**Logotipo, nome da empresa

Descrição gerada automaticamente**

**Relatório Técnico**

Projeto de Desenvolvimento: Aplicativo PagJur

**Autor:**

Raquel Gonçalves dos Santos

**Data Entrega para Teste:**

04/01/2024

**Período de Teste:**

04/01/2024 – 12/01/2024

**Data Entrega para PRD:**

12/01/2024

**Resumo:**

Aplicativo PagJur tem a função de unificar capas de requisições de pagamentos com seus documentos correspondentes

# **Sumário**

|  |  |
| --- | --- |
| **1. Introdução** | **Página 3** |
| **1.0 Como Funciona?** | **Página 3** |
| **2. SCRIPTS BACK-END** | **Página 3** |
| **2.0 Script 1 escrito** | **Páginas 3 e 4** |
| **2.1 Script 2 escrito** | **Página 4 e 5** |
| **3. SCRIPTS INTERFACE** | **Página 6 e 7** |
| **4. IMPLEMENTAÇÃO** | **Página 7** |
| **4.0 Instalando aplicativo na máquina do usuário final** | **Página 7 ao 10** |
| **4.1 Manuseio** | **Página 10** |
| **4.2 Manutenção** | **Página 10** |

**1. Introdução**

**1.0 Como Funciona?**

**Tarefa: Automatizar a junção de arquivos que contém cód. Interno colocando como capa em primeira posição na página 1 de cada requisição e pagamento.**

**2. Script**

**2.0 Script escrito:**

Primeiro o código realiza a leitura das pastas armazena fazendo alocação de memória com array, depois de ter armazenado a sequência dos seis dígitos que se refere ao cód. Interno ele ler cada arquivo que começa com nomenclatura “Pagamentos para GEAFIC” que é a capa de cada requisição de pagamento, comparando a sequência de seis dígitos ele move a capa para as respectivas pastas e unifica tornando os arquivos em apenas um recebendo o nome do arquivo com o cód. Interno.

import tkinter as tk

from tkinter import filedialog

from tkinter import messagebox

from PyPDF2 import PdfReader, PdfWriter

import fitz  # PyMuPDF

import re

import os

import shutil

class PDFProcessorApp:

    def \_\_init\_\_(self, root):

        self.root = root

        self.root.title("PagJur")

        self.label = tk.Label(root, text="Selecione a pasta com os arquivos PDF:")

        self.label.pack(pady=10)

        self.folder\_path = tk.StringVar()

        self.entry = tk.Entry(root, textvariable=self.folder\_path, width=40)

        self.entry.pack(pady=10)

        self.browse\_button = tk.Button(root, text="Procurar", command=self.browse\_folder)

        self.browse\_button.pack(pady=10)

        self.process\_button = tk.Button(root, text="Unificar PDFs", command=self.process\_pdfs)

        self.process\_button.pack(pady=10)

    def browse\_folder(self):

        folder\_selected = filedialog.askdirectory()

        self.folder\_path.set(folder\_selected)

    def extract\_and\_move\_cover(self, file\_path):

        try:

            pdf\_document = fitz.open(file\_path)

            first\_page = pdf\_document[0]

            text = first\_page.get\_text()

            pdf\_document.close()

            if 'Pagamentos para GEAFI - Requisições' in text:

                # Encontra todas as sequências de 6 dígitos no texto

                all\_matches = re.findall(r'\b\d{6}\b', text)

                for folder\_number in all\_matches:

                    if folder\_number != '109106' and folder\_number != '103812':

                        source\_folder = os.path.dirname(file\_path)

                        destination\_folder = os.path.join(source\_folder, folder\_number)

                        if not os.path.exists(destination\_folder):

                            os.makedirs(destination\_folder)

                            print(f"Pasta criada: {destination\_folder}")

                        destination\_path = os.path.join(destination\_folder, os.path.basename(file\_path))

                        shutil.move(file\_path, destination\_path)

                        print(f"Arquivo de capa movido para {destination\_path}")

                        break  # Para após a primeira correspondência

        except Exception as e:

            messagebox.showerror("Erro na Extração e Movimentação", f"Erro ao processar o arquivo {file\_path}: {str(e)}")

    def move\_covers(self):

        folder\_path = self.folder\_path.get()

        # Executa a extração da capa e movimento para a pasta correspondente

        pdf\_files = [f for f in os.listdir(folder\_path) if f.lower().endswith('.pdf')]

        for pdf\_file in pdf\_files:

            file\_path = os.path.join(folder\_path, pdf\_file)

            if os.path.isfile(file\_path):

                self.extract\_and\_move\_cover(file\_path)

    def merge\_pdfs\_recursive(self, base\_folder):

        try:

            folder\_files = {}

            for folder\_name, subfolders, files in os.walk(base\_folder):

                for file in files:

                    file\_path = os.path.join(folder\_name, file)

                    if file.lower().endswith('.pdf'):

                        if folder\_name not in folder\_files:

                            folder\_files[folder\_name] = [file\_path]

                        else:

                            folder\_files[folder\_name].append(file\_path)

            for folder\_name, file\_paths in folder\_files.items():

                if folder\_name != '109106':

                    merged\_pdf = PdfWriter()

                    capa\_file = next((f for f in file\_paths if 'Pagamentos para GEAFI - Requisições' in f), None)

                    if capa\_file:

                        with open(capa\_file, 'rb') as capa:

                            pdf\_reader = PdfReader(capa)

                            for page\_num in range(len(pdf\_reader.pages)):

                                page = pdf\_reader.pages[page\_num]

                                merged\_pdf.add\_page(page)

                    for file\_path in file\_paths:

                        if file\_path != capa\_file:

                            with open(file\_path, 'rb') as file:

                                pdf\_reader = PdfReader(file)

                                for page\_num in range(len(pdf\_reader.pages)):

                                    page = pdf\_reader.pages[page\_num]

                                    merged\_pdf.add\_page(page)

                    output\_path = os.path.join(base\_folder, f'{folder\_name}.pdf')

                    with open(output\_path, 'wb') as output\_file:

                        merged\_pdf.write(output\_file)

                    print(f"PDF unificado em {output\_path}")

        except Exception as e:

            messagebox.showerror("Erro na Unificação de PDFs", f"Erro ao unificar PDFs na pasta {base\_folder}: {str(e)}")

    def process\_pdfs(self):

        self.move\_covers()

        folder\_path = self.folder\_path.get()

        # Ordena as pastas antes de unificar os PDFs

        subfolders = sorted([f.path for f in os.scandir(folder\_path) if f.is\_dir()])

        # Executa a unificação dos PDFs

        for subfolder in subfolders:

            self.merge\_pdfs\_recursive(subfolder)

        # Exibe a mensagem de conclusão

        messagebox.showinfo("Concluído", "Unificação foi realizada, obrigada por hoje!")

if \_\_name\_\_ == "\_\_main\_\_":

    root = tk.Tk()

    app = PDFProcessorApp(root)

    root.mainloop()

**4. IMPLEMENTAÇÃO**

**4.0 Instalando aplicativo na máquina do usuário final**

Basta enviar arquivo compactado e na máquina do usuário clicar com botão direito sob o arquivo e clicar em extrair tudo, após extraído ele abrirá a interface com funcionalidades intuitivas de fácil manuseio.

**4.1 Manuseio:**

* Ao clicar aparecerá esta tela:

**Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente**

* Clicando em Procurar, o sistema acessará o explorador de arquivo, a partir disso é possível escolher a pasta onde encontra-se as requisições de pagamentos

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

* Selecione a pasta desejada

**Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente**

* Clique em ‘Selecionar pasta’ para que pasta seja o alvo da automação

**Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente**

* Clique em ‘Unificar PDFs’ para que o Script 1 e 2 seja executado

**Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente**

* Quando o processo for concluído o sistema emitirá uma mensagem de sucesso ‘Unificação foi realizada, obrigada por hoje’

**Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente**

**4.2 Manutenção:**

Todos os códigos foram escritos de forma simples e cada bloco com comentários objetivos no intuito de que seja fácil a manutenção caso venha alguma mudança, mas para que seja feito qualquer mudança, é necessário realizar versionamento do código salvando o original em backup e as versões em pastas ou no GITHUB que faz o controle de versão.