Relatório do Projeto Final - IAgentes

Nome do Grupo: IAgentes

Integrantes do Grupo: Antônio, Camila, Gabriela, Henrique, Jorge, Marcelo, Michel,
Rômulo, willian

Descrição do Tema Escolhido

Desenvolvemos um sistema inteligente para extração, validação e análise de dados de documentos fiscais brasileiros. A solução processa XML (NF-e), PDFs e imagens de DANFEs/cupons, combinando OCR (Tesseract) com IA Generativa (Google Gemini) para estruturar as informações em um schema validado (Pydantic), apresentando um dashboard interativo (Streamlit) com KPIs, tabela de itens e gráficos.

Público alvo

Empresas de qualquer porte, escritórios de contabilidade, auditores fiscais e times financeiros que lidam com grandes volumes de documentos fiscais (NF-e, DANFE, cupons) e necessitam de automação confiável, validação de consistência e visibilidade analítica.

Justificativa do Tema Escolhido

O processamento manual de documentos fiscais é lento, caro e sujeito a erros. Documentos apresentam formatos variados e alto volume mensal. A solução agrega valor ao reduzir tempo de processamento, erros de digitação e inconsistências, fornecendo validações automáticas, auditoria pós-extração e visualizações analíticas para tomada de decisão.

Detalhamento do que foi desenvolvido

Arquitetura e Fluxos:

- Upload de arquivos (XML, PDF, PNG/JPG) pela interface Streamlit.
- Para XML: remoção de namespaces, parsing com ElementTree e mapeamento direto para o schema Pydantic (DocumentoProcessado).
- Para PDF/Imagem: conversão para imagem (pdf2image), OCR com Tesseract (pt-BR) e envio do texto bruto ao Gemini com prompt

especializado.

- O LLM retorna JSON estruturado que é validado pelo Pydantic; em seguida, regras adicionais enriquecem/validam dados (regex de CFOP/CST, checagem de totais).
- O dashboard exibe KPIs, detalhes de remetente/receptor, tabela dos itens e gráficos com Plotly (por CFOP, composição de custos e top itens).

Operação da Solução:

1) Requisitos: Python 3.11+, Tesseract instalado, chave GOOGLE_API_KEY no .env.

2) Execução: `streamlit run main.py`.

3) Suba o arquivo fiscal desejado. Para XML, a extração é imediata. Para PDF/Imagem, o OCR é aplicado e os dados são extraídos via LLM.

4) Revise alertas de qualidade e, se necessário, utilize a edição manual para valores não capturados pelo LLM.

5) Exporte JSON completo e CSV dos itens (padrão ABNT).

Principais componentes (Stack): Streamlit (UI), Tesseract e OpenCV/pdf2image (OCR/visão), Gemini 2.5 Flash + LangChain (extração), Pydantic (validação), Pandas/Plotly (análises e gráficos). Temperatura do LLM: 0.1 para reduzir alucinações.

Tabela: Stack Tecnológico

Componente	Tecnologias
Interface	Streamlit
OCR	Tesseract, pdf2image, OpenCV
IA Generativa	Google Gemini 2.5 Flash, LangChain
Validação	Pydantic (schemas estruturados)
Análises	Pandas, Plotly Express

Tabela: KPIs e Gráficos Disponíveis

Indicador/Gráfico	Descrição
Valor Total do Documento	Extraído e exibido com formatação BRL
Nº de Itens	Contagem e ranking por valor
Totais de Impostos	ICMS, IPI, PIS, COFINS

Valor Aprox. Taxas	Somado por item ou total do documento

Link para o repositório do GitHub

> https://github.com/henriqueserafin/I2A2-Projeto_final

Tabela: Requisitos e Execução

Item	Detalhe
Requisitos	Python 3.11+, Tesseract instalado, GOOGLE_API_KEY no .env
Execução	`streamlit run main.py`
Entrada	XML (NF-e), PDF, PNG/JPG (DANFE/cupons)
Saídas	JSON completo e CSV dos itens (ABNT)

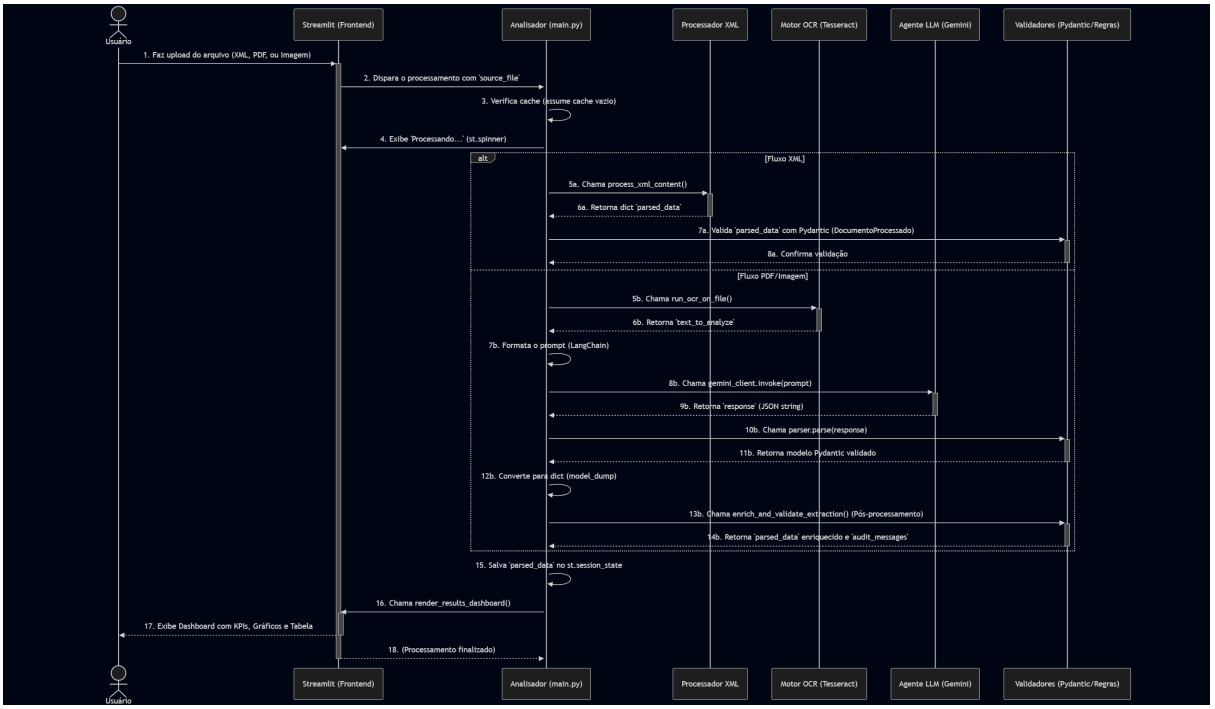
Tabela: KPIs exibidos no dashboard

Indicador	Fonte no pipeline	Observação
Valor Total do Documento	XML/LLM validado por Pydantic	Formato BRL
Nº de Itens	Parsing de itens	Ranking por valor
Totais de Impostos	ICMS, IPI, PIS, COFINS	Soma por documento
Taxas/Encargos	Item a item ou total	Agregado no dashboard

Fluxo Operacional

1. Abrir o app com `streamlit run main.py`.
2. Fazer upload de XML, PDF ou imagem.
3. Executar extração conforme o tipo de arquivo.
4. Validar alertas e ajustar manualmente se necessário.
5. Visualizar KPIs, itens e gráficos.
6. Exportar JSON e CSV dos itens.

Diagrama de sequencia:


Usuário
Streamlit (Frontend)
Analisador (main.py)
Processador XML
Motor OCR (Tesseract)
Agente LLM (Gemini)
Validadores (Pydantic/Regras)
1. Faz upload do arquivo (XML, PDF, ou Imagem)
2. Dispara o processamento com 'source_file'
3. Verifica cache (assume cache vazio)
4. Exibe 'Processando...' (st.spinner)
alt
[Fluxo XML]
5a. Chama process_xml_content()
6a. Retorna dict 'parsed_data'
7a. Valida 'parsed_data' com Pydantic (DocumentoProcessado)
8a. Confirma validação
[Fluxo PDF/Imagem]
5b. Chama run_ocr_or_file()
6b. Retorna 'text_to_analyze'
7b. Formata o prompt (LangChain)
8b. Chama gemini_client.invoke(prompt)
9b. Retorna 'response' (JSON string)
10b. Chama parser.parse(response)
11b. Retorna modelo Pydantic validado
12b. Converte para dict (model_dump)
13b. Chama enrich_and_validate_extraction() (Pós-processamento)
14b. Retorna 'parsed_data' enriquecido e 'audit_messages'
15. Salva 'parsed_data' no st.session_state
16. Chama render_results_dashboard()
17. Exibe Dashboard com KPIs, Gráficos e Tabela
18. (Processamento finalizado)