

LocalDrive: Uma Aplicação Web para Gerenciamento de Arquivos com NFS

Edilson de L. Leão, Klebson do C. Silva, Maick da S. Damasceno, Nalberth de L. Castro, Wendew M. S. Magalhães

Faculdade de Sistemas de Informação - Universidade Federal do Pará (UFPA) -
Campus Universitário do Tocantins / Cametá.

{leaoedilson44, klebsoncarmo1, maicksd2017,
nalberthcastro1510}@gmail.com, {wm.magalhaes}@outlook.com

Abstract. *This article discusses the development of a web application dedicated to the efficient management of local files using the NFS (Network File System) protocol. The main goal is to provide an affordable and robust solution to facilitate file sharing and manipulation in local environments.*

Resumo. *Este artigo aborda o desenvolvimento de uma aplicação web dedicada ao gerenciamento eficiente de arquivos locais utilizando o protocolo NFS (Network File System). O objetivo principal é fornecer uma solução acessível e robusta para facilitar o compartilhamento e manipulação de arquivos em ambientes locais.*

1. Introdução

O avanço constante da tecnologia e a crescente necessidade de armazenamento e compartilhamento de dados, faz com que surjam soluções inovadoras para simplificar o gerenciamento de arquivos. Nesse contexto, o desenvolvimento da aplicação web denominada "LocalDrive" destaca-se como uma ferramenta eficiente e intuitiva para o gerenciamento de arquivos, utilizando o protocolo NFS (*Network File System*).

O NFS é uma tecnologia que permite a comunicação entre sistemas operacionais distintos, possibilitando o acesso remoto a arquivos como se estivessem armazenados localmente. Neste contexto, o "LocalDrive" se apresenta como uma solução abrangente, unindo a praticidade da interface web com a eficiência do NFS, proporcionando uma experiência integrada e eficaz para o armazenamento e compartilhamento de dados. Sendo assim, delineia-se a importância do "LocalDrive" no contexto atual, destacando como sua abordagem inovadora simplifica o gerenciamento de arquivos em um ambiente cada vez mais interconectado e dinâmico.

2. Justificativa

O desenvolvimento de uma aplicação web é uma escolha estratégica fundamental nos dias atuais, devido às suas inúmeras vantagens e benefícios. Com a crescente dependência da tecnologia e da conectividade, investir nesse tipo de aplicação é crucial para alcançar um público amplo e diversificado, além de proporcionar uma experiência de usuário fluida e acessível em diferentes dispositivos. É relevante destacar os motivos que tornam essa abordagem tão relevante no cenário atual, ressaltando suas contribuições para a eficiência

operacional, escalabilidade, acessibilidade e competitividade no mercado.

2. Fundamentação Teórica

2.1 Conceitos de Engenharia de Software

A Engenharia de Software é guiada por princípios e práticas essenciais para o desenvolvimento eficiente e eficaz de software. No âmbito dos princípios, destaca-se a importância da clareza no código e na documentação, garantindo compreensibilidade e facilitando a colaboração entre desenvolvedores. Os processos de desenvolvimento são essenciais, com modelos como o cascata, incremental e ágil, oferecendo abordagens distintas para lidar com os desafios do desenvolvimento de software. A compreensão e documentação de requisitos são cruciais, envolvendo a identificação e análise detalhada das funcionalidades desejadas pelos usuários. O princípio da abstração destaca a relevância de focar apenas as características essenciais do sistema, tornando-o mais compreensível.

2.2 NFS (*Network File System*)

Quando consideramos o NFS como uma tecnologia subjacente, trabalhos como o de Sandberg et al. (1985) são fundamentais para entender os conceitos por trás do protocolo NFS. Esse trabalho da Sun Microsystems detalha a arquitetura e os princípios fundamentais que tornam o NFS uma escolha robusta para o compartilhamento de sistemas de arquivos em ambientes distribuídos.

2.3 Desenvolvimento de Aplicações Web

O desenvolvimento de uma aplicação web para o gerenciamento de arquivos com NFS requer uma abordagem rigorosa de engenharia de software, incorporando os princípios fundamentais e adotando práticas modernas. O entendimento das particularidades do desenvolvimento web, juntamente com a consideração dos desafios associados ao uso do NFS, é crucial para criar uma solução eficiente, segura e de alta qualidade.

No âmbito do desenvolvimento de aplicações web, autores como Flanagan (2011) oferecem percepções valiosas sobre as particularidades do desenvolvimento do lado do cliente (front-end) e do lado do servidor (back-end). Abordagens ágeis, conforme propostas por autores como Beck e Andres (2004), também têm sido amplamente adotadas para lidar com a complexidade inerente ao desenvolvimento web.

3. Metodologia

Observa-se que o processo de revisão de literatura demanda a descrição de tópicos, assim, causando um melhor entendimento a respeito do desenvolvimento de uma aplicação web para armazenamento de arquivos com uso de NFS. Trata-se do passo inicial para a configuração do conhecimento científico, uma vez que por meio de tais procedimentos em prol da construção do aperfeiçoamento, irão surgir novas teorias, possibilitando uma série de oportunidades para que outros estudos venham se aprofundar ainda mais no tema proposto.

Tendo isso em vista, pretendeu-se desenvolver um sistema observando as seguintes métricas, foi realizado o levantamento bibliográfico sobre NFS, aplicações web, engenharia de software com intuito de ter um melhor entendimento do problema e conhecimento das melhores técnicas e práticas para solucioná-lo. Em seguida, foi feito o detalhamento dos requisitos funcionais de acordo com as necessidades dos usuários do sistema e os requisitos não funcionais de acordo com as necessidades do próprio sistema.

Após essas atividades primordiais, coube então realizar para especificar o sistema foi necessário definir a estrutura da aplicação por meio de diagramas propostos pela engenharia de software, tendo como base os requisitos previamente definidos. Para auxiliar na diagramação, foi crucial o uso da ferramenta *Lucidchart*. Por fim, após um escopo definido cabe então a Implementação: implementar o sistema utilizando uma linguagem TypeScript, e o frontend desenvolvido com o *Framework ReactJs*, e estilização com o *Framework Tailwind CSS*. Para auxílio no desenvolvimento foi imprescindível o uso do ambiente de desenvolvimento integrado Visual Studio Code.

4. Projeto da Aplicação

4.1 Requisitos do Sistema

O levantamento de requisitos é fundamental para o desenvolvimento bem-sucedido de qualquer aplicação. Para a criação desta aplicação que utiliza o serviço NFS (Network File System) como meio de armazenamento, foi necessária uma análise cuidadosa das necessidades e expectativas do usuário, bem como dos requisitos técnicos e de segurança. Abaixo veremos os detalhes dos requisitos, sendo eles divididos em duas partes, Funcionais e Não funcionais.

4.1.1 Requisitos Funcionais

Os requisitos funcionais referem-se aos requisitos que estão relacionados com a maneira com que o sistema deve operar, onde se especificam as entradas e saídas do sistema e o relacionamento comportamental entre elas, assim como a interação com o usuário.

Tabela 1. Requisitos funcionais do sistema

ID	Ação	Descrição
RF 01	Criar perfil de usuário do Sistema.	O usuário se cadastra no sistema, adicionando um nome de usuário, e-mail, e cria uma senha para acesso.
RF 02	Fazer Login	O usuário poderá entrar no sistema utilizando e-mail e senha previamente cadastrados.
RF 03	Fazer Upload de Arquivos	O usuário poderá fazer Uploads de arquivos locais para o sistema.

ID	Ação	Descrição
RF 04	Gerenciar Arquivos	O usuário poderá manipular os arquivos de seu domínio dentro do sistema.
RF 05	Fazer Download de Arquivos	O usuário poderá baixar os arquivos de seu domínio previamente enviados.
RF 06	Gerenciar Perfil de Usuário	O usuário poderá gerenciar seu perfil dentro do sistema, alterando dados ou excluindo o cadastro.
RF 07	Recuperar Senha de Acesso	O usuário poderá recuperar sua senha de acesso ao sistema.

4.1.2 Requisitos Não Funcionais

Os Requisitos não funcionais são características ou propriedades que não dizem respeito diretamente ao comportamento específico de uma funcionalidade do sistema, mas sim a atributos globais que impactam sua operação e desempenho. Eles definem as condições sob as quais o sistema deve operar. A Tabela 2 mostra os requisitos não funcionais da aplicação “Local Drive”.

Tabela 2. Requisitos Não Funcionais do sistema

ID	Configuração	Descrição
RNF 01	Interface Intuitiva e Amigável	O usuário do sistema deve ter facilidade de uso do sistema, ou seja, realizar tarefas (inclusão, alteração, consulta, exclusão) com menos de 30 minutos de treinamento. Para confirmação disso, será realizado um teste de usabilidade.
RNF 02	Sistema WEB	O usuário utilizará o sistema através de um navegador web.
RNF 03	Segurança	O sistema deverá possuir login e senha individuais para cada usuário da interface de cadastro, de forma a restringir o seu acesso.
RNF 04	Limite no Tamanho de Arquivos para Upload	O sistema deverá limitar o envio de arquivos, não podendo ultrapassar 500 MB por upload.
RNF 05	Uso de NFS	O sistema deverá utilizar os serviços de NFS para gerenciamento de arquivos enviados pelos usuários.

RNF 06	Limites de Armazenamento	O sistema deverá disponibilizar uma capacidade de armazenamento de 5 GB para cada usuário cadastrado.
RNF 07	Opções de Recuperação de Acesso	O sistema deverá oferecer opções de recuperação de acesso, em casos de perda das informações de login.

4.2 Diagramas do Sistema

Os diagramas são representações visuais que ilustram diferentes aspectos e componentes do sistema, oferecendo uma visão gráfica para facilitar a compreensão, análise, projeto e documentação desse sistema. Esses diagramas são amplamente utilizados na engenharia de software e outras disciplinas para representar a arquitetura, estrutura, comportamento, interações e fluxos de um sistema. Abaixo temos alguns diagramas feitos para o sistema LocalDrive.

4.2.1 Diagrama de casos de uso

Esse diagrama representa as interações entre os usuários e o sistema, identificando os diferentes casos de uso que descrevem as funcionalidades do sistema do ponto de vista do usuário. A Figura 1 apresenta os casos de uso geral do sistema, onde os usuários poderão criar um perfil de acesso, efetuar o login, em seguida fazer uploads de arquivos. Além dessas funcionalidades, será possível fazer downloads, atualizar e deletar arquivos de seu domínio adicionados no sistema.

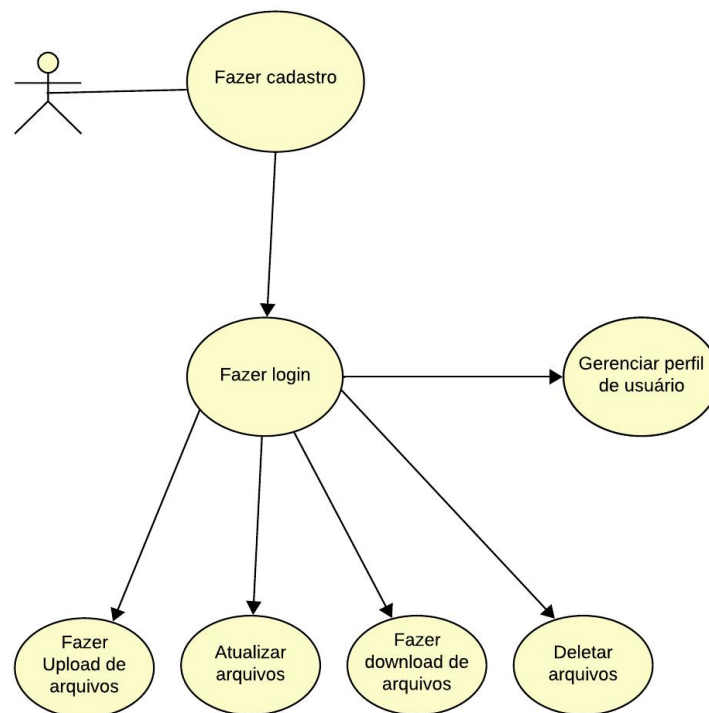


Figura 1. Diagrama de Casos de uso geral.

4.2.2 Diagrama de entidade e relacionamento

O Diagrama de Entidade e Relacionamento (DER) é uma ferramenta visual que representa as entidades relevantes do sistema e os relacionamentos entre elas. Foi usado no projeto, especificamente no banco de dados para modelar a estrutura de dados e as interações entre as entidades.

Na Figura 2 temos uma representação visual de como os usuários e arquivos estão relacionados em um sistema de banco de dados. Ela ajuda a entender a estrutura e o funcionamento do sistema. Neste diagrama de relacionamento entre um usuário e um arquivo, a classe Usuário possui um relacionamento de um para muitos com a classe Arquivo, isso significa que um usuário pode ter muitos arquivos. A classe Usuário possui atributos como Id, Nome, Username, E-mail e Senha, que armazenam informações sobre o usuário, enquanto a classe Arquivo possui atributos como Id, Nome, Chave, Caminho do arquivo e Tamanho, que armazenam informações sobre o arquivo.

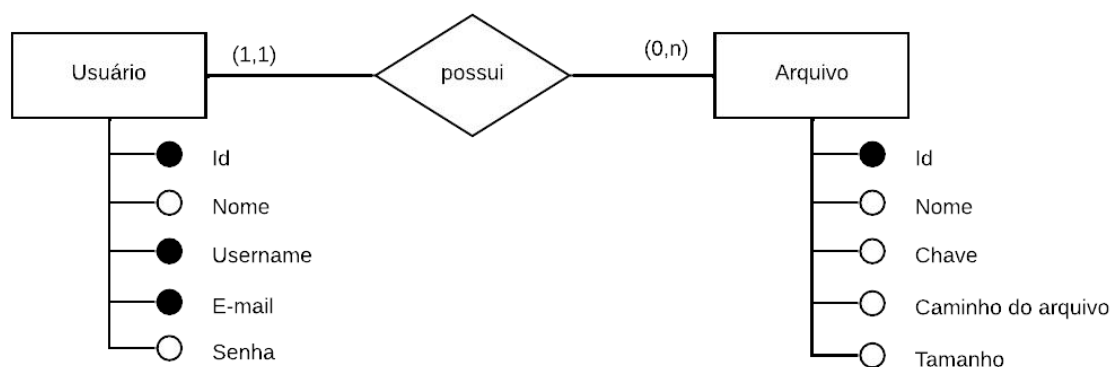


Figura 2. Diagrama de Entidade e Relacionamento da Aplicação Local Drive

4.2.3 Diagrama de classe

Esse diagrama ilustra a estrutura de classes e seus relacionamentos no sistema, mostrando como os objetos são organizados e interagem entre si. A Figura 3 é uma representação visual das classes *User* e *File* e suas interações. A classe *User* representa um usuário em um sistema. Ela tem vários atributos, incluindo *id*, *name*, *email*, *password*, *tokenResetPass*, *tokenExpiration*, *created_at*, e *updated_at*, que armazenam informações sobre o usuário. Além disso, a classe *User* tem métodos como *createUser*, *updateUser*, e *deleteUser* que permitem criar, atualizar e deletar um usuário, respectivamente.

A classe *File* representa um arquivo no sistema. Ela tem atributos como *id*, *name*, *key*, *size*, *created_at*, e *updated_at* que armazenam informações sobre o arquivo. A classe *File* também tem métodos como *uploadFile*, *deleteFile*, e *downloadFile* que permitem fazer upload, deletar e baixar um arquivo, respectivamente. O diagrama mostra como essas duas classes estão relacionadas e interagem entre si através de seus atributos e métodos.

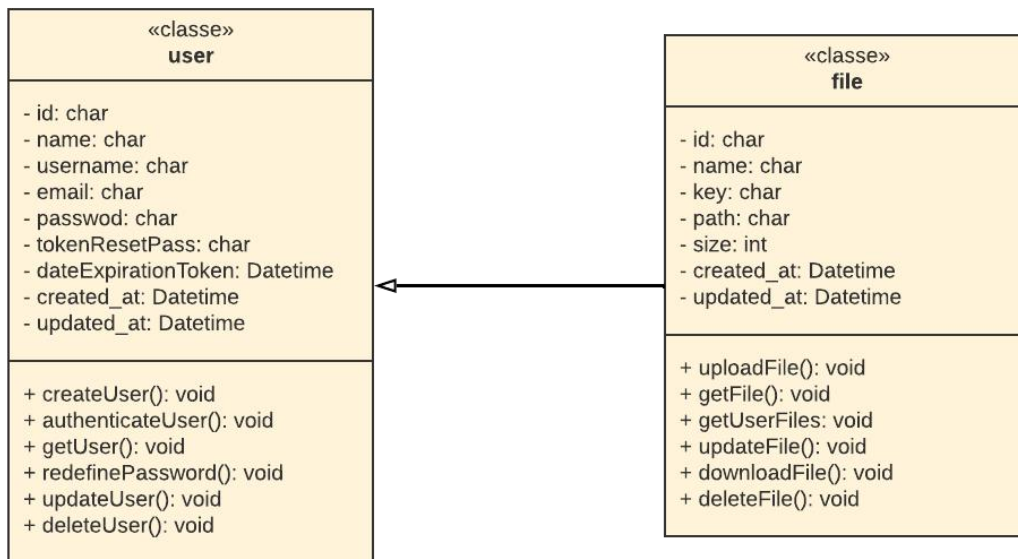


Figura 3: Diagrama de Classes da aplicação

4.3 Arquitetura

O sistema foi desenvolvido utilizando a arquitetura de três camadas, é um modelo amplamente utilizado no desenvolvimento de aplicativos, proporcionando uma separação clara e organizada das diferentes funcionalidades do sistema. Para Sommerville (8ª Edição), essa arquitetura consiste em três camadas principais: a camada de apresentação, camada de negócio e a camada de dados, a figura 4 representa a estrutura detalhada de cada camada.

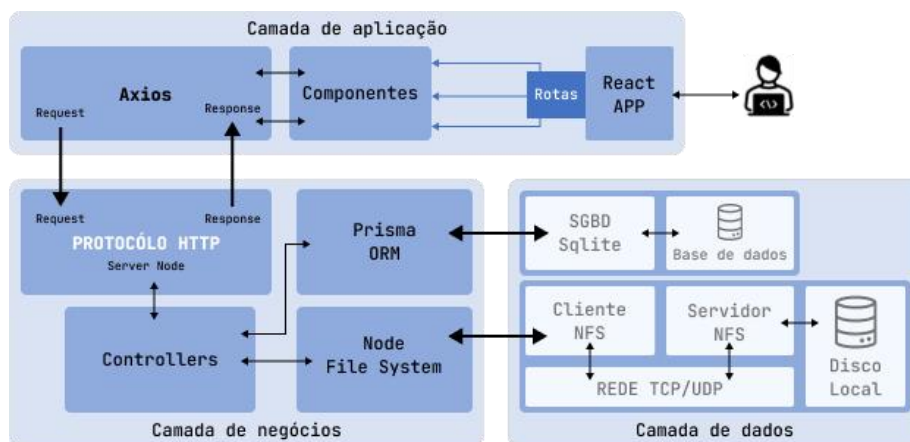
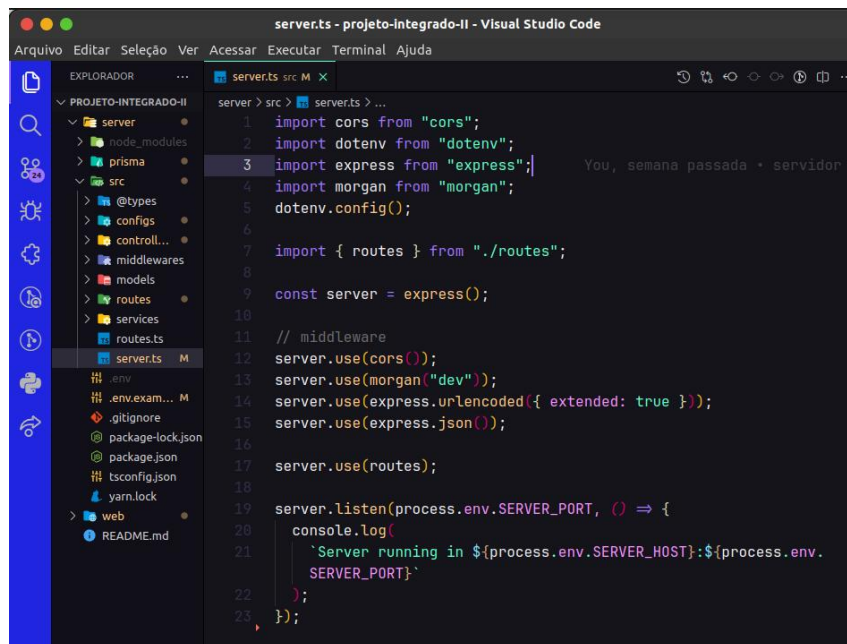


Figura 4. Arquitetura de Três Camadas usada no LocalDrive

A camada de apresentação do sistema é responsável pela interação com o usuário, onde possui as interfaces gráficas e as funcionalidades de entrada e saída de dados. A camada de negócio é onde se concentra a lógica e as regras de negócios do sistema. Por fim, a camada de dados é responsável pela persistência e acesso aos dados da aplicação. Ambas camadas conversam respectivamente entre si utilizando de suas tecnologias de implementação para que todas elas atuem como uma só.

5. Implementação

Nesta seção será abordado o desenvolvimento da aplicação, para codificação optou-se pelo uso da IDE Visual Studio Code, pois apresenta recursos que facilitam no desenvolvimento com o uso de extensões e outros tipos de integrações, a Figura 5 representa um fragmento do código fonte, porém todo conteúdo e artefatos deste projeto estão disponíveis na íntegra através do GitHub (CASTRO, 2023).



```
server.ts - projeto-integrado-II - Visual Studio Code
Arquivo  Editar  Seleção  Ver  Acessar  Executar  Terminal  Ajuda

EXPLORADOR
PROJETO-INTEGRADO-II
  server
  node_modules
  prisma
  src
    @types
    configs
    controll...
    middlewares
    models
    routes
    services
    routes.ts
  server.ts M
  .env
  .env.exam... M
  .gitignore
  package-lock.json
  package.json
  tsconfig.json
  yarn.lock
  web
  README.md

server > src > server.ts > ...
1  import cors from "cors";
2  import dotenv from "dotenv";
3  import express from "express";
4  import morgan from "morgan";
5  dotenv.config();
6
7  import { routes } from "./routes";
8
9  const server = express();
10
11 // middleware
12 server.use(cors());
13 server.use(morgan("dev"));
14 server.use(express.urlencoded({ extended: true }));
15 server.use(express.json());
16
17 server.use(routes);
18
19 server.listen(process.env.SERVER_PORT, () => {
20   console.log(
21     `Server running in ${process.env.SERVER_HOST}:${process.env.
22       SERVER_PORT}`
23   );
24 });
```

Figura 5: Fragmento do código fonte da aplicação.

6. NFS

Para configuração do NFS o servidor deve estar conectado na mesma rede na qual os clientes NFS serão configurados. Para facilitar a implementação desta aplicação foi utilizada a rede local do ambiente de desenvolvimento, Linux/Ubuntu. A tabela 3 lista os comandos utilizados no terminal para a configuração em ambiente Unix.

Tabela 3. Comandos utilizados no Terminal para Configuração em Ambiente Linux

Comando no terminal Unix	Descrição
sudo apt-get install nfs-kernel-server	Para instalar pacotes do protocolo NFS.
sudo mkdir /mnt/server_nfs_localDrive /mnt/localDrive	Para criar os repositórios que funcionarão como servidor e cliente respectivamente.
sudo chmod 777 /mnt/server_nfs_localDrive /mnt/localDrive	Para adicionar as permissões de escrita e leitura nos diretórios cliente e servidor.

sudo nano /etc/exports	Para configuração de caminhos e clientes NFS, e adicionado a configuração “/mnt/server_nfs_localDrive *(rw,sync,no_root_squash)” para definição de um cliente NFS e suas permissões de acesso ao servidor.
sudo systemctl restart nfs-server	Para reiniciar o servidor e ler o arquivo de configuração.
sudo apt-get install nfs-common	Para instalação do cliente NFS.
sudo mount 127.17.0.1:/mnt/server_nfs_localDrive /mnt/localDrive	Montar a unidade de conexão do cliente e atribuir a unidade do diretório disponível para o dispositivo que foi configurado no servidor NFS.

7. Aplicação Server-side/Backend

Foi utilizado o NodeJS para criar o servidor que carrega toda a lógica e regras de negócio da aplicação, implementadas com a linguagem de programação TypeScript, que receberá todas as requisições HTTP (*Pegar acrônimo*) das aplicações frontend. Esta linguagem traz recursos de orientação a objetos, permitindo o processamento ao estilo MVC (*Model, View e Controller*). Foi utilizado também o ORM (*Object Relational Model*) Prisma que atribui o paradigma de objetos para o banco de dados, o que facilita a implementação e conexão com o banco de dados.

7.1. Banco de Dados

Para persistência de dados da aplicação foram utilizados o SGBD (Sistema Gerenciador de Banco de Dados) SQLite, o SGBD utiliza a linguagem SQL para a criação e o gerenciamento do banco de dados. A escolha deste SGBD deu-se por questão da sua fácil configuração de ambiente e armazenamento, o que economiza bastante tempo para configurá-lo, ideal para aplicações desenvolvidas em curto espaço de tempo.

7.2. Aplicação Client-side/Frontend

Foi explorado os recursos do Framework ReactJs, consumindo dinamicamente a API (*Falta pegar o acrônimo*) provida pelo backend, para estilização foi utilizado o Framework Tailwind CSS (*Cascading Style Sheets*), ferramentas amplamente utilizadas em interfaces web modernas e suportadas pela maioria dos navegadores disponíveis na atualidade. Para fazer as requisições ao servidor o uso do Axios foi indispensável, pois permite enviar dados em vários formatos nas requisições inclusive nos formatos, JSON (*JavaScript Object Notation*) e Multipart Form.

8. Resultados Obtidos

Durante a execução do projeto, foram obtidos resultados relevantes e satisfatórios com

relação ao funcionamento da aplicação, a mesma apresentou eficiência no gerenciamento de arquivos, simplificou o acesso e a manipulação de dados por meio do protocolo NFS. Além disso, foram incorporadas medidas de segurança para garantir a integridade e a confidencialidade dos dados, proporcionando um ambiente seguro para o armazenamento e compartilhamento de arquivos.

A aplicação apresentou um desempenho otimizado, garantindo tempos de resposta rápidos e eficientes, mesmo em ambientes com volume de usuários simultâneos. O desenvolvimento priorizou uma interface de usuário intuitiva, proporcionando uma experiência amigável e fácil de usar, facilitando o aprendizado para novos usuários. A Figura 5 demonstra a tela principal da aplicação com a imagem do diretório que funciona como servidor NFS, assim como as informações referentes aos arquivos que ficam salvos no banco de dados, ambas compartilham as mesmas informações. É importante ressaltar que essas informações técnicas não ficam visíveis para o usuário, foram mostradas apenas para fins de demonstração sobre como o sistema funciona na prática.

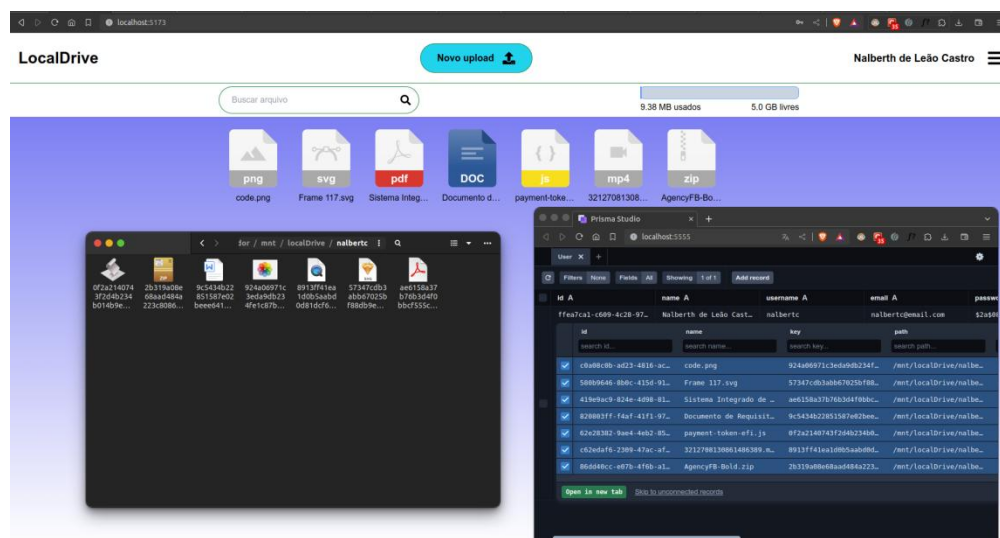


Figura 5. Interface inicial da aplicação e o diretório do servidor NFS

8. Considerações Finais

Ao longo deste trabalho, foi desenvolvida e explorada uma aplicação web dedicada ao gerenciamento de arquivos, utilizando o Protocolo de Sistema de Arquivos em Rede (NFS) como base para a integração entre sistemas distribuídos. Durante o processo de desenvolvimento e análise, diversos aspectos fundamentais foram analisados e mereceram destaque, como os avanços tecnológicos e funcionalidades implementadas.

A aplicação concebida apresentou um conjunto robusto de funcionalidades, permitindo o acesso remoto e a manipulação de arquivos de forma eficiente e segura. A utilização do NFS se mostrou crucial para a criação de um ambiente colaborativo e integrado, permitindo o compartilhamento de recursos entre diferentes máquinas de forma transparente. Em resumo, a criação desta aplicação web para gerenciamento de arquivos com o uso do NFS representa um passo significativo na facilitação do acesso e

compartilhamento de dados em ambientes distribuídos. As lições aprendidas e os desafios superados durante este processo são valiosos e servirão como base para futuros avanços na área de sistemas distribuídos e gerenciamento de dados.

Este trabalho representa apenas o início de uma jornada rumo a soluções mais abrangentes e eficazes para o gerenciamento de arquivos, espera-se que as contribuições aqui apresentadas sirvam como inspiração para pesquisadores e desenvolvedores neste campo em constante evolução. Embora tenhamos alcançado resultados abrangentes, permitimos que haja espaço para melhorias e expansões futuras nesta aplicação. A integração com outras tecnologias de virtualização e armazenamento em nuvem, por exemplo, poderia aumentar ainda mais a escalabilidade e a disponibilidade da solução.

9. Referências

- Beck, K. e Andres, C. (2004) “Extreme Programming Explained: Embrace Change”. Addison-Wesley Professional.
- Bezerra, Eduardo. “Princípios de Análise e Projeto de Sistemas com UML”. 3ª Tiragem, Rio de Janeiro, Editora Campus.
- Castro, Nalberth de Leão. (2023) “Código fonte da aplicação.” <<https://github.com/NalbertC/projeto-integrado-II>>. novembro.
- Elmasri, Ramez. Navathe, Shamkant B. (2005) “Sistemas de banco de dados.” São Paulo, Pearson Addison Wesley.
- Flanagan, D. (2011). Javascript The Definitive Guide. O'Reilly Media.
- Gamma, Erich. (2007) “Padrões de Projeto: recursos reutilizáveis de software orientado a objetos.” Porto Alegre, Bookman.
- Presmman, Roger S. (2011) “Engenharia de Software: uma abordagem profissional”. 7. ed. Porto Alegre, AMGH.
- Sandberg, M., et al. (1985) “Ischemia-Induced Shift of Inhibitory and Excitatory Amino Acids from Intra-to Extracellular Compartments.
- Sommerville, Ian. “Engenharia de Software”, 8ª Edição. Editora Pearson Education.