



Agentes Autônomos com Redes Generativas

Projeto de Conclusão do Curso

Grupo: Átomos Digitais I2A2

Componentes do Grupo:

Nome	Telefone	Email
Antônio José	553186025219	aj.rodrigues@ymail.com
Helena	5531995548033	hlanza@fiemg.com.br
Laiane	5591993168673	laianesilvasousa03@gmail.com
Marcos Vinicius	558589444479	vinicius.uchoa2002@gmail.com

Projeto de Automação em Avaliação e Auditoria Fiscal, Tributária/Contábil para Empresas

- Objetivo:**

O presente projeto tem como objetivo principal o desenvolvimento e implementação de ferramentas automatizadas para avaliação e auditoria dos processos de rotinas obrigatórias das empresas. A iniciativa visa identificar proativamente inconsistências nas áreas fiscal e tributária/contábil.

A automação proposta possibilitará a construção, análise, manutenção, orientação e adequação de procedimentos, garantindo que as empresas atendam plenamente às demandas da Receita Federal. Em última instância, o projeto visa aprimorar o controle sobre o cumprimento das normas e regras aplicáveis às operações empresariais.

- **Justificativa:**

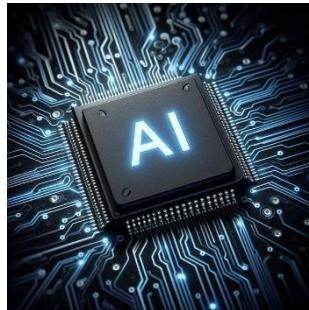
A saúde financeira e jurídica de qualquer negócio, precisa levar em consideração, avaliações e auditorias internas de seus processos rotineiros, para detalhamentos de suas principais obrigações, entre elas, as fiscais e as tributárias/contábeis, que as empresas brasileiras devem cumprir na garantia de estarem em conformidade legal e minimizar riscos em auditorias e fiscalizações advindas da Receita Federal.

Automatizar esses controles e ferramentas, trará não só agilidade nos processos empresariais nos campos Fiscais, mas também facilitará o filtro de irregularidades possíveis, prevenindo prejuízos com multas, autuações, necessidades de assinaturas de Termos de Ajustes de Condutas e ajuizamentos de causas.

- **Beneficiários Potenciais do Projeto**

Os principais beneficiários desta iniciativa abrangem múltiplos stakeholders, refletindo o impacto abrangente da automação na conformidade e eficiência empresarial.

- **Empresas:** A implementação das ferramentas resultará em uma redução significativa de riscos de autuações e multas, otimizando processos internos e liberando recursos. A tomada de decisões se tornará mais assertiva com base em dados qualificados, podendo gerar economia financeira pela recuperação de valores pagos indevidamente ou prevenção de desembolsos excessivos. A conformidade legal será reforçada, contribuindo para a solidez e confiabilidade da imagem corporativa.
- **Profissionais das Áreas Fiscal e Contábil:** Para os profissionais internos e externos, a automação trará um aumento substancial na eficiência, minimizando erros manuais e permitindo que se dediquem a atividades de maior valor agregado, como análise estratégica e consultoria. O acesso a dados estruturados e confiáveis facilitará a identificação de pontos críticos, promovendo o desenvolvimento profissional em habilidades tecnológicas e analíticas.



ATOMIZE :Agente de Análise de Notas Fiscais: Validação e Auditoria

Desenvolvimento do Projeto

☒ Documentação de Arquitetura - Sistema Auditor Fiscal

📋 Sumário Executivo

Este documento descreve a arquitetura do **Sistema Auditor Fiscal**, uma aplicação web desenvolvida em Python que utiliza inteligência artificial para análise e auditoria de dados fiscais, especificamente Notas Fiscais Eletrônicas (NF-e). O sistema permite upload de arquivos CSV e XML, validação automática, detecção de irregularidades fiscais e análise inteligente através de consultas em linguagem natural.

🎯 Visão Geral do Sistema

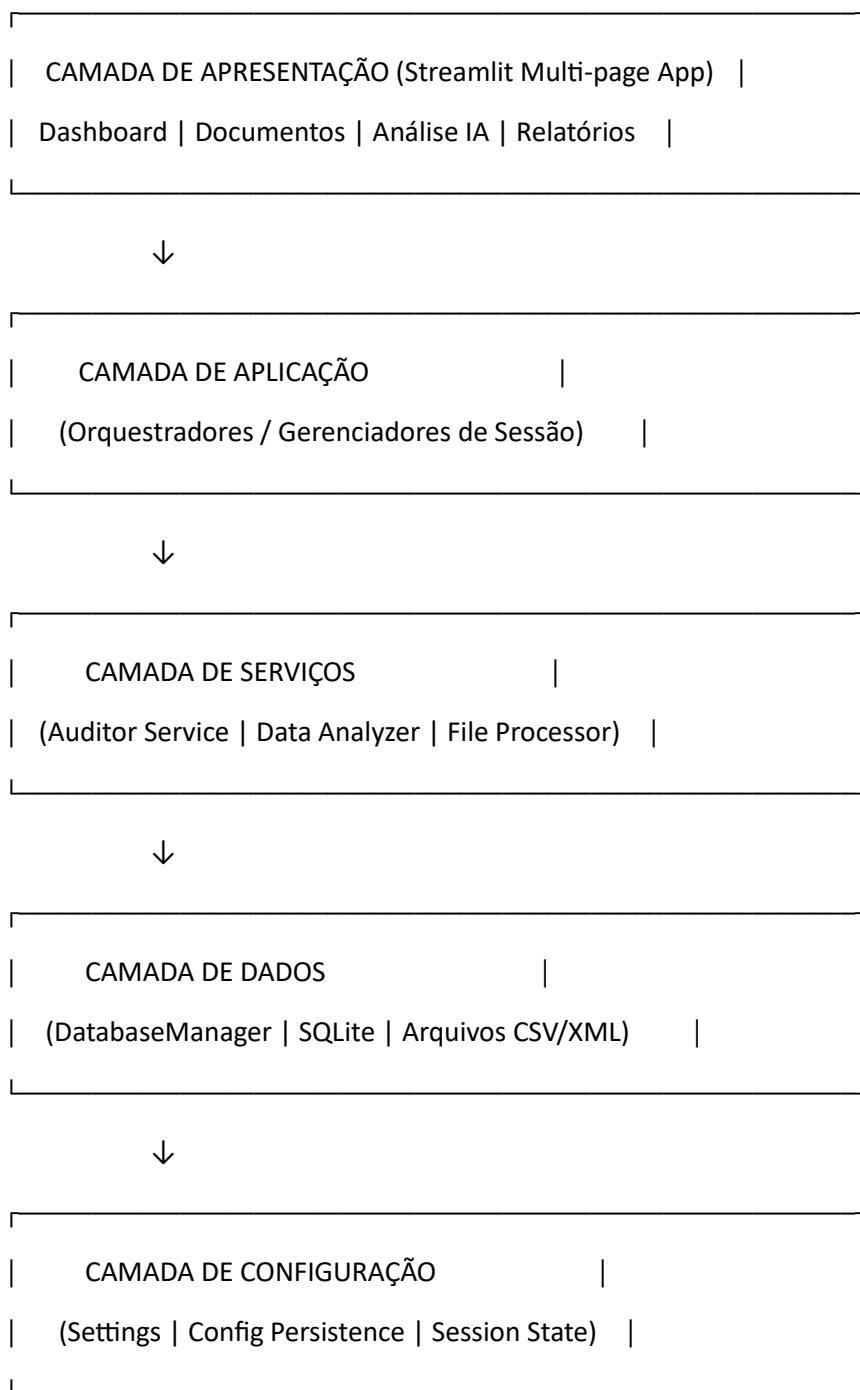
O Sistema Auditor Fiscal é uma solução completa para auditoria fiscal automatizada que oferece:

- **Upload e Processamento:** Suporte a arquivos CSV e XML (NF-e)
- **Validação Automática:** Detecção de duplicatas, campos obrigatórios e cálculos
- **Análise com IA:** Consultas inteligentes em linguagem natural usando OpenAI GPT
- **Dashboard Executivo:** Métricas fiscais em tempo real
- **Relatórios Detalhados:** Visualizações e exportação de dados
- **Persistência:** Banco de dados SQLite para armazenamento estruturado

Arquitetura Geral

Padrão Arquitetural: Arquitetura em Camadas (Layered Architecture)

O sistema segue uma arquitetura em camadas bem definida:



Estrutura do Projeto

```
auditor_fiscal/
├── app.py          # Ponto de entrada principal
├── 0_Inicio.py    # Aplicação Streamlit principal
├── pages/          # Páginas do Streamlit
│   ├── 1_🏠_Dashboard.py    # Dashboard com métricas fiscais
│   ├── 2_📋_Notas.py       # Listagem de notas fiscais
│   ├── 3_📤_Documentos.py  # Upload de documentos
│   ├── 4_🔍_Análise_IA.py  # Análise com IA
│   ├── 5_📊_Relatórios.py  # Relatórios e visualizações
│   └── 6_⚙️_Configurações.py  # Configurações do sistema
└── src/
    ├── web_interface/      # Componentes de interface
    │   ├── components/
    │   │   ├── metrics.py    # Componentes de métricas
    │   │   └── sidebar.py    # Menu lateral
    │   └── utils/
    │       └── session_manager.py # Gerenciamento de sessão
    ├── database/           # Gerenciamento de banco
    │   └── db_manager.py     # DatabaseManager SQLite
    ├── services/            # Serviços de negócio
    │   └── auditor_service.py # Serviço de auditoria fiscal
    ├── ai_service/          # Serviços de IA
    │   ├── data_analyzer.py  # Análise de dados com IA
    │   ├── fiscal_analyzer.py # Analisador fiscal v1
    │   └── fiscal_analyzer_v2.py # Analisador fiscal v2 (validador automático)
    └── analysis_schema.py   # Schemas de análise
        ├── data_loader/      # Carregamento de dados
        └── file_processor.py  # Processamento de arquivos
```

```
| |   |-- nfe_xml_parser.py    # Parser de XML NFe
| |   |-- nfe_xml_processor.py # Processador completo de XML
|   |-- config/              # Configurações
|       |-- settings.py      # Configurações gerais
|       |-- config_persistence.py # Persistência de configurações
|
|   |-- data/
|       |-- input/            # Dados de entrada
|           |-- *.csv          # Arquivos CSV
|           |-- *.xml          # Arquivos XML
|
|   |-- auditor_database.db  # Banco SQLite
```

Padrões de Design Utilizados

1. Facade Pattern

- **Onde:** SessionManager, AuditorService
- **Objetivo:** Simplificar a interface complexa dos módulos internos
- **Benefício:** Interface única e consistente para as páginas Streamlit

2. Repository Pattern

- **Onde:** DatabaseManager
- **Objetivo:** Abstração da camada de acesso a dados
- **Benefício:** Isolamento da lógica de negócio da persistência SQLite

3. Strategy Pattern

- **Onde:** Diferentes analisadores fiscais
(fiscal_analyzer.py, fiscal_analyzer_v2.py)
- **Objetivo:** Permitir algoritmos intercambiáveis de análise
- **Benefício:** Extensibilidade e facilidade de testes

4. Singleton Pattern

- **Onde:** Gerenciamento de configurações e conexões de banco
- **Objetivo:** Garantir uma única instância de recursos compartilhados
- **Benefício:** Controle de recursos e estado compartilhado

5. Dependency Injection

- **Onde:** Inicialização de serviços através de construtores
- **Objetivo:** Baixo acoplamento e alta testabilidade

- **Benefício:** Facilita testes unitários e substituição de implementações
-

Componentes Principais

Camada de Apresentação

Streamlit Multi-page Application

- **Estrutura:** 6 páginas principais organizadas por funcionalidade
- **Navegação:** Menu lateral automático baseado em arquivos em pages/
- **Estado:** Gerenciado via st.session_state e SessionManager

Páginas do Sistema

1.  **Dashboard** (1_  _Dashboard.py)
 - Métricas fiscais em tempo real
 - Status das auditorias
 - Histórico de análises
 - Alertas de irregularidades
2.  **Notas** (2_  _Notas.py)
 - Listagem de notas fiscais carregadas
 - Filtros e busca
 - Detalhes de cada nota
3.  **Documentos** (3_  _Documentos.py)
 - Upload de arquivos CSV e ZIP
 - Processamento automático
 - Preview dos dados carregados
4.  **Análise IA** (4_  _Análise_IA.py)
 - Interface de consultas em linguagem natural
 - Análise com OpenAI GPT
 - Histórico de consultas
 - Sugestões de perguntas
5.  **Relatórios** (5_  _Relatórios.py)
 - Dashboard executivo
 - Visualizações interativas (Plotly)
 - Exportação de dados
 - Análise estatística

6. Configurações (6_Configurações.py)

- Configuração de API OpenAI
- Gerenciamento de dados
- Informações do sistema
- Logs e debug

Componentes Reutilizáveis

- **Metrics:** Componentes de métricas e KPIs (components/metrics.py)
- **Sidebar:** Menu de navegação (components/sidebar.py)

Camada de Aplicação

SessionManager

- **Responsabilidades:**
 - Gerenciamento de estado da sessão
 - Verificação de status do sistema
 - Métricas do sistema
 - Configurações persistentes

Ponto de Entrada

- **app.py:** Importa e executa 0_Inicio.py
- **0_Inicio.py:** Configuração inicial e página de boas-vindas

Camada de Serviços

AuditorService

- **Responsabilidades:**
 - Gerenciamento de resultados de auditoria
 - Persistência de análises
 - Agregação de métricas fiscais
 - Consultas ao histórico de auditorias

Funcionalidades:

- `save_audit_result()`: Salva resultado de auditoria
- `get_audit_results()`: Recupera resultados
- `get_fiscal_metrics()`: Calcula métricas fiscais agregadas
- `get_audit_history()`: Retorna histórico de auditorias

DataAnalyzer

- **Responsabilidades:**

- Orquestração de agentes CrewAI
- Geração de consultas SQL a partir de perguntas
- Análise de resultados com IA
- Formatação de respostas

Funcionalidades:

- `generate_sql_query()`: Gera SQL a partir de pergunta
- `format_result()`: Formata resultados em linguagem natural
- `analyze_question()`: Análise completa (SQL + formatação)

FiscalAnalyzer (v1 e v2)

- **FiscalAnalyzer v1**: Análise usando IA diretamente
- **FiscalAnalyzer v2**: Validação automática + IA quando necessário

Estratégia v2 (Recomendada):

1. Validação automática (SQL) - SEM IA
2. Análise IA apenas quando necessário
3. Fato-verificação pós-análise

FileProcessor

- **Responsabilidades:**
 - Processamento de arquivos CSV
 - Detecção automática de encoding e separador
 - Validação de estrutura

NFEXMLProcessor

- **Responsabilidades:**
 - Parsing de XML de NF-e
 - Extração de dados estruturados
 - Validação de schema XML
 - Tratamento de erros de estrutura

Camada de Dados

DatabaseManager

- **Responsabilidades:**
 - Gerenciamento de conexões SQLite thread-safe
 - Criação dinâmica de tabelas a partir de CSV
 - Execução de consultas SQL

- Exportação de banco de dados
- Cache de schema

Características Especiais:

- **Thread Safety:** Conexões locais por thread (threading.local)
- **Schema Dinâmico:** Tabelas criadas automaticamente
- **Campos Adicionais NFe:** Tabelas NFe com campos de auditoria
- **Cache Inteligente:** Detecção de mudanças via hash de arquivos

Estrutura de Tabelas

Tabelas Dinâmicas:

- Criadas a partir de arquivos CSV carregados
- Nomes sanitizados automaticamente
- Tipos inferidos ou padronizados para TEXT

Tabelas NFe Especiais:

- nfe_notas_fiscais: Campos adicionais de auditoria
- nfe_itens_nota: Itens com campos fiscais detalhados

Tabela de Auditoria:

- auditor_results: Resultados de análises e auditorias
 - Chave de acesso
 - Métricas (documentos, valores, inconsistências)
 - Tempo de processamento
 - Resultado da IA (JSON)
 - Status da auditoria

🔗 Fluxos de Dados Principais

Fluxo de Upload e Processamento

Usuário faz Upload (CSV/XML/ZIP)



FileProcessor / NFEXMLProcessor



Validação de Formato e Estrutura



Parsing e Extração de Dados

↓
DatabaseManager
↓
Sanitização de Colunas e Criação de Tabelas
↓
Inserção no SQLite
↓
Atualização de Métricas
↓
Interface Atualizada

Fluxo de Análise com IA

Usuário faz Pergunta em Linguagem Natural
↓
SessionManager verifica API Key e Banco
↓
DataAnalyzer recebe pergunta
↓
Obter Schema do Banco (DatabaseManager)
↓
CrewAI Agent (SQL Analyst) gera SQL
↓
DatabaseManager executa Query
↓
CrewAI Agent (Data Analyst) formata Resultado
↓
Salvar no Histórico (AuditorService)
↓
Exibir Resposta Formatada

Fluxo de Auditoria Automática

Seleção de Chave de Acesso (NFe)

↓

FiscalAnalyzer v2 inicia validação

↓

AutoValidator (SQL)

 |— Validação de Duplicatas

 |— Campos Obrigatórios

 └— Cálculos Fiscais

↓

Se problemas encontrados → Análise IA

↓

Fato-verificação dos Resultados IA

↓

Salvar Resultado (AuditorService)

↓

Atualizar Dashboard

💡 Integrações Externas

OpenAI API

Uso:

- Geração de consultas SQL a partir de perguntas
- Análise de dados fiscais
- Formatação de respostas em linguagem natural
- Detecção de irregularidades

Modelos:

- gpt-4o-mini (padrão para consultas SQL)
- Configurável via settings

Via CrewAI:

- Agentes especializados (SQL Analyst, Data Analyst)
- Orquestração de tarefas complexas

- Processo sequencial ou hierárquico

SQLite

Versão: SQLite3 com `pysqlite3` (opcional)

Características:

- Banco em arquivo (`data/auditor_database.db`)
 - Thread-safe com conexões locais
 - Suporte a transações
 - Exportação de banco completo
-

Arquitetura de Dados

Banco de Dados SQLite

Estrutura Dinâmica

- Tabelas criadas a partir de CSV
- Schema inferido dos dados
- Sanitização automática de nomes

Tabelas Fixas

- `auditor_results`: Resultados de auditoria
- Tabelas NFe com estrutura especializada

Cache e Performance

- Hash de arquivos para detectar mudanças
- Índices em campos frequentemente consultados
- Cache de schema do banco
- Queries otimizadas

Persistência de Configurações

- **Session State:** Streamlit `st.session_state`
 - **Config Files:** YAML/JSON (via `config_persistence.py`)
 - **Environment Variables:** `.env` files
 - **Banco de Dados:** Tabelas de configuração (futuro)
-

Segurança e Configuração

Gerenciamento de Segredos

- **API Keys:** Armazenadas em `st.session_state` (memória)
- **Configuração Persistente:** Arquivo local (criptografado opcional)
- **Isolamento:** Cada sessão tem suas próprias configurações

Validação de Dados

- **Entrada:** Validação de formato antes do processamento
 - **Sanitização:** Escape de SQL e sanitização de nomes
 - **Schema Validation:** Validação de XML via schemas
-

Monitoramento e Logging

Sistema de Logging

- **Níveis:** DEBUG, INFO, WARNING, ERROR, CRITICAL
- **Formatos:** Estruturados com timestamp e contexto
- **Destinos:** Console (via Streamlit) e arquivo (opcional)
- **Contexto:** Logger específico por módulo

Métricas e Monitoramento

- **Sistema Metrics:** Status de banco, API, tabelas, consultas
 - **Performance:** Tempo de processamento, queries executadas
 - **Auditoria:** Histórico completo de análises
 - **Health Checks:** Verificação de saúde do sistema via SessionManager
-

Padrões de Execução

Inicialização

Ordem de Inicialização:

1. Carregamento de configurações (`settings.py`)
2. Inicialização do SessionManager
3. Verificação e inicialização do DatabaseManager
4. Carregamento automático de dados CSV (opcional)
5. Inicialização de serviços de IA (quando necessário)
6. Renderização da interface Streamlit

Processamento

- **Síncrono:** Para upload e processamento local
- **Assíncrono:** Para chamadas à API OpenAI (via `asyncio`)

- **Thread-Safe:** Conexões de banco isoladas por thread
-

Tecnologias e Dependências

Stack Tecnológico

Linguagem:

- Python 3.8+

Framework Web:

- Streamlit 1.28+

Banco de Dados:

- SQLite3 (pysqlite3 opcional)

Processamento de Dados:

- Pandas 2.0+
- xml.etree.ElementTree (parsing XML)

Inteligência Artificial:

- OpenAI API (via langchain-openai)
- CrewAI (para agentes multi-agente)
- LangChain (para orquestração de LLMs)

Visualização:

- Plotly (gráficos interativos)
- Streamlit components

Utilitários:

- python-dotenv (variáveis de ambiente)
 - pathlib (manipulação de caminhos)
 - logging (sistema de logs)
-

Convenções de Código

Estrutura de Arquivos

- Módulos organizados por responsabilidade
- Nomes descritivos em português para páginas
- Docstrings em português
- Type hints para melhor documentação

Nomenclatura

- **Classes:** PascalCase (DatabaseManager)
 - **Funções/Métodos:** snake_case (execute_query)
 - **Constantes:** UPPER_SNAKE_CASE (MAX_ATTEMPTS)
 - **Variáveis:** snake_case (db_connection)
-

🎓 Boas Práticas Implementadas

Princípios SOLID

- **S (Single Responsibility):** Cada módulo tem responsabilidade única
- **O (Open/Closed):** Extensível via Strategy Pattern
- **L (Liskov Substitution):** Implementações intercambiáveis
- **I (Interface Segregation):** Interfaces específicas
- **D (Dependency Inversion):** Injeção de dependências

DRY (Don't Repeat Yourself)

- Componentes reutilizáveis (components/)
- Funções utilitárias compartilhadas
- Configurações centralizadas

KISS (Keep It Simple, Stupid)

- Soluções diretas e simples
 - Evitação de over-engineering
 - Código legível e manutenível
-

🚧 Roadmap de Melhorias Arquiteturais

Curto Prazo

- [] Implementação de testes unitários
- [] Documentação de APIs com docstrings
- [] Melhoria de tratamento de erros com retry logic
- [] Cache de consultas frequentes

Médio Prazo

- [] Migração para arquitetura assíncrona completa
- [] Implementação de cache distribuído
- [] Dashboard de monitoramento avançado

- [] Exportação de relatórios em PDF

Longo Prazo

- [] Suporte a múltiplos bancos de dados
 - [] Arquitetura de micro serviços
 - [] Sistema de plugins/extensões
 - [] API REST para integração externa
-

Referências e Documentação Adicional

- Documentação específica: auditor_fiscal/README.md
- Documentação de páginas: auditor_fiscal/README_PAGES.md
- Diagramas de fluxo: Ver arquivo FLUXOGRAMAS.md
-

FLUXOGRAMA DO SISTEMA AUDITOR FISCAL

Equipe: Átomos Digitais

Este documento visa demostrar o funcionamento do sistema, desenvolvido para análise e auditoria de dados fiscais utilizando Inteligência artificial.

Etapas:

- **Fluxo de Inicialização**
- **Inicialização do Banco de Dados**
- **Upload e Processamento de Documentos**
- **Análise com IA**
- **Auditoria Automática de Notas Fiscais**

1. FLUXO DE INICIALIZAÇÃO

Nessa Etapa, com intuito de inicializar o sistema é necessário passar pelas etapas apresentadas no diagrama abaixo.

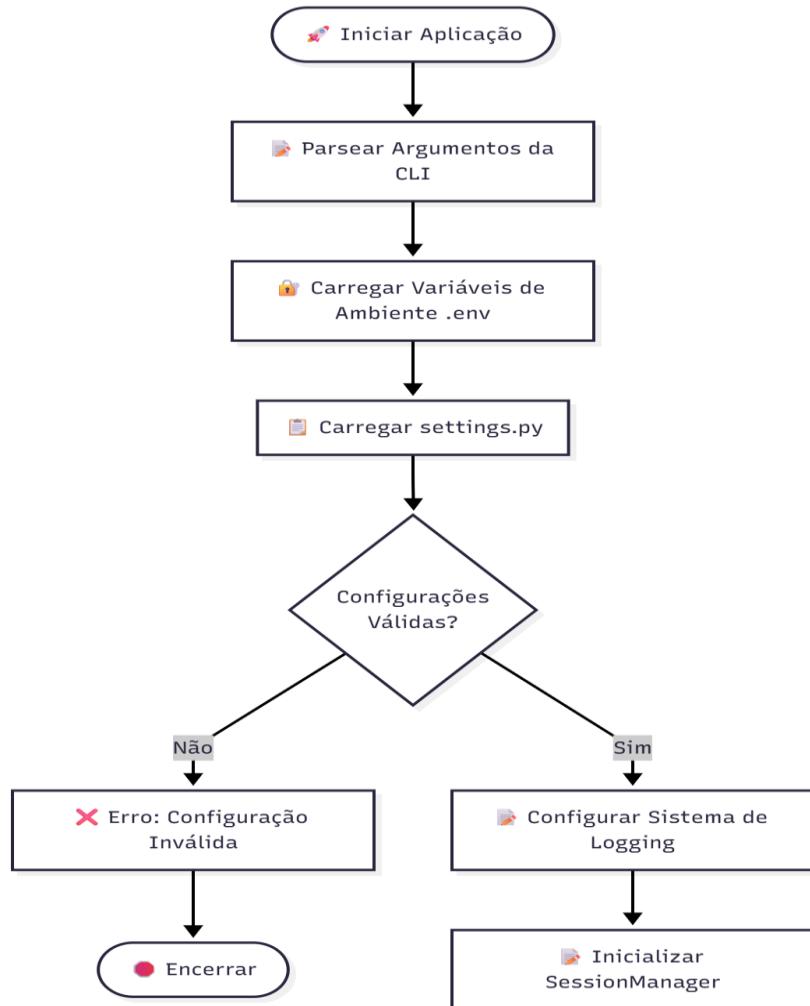


Diagrama 1 – Fluxo de inicialização do Sistema.

2. INICIALIZAÇÃO DO BANCO DE DADOS

A inicialização do banco de dados visa otimizar o processamento de resposta do agente diante das atividades que o sistema dispõe. O banco de dados escolhido foi SQLite versão 3 devido apresentar suporte a transações mais rápida e com memória. O diagrama 2, esclarece o funcionamento dessa etapa do projeto.

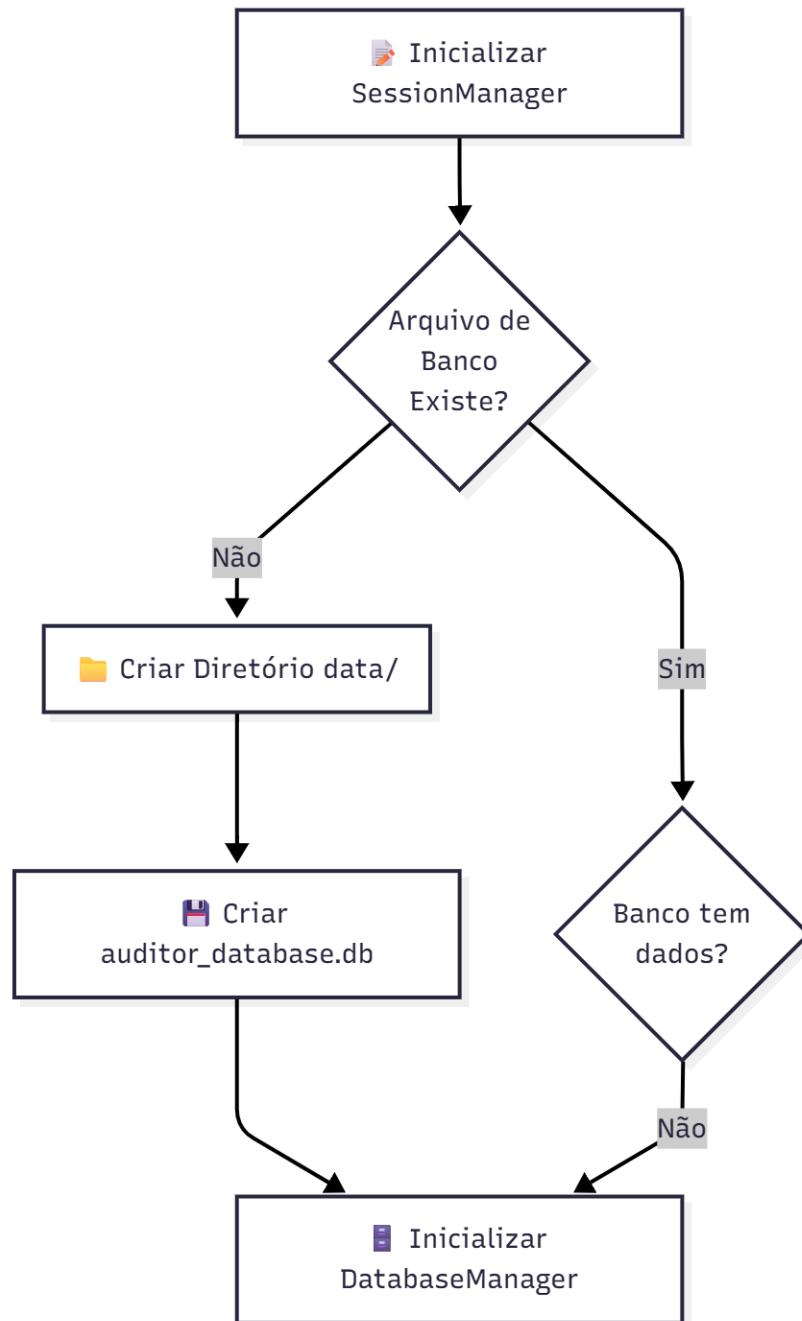
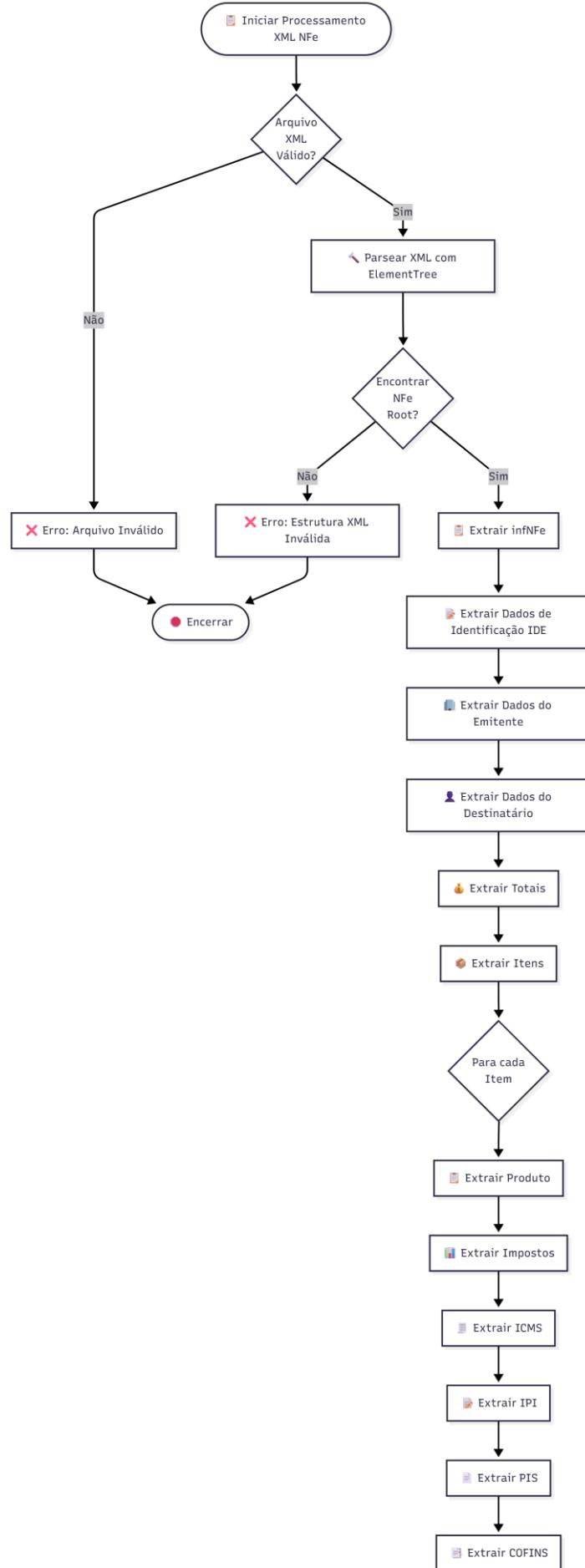


Diagrama 2 - Fluxograma da inicialização do Banco de dados

3. UPLOAD E PROCESSAMENTO DE DOCUMENTOS

Nessa etapa, é apresentado o processo de upload dos arquivos XML para análise. O diagrama 3 está dividido em duas partes.



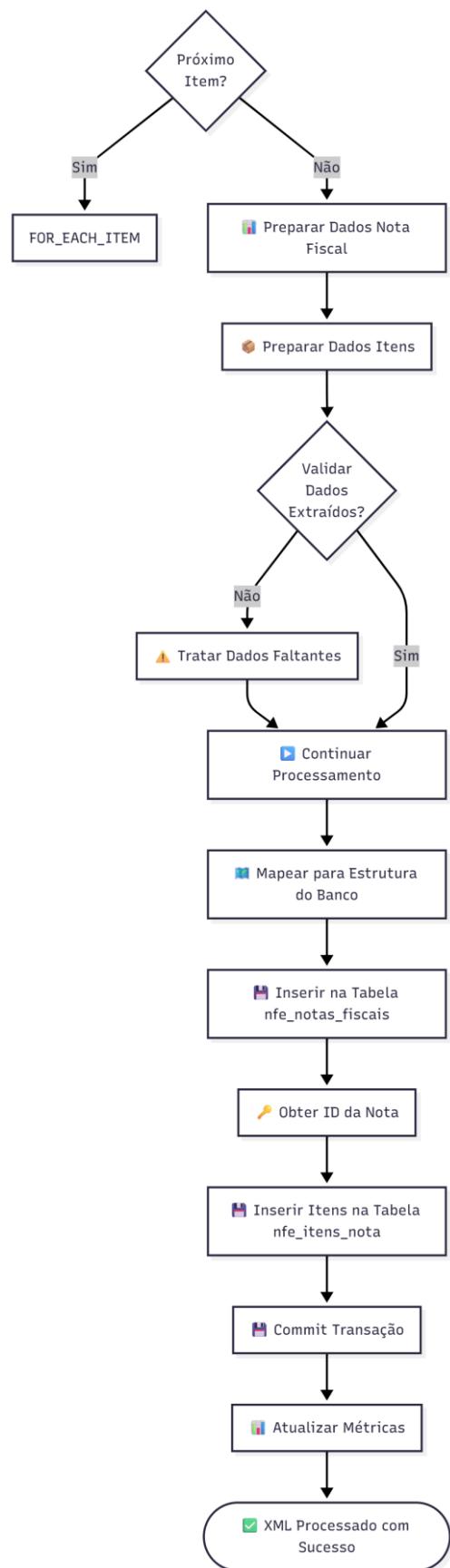


Diagrama 3 – Fluxograma de upload e carregamento dos dados

4. ANÁLISE COM IA

Nessa etapa, o usuário deve possuir chave de acesso para utilizar o LLM do ChatGPT.

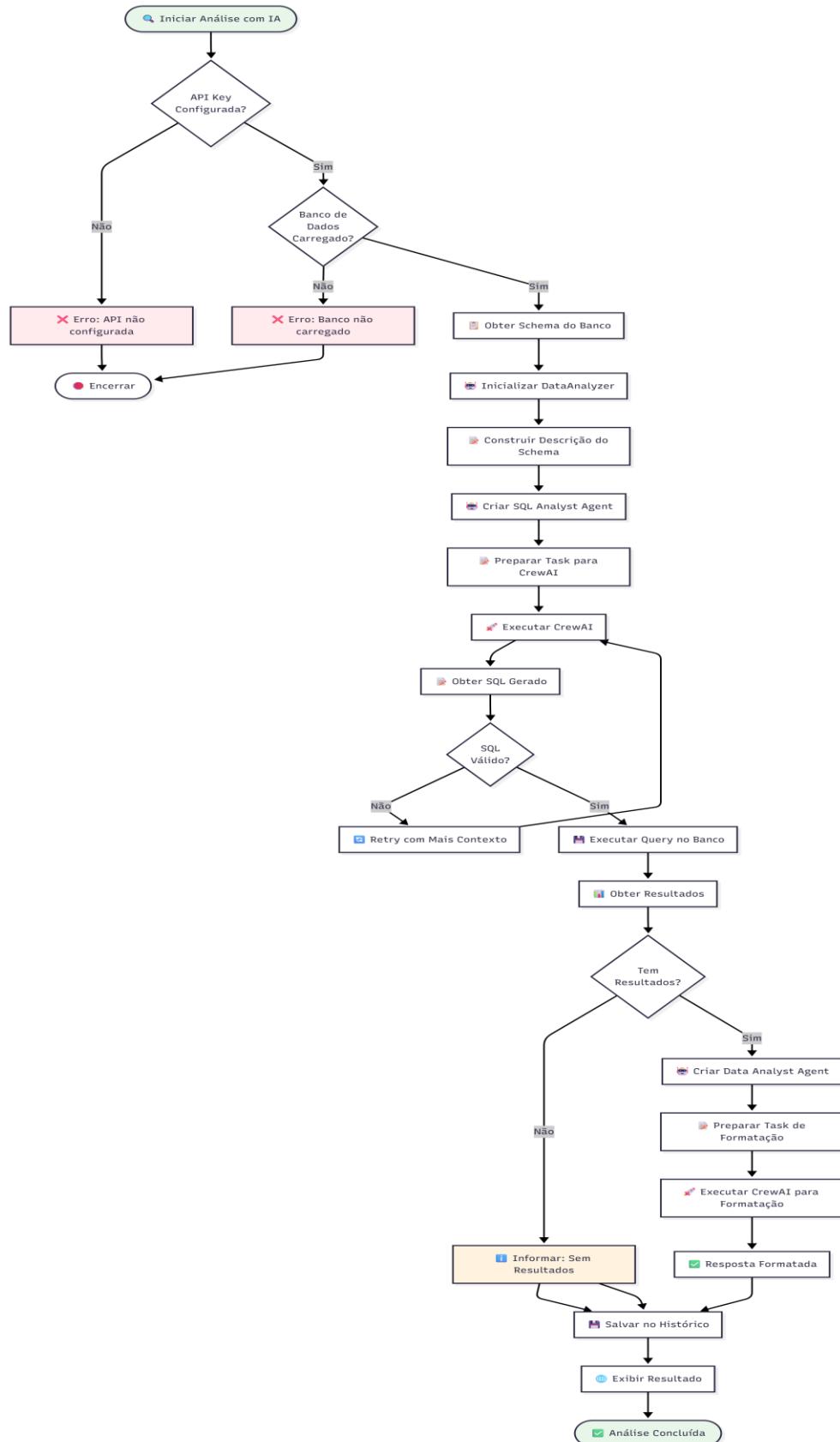


Diagrama 4 – Fluxograma da análise com Inteligência artificial.

5. AUDITORIA AUTOMÁTICA DE NOTAS FISCAIS

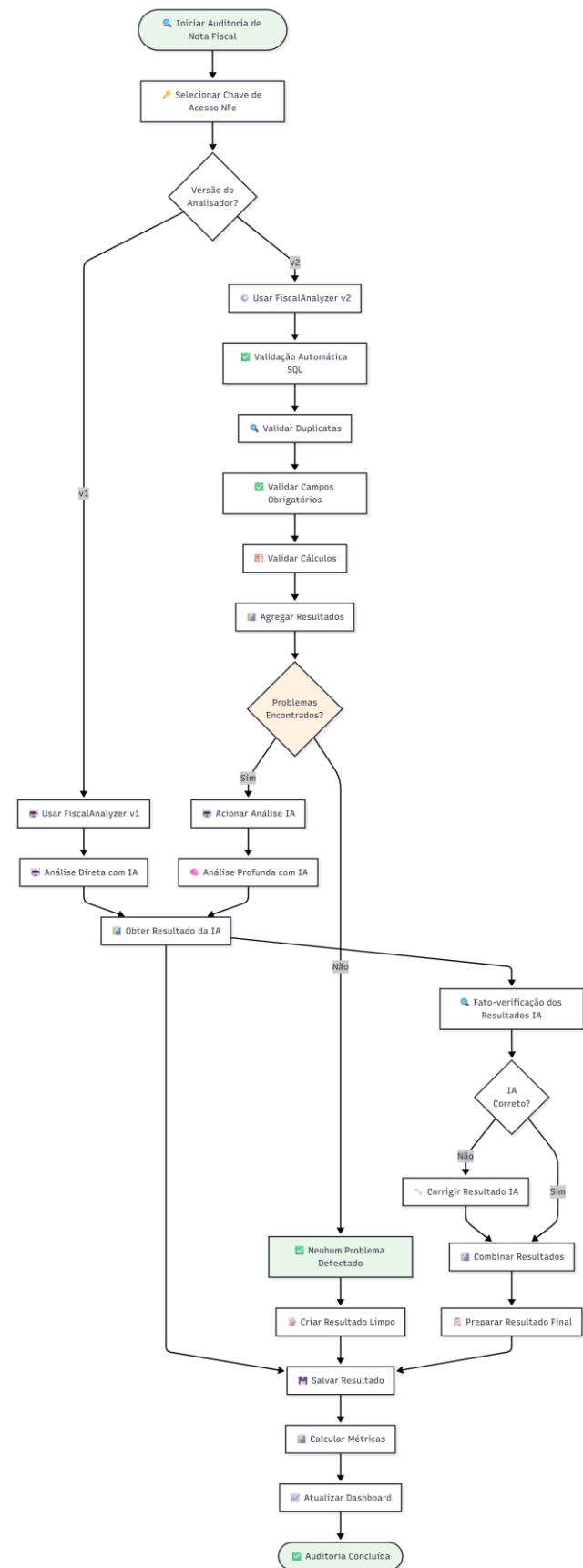


Diagrama 5 – Fluxograma de auditoria automática