UNIVERSIDADE FEDERAL DE ITAJUBÁ

SISTEMAS DE INFORMAÇÃO

Uma Solução Descentralizada para o Compartilhamento Seguro de Mídias em uma Agência de Marketing Colaborativa

2020004243 - MARCELO MAGALHÃES SILVA
2018003703 - HENRIQUE CASTRO OLIVEIRA
2020032785 - ANTONIO BITTENCOURT PAGOTTO

COM242 - SISTEMAS DISTRIBUÍDOS

Prof. Rafael Frinhani

INSTITUTO DE MATEMÁTICA E COMPUTAÇÃO
UNIFEI - Itajubá

2023

UNIVERSIDADE FEDERAL DE ITAJUBÁ
INSTITUTO DE MATEMÁTICA E COMPUTAÇÃO
SISTEMAS DE INFORMAÇÃO
COM242 - SISTEMAS DISTRIBUÍDOS

Uma Solução Descentralizada para o Compartilhamento Seguro de Mídias em uma Agência de Marketing Colaborativa

1 Introdução

O armazenamento de dados no meio corporativo tem sido uma peça fundamental na evolução da gestão de informações nas empresas. No passado, as organizações confiavam em servidores locais para armazenar e gerenciar seus dados, um processo que exigia investimentos substanciais em infraestrutura e recursos especializados. Esses servidores locais ofereciam controle direto sobre os dados, mas também apresentavam desafios significativos, como custos elevados, complexidade de manutenção e a falta de flexibilidade para acompanhar o crescimento das necessidades empresariais. No entanto, com o avanço tecnológico e a ascensão da computação em nuvem, uma mudança fundamental começou a ocorrer. As empresas começaram a migrar seus dados para a nuvem, aproveitando os benefícios proporcionados por esse modelo de armazenamento remoto e escalável.

A computação em nuvem oferece uma gama de vantagens, incluindo redução de custos, acesso global, elasticidade, segurança aprimorada e facilidade de colaboração. Essa transição gradual para a nuvem não apenas revolucionou a forma como as empresas lidam com seus dados, mas também abriu caminho para a inovação e transformação digital, permitindo que as organizações se concentrem em suas principais competências e impulsionem o crescimento de maneira mais eficiente.

A computação em nuvem revolucionou a forma como as organizações acessam servidores, eliminando a necessidade de criar uma infraestrutura própria para hospedar suas aplicações. Anteriormente, as empresas precisavam investir em hardware, data centers e equipes especializadas para gerenciar seus servidores locais. No entanto, com a adoção da nuvem, o acesso a servidores tornou-se muito mais fácil e rápido. As empresas podem simplesmente provisionar recursos de computação e armazenamento na nuvem, sem se preocupar com a manutenção física dos servidores. Isso resulta em maior agilidade, escalabilidade e redução de custos, permitindo que as organizações se concentrem em suas atividades principais, em vez de se preocuparem com a infraestrutura tecnológica.

Embora a computação em nuvem seja uma evolução significativa em sistemas distribuídos, é importante reconhecer que ela não é a solução mais segura em todos os cenários. A segurança dos dados é uma preocupação legítima para muitas organizações, especialmente quando se trata de informações sensíveis ou confidenciais. É por isso que algumas empresas optam por manter certas aplicações em ambientes e máquinas locais, onde têm maior controle e segurança sobre seus dados. Essa abordagem híbrida, combinando ambientes locais e serviços em nuvem, permite que as organizações personalizem sua estratégia de segurança de acordo com suas necessidades específicas, encontrando o equilíbrio adequado entre conveniência e proteção dos dados.

1.1 Problema

Embora a computação em nuvem tenha trazido uma série de benefícios para o armazenamento de dados corporativos, não está isenta de desafios, especialmente no que diz respeito à segurança, como citado na introdução. Com os dados armazenados em servidores remotos e acessíveis pela internet, há uma preocupação legítima em relação à privacidade e à vulnerabilidade a possíveis violações de segurança. Empresas estão começando a reconhecer essas preocupações e optando por trazer de volta algumas aplicações e dados para servidores locais.

Essa mudança de paradigma permite um maior controle sobre a segurança dos dados, já que eles estão armazenados em um ambiente relativamente mais protegido e de acesso restrito. Além disso, existem abordagens peer-to-peer (P2P), que é o principal tema a ser abordado neste artigo, onde são criadas redes de máquinas locais para armazenar dados de forma distribuída e descentralizada. Essa abordagem oferece maior segurança, pois os dados são armazenados localmente, reduzindo a exposição a possíveis ameaças externas e é mais barato desenvolver uma rede assim do que investir em um servidor local dedicado.

A combinação de máquinas locais e aplicativos P2P oferece uma solução mais resiliente e segura para empresas que buscam equilibrar a conveniência da nuvem com a necessidade de proteger seus dados sensíveis.

1.2 Objetivo

O objetivo deste artigo é abordar o problema enfrentado por uma agência de marketing, composta por várias filiais em todo o Brasil. Com a descentralização imposta pelo corte de investimentos no setor de TI, tornou-se crucial encontrar soluções eficientes que permitam manter a confidencialidade das campanhas até o lançamento, ao mesmo tempo em que garantem a acessibilidade e a colaboração entre as filiais. Este artigo pretende explorar estratégias e tecnologias que possam auxiliar a

agência a superar esses desafios, melhorando a gestão de mídias e promovendo uma maior eficiência operacional em seu ambiente altamente dinâmico.

2 Referencial

Uma referência relevante que discute o emprego do modelo Peer-to-Peer (P2P) em compartilhamento de arquivos entre um grupo de pessoas é um artigo da Universidade Federal de Santa Catarina, (art). Neste trabalho, o sistema desenvolvido utiliza o modelo P2P para distribuir o esforço computacional de armazenamento das informações entre os participantes da comunidade. Uma abordagem eficiente para o compartilhamento de arquivos é adotada, na qual os arquivos são segmentados em partes que podem ser baixadas simultaneamente de várias fontes. O estudo demonstra que o modelo P2P é uma solução viável e eficiente em termos de custos de implantação de serviços específicos, como armazenamento e compartilhamento de arquivos. A implementação do sistema foi realizada utilizando as linguagens de programação PHP e Java, com o objetivo de facilitar a distribuição dos processos e objetos, foi utilizado o Java RMI (Remote Method Invocation) em um ambiente de rede (Internet).

É importante ressaltar que, em contraste com a referência mencionada, a solução proposta neste artigo adota uma abordagem descentralizada para o compartilhamento seguro de mídias em uma agência de marketing colaborativa. Ao contrário do uso de uma interface web e do Java RMI, a solução proposta expõe uma API como interface principal. Através dessa API, os usuários poderão interagir com o sistema de compartilhamento de mídias, permitindo o acesso e a troca de arquivos de forma segura. Essa abordagem descentralizada e a exposição da API como interface têm como objetivo promover a flexibilidade e a interoperabilidade com outras aplicações e sistemas, possibilitando uma integração mais ampla e personalizada dentro do ecossistema da agência de marketing colaborativa.

Outra diferença significativa entre o artigo citado e a solução proposta neste trabalho é a abordagem utilizada para o envio de arquivos. Enquanto o artigo mencionado adota a estratégia de dividir os arquivos em partes e distribuí-las entre os nós da rede, na solução apresentada neste artigo, o envio dos arquivos é realizado de forma completa, de nó a nó. Isso significa que cada nó da rede receberá o arquivo completo, sem a necessidade de particioná-lo.

Essa abordagem de envio de arquivos completo de nó a nó oferece algumas vantagens específicas. Em primeiro lugar, simplifica o processo de compartilhamento, eliminando a necessidade de segmentar e rastrear partes individuais dos arquivos. Além disso, permite uma recuperação mais rápida e eficiente dos arquivos, pois cada nó tem acesso imediato à versão completa do arquivo. Essa abordagem também pode reduzir a complexidade do sistema, uma vez que não é necessário gerenciar a distribuição e o rastreamento de partes fragmentadas do arquivo entre os nós da rede.

Ao adotar o envio de arquivos completo de nó a nó, a solução proposta neste artigo busca simplificar o processo de compartilhamento e melhorar a eficiência no acesso e recuperação de arquivos. Essa abordagem, combinada com a descentralização e a exposição de uma API como interface principal, visa proporcionar um sistema flexível, interoperável e de fácil integração com outras aplicações e sistemas no contexto de uma agência de marketing colaborativa.

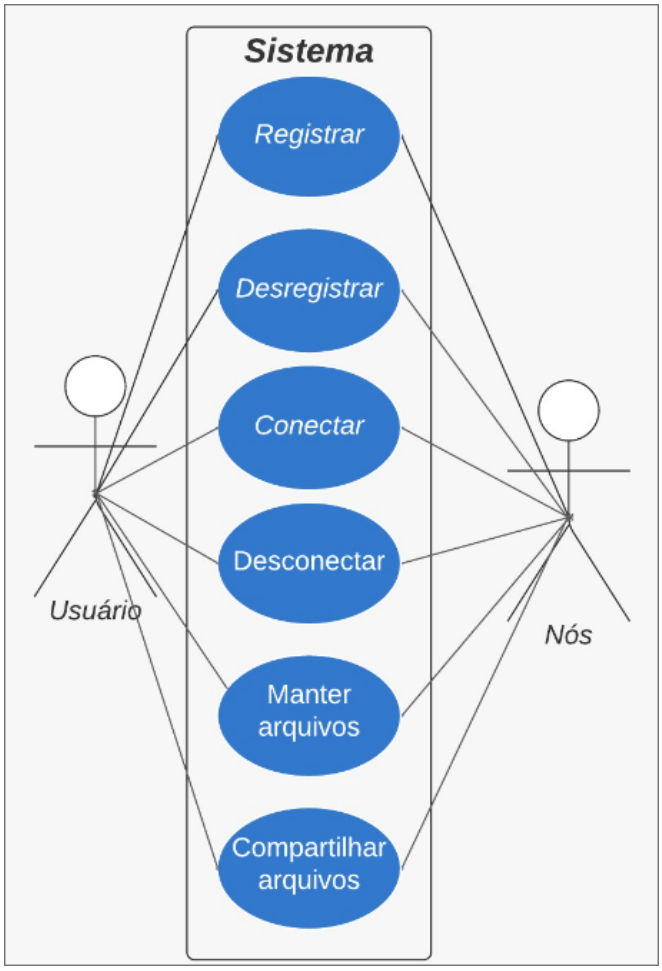
3 Método

Nessa sessão será detalhado como funciona a conexão da rede e a gerência de arquivos por parte de cada um dos Nós.

A aplicação foi planejada para permitir o máximo de disponibilidade de todos os arquivos na rede. Levando em conta o pior caso, onde os nós donos do arquivo podem estar desconectados, têm-se o sistema de cópias do arquivo em outros Nós, mas somente o Nó dono do arquivo pode executar as operações de exclusão, atualização e compartilhamento seletivo do arquivo da rede.

3.1 Ilustração do Sistema

O diagrama a seguir ilustra sem muitos detalhes técnicos o funcionamento do sistema com relação aos usuários, que seriam os funcionários da empresa que necessitam de acesso aos arquivos assim como a relação entre os nós da aplicação.



3.2 Processos

3.2.1 Criação de um Nó

Para se iniciar a conexão, o Nó que deseja se conectar a rede necessita do IP de algum dos nós, e a partir disso ele envia uma requisição para esse IP através da API a qual cada um dos Nós terá rodando localmente.

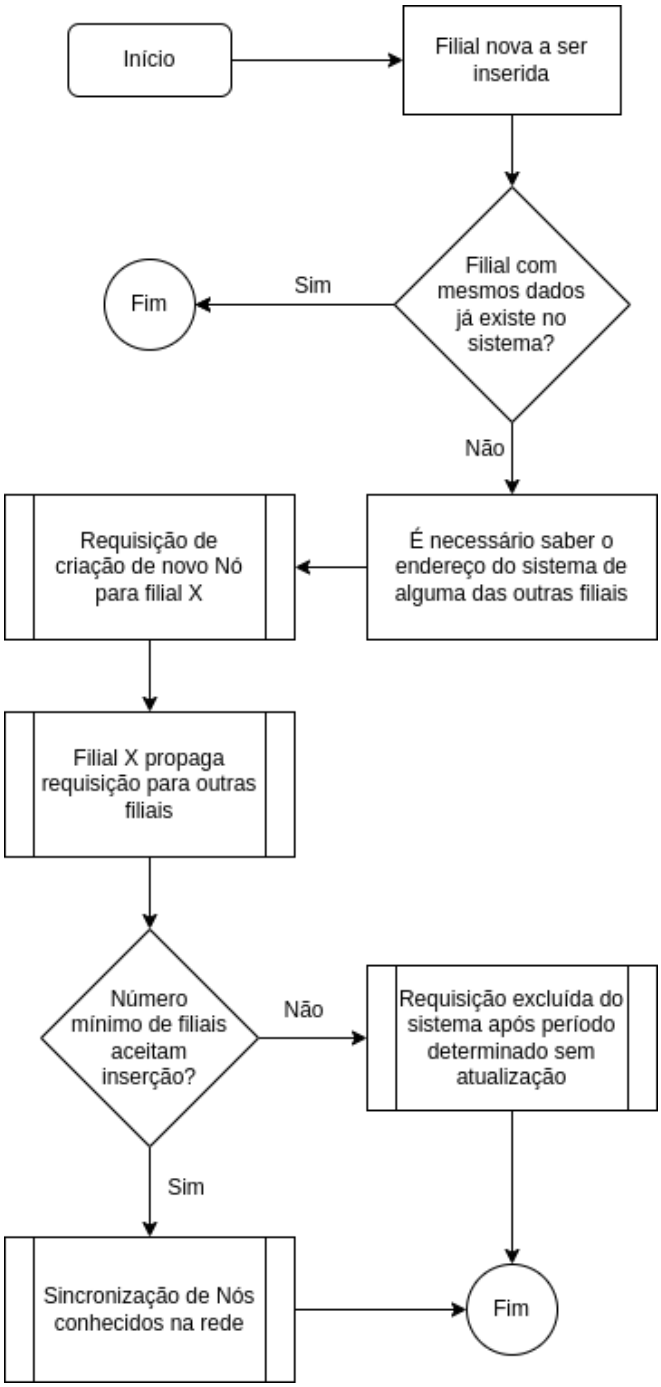


Figura 1: Exemplo do processo de criação de um Nó

O Nó requerente enviará uma requisição no endpoint POST /register , passando o IP de um Nó já conectado a rede como parâmetro. Após isso, será feita uma validação se o software do Nó requerente é válido ou não, a partir de uma autenticação JWT implícita dentro de cada uma das aplicações que estarão rodando nos Nós.

Caso a autenticação esteja errada, então a conexão é finalizada. Caso a autenticação esteja correta, então o Nó requerente recebe do Nó conectado a lista de todos os nós conectados em sua rede. Essas informações são salvas no banco de dados local do Nó requerente.

Após isso, o Nó requerente notifica cada um dos nós de sua base de dados que ele agora está conectado a essa rede, então todos os outros nós irão armazenar os dados do Nó requerente em seus respectivos bancos de dados.

3.2.2 Criação de um arquivo

Para se inserir um novo arquivo na rede tem-se o endpoint POST /files. Quando esse endpoint é invocado, será feita uma consulta no banco de dados local do Nó que invocou esse endpoint, então é buscado cada IP dos nós conectados a rede, e é enviada uma requisição para cada nó notificando que o nó o qual fez o upload de arquivos agora tem esse arquivo salvo em sua base de dados, atualizando a base de dados de todos os outros membros da rede.

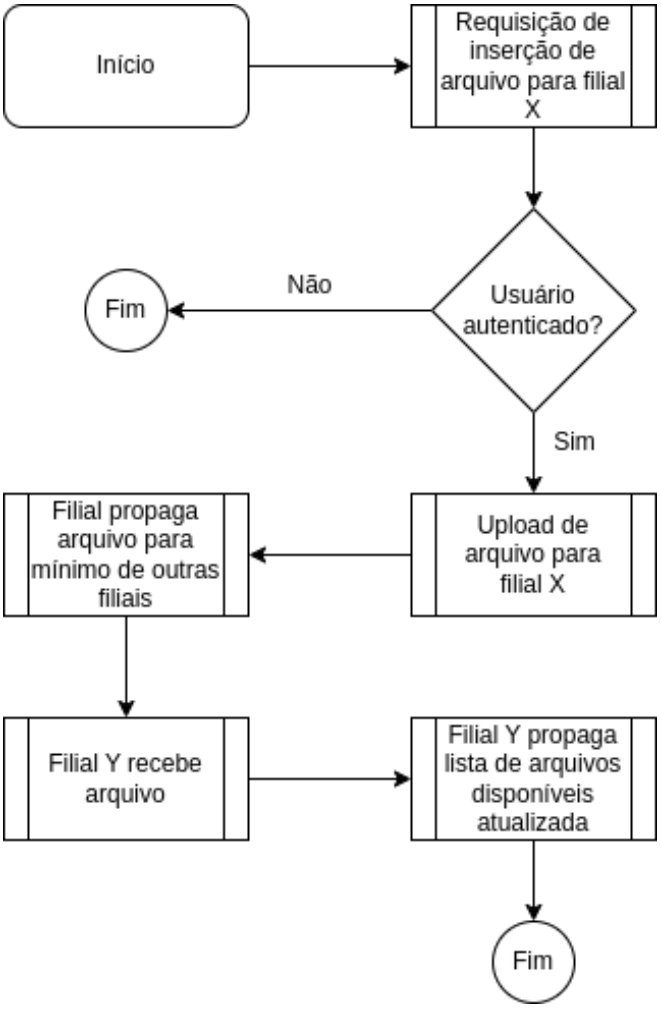


Figura 2: Exemplo do processo de criação de um arquivo

3.2.3 Download de arquivos

Dentro da arquitetura Peer-to-peer, todos dispositivos estão conectados entre si, devido a isso, é feito o compartilhamento do arquivo por partes.

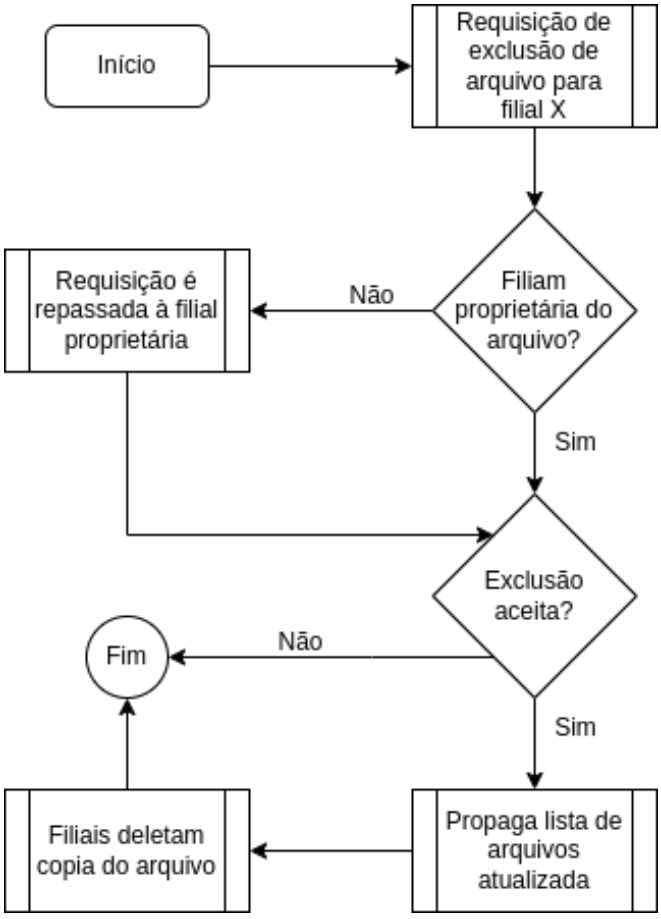


Figura 3: Exemplo do processo de download de arquivos

No sistema, há a possibilidade de mais de um dispositivo possuir um arquivo. Quando um dos dispositivos em questão deseja este arquivo, deve verificar em seu banco quais nós possuem o arquivo e solicitar para os mesmos. Após isso, os nós solicitados irão iniciar o envio do arquivo de forma particionada e todos ao mesmo tempo. Este método irá facilitar o envio, deixando mais rápido e fazendo com que o dispositivo destinatário consiga montar o arquivo final de acordo com que as partes vão chegando.

3.2.4 Atualização de arquivos

A atualização de um arquivo é feita também através do endpoint PUT /files/fileId. O procedimento é similar ao de inserção de arquivo, a diferença é que deve-se passar o ID do arquivo como path param, e o arquivo é anexado no request da interface, no Swagger.

Se o arquivo a ser atualizado existe apenas no nó o qual está fazendo o upload, então o arquivo é atualizado localmente. Mas se o arquivo existir em outro nó, então é feito o envio desse arquivo para o nó o qual já contém esse arquivo para que seja feita a atualização do arquivo existente.

3.2.5 Exclusão de arquivos

Quando há a exclusão de um arquivo na rede, como há mais de uma cópia em diferentes nós, é necessário ocorrer a deleção em todos nós. Para isto, é necessário que o nó que iniciou a exclusão verifique quais outros nós possuem este arquivo e envie um aviso para exclusão.

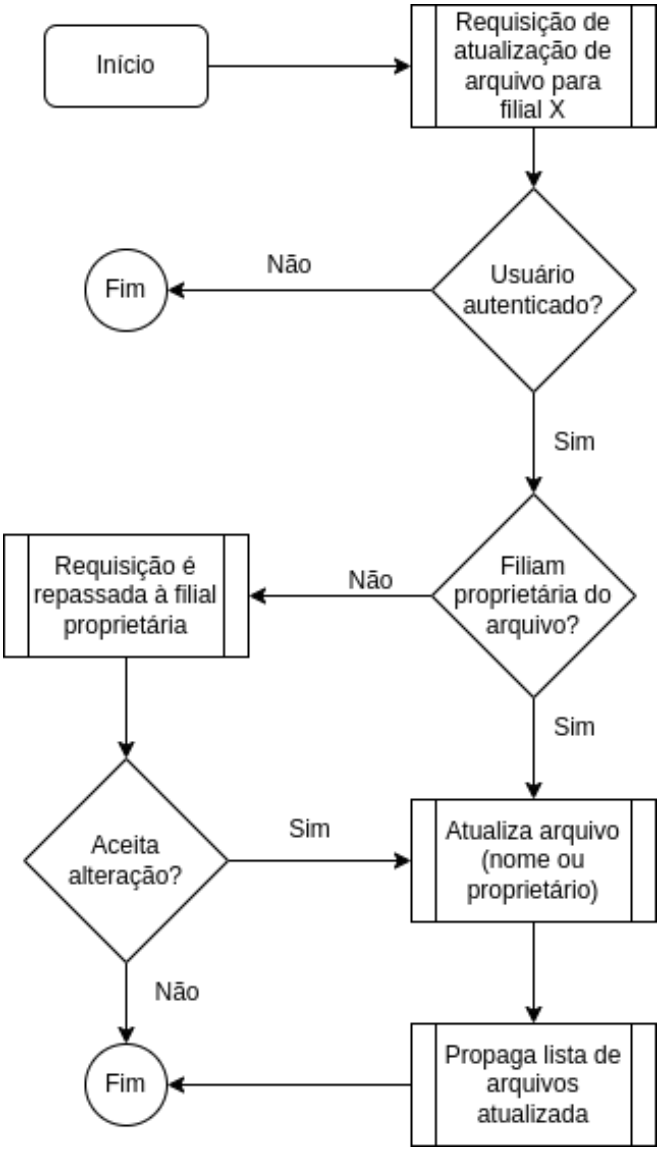


Figura 4: Exemplo do processo de atualização de arquivos

3.2.6 Indisponibilidade planejada de um Nó

Quando um nó de uma filial será desativado, seja temporariamente para manutenção do servidor local por exemplo, ou quando uma filial for de fato desativada, o sistema passa por um processo para garantir a integridade dos arquivos e garantir que os mesmos ainda continuarão disponíveis.

3.2.7 Indisponibilidade não planejada de um Nó

Quando um nó se torna inativo por conta de uma indisponibilidade da rede, os nós adjacentes serão capazes de detectar essa anomalia por meio da tentativa de comunicação. Caso o nó em questão não consiga contato ele repassa o status ao outros nós do sistema.

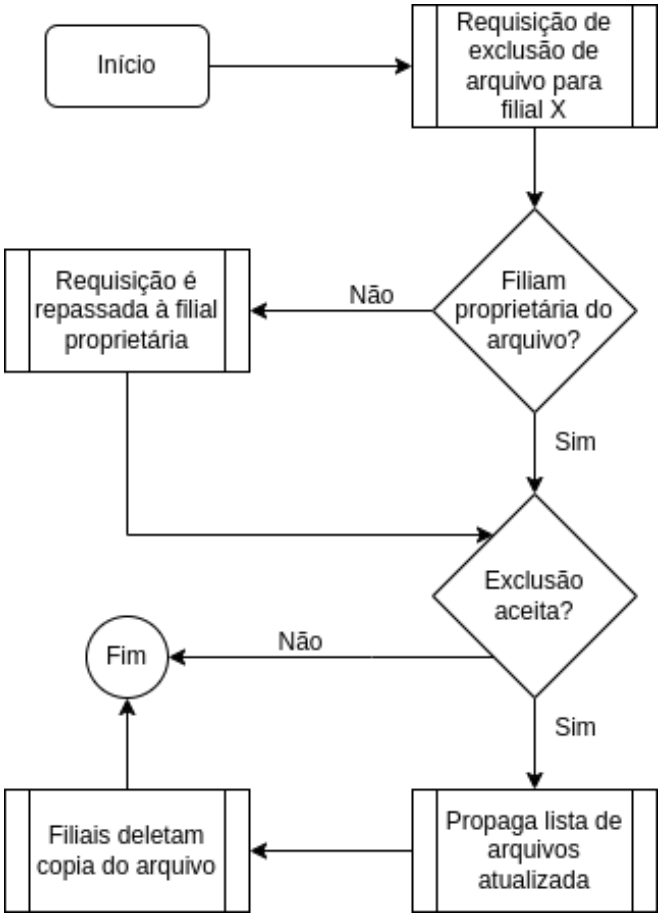


Figura 5: Exemplo do processo de exclusão de arquivos

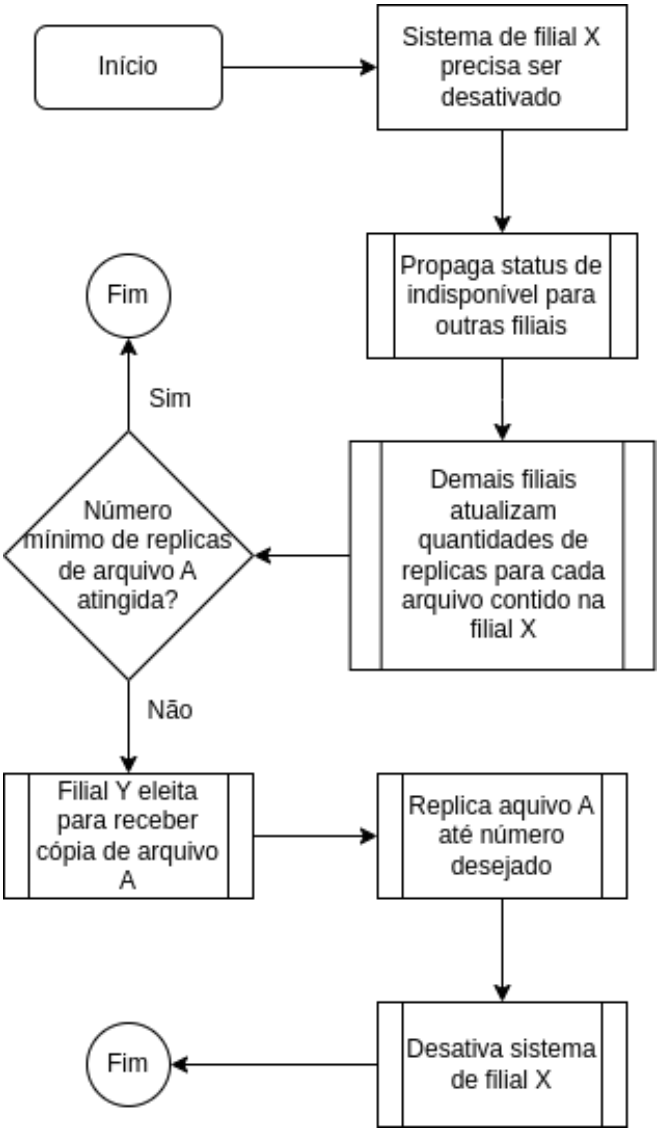


Figura 6: Exemplo do processo de indisponibilidade planejada de um Nó

3.2.8 Compartilhamento seletivo dos arquivos

Dentro de uma empresa, muitas vezes é necessário certo nível de sigilo, para isso, há a necessidade de implementação de uma blacklist.

Para adicionar um nó na blacklist usa-se o endpoint POST /files/blacklist , e então deve-se passar como parâmetro no body o id do arquivo e o id do usuário o qual estará restrito de visualizar este arquivo.

Pode-se também remover algum usuário da blacklist de um tal arquivo através do endpoint DELETE /files/blacklist/-fileId.

Conforme já contemplado na estrutura do banco de dados, a tabela de relacionamento entre arquivos e Nós também implementa o atributo booleano de is_blacklisted para cada registro individual desse relacionamento.

3.2.9 Sincronização do banco de dados

Em vários momentos citamos a propagação de alguma alteração de um nó para a rede. O problema é que por se tratar de uma rede com diversos nós, cada um contendo sua própria cópia do banco de dados, precisamos pensar em como garantir a integridade dos dados, de for assíncrona. A solução está ilustrada e ela se aproveita da natureza da aplicação de usar somente valores fixos, sem operações matemática, por exemplo, o que garante que, no pior caso, de transações não chegarem de forma ordenadas, contanto que elas sejam reexecutadas de forma ordenada, todos os bancos estarão corretamente sincronizados.

3.3 Segurança

Um sistema distribuído de compartilhamento de mídias geralmente implementa uma série de medidas de segurança para garantir a integridade e autenticidade das informações compartilhadas. Uma das principais medidas é a validação da aplicação, que verifica se a aplicação não foi alterada de forma maliciosa ou comprometida. Isso pode ser feito através de assinaturas digitais ou hashes criptográficos, onde a aplicação é assinada com uma chave privada e a assinatura é verificada com uma chave pública confiável. Dessa forma, qualquer modificação não autorizada na aplicação seria detectada, protegendo contra ameaças como injeção de código malicioso ou exploração de vulnerabilidades.

Além disso, um sistema distribuído de compartilhamento de mídias pode implementar a autenticação e autorização adequadas para garantir que apenas usuários autorizados tenham acesso aos recursos. Isso envolve o uso de técnicas como autenticação de dois fatores, onde além de uma senha, é exigido

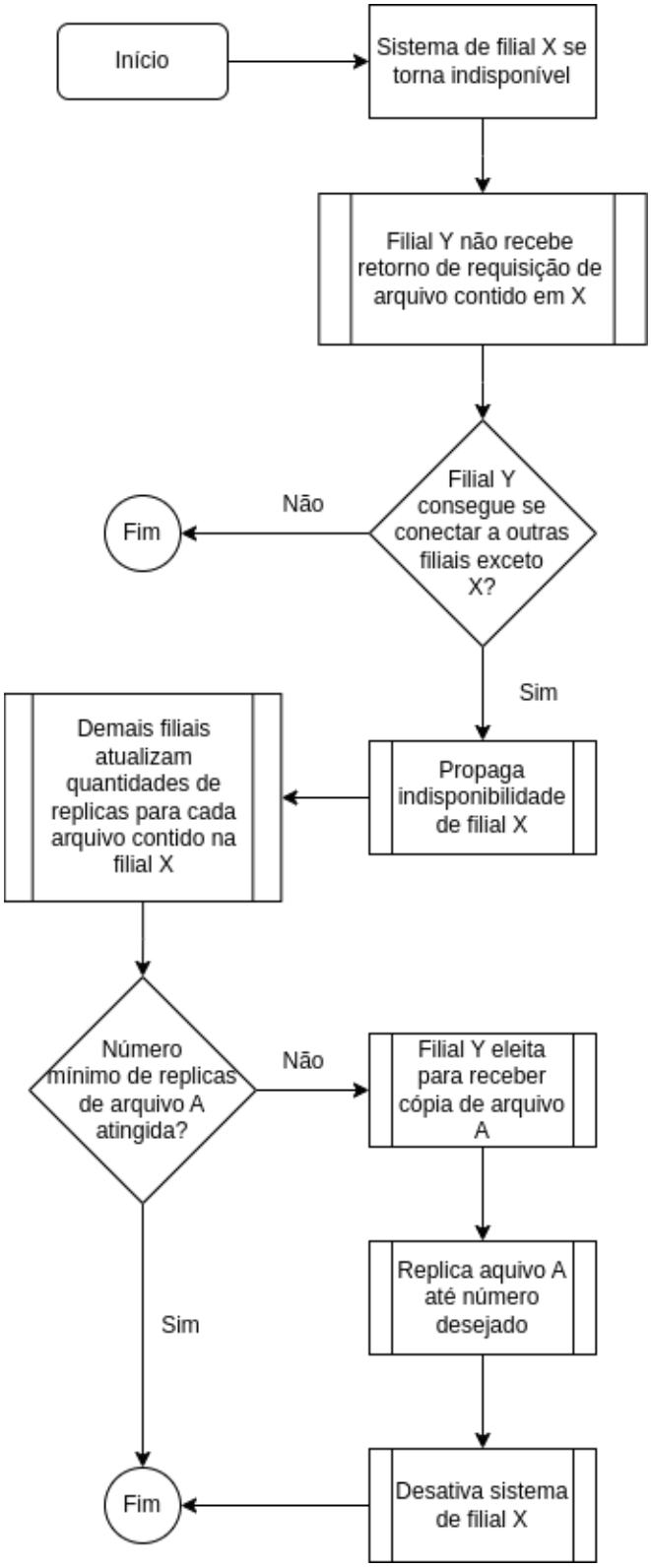


Figura 7: Exemplo do processo de indisponibilidade não planejada de um Nó

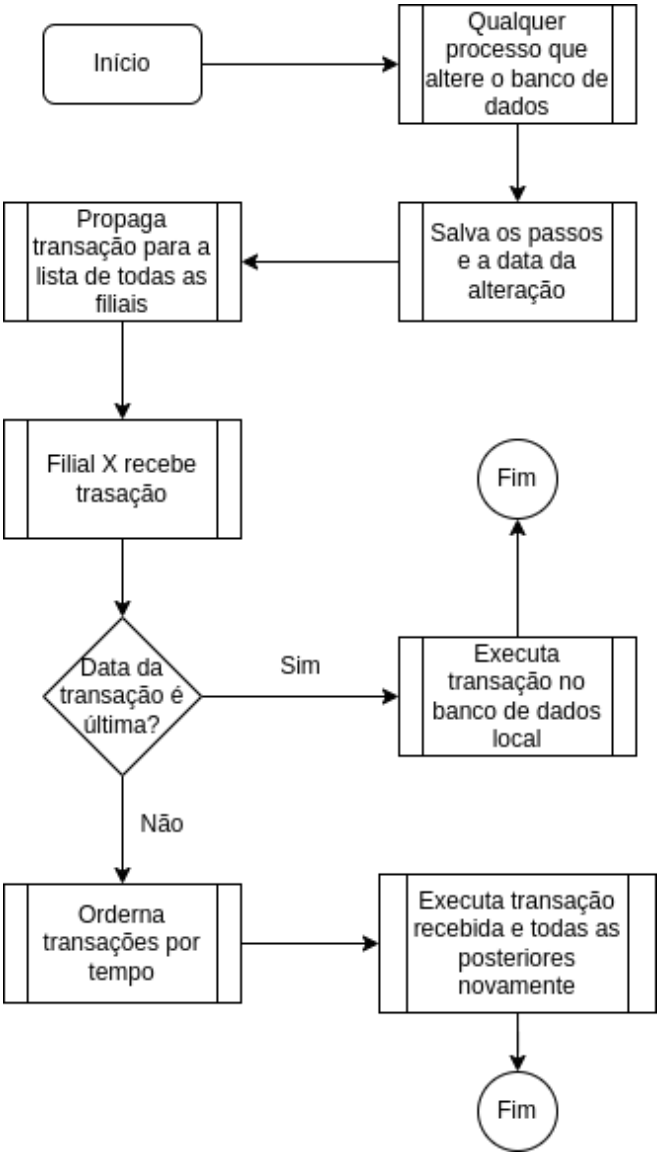


Figura 8: Exemplo do processo de sincronização do banco de dados

um código adicional enviado para o dispositivo do usuário. Também são utilizados protocolos seguros de comunicação, como o SSL/TLS, para criptografar as transações e proteger os dados durante a transferência. O sistema também pode aplicar restrições de acesso com base em níveis de permissão, garantindo que cada usuário tenha acesso apenas aos recursos e funcionalidades apropriados de acordo com seu papel ou privilégios designados. Essas medidas combinadas ajudam a garantir a segurança e a confidencialidade dos dados em um ambiente distribuído de compartilhamento de mídias.

3.4 Implantação

A implementação do projeto foi feita com Python, Docker e SQLite devido à facilidade de uso e ao conhecimento prévio do grupo em relação a essas tecnologias.

Python é uma linguagem de programação versátil e popular, com uma vasta gama de bibliotecas disponíveis, o que facilita o desenvolvimento de aplicativos distribuídos. Sua sintaxe clara e legível torna o código mais fácil de ser compreendido e mantido, além de oferecer suporte para trabalhar com redes e protocolos de comunicação. Além disso, o framework FastAPI foi implementado fazendo um api para comunicação com o banco de dados local, além de gerar a documentação interativa que serve como interface, o Swagger.

O Docker é uma plataforma de contêiner que permite empacotar e distribuir aplicativos de forma consistente em diferentes ambientes. Com o Docker, é possível criar contêineres

isolados que contêm todas as dependências necessárias para executar o sistema de compartilhamento de mídias. Isso torna a implantação e o gerenciamento da infraestrutura mais simples, uma vez que os contêineres podem ser executados em qualquer máquina que possua o Docker instalado.

O SQLite é um banco de dados leve e incorporado, amplamente utilizado em aplicativos de pequeno e médio porte. Ele é escrito em linguagem C e oferece uma interface simples para a manipulação de dados em bancos de dados relacionais. O uso do SQLite é vantajoso quando se trabalha com uma quantidade relativamente pequena de dados em um ambiente distribuído, e sua implementação é fácil de ser compreendida e integrada a aplicativos Python.

3.4.1 Interface do Sistema

Para controlar o sistema, será criada uma interface de API usando o Swagger, que disponibilizará endpoints de uma API, a qual será responsável pela execução do sistema.

1. POST /register : Esse endpoint será usado quando um novo nó quiser entrar na rede, ele precisará passar o IP de um nó já membro da rede como parâmetro no body para que a interface identifique a rede a qual o usuário deseja entrar. O body desse endpoint deverá conter o parâmetro "ip", que será o IP de algum nó já conectado a alguma rede existente.

2. POST /unregister : Esse endpoint será usado quando um novo nó quiser sair permanentemente da rede na rede. Esse endpoint não terá um body.

3. POST /connect : Esse endpoint será usado quando um nó, já membro da rede, quiser se conectar a rede. Quando esse endpoint é invocado, o nó o qual está se conectando envia uma requisição a todos os outros nós membros da rede, para notificar que esse nó agora está conectado, então o parâmetro "connected"do registro do nó o qual está se conectando é é alterado para "true"em todos os outros nós da rede. Esse endpoint não terá um body.

4. POST /disconnect : Esse endpoint será usado quando um nó, já membro da rede, quiser se desconectar a rede. Quando esse endpoint é invocado, o nó o qual está se desconectando envia uma requisição a todos os outros nós membros da rede, para notificar que esse nó agora está desconectado, então o parâmetro "connected"do registro do nó o qual está se conectando é é alterado para "false"em todos os outros nós da rede. Esse endpoint não terá um body.

5. POST /files : Nesse endpoint o usuário poderá adicionar novos arquivos a rede. Haverá um attachment do arquivo nesse endpoint, na interface do Swagger, a qual o usuário poderá selecionar o arquivo a ser inserido na rede. Após isso, a API irá notificar todos os outros nós que esse nó agora possui um novo arquivo, e o arquivo será salvo no diretório determinado pelo sistema localmente. Esse endpoint não terá um body.

6. GET /files: Esse endpoint é usado para o usuário visualizar os arquivos que estão disponíveis na rede. Quando essa interface é acionada, a API retorna todos os arquivos de cada nó e os coloca em uma lista única e retorna essa informação para o usuário. O response desse endpoint irá verificar qual Nó fez a requisição, e se esse nó estiver na blacklist com algum arquivo, esse arquivo então não será retornado nessa lista. Esse endpoint não terá um body.

7. PUT /files/fileId: Essa interface é responsável por atualizar um arquivo já existente na rede. O usuário necessitará do ID do arquivo para atualiza-lo, o qual o usuário terá acesso através do endpoint GET /files. Tendo esse ID, o usuário o inserirá no PUT como parâmetro e também irá fazer o upload do novo arquivo através da interface.

8. DELETE /files/fileId: Quando se quiser apagar um arquivo da rede basta acionar esse endpoint, neste caso basta passar o ID do arquivo e então o mesmo será apagado da rede. Esse endpoint não terá um body.

9. POST /files/blacklist: Esse endpoint será responsável por adicionar arquivos na blacklist. O body desse endpoint deverá conter os seguintes parâmetros: id do arquivo e usuário. Ou seja, o usuário poderá adicionar arquivos na blacklist um a um, enviando o usuário e arquivo os quais estarão restritos de tal visualização.

10. GET /files/blacklist: Através desse endpoint será possível ler todos os arquivos e usuários que estarão na blacklist. Caso o usuário que tenha feito a requisição esteja na blacklist ele não vai ter no retorno da requisição o registro referente ao seu usuário.

11. PUT /files/blacklist/fileId: Esse endpoint será responsável por modificar arquivos na blacklist. O body desse endpoint deverá conter o seguinte parâmetro: "usuário".

12. DELETE /files/blacklist/fileId: Neste caso esse endpoint irá remover algum arquivo da blacklist. Esse endpoint não terá body.

3.4.2 Banco de dados

O banco de dados contempla o Nós do sistema, os arquivos compartilhados na rede e uma tabela de relacionamento entre as duas para salvar algumas informações como a propriedade de cada arquivo e a blacklist.

4 Resultados

Em relacão aos resultados obtidos, obtivemos êxito no objetivo do trabalho sobre a questão de compartilhamento de arquivo entre dois peers. A grande questão foi que de acordo com o problema descrito e o projeto do sistema, existiam muitas funcões automaticas, como a atualizacão de arquivos nos nós, notificacões automáticas de conexão e desconexão, porém, não foi possível implementar estas funcionalidades devido ao nível de complexidade do problema.

Na questão de arquitetura de rede, o projeto inicial seria todos nós estarem ligados uns com os outros, formando uma rede gigantesca de nós. Durante análise do problema e estudos de possíveis solucões, notamos que seria muito difícil gerar IPs publicos para todos nós da rede. Outra provável solucão levantada seria abrir a porta do roteador e mapear para que fosse possível acessar os nós, porém, é possível apenas mapear um

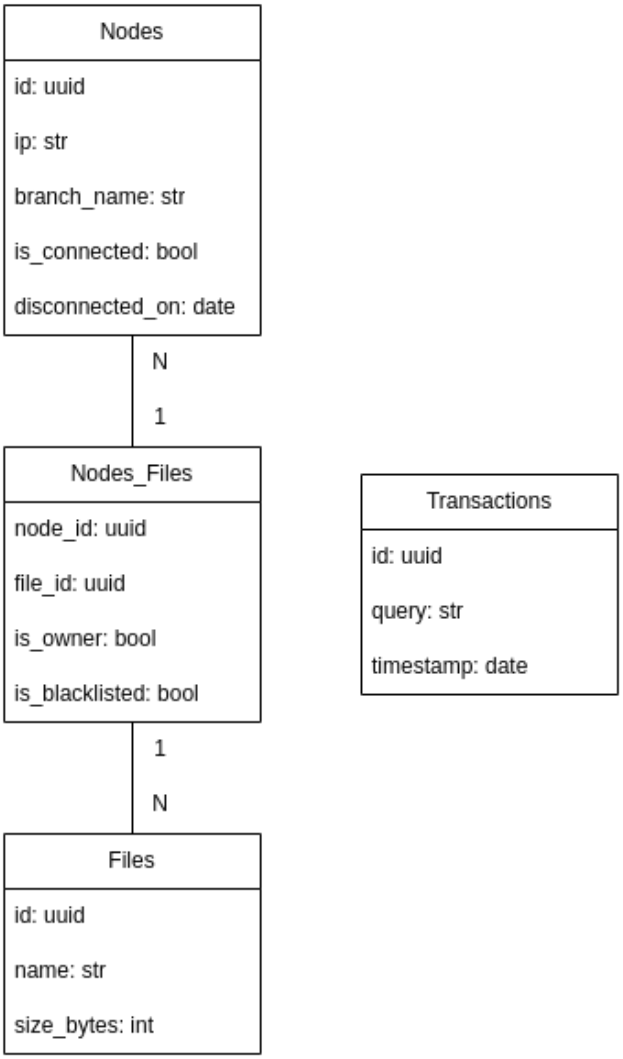


Figura 9: Esquema do banco de dados contemplando as entidades do sistema.

nó por porta, o que ficaria inviável de abrir tantas portas em um roteador. A solucão encontrada e que funcionou melhor foi expor apenas um host para a rede externa e este fazendo o trabalho de um roteador de borda, ligando a sub-rede da filial da agência com as demais filiais.

Sobre a aplicacão que será usada para fazer todas comunicacões, no começo a ideia era que cada host rodasse a propria aplicação de comunicação, porém, em análise da situação foi percebido que seria possível apenas o host exposto rodar e os demais hosts da sub-rede apenas consumir a aplicação disponibilizada, conseguindo da mesma forma realizar a comunicação com as outras filiais.

Sobre a infraestrutura que obtivemos como resultado, cada host que será exposto trabalhará com 3 portas abertas, duas portas para fazer a comunicacão com o mundo externo, enviar e receber arquivos e a porta 1337 que será usada para possibilitar o acesso dos outros hosts da sub-rede à aplicação.

5 Conclusão

Em conclusão, o processo de planejamento do nosso projeto foi desafiador, mas extremamente enriquecedor. Identificamos as principais dificuldades, como a sincronização dos nós, a integridade dos dados e a definição de casos de uso relevantes para a empresa. No entanto, com dedicação, colaboração e pesquisa, conseguimos encontrar soluções eficazes para esses desafios. Nos testes realizados, concluímos que o resultado encontrado foi favorável, já que obtivemos êxito no compartilhamento de arquivos.

Porém, mesmo obtendo resultado favorável, em trabalhos futuros serão necessários a melhoras das tecnologias implantadas neste trabalho. No cenário ideal, os computadores das filiais deveriam estar todos interligados entre si, sem sub-redes por filial, porém concluímos que é algo custoso em relação a configuração dos roteadores de borda e portas em cada dispositivo, além de gerar uma brecha de segurança em relação a ataques à rede.

Mesmo a forma de implementação do peer-to-peer apresentado neste trabalho não sendo a mais comum, foi a melhor solução encontrada comparada a outros serviços. Concluímos que usar cloud services para o problema apresentado seria muito custoso e não seguro em relação a sigilo dos arquivos, já que qualquer um que conseguisse invadir o servidor cloud teria acesso. Outra solução levantada seria o uso de infraestrutura local com armazenamento próprio em servidor centralizado, o que aumentaria o custo também e necessitaria de lugar específico, com sala refrigerada e todo um aparato de backup e segurança para o servidor.

6 Divisão do trabalho

- **HENRIQUE:** Ficou responsável pela implementação do protótipo, participou das reuniões de planejamento e por descrever boa parte do tópico 3, Método.
- **ANTONIO:** Foi responsável por introduzir o assunto no artigo, de forma contextualizada, além de entender a descrição das tecnologias usadas e inseri-las no tópico de referencial e também foi responsável pela elaboração dos slides.
- **MARCELO:** Responsável por analisar os resultados obtidos, de forma que fosse possível tirar uma conclusão em cima de todo estudo realizado, além de participar de todo planejamento de pesquisa e projeto de implementacão.

Referências

[art] Um sistema de compartilhamento de arquivos entre grupos de pessoas, gian carlo salvati, 2014). Disponível em: https://repositorio.ufsc.br/bitstream/handle/123456789/183801/TCC-impressao.pdf?sequence=-1.