PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
GABINETE DO REITOR

Av. Universitária, 1069 ● Setor Universitário
Caixa Postal 86 ● CEP 74605-010
Goiânia ● Goiás ● Brasil
Fone: (62) 3946.1000
www.pucgoias.edu.br ● reitoria@pucgoias.edu.br

RESOLUÇÃO n° 038/2020 – CEPE

ANEXO I

APÊNDICE ao TCC

Termo de autorização de publicação de produção acadêmica

O estudante DANIEL OLIVEIRA DE SOUSA do Curso de ENGENHARIA DE COMPUTAÇÃO matrícula 2021.1.0033.0050-9, telefone: (62) 99233-2841 e-mail 2021.1.0033.0050-9@pucgo.edu.br, na qualidade de titular dos direitos autorais, em consonância com a Lei nº 9.610/98 (Lei dos Direitos do Autor), autoriza a Pontifícia Universidade Católica de Goiás (PUC Goiás) a disponibilizar o Trabalho de Conclusão de Curso intitulado SOFTWARE PARA RECUPERAÇÃO DE ARQUIVOS EXCLUÍDOS, gratuitamente, sem ressarcimento dos direitos autorais, por 5 (cinco) anos, conforme permissões do documento, em meio eletrônico, na rede mundial de computadores, no formato especificado (Texto(PDF); Imagem (GIF ou JPEG); Som (WAVE, MPEG, AIFF, SND); Vídeo (MPEG, MWV, AVI, QT); outros, específicos da área; para fins de leitura e/ou impressão pela internet, a título de divulgação da produção científica gerada nos cursos de graduação da PUC Goiás.

Goiânia, 06 de abril de 2026 .

Assinatura do autor: birdID Documento assinado digitalmente por: DANIEL OLIVEIRA DE SOUSA 066.176.161-46 2026-04-06T18:32:25.286440

Nome completo do autor: DANIEL OLIVEIRA DE SOUSA

gov.br Documento assinado digitalmente GUSTAVO SIQUEIRA VINHAL Data: 06/04/2026 19:46:16-0300 Verifique em https://validar.iti.gov.br

Assinatura do professor–orientador: _____

Nome completo do professor-orientador: GUSTAVO SIQUEIRA VINHAL

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA POLITÉCNICA E DE ARTES
GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

IMPLEMENTAÇAO DE UM SOFTWARE PARA RECUPERAÇÃO DE ARQUIVOS DELETADOS

DANIEL OLIVEIRA DE SOUSA

GOIÂNIA
2026

DANIEL OLIVEIRA DE SOUSA

IMPLEMENTAÇÃO DE UM SOFTWARE PARA RECUPERAÇÃO DE ARQUIVOS DELETADOS

Trabalho para conclusão de curso apresentado na Escola Politécnica e de Artes da Pontifícia Universidade Católica de Goiás como requisito básico para a conclusão do curso de Engenharia de Computação.

Orientador: Prof. Me. Gustavo Siqueira Vinhal

GOIÂNIA

2026

DANIEL OLIVEIRA DE SOUSA

ELABORAÇÃO DE SOFTWARE PARA RECUPERAÇÃO DE ARQUIVOS DELETADOS

Este trabalho foi julgado adequado para obtenção do título de Bacharel em Engenharia de Computação, e aprovado em sua forma final pela Escola Politécnica e de Artes, da Pontifícia Universidade Católica de Goiás em _____/_____/_____.

_____

Banca examinadora:

_____

Prof. Me. Gustavo Siqueira Vinhal

_____

Prof. Me. Carlos Alexandre Ferreira de Lima

_____

Prof. Dr. Nilson Cardoso Amaral

Dedico este trabalho aos meus pais, Avany de Sousa e Meiriane de Souza Oliveira, que sempre foram minha base, minha inspiração. Também à minha irmã, Bruna Oliveira de Sousa cuja trajetória acadêmica, profissional e pessoal me inspiram cada dia mais.

AGRADECIMENTOS

A Deus, pois sem Ele eu não teria conseguido chegar até este ponto, Ele me deu forças e o conhecimento para seguir em frente mesmo com adversidades.

A minha família e aos meus amigos, pelo apoio incondicional ao longo dos anos e pelos momentos inesquecíveis que tornaram minha jornada leve, divertida e significativa.

Ao meu orientador acadêmico, Professor Me. Gustavo Siqueira Vinhal, por acreditar em mim, pela paciência e por todo apoio prestado durante o desenvolvimento deste trabalho.

Aos professores com os quais tive a honra de aprender e conviver ao longo de minha trajetória acadêmica, cujos ensinamentos foram fundamentais para minha formação.

Sou grato a todo corpo docente, à direção e à administração desta universidade.

E, por fim, sou grato a todos que, de alguma forma, contribuíram para a realização e conclusão deste projeto.

“Quer um conselho? Se ninguém disse que é impossível, é porque dá pra fazer. Então vai e faz!”

Ezreal

RESUMO

A crescente dependência de dispositivos digitais para armazenamento de informações tem intensificado a ocorrência de perdas de dados em ambientes domésticos e corporativos, tornando a recuperação de arquivos deletados uma necessidade cada vez mais relevante. O presente trabalho descreve a elaboração de uma ferramenta computacional para recuperação de arquivos deletados, desenvolvida em linguagem *Python* com auxílio da biblioteca *PyTSK3*, que fornece interface programática para o *The Sleuth Kit (TSK)*. A ferramenta opera sobre imagens de disco no formato .dd, percorrendo recursivamente a estrutura do sistema de arquivos FAT32 em busca de entradas marcadas como não alocadas, que correspondem a arquivos excluídos cujo conteúdo ainda não foi sobrescrito. O usuário pode visualizar a listagem dos arquivos encontrados e selecionar quais deseja recuperar, com os arquivos sendo salvos em um diretório de saída definido pelo próprio usuário. Foram realizados testes com arquivos dos formatos .txt, .jpg, .png, .bmp, .pdf, .docx, .pptx e .xlsx, com recuperação bem-sucedida em todos os casos. A ferramenta tem como foco principal a aplicação educacional, buscando aliar a robustez das técnicas forenses à simplicidade de uma implementação acessível, com potencial de expansão para contextos práticos em trabalhos futuros.

Palavras-Chave: Computação forense; *The Sleuth Kit*; *Python*; Recuperação de arquivos; FAT32

ABSTRACT

The growing reliance on digital devices for information storage has led to an increase in data loss in both home and corporate environments, making the recovery of deleted files an increasingly critical need. This paper describes the development of a computational tool for recovering deleted files, written in Python with the assistance of the PyTSK3 library, which provides a programming interface for The Sleuth Kit (TSK). The tool operates on disk images in .dd format, recursively traversing the FAT32 file system structure in search of entries marked as unallocated, which correspond to deleted files whose content has not yet been overwritten. The user can view the list of found files and select which ones to recover, with the files being saved to an output directory defined by the user. Tests were conducted with files in the .txt, .jpg, .png, .bmp, .pdf, .docx, .pptx, and .xlsx formats, with successful recovery in all cases. The tool's primary focus is educational application, aiming to combine the robustness of forensic techniques with the simplicity of an accessible implementation, with potential for expansion into practical contexts in future work.

Keywords: Computer forensics; The Sleuth Kit; Python; File recovery; FAT32

LISTA DE IMAGENS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A crescente dependência de dispositivos digitais para o armazenamento de informações tem intensificado a ocorrência de perda de dados, tanto em ambientes domésticos quanto corporativos. Situações como exclusão acidental de arquivos, falhas em sistemas, formatação indevida e corrupção de mídias de armazenamento são cada vez mais frequentes, evidenciando a necessidade de soluções eficientes para recuperação de dados.

A computação forense destaca-se como uma área essencial, uma vez que permite a análise detalhada de dispositivos digitais com o objetivo de recuperar, preservar e interpretar informações armazenadas. De acordo com Carrier (2005), a análise forense de sistemas de arquivos possibilita compreender como os dados são organizados e como podem ser recuperados mesmo após sua exclusão lógica.

Entre os sistemas de arquivos amplamente utilizados, o *File Allocation Table* (FAT32) apresenta grande relevância, especialmente em dispositivos removíveis, como *pendrives* e cartões de memória. Sua estrutura simplificada facilita a implementação e o uso, porém também torna os dados mais suscetíveis à sobrescrita após a exclusão, o que pode dificultar sua recuperação (MICROSOFT, 2000).

Embora existam diversas ferramentas disponíveis para recuperação de dados, muitas delas apresentam limitações, como alto custo, complexidade de uso ou falta de transparência quanto aos métodos empregados. Segundo Garfinkel (2007), ferramentas forenses baseadas em análise estrutural oferecem maior confiabilidade, pois atuam diretamente nos metadados e nos blocos de armazenamento dos sistemas de arquivos.

Dentre essas ferramentas, destaca-se o The Sleuth Kit (TSK), desenvolvido por Carrier (2005) e amplamente utilizado em investigações forenses digitais por sua robustez e precisão na análise de sistemas de arquivos. No entanto, sua utilização exige conhecimentos técnicos específicos, o que pode dificultar seu uso por usuários menos experientes.

Diante disso, o presente trabalho tem como objetivo apresentar o desenvolvimento de uma ferramenta para recuperação de arquivos deletados utilizando técnicas de computação forense, com base na linguagem Python e na biblioteca PyTSK3. A proposta busca aliar a robustez das ferramentas forenses à acessibilidade de uma aplicação mais simples, com potencial aplicação tanto em contextos educacionais quanto práticos.

1.1 Objetivos

1.1.1 Objetivo geral

Desenvolver uma ferramenta computacional para recuperação de arquivos deletados em imagens de disco.

1.1.2 Objetivos específicos

- Estudar os fundamentos da computação forense, com ênfase na estrutura do sistema de arquivos FAT32 e nos mecanismos de exclusão e recuperação de dados;
- Analisar as principais ferramentas de recuperação de arquivos existentes, identificando suas características, limitações e o contexto de uso de cada uma;
- Implementar uma ferramenta funcional capaz de percorrer a estrutura de um sistema de arquivos em busca de entradas marcadas como não alocadas e recuperar seu conteúdo;
- Utilizar técnicas de computação forense implementadas em linguagem *Python* com auxílio da biblioteca PyTSK3 para recuperar arquivos no formato *.dd;*
- Validar o funcionamento da ferramenta por meio de testes com diferentes formatos de arquivo, verificando a integridade dos dados recuperados.

1.2 Justificativa

O interesse pelo tema surgiu da afinidade do autor com a área da segurança da informação, especialmente no que diz respeito às técnicas e ferramentas utilizadas em computação forense digital. A recuperação de arquivos

deletados representa um problema prático e recorrente, tanto em contextos domésticos quanto profissionais, e compreender os mecanismos que a tornam possível é fundamental para qualquer profissional que atue na área.

A escolha da linguagem *Python* se deu pela familiaridade do autor com a linguagem, aliada à sua ampla adoção acadêmica e à disponibilidade de bibliotecas especializadas, como o PyTSK3, que viabilizou a integração com o *The Sleuth Kit* (TSK) sem a necessidade de desenvolvimento em linguagens de mais baixo nível. Essa escolha também contribui para o caráter educacional da ferramenta, uma vez que o Python é uma linguagem de fácil leitura e amplamente ensinada em cursos de computação.

1.3 Metodologia

O presente trabalho classifica-se como uma pesquisa aplicada, uma vez que tem como finalidade o desenvolvimento de uma solução prática para um problema real, a recuperação de arquivos deletados em dispositivos de armazenamento. Segundo Gil (2002), a pesquisa aplicada objetiva gerar conhecimentos voltados à aplicação prática e direcionados à solução de problemas específicos.

O desenvolvimento seguiu três etapas principais. Na primeira etapa foi realizada uma revisão bibliográfica sobre os fundamentos da computação forense, sistemas de arquivos e ferramentas existentes, com o objetivo de embasar teoricamente as decisões de implementação. Na segunda etapa foi realizado o desenvolvimento da ferramenta, com codificação iterativa em *Python* e integração com a biblioteca PyTSK3. Na terceira etapa foram conduzidos testes práticos sobre imagens de disco geradas a partir de um *pendrive* formatado em FAT32, com arquivos de diferentes formatos, avaliando a capacidade da ferramenta de identificar e recuperar corretamente os dados deletados.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Computação Forense

A computação forense é uma disciplina da segurança da informação voltada à coleta, preservação, análise e apresentação de evidências digitais, com o objetivo de reconstruir eventos ocorridos em sistemas computacionais. Segundo Carrier (2005), a análise forense de sistemas de arquivos permite compreender como os dados são organizados em dispositivos de armazenamento e como podem ser recuperados mesmo após sua exclusão lógica, tornando-se uma ferramenta essencial tanto em investigações criminais quanto em processos de recuperação de dados em ambientes corporativos e domésticos.

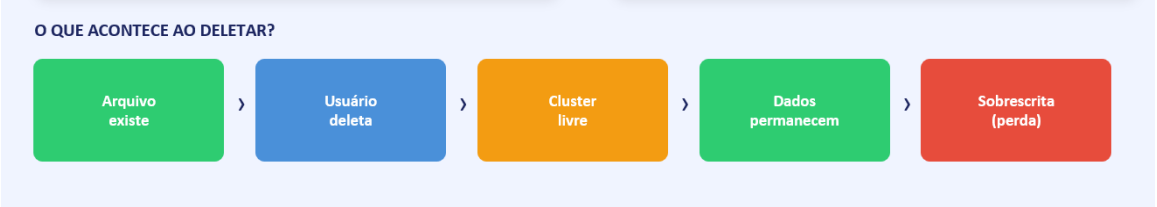
A área se consolidou a partir da crescente digitalização das atividades humanas, que passou a exigir métodos rigorosos para lidar com evidências em formato digital. Diferentemente da perícia tradicional, a computação forense demanda cuidados específicos para garantir a integridade dos dados analisados, uma vez que qualquer alteração no dispositivo original pode comprometer a validade das evidências obtidas. Por essa razão, a prática forense digital geralmente opera sobre cópias bit a bit dos dispositivos originais, conhecidas como imagens de disco, preservando o estado original da mídia.

2.2 Sistemas de Arquivos e o Processo de Exclusão

Para compreender como a recuperação de arquivos é possível, é necessário entender como os sistemas de arquivos gerenciam os dados armazenados. Um sistema de arquivos é uma estrutura lógica responsável por organizar, armazenar e recuperar arquivos em um dispositivo de armazenamento, controlando como os dados são gravados e localizados.

Quando um arquivo é excluído pelo usuário, o sistema operacional não apaga imediatamente seu conteúdo do dispositivo. Em vez disso, apenas marca o espaço ocupado como disponível para uso futuro, mantendo os dados fisicamente presentes até que sejam sobrescritos por novas informações.

Figura 1 - Processo de exclusão de arquivo FAT32



Fonte: Elaborado pelo autor

A figura 1 ilustra esse processo de forma simplificada: o arquivo existe normalmente até que o usuário o exclui, momento em que o cluster correspondente é marcado como livre na tabela de alocação. Os dados, no entanto, permanecem fisicamente no dispositivo até serem sobrescritos por novas informações, etapa que representa a perda definitiva do conteúdo.

Esse comportamento é o fundamento técnico que torna a recuperação de arquivos deletados possível e é explorado pelas ferramentas forenses para localizar e restaurar dados perdidos.

Entre os sistemas de arquivos mais relevantes para o presente trabalho, destaca-se o FAT32, amplamente utilizado em dispositivos removíveis como *pendrives* e cartões de memória. Sua estrutura é baseada em uma tabela de alocação de arquivos que registra quais clusters do dispositivo estão ocupados e quais estão livres. Ao excluir um arquivo, o FAT32 marca seus clusters como disponíveis na tabela de alocação, porém os dados permanecem intactos até serem sobrescritos, o que viabiliza sua recuperação por ferramentas especializadas (CARRIER, 2005).

2.3 Técnicas de Recuperação de Arquivos

A recuperação de arquivos deletados pode ser realizada por meio de duas abordagens principais: a análise de metadados e o *file carving*.

A análise de metadados consiste em examinar as estruturas internas do sistema de arquivos, como entradas de diretório e tabelas de alocação, em busca de registros de arquivos marcados como excluídos. Essa abordagem oferece maior precisão, pois utiliza informações estruturais do próprio sistema de arquivos para localizar e reconstruir os arquivos deletados. Garfinkel (2007) destaca que

ferramentas baseadas nessa técnica atuam diretamente nos metadados e blocos de armazenamento, oferecendo resultados mais confiáveis quando as estruturas do sistema de arquivos ainda estão íntegras.

O *file carving*, por sua vez, opera de forma independente das estruturas do sistema de arquivos, varrendo o dispositivo em busca de assinaturas conhecidas de tipos de arquivo, como os *bytes* iniciais que identificam um arquivo PDF (*Portable Document Format)* ou uma imagem JPEG (*Joint Photographic Experts Group)*. Essa técnica é útil quando os metadados foram corrompidos ou sobrescritos, mas tende a gerar mais falsos positivos e pode recuperar arquivos fragmentados de forma incompleta.

A ferramenta desenvolvida neste trabalho adota a abordagem de análise de metadados, por oferecer maior precisão e por ser mais adequada ao contexto educacional proposto.

3 FERRAMENTAS EXISTENTES

Diversas ferramentas para recuperação de arquivos deletados estão disponíveis atualmente, cada uma com características, limitações e públicos-alvo distintos. Esta seção apresenta as principais soluções existentes, contextualizando a proposta desenvolvida neste trabalho.

O Autopsy é uma plataforma forense de código aberto amplamente utilizada em investigações digitais. Oferece uma interface gráfica completa e suporta múltiplos sistemas de arquivos, combinando análise de metadados e *file carving*. Por sua abrangência, é considerado uma das ferramentas mais completas disponíveis gratuitamente, porém sua complexidade pode ser uma barreira para usuários iniciantes.

O Recuva, desenvolvido pela Piriform, é uma ferramenta voltada ao usuário doméstico, com interface simples e foco na recuperação de arquivos em sistemas *Windows*. Embora seja de fácil utilização, oferece pouca transparência sobre os métodos empregados internamente, o que limita seu uso em contextos acadêmicos e forenses.

O TestDisk e o PhotoRec são ferramentas de linha de comando multiplataforma e de código aberto. O TestDisk foca na recuperação de partições e estruturas de disco, enquanto o PhotoRec utiliza *file carving* para recuperar arquivos de diversos formatos. Ambos são amplamente utilizados por profissionais da área, mas exigem familiaridade com ambientes de terminal.

O FTK Imager, da AccessData, é uma ferramenta profissional voltada à criação e análise de imagens forenses. Possui versão gratuita com recursos limitados e é amplamente utilizado em investigações corporativas e policiais.

O EnCase, da OpenText, é uma solução comercial completa voltada ao mercado corporativo e governamental. Oferece recursos avançados de análise forense, porém seu alto custo a torna inacessível para uso educacional ou individual.

A Tabela 1 demonstra as principais características dessas ferramentas:

Tabela 1 - Comparativo entre softwares forenses

Ferramenta	Plataforma	Licença	Abordagem
Autopsy	Windows / Linux	Gratuita	Metadados + Carving
Recuva	Windows	Gratuita	Metadados
TestDisk / PhotoRec	Multiplataforma	Gratuita	Carving
FTK Imager	Windows	Gratuita (Limitada)	Metadados
EnCase	Windows	Paga	Completa

Fonte: Elaborado pelo autor.

Portanto, observa-se a falta de ferramentas que combinem simplicidade, transparência metodológica e potencial didático. É neste espaço que surge a proposta deste trabalho.

4 A FERRAMENTA DESENVOLVIDA

4.1 Ambiente de desenvolvimento

O desenvolvimento e os testes da ferramenta foram realizados em uma máquina equipada com processador *Ryzen 5 3400G* e 16GB memória RAM, operando sob o sistema *Windows* 11. Esta configuração mostrou-se suficiente para a execução das tarefas de varredura e recuperação sobre as imagens de disco utilizadas nos testes, não apresentando limitações de desempenho relevantes durante o desenvolvimento.

Como ambiente de desenvolvimento integrado foi utilizado o *PyCharm*, escolhido por sua robustez no suporte à linguagem de programação *Python* e por recursos como depuração integrada e análise estática de código, que auxiliaram na identificação e correção de erros durante o desenvolvimento. A linguagem utilizada foi o *Python* 3.14.3, em conjunto com a biblioteca *PyTSK3* na versão 20250801, instalada por meio do gerenciador de pacotes pip (Pip Install Packages).

4.2 Objetivos e Escopo

A ferramenta desenvolvida neste trabalho tem como objetivo realizar a recuperação de arquivos deletados a partir de imagens de disco no formato *.dd*, utilizando técnicas de análise de metadados do sistema de arquivos. O desenvolvimento foi realizado em *Python* 3, com auxílio da biblioteca *PyTSK3*, que fornece uma interface de alto nível para as funcionalidades do *The Sleuth Kit.*

O escopo da ferramenta é intencionalmente delimitado ao contexto educacional, priorizando clareza de funcionamento e simplicidade de uso em detrimento de funcionalidades avançadas presentes em soluções comerciais. A operação sobre imagens de disco, em vez de dispositivos físicos diretamente, garante a preservação da mídia original, prática fundamental na computação forense, que exige que a fonte de evidências permaneça inalterada durante a análise.

4.3 Tecnologias Usadas

O desenvolvimento da ferramenta baseou-se nas seguintes tecnologias:

- Python 3 foi escolhido como linguagem principal por sua legibilidade, ampla adoção acadêmica e vasta disponibilidade de bibliotecas voltadas para análise de dados e segurança da informação.
- *PyTSK3* é uma biblioteca *Python* que fornece *bindings* para o *The Sleuth Kit,* permitindo o acesso programático às estruturas internas de sistemas de arquivos. Por meio dela, é possível navegar por diretórios, inspecionar metadados de arquivos e identificar entradas marcadas como não alocadas, que correspondem aos arquivos deletados.
- *The Sleuth Kit* (TSK), desenvolvido por Brian Carrier, é um conjunto de ferramentas forenses de código aberto amplamente utilizado em investigações digitais. Sua robustez e precisão na análise de sistemas de arquivos o tornam uma base sólida para o desenvolvimento de ferramentas forenses (CARRIER, 2005).
- *Win32 Disk Imager* foi utilizado para a geração das imagens *.dd* dos dispositivos analisados durante os testes. A ferramenta realiza uma cópia bit a bit do dispositivo de armazenamento, preservando todas as estruturas do sistema de arquivos, incluindo entradas de arquivos deletados.

4.4 Arquitetura e funcionamento

A ferramenta é composta por quatro funções principais, cada uma com responsabilidade bem definida:

detectar_offset() - Realiza a leitura da tabela de partições da imagem de disco e retorna automaticamente o offset em setores da primeira partição válida encontrada. Caso não seja possível detectar automaticamente, o usuário é solicitado a informar o valor manualmente. Esta etapa é necessária pois o sistema de arquivos não começa necessariamente no início da imagem, há uma reservada para a tabela de partições antes da partição principal.

abrir_imagem() - Recebe o caminho da imagem *.dd* e o offset detectado, e instancia os objetos *Img_Info* e *FS_Info* do *PyTSK3*, que representam

respectivamente a imagem de disco e o sistema de arquivos nela contido. O offset é convertido de setores para bytes multiplicando-se por 512, que é o tamanho padrão de um setor.

listar_arquivos() - Percorre recursivamente a estrutura de diretórios do sistema de arquivos em busca de entradas cujos metadados possuam flag *TSK_FS_META_FLAG_UNALLOC*, que indica que o espaço ocupado pelo arquivo foi marcado como disponível pelo sistema operacional. Para cada arquivo encontrado, são coletados nome, caminho, tamanho e número de inode, armazenados em uma lista de dicionários retornada ao final da varredura.

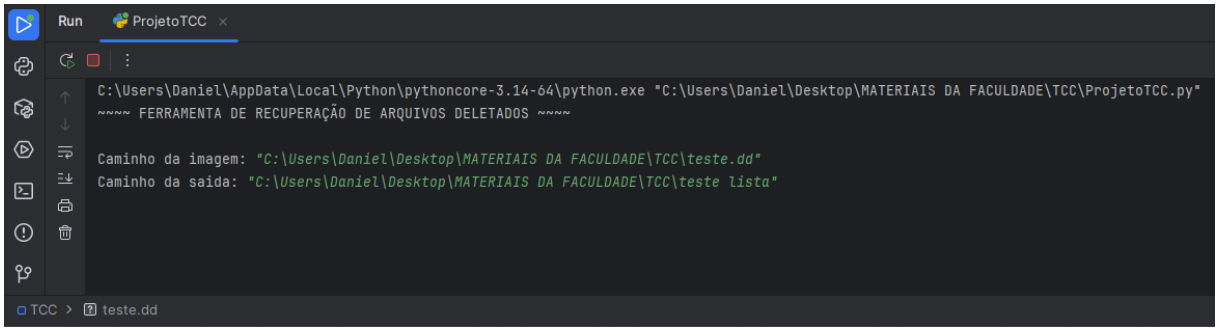
recuperar_arquivos() - Recebe as informações de um arquivo específico e realiza sua recuperação acessando diretamente seu inode por meio do método *open_meta()* do *PyTSK3.* Os dados são lidos em blocos de 1MB e escritos em um arquivo de saída no caminho definido pelo usuário. Caso dois arquivos recuperados gerem o mesmo nome, um contador numérico é adicionado automaticamente para evitar sobrescrita.

4.5 Fluxo de utilização

O fluxo de utilização segue as seguintes etapas:

1. O usuário informa o caminho da imagem *.dd* e o caminho da pasta de saída;

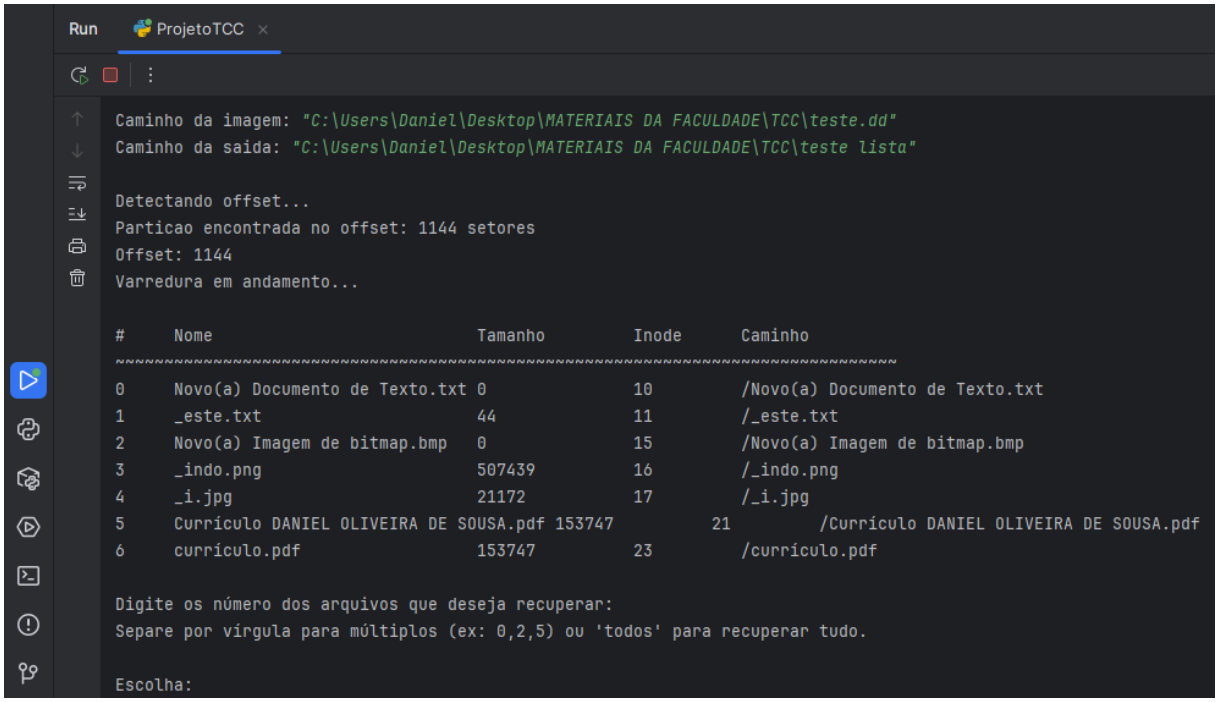
Figura 2 - Inserção de caminho de imagem e saída



Fonte: Elaborado pelo autor

2. A ferramenta detecta automaticamente o offset da partição;
3. A imagem é aberta e o sistema de arquivos é montado;
4. Uma varredura completa é realizada e os arquivos deletados são listados numericamente;

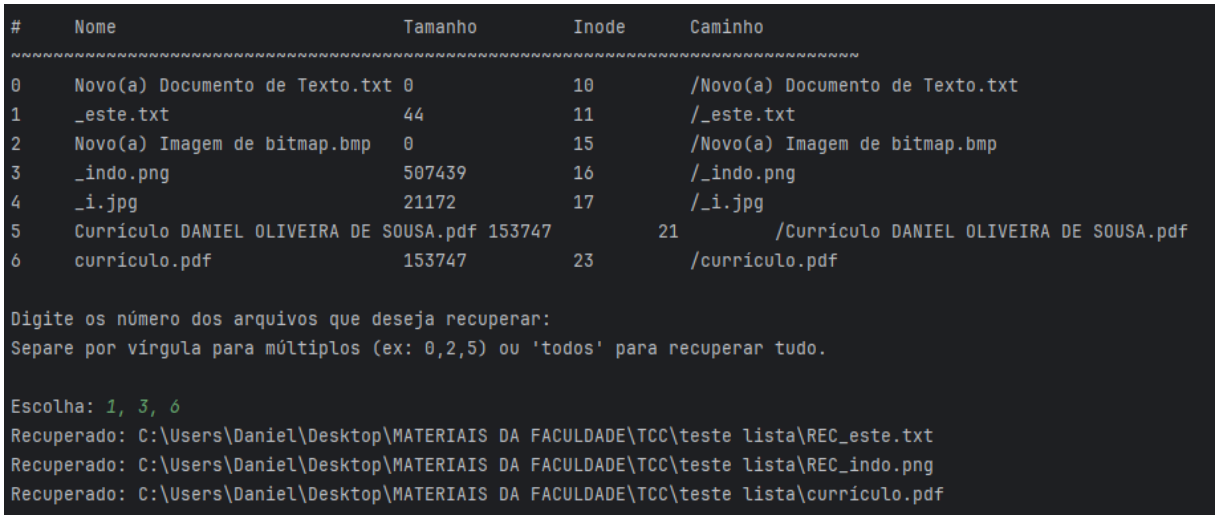
Figura 3 - Lista de arquivos deletados



Fonte: Elaborado pelo autor

5. O usuário seleciona quais arquivos deseja recuperar, podendo escolher individualmente por índice, múltiplos separados por vírgula ou todos de uma vez;

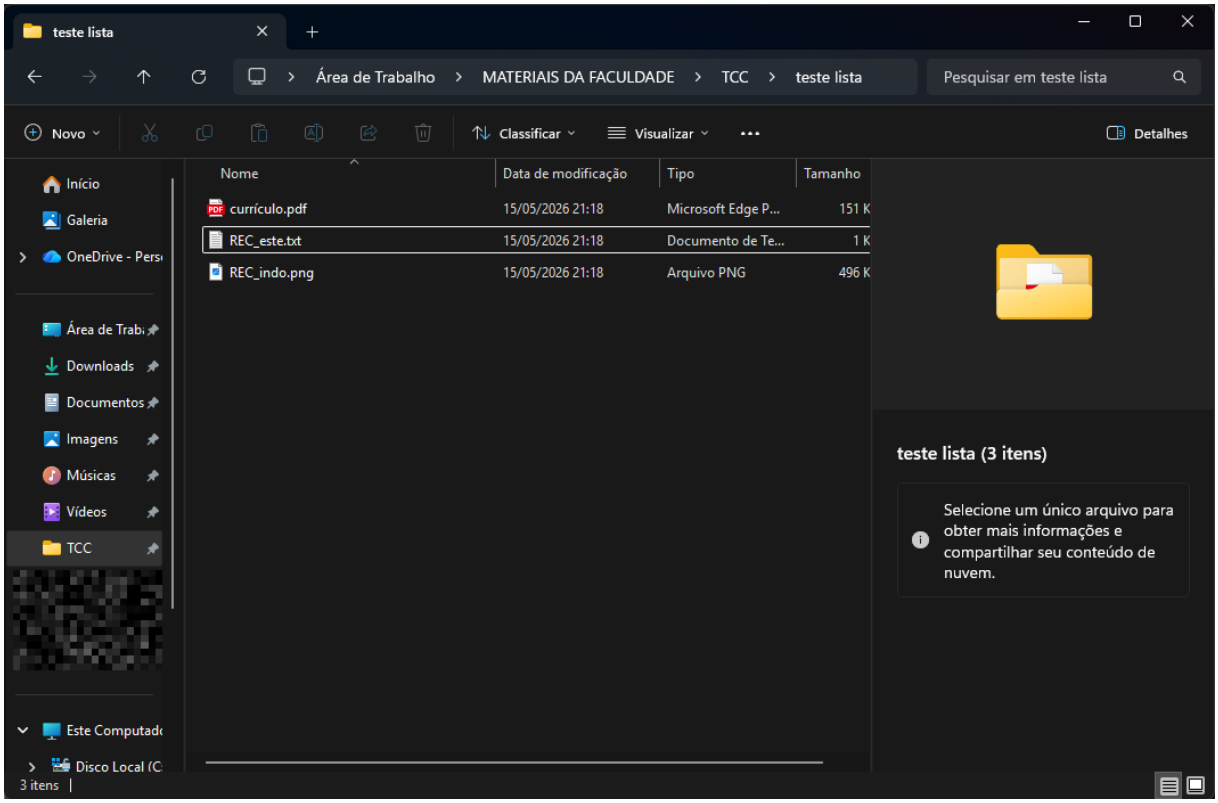
Figura 4 - Demonstração de recuperação de arquivos



Fonte: Elaborado pelo autor

6. Os arquivos selecionados são recuperados e salvos na pasta de saída.

Figura 5 - Arquivos recuperados



Fonte: Elaborado pelo autor

Figura 6 - Nova varredura após período de uso do dispositivo

```
Caminho da imagem: "C:\Users\Daniel\Desktop\MATERIAIS DA FACULDADE\TCC\Teste"
Caminho da saida: C:\Users\Daniel\Desktop\MATERIAIS DA FACULDADE\TCC\Demonstração

Detectando offset...
Particao encontrada no offset: 1144 setores
Offset: 1144
Varredura em andamento...

#     Nome                                Tamanho         Inode      Caminho
-------------------------------------------------------------------------------------------
0     _este.txt                           44              11         /_este.txt
1     _indo.png                           507439          16         /_indo.png
2     _i.jpg                              21172           17         /_i.jpg
3     Currículo DANIEL OLIVEIRA DE SOUSA.pdf 153747             21         /Currículo DANIEL OLIVEIRA DE SOUSA.pdf
4     currículo.pdf                       153747          23         /currículo.pdf
5     ~$teste.xlsx                        165             27         /~$teste.xlsx
6     _8B18200                            9563            28         /_8B18200
7     teste.xlsx                          9563            30         /teste.xlsx
8     _WRD0000.tmp                        13489           33         /_WRD0000.tmp
9     TESTE.docx                          13489           35         /TESTE.docx
10    _WRD0001.tmp                        13486           36         /_WRD0001.tmp
11    _WRL0002.tmp                        13489           37         /_WRL0002.tmp
12    TESTE.docx                          13486           39         /TESTE.docx
13    ~$TESTE.pptx                        165             43         /~$TESTE.pptx
14    TESTE.pptx                          33677           45         /TESTE.pptx

Digite os número dos arquivos que deseja recuperar:
Separe por vírgula para múltiplos (ex: 0,2,5) ou 'todos' para recuperar tudo.

Escolha: 0, 4, 7, 9, 12, 14
Recuperado: C:\Users\Daniel\Desktop\MATERIAIS DA FACULDADE\TCC\Demonstração\REC_este.txt
Recuperado: C:\Users\Daniel\Desktop\MATERIAIS DA FACULDADE\TCC\Demonstração\currículo.pdf
Recuperado: C:\Users\Daniel\Desktop\MATERIAIS DA FACULDADE\TCC\Demonstração\teste.xlsx
Recuperado: C:\Users\Daniel\Desktop\MATERIAIS DA FACULDADE\TCC\Demonstração\TESTE.docx
Recuperado: C:\Users\Daniel\Desktop\MATERIAIS DA FACULDADE\TCC\Demonstração\TESTE_1.docx
Recuperado: C:\Users\Daniel\Desktop\MATERIAIS DA FACULDADE\TCC\Demonstração\TESTE.pptx
```

Fonte: Elaborado pelo autor

4.6 Testes realizados

Para validar o funcionamento da ferramenta, foram realizados testes utilizando um pendrive formatado em FAT32, sobre o qual diferentes tipos de arquivo foram deletados deliberadamente. A imagem do dispositivo foi gerada com o *Win32 Disk Imager,* produzindo um arquivo *.dd* que serviu como entrada para a ferramenta.

Os tipos de arquivo testados foram: .txt, .bmp, .pdf, .jpg, .png, .docx, .xlsx e .pptx. Em todos os casos, a ferramenta foi capaz de identificar os arquivos deletados durante a varredura e recuperar seu conteúdo de forma íntegra, com os arquivos abrindo normalmente em seus respectivos programas após a recuperação.

Para demonstrar o efeito da sobrescrita de dados ao longo do tempo, foi realizada uma nova varredura do mesmo dispositivo após um período adicional de uso. Conforme observado na Figura 6, o arquivo “Novo(a) Imagem de bitmap.bmp”, presente na varredura anterior (Figura 3), não foi mais identificado na nova listagem. Isso ocorre porque os clusters anteriormente ocupados por esse arquivo foram sobrescritos por novas gravações realizadas no dispositivo, tornando seu conteúdo original irrecuperável.

4.7 Limitações

Por tratar-se de uma ferramenta de escopo educacional, algumas limitações são inerentes ao projeto em sua versão atual:

- A recuperação depende da integridade dos metadados do sistema de arquivos. Arquivos cujos setores foram sobrescritos por novos dados após a exclusão não podem ser recuperados, pois o conteúdo original não está mais presente na mídia.
- O sistema de arquivos FAT32, utilizado nos testes, pode corromper o nome original do arquivo no momento da exclusão, substituindo o primeiro caractere por um símbolo especial. Nesses casos, a ferramenta aplica o prefixo “REC_” ao nome recuperado, porém a extensão original pode ser perdida, exigindo renomeação manual pelo usuário.

- Durante testes com arquivos do Microsoft Office, observou-se que o FAT32 registra múltiplas entradas na tabela de alocação durante o ciclo de criação e edição desses arquivos, gerando entradas com tamanho zero e arquivos temporários de sessão, identificados pelos prefixos *"~$"* e extensões *".tmp"*.

5 CONSIDERAÇÕES FINAIS

O presente trabalho teve como objetivo o desenvolvimento de uma ferramenta computacional para recuperação de arquivos deletados em imagens de disco no formato *.dd,* utilizando técnicas de computação forense implementadas em linguagem *Python* com auxílio da biblioteca PyTSK3. Ao término do desenvolvimento, é possível afirmar que o objetivo geral foi plenamente atingido, a ferramenta é capaz de identificar arquivos deletados em sistemas de arquivos FAT32 e recuperar seu conteúdo de forma íntegra, conforme demonstrado pelos testes realizados com arquivos dos formatos .txt, .jpg, .png, .bmp, .pdf, .docl, .pptx e .xlsx.

Todos os objetivos específicos propostos foram alcançados. Os fundamentos da computação forense e da estrutura do sistema de arquivos FAT32 foram estudados e serviram de base teórica para as decisões de implementação. As principais ferramentas de recuperação existentes foram analisadas e contextualizadas, evidenciando a lacuna que justifica a proposta do trabalho. A ferramenta foi implementada com sucesso, percorrendo recursivamente a estrutura do sistema de arquivos em busca de entradas não alocadas e recuperando seu conteúdo. Por fim, os testes realizados validaram o funcionamento da ferramenta, confirmando a integridade dos arquivos recuperados.

Durante o desenvolvimento, alguns desafios técnicos se destacaram. A identificação incorreta do offset da partição foi um dos primeiros obstáculos encontrados, uma vez que o valor utilizado inicialmente não correspondia ao início real do sistema de arquivos na imagem de disco, o que impedia a abertura correta da imagem pelo PyTSK3. Outro desafio relevante foi a utilização incorreta da flag de identificação de arquivos não alocados, que inicialmente foi referenciada com nomenclatura equivocada, impedindo que os arquivos deletados fossem reconhecidos durante a varredura. Por fim, um erro de iteração sobre o objeto de diretório — onde se iterava sobre a string do caminho em vez do handle retornado pelo PyTSK3 — comprometia a varredura completa do sistema de arquivos. A identificação e correção desses problemas contribuiu significativamente para o aprofundamento do conhecimento técnico sobre o funcionamento interno das bibliotecas utilizadas.

5.1 Trabalhos futuros

Apesar dos resultados satisfatórios, a ferramenta apresenta limitações que abrem espaço para evoluções futuras. A detecção automática de extensão de arquivos por meio de assinatura de *bytes* representa o próximo passo mais imediato, permitindo identificar o tipo do arquivo pelos seus *bytes* iniciais independentemente do nome armazenado no sistema de arquivos.

O desenvolvimento de uma interface gráfica tornaria a ferramenta mais acessível a usuários sem familiaridade com ambientes de terminal, ampliando seu potencial de uso em contextos educacionais e práticos. A expansão do suporte a outros sistemas de arquivos, como NTFS e ext4, ampliaria significativamente o escopo de aplicação da ferramenta, permitindo sua utilização em dispositivos além de *pendrives* e cartões de memória formatados em FAT32. Por fim, a incorporação de um módulo para geração da própria imagem .dd diretamente pelo programa eliminaria a dependência de ferramentas externas como o *Win32 Disk Imager*, tornando o fluxo de trabalho mais completo e integrado.

REFERÊNCIAS

ÅRNES, André (Ed.). **Digital forensics**. John Wiley & Sons, 2017.

AUTOPSY. **Autopsy Digital Forensics**. Disponível em: https://www.autopsy.com. Acesso em: 2025.

CARRIER, Brian. **File system forensic analysis**. Addison-Wesley Professional, 2005.

CARVEY, Harlan; ALTHEIDE, Cory. **Digital forensics with open source tools**. Elsevier, 2011.

EXTERRO. **FTK Imager**. Disponível em: https://www.exterro.com. Acesso em: 30 abr. 2025.

GARFINKEL, Simson L. Carving contiguous and fragmented files with fast object validation. **Digital Investigation**, v. 4, p. 2-12, 2007.

GIL, Antonio Carlos. **Como elaborar projetos de pesquisa**. 4. ed. São Paulo: Atlas, 2002.

MICROSOFT. **FAT32 File System Specification**. 2000. Disponível em: https://www.cs.fsu.edu/~cop4610t/assignments/project3/spec/fatspec.pdf. Acesso em: 2025.

NELSON, Bill et al. **Guide to computer forensics and investigations**. Course Technology Cengage Learning, 2010.

OPENTEXT. **EnCase**. Disponível em: https://www.opentext.com. Acesso em: 30 abr. 2025.

PIRIFORM. **Recuva**. Disponível em: https://www.ccleaner.com/recuva. Acesso em: 30 abr. 2025.

PY4N6. **pytsk — Python bindings for The Sleuth Kit**. Disponível em: https://github.com/py4n6/pytsk. Acesso em: 2025.

RUBARSKY, Kelsey Laine. **Digital Forensics Research Paper**. Marshall University, 2012. Disponível em: https://www.marshall.edu/forensics/files/RusbarskyKelsey_Research-Paper-Summer-2012.pdf. Acesso em: 2025.

SLEUTH KIT. **The Sleuth Kit**. Disponível em: https://www.sleuthkit.org. Acesso em: 30 abr. 2025.

TESTDISK. **TestDisk & PhotoRec**. Disponível em: https://www.cgsecurity.org. Acesso em: 2025.

APÊNDICE – CÓDIGO FONTE

FileRec.py

```
#Importação de bibliotecas necessárias
import pytsk3
import os

def abrir_imagem(caminho_imagem, offset=0):
    #Abre uma imagem para análise forense
    try:
        img = pytsk3.Img_Info(caminho_imagem)
        fs = pytsk3.FS_Info(img, offset * 512)  #offset em
setores convertido para bytes
        return fs
    except Exception as e:
        print(f"Erro ao abrir imagem: {e}")
        return None

def detectar_offset(caminho_imagem):
    #Detecta o offset da partição  em setores
    try:
        img = pytsk3.Img_Info(caminho_imagem)
        volume = pytsk3.Volume_Info(img)
        for particao in volume:
            if particao.len > 2048:
                print(f"Particao encontrada no offset:
{particao.start} setores")
                return particao.start
        print("Nenhuma particao válida encontrada")
        return None
    except Exception as e:
        print(f"Erro ao detectar offset: {e}")
        return None

def listar_arquivos(fs, diretorio='/', lista=None):
    #Percorre a imagem e retorna lista de arquivos
    if lista is None:
        lista = []
    dir_handle = fs.open_dir(diretorio)
    for entry in dir_handle:
        if not hasattr(entry, "info") or not
hasattr(entry.info, "name"):
            continue
        nome =
entry.info.name.name.decode(errors='ignore')
        if nome in [".",".."]:
            continue
        caminho_completo = os.path.join(diretorio, nome)

        #Recursão se for diretório
```

```
        if entry.info.name.type ==
pytsk3.TSK_FS_NAME_TYPE_DIR:
            try:
                listar_arquivos(fs, caminho_completo,
lista)
            except Exception:
                pass
            continue

        #Se o arquivo for deletado, adiciona na lista
        if (entry.info.name.type ==
pytsk3.TSK_FS_NAME_TYPE_REG and
                entry.info.meta and
                entry.info.meta.flags &
pytsk3.TSK_FS_META_FLAG_UNALLOC):

            #Elimina arquivos com tamanho 0
            if entry.info.meta.size == 0:
                continue

            lista.append({
                "nome": nome,
                "caminho": caminho_completo,
                "tamanho": entry.info.meta.size,
                "inode": entry.info.meta.addr
            })
    return lista

def recuperar_arquivos(fs, arquivo, pasta_saida):
    #Recupera um único arquivo através do seu inode
    nome = arquivo["nome"]
    inode = arquivo["inode"]
    tamanho = arquivo["tamanho"]

    if nome[0] in ("_", "?"):
        nome = "REC_" + nome[1:]

    try:
        file_obj = fs.open_meta(inode)
        caminho_saida = os.path.join(pasta_saida, nome)

        if os.path.exists(caminho_saida):
            base, ext = os.path.splitext(nome)
            contador = 1
            while os.path.exists(caminho_saida):
                nome = f"{base}_{contador}{ext}"
                caminho_saida = os.path.join(pasta_saida,
nome)
                contador += 1

        with open(caminho_saida, 'wb') as out:
```

```
            CHUNK = 1024*1024
            lido = 0
            while lido < tamanho:
                restante = min(CHUNK, tamanho - lido)
                data = file_obj.read_random(lido,
restante)
                if not data:
                    break
                out.write(data)
                lido += len(data)
        print(f"Recuperado: {caminho_saida}")
    except Exception as e:
        print(f"Erro ao recuperar arquivo {nome}: {e}")


if __name__ == "__main__":
    print("~~~~ FERRAMENTA DE RECUPERAÇÃO DE ARQUIVOS
DELETADOS ~~~~\n")

    #Pedido ao usuário no terminal
    caminho_imagem = input("Caminho da imagem:
").strip('"')
    caminho_saida = input("Caminho da saida: ").strip('"')

    print("\nDetectando offset...")
    offset = detectar_offset(caminho_imagem)

    if offset is None:
        print("Não foi possível detectar o offset")
        offset = int(input("Digite o offset manualmente:
"))
    else:
        print(f"Offset: {offset}")

    fs = abrir_imagem(caminho_imagem, offset)

    if not fs:
        print("Falha ao abrir imagem. Verifique o caminho
e o offset.")
    else:
        print("Varredura em andamento...\n")
        arquivos = listar_arquivos(fs)
        if not arquivos:
            print("Nenhum arquivo encontrado.")
        else:
            print(f"{'#':<5} {'Nome':<30} {'Tamanho':<15}
{'Inode':<10} Caminho")
            print("~" * 80)
            for i, arq in enumerate(arquivos):
                print(f"{i:<5} {arq['nome']:<30}
{arq['tamanho']:<15} {arq['inode']:<10} {arq['caminho']}")
```

```
            print("\nDigite os número dos arquivos que
deseja recuperar:")
            print("Separe por vírgula para múltiplos (ex:
0,2,5) ou 'todos' para recuperar tudo.\n")

            escolha = input("Escolha: ").strip().lower()
            os.makedirs(caminho_saida, exist_ok=True)

            if escolha == "todos":
                for arq in arquivos:
                    recuperar_arquivos(fs, arq,
caminho_saida)
            else:
                try:
                    indices = [int(x.strip()) for x in
escolha.split(",")]
                    for i in indices:
                        if 0 <= i < len(arquivos):
                            recuperar_arquivos(fs,
arquivos[i], caminho_saida)
                        else:
                            print(f"Indice {i} inválido,
ignorado.")
                except ValueError:
                    print("Entrada inválida. Use números
separados por vírgula ou 'todos'.")
```