PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

Safe Vault

Sistema para proteção de pastas e arquivos

Thiago Lages de Alencar

PROJETO FINAL DE GRADUAÇÃO

CENTRO TÉCNICO CIENTÍFICO - CTC

DEPARTAMENTO DE INFORMÁTICA

Curso de Graduação em Ciência da Computação

Rio de Janeiro, Julho de 2019

Thiago Lages de Alencar

Safe Vault

Sistema para proteção de pastas e arquivos

Relatório de Projeto Final 2, apresentado para **Bacharelado em Ciência da Computação** da PUC-Rio como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Anderson Oliveira da Silva.

Rio de Janeiro

Julho de 2019.

Resumo

Lages de Alencar, Thiago. da Silva, Anderson Oliveira. **Safe Vault – Sistema para proteção de pastas e arquivos**. Rio de Janeiro, 2019, 51p. Relatório de Projeto Final 2. Pontifícia Universidade Católica do Rio de Janeiro.

Este projeto final implementa uma aplicação que permite armazenar arquivos localmente ou em serviços de armazenamento (Google Drive, Dropbox, Onedrive, etc), provendo a segurança e a privacidade das pastas e dos arquivos do usuário. O sistema utilizará protocolos de segurança e algoritmos de criptografia para controlar a integridade, confidencialidade e autenticidade dos arquivos. A aplicação também é extensível para diversos provedores de serviços de armazenamentos e diferentes plataformas (Windows, Android, Mac, etc).

Palavras-chave

Criptografia, Segurança da Informação, Serviço de armazenamento, Plataformas.

Abstract

Lages de Alencar, Thiago. da Silva, Anderson Oliveira. **Safe Vault – System for protection of folders and files**. Rio de Janeiro, 2019. 51p. Relatório de Projeto Final 2. Pontifícia Universidade Católica do Rio de Janeiro.

This final project implements an application that allows the user to store files locally or on storages services (Google Drive, Dropbox, Onedrive, etc), providing the security and privacy of his/her folders and files. The system will use security protocols and encryption algorithms to control integrity, confidentiality and authenticity of files. The application is also extensible to multiple storage services and different platforms (Windows, Android, Mac, etc).

Keywords

Cryptography, Information Security, File Hosting Service, Platforms.

1. Introdução

1.1. Motivação

Serviços de armazenamento como *Google Drive*[1], *Dropbox*[2] e *Onedrive*[3], permitem ao usuário o conforto de acesso remoto às suas informações, sem se preocupar com a manutenção dos recursos físicos. Porém, existem usuários que se recusam a armazenar informações sensíveis nesses provedores, pois consideram a proteção insuficiente. Portanto, observa-se que a confidencialidade é uma grande preocupação quando se fala de armazenamento na Internet[4][5].

Armazenamento na nuvem aumenta o risco de comprometimento das informações de diversas maneiras, como por exemplo: vazamento de informações, furto de conta, fraudes, uso indevido, etc[5][6]. Esses riscos podem ser reduzidos se os dados forem protegidos utilizando protocolos de segurança. Tais protocolos dão ao usuário controle sobre quem pode acessar os dados, e isso garante a proteção das informações do usuário.

Diante desse cenário, a aplicação proposta visa a funcionar como uma ferramenta para o usuário armazenar seus arquivos de forma segura em serviços de armazenamento ou no próprio dispositivo que está utilizando. A ideia é que a aplicação seja apenas uma ponte para o usuário garantir a segurança de seus dados nos serviços de armazenamento.

Com o intuito de respeitar a privacidade do usuário, nenhuma fiscalização é realizada nos dados durante o armazenamento no serviço. Durante a recuperação dos dados, a aplicação fará a validação dos recursos de segurança para identificar qualquer invasão na privacidade dos dados.

1.2. Problema

Grande parte dos serviços de armazenamento oferecem o armazenamento não criptografado, o que implica no usuário confiar que suas informações terão a confidencialidade respeitada. Não existe certeza dessa confidencialidade, pois se

alguém obter a credencial de acesso do usuário, toda a informação no provedor do serviço estará comprometida.

Provedores e empresas de terceiros que garantem essa segurança, costumam armazenar a chave que protege as informações do usuário[7]. Isso delega o controle de acesso aos dados para os provedores e empresas, não resolvendo o problema de confidencialidade[4].

1.3. Ambiente de Desenvolvimento

O desenvolvimento do software ocorreu em um desktop com o sistema operacional *Ubuntu*[9]. A linguagem de programação escolhida foi *Java*, versão SE 11. Foi empregado provedores criptográficos disponibilizados pela *Java Cryptography Architecture* (JCA)[10]. A API do *Google Drive*[11][12] foi utilizada para se comunicar com o serviço de armazenamento do *Google*. A IDE utilizada foi *Eclipse*[13], versão *Eclipse Photon*.

A adoção do Java se deve ao fato de haver conhecimento dessa linguagem e da JCA, ambas estudadas em disciplinas do curso de graduação, e por causa da sua estabilidade e disponibilidade em diversas plataformas (*Windows*, *Linux*, *Mac*, *Android*, etc). A versão usada é a mais recente e estável que satisfez os requisitos do projeto.

1.4. Organização do Relatório

Este relatório está organizado de forma ser autocontido, ou seja, aborda todos os assuntos e conhecimentos necessários para o desenvolvimento da aplicação proposta.

Nesse sentido, o capítulo 2 aborda os conceitos criptográficos relevantes para a segurança da informação. O capítulo 3 descreve o funcionamento da Google Drive API, usada para manipular com o sistema de arquivos da Google. No capítulo 4, são apresentadas as camadas do framework desenvolvido, requisitos funcionais, casos de uso e matriz de rastreabilidade. O capítulo 5 apresenta o código e diagramas de classes para o entendimento da aplicação. E, para encerrar, o capítulo 6 apresenta a conclusão e as metas futuras para o projeto.

2. Segurança da Informação

Este capítulo apresenta os conceitos básicos de segurança da informação que são utilizados pela aplicação para proteger a integridade, a autenticidade e o sigilo dos arquivos e pastas do usuário.

2.1. Integridade

Integridade envolve proteger a informação contra alterações sem permissão explícita do proprietário. O estado da informação quando resgatada, deve ser igual ao estado quando gerada[24]. Ou seja, durante todo o percurso a informação pode sofrer alterações, porém durante o resgate, a informação deve se mostrar íntegra.

2.1.1. Função Hash Criptográfica

Funções hash criptográficas são funções hash que se mostram adequadas para o uso em criptografia. São utilizadas para mapeamento de dados, recebem um dado (texto) como entrada e retornam um valor (hash).

Toda a execução sobre a mesma entrada, deve gerar o mesmo valor hash como saída. Desta maneira, é possível relacionar um texto específico a um hash.

A probabilidade de dois textos diferentes gerarem o mesmo hash deve ser baixa. Ou seja, as chances de se criar um texto com o mesmo conteúdo e mesmo hash deve ser praticamente irrealizável.

A partir do valor hash produzido pela função hash criptográfica, é praticamente inviável revertê-lo para o texto passado como entrada. Isto é, mesmo que se obtenha o hash, não é possível descobrir a entrada utilizada na função.

Outra característica do hash é o seu efeito avalanche. Uma mínima alteração no texto de entrada faz com que o hash seja alterado de forma que não seja possível identificar nenhuma relação com o hash antes da alteração. Alterar um único bit em um texto, produz um hash significativamente diferente ao anterior.

2.1.2. Resumo de Mensagem (Message Digest)

Resumo de mensagem (digest) é o valor obtido por uma função hash criptográfica ao ser executada sobre um texto plano[5]. Digest são utilizados como uma identificação única para uma mensagem ou documento específico.

A ideia é utilizar algoritmos de resumo de mensagem para verificar a integridade das informações. Se um texto possuir um bit alterado, o digest produzido será diferente do anterior, com isso confirmando adulteração do texto.

O algoritmo de resumo de mensagem adotado neste projeto foi o SHA-256, embora qualquer algoritmo SHA-2 acima de 256 bits providenciasse a segurança necessária, foi escolhido aquele com maior chance de compatibilidade com diversas máquinas[26][27].

2.2.Confidencialidade

Confidencialidade (sigilo) envolve garantir que apenas as pessoas às quais a informação é destinada consigam compreendê-la[5]. A informação é alterada de forma que apenas os destinatários deveriam conseguir resgatar a informação ao estado original. Algoritmos de criptografia são utilizado para mascarar a informação original, através de cifragem. A segurança da informação depende da garantia de segredo da chave utilizada no algoritmo.

2.2.1. Criptografia Simétrica

Criptografia simétrica implica utilizar algoritmos de criptografia onde a chave criptográfica para cifrar o texto plano é a mesma para decifrar o texto cifrado. A chave representa um segredo compartilhado entre a origem e o destino da informação (um ou mais destinos). O maior problema dessa criptografia está em compartilhar essa chave entre os participantes[28].

O algoritmo de criptografia simétrica escolhido para este projeto foi o AES, o mais utilizado globalmente pela segurança e desempenho que proporciona. Durante o projeto, as chaves simétricas serão geradas utilizando uma semente que é criada com a utilização do algoritmo SHA1, o que torna possível reconstruir a semente.

2.2.2. Criptografia Assimétrica

Criptografia assimétrica lida com algoritmos que utilizam um par de chaves (duas chaves) matematicamente relacionadas. Um texto cifrado com uma dessas chaves só pode ter a informação resgatada pela chave par dela. Essas chaves são conhecidas como *chave privada* e *chave pública*[5].

A chave privada deve ser conhecimento único da pessoa que a criou, enquanto a chave pública deve ser distribuída para outras pessoas. Isso permite que qualquer pessoa cifre uma informação com a chave pública de alguém e que apenas o criador do par de chaves consiga decifrar a cifra produzida com a sua chave privada. Por outro lado, qualquer pessoa pode decifrar o conteúdo cifrado com a chave privada de alguém usado a respectiva chave pública do cifrador.

O processo de cifrar com algoritmo de criptografia assimétrica costuma ser bem custoso pois requer muito processamento e tempo comparado com a criptografia simétrica[5]. O texto cifrado gerado pelo processo costuma ter o tamanho bem maior que o texto plano. Esses dois problemas fazem com que não seja uma boa ideia utilizar essa criptografia para a cifragem do texto plano de um arquivo ou documento já que a massa de dados pode ser muito grande. Em geral, esse algoritmo é usado para criptografar informações sensíveis que são pequenas, como, por exemplo, chaves criptográficas simétricas.

O projeto adotou o RSA como algoritmo de criptografia assimétrica neste projeto por ser um dos melhores atualmente.

2.2.3. Envelope Digital

Envelope Digital é um esquema que utiliza chave pública e privada para trocar chaves simétricas entre entidades. Isto permite que utilizemos os benefícios de ambos os métodos de criptografia.

Neste esquema, o dado a ser protegido é cifrado com a chave simétrica para garantir agilidade na criptografia. Por sua vez, a chave simétrica é cifrada utilizando a chave pública da entidade ao qual o dado protegido se destina. Dessa forma, apenas a entidade destino consegue recuperar a chave simétrica e, consequentemente, decriptar o dado protegido.

2.3. Disponibilidade

Disponibilidade visa a manter a informação disponível para uso sempre que houver necessidade dela. O serviço de armazenamento compartilha desse princípio básico da segurança da informação, por isso não é preciso que ele seja implementado durante este projeto. Serviços como Google Drive, já buscam manter os arquivos do usuário acessíveis a qualquer momento.

2.4. Autenticidade

Autenticidade consiste em identificar corretamente um usuário ou máquina[5]. É uma maneira de medir o grau de confiança de que a origem da informação é a mesma que alega ser[25]. É necessário verificar a autenticidade em todo o processo de identificação e transferência de informação, pois qualquer entidade maliciosa pode tentar fingir ser a entidade que se deseja contactar ou interagir.

2.4.1. Assinatura Digital

Assinatura digital é um esquema matemático para verificar a autenticidade de um documento ou uma mensagem. Um documento que passa pelo processo de verificação da assinatura digital com sucesso, fornece ao receptor grande confiança que o documento veio pelo remetente e não foi adulterado durante o percurso até o receptor. Esse esquema, segundo o padrão RSA, criptografa o resumo de mensagem de um documento com a chave privada do signatário, o que produz a assinatura digital do documento. Por sua vez, essa assinatura digital pode ser decriptada com a chave pública do signatário, o que possibilita recuperar o resumo de mensagem original e confrontá-lo com o resumo de mensagem corrente do documento. Se os resumos comparados são idênticos, o documento é considerado íntegro e autêntico.

3. Google Drive API

A Google Drive API (Drive API) permite que a aplicação interaja com o serviço de armazenamento, oferecendo serviços como download de arquivos, upload de arquivos, busca por arquivos e pastas, etc. Porém, certas interações estão limitadas a arquivos e pastas que não gerados pelos aplicativos Google, ou seja, não é possível fazer download ou upload de um arquivo do Google Docs, Google Sheet, etc[11].

3.1. OAuth 2.0

Toda requisição enviada pela aplicação para a API, precisa incluir um token de autenticação e esse token identifica a sua aplicação para o Google. O único protocolo que a API suporta é OAuth 2.0.

Para se obter o token é necessário passar ao Google as credenciais OAuth 2.0, que possuem informações como *client ID* e *client secret*. Essa credencial pode ser obtida na Google API Console[17]. Para o Google fornecer o token de acesso, o usuário deve selecionar uma conta do Google e consentir os acessos permitidos (ler, escrever, acessar pasta, etc).

3.2. Arquivos e Pastas

A Drive API representa arquivos armazenados no Google Drive como do tipo File. Cada arquivo armazenado possui diversos atributos e métodos para facilitar a manipulação dele. Pastas são tratadas como do tipo File, porém possuem o campo MIME como pasta. Exemplos de campos:

- id: string
- name: string
- mimeType: string
- trashed: boolean
- parents: collection of strings

Todos os arquivos File possuem o atributo *parents*, uma coleção de ids (identificadores). Este atributo indica as pastas em quais esse arquivo está contido. É importante ressaltar que um arquivo pode estar contido em múltiplas pastas.

Cada usuário possui uma pasta raiz chamada *My Drive*, que consiste de tudo que essa pasta raiz pode ter como pai. Como o usuário é o dono primário da pasta raiz, ele tem controle sobre todos os arquivos dentro dela.

O Google Drive divide os arquivos em quatro tipos: *Blob*, *Folder*, *Shortcut*, *G Suite document*. Dentre esses, apenas dois são usados no projeto: *Blob* e *Folder*.

O tipo *G Suite document* representa arquivos de aplicações do Google (Google Sheet, Google Docs, etc). O tipo *Shortcut* representa arquivos que ligam a conteúdo armazenado em sistema de terceiros. O tipo *Folder* representa arquivos do tipo *File* que apenas possuem metadados, ou seja, não é possível interagir com seu conteúdo. O tipo *Blob* representa arquivos *File* com conteúdo do tipo texto ou binário, tais como imagens, vídeos e áudio.

Todos os arquivos do tipo *File* possuem um identificador único e metadados. Esse identificador é fixo durante o tempo de vida deste arquivo do tipo File, mesmo que o nome seja alterado. Os metadados descrevem o conteúdo do arquivo, o que inclui: nome, tipo, data de criação, data de modificação, etc.

3.3. Busca

O Google Drive não armazena as informações utilizando topologia hierárquica (topologia em árvore). Como explicado anteriormente, um arquivo pode estar contido em diversas pastas e nomes são apenas metainformações (metadados). Por causa disto, não é oferecida nenhuma espécie de busca por caminho (path). Por exemplo, suponha uma busca pelo arquivo *1121561.doc* localizado dentro da pasta *provas*, que por sua vez, está dentro da pasta *inf1416*, e que por sua vez, está dentro da pasta *disciplinas* localizada na raiz do sistema de arquivos. Embora a representação do caminho seja */disciplinas/inf1416/provas/1121561.doc*, a busca implicará em:

- Executar uma requisição por arquivos na raiz (root).
- Executar uma requisição por arquivos em disciplinas.
- Executar uma requisição por arquivos em inf1416.
- Executar uma requisição por arquivos em provas.

Para cada pasta no caminho, é necessário fazer uma nova requisição para determinar as informações do identificador do arquivo ou dos identificadores dos

pais, o que provoca um crescimento linear de requisições. O nome do arquivo apenas é considerado como parâmetro na busca quando existe certeza de que o mesmo se encontra dentro de uma pasta específica.

4. Projeto

A aplicação é separada em 3 partes: *Plugin*, *Engine* e *Interface*. Essas partes se complementam para prover a segurança da informação do usuário nos serviços de armazenamento, através de controles de integridade, autenticidade e confidencialidade.

4.1. Plugin

O Plugin é responsável por interagir com o serviço de armazenamento para realizar tarefas básicas requisitadas pela Engine. Cada Plugin é responsável por se comunicar com um dos provedores de armazenamento. Todos os Plugins executam as mesmas operações básicas sobre o serviço de armazenamento do qual é responsável. Existe a implementação de um Plugin padrão para interagir com o sistema de arquivos local na máquina do usuário. As operações básicas são:

- Criar pasta
- Listar pasta
- Deletar pasta
- Criar arquivo
- Ler arquivo
- Escrever no arquivo
- Deletar arquivo

Com essas operações, é possível reproduzir ações mais avançadas, como: renomear arquivo, copiar arquivo, mover arquivo, etc.

4.2. Engine

A Engine aplica todos os protocolos de segurança nas informações do usuário, recebe requisições das Interfaces e faz requisições aos Plugins. A Engine deve ser única para todas as plataformas, deve interagir com diversos serviços de armazenamentos (através de Plugins) e receber requisições de Interfaces de diferentes sistemas operacionais. Para garantir a segurança da informação, a Engine fará uso de algoritmos de hash (digest), padrões de assinatura digital e

algoritmos criptográficos simétricos e assimétricos. Utilizando as operações fornecidas pelos Plugins, é possível:

- Copiar pasta
- Copiar arquivo
- Mover pasta
- Mover arquivo
- Cifrar pasta
- Cifrar arquivo
- Decifrar pasta
- Decifrar arquivo
- Criar vault
- Criar pasta segura
- Listar pasta segura
- Copiar pasta para pasta segura
- Copiar arquivo para pasta segura
- Mover pasta para pasta segura
- Mover arquivo para pasta segura
- Deletar pasta segura
- Deletar arquivo seguro

4.2.1. Vault

O Vault é a pasta pela qual a Engine providência segurança sobre os arquivos e pastas. Qualquer arquivo que entre ou saia do vault, com a utilização da aplicação Safe Vault, passa por diversos protocolos de segurança para assegurar a segurança da informação. Dentre eles:

- O nome do arquivo é alterado para proteger o nome original
- É criado um container para proteger o conteúdo do arquivo

4.2.2. Container

O Container é montado utilizando os conceitos de proteção da informação descritos no capítulo 2, como envelope digital e assinatura digital. Porém, no envelope digital, é criptografado a semente da chave simétrica. Essa semente é

usada para reconstruir a chave simétrica. O procedimento de criação do container é:

- Criar semente aleatória
- Criar chave simétrica a partir da semente
- Criar o envelope digital cifrando a semente com a chave pública do usuário
- Criar a assinatura digital cifrando o hash do conteúdo do arquivo com a chave privada do usuário
- Cifrar o conteúdo do arquivo com a chave simétrica
- Criar um container contendo semente cifrada, assinatura digital e conteúdo cifrado

4.2.3. Index

Todo vault possui um arquivo index que contém informações essenciais para resgatar o arquivo original. Por exemplo, localização do arquivo, seu nome original e o tipo (arquivo/pasta). Sempre que um arquivo for retirado ou adicionado ao vault, o index é atualizado para conter essas informações do arquivo. O conteúdo do index também está protegido com a utilização de um container. Para que uma entidade maliciosa não consiga identificar o index, o nome do arquivo é gerado utilizando uma frase secreta fornecida pelo usuário.

4.3. Interface

A Interface é o meio de comunicação do usuário com a Engine. Através dela, o usuário descobre o estado das informações na nuvem e solicita ações à Engine. Todas as Interfaces de usuário, independentemente de sistema operacional, solicitam as mesmas ações de forma padronizada para a Engine.

Diferentemente da Engine e do Plugin, que seguem uma estrutura específica e ações padronizadas, a Interface dá a liberdade para o desenvolvedor apresentar as informações da maneira que desejar para o usuário. Tal que, dois desenvolvedores podem construir Interfaces muito diferentes para uma mesma plataforma.

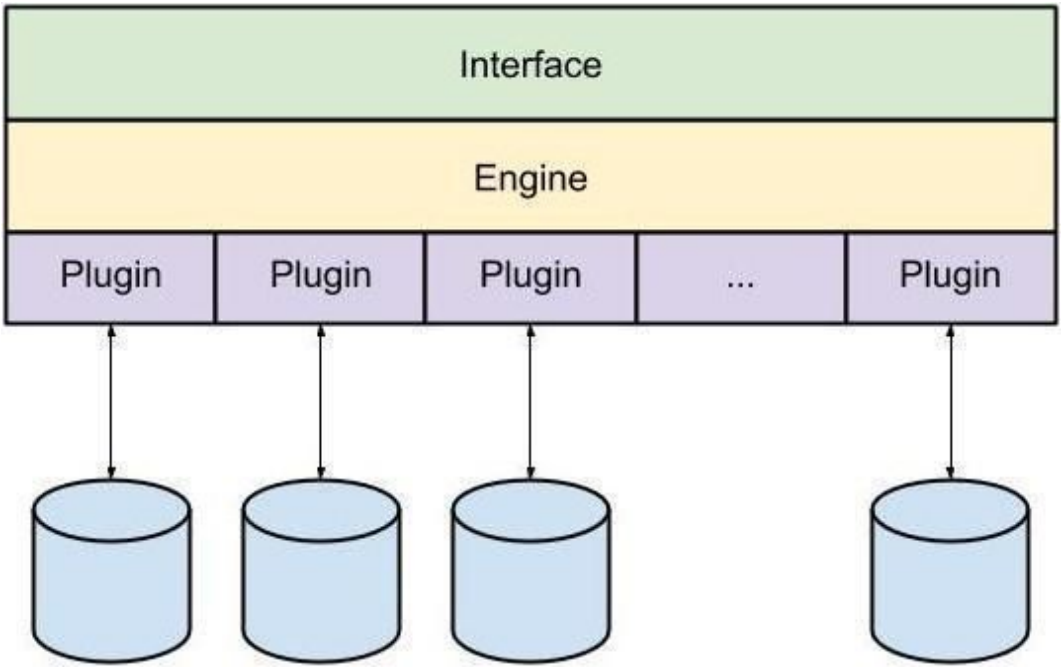

Figura: Exemplo das 3 camadas juntas.

4.4.1. Requisitos Funcionais

- RF1: Plugin deve utilizar caminho (path) para se referenciar a arquivos ou pastas no serviço de armazenamento.
- RF2: Plugin deve criar pastas na área de armazenamento.
- RF3: Plugin deve listar pastas e arquivos contidos em pastas da área de armazenamento.
- RF4: Plugin deve remover pastas da área de armazenamento.
- RF5: Plugin deve criar arquivos na área de armazenamento.
- RF6: Plugin deve obter/ler o conteúdo de arquivos da área de armazenamento.
- RF7: Plugin deve definir/escrever o conteúdo de arquivos na área de armazenamento.
- RF8: Plugin deve remover arquivos da área de armazenamento.

- RF9: Engine deve criar pastas na área de armazenamento.
- RF10: Engine deve listar pastas e arquivos contidos em pastas da área de armazenamento.
- RF11: Engine deve remover arquivos da área de armazenamento.

- RF12: Engine deve remover pastas da área de armazenamento.
- RF13: Engine deve copiar arquivos de uma área de armazenamento para outra (ou a mesma).
- RF14: Engine deve copiar pastas, e seus conteúdos, de uma área de armazenamento para outra (ou a mesma).
- RF15: Engine deve mover arquivos de uma área de armazenamento para outra (ou a mesma).
- RF16: Engine deve mover pastas, e seus conteúdos, de uma área de armazenamento para outra (ou a mesma).
- RF17: Engine deve cifrar arquivos.
- RF18: Engine deve cifrar os arquivos dentro de pastas.
- RF19: Engine deve decifrar arquivos.
- RF20: Engine deve decifrar os arquivos dentro de pastas.

- RF21: Engine deve criar o vault.
- RF22: Engine deve criar pastas seguras dentro do vault.
- RF23: Engine deve listar pastas seguras e arquivos seguros contidos em pastas seguras do vault.
- RF24: Engine deve remover arquivos seguros de dentro do vault.
- RF25: Engine deve remover pastas seguras, e seus conteúdos, de dentro do vault.
- RF26: Engine deve copiar arquivos seguros de um vault para outra área de armazenamento/vault.
- RF27: Engine deve copiar pastas seguras, e seus conteúdos, de um vault para outra área de armazenamento/vault.
- RF28: Engine deve mover arquivos seguros de um vault para outra área de armazenamento/vault.
- RF29: Engine deve mover pastas seguras, e seus conteúdos, de um vault para outra área de armazenamento/vault.

4.6. Casos de Uso

4.6.1 Plugin

4.6.1.1 Criar pasta

Nome do Caso de Uso	Criar pasta
Nível	Objetivo do Plugin
Ator principal	Plugin
Interessados e interesses	**- Plugin**: Deseja criar uma pasta vazia no caminho especificado.
Pré-condições	- Plugin consegue se comunicar com o armazenamento.
Garantia de sucesso	- Pasta vazia criada no caminho especificado.
Cenário de sucesso principal	1. Plugin recebe o caminho da nova pasta. Por exemplo, “home/downloads/newFolder”. 2. Plugin faz requisição para o armazenamento criar a pasta nesse caminho. 3. Armazenamento cria a nova pasta.
Exceções	3a. Pasta já existe. 1. Levanta uma exceção. 3b. Uma pasta do caminho não existe. 1. Levanta uma exceção.

4.6.1.2 Listar pasta

Nome do Caso de Uso	Listar pasta
Nível	Objetivo do Plugin
Ator principal	Plugin
Interessados e interesses	**- Plugin**: Deseja obter os nomes de todos os arquivos e pasta dentro um caminho especificado.
Pré-condições	- Plugin consegue se comunicar com o armazenamento.
Garantia de sucesso	- Lista dos nomes dos arquivos e pastas.
Cenário de sucesso principal	1. Plugin recebe o caminho da pasta. Por

	exemplo, “home/downloads”. 2. Plugin faz requisição para o armazenamento da lista de arquivos e pastas no caminho especificado. 3. Armazenamento retorna uma lista com os arquivos e pastas.
Exceções	3a. Não é uma pasta. 1. Levanta uma exceção. 3b. Uma pasta do caminho não existe. 1. Levanta uma exceção.

4.6.1.3 Deletar pasta

Nome do Caso de Uso	Deletar pasta
Nível	Objetivo do Plugin
Ator principal	Plugin
Interessados e interesses	**- Plugin**: Deseja deletar pasta no caminho especificado.
Pré-condições	- Plugin consegue se comunicar com o armazenamento.
Garantia de sucesso	- Pasta no caminho especificado não existe mais.
Cenário de sucesso principal	1. Plugin recebe o caminho da pasta. Por exemplo, “home/downloads/folderToDelete”. 2. Plugin faz requisição para o armazenamento deletar a pasta nesse caminho. 3. Armazenamento deleta a pasta do caminho.
Exceções	3a. Pasta não está vazia. 3b. Uma pasta do caminho não existe. 3c. Não é uma pasta.

4.6.1.4 Criar arquivo

Nome do Caso de Uso	Criar arquivo
Nível	Objetivo do Plugin
Ator principal	Plugin
Interessados e interesses	**- Plugin**: Deseja criar um arquivo vazio no caminho especificado.

Pré-condições	- Plugin consegue se comunicar com o armazenamento.
Garantia de sucesso	- Arquivo vazio criado no caminho especificado.
Cenário de sucesso principal	1. Plugin recebe o caminho do novo arquivo. Por exemplo, “home/downloads/newFile”. 2. Plugin faz requisição para o armazenamento criar um arquivo vazio nesse caminho. 3. Armazenamento cria o novo arquivo.
Exceções	3a. Arquivo já existe. 3b. Uma pasta do caminho não existe.

4.6.1.5 Ler arquivo

Nome do Caso de Uso	Ler arquivo
Nível	Objetivo do Plugin
Ator principal	Plugin
Interessados e interesses	**- Plugin**: Deseja obter os dados do arquivo no caminho especificado.
Pré-condições	- Plugin consegue se comunicar com o armazenamento.
Garantia de sucesso	- Os dados do arquivo.
Cenário de sucesso principal	1. Plugin recebe o caminho do arquivo. Por exemplo, “home/downloads/fileXYZ”. 2. Plugin faz requisição para o armazenamento dos dados dentro do arquivo no caminho especificado. 3. Armazenamento retorna os dados.
Exceções	3a. Não é um arquivo. 3b. Uma pasta do caminho não existe. 3c. O arquivo não existe.

4.6.1.6 Escrever no arquivo

Nome do Caso de Uso	Escrever no arquivo
Nível	Objetivo do Plugin
Ator principal	Plugin
Interessados e interesses	**- Plugin**: Deseja salvar dados em um arquivo

	no caminho especificado.
Pré-condições	- Plugin consegue se comunicar com o armazenamento.
Garantia de sucesso	- Os dados do arquivo.
Cenário de sucesso principal	1. Plugin recebe o caminho do arquivo. Por exemplo, “home/downloads/fileXYZ”. 2. Plugin recebe os dados a serem salvo no arquivo. 3. Plugin faz requisição para o armazenamento escrever os dados dentro do arquivo no caminho especificado. 4. Armazenamento escreve os dados.
Exceções	3a. Não é um arquivo. 3b. Uma pasta do caminho não existe. 3c. O arquivo não existe.

4.6.1.7 Deletar arquivo

Nome do Caso de Uso	Deletar arquivo
Nível	Objetivo do Plugin
Ator principal	Plugin
Interessados e interesses	**- Plugin**: Deseja deletar arquivo no caminho especificado.
Pré-condições	- Plugin consegue se comunicar com o armazenamento.
Garantia de sucesso	- Arquivo no caminho especificado não existe mais.
Cenário de sucesso principal	1. Plugin recebe o caminho do arquivo. Por exemplo, “home/downloads/fileToDelete”. 2. Plugin faz requisição para o armazenamento deletar o arquivo nesse caminho. 3. Armazenamento deleta o arquivo do caminho.
Exceções	3a. Não é um arquivo. 3b. Uma pasta do caminho não existe. 3c. O arquivo não existe.

4.6.2 Engine

4.6.2.1 Criar pasta

Nome do Caso de Uso	Criar pasta
Nível	Objetivo da Engine
Ator principal	Engine
Interessados e interesses	- **Engine**: Deseja criar uma pasta vazia dentro de uma pasta especificada.
Pré-condições	- Engine tem acesso ao Plugin responsável pelo armazenamento utilizado.
Garantia de sucesso	- Pasta vazia criada no caminho especificado.
Cenário de sucesso principal	1. Engine recebe o nome da nova pasta. 2. Engine recebe a pasta onde a nova pasta deve ser criada. 3. Engine formula o caminho da nova pasta. 4. Engine passa o caminho da nova pasta para o Plugin. 5. Plugin cria pasta. 4.6.1.1 Criar pasta

4.6.2.2 Listar pasta

Nome do Caso de Uso	Listar pasta
Nível	Objetivo da Engine
Ator principal	Engine
Interessados e interesses	- **Engine**: Deseja obter todos os arquivos e pasta dentro de uma pasta especificada.
Pré-condições	- Engine tem acesso ao Plugin responsável pelo armazenamento utilizado.
Garantia de sucesso	- Lista de arquivos e pastas.
Cenário de sucesso principal	1. Engine recebe a pasta que deseja listar. 2. Engine passa o caminho da pasta para o Plugin. 3. Plugin lista pasta. 4.6.1.2 Listar pasta

4.6.2.3 Deletar arquivo

Nome do Caso de Uso	Deletar arquivo

Nível	Objetivo da Engine
Ator principal	Engine
Interessados e interesses	**- Engine**: Deseja deletar um arquivo do armazenamento.
Pré-condições	- Engine tem acesso ao Plugin responsável pelo armazenamento utilizado.
Garantia de sucesso	- Arquivo não existe.
Cenário de sucesso principal	1. Engine recebe o arquivo que deseja deletar. 2. Engine passa para o Plugin o caminho desse arquivo. 3. Plugin remove o arquivo. 4.6.1.7 Deletar arquivo

4.6.2.4 Deletar pasta

Nome do Caso de Uso	Deletar pasta
Nível	Objetivo da Engine
Ator principal	Engine
Interessados e interesses	**- Engine**: Deseja deletar uma pasta do armazenamento.
Pré-condições	- Engine tem acesso ao Plugin responsável pelo armazenamento utilizado.
Garantia de sucesso	- Pasta não existe.
Cenário de sucesso principal	1. Engine recebe a pasta que deseja deletar. 2. Engine deleta o conteúdo da pasta. 3. Engine passa para o Plugin o caminho dessa pasta. 4. Plugin remove a pasta. 4.6.1.3 Deletar pasta

4.6.2.5 Copiar arquivo

Nome do Caso de Uso	Copiar arquivo
Nível	Objetivo da Engine
Ator principal	Engine
Interessados e interesses	**- Engine**: Deseja copiar um arquivo de um armazenamento (origem) para outro armazenamento (destino).

Pré-condições	- Engine tem acesso ao Plugin responsável pelo armazenamento origem. - Engine tem acesso ao Plugin responsável pelo armazenamento destino.
Garantia de sucesso	- Copia do arquivo.
Cenário de sucesso principal	1. Engine recebe o arquivo que deseja copiar. 2. Engine recebe a pasta onde vai armazenar a cópia. 3. Engine formula o caminho do arquivo cópia no armazenamento destino. 4. Engine passa o caminho do arquivo cópia para o Plugin do armazenamento destino. 5. Plugin cria arquivo. 4.6.1.4 Criar arquivo 6. Engine passa o caminho do arquivo para o Plugin do armazenamento origem. 7. Plugin lê arquivo. 4.6.1.5 Ler arquivo 8. Engine passa o caminho do novo arquivo para o Plugin do armazenamento destino. 9. Engine passa os dados do novo arquivo para o Plugin do armazenamento destino. 10. Plugin escreve no arquivo. 4.6.1.6 Escrever no arquivo
Extensão	6a. Arquivo se localiza dentro de um Vault 1. Engine descobre o nome do arquivo no Vault. 2. Engine formula o caminho para o arquivo dentro do Vault. 3. Engine passa o caminho do arquivo no Vault para o Plugin do armazenamento origem. - Continua para a etapa 7 -

4.6.2.6 Copiar pasta

Nome do Caso de Uso	Copiar pasta
Nível	Objetivo da Engine
Ator principal	Engine
Interessados e interesses	**- Engine**: Deseja copiar uma pasta de um armazenamento (origem) para outro armazenamento (destino).
Pré-condições	- Engine tem acesso ao Plugin responsável pelo armazenamento origem.

	- Engine tem acesso ao Plugin responsável pelo armazenamento destino.
Garantia de sucesso	- Copia da pasta.
Cenário de sucesso principal	1. Engine recebe a pasta que deseja copiar, pasta origem. 2. Engine recebe a pasta onde vai armazenar a cópia, pasta destino. 3. Engine cria uma pasta vazia com o nome da pasta origem dentro da pasta destino. 4.6.2.1 Criar pasta 4. Engine passa o caminho da pasta origem para o Plugin do armazenamento origem. 5. Plugin lista pasta. 4.6.1.2 Listar pasta 6. Copiar cada pasta e arquivo para a nova pasta.

4.6.2.7 Mover arquivo

Nome do Caso de Uso	Mover arquivo
Nível	Objetivo da Engine
Ator principal	Engine
Interessados e interesses	**- Engine**: Deseja mover um arquivo de um armazenamento (origem) para outro armazenamento (destino).
Pré-condições	- Engine tem acesso ao Plugin responsável pelo armazenamento origem. - Engine tem acesso ao Plugin responsável pelo armazenamento destino.
Garantia de sucesso	- Arquivo movido.
Cenário de sucesso principal	1. 4.6.2.5 Copiar arquivo 2. 4.6.2.3 Deletar arquivo

4.6.2.8 Mover pasta

Nome do Caso de Uso	Mover pasta
Nível	Objetivo da Engine
Ator principal	Engine
Interessados e interesses	**- Engine**: Deseja mover uma pasta de um armazenamento (origem) para outro

	armazenamento (destino).
Pré-condições	- Engine tem acesso ao Plugin responsável pelo armazenamento origem. - Engine tem acesso ao Plugin responsável pelo armazenamento destino.
Garantia de sucesso	- Pasta movida.
Cenário de sucesso principal	1. 4.6.2.6 Copiar pasta 2. 4.6.2.4 Deletar pasta

4.6.2.9 Cifrar arquivo

Nome do Caso de Uso	Cifrar arquivo
Nível	Objetivo da Engine
Ator principal	Engine
Interessados e interesses	**- Engine**: Deseja salvar no arquivo o seu conteúdo cifrado.
Pré-condições	- Engine tem acesso ao Plugin responsável pelo armazenamento. - Engine possui a chave pública e chave privada daquele armazenamento.
Garantia de sucesso	- Conteúdo cifrado.
Cenário de sucesso principal	1. Engine recebe o arquivo a ser cifrado. 2. Engine obtém o conteúdo do arquivo. 4.6.1.5 Ler arquivo 3. Engine cifra o conteúdo. 4. Engine escreve o conteúdo cifrado no arquivo. 4.6.1.6 Escrever no arquivo

4.6.2.10 Cifrar pasta

Nome do Caso de Uso	Cifrar pasta
Nível	Objetivo da Engine
Ator principal	Engine
Interessados e interesses	**- Engine**: Deseja que todos os arquivos contidos na pasta tenham seu conteúdo cifrado.
Pré-condições	- Engine tem acesso ao Plugin responsável pelo armazenamento. - Engine possui a chave pública e chave privada

	daquele armazenamento.
Garantia de sucesso	- Conteúdo cifrado.
Cenário de sucesso principal	1. Engine recebe a pasta a qual deseja que os arquivos dentro sejam cifrados. 2. Engine lista os arquivos e pastas contidos na pasta que deseja cifrar. 4.6.2.2 Listar pasta 3. Engine cifra cada pasta e arquivo.

4.6.2.11 Decifrar arquivo

Nome do Caso de Uso	Decifrar arquivo
Nível	Objetivo da Engine
Ator principal	Engine
Interessados e interesses	**- Engine**: Deseja resgatar o conteúdo original do arquivo.
Pré-condições	- Engine tem acesso ao Plugin responsável pelo armazenamento. - Engine possui a chave pública e chave privada daquele armazenamento.
Garantia de sucesso	- Arquivo contém o conteúdo original.
Cenário de sucesso principal	1. Engine recebe o arquivo a ser decifrado. 2. Engine obtém o conteúdo do arquivo. 4.6.1.5 Ler arquivo 3. Engine decifra o conteúdo. 4. Engine escreve o conteúdo decifrado no arquivo. 4.6.1.6 Escrever no arquivo

4.6.2.12 Decifrar pasta

Nome do Caso de Uso	Decifrar pasta
Nível	Objetivo da Engine
Ator principal	Engine
Interessados e interesses	**- Engine**: Deseja que todos os arquivos contidos na pasta tenham seu conteúdo original.
Pré-condições	- Engine tem acesso ao Plugin responsável pelo armazenamento. - Engine possui a chave pública e chave privada daquele armazenamento.

Garantia de sucesso	- Arquivos dentro da pasta tem seu conteúdo original.
Cenário de sucesso principal	1. Engine recebe a pasta a qual deseja que os arquivos dentro sejam decifrados. 2. Engine lista os arquivos e pastas contidos na pasta que deseja cifrar. 4.6.2.2 Listar pasta 3. Engine decifra cada pasta e arquivo.

4.6.3 Engine Vault

4.6.3.1 Criar pasta segura

Nome do Caso de Uso	Criar pasta segura
Nível	Objetivo da Engine
Ator principal	Engine
Interessados e interesses	- **Engine**: Deseja criar uma pasta segura vazia dentro de uma pasta especificada.
Pré-condições	- Engine tem acesso ao Plugin responsável pelo armazenamento utilizado.
Garantia de sucesso	- Pasta segura vazia criada no caminho especificado. - Engine possui o arquivo Index.
Cenário de sucesso principal	1. Engine recebe o nome da nova pasta. 2. Engine recebe a pasta onde a nova pasta deve ser criada. 3. Engine cria um nome aleatório não utilizado. 4. Engine formula o caminho da nova pasta. 5. Engine adiciona a nova pasta no Index. 4.6.3.11 Adicionar ao Index

4.6.3.2 Listar pasta segura

Nome do Caso de Uso	Listar pasta segura
Nível	Objetivo da Engine
Ator principal	Engine
Interessados e interesses	- **Engine**: Deseja obter todos os arquivos e pasta dentro de uma pasta especificada.
Pré-condições	- Engine tem acesso ao Plugin responsável pelo armazenamento utilizado.

	- Engine possui o arquivo Index.
Garantia de sucesso	- Lista de arquivos e pastas.
Cenário de sucesso principal	1. Engine recebe a pasta que deseja listar. 2. Engine lê o Index. 4.6.3.9 Ler Index 3. Engine lista todos os arquivos onde o caminho está diretamente abaixo da pasta recebida.

4.6.3.3 Deletar arquivo seguro

Nome do Caso de Uso	Deletar arquivo seguro
Nível	Objetivo da Engine
Ator principal	Engine
Interessados e interesses	**- Engine**: Deseja deletar um arquivo do armazenamento.
Pré-condições	- Engine tem acesso ao Plugin responsável pelo armazenamento utilizado. - Engine possui o arquivo Index.
Garantia de sucesso	- Arquivo não existe.
Cenário de sucesso principal	1. Engine recebe o arquivo que deseja deletar. 2. Engine pega o nome cifrado do arquivo. 3. Engine formula o caminho do arquivo cifrado. 4. Plugin remove o arquivo seguro. 4.6.1.7 Deletar arquivo 5. Engine remove arquivo do Index. 4.6.3.12 Remover do Index

4.6.3.4 Deletar pasta segura

Nome do Caso de Uso	Deletar pasta segura
Nível	Objetivo da Engine
Ator principal	Engine
Interessados e interesses	**- Engine**: Deseja deletar uma pasta do armazenamento.
Pré-condições	- Engine tem acesso ao Plugin responsável pelo armazenamento utilizado. - Engine possui o arquivo Index.

Garantia de sucesso	- Pasta não existe.
Cenário de sucesso principal	1. Engine recebe a pasta que deseja deletar. 2. Engine lista os arquivos da pasta segura. 4.6.3.2 Listar pasta segura 3. Engine deleta os arquivos da pasta segura. 4. Engine remove pasta segura do Index. 4.6.3.12 Remover do Index

4.6.3.5 Copiar arquivo seguro

Nome do Caso de Uso	Copiar arquivo seguro
Nível	Objetivo da Engine
Ator principal	Engine
Interessados e interesses	**- Engine**: Deseja copiar um arquivo de um armazenamento (origem) para outro armazenamento (destino) dentro de um Vault.
Pré-condições	- Engine tem acesso ao Plugin responsável pelo armazenamento origem. - Engine tem acesso ao Plugin responsável pelo armazenamento destino. - Engine possui o arquivo Index.
Garantia de sucesso	- Copia do arquivo.
Cenário de sucesso principal	1. Engine recebe o arquivo que deseja copiar. 2. Engine recebe a pasta onde vai armazenar a cópia. 3. Engine cria um nome aleatório não utilizado. 4. Engine formula o caminho do arquivo cópia no armazenamento destino. 5. Engine passa o caminho do arquivo cópia para o Plugin do armazenamento destino. 6. Plugin cria arquivo. 4.6.1.4 Criar arquivo 7. Engine passa o caminho do arquivo para o Plugin do armazenamento origem. 8. Plugin lê arquivo. 4.6.1.5 Ler arquivo 9. Engine cifra o conteúdo lido. 8. Engine passa o caminho do novo arquivo para o Plugin do armazenamento destino. 9. Engine passa o conteúdo cifrado para o Plugin do armazenamento destino. 10. Plugin escreve no novo arquivo. 4.6.1.6 Escrever no arquivo 11. Engine formula o caminho dentro do Vault

	para o novo arquivo. 12. Engine adiciona o novo arquivo ao Index. 4.6.3.11 Adicionar ao Index
Extensões	7a. Arquivo se localiza dentro de um Vault 1. Engine descobre o nome do arquivo no Vault. 2. Engine formula o caminho para o arquivo dentro do Vault. 3. Engine passa o caminho do arquivo no Vault para o Plugin do armazenamento origem. - Contínua para a etapa 8 -

4.6.3.6 Copiar pasta segura

Nome do Caso de Uso	Copiar pasta segura
Nível	Objetivo da Engine
Ator principal	Engine
Interessados e interesses	**- Engine**: Deseja copiar uma pasta de um armazenamento (origem) para outro armazenamento (destino) dentro de um Vault.
Pré-condições	- Engine tem acesso ao Plugin responsável pelo armazenamento origem. - Engine tem acesso ao Plugin responsável pelo armazenamento destino.
Garantia de sucesso	- Copia da pasta.
Cenário de sucesso principal	1. Engine recebe a pasta que deseja copiar, pasta origem. 2. Engine recebe a pasta onde vai armazenar a cópia, pasta destino. 3. Engine cria uma pasta vazia com o nome da pasta origem dentro da pasta destino, utilizando o método seguro de criar pastas no Vault. 4.6.3.1 Criar pasta segura 4. Engine passa o caminho da pasta origem para o Plugin do armazenamento origem. 5. Plugin lista pasta. 4.6.2.2 Listar pasta 6. Copiar cada pasta e arquivo para a nova pasta.

4.6.3.7 Mover arquivo seguro

Nome do Caso de Uso	Mover arquivo seguro
Nível	Objetivo da Engine
Ator principal	Engine
Interessados e interesses	**- Engine**: Deseja mover um arquivo de um armazenamento (origem) para outro armazenamento (destino).
Pré-condições	- Engine tem acesso ao Plugin responsável pelo armazenamento origem. - Engine tem acesso ao Plugin responsável pelo armazenamento destino. - Engine possui o arquivo Index.
Garantia de sucesso	- Arquivo movido.
Cenário de sucesso principal	1. 4.6.3.5 Copiar arquivo seguro 2. 4.6.3.3 Deletar arquivo seguro

4.6.3.8 Mover pasta segura

Nome do Caso de Uso	Mover pasta segura
Nível	Objetivo da Engine
Ator principal	Engine
Interessados e interesses	**- Engine**: Deseja mover uma pasta de um armazenamento (origem) para outro armazenamento (destino).
Pré-condições	- Engine tem acesso ao Plugin responsável pelo armazenamento origem. - Engine tem acesso ao Plugin responsável pelo armazenamento destino. - Engine possui o arquivo Index.
Garantia de sucesso	- Pasta movida.
Cenário de sucesso principal	1. 4.6.3.6 Copiar pasta segura 2. 4.6.3.4 Deletar pasta segura

4.6.3.9 Ler Index

Nome do Caso de Uso	Ler Index

Nível	Objetivo da Engine
Ator principal	Engine
Interessados e interesses	**- Engine**: Deseja ler o Index para obter os arquivos dentro do Vault.
Pré-condições	- Engine tem acesso ao Plugin responsável pelo armazenamento. - Engine possui o arquivo Index.
Garantia de sucesso	- Lista de arquivos dentro do Index.
Cenário de sucesso principal	1. Engine passa o caminho do Index para o Plugin do armazenamento. 2. Plugin lê arquivo Index. 4.6.1.5 Ler arquivo 3. Engine divide o conteúdo em linhas pois cada linha representa um arquivo. 4. Engine divide as linhas pelo caractere responsável por separar informações, cada separação representa uma informação diferente. 5. Engine gera a lista de arquivos que o Index possui. 6. Engine devolve essa lista de arquivos.

4.6.3.10 Escrever Index

Nome do Caso de Uso	Escrever Index
Nível	Objetivo da Engine
Ator principal	Engine
Interessados e interesses	**- Engine**: Deseja escrever no Index a nova lista de arquivos no Vault.
Pré-condições	- Engine tem acesso ao Plugin responsável pelo armazenamento. - Engine possui o arquivo Index.
Garantia de sucesso	- Index possuindo a lista de arquivos no Vault.
Cenário de sucesso principal	1. Engine recebe uma lista de arquivos. 2. Engine pega as informações dos arquivos na lista e formata essa informações em uma String de acordo com o padrão do Index. 3. Engine cifra a informação dessa String. 4. Engine passa o caminho do Index para o Plugin. 5. Plugin escreve a informação cifrada. 4.6.1.6

	Escrever no arquivo

4.6.3.11 Adicionar ao Index

Nome do Caso de Uso	Adicionar ao Index
Nível	Objetivo da Engine
Ator principal	Engine
Interessados e interesses	**- Engine**: Deseja adicionar um arquivo ao Index.
Pré-condições	- Engine tem acesso ao Plugin responsável pelo armazenamento. - Engine possui o arquivo Index.
Garantia de sucesso	- A lista de arquivos no Index foi incrementada com o arquivo novo passado.
Cenário de sucesso principal	1. Engine recebe o arquivo a ser adicionado. 2. Engine obtém a lista de arquivos no Index. 4.6.3.9 Ler Index 3. Engine adiciona o arquivo passado. 4. Engine escreve no Index a nova lista. 4.6.3.10 Escrever Index

4.6.3.12 Remover do Index

Nome do Caso de Uso	Remover do Index
Nível	Objetivo da Engine
Ator principal	Engine
Interessados e interesses	**- Engine**: Deseja remover um arquivo do Index.
Pré-condições	- Engine tem acesso ao Plugin responsável pelo armazenamento. - Engine possui o arquivo Index.
Garantia de sucesso	- A lista de arquivos no Index não possui o arquivo.
Cenário de sucesso principal	1. Engine recebe o arquivo a ser removido. 2. Engine obtém a lista de arquivos no Index. 4.6.3.9 Ler Index 3. Engine procura na lista de arquivos o arquivo a ser removido.

	4. Engine remove da lista o arquivo. 5. Engine escreve no Index a nova lista. 4.6.3.10 Escrever Index

4.7. Matriz Rastreabilidade

		Requisitos Funcionais							
		1	2	3	4	5	6	7	8
Caso de Uso	4.6.1.1	x	x						
	4.6.1.2	x		x					
	4.6.1.3	x			x				
	4.6.1.4	x				x			
	4.6.1.5	x					x		
	4.6.1.6	x						x	
	4.6.1.7	x							x

		Requisitos Funcionais											
		9	10	11	12	13	14	15	16	17	18	19	20
Caso de Uso	4.6.2.1	x											
	4.6.2.2		x										
	4.6.2.3			x									
	4.6.2.4				x								
	4.6.2.5					x							
	4.6.2.6						x						
	4.6.2.7							x					
	4.6.2.8								x				
	4.6.2.9									x			
	4.6.2.10										x		
	4.6.2.11											x	

	4.6.2.12												x

		Requisitos Funcionais								
		21	22	23	24	25	26	27	28	29
Caso de Uso	4.6.3.1		x							
	4.6.3.2			x						
	4.6.3.3				x					
	4.6.3.4					x				
	4.6.3.5						x			
	4.6.3.6							x		
	4.6.3.7								x	
	4.6.3.8									x
	4.6.3.9		x	x	x	x	x	x	x	x
	4.6.3.10	x	x		x	x	x	x	x	x
	4.6.3.11		x				x	x	x	x
	4.6.3.12				x	x	x	x	x	x

5. Implementação

Este capítulo destaca as partes do código relevantes para o entendimento da operação da aplicação.

5.1. Plugin

A arquitetura do Plugin é focada em ser minimalista e de fácil implementação. Qualquer Plugin desenvolvido deve estender desta interface:

```
public interface Plugin {

    public void createFolder(String folderPath) throws Exception;
    public ArrayList<String[]> listFolder(String folderPath) throws Exception;
    public void deleteFolder(String folderPath) throws Exception;

    public void createFile(String filePath) throws Exception;
    public byte[] readFile(String filePath) throws Exception;
    public void writeFile(String filePath, byte[] fileBytes) throws Exception;
    public void deleteFile(String filePath) throws Exception;

}
```

Figura 2: Interface a qual as classes Plugin devem estender.

createFolder(): Cria uma pasta vazia no caminho especificado.

listFolder(): Retorna uma lista de **nomes** e **tipos** (arquivo/pasta).

deleteFolder(): Remove uma pasta **vazia**.

createFile(): Cria um arquivo vazio.

readFile(): Retorna os bytes de um arquivo.

writeFile(): **Substitui** os bytes de um arquivo.

deleteFile(): Remove um arquivo.

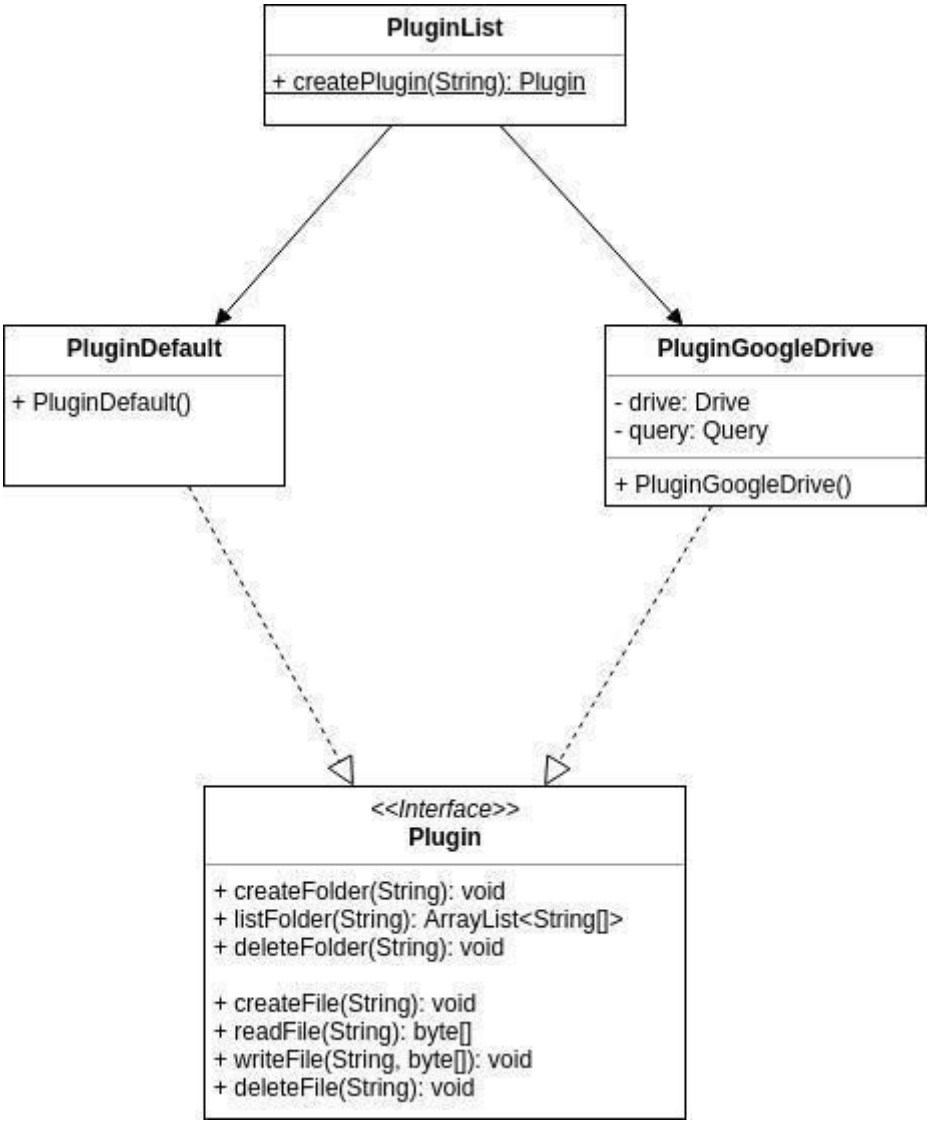

Figura 3: Diagrama de classe dos Plugins.

5.2. Google Drive Plugin

Para o Plugin do Google Drive, foram criadas classes para darem suporte às operações básicas e à criação do Plugin. A classe Utility ajuda com a autenticação do usuário e obtenção do token. Por sua vez, a classe Query faz operações que são muitas vezes necessárias durante as operações básicas.

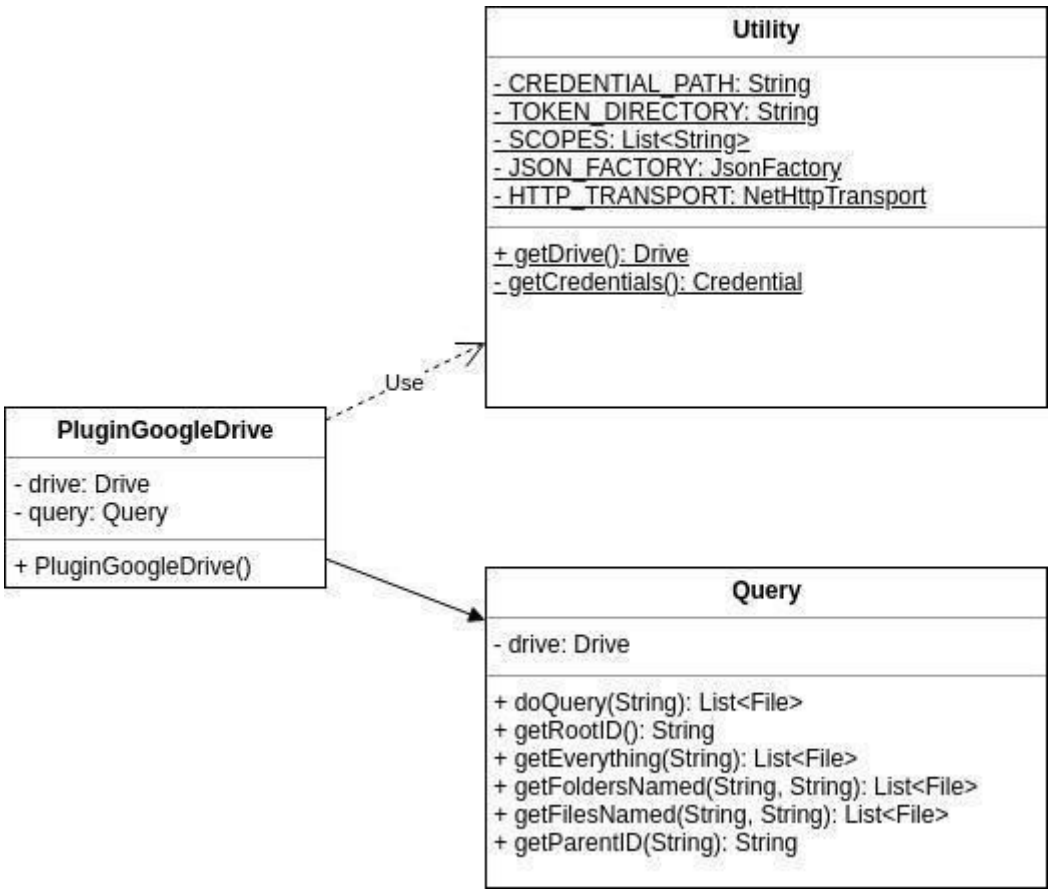

Figura 4: Diagrama de classes do Plugin Google Drive.

5.3. Drive e File

O Drive é a classe responsável por guardar informações daquele serviço de armazenamento, ou seja, como manipular com o armazenamento e as chaves utilizadas para a proteção do vault daquele serviço.

Para representar arquivos e pastas no armazenamento, foi criada a classe File. Esta classe possui todas as informações necessárias para a manipulação do arquivo/pasta no serviço de armazenamento e para apresentar ao usuário.

A partir de um objeto File, consegue-se descobrir a qual Drive o mesmo pertence e qual Plugin deve ser utilizado para se comunicar com o serviço de armazenamento.

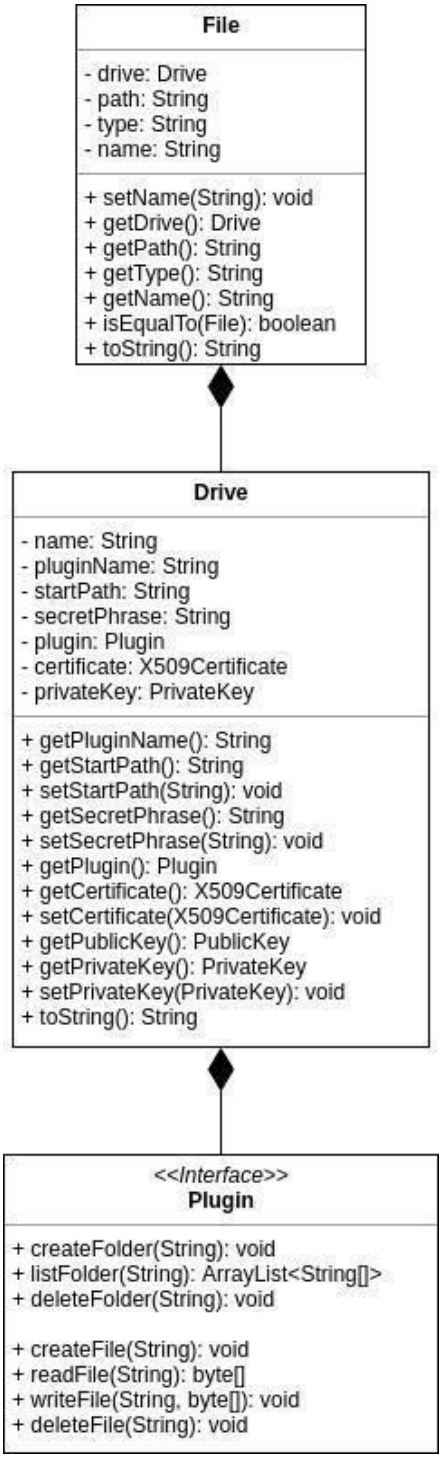

Figura 5: Diagrama de classes do File e Drive.

5.4 Engine

A classe Engine trabalha manipulando os objetos File e Drive para garantir as funcionalidades básicas. As funcionalidades básicas da Engine tiveram que ser desenvolvidas para caso estivessem lidando com o vault.

```
public class Engine {

    public static File createFolder(File folder, String newFolderName) throws Exception {

    public static ArrayList<File> listFolder(File folder) throws Exception {

    public static void copy(File file, File toFolder) throws Exception {

    public static void move(File file, File toFolder) throws Exception {

    public static void delete(File file) throws Exception {

    public static void cipher(File file) throws Exception {

    public static void decipher(File file) throws Exception {

}
```

Figura 6: Classe Engine com as funcionalidades.

```
public class Create {

    public static File createFolder(File folder, String newFolderName) throws Exception {

    public static File createSafeFolder(File folder, String newFolderName) throws Exception {
}
```

Figura 7: Método para criação de pasta normal e para criação de pasta no vault.

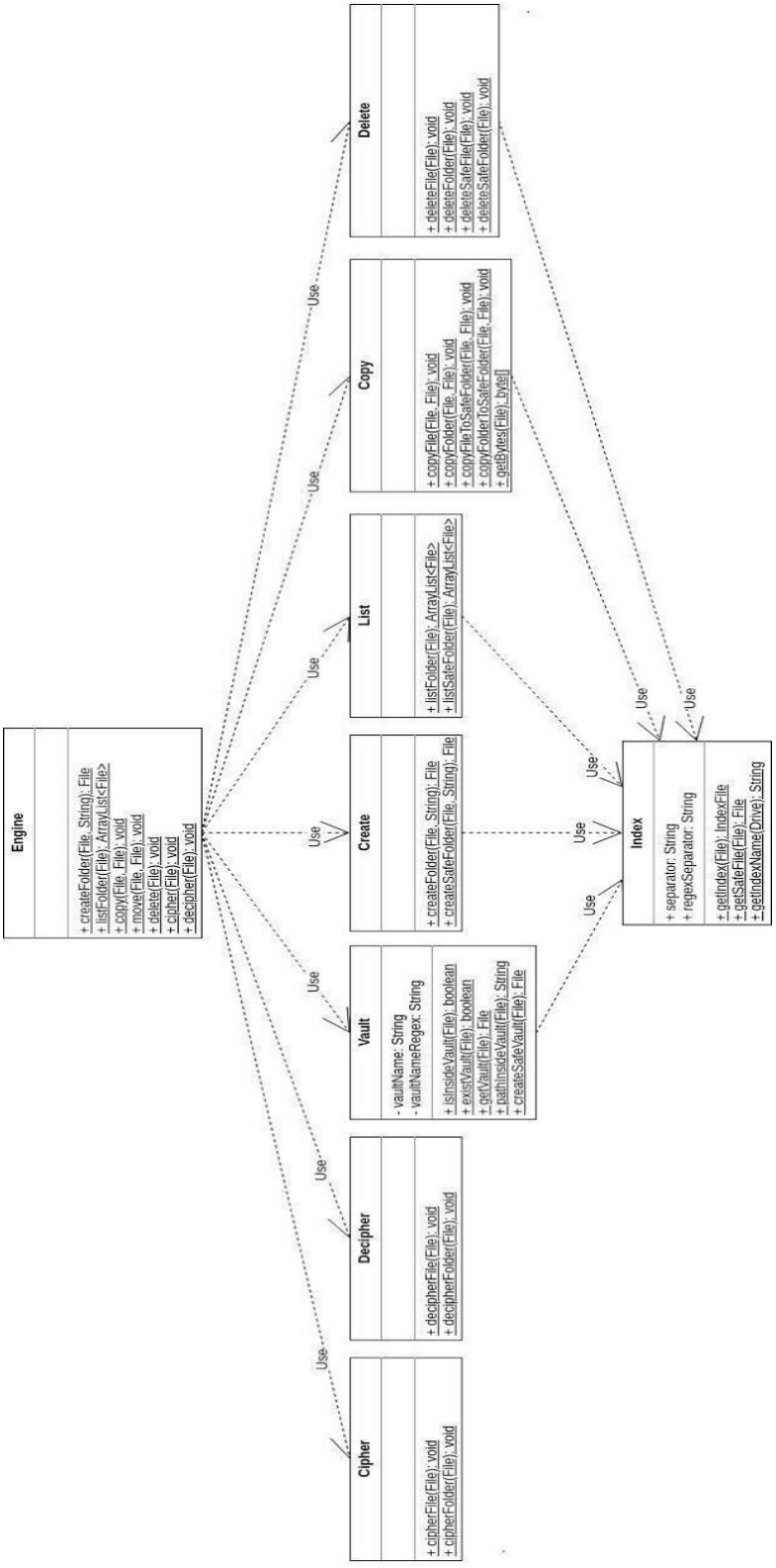

Figura 8: Diagrama de Classe da Engine.

5.5. Encrypt e Decrypt

Foram desenvolvidas duas classes para prover as os recursos de criptografia, assinatura digital e envelope digital. Estes métodos são utilizados em qualquer operação sobre informações no vault (arquivos e index).

```
public class Encrypt {

    public static byte[] createRandomSeed() {

    public static Key getSecretKey(byte[] seed) throws Exception {

    public static byte[] getEncryptedSeed(byte[] seed, PublicKey publicKey) throws Exception {

    public static byte[] getEncryptedContent(byte[] content, Key key) throws Exception {

    public static byte[] getSignature(byte[] content, PrivateKey privateKey) throws Exception {

    public static byte[] getEncryptedFile(byte[] content, PrivateKey privateKey, PublicKey publicKey)
}
```

Figura 9: Classe Encrypt.

```
public class Decrypt {

    public static Key getSecretKey(byte[] seed) throws Exception {

    public static byte[] getDecryptedSeed(byte[] encryptedSeed, PrivateKey privateKey) throws Exception

    public static byte[] getDecryptedContent(byte[] encryptedContent, Key key) throws Exception {

    public static boolean isSignatureOK(byte[] content, byte[] signature, PublicKey publicKey) throws E

    public static byte[] getDecryptedFile(byte[] container, PrivateKey privateKey, PublicKey publicKey)

}
```

Figura 10: Classe Decrypt.

5.6. Interface

Na linguagem Java, o nome *interface* é reservado, por isso o nome utilizado neste projeto para representar essa camada foi o *view*.

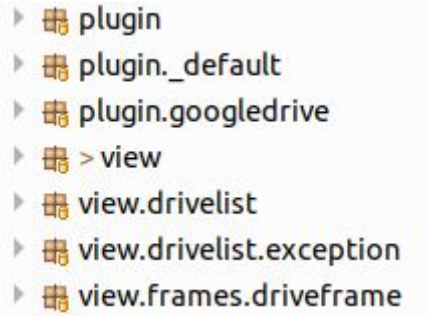

Figura 11: Nome dos pacotes relacionados a Interface estão como View

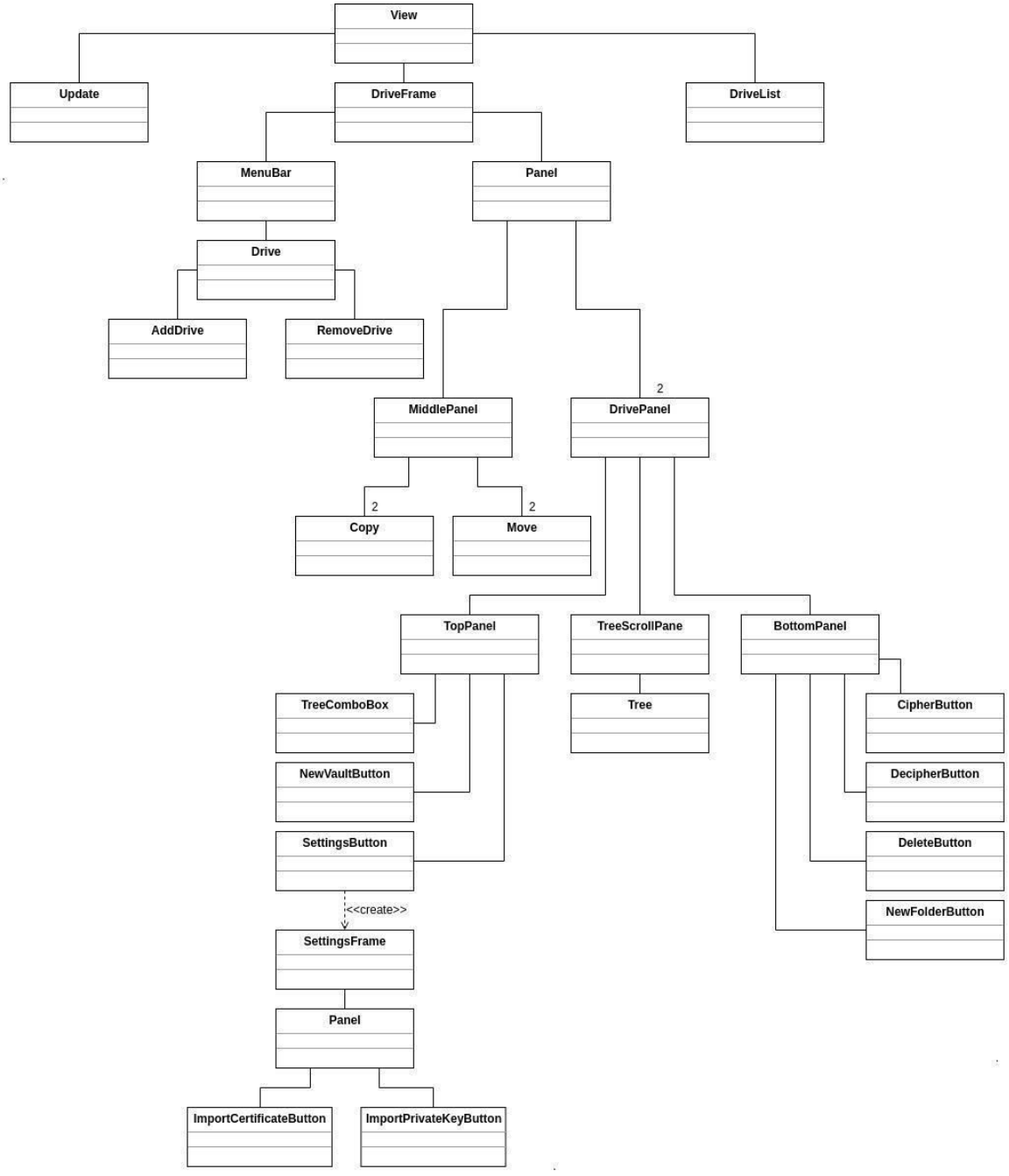

Figura 12: Diagrama de Classe da Interface.

A janela da aplicação é dividida em 2 painéis principais: volume real (à esquerda) e volume seguro (à direita). No volume real, o usuário consegue ver como os arquivos protegidos estão realmente armazenados no sistema de arquivos do seu dispositivo local ou remoto, enquanto que no volume seguro, o usuário consegue manipular com os arquivos e pastas protegidos, que são mostrados nos respectivos formatos originais.

Figura 13: Ferramenta Safe Vault aberta, demonstrando o volume real e seguro.

6. Conclusão

A aplicação construída durante este projeto foi focada na segurança dos dados que são armazenados em diversos serviços de armazenamentos, criando uma proteção vinda do próprio usuário, em vez de depender dos armazenamentos.

Todo o desenvolvimento da aplicação foi pensado de forma a criar um framework capaz de atingir diversas plataformas e serviços de desenvolvimento. Por isso, foi organizada em 3 camadas: Interface, Engine e Plugin.

Para alcançar o objetivo, foi necessário aplicar técnicas de criptografia simétrica e assimétrica, resumo de mensagem, assinatura digital e envelope digital.

A aplicação garante uma ferramenta única para a movimentação de dados entre sistemas de armazenamento de diferentes dispositivos com proteção dos dados. Para adicionar novos dispositivos, basta desenvolver novos plug-ins sem ter que mudar as camadas Engine e Interface.

6.1. Metas futuras

A fim de demonstrar a capacidade de integração com diversas plataformas, deseja-se desenvolver uma Interface para mobile, como por exemplo, para Android.

O método de busca utilizado no Plugin do Google Drive se torna lento em diversas situações. Isso pode ser aperfeiçoado com implementação de um cache para não executar as mesmas requisições sobre o serviço de armazenamento.

Atualmente, o container foi desenvolvido pensando que a assinatura digital e a semente cifrada terão sempre o mesmo tamanho. Para atingir a todos os tamanhos é necessário adotar um protocolo para identificar o tamanho de cada item no container. Por exemplo, adotar o protocolo de representação de dados TLV (type-length-value).

As metainformações dos arquivos no vault, estão todas concentradas em um único index. Dependendo da quantidade dos arquivos do vault, o index pode se tornar muito grande, aumentando o custo para se obter a informação de um único arquivo. Para resolver isto, deseja-se separar as metainformações por pastas que

elas representam, ou seja, um arquivo irá conter toda a metainformação que uma pasta possui.

7. Referências Bibliográficas

[1] Google. **Google Drive**. Acessado em: 13 Set. 2018. Disponível em: <https://www.google.com/drive/>

[2] Dropbox Inc. **Dropbox**. Acessado em: 13 Set. 2018. Disponível em: <https://www.dropbox.com>

[3] Microsoft. **OneDrive**. Acessado em: 13 Set. 2018. Disponível em: <https://onedrive.live.com>

[4] YUSUF HAIDER M, SIVA SELVAN. **Confidentiality Issues in Cloud Computing and Countermeasures: A Survey**. Departamente of Computer Science and Engineering. Manipal Institute of Technology (Manipal University). Acessado em: 9 Fev. 2019. Disponível em: <https://www.researchgate.net/publication/305689086_Confidentiality_Issues_in_Cloud_Computing_and_Countermeasures_A_Survey>

[5] ALFREDO KURY ABÍLIO, ANDRÉ CONCEIÇÃO DA GRAÇA, CRISTIANO PEDRO DA SILVA, MAURO SÉRGIO DOS SANTOS AMORIM. **Comunicação Segura na Internet: Métodos, Infra-estrutura de Chaves Públicas e Padrões**. Trabalho de Conclusão de Curso.

[6] PRADEEP KUMAR TIWARI. **Cloud Computing Security Issues, Challenges and Solution**. International Journal of Emerging Technology and Advanced Engineering, 2012. Disponível em: <https://www.researchgate.net/publication/271522943/download>

[7] Google Cloud. **How Google Uses Encryption to Protect Your Data**. Acessado em: 13 Set. 2018. Disponível em: <https://storage.googleapis.com/gfw-touched-accounts-pdfs/google-encryption-whitepaper-gsuite.pdf>

[8] Boxcryptor. **Boxcryptor**. Disponível em: <https://www.boxcryptor.com>

[9] Canonical Ltd. **Ubuntu**. Acessado em: 5 Fev. 2019. Disponível em: <https://www.ubuntu.com/>

[10] Oracle. **Java Cryptography Architecture (JCA) Reference Guide**. Acessado em: 13 Set. 2018. Disponível em:

<https://docs.oracle.com/javase/8/docs/technotes/guides/security/crypto/CryptoSpec.html>

[11] Google. **Google Drive APIs REST**. Acessado em: 13 Set. 2018. Disponível em:

<https://developers.google.com/drive/api/v3/about-sdk>

[12] Google. **Google API Client Libraries**. Acessado em: 13 Set. 2018. Disponível em:

<https://developers.google.com/api-client-library/>

[13] Eclipse Foundation. **Eclipse**. Acessado em: 5 Fev. 2019. Disponível em:

<https://www.eclipse.org/>

[14] UML. **UML**. Acessado em: 13 Set. 2018. Disponível em:

<http://www.uml.org/>

[15] Google. **Google Cloud Platform**. Acessado em: 5 Fev. 2019. Disponível em:

<https://console.cloud.google.com/>

[16] Google. **Google Drive API**. Acessado em: 5 Fev. 2019. Disponível em:

<https://console.cloud.google.com/apis/library/drive.googleapis.com?q=Drive>

[17] Google. **Using OAuth 2.0 to Access Google APIs**. Acessado em: 5 Fev. 2019. Disponível em:

<https://developers.google.com/identity/protocols/OAuth2>

[18] Cryptomator. **Cryptomator**. Acessado em: 9 Jul. 2019. Disponível em:

<https://cryptomator.org/>

[19] Cyberduck. **Cyberduck**. Acessado em: 9 Jul. 2019. Disponível em:

<https://cyberduck.io/>

[20] Oracle. **Class Files documentation**. Acessado em: 12 Jul. 2019. Disponível em:

<https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/nio/file/Files.html#delete(java.nio.file.Path)>

[21] Microsoft. **DeleteFile function**. Acessado em: 12 Jul. 2019. Disponível em:

<https://docs.microsoft.com/en-us/windows/win32/api/fileapi/nf-fileapi-deletefilea#remarks>

[22] Google. **Files reference, Google Drive API**. Acessado em: 13 Jul. 2019. Disponível em:

<https://developers.google.com/drive/api/v3/reference/files>

[23] Gilleanes T. A. Guedes. **UML 2 uma abordagem prática**. Novatec Editora Ltda.

[24] CAROLINA GWOZDZ POERSCH, GIULLYAN METZKER KUNTZE. Modelo de Coleta e Análise de Evidências em Sistemas Computacionais. Universidade do Sul de Santa Catarina, 2010. Disponível em: <https://www.riuni.unisul.br/bitstream/handle/12345/3229/100889_Carolina.pdf?sequence=1>

[25] SEAN PEISERT, ED TALBOT, TOM KROEGER. **Principles of Authentication**. California, USA, 2013. Disponível em: <https://www.nspw.org/papers/2013/nspw2013-peisert.pdf>

[26] Dmitry Khovratovich and Christian Rechberger and Alexandra Savelieva. **Bicliques for Preimages: Attacks on Skein-512 and the SHA-2 family**. Acessado em: 20 Jul. 2019. Disponível em:

<https://eprint.iacr.org/2011/286.pdf>

[27] Mario Lamberger and Florain Mendel. **Higher-Order Differential Attack on Reduced SHA-256**. Acessado em: 20 Jul. 2019. Disponível em:

<https://eprint.iacr.org/2011/037.pdf>

[28] Kartit, Zaid. **Applying Encryption Algorithms for Data Security in Cloud Storage, Kartit, et al**. Acessado em: 20 Jul. 2019. Disponível em:

<https://books.google.com.br/books?id=uEGFCwAAQBAJ&pg=PA147&dq=%22keys+may+be+identical%22&redir_esc=y#v=onepage&q=%22keys%20may%20be%20identical%22&f=false>