INSTITUTO FEDERAL DE RONDÔNIA
Campus PORTO VELHO CALAMA
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

BRUNO DA SILVA CHAVES

ORGANIZAÇÃO DE DOCUMENTOS DIGITALIZADOS COM O USO DA TECNOLOGIA OCR

PORTO VELHO
2025

BRUNO DA SILVA CHAVES

ORGANIZAÇÃO DE DOCUMENTOS DIGITALIZADOS COM O USO DA TECNOLOGIA OCR

Artigo entregue como Trabalho de Conclusão de Curso ao Instituto Federal de Educação, Ciência e Tecnologia de Rondônia (IFRO), *Campus* Porto Velho Calama, como requisito parcial para obtenção do grau de tecnólogo, junto ao Curso Análise e desenvolvimento de sistemas, sob a orientação do professor Fernando Dall Igna.

PORTO VELHO
2025

Ficha catalográfica elaborada pelo Sistema Gerador de Ficha Catalográfica do IFRO.

Chaves, Bruno da Silva.
Organização de documentos digitalizados com o uso da tecnologia OCR / Bruno da Silva Chaves. - Porto Velho, 2025.
24 f. : il.

Orientador(a): Prof. Dr Fernando Dall Igna.

Trabalho de Conclusão de Curso (Superior de Tecnologia em Análise e Desenvolvimento de Sistemas) – Instituto Federal de Educação, Ciência e Tecnologia de Rondônia - IFRO, Porto Velho, 2025.

1. Automatizar. 2. Organizar. 3. Renomear. 4. Tesseract. I. Igna, Fernando Dall (orient.). II. Instituto Federal de Educação, Ciência e Tecnologia de Rondônia - IFRO. III. Título.

Bibliotecário(a) Responsável: Miria Santana Veiga, CRB-11/898

BRUNO DA SILVA CHAVES

ORGANIZAÇÃO DE DOCUMENTOS DIGITALIZADOS COM O USO DA TECNOLOGIA OCR

Artigo entregue como Trabalho de Conclusão de Curso ao Instituto Federal de Educação, Ciência e Tecnologia de Rondônia (IFRO), Campus Porto Velho Calama, como requisito parcial para obtenção do grau de tecnólogo, junto ao Curso Análise e desenvolvimento de sistemas, sob a orientação do professor Fernando Dall Igna.

Porto Velho/RO, 05/12/2025.

Comissão Examinadora

gov.br Documento assinado digitalmente
FERNANDO DALL IGNA
Data: 12/12/2025 09:58:01-0300
Verifique em https://validar.iti.gov.br

Prof. Fernando Dall'Igna - IFRO
(Orientador)

LEANDRO FERRAREZI VALIANTE:82694095291

Assinado de forma digital por LEANDRO FERRAREZI VALIANTE:82694095291
Dados: 2025.12.10 21:24:32 -04'00'

Prof. Leandro Ferrarezi Valiante - IFRO

gov.br Documento assinado digitalmente
CELSO GUEDES GOMES
Data: 11/12/2025 16:00:55-0300
Verifique em https://validar.iti.gov.br

Prof. Celso Guedes Gomes - IFRO

ORGANIZAÇÃO DE DOCUMENTOS DIGITALIZADOS COM O USO DA TECNOLOGIA OCR

RESUMO: Atualmente, temos à disposição diversas ferramentas tecnológicas para auxiliar com as atividades que fazemos no cotidiano.Tratando-se do ambiente de trabalho, ferramentas tecnológicas podem auxiliar e automatizar algumas tarefas que são executadas rotineiramente poupando esforço manual. Neste estudo, abordaremos o desenvolvimento de um software para automatizar a organização de documentos digitalizados. O objetivo é desenvolver um *software* de interface gráfica *web* que utiliza OCR para renomear, organizar e reconhecer textos em arquivos de documentos digitalizados. Para o reconhecimento do texto foi utilizado o software de código aberto *Tesseract*. Também foi utilizado a linguagem de programação *Python* para manipular o texto dos arquivos. A interface gráfica do usuário foi desenvolvida com o *Framework Flutter*. Para verificar a viabilidade de desenvolver o projeto proposto, foi realizado um teste que será abordado adiante, no qual utilizou-se a ferramenta *Tesseract* para extrair o texto de alguns documentos digitalizados, como a funcionalidade principal do software a ser desenvolvido é renomear arquivos digitalizados com base em um ou mais padrões presentes no próprio documento. Consideramos como fator principal para taxa de sucesso do teste o reconhecimento ou não de palavras como data e nome, presentes em cada documento analisado. Os dados verificados nos testes foram satisfatórios, apresentando uma taxa de erro de apenas 10% com os demais resultados sendo 40% considerado bom e 50% considerado ótimo. Os critérios de classificação do teste serão apresentados adiante.

PALAVRAS-CHAVE: Automatizar. Organizar. Renomear. *Tesseract*.

ABSTRACT: Currently, we have various technological tools available to assist with the activities we perform daily. In the workplace, technological tools can assist and automate some routine tasks, saving manual effort. In this study, we will address the development of software to automate the organization of digitized documents. The objective is to develop a web graphical interface software that uses OCR (Optical Character Recognition) to rename, organize, and recognize text in digitized document files. The open-source software Tesseract was used for text recognition, and the Python programming language was also used to manipulate the text within the files. The user graphical interface was developed using the Flutter Framework. To verify the feasibility of developing the proposed project, a test was carried out, which will be discussed later. This test utilized the Tesseract tool to extract text from some digitized documents, as the main functionality of the software to be developed is to rename digitized files based on one or more patterns present in the document itself. We considered the recognition or non-recognition of words such as date and name, present in each analyzed document, as the main factor for the test's success rate. The data verified in the tests were satisfactory, showing an error rate of only 10%, with the remaining results being 40% considered good and 50% considered excellent. The test classification criteria will be presented later.
KEYWORDS: Automate. Organize. Rename. *Tesseract*.

1 INTRODUÇÃO

Atualmente temos à disposição diversas ferramentas tecnológicas para auxiliar com as atividades que fazemos no cotidiano. Se tratando do ambiente de trabalho, ferramentas tecnológicas podem auxiliar e automatizar algumas tarefas que são executadas rotineiramente.

Entre as atividades que podem ser comum em algumas empresas públicas e privadas (podendo ser também em atividades pessoais) é a digitalização de documentos. Estes documentos digitalizados podem ser dos mais variados tipos, como por exemplo, carteira nacional de habilitação (CNH), cadastro de pessoa física (CPF), contratos de compra e venda, contratos de aluguéis, notas fiscais entre outros.

Documentos digitalizados podem ser salvos em um computador em formato de arquivo PDF ou arquivos de imagem como o JPEG. Esses arquivos gerados por sua vez podem ser armazenados em disco compartilhado na rede e/ou *cloud*, nos próprios computadores das empresas, ou em serviços de armazenamento em nuvem como Google Drive ou Onedrive, por exemplo.

Com o passar do tempo, dias, semanas e meses, o volume de documentos citados anteriormente pode gerar uma quantidade substancial de dados (a depender da quantidade gerada diariamente) tornando trabalhoso e pouco produtivo analisar e filtrar as informações nesses documentos de forma manual. Este trabalho propõe um sistema para ajudar a consultar informações em tais documentos.

O sistema foi desenvolvido com a linguagem de programação *Python*, enquanto a interface *web* foi desenvolvida com o *Framework Flutter,* a interface *web* por ser acessível em diversos dispositivos, dispensando a necessidade de instalar softwares adicionais para o uso do sistema. Outro ponto a se destacar é o fato de que para a utilização do sistema em empresas, ele pode ser configurado localmente na própria empresa, para armazenar e consultar os documentos gerados no cotidiano. Considerando as diretrizes da lei geral de proteção de dados (LGPD), tratar tais documentos localmente pode ser mais adequado, para isso a solução proposta neste trabalho dispensa o compartilhamento do sistema publicamente na internet.

Pensando em mitigar ou minimizar o trabalho manual ao analisar tais dados, este projeto tem o objetivo de desenvolver um software *web* com a funcionalidade

principal de extrair texto de imagens com a tecnologia *Optical Character Recognition* (OCR Reconhecimento Óptico de Caracteres), e a partir dos dados extraídos com o OCR proporcionar funcionalidades como: copiar o texto de documentos no formato de imagem ou PDF, gerar uma planilha com as informações de texto e padrões presentes nesses documentos, renomear esses arquivos considerando padrões de texto presentes nos documentos, tais padrões de texto são informados pelo usuário. A interface gráfica deverá permitir a seleção dos documentos a serem processados.

Após o reconhecimento dos textos, o usuário pode escolher exportar esses dados nos seguintes formatos: em um novo arquivo PDF no qual será possível copiar o texto gerado, ou em formato de planilha Excel, sendo o formato de planilha indicado para um grande volume de dados, o que permite a consulta desses dados de forma rápida e centralizada.

Embora a aplicação possa ser usada para um propósito geral, o foco principal do desenvolvimento terá como base, imagens de documentos como notas fiscais, fichas de EPIs, ou itens semelhantes. Tais documentos possuem elementos em comum, mas também contém características individuais, como data, hora, nome das pessoas envolvidas na transação e outros elementos que podem categorizá-los.

1.1. Objetivos

1.1.1. Objetivo Geral

O objetivo central deste trabalho é desenvolver e implementar um sistema *web* que sirva como uma plataforma para leitura e extração de texto de documentos em formato PDF utilizando a biblioteca *Tesseract*.

1.1.2. Objetivos Específicos

Desta forma, para alcançar o objetivo geral, o trabalho se propõem a:

1. Realizar revisão bibliográfica sobre o tema;
2. Desenvolver sistema *web* utilizando *Framework Flutter*;
3. Implementar funcionalidade de Importação e exportação de PDF;
4. Implementar funcionalidade de Importação e exportação de Imagem;

5. Implementar funcionalidade de Importação e exportação de Planilha Eletrônica no formato XLSX;
6. Implementar funcionalidade de exportação de documentos compactados no formato ZIP;
7. Implementar funcionalidade de busca por palavras-chave, e organização dos documentos importados utilizando tecnologia OCR;

2 FUNDAMENTAÇÃO TEÓRICA

1.2. Reconhecimento Óptico de Caracteres

Os primeiros aparelhos para se trabalhar com o OCR surgiram na década de 1929, o equipamento se chamava “máquina de leitura”, criado pelo engenheiro austríaco Gustav Tauschek. Em 1974 a empresa *Kurzweil Computer Products*, Inc., tinha um produto capaz de reconhecer textos impressos em vários tipos de fontes diferentes, mais tarde o fundador da empresa (Ray Kurzweil) criou uma máquina que era capaz de ler textos em voz alta a partir da conversão de texto em fala.

Mas foi a partir da década de 1990 que essa tecnologia se tornou mais popular, um dos fatores que contribuíram para essa popularização foi a digitalização de jornais históricos.

A tecnologia OCR serve para reconhecer textos a partir de documentos digitalizados ou imagens, após o reconhecimento do texto o conteúdo pode ser convertido em documento editável. Com a conversão de documentos em texto editável, é possível analisar os dados de tais documentos de forma ágil. A seguir descrevemos algumas etapas do processo OCR.

As principais etapas do reconhecimento óptico são: escaneamento ótico, pré-processamento e extração de características.

Consistem em alterar o contraste do documento para preto e branco, os campos do documento que contém o texto serão atribuídos o valor preto, os campos que não contém texto serão atribuídos o valor branco, nessa etapa podem ocorrer erros por parte do software o que impacta na precisão durante a detecção de textos. Esse método é conhecido como *Iterative Global Thresholding* (IGT). O algoritmo tem como pressuposto o fato de que geralmente os *pixels* que contém o texto (*pixels* pretos) raramente ultrapassam 10% da quantidade total de *pixels* da imagem. O

processo de separar as partes da imagem que contém caracteres de texto das partes que não contém texto (*pixels* brancos) é fundamental no processo como um todo.

Nessa etapa o algoritmo determina as partes da imagem onde os dados foram impressos, o que inclui distinguir textos, figuras e gráficos. Outro passo importante desse processo é calcular a diferença entre os *pixels* que contém texto (caracteres impressos) dos *pixels* de fundo (que não contém caracteres impressos). O processo de segmentação pode ser dividido em implícito e explícito, sendo que a segmentação explícita tenta separar as imagens das palavras nas bordas dos caracteres, essa técnica é conhecida como "*oversegmentation*". Já a segmentação implícita corta a imagem em vários pedaços de comprimentos iguais. Os pedaços, cada qual representado por seu vetor de características são agrupados em caracteres por reconhecimento (CHERIET, 2007).

No entanto, durante o processo de segmentação podem ocorrer alguns problemas que podem comprometer a taxa de sucesso em detectar os caracteres e consequentemente os textos presentes no arquivo de imagem. Veremos adiante os principais problemas desse processo de acordo com (EIKVIL, 1993).

Vários caracteres podem ser interpretados como um único caractere, acentos e pontos podem ser confundidos com ruído e vice-versa, interpretação incorreta de gráficos e geometria, o texto pode não ser passado para o estágio de reconhecimento.

Na etapa seguinte são capturadas as características de cada caractere individualmente, são avaliados elementos como ruído, distorções, variação de estilo, translação, rotação. Também é verificada a distribuição dos *pixels*, distorções, e análise estrutural, para determinar elementos como ranhuras, voltas, pontos de término e intersecções de linhas.

Existem vários aplicativos que usam OCR para detecção de texto em documentos digitalizados, algumas aplicações populares são:

Tesseract Open Source OCR *Engine*, *software* de código aberto criado entre os anos de 1985 e 1994 nos Estados Unidos da América, foi desenvolvido e mantido pela empresa Google de 2006 a 2018, o código fonte do *Tesseract* está disponível no link: https://github.com/tesseract-ocr/tesseract; *gImageReader* que é um *front-end*

Gtk/Qt simples para *Tesseract* OCR, o código fonte do *gImageReader* está disponivel no link: https://github.com/manisandro/gImageReader;

Adobe Acrobat popular leitor de PDF com várias funcionalidades, também disponibiliza o reconhecimento OCR em documentos, a licença de uso pode ser adquirida no link: https://www.adobe.com/br;

Neste trabalho, foi utilizado a ferramenta *Tesseract* para extrair textos de documentos e verificar a viabilidade de renomear tais arquivos com base em um padrão de texto extraído do documento. O objetivo aqui não foi o conteúdo do texto em si, mas padrões presentes em documentos que tornassem possível renomear os arquivos em questão.

Podemos destacar que, mesmo que a extração dos textos não seja absoluta, o programa pode lograr êxito na execução do objetivo principal.

3 METODOLOGIA

Nesta seção, abordaremos os métodos e algoritmos estudados e implementados no desenvolvimento do sistema proposto por este trabalho, os principais tópicos a serem abordados adiante explicam detalhadamente o processamento dos dados durante o processo de desenvolvimento.

1.3. Construção do banco de imagens e documentos

Para implementação prática dos testes, criamos um banco de imagens e documentos PDF, ambos digitalizados em um *Scanner* de impressora da marca *Samsung* modelo SL-M4070FR, sendo 10 documentos em formato PDF e 10 em formato de imagem.

Os documentos são papéis de fichas de Equipamento de proteção individual (EPI) entregues aos colaboradores de uma empresa. Tais EPIs são entregues rotineiramente a partir de um determinado período, ou por solicitação dos próprios colaboradores. Cada ficha de EPI contém campos para descrição dos equipamentos que são entregues em cada ato de entrega, podendo ser por exemplo botina, calça, camisa, luva entre outros, incluindo os respectivos tamanhos de cada equipamento entregue. Visualiza-se na Figura a seguir (Figura 1) o modelo padrão de cada ficha impressa para registrar a entrega.

A ficha de EPI apresentada na Figura 1 possui um cabeçalho com vários campos, sendo que alguns são de informações individuais de cada colaborador como nome, matrícula e departamento, enquanto outros campos são detalhes referentes a cada entrega, como a data e os tipos de equipamentos entregues. Nas informações da Figura 1 observamos que foi entregue uma botina tamanho 42 para o colaborador Pedro da Silva no dia 21 de setembro de 2023.

Figura 1 — Modelo de uma ficha de EPI.

FICHA DE CONTROLE DE EPI

MATRICULA	NOME DO COLABORADOR	FUNÇÃO	DEPARTAMENTO	LOCAL ENTREGA
123456	PEDRO DA SILVA	AUX COMERCIAL	DESC	Porto Veiho

Declaro que recebi os equipamentos de proteção individuais (EP.I. relacionados nesta ficha,pelos quais assume total responsabilidade conforme NR-6 Portaria MTb n.7 877,de 24 de outubro de 2018:comprometendo-me a usá-lo apenas para a finalidade a que se destina,responsabilizo-me por aguarda,conservacão e higienização,comunicação ao empregador qualquer alteração que o tome impróprio para o uso, cumprir as determinacões do empregador sobre o uso adequado, estou ciente que, a recusa injustificada do uso dos mesmos, estarei sujeito as penalidades previstas em lei.

		Porto Veho/RO-quinta-fera, 21 de setenbro de 2023.
Responsável Imediato	assinatura do Colaborador	Data Registro Inicial

ITEM	EQUIPAMENTO FORNECID0	C.A	QUANT	RECEBMENTO			DEVOLUCAO		
				DATA	RUBRICA COLAB.	RUBRICA RESP TREGA	DATA	RUBRICA COLAB.	RUBRICA RESP.
1	BOTINA TAM 42			21/09					
		-	--	-	-	--	-	-	-

Fonte: o próprio autor.

Esses elementos textuais servem como base para os testes de reconhecimento de texto, distinção e organização das fichas de EPIs.

1.4. Desenvolvimento dos módulos para leitura extração de texto

Nesta etapa, criamos dois módulos com a linguagem de programação *Python*, sendo um para ler e gravar dados nos arquivos que foram utilizados no sistema, e outro para o reconhecimento de texto em tais arquivos. O objetivo dessa etapa foi criar um padrão para realizar as operações descritas anteriormente. Destacamos também que aproveitamos diversos módulos de código aberto que estão disponíveis gratuitamente no *Github* (O *Github* é uma plataforma de hospedagem e versionamento de código baseada na *web*).

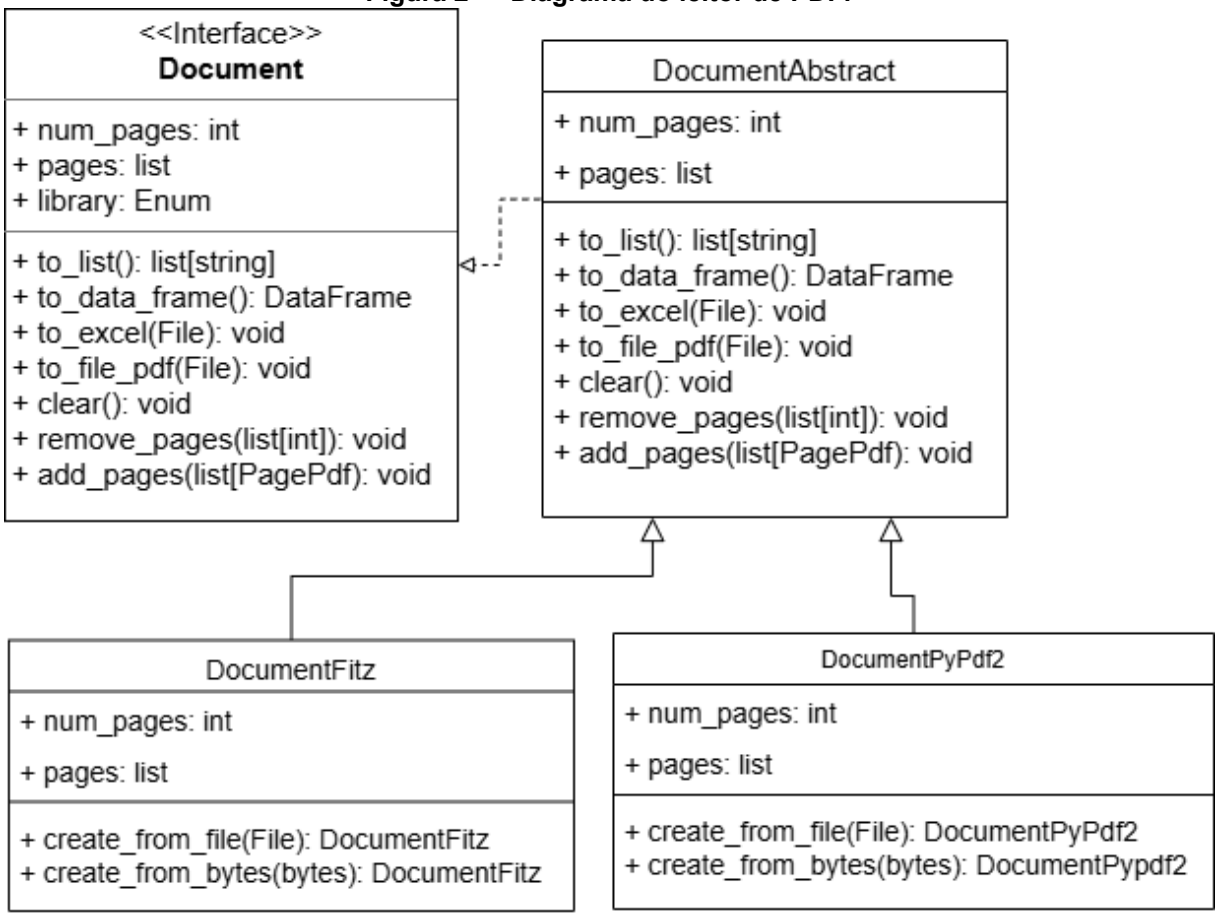
Alguns dos módulos utilizados foram *Pytesseract* para o reconhecimento de texto nas imagens, *Fitz* e *Pypdf* para ler e salvar arquivos em formato PDF no sistema de arquivos, *Pandas* para filtrar texto e salvar os dados no formato de planilhas, *Opencv* para processar imagens, entre outros. Os demais módulos instalados durante o desenvolvimento estão descritos detalhadamente no arquivo *requirements.txt* na pasta do projeto junto com os demais arquivos, sendo que o código fonte completo está disponível em um repositório do *Github*.

A seguir, detalhamos a estrutura dos módulos citados anteriormente, ambos desenvolvidos com o paradigma da orientação a objetos e o uso dos padrões de projeto, *Adapter,Singleton, Iterator* entre outros. A escolha por padrões de projeto no desenvolvimento do sistema se justifica para evitar grandes alterações no código do sistema, em alguns casos como implementação de novas funcionalidades, ou ajustes futuros, caso algum dos módulos de código aberto que foram utilizados fiquem obsoletos ou apresentem alguma vulnerabilidade de segurança, por exemplo.

3.1.1. Leitor de documentos

A Figura 2 apresenta um diagrama com as principais funcionalidades do leitor de PDF, que pode ler arquivos no disco rígido do computador ou os *bytes* de um arquivo. Após a leitura do documento são disponibilizadas algumas operações como a leitura de todas as páginas do documento, a extração do texto bruto de uma ou mais páginas do documento, a Figura 2 também apresenta a interface para trabalhar com uma página do documento. O leitor de documentos também permite adicionar ou remover páginas e salvar o documento final e um arquivo no disco rígido ou exportar os *bytes* do documento final.

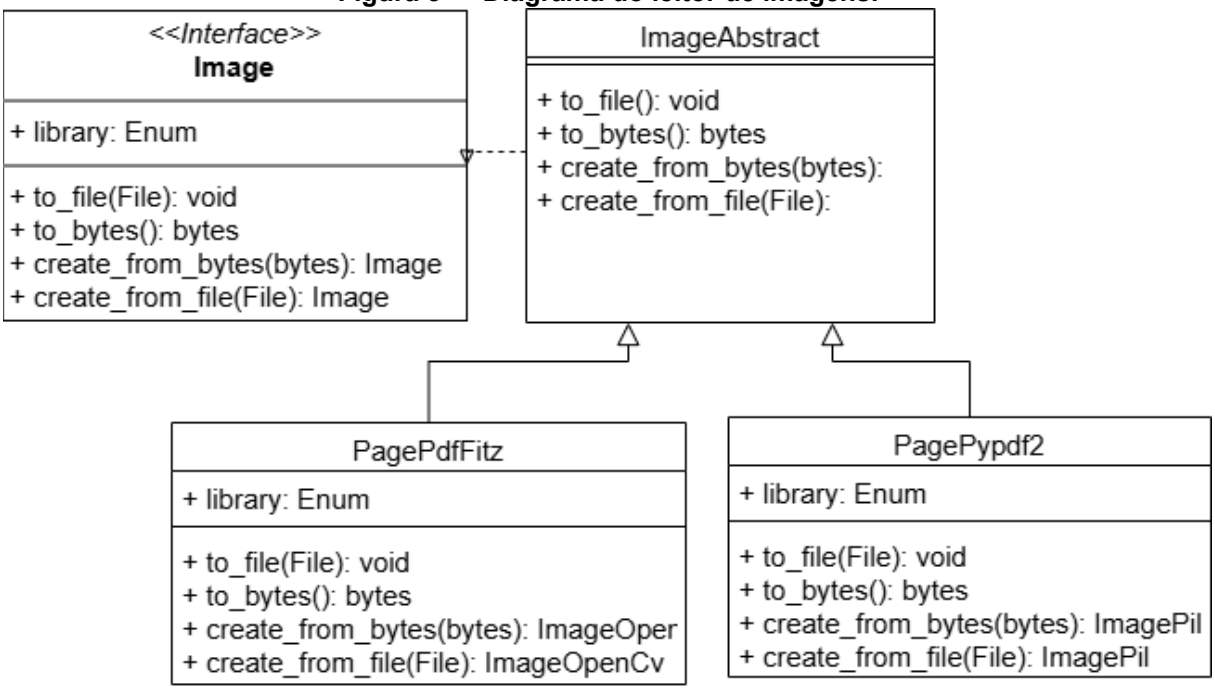
Figura 2 — Diagrama do leitor de PDF.



Fonte: o próprio autor.

A Figura 3 apresenta o diagrama do leitor de imagens que faz parte do módulo leitor de documentos. A interface principal chamada *Image* também segue o padrão de projeto *Adapter*, permitindo a leitura de uma imagem a partir dos *bytes* ou de um arquivo em disco. O leitor de imagens pode utilizar uma implementação com o *Opencv*(módulo *Python* para processamento de imagens) ou *Pil* outro módulo *Python* para processamento de imagens.

Figura 3 — Diagrama do leitor de imagens.



Fonte: o próprio autor.

Essa abordagem permite que o sistema tenha compatibilidade com mais de um módulo de processamento de imagens.

3.1.2. Módulo para o Reconhecimento Óptico de caractere em Imagens

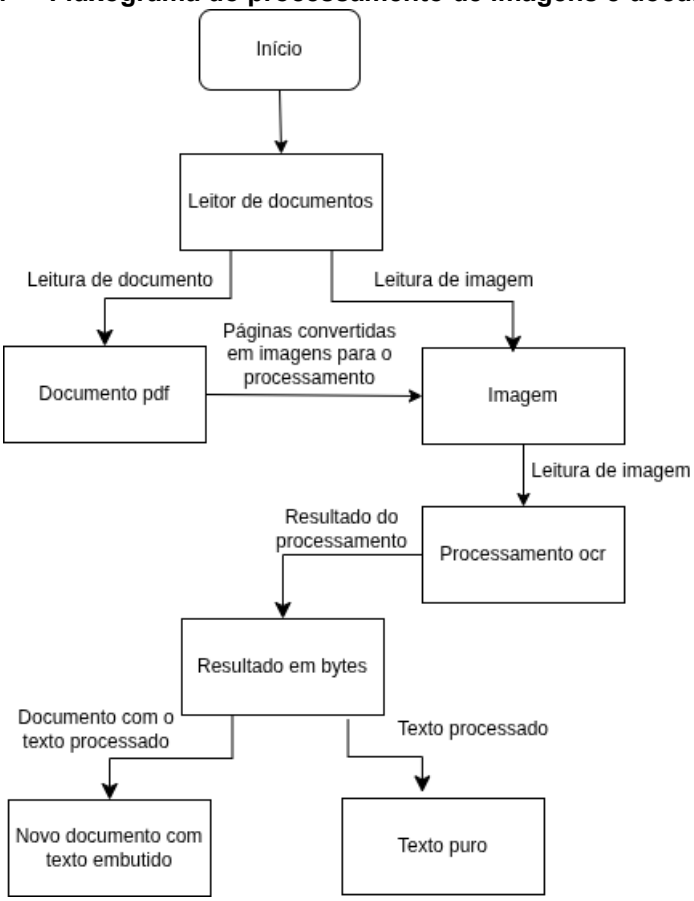
Nesta etapa, desenvolvemos o módulo leitor OCR para extrair os textos em imagens, seguindo a mesma lógica do leitor de documentos apresentada anteriormente, aproveitamos códigos existentes no *Github* para o reconhecimento de texto com o OCR, criamos uma interface com o padrão de projeto *Adapter* o processamento de texto com o *Pytesseract* ou *Easyocr*, ambos são softwares de código aberto para reconhecimento de texto em imagens.

A interface do leitor OCR faz o uso do leitor de documentos e suas funcionalidades, após o processamento podemos obter o texto, ou gerar um novo documento no qual o texto é pesquisável. Gerar um documento com o texto pesquisável é útil para evitar o reprocessamento sempre que houver a necessidade de acessar o texto extraído.

Na Figura 4 é apresentado o fluxo para leitura, processamento e análise dos documentos.

Como o *Tesseract* faz o reconhecimento óptico em imagens, não em arquivos PDF, precisamos converter cada página de PDF em imagem antes de aplicar o reconhecimento óptico. Uma maneira de mitigar essa limitação é salvar o documento gerado ao final do processo, como um novo documento com o texto embutido conforme citamos anteriormente. A interface que criamos para o processamento OCR possui essa funcionalidade implementada.

Figura 4 — Fluxograma do processamento de imagens e documentos.



Fonte: o próprio autor.

O processo apresentado na Figura 4 é aplicado individualmente em cada documento analisado e processado pelo sistema.

1.5. Desenvolvimento da interface web

Para utilização do sistema proposto desenvolvemos uma interface *web* utilizando o *Framework Flutter* na versão 3.29. Esse sistema deverá possuir as seguintes funcionalidades: (I) Adicionar PDF; (II) Adicionar Imagem; (III) Carregar

Planilha XLSX, (IV) nomear um documento importado a partir de palavras chaves presentes no conteúdo do próprio documento.

Essas funcionalidades foram implementadas em conjunto com as funções do *Tesseract*.

1.6. Eficiência do sistema, análise quantitativa da extração de texto.

Após a etapa de codificação, realizamos alguns testes para verificar a eficiência do sistema. O teste consistiu em selecionar dez arquivos PDF e 10 arquivos de imagem de fichas de EPIs distintas, tais fichas foram digitalizadas em um *Scanner* de impressora citado no tópico 1.3. Considerando que as principais palavras-chave da ficha de EPI são: data, nome, matrícula e local de entrega, classificamos a eficiência conforme o quadro 1 apresentado a seguir.

Quadro 1— Descrição para qualificação abordada no quadro 2

Ótimo	Quando identificar três ou mais palavras-chave
Bom	Quando identificar pelo menos duas palavras-chave
Insuficiente	Quando identificar menos de duas palavras-chave

Fonte: o próprio autor

.

A eficiência desse teste focou na quantidade de palavras-chave reconhecidas nas fichas de EPIs, sendo que são essenciais para organização desses documentos.

4 RESULTADOS E DISCUSSÃO

1.7. Desenvolvimento do sistema Web

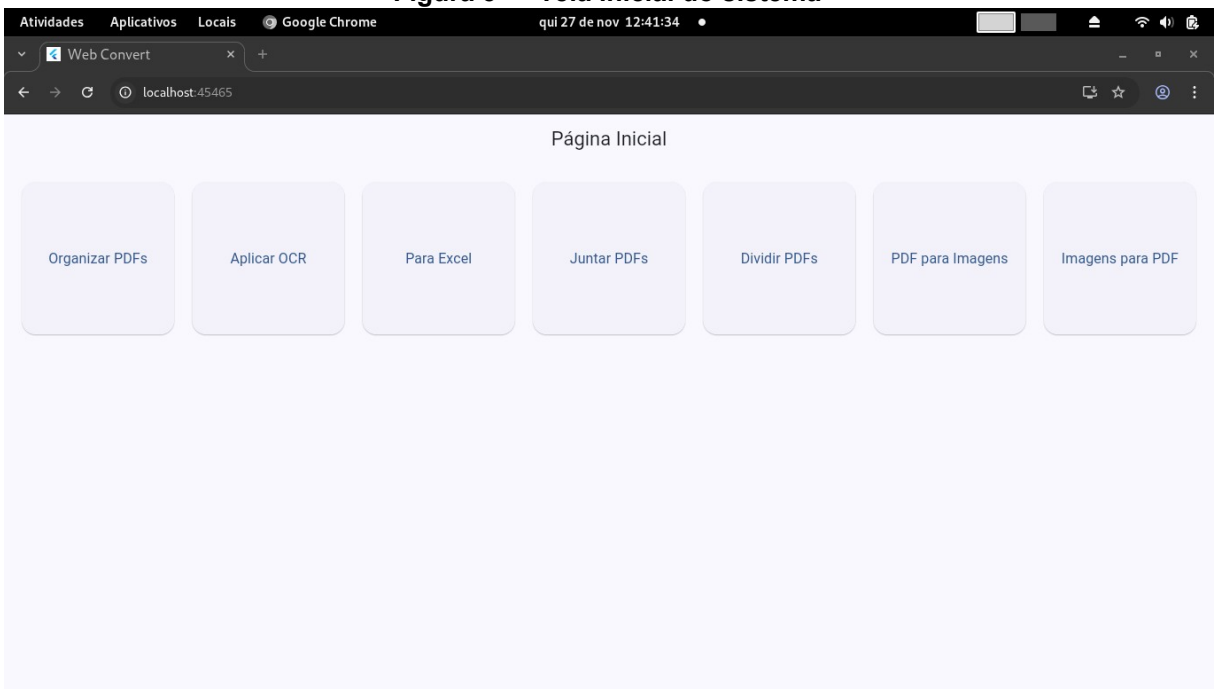
Para o desenvolvimento deste sistema *web* utilizou-se o *Flutter* que é um *Framework* de código aberto desenvolvido pelo Google, utilizado para a criação de aplicações multiplataforma, incluindo sistemas *web*, aplicativos móveis e desktop a partir de um único código-fonte.

O *Flutter* permite que aplicações sejam executadas diretamente em navegadores sem a necessidade de plug-ins adicionais. Além disso, o *Framework* facilita a manutenção e evolução da aplicação.

A Figura 5 apresenta a tela inicial do sistema para importação de documentos e extração de texto. Projetada para permitir aos usuários a importação de

documentos nos formatos PDF, imagem e planilhas XLSX. Além disso, para permitir que os usuários selecionem arquivos para processamento por meio de um navegador, a interface foi projetada para ser acessível em diversos dispositivos, como computadores e celulares.

Figura 5 — Tela inicial do sistema



Fonte: o próprio autor.

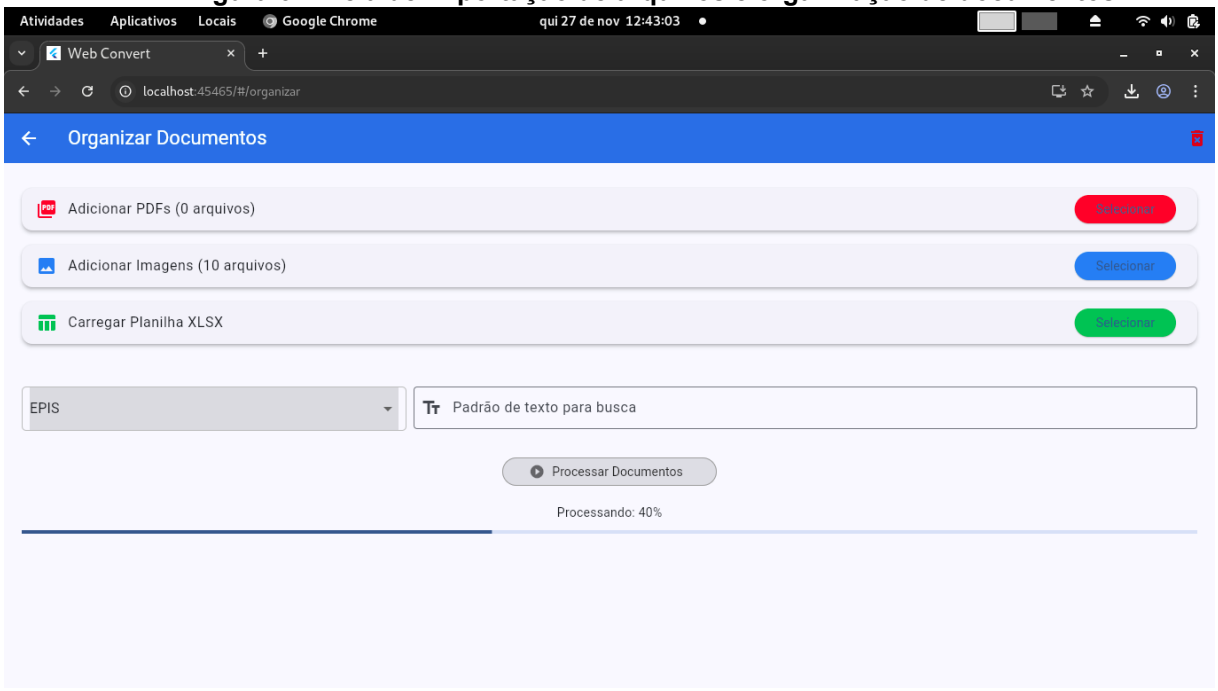
Os módulos para importação e reconhecimento de caracteres foram desenvolvidos em linguagem de programação *Python*, sendo um para ler e gravar dados nos arquivos analisados pelo sistema, e outro para o reconhecimento de texto em tais arquivos.

Após a conclusão dos algoritmos de importação (Figura 8), desenvolvemos uma interface *web* com a funcionalidade de organizar arquivos, para que os usuários do sistema possam selecionar e organizar os arquivos por meio de um navegador de *internet* (Figura 7) em computadores ou celulares, por exemplo, a Figura 6 apresenta a tela dessa funcionalidade.

A interface para organização de documentos (Figura 6), permite aos usuários selecionarem o tipo de documento importado, como fichas de EPIs e documentos genéricos, sendo que documentos genéricos são todos os outros tipos de documentos que não são fichas de EPIs. Para o processamento de um documento

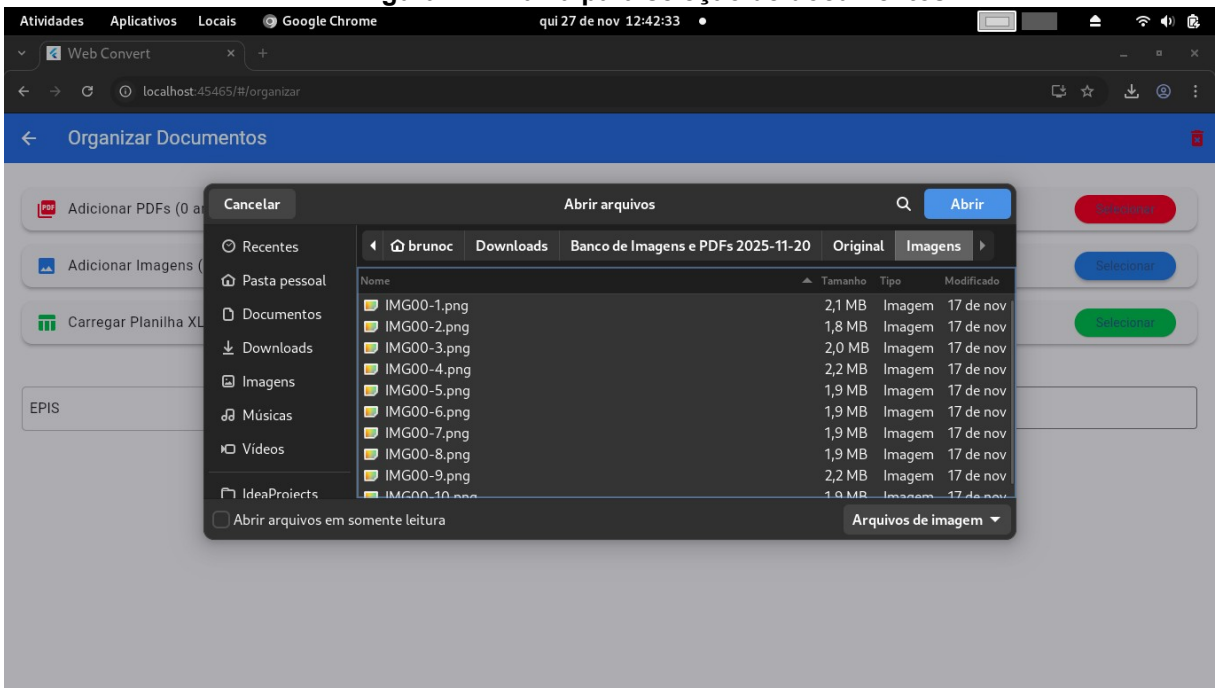
genérico o usuário precisa informar uma palavra-chave (no campo “padrão de texto para busca”) que será utilizada para o processamento e organização.

Figura 6 —Tela de importação de arquivos e organização de documentos.



Fonte: o próprio autor.

Figura 7 — Caixa para seleção de documentos.



Fonte: o próprio autor.

Terminada a codificação dos módulos e da interface, realizamos alguns testes tanto no módulo individualmente, quanto no sistema por meio da interface, a fim de verificar a eficiência do sistema. Os resultados obtidos são apresentados adiante.

1.8. Teste de Extração de Texto

O código mostrado na Figura 8 apresenta a importação e utilização dos módulos para extração de texto. Sendo que o leitor de imagem recebe um arquivo de imagem como parâmetro, o objeto gerado é passado para o método *image_to_string* da classe “LeitorOcr”, o resultado obtido (Figura 9) é o texto da imagem na Figura 1.

É importante destacar que a qualidade dos documentos digitalizados impacta nos resultados, de modo que se os papéis digitalizados no *Scanner* estiverem com ondulações e/ou amassados, terá impacto negativo nos resultados. Tal limitação pode ser mitigada utilizando-se técnicas de processamento de imagem, mas essa funcionalidade não foi implementada no sistema.

Para este experimento utilizou-se um computador pessoal com o sistema *Linux Debian* versão 12, com processador *Intel Core* i3-8130U com 8GB de memória RAM.

Figura 8 — Código para o reconhecimento óptico de uma imagem.

```
from soup_files import File
from ocr_stream import RecognizeImage as LeitorOcr
from convert_stream import ImageObject as LeitorImagem

leitor_imagem = LeitorImagem(File('ficha de epi.png'))
ocr = LeitorOcr()
texto = ocr.image_to_string(leitor_imagem)
print(texto)
```

Fonte: o próprio autor.

Após a implementação da interface *web* (Figura 6) e dos algoritmos de importação e extração de texto, iniciamos os testes do reconhecimento de caracteres e organização de documentos. Durante essa fase, avaliamos a

capacidade do sistema em processar diferentes tipos de imagens e identificar corretamente as informações presentes nos documentos.

Desta forma realizamos o teste de extração de texto com 10 arquivos em formato PDF e 11 arquivos em formato imagem (Cupom fiscal e ficha de EPI). Para a ficha de EPI foi utilizado o conjunto de palavras-chave data, matrícula e local, enquanto para o cupom fiscal foi utilizado as palavras data e protocolo, sendo que as palavras-chave encontradas são utilizadas para nomear os próprios documentos.

Notamos na Figura 9 que o resultado apresenta algumas inconsistências, sendo algumas delas caracteres reconhecidos erroneamente, sinais de acentuação ausentes em algumas palavras, algumas frases não estão na posição original entre outras. Em nossos testes identificamos que o reconhecimento óptico pode confundir alguns caracteres, trocando a letra “l” pela letra “i”, a letra “B” pelo número “8” e vice-versa.

Entretanto, apesar das limitações apresentadas, o sistema pode realizar os objetivos propostos, visto que no conjunto de informações presentes na Figura 9, temos elementos suficientes para nomear a ficha de EPIs.

Considerando que esse processo pode ser repetido sequencialmente com dezenas, centenas ou até milhares de imagens por vez, gerando um volume elevado de dados para análise, optamos por usar o módulo *Pandas* para análise e filtro dos resultados processados.

Figura 9 — Captura de tela do texto extraído com o reconhecimento óptico da figura 1.

FICHA DE CONTROLE DE EPI

MATRICULA NOME DO COLABORADOR FUNCAO DEPARTAMENTO LOCAL ENTREGA

123456 PEDRO DA SILVA. AUX COMERCIAL, DESC Porto Veiho

Declaro que recebi os equipamentos de protecao individuais (EPI. relacionados nesta ficha,pelos quais assume total|
responsabilidade conforme NR-6 Portaria MTb n.7 877,de 24 de outubro de 2018:comprometendo-me a usé-lo apenas|
para a finalidade a que se destina,responsabilizo-me por aguarda,conservacdo e higienizac&o,comunicacao ad
empregador qualquer alteracao que o tome impréprio para o uso, cumprir as determinacées do

empregador sobre 0 uso adequado, estou ciente que, a recusa injustificada do uso dos mesmos, estarei sujeito as penalidades|
revistas em lei.

Porto Veho/RO-quinta-fera, 21 de setenbro de 2023.

Responsavel Imediato ‘assinatura do Colaborador Data Registro Tnicial DEVOLUCAO
ITEM | EQUIPAMENTO QUANT RECEBMENTO
CA | QU ‘UBRICA, RUBRICA RESP.
FORNECIDO DATA | RUBRICA RESP TREGA DATA IruBrica “ARES COLAB. [CcOLaB.

1 [BOTINA TAM 42 21/09

Fonte: o próprio autor.

Os resultados obtidos ao processar as fichas de EPIs com a interface do sistema (Figura 6) podem ser observados no quadro 2.

Observamos no quadro 2 que, duas fichas de EPIs apresentaram resultado “insuficiente”, oito apresentaram resultado “bom” e dez “ótimo”, sendo um total de 90% de eficiência para a amostragem de dados analisada. Destacamos que o objetivo do sistema não é o conteúdo do texto em si, mas padrões presentes em documentos que tornem possível renomear tais arquivos com base em padrões de texto.

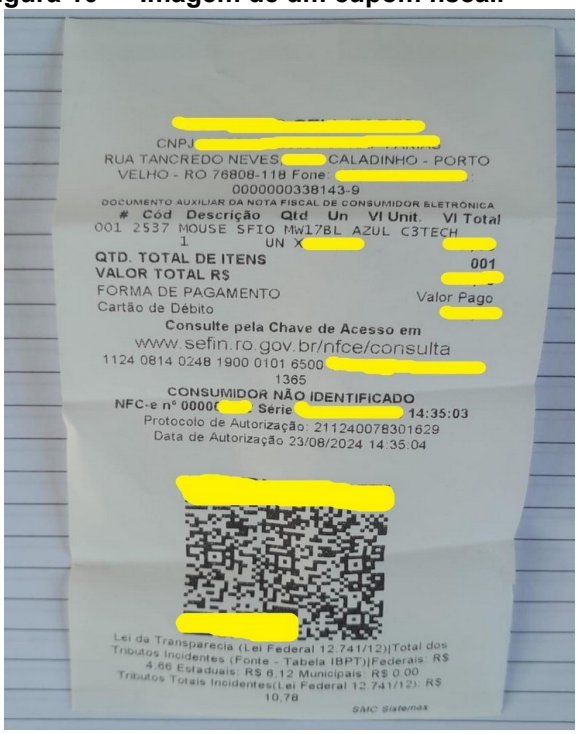
Quadro 2: Resultado gerado pelo sistema com a interface *web.*

Elementos Reconhecidos	Qualificação	Porcentagem
Matrícula e nome	Bom	40,00%
Data	Insuficiente	10,00%
Matrícula, nome e data	Ótimo	50,00%

Fonte: o próprio autor.

O sistema também foi testado com outro tipo de documento, a fim de verificar a sua eficiência com documentos diversos. Na Figura 10 vemos a fotografia de um cupom fiscal o qual submetemos ao mesmo processamento.

Figura 10 — Imagem de um cupom fiscal.

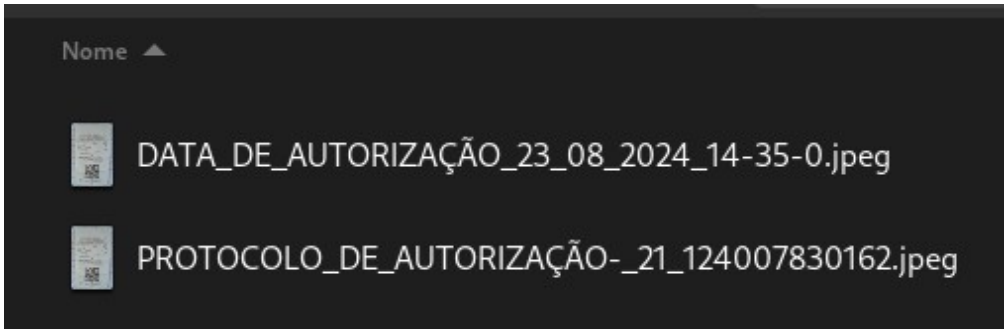


Fonte: O próprio autor.

Após a importação do Cupom Fiscal (Figura 10), o sistema executa automaticamente a rotina de extração de palavras chaves que são digitadas pelo usuário (no campo “padrão de texto para busca” Figura 6), tais como: data e protocolo. O sistema identifica e extrai as informações contidas no documento importado, e em seguida, o sistema realiza uma etapa de pós-processamento: com base nos dados extraídos, ele gera um novo arquivo que contém o texto (ou os dados) extraídos. O aspecto importante dessa geração é que o nome do novo arquivo é construído utilizando o próprio texto extraído, neste caso, "Data_de_autorização_23_08_2024_14_35_0.jpeg.pdf" (Figura 11).

Desta forma, esta padronização no nome do arquivo facilita significativamente o processo de indexação e busca de documentos. A utilização dos metadados extraídos diretamente no nome permite que a localização e a recuperação de documentos sejam realizadas de forma rápida melhorando a eficiência operacional do gerenciamento documental.

Figura 11 — Dois arquivos de um cupom fiscal.



Fonte: O próprio autor.

5 CONSIDERAÇÕES FINAIS OU CONCLUSÃO

Considerando as necessidades propostas no início do projeto, os resultados obtidos foram satisfatórios, ainda que em alguns casos o algoritmo não reconheça corretamente alguns caracteres, e forme algumas palavras incorretas. No entanto, se palavras-chave como data, nome, matrícula, local entre outras forem reconhecidas corretamente o objetivo geral do sistema será atingido.

Além disso, foi observado que para aumentar a taxa de reconhecimento do texto é importante que o papel a ser digitalizado ou fotografado seja o mais plano possível, pois ainda que a foto ou digitalização tenha melhor qualidade, inclinações e ondulações nos papéis comprometem a taxa de detecção dos textos. No entanto, para a funcionalidade de renomear os arquivos está limitação pode ser mitigada com algoritmos de processamento de imagem, para elevar a taxa de acerto ao aplicar o OCR.

Em resumo, foi verificado nos testes realizados com banco de imagens (tópico 1.3) que 10% dos documentos processados pelo sistema apresentaram resultados considerados insuficiente (Quadro 1), com os demais resultados sendo 40% considerado bom e 50% considerado ótimo.

Para trabalhos futuros recomendamos o uso da versão mais recente do *Tesseract* por possuir maior taxa de acerto durante o reconhecimento de texto e formação das palavras.

6 REFERÊNCIAS

CHERIET, M. Character recognition systems. New Jersey, 2007. Disponível em https://pt.scribd.com/document/901437943/Character-Recognition-Systems-A-Guide-for-Students-and-Practioners-1st-Edition-Mohamed-Cheriet-download. Acesso em: 15 ago 2024.

CUNHA, R.C. Estudo comparativo sobre ferramentas de reconhecimento óptico de caracteres. Formiga - MG, 21 nov 2018. Disponível em https://www.formiga.ifmg.edu.br/documents/2018/Biblioteca/TCCs_e_Artigos/Renata_Carolina_Cunha.pdf. Acesso em: 15 ago 2024.

EIKVIL, L. Opticalcharacterrecognition Oslo. dez 1993. Disponível em https://home.nr.no/~eikvil/OCR.pdf. Acesso em: 15 ago. 2025.

FARIAS W.N.D. Aperfeiçoando o reconhecimento óptico de caracteres em imagens de documentos pessoais. Campina Grande - PB. 17 nov. 2023. Disponível em http://dspace.sti.ufcg.edu.br:8080/xmlui/bitstream/handle/riufcg/34848/WALISSON%20NASCIMENTO%20DE%20FARIAS-ARTIGO-CEEI-CI%C3%8ANCIA%20DA%20COMPUTA%C3%87%C3%83O%20%282023%29.pdf?sequence=1&isAllowed=y. Acesso em: 15 ago. 2024.

GAMMA, E. Padrões de projeto. Soluções reutilizáveis de software orientado a objetos. Massachusetts, 1995. Disponível em: https://www.kufunda.net/publicdocs/Padr%C3%B5es%20de%20Projetos%20Solu%C3%A7%C3%B5es%20Reutiliz%C3%A1veis%20(Gamma,%20Erich%20%20johnson,%20Ralph%20%20Helm%20etc.).pdf. Acesso em: 21 set. 2025.

IBM. International Business Machines Corporation. O que é reconhecimento ótico de caracteres (OCR)?. Disponível em https://www.ibm.com/br-pt/think/topics/optical-character-recognition#:~:text=A%20tecnologia%20OCR%20tornou-se,precis%C3%A3o%20de%20OCR%20quase%20perfeita. Acesso em: 15 ago. 2024.

LINS, L.F.M.V. Reconhecimento ótico de caracteres (ocr) e análise de sistemas ocr baseados em código aberto. São Paulo, 2012. Disponível em https://www.fatecsp.br/dti/tcc/tcc00068.pdf.. Acesso em: 15 ago. 2024.

RAAB, A.L.A.; FILHO, O.P. Estudo comparativo entre sistemáticas de digitalização de documentos formatos HTML e PDF. SciELO, Brasília,Vol. 27. 300-310, 1998. Disponível em Estudo comparativo entre sistemáticas de digitalização de documentos: formatos HTML e PDF. **Semantic Scholar**. Acesso em: 15 ago. 2024.