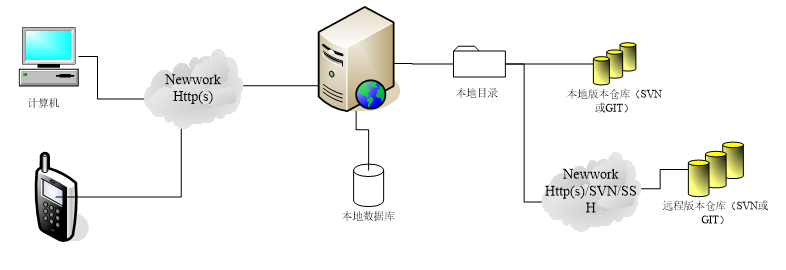
**文件管理系统设计报告**

1. **系统简介**

一个基于Web的文件管理系统，类似网盘。支持文件权限管理和历史版本管理，支持文件在线预览和在线编辑，支持压缩文件在线解压预览，支持文件在线分享、支持重复文件的秒传和大文件的断点续传，支持文件名和文件内容搜索。NextCloud虽然强大，但是要能随心所欲修改不太可能，因此在某些场景下还是需要一款能由我们自己掌控的文件管理系统存在。

1. **概要设计**
   1. **系统构架**

系统采用BS架构，使用数据库记录文件节点信息， GIT仓库存储文件历史版本，系统拓扑结构如下图所示：



* 1. **技术选型**

后端框架:SpringBoot+mybatis+mysql

版本控制：JGit

全文搜索：lucene、ikanalyzer

Office在线预览：jodconverter、poi

Office在线编辑：OpenOffice、only Office

* 1. **关键技术点**
     1. **文件并发访问**

所有的仓库、文件、目录都统一抽象为一个doc对象。通过数据库来实现文件锁，保证同一时间只有一个线程能获取到该节点。这种文件管理系统对数据库的访问是非常少的，如果真的在乎这点，后面可以改为SETNX。关于文件锁失效的问题，失效时间在编码时具体考虑。

* + 1. **文件版本控制**

使用git做版本控制，放弃svn。因为git的版本切换非常快，而svn相对较慢，且svn属于商用协议只是目前免费而已。

2.3.2.1 新增文件

Lock Doc 🡪增加到本地🡪 Commit To 版本仓库🡪 Unlock Doc

失败：还原现场（删除新增的本地文件）

2.3.2.2 删除文件

Lock Doc 🡪 删除本地文件 🡪 Commit To 版本仓库 🡪 Delete Doc

失败：还原现场（恢复被删除的本地文件）

2.3.2.3 编辑文件 （3期在做）

Lock Doc 🡪本地文件替换🡪 Commit To 版本仓库 🡪 unlock Doc

失败处理：还原现场（恢复被修改的本地文件）

2.3.2.4 上传文件

上传文件如果是新增操作，流程同新增文件。

上传文件如果是覆盖操作，流程同修改文件。

2.3.2.4 重命名

Lock Doc 🡪 Rename 本地文件 🡪 Commit To 版本仓库 🡪 Commit To 版本仓库（不会失败） 🡪Unlock Doc

源本地文件不存在 🡪 失败

目标本地文件已存在🡪 失败

提交版本失败🡪失败

失败处理：还原现场（将目标本地文件重命名回源本地文件）

2.3.2.4 移动文件

Lock Doc 🡪 Move本地文件 🡪 Commit To 版本仓库🡪Commit To 版本仓库（不会失败） 🡪Unlock Doc

源本地文件不存在 🡪 失败

目标本地文件已存在🡪 失败

提交版本失败🡪失败

失败处理：还原现场（将目标本地文件复制回源本地文件）

2.3.2.4 复制文件

Lock Doc 🡪 复制文件🡪 Commit To 版本仓库 🡪 Commit To 版本仓库（不会失败） 🡪Unlock Doc

源本地文件不存在 🡪 失败

目标本地文件已存在🡪 失败

提交版本失败🡪失败

失败处理：还原现场（将目标本地文件删除）

* + 1. **虚拟文件系统(延后)**

该功能看情况开发。因为我们会有uam来做文件编辑支持。虚文件因为会存在文件引用，在web上控制是一件非常消耗资源的事情。

* + 1. **权限管理**

这里的权限管理分为：仓库权限和文件权限。系统需要能对仓库及文件设置每个用户的访问权限，访问权限分为：读、写、增、删、完全控制。为了减少授权设置的记录数量，系统采用权限继承的方式来实现用户文件权限控制(就是说当数据库查不到改文件权限时递归向上查找它父级文件夹的权限。)

新增的用户是没有仓库的，用户需要自己创建仓库。仓库分3种：普通仓库、版本仓库、协作仓库。

普通仓库：就是一般的网盘，不提供版本控制、秒传等功能，用户可与保存一些私有的东西，但是会提供分享功能。(这种仓库占用磁盘空间最小)

版本仓库：在普通仓库基础上添加版本控制功能，允许版本回溯。但是不提供协同开发功能。所有文件都是自己独占的。新旧会强推覆盖，但是会留下版本记录。因此这种仓库会占用较大的磁盘空间，每个版本文件都是一份全量数据。

协作仓库：由管理员创建并初始化仓库内容，普通用户可以在现有的协作仓库种选择一个来初始化协作仓库。协作仓库中修改的内容会自动合并，因此会有冲突的情况存在。这个仓库的文件会被设计为两阶段提交：参考git的commit 和push。会有一个强推功能，由配置决定谁能使用，默认是仓库的创建者能使用强推功能。管理员在创建协作仓库后能为仓库指定仓库成员，只有被邀请的人才能拉取该仓库进行团队协作。参考GitLib.

仓库可以设置密码，也能将整个仓库分享出去。用户可以将自己的仓库分享出去，和别人共享文件或者让别人来帮忙整理仓库中的文件。(协作仓库不能做该操作)

注：本来不想做仓库的，默认每个用户只有一个版本仓库。考虑到后面的协同，还是做了吧！

* + 1. **用户**

用户模块还是分3层，用户，角色，权限。角色充当权限组的作用，一个用户可以有多个角色，权限叠加。系统采用jwt令牌校验身份，其他系统的令牌，只要是符合规范都认可(方便后期系统整合)。

用户权限指的是对于API的调用。

1. **详细设计**
   1. **数据库设计**
   2. **系统设计**

系统页面采用尽可能简洁的设计，前期做起来比较方便。

后台文件关系系统设计：每个用户都可以创建多个仓库，~~每个仓库都会对应一个本地仓库和一个版本仓库。用户所有的操作都是先在本地仓库操作，然后再复制到版本仓库提交。同时、每个本地仓库都会有一个额外的temp文件夹，用于后期的zip和office编辑缓存。~~

先尝试大粒度的控制方法，每个用户都有一个temp目录，文件先在temp中保存，然后再复制到版本仓库做提交，上传时就开始锁定文件。

* + 1. 登陆

因为是内部系统，不开放注册。用户只能由管理员在系统中添加。用户登陆jwt令牌请使用SessionStorage保存，有效期为4小时，当浏览器关闭时必须重新登陆(参考网盘客户端关闭)。

* + 1. 首页

提供一个搜索框，触发全文检索，方面用户跨仓库查找文件

提供其他设置页面的进入入口

为管理员提供后台管理页面入口(管理员也会有自己的仓库，只是比普通用户多一个用户管理页面而已

页面内容主体部分显示仓库列表

后期可以出一个和即时通讯融合的版本，一半页面显示仓库，另一半显示通讯

* + 1. 个人中心

提供个人信息编辑，密码修改等功能

* + 1. 我的分享

一个列表，可以对自己的分享链接进行管理

* + 1. 仓库管理

一个列表，可以对自己的仓库列表进行管理，影响首页的仓库显示

* + 1. 管理后台(root)

提供用户管理、用户组管理、仓库管理和系统管理三个模块

用户管理：提供用户的增删改查、权限修改等功能

用户组管理：对用户进行分组、管理员可以作用在组上，从而管理对应用户

仓库管理：管理员可以添加协作仓库，可以对用户组下的其他用户的仓库经行操作。

系统管理：可以对系统的配置参数进行修改，仅root可用