**Relatório da aplicação**

**Agente Fiscal: CSV/XML → SQLite → Perguntas em Português**

**1) Visão geral (o que o sistema se propõe a fazer)**

O sistema permite **carregar dados fiscais estruturados** a partir de **CSVs, ZIPs com CSVs e XMLs**, armazená-los automaticamente em um **banco SQLite** (uma tabela por arquivo) e, em seguida, **consultá-los em linguagem natural** (PT-BR).  
O agente converte a pergunta para **SQL** e executa no banco, retornando **tabelas e análises** úteis (ex.: faturamento, produtos mais vendidos, impostos, desempenho por cliente/fornecedor).

**Benefício principal:** criar um pipeline simples e universal para **ingestão** + **consulta inteligente** de dados fiscais para **KPIs contábeis e fiscais**, relatórios gerenciais e integração posterior com ERPs/BI.

**2) Principais funcionalidades**

* **Upload multi-arquivo**: CSV(s), **ZIP** com CSVs e **XMLs**.
* **Criação automática do banco**: um arquivo ⇒ **uma tabela** (nome higienizado).
  + Para **XML de NF-e**: cria **duas tabelas** (\*\_cabecalho, \*\_itens).
  + Para **XML no formato “Envio → detList → det → prod”**: cria \*\_cabecalho (mínimo 1 coluna) + \*\_itens com cProd, xProd, qCom, vUnCom, vProd.
  + Para XML genérico: **flatten** em uma tabela única.
* **Normalização leve**: nomes de colunas sem espaços; números convertidos para float quando possível.
* **Pré-visualização de dados** e **PRAGMA** (schema) direto na UI.
* **Agente NL → SQL**: perguntas em PT-BR; gera e **executa** o SELECT (com limpeza para evitar textos extras).
* **SQL manual (opcional)**: campo para você rodar SELECT diretamente.
* **Logs do agente**: configuráveis (verbose/debug) para auditoria de como o SQL foi gerado.

**3) Arquitetura resumida**

* **Frontend/Orquestração**: Streamlit.
* **Persistência**: SQLite local (dados\_fiscais.db, nome ajustável).
* **Ingestão**:
  + CSV: leitura **resiliente** (tenta sep=None/engine python + encodings comuns; fallback para ,).
  + ZIP: extração em memória e ingestão dos CSVs.
  + XML:
    - **NF-e**: mapeia nós ide, emit, dest, total(ICMSTot), det/prod.
    - **Envio(det/prod)**: extrai cabeçalho (quando houver) e **explode** os itens para linhas.
    - Genérico: flatten profundo.
* **IA/Agente**: LangChain + LLM (config via .env).
  + Prompt do agente instruído para **gerar apenas SQL SELECT**.
  + **Pós-processamento**: extrai apenas o SQL a partir do primeiro SELECT (remove texto/markdown) e executa.
* **Observabilidade**: verbose/debug do LangChain habilitáveis para imprimir o raciocínio/ferramentas no terminal.

**4) Como usar (passo a passo)**

**4.1. Preparar ambiente**

1. (Opcional) Criar venv:
2. python -m venv .venv
3. .\.venv\Scripts\activate # Windows
4. # ou
5. source .venv/bin/activate # macOS/Linux
6. Instalar dependências:
7. pip install -r requirements\_any\_csv.txt
8. Configurar .env (na pasta do app):
9. LLM\_API\_KEY=SEU\_TOKEN
10. LLM\_MODEL\_NAME=gpt-4o-mini # ajuste conforme seu provedor
11. LLM\_BASE\_URL=https://api.openai.com/v1

**4.2. Executar**

streamlit run app.py # ou app\_any\_csv\_xml\_v2.py, conforme o seu arquivo

**4.3. Carregar dados**

* Na UI: **Upload de CSV/XML/ZIP** (multi-arquivo).
* Se “Recriar tabelas” estiver marcado, tabelas homônimas são **substituídas**.
* O app mostra **prévia** e lista as **tabelas** detectadas.

**4.4. Consultar em linguagem natural**

* Área “🤖 Faça sua pergunta em Português”.
* Exemplos:
  + “**Quais os 5 produtos mais vendidos?**”
  + “**Top 10 por valor vendido**”
  + “**Total de faturamento por mês**”
* O sistema:
  + gera SQL (apenas SELECT),
  + **limpa** qualquer texto extra,
  + **executa** no SQLite,
  + mostra o **resultado**.

**4.5. Consultar em SQL manual (opcional)**

* Campo “Executar SQL” para SELECT direto.
* Exemplos:
* SELECT xProd, SUM(qCom) AS total\_qtd
* FROM envio\_construcao\_10k\_itens\_itens
* GROUP BY xProd
* ORDER BY total\_qtd DESC
* LIMIT 5;

**5) Casos de uso e indicadores gerados**

* **Vendas/Produtos**: top itens por **quantidade** (SUM(qCom)) ou por **valor** (SUM(vProd)).
* **Financeiro/Fiscal** (NF-e): **faturamento** (SUM(vNF)), **base de produtos** (SUM(vProd)), **impostos** (SUM(vICMS, vPIS, vCOFINS, vIPI)), **margens** (se houver campos de custo), **distribuição por UF/cliente**.
* **Clientes/Fornecedores**: ranking por dest\_CNPJ/emit\_CNPJ, volume por período, ticket médio.

**6) Esquemas de tabelas (principais)**

**6.1. CSV (genérico)**

* Tabela: <nome\_do\_arquivo>
* Colunas: conforme o header do CSV (espaços viram \_).

**6.2. XML NF-e**

* <base>\_cabecalho:  
  nNF, serie, dhEmi, tpNF, natOp, cUF, tpAmb, mod, emit\_CNPJ, emit\_xNome, emit\_UF, dest\_CNPJ, dest\_xNome, dest\_UF, vNF, vProd, vICMS, vIPI, vPIS, vCOFINS
* <base>\_itens:  
  nNF, cProd, xProd, NCM, CFOP, uCom, qCom, vUnCom, vProd

**6.3. XML “Envio → detList → det → prod”**

* <base>\_cabecalho: pelo menos arquivo (e quaisquer metadados que o topo do XML tiver).
* <base>\_itens: cProd, xProd, qCom, vUnCom, vProd (tipados como numéricos quando possível).

**7) Boas práticas de uso**

* **Padronize nomes** de arquivos para refletir o conteúdo (ex.: 202401\_nfs\_itens.csv).
* **Evite colunas sem nome** ou misturas de tipos no mesmo campo no CSV.
* Para **XML**, use **o arquivo de fato** (não HTML do DANFE, nem PDF ou ZIP renomeado).
* Para perguntas ao agente, use **termos que existam nas colunas** (ex.: xProd, qCom, vProd).

**8) Troubleshooting (erros comuns e correções)**

| **Sintoma** | **Causa** | **Correção** |
| --- | --- | --- |
| The 'low\_memory' option is not supported with the 'python' engine | read\_csv com engine="python" não aceita low\_memory | Removido do código (feito). |
| ModuleNotFoundError: xmltodict | Pacote não instalado | pip install xmltodict ou pip install -r requirements\_any\_csv.txt. |
| XML inválido: not well-formed (invalid token) | Arquivo não é XML (HTML/PDF/ZIP) ou tem BOM/lixo | Versão robusta remove BOM, detecta formatos; usar o XML correto. |
| Consulta não executa e aparece texto “A tabela … contém …” | O LLM devolveu explicação + SQL | **Limpador de SQL** implementado (pega só o SELECT). |
| near ")": syntax error ao gravar \_cabecalho | DataFrame sem colunas | Garantido arquivo como coluna mínima ou pular a criação (feito). |
| Não vejo o “pensamento” no terminal | Verbose/desligado | verbose=True no LLM e no agente + set\_debug(True)/set\_verbose(True). |

**9) Segurança e configuração**

* As chaves do LLM ficam no **.env** (não commitar).
* O SQLite fica **local**; proteja a pasta do projeto se houver dados sensíveis.
* Para ambientes corporativos, considere apontar para modelos **self-hosted**/VPC e armazenar o banco em volumes seguros/criptografados.

**10) Limitações atuais**

* **Inferência de joins** depende de nomes usuais (ex.: nNF, cProd, CNPJ); para schemas diferentes, o agente pode errar.
* **XML genérico** ainda vira **uma** tabela (flatten) se não casar com NF-e ou Envio(det/prod).
* Não há ainda **catálogo** de metadados/linhagem ou **mapeamento de chaves** persistente entre tabelas heterogêneas (opção futura).

**11) Roadmap sugerido (próximas melhorias)**

1. **Dicionário de colunas** (“alias”) por domínio fiscal para orientar o agente (ex.: tratar descricao como xProd).
2. **Config UI de delimitador/encoding** para CSV (override manual).
3. **Suporte a NFS-e** (padrões mais comuns) com tabelas normalizadas.
4. **Validações fiscais**: regras de integridade (ex.: vProd ≈ qCom \* vUnCom).
5. **Camada de métricas** (views) com KPIs prontos: faturamento mês, mix de produtos, ticket médio, top clientes.
6. **Exportação** (CSV/Excel) de resultados e integração nativa com Power BI.

**12) Exemplos de perguntas úteis**

* “**Top 5 produtos por quantidade**”
* “**Top 10 por valor vendido**”
* “**Total de faturamento (vNF) por mês**”
* “**Ranking de clientes por valor total**”
* “**Quantidade total por NCM**”
* “**Média de preço por produto (vUnCom)**”

**Conclusão**

Um **pipeline único** para ingerir dados fiscais (CSV/XML), **estruturar** em SQLite e **perguntar em PT-BR** com geração/execução de SQL. Isso acelera a criação de **KPIs e análises fiscais/contábeis** e prepara o terreno para **integração com BI/ERP** e automação de obrigações acessórias.