Relatório de Implementação e Análise: "Meu Estagiário" - Gestão Financeira para Pequenos Empreendedores com IA

Nome do grupo: Grupo do Raoni - “Meu estagiário”

Nome dos integrantes: Raoni Rafael de Limas

Tema escolhido: Automação de Processos Fiscais/Contábeis

1. Introdução

A gestão financeira é um pilar fundamental para a sustentabilidade e o crescimento de qualquer negócio, especialmente para microempreendedores individuais (MEIs) e pequenos empresários, que frequentemente lidam com recursos limitados e múltiplas responsabilidades. A complexidade na organização de notas fiscais, o registro de transações e a análise de dados financeiros podem consumir um tempo valioso que poderia ser dedicado ao core business.

Este relatório apresenta o desenvolvimento e a funcionalidade da ferramenta "Meu Estagiário", uma solução baseada em Inteligência Artificial (IA) projetada para auxiliar pequenos empresários na gestão de suas notas fiscais e na obtenção de insights financeiros de forma simplificada. A ferramenta automatiza a extração de dados de PDFs, a estruturação dessas informações em um formato CSV e oferece uma interface de consulta interativa, facilitando a tomada de decisões financeiras.

2. Cenário e Desafios do Pequeno Empreendedor

Pequenos empreendedores e MEIs enfrentam desafios únicos na gestão financeira, incluindo:

- **Volume de Documentos:** Acúmulo de notas fiscais (DANFEs) em formato PDF, que exigem tempo para serem processadas manualmente.
- **Organização e Acesso a Dados:** Dificuldade em manter os dados financeiros organizados e acessíveis para consultas rápidas ou análises.
- **Falta de Ferramentas Especializadas:** Muitas vezes, a falta de acesso a softwares de gestão financeira robustos e caros.
- **Tomada de Decisão:** Dificuldade em extrair informações relevantes para apoiar decisões estratégicas e operacionais do negócio.
- **Tempo:** A gestão financeira manual consome um tempo precioso que poderia ser investido em atividades mais estratégicas ou produtivas.

A ferramenta "Meu Estagiário" visa mitigar esses desafios, oferecendo uma solução acessível e eficiente que automatiza processos tediosos e fornece informações financeiras de maneira intuitiva.

3. A Solução "Meu Estagiário"

"Meu Estagiário" é uma aplicação que atua como um assistente virtual financeiro, utilizando IA para processar notas fiscais e disponibilizar essas informações de forma estruturada e interativa. A arquitetura da ferramenta compreende as seguintes etapas principais:

1. **Extração de Texto de PDFs:** Leitura e extração do conteúdo textual de notas fiscais em formato PDF.
2. **Análise e Geração de CSV por LLM:** Uma Large Language Model (LLM) analisa o texto extraído e formata as informações financeiras em uma linha de CSV padronizada.
3. **Geração e Armazenamento do CSV Local:** Consolidação de todas as linhas CSV em um arquivo `dados_nf.csv`, que serve como banco de dados estruturado.
4. **Interface Gráfica de Consulta (Chat):** Uma aplicação de desktop com interface de chat permite ao usuário interagir com os dados do CSV, fazendo perguntas em linguagem natural e recebendo respostas da IA.

3.1. Detalhamento Técnico e Explicação do Código

O código-fonte fornecido implementa a solução "Meu Estagiário" utilizando bibliotecas Python. A seguir, detalhamos cada seção:

```
import os
import csv
import pandas as pd
import tkinter as tk
from tkinter import scrolledtext
from dotenv import load_dotenv
from PyPDF2 import PdfReader
from langchain_openai import ChatOpenAI
from langchain.prompts import PromptTemplate

# ======================================================
# 1. CONFIGURAÇÃO INICIAL
# ======================================================
load_dotenv()
llm = ChatOpenAI(
    model="gpt-4o-mini",
    temperature=0,
    api_key=""
)
```

1. Configuração Inicial:

Esta seção importa as bibliotecas necessárias, como `os` para operações de sistema de arquivos, `csv` para manipulação de CSV, `pandas` para tratamento de dados, `tkinter` para a interface gráfica, `dotenv` para carregar variáveis de ambiente, `PyPDF2` para leitura de PDFs e `langchain_openai` e `PromptTemplate` para interação com modelos de linguagem. `load_dotenv()` carrega variáveis de ambiente (como a chave da API da OpenAI) de um arquivo `.env`, garantindo a segurança das credenciais. `ChatOpenAI` inicializa a interface com o modelo `gpt-4o-mini` da OpenAI, definindo a `temperature` (criatividade) como 0 para respostas mais objetivas e consistentes. A chave da API é

fornecida diretamente para este exemplo, mas em produção seria carregada de forma segura.

```
# ======================================================
# 2. EXTRAÇÃO DE TEXTO DO PDF
# ======================================================
def extrair_texto_pdf(caminho_pdf: str) -> str:
    reader = PdfReader(caminho_pdf)
    texto = ""
    for pagina in reader.pages:
        texto += pagina.extract_text() + "\n"
    return texto
```

2. Extração de Texto do PDF:

A função `extrair_texto_pdf` recebe o caminho de um arquivo PDF. Utiliza a biblioteca PyPDF2 para ler o documento. Percorre todas as páginas do PDF e concatena o texto extraído de cada uma delas, retornando uma única string contendo todo o conteúdo textual da nota fiscal.

```
# ======================================================
# 3. GERAÇÃO DE LINHAS CSV PELO LLM
# ======================================================
def analisar_nf_csv(texto_pdf: str, nome_arquivo: str) -> str:
    prompt_template = PromptTemplate(
        input_variables=["texto_nf", "nome_arquivo"],
        template=(
            "Você é um especialista em Notas Fiscais eletrônicas (DANFE). "
            "Analise o texto da nota abaixo e extraia as seguintes informações:\n\n"
            "- Emitente (nome e CNPJ)\n"
            "- Destinatário (nome e CNPJ)\n"
            "- Data de emissão\n"
            "- Valor total da nota\n"
            "- Chave de acesso (se houver)\n"
            "- Lista de produtos (descrição, quantidade, valor unitário, valor
total)\n\n"
            "Retorne APENAS uma linha de CSV, no formato:\n"

"arquivo_origem,emitente,destinatario,data_emissao,valor_total,chave_acesso,produtos\
n\n"
            "Use aspas duplas para campos que contêm vírgulas ou listas. "
            "Não escreva nenhuma explicação ou comentário adicional.\n\n"
            "Nome do arquivo: {nome_arquivo}\n\n"
            "Texto da NF:\n{texto_nf}\n"
        )
    )
    prompt = prompt_template.format(texto_nf=texto_pdf, nome_arquivo=nome_arquivo)
    resposta = llm.invoke(prompt)
    return resposta.content.strip()
```

3. Geração de Linhas CSV pelo LLM:

A função `analisar_nf_csv` é crucial para o processamento inteligente. Ela recebe o texto de um PDF e o nome do arquivo. Um `PromptTemplate` é configurado para instruir a LLM (o modelo GPT-4o-mini) a atuar como um especialista em DANFEs. O prompt solicita a extração de informações específicas (Emitente, Destinatário, Data de Emissão, Valor Total, Chave de Acesso e Lista de Produtos) e exige que a resposta seja formatada estritamente como uma única linha CSV, com aspas duplas para campos complexos e sem texto adicional. A LLM `invoke` é chamada com o prompt formatado, e o conteúdo da resposta é retornado.

```
# ======================================================
# 4. GERAÇÃO DO CSV LOCAL
# ======================================================
def gerar_csv_notas(pasta_notas="notas", caminho_csv="dados_nf.csv"):
    """Extrai dados de todas as NFs e grava em CSV robusto (vírgulas internas não
quebram)."""
    registros = []

    for arquivo in os.listdir(pasta_notas):
        if arquivo.lower().endswith(".pdf"):
            caminho_pdf = os.path.join(pasta_notas, arquivo)
            print(f"\n🔍 Lendo {arquivo}...")
            texto = extrair_texto_pdf(caminho_pdf)
            print("🤖 Analisando e formatando CSV...")
            linha_csv = analisar_nf_csv(texto, arquivo)

            # A LLM retorna uma linha CSV com vírgulas — vamos parsear com csv.reader
            # para respeitar aspas internas e recuperar os 7 campos esperados.
            try:
                # csv.reader espera um iterable de linhas
                parsed = next(csv.reader([linha_csv], quotechar='"',
skipinitialspace=True))
            except Exception:
                # fallback: split simples (menos seguro)
                parsed = [p.strip().strip('"') for p in linha_csv.split(",")]

            # Se for menor/maior que esperado, ainda assim armazenamos (para
inspeção)
            registros.append(parsed)

    # Cabeçalho fixo
    header = ["arquivo_origem", "emitente", "destinatario", "data_emissao",
"valor_total", "chave_acesso", "produtos"]

    # Gravação: delimitador ',' (padrão), quoting=QUOTE_ALL e escapechar='\\'
    with open(caminho_csv, "w", encoding="utf-8", newline="") as f:
```

```
        writer = csv.writer(f, delimiter=",", quotechar='"', quoting=csv.QUOTE_ALL,
escapechar="\\")
        writer.writerow(header)
        for linha in registros:
            # Garantir que cada linha tenha exatamente N colunas:
            if len(linha) < len(header):
                # completa com strings vazias
                linha = linha + [""] * (len(header) - len(linha))
            elif len(linha) > len(header):
                # junta colunas extras no último campo (produtos) — evita quebrar
estrutura
                linha = linha[: len(header) - 1] + [",".join(linha[len(header) - 1
:])]
            writer.writerow(linha)

    print(f"\n✅ CSV gerado com sucesso: {caminho_csv}")
    return caminho_csv
```

4. Geração do CSV Local:

A função `gerar_csv_notas` orquestra o processo de extração e gravação. Ela itera sobre todos os arquivos PDF em uma pasta (`notas` por padrão), chama `extrair_texto_pdf` e `analisar_nf_csv` para cada um. É fundamental notar o uso de `csv.reader` para parsear a linha CSV retornada pela LLM. Isso é crucial para lidar corretamente com campos que contêm vírgulas internas ou que estão entre aspas duplas, evitando quebras na estrutura dos dados. Caso o parsing falhe, um `split` simples é usado como fallback. Antes de gravar no arquivo `dados_nf.csv`, a função garante que cada linha tenha o número correto de colunas, preenchendo com vazios ou concatenando campos extras, para manter a integridade do CSV. O arquivo é gravado com codificação UTF-8 e com `csv.QUOTE_ALL` para garantir que todos os campos sejam citados, tornando o CSV mais robusto.

```
# ======================================================
# 5. INTERFACE GRÁFICA (CHAT)
# ======================================================
class ChatJanela:
    def __init__(self, caminho_csv):
        # Leitura segura do CSV; on_bad_lines='skip' evita crash por linhas
corrompidas
        try:
            self.df = pd.read_csv(caminho_csv, dtype=str, on_bad_lines="skip")
        except Exception as e:
            print(f"⚠️ Erro ao ler o CSV: {e}")
            self.df = pd.DataFrame()

        self.contexto = (
            "Você é o estagiário virtual da empresa Gabriela Gruer Arquitetura e
Interiores. "
            "Seu nome é 'Gabriela Gruer Arquitetura e Interiores - Estagiário'. "
```

```
            "Você ajuda a equipe analisando informações das Notas Fiscais armazenadas
em um CSV. "
            "Responda sempre em linguagem natural, de forma objetiva e educada, sem
mostrar código Python. "
            "Se precisar realizar cálculos, explique apenas o raciocínio e o
resultado final, sem exibir o código."
        )

        # === Janela principal ===
        self.janela = tk.Tk()
        self.janela.title("💬 Gabriela Gruer Arquitetura e Interiores - Estagiário")
        self.janela.geometry("800x600")

        # === Área de texto ===
        self.area_texto = scrolledtext.ScrolledText(self.janela, wrap=tk.WORD,
font=("Arial", 12))
        self.area_texto.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)
        self.area_texto.insert(tk.END, "👋 Olá\! Sou o estagiário virtual da Gabriela
Gruer Arquitetura e Interiores.\n")
        self.area_texto.insert(tk.END, "Posso te ajudar a entender e analisar as
notas fiscais do sistema.\n\n")
        self.area_texto.configure(state="disabled")

        # === Campo de entrada ===
        self.frame_input = tk.Frame(self.janela)
        self.frame_input.pack(fill=tk.X, padx=10, pady=10)

        self.entrada = tk.Entry(self.frame_input, font=("Arial", 12))
        self.entrada.pack(side=tk.LEFT, fill=tk.X, expand=True, padx=(0, 10))
        self.entrada.bind("<Return>", self.enviar_pergunta)

        self.botao = tk.Button(self.frame_input, text="Enviar",
command=self.enviar_pergunta)
        self.botao.pack(side=tk.RIGHT)

    def enviar_pergunta(self, event=None):
        pergunta = self.entrada.get().strip()
        if not pergunta:
            return
        self.entrada.delete(0, tk.END)

        self.adicionar_texto(f"🧑 Você: {pergunta}\n", "user")

        # Em vez de enviar CSV bruto, enviamos JSON (mais claro para a LLM)
        amostra_json = "[]"
        try:
            amostra_json = self.df.head(10).to_json(orient="records",
force_ascii=False)
            # opcional: indentar somente para legibilidade no prompt (não necessário)
            import json # import local para evitar dependência global se não usar
```

```
            amostra_json_pretty = json.dumps(json.loads(amostra_json),
ensure_ascii=False, indent=2)
        except Exception:
            amostra_json_pretty = str(self.df.head(10))

        prompt = (
            f"{self.contexto}\n\nAqui está uma AMOSTRA dos dados das notas fiscais em
JSON:\n"
            f"{amostra_json_pretty}\n\n"
            f"Pergunta do usuário: {pergunta}\n\n"
            "Responda de forma natural e direta, sem mostrar código, apenas o
resultado ou explicação."
        )

        resposta = llm.invoke(prompt)
        self.adicionar_texto(f"💼 Gabriela Gruer Arquitetura e Interiores -
Estagiário: {resposta.content}\n\n", "bot")

    def adicionar_texto(self, texto, origem="bot"):
        self.area_texto.configure(state="normal")
        self.area_texto.insert(tk.END, texto)
        self.area_texto.configure(state="disabled")
        self.area_texto.yview(tk.END)

    def iniciar(self):
        self.janela.mainloop()
```

5. Interface Gráfica (Chat):
A classe `ChatJanela` constrói a interface de usuário usando `tkinter`. No construtor (`__init__`), ela lê o arquivo CSV gerado usando `pandas.read_csv`, com `on_bad_lines='skip'` para robustez. Define um `contexto` para a LLM, instruindo-a sobre seu papel como estagiário virtual e as regras de interação. A interface consiste em uma área de texto (`ScrolledText`) para exibir o histórico da conversa e um campo de entrada (`Entry`) com um botão para enviar perguntas.

A função `enviar_pergunta` é acionada quando o usuário digita uma pergunta. Ela formata um novo prompt para a LLM, incluindo o contexto predefinido, uma **amostra dos dados do DataFrame convertida para JSON** (mais fácil para a LLM interpretar do que o CSV bruto), e a pergunta do usuário. A LLM `invoke` é então chamada para gerar uma resposta, que é exibida na área de texto. A função `adicionar_texto` gerencia a inserção de novas mensagens na área de chat. `iniciar` inicia o loop principal da janela Tkinter.

```
# ======================================================
# 6. EXECUÇÃO PRINCIPAL
# ======================================================
if __name__ == "__main__":
    caminho_csv = gerar_csv_notas()
    chat = ChatJanela(caminho_csv)
```

6. Execução Principal:
Este bloco `if __name__ == "__main__":` garante que o código dentro dele só seja executado quando o script é rodado diretamente. Primeiro, chama `gerar_csv_notas()` para processar todos os PDFs e criar o arquivo CSV. Em seguida, instancia a `ChatJanela` com o caminho do CSV gerado e inicia a interface gráfica chamando `chat.iniciar()`.

4. Benefícios e Aplicações

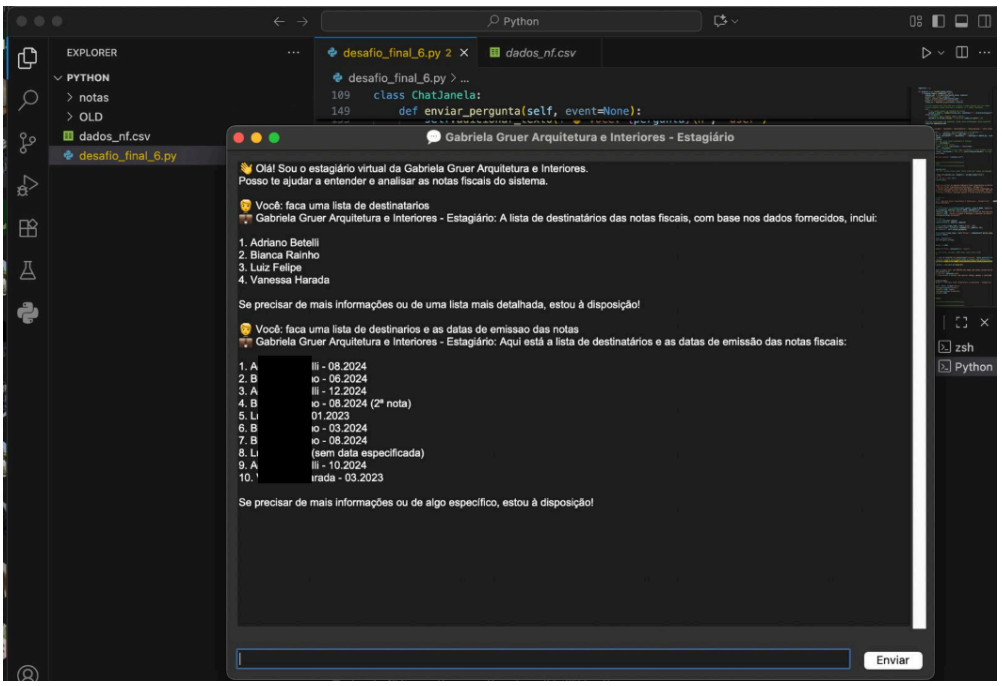
A ferramenta "Meu Estagiário" oferece diversos benefícios para MEIs e pequenos empresários:

- **Automação:** Reduz drasticamente o tempo gasto na organização e digitação de dados de notas fiscais.
- **Acessibilidade:** Oferece uma forma intuitiva de consultar dados financeiros sem a necessidade de conhecimento técnico avançado.
- **Insights Rápidos:** Permite fazer perguntas em linguagem natural sobre as finanças do negócio, obtendo respostas rápidas e diretas.
- **Organização Centralizada:** Consolida todos os dados de notas fiscais em um único arquivo estruturado e fácil de gerenciar.
- **Redução de Erros:** A automação e a inteligência da LLM minimizam erros humanos na transcrição de dados.
- **Foco no Negócio:** Libera o empresário para focar em atividades estratégicas, em vez de tarefas administrativas repetitivas.

Exemplos de Interações:

- "Qual o valor total das compras no mês de outubro?"
- "Quem foi o fornecedor com o maior número de notas?"
- "Liste todos os produtos comprados da empresa X."
- "Qual o total das notas emitidas para o destinatário Y?"
- "Mostre os valores totais por mês no último trimestre."

Veja abaixo o modelo real de interface com o usuário:



5. Conclusão

"Meu Estagiário" representa um avanço significativo na democratização da gestão financeira inteligente para pequenos negócios. Ao combinar a capacidade de processamento de documentos com a inteligência de modelos de linguagem, a ferramenta empodera empreendedores a terem um controle financeiro mais eficiente e a tomarem decisões mais informadas. A arquitetura flexível e a interface amigável tornam-na uma solução valiosa para o dia a dia do micro e pequeno empresário, comprovando o potencial da IA como uma aliada estratégica no desenvolvimento de negócios.