ATIVOS S.A.
Securitizadora de Créditos Financeiros

Relatório Técnico

Projeto de Desenvolvimento: Aplicativo PagJur

Autor:

Raquel Gonçalves dos Santos

Data Entrega:

10/01/2024

Resumo:

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

Sumário

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:
            if not file_path.lower().endswith('.pdf'):
                return  # Ignorar arquivos que não sejam PDFs

            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()
```

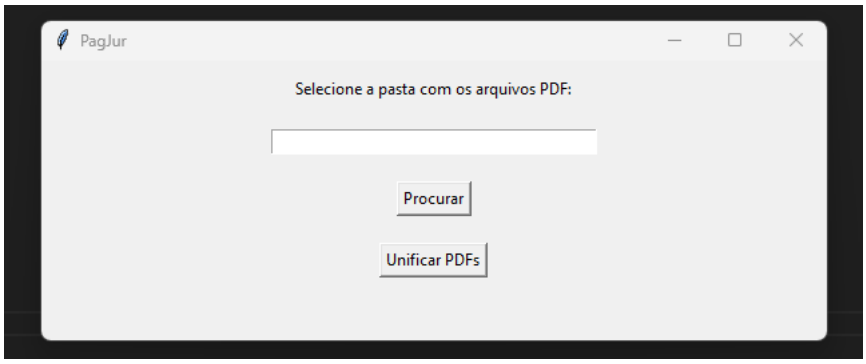
4. IMPLEMENTAÇÃO

4.0 Instalando aplicativo na máquina do usuário final

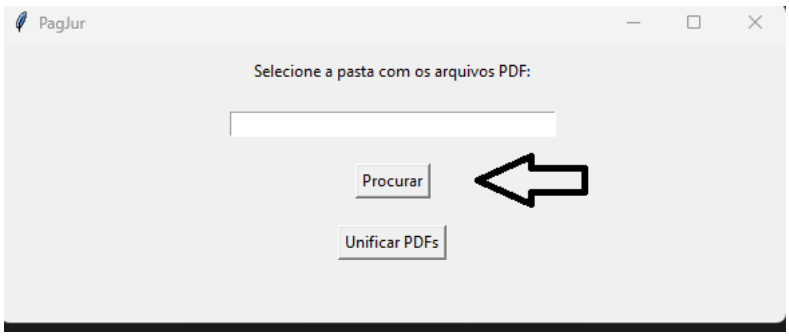
Basta enviar arquivo compactado e na máquina do usuário clicar com botão direito sob o arquivo e clicar em extrair tudo, após extraído ele abrirá a interface com funcionalidades intuitivas de fácil manuseio.

4.1 Manuseio:

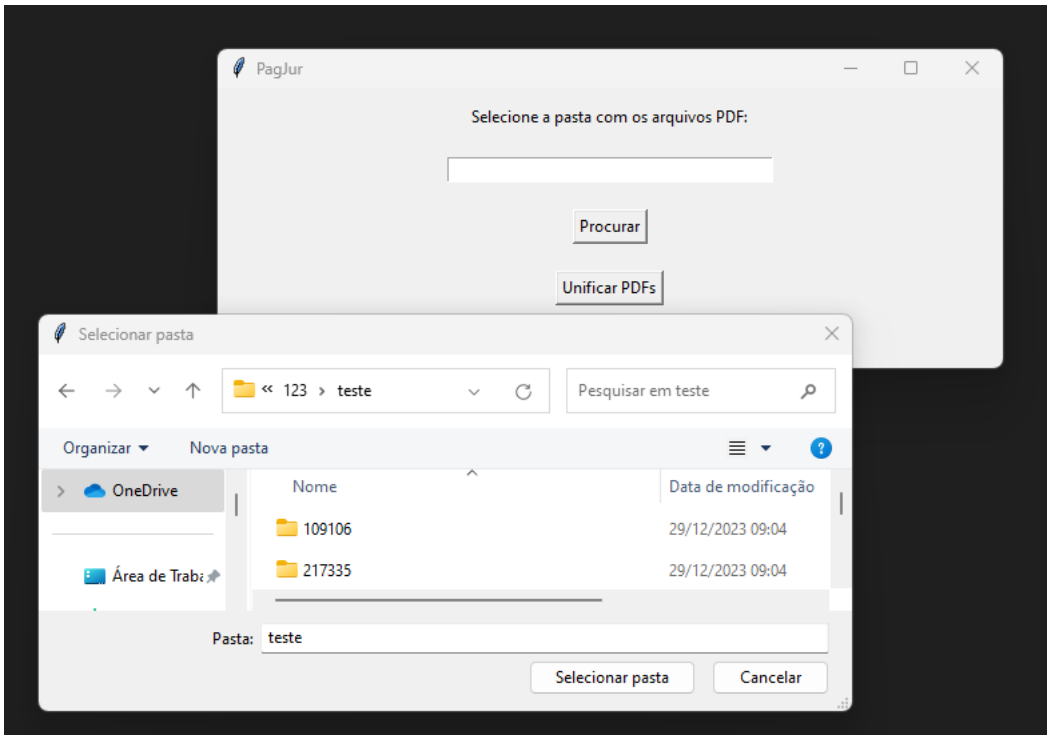
- Ao clicar aparecerá esta tela:



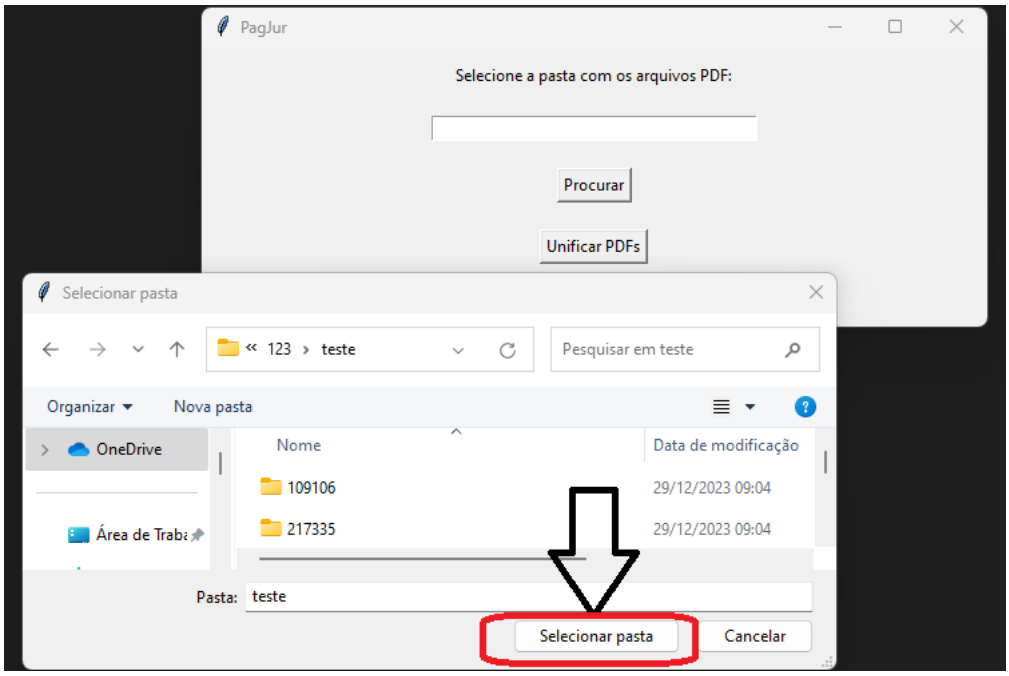
- Clicando em Procurar, o sistema acessará o explorador de arquivo, a partir disso é possível escolher a pasta onde encontra-se as requisições de pagamentos



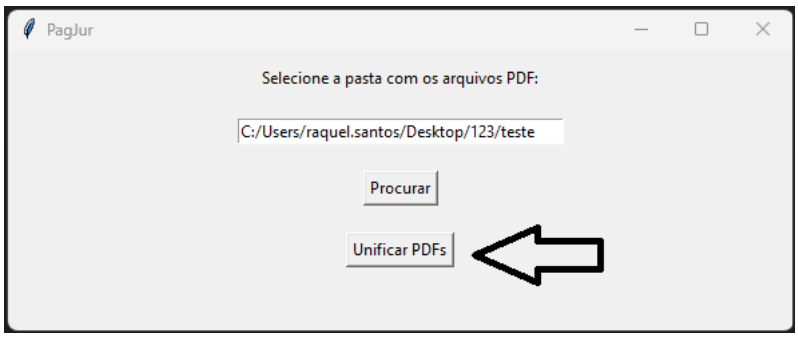
- Selecione a pasta desejada



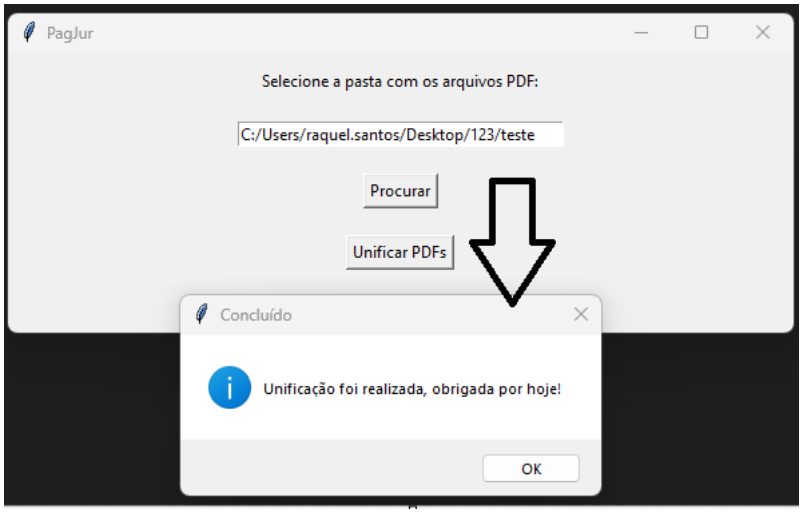
- Clique em ‘Selecionar pasta’ para que pasta seja o alvo da automação



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



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



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.

4.2 Versionamento GITHUB:

Caso necessite versionamento:

Link: https://github.com/raqueldebug/Monta_Processo_Judicial.git