。此外，当协同办公的人数增加时，版本冲突的概率将会大大增加。

团队成员可以下载文件，通过对本地文件进行更新有效地避免文件格式支持、同步过程中的网络延时等问题，团队成员在本地更新文件后，将文件上传到系统，向团队同步自己更新的版本文件数据。

基于这种方式，团队多人协同办公过程中仍会出现版本冲突的问题。对于离线文件，文件修改后得到新文件视为原文件的新版本。一种常见的版本冲突场景如图 4-10 所示，在 $t_0$时刻，成员 A 下载了团队的文件 F，初始版本为 v0，记为 F(v0)。成员 A 在本地对文件 F 进行修改，对应的版本为 v1，记为 F(v1)。在 $t_1$时刻，成员 B 下载了文件 F，并在本地对其进行修改，得到 F(v2)。在 $t_4$时刻，成员 A 上传了自己更新的文件到系统，更新了团队文件 F 的版本。在 $t_5$时刻，成员 B 更新系统中的文件 F 为自己修改的文件，此时会导致成员 A 的修改丢失，即成员 A 和成员 B 的版本冲突。

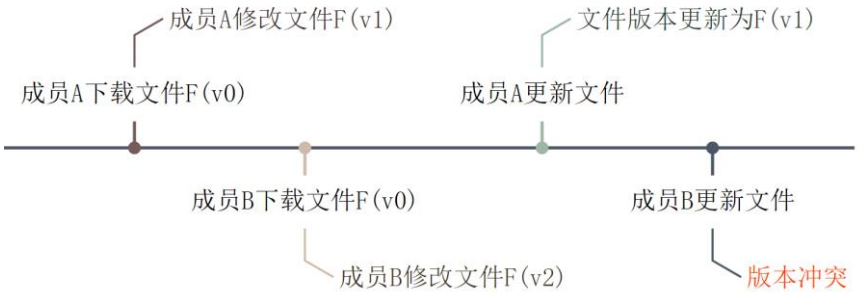


图 4-10 版本冲突的案例

为了解决上述的离线文件版本冲突的问题，本文设计了一种文件同步算法——版本控制树算法。在协同办公过程中，一个文件会产生许多的版本，算法基于树的数据结构来组织和管理文件的多个版本。树的节点存储了与文件版本信息相关的数据，这些数据包括文件的 MD5 哈希值、更新该文件版本的用户主键以及版本的更新时间等。

在这个树状结构中，原始文件被表示为树的根节点。每当文件被更新并上传到系统时，新版本的文件将被作为原始文件的一个孩子节点，并添加到树中。通过这种方式，文件的每一次更新都会在树中产生一个新的节点，从而形成了一个多叉树的结构。这种结构清晰地记录了文件的各个版本，展示了文件版本的演变过程，有效地避免不同用户的版本冲突，同时提供了一种直观且高效的方式来管理和查询文件的各个版本。

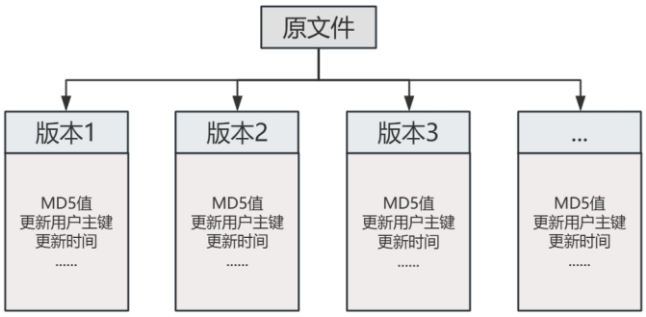


图 4-11 树的结构

对应成员更新的版本，其他成员也可以进行更新，得到新的版本。如图 4-12 所示。

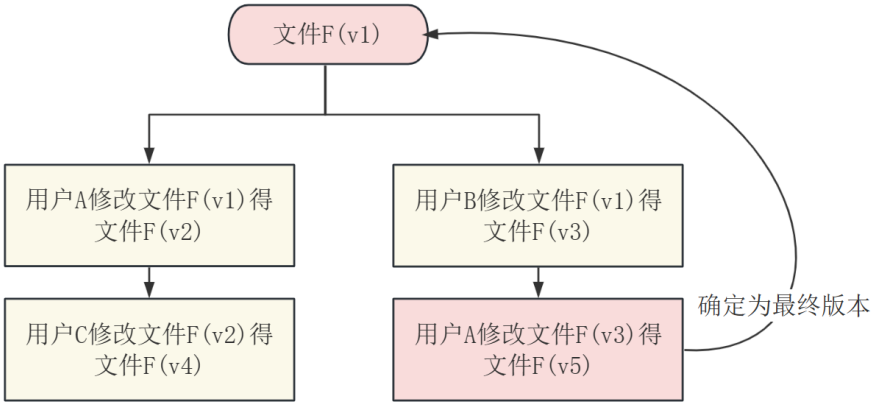


图 4-12 版本控制树

对于团队文件 F，初始版本为 v1，成员 A 和成员 B 下载后，成员 A 和成员 B 可以在本地进行修改，修改的行为相互独立，不会相互影响。成员 A 和成员 B 完成修改并上传到系统后，两个版本都会成为文件 F(v1)的孩子结点，共同存在于系统中，避免了图 4-10 出现的版本冲突问题。对于文件的新版本，其他成员可以对其进一步更新。最后，团队的版本管理员可以查看、对比文件的各个版本，选择最合适的版本作为文件的最终版本，从而实现原文件版本的全局更新。如图 4-12，团队的版本管理员通过对比各个版本的内容，最终选择了文件 F(v5)作为原文件的最终版本，此后团队成员可以下载获取最新的版本数据 F(v5)。

版本控制树算法基于树的结构，允许文件同时存在多个版本，避免的文件多版本的冲突。此外，各成员之间可以并行的更新，成员之间的更新相互独立，提高了协同办公的效率。同时成员间的协作不受网络延时的制约，本地查看和编辑的方式提高了系统的灵活度和便捷度。基于对象存储技术的文件管理，使得系统适用于更复杂的业务场景。

### 4.3.5 算法的实现

在数据库中，文件版本通过其主键 id 进行唯一标识。每个文件版本都与一个父节点相关联，即版本对应的原始文件。在数据库中通过数据表关联规则[29]实现，文件元数据表 file_metadata 中的主键 file_id 表示树的父结点信息，文件版本信息表 file_version 中的有序对$< id, file_id >$表示主键为 id 的版本，其对应的原文件为文件元数据表 file_metadata 中主键为 file_id 的文件。

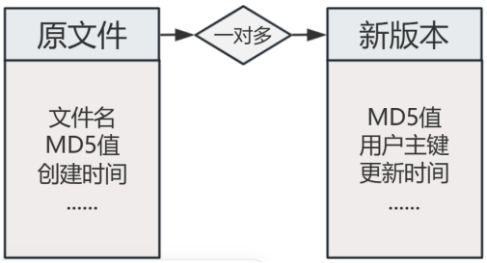


图 4-13 原文件和版本的关系

系统还可以通过 team_id 字段来检索特定团队关联的所有文件版本的能力。这使得用户能够有效地追踪和管理团队内文件的历史版本，确保对文件版本控制的灵活性和可追溯性。

用户进入团队页面后，系统展示团队文件所有的离线文档和在线文档。

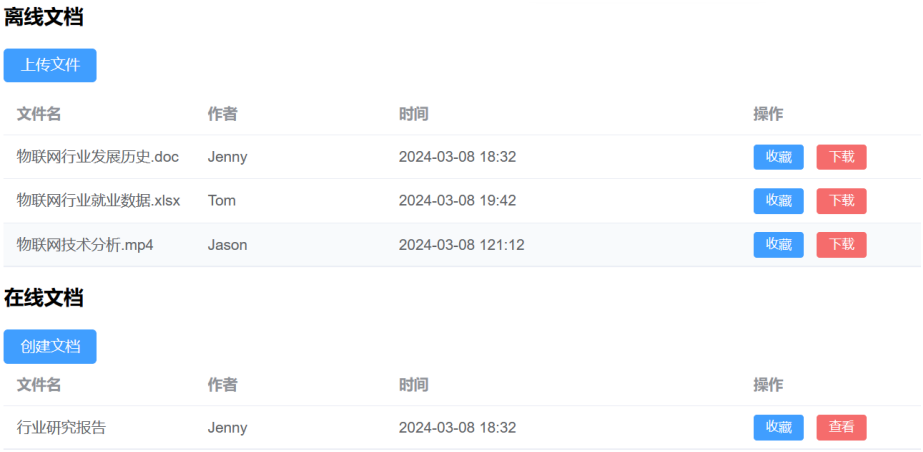


图 4-14 团队文件页面

其中的离线文档为用户通过“上传文件”的方式创建，上传后的文件在版本控制树算法中为树的父结点。用户通过点击“收藏”，将文件添加到“我的文件”模块。


图 4-15 用户收藏操作

“我的文件”模块展示了用户“收藏”的文件。模块通过不同的颜色区分文件的状态。

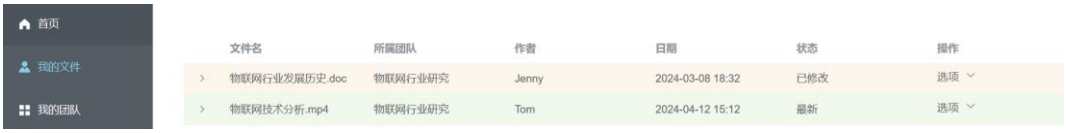


图 4-16 我的文件模块

对每一个原文件，用户可以选择更新和查看。“更新”即用户可以在对应的文件下上传自己更新的版本，即在原文件下添加孩子结点。“查看”操作会将文件下载到本地。通过本地查看的方式，有效解决网页文档格式不支持的问题。


图 4-17 文件的操作选项

“我的文件”模块展示了文件的更新记录，即原文件的版本列表。


图 4-18 文件的更新记录列表

对于原文件的每个版本，用户可以选择“查看”和“设置为最新版本”操作。“设置为最新版本”即将当前的版本设置为原文件的最新版本，此后团队文件列表中显示的该文件为最新版本的文件，从而实现文件的版本同步。

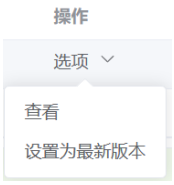


图 4-19 文件版本的操作选项

## 4.4 开发工具与项目部署

本文选择 Goland、Visual Studio Code 和 API Post 三款软件作为系统开发的工具。

Goland 是 JetBrains 公司的 Go 语言集成开发环境，它提供了丰富的 Go 语言支持，包括智能代码补全、重构工具等，提高了开发效率和代码质量。

Visual Studio Code 是一款轻量级源代码编辑器，支持多种编程语言。其丰富的插件和高度可定制性，使得能够根据不同项目的需求进行灵活配置。

API Post 是一款 API 测试工具。在系统的开发过程中，需要接口测试，API Post 的直观界面和丰富的功能简化了测试流程，帮助快速发现和解决问题。

本项目采用 Docker 容器技术，实现了将系统部署至云服务器。用户可以通过浏览器输入 URL 地址，登录并使用系统提供的各项服务。

在构建 Docker 镜像时，采用了多阶段构建（multi-stage build）策略，通过分离构建和运行环境减少最终镜像的体积。Dockerfile 中的配置如下所示：

```
1.  FROM golang:1.19 AS gin-builder
2.  MAINTAINER Jason<2282727499@qq.com>
3.  WORKDIR /gin-server
4.  COPY go.mod go.sum ./
5.  ENV GO111MODULE on
6.  ENV GOPROXY https://goproxy.cn,direct
7.  ENV CGO_ENABLED 0
8.  ENV GOOS linux
9.  RUN go mod download
10. COPY . .
11. RUN go build -o ./gin-docker
12. FROM alpine:latest
13. WORKDIR /gin-server
14. COPY --from=gin-builder /gin-server/Config ./Config
15. COPY --from=gin-builder /gin-server/gin-docker ./gin-docker
16. EXPOSE 8080
17. ENTRYPOINT ["./gin-docker"]
```

通过这种多阶段构建方法，提高了部署的效率，优化了资源使用，确保了系统的轻量化和快速启动。

# 5. 总结与展望

随着企业协作需求的日益增长，协同办公已成为现代工作模式的重要组成部分。在本设计中，针对在线文档和离线文档分别提出了不同的同步解决方案，旨在提升团队协作的效率和文档管理的便捷性。

对于在线文档，通过实时在线文档技术，实现了多人同时编辑、实时更新的功能，消除了传统文档编辑中因版本不同步而导致的工作重复或冲突。此外，系统还支持历史版本记录和回溯、在线聊天的功能，提高了团队协作的灵活性和响应速度。

对于离线文档，本文设计了一种文件同步算法——版本控制树算法，该算法有效地解决了文件版本冲突的问题，实现了文件多版本的同步。该算法使得团队成员之间的更新行为不受影响，相互独立。算法支持丰富的文档格式，使得系统可以应用于更丰富复杂的业务场景。

面向协同办公的文件管理系统的设计和实现，解决了当前团队协作中的文件管理和同步问题，还为未来的办公模式提供了新的可能。

随着全球化和移动办公趋势的加速，面向协同办公的文件管理系统能连接不同地域和时区的团队成员，推动跨地域、跨时区协作的效率和效果。期待这一系统能在更多领域得到广泛应用，为现代办公模式带来解决方案。

# 参考文献

[1] Kevin L M. Introduction to the electronic symposium on computer-supported cooperative work[J].ACM Computing Surveys (CSUR),1999,31(2):105-115.

[2] 谷金宇,刘志春,高峰,等.基于网、云、端架构的自主创新云协同办公平台[J].电子技术应用,2023,49(4):133-136.

[3] Spinellis D. Git[J].IEEE Software,2012,29(3):100-101.

[4] Nasri N F, Habali A H M, Adam M H M, et al. Google Docs: Students' Perceptions as Online Collaborative Tool in Learning Writing Skills[J].International Journal of Academic Research in Progressive Education and Development,2022,11(3):690–705.

[5] Frey K T, Bloch S B. Using Microsoft Teams to Facilitate Asynchronous Online Foc us Groups[J].International Journal of Qualitative Methods,2023(22):1-15.

[6] 张华,翟新军,胥勇.基于钉钉构建远程运维管理一体化云平台的创新研究应用[J].产业创新研究,2023(12):127-129.

[7] 徐文昭.运用腾讯文档与 VBA 实现图片信息的收集整理[J].内蒙古科技与经济,2022(3):91-92.

[8] 杨嘉欣.基于飞书的办公交互系统的设计与实现[D].杭州:浙江大学,2021.

[9] 张耀春,黄轶,王静,等.Vue.js 权威指南[M].北京:电子工业出版社,2016.

[10] 李晓薇.vue.js 前端应用技术分析[J].网络安全技术与应用,2022(4):44-45.

[11] 陆凌牛.HTML5 与 CSS3 权威指南[M].北京:机械工业出版社,2011.

[12] 张鑫旭.CSS 世界[M].北京:人民邮电出版社,2017.

[13] 王艳梅,丁玲.JavaScript 技术在 Web 网页中实践分析[J].自动化应用,2022(12):62-65.

[14] 徐永春,蔡龙飞.AJAX 在前端开发的设计及应用[J].工程技术研究,2023,8(15):170-171+175.

[15] Tapia F, Mora M, Fuertes W, et al. A Container Orchestration Development that Optimizes the Etherpad Collaborative Editing Tool through a Novel Management System. Electronics[J]. 2020,9(5):828.

[16] 黄靖钧.Go 语言编程入门与实战技巧[M].北京:电子工业出版社,2018.

[17] 周明辉.基于 Golang+Gin 的技术运维系统设计与实现[J].现代电视技术,2022(10):134-137.

[18] 李辉. 数据库技术与应用[M]. 北京:人民邮电出版社, 2018.

[19] 张士军, 陆海伦. 索引在 MySQL 查询优化中的应用[J]. 计算机与数字工程, 2007,35(1):37-39+2.

[20] 熊群毓. 大数据时代 MySQL 数据库的应用分析[J]. 信息与电脑(理论版), 2023(14): 209-212.

[21] 张玉冰. 计算机软件开发中 Docker 技术应用探讨[J]. 电脑编程技巧与维护, 2023(12): 16-18+22.

[22] 张荻, 孙蓉. 计算机软件开发中 Docker 技术应用分析[J]. 产业创新研究, 2023(12):145-147.

[23] 王彬. 基于 WebSocket 协议的网页即时通信系统研究与实现[J]. 无线互联科技, 2023(24):4-7.

[24] 周义盼. 基于 WebSocket 的协同编辑系统的设计与实现[D]. 哈尔滨:哈尔滨工业大学, 2020.

[25] 李志. 论 E-R 图在数据库建模过程中的重要性[J]. 信息记录材料, 2020,21(6):143-145.

[26] 许俊龙, 徐爱华, 王松. SSL 数字证书在 Web 安全中的应用探析[J]. 网络安全技术与应用, 2023(10):10-13.

[27] 项武铭, 鲍亮, 俞少华. 基于 JWT 的 RESTful API 角色权限验证方案设计[J]. 现代计算机(专业版), 2018(34):82-85.

[28] 徐蕊. MD5 加密算法的研究与应用[J]. 中国新通信 2015,17(21):72.

[29] 朱彦霞. 多关系关联规则及其在 HRM 中的应用[D]. 天津:河南工业大学, 2011.

# 致谢

时光荏苒，岁月如梭。转眼间，四年的本科学习已经接近尾声。在此，我怀着感激的心情，书写这篇致谢，以表达在我本科学习期间给予我无私帮助和支持的人们的深深谢意。

首先，衷心感谢我的导师史诗洁副教授，她的严谨治学、深厚学识和无私奉献，是我求学路上的明灯。从论文选题、实验设计到论文撰写，导师都给予了我悉心的指导和帮助。每当我遇到困难和问题时，导师总是耐心指导，并给予我鼓励和支持，使我能够坚定信心，勇往直前。

接下来，感谢学院的各位领导和老师，在我本科期间提供了良好的学术氛围和实验条件。各位老师的严谨教风和敬业精神，使我本科学习受益匪浅。

感谢我的家人，他们始终是我坚强的后盾。在我求学过程中，他们给予了我无尽的关爱和支持。每当我遇到困难和压力时，家人总是我最温暖的港湾，他们的理解和鼓励成为我前进的动力。

在本科的学习生活中，我深刻体会到了学习和研究的艰辛与乐趣。这段经历不仅丰富我的知识体系，还锻炼了我的意志品质。在未来的道路上，我将继续努力，不断探索，为实现自己的人生价值而奋斗。

最后，再次感谢所有关心、支持和帮助过我的人！愿我们都能在未来的道路上继续前行，创造更加美好的人生。