

FISCALIA

Sistema de Processamento Inteligente de Notas Fiscais Eletrônicas

Relatório Técnico Final

Versão 1.0
Outubro 2025

1. Resumo Executivo

O Fiscalia é um sistema inovador de processamento automatizado de Notas Fiscais Eletrônicas portuguesas (NFe PT), desenvolvido com tecnologias de Inteligência Artificial e arquitetura multi-agente. O sistema foi concebido para automatizar a extração, validação e preparação de dados fiscais para integração em sistemas ERP.

1.1 Objetivos Principais

- Automatizar o processamento de arquivos XML de NFe
- Validar dados fiscais e comerciais automaticamente
- Preparar dados estruturados para sistemas ERP
- Fornecer insights e análises através de agentes AI
- Disponibilizar interface web intuitiva e acessível

1.2 Resultados Alcançados

O projeto foi concluído com sucesso, alcançando todos os objetivos estabelecidos:

- **Sistema 100% funcional** em ambiente de produção
- **Deploy realizado com sucesso** no Railway Cloud Platform
- **Interface web operacional** em <https://fiscalia.up.railway.app/>
- **3 agentes AI especializados** a funcionar corretamente
- **Base de dados SQLite** com persistência de dados

2. Arquitetura do Sistema

2.1 Visão Geral

O Fiscalia utiliza uma arquitetura modular baseada em microserviços, com separação clara entre camadas de apresentação, lógica de negócio e persistência de dados. O sistema está estruturado em quatro camadas principais:

Camada	Descrição
Apresentação	Interface web Streamlit com páginas multi-funcionalidade
Agentes AI	Sistema CrewAI com 3 agentes especializados
Lógica de Negócio	Processamento XML, validação, transformação de dados
Persistência	SQLite com SQLAlchemy ORM, estrutura normalizada

2.2 Componentes Principais

2.2.1 Interface Streamlit

Interface web moderna e responsiva com as seguintes funcionalidades:

- **Dashboard principal** com estatísticas em tempo real
- **Página de upload** com validação e processamento automático
- **Visualização de dados** com tabelas interativas
- **Exportação** para CSV e Excel
- **Sistema de análise AI** com insights detalhados

2.2.2 Sistema Multi-Agente CrewAI

O sistema utiliza três agentes especializados que trabalham colaborativamente:

Agente	Responsabilidade	Modelo LLM
Extrator	Extração e estruturação de dados XML	Llama 3.3 70B (Groq)
Validador	Validação de conformidade fiscal e comercial	GPT-4o Mini (OpenAI)
Analizador	Geração de insights e recomendações	Llama 3.3 70B (Groq)

2.2.3 Processamento de Dados

O módulo de processamento implementa:

- Parsing seguro de XML com lxml e defusedxml
- Validação de estrutura NFe PT
- Extração de campos obrigatórios e opcionais
- Transformação para formato ERP
- Registo de histórico de processamento

3. Tecnologias Utilizadas

3.1 Stack Tecnológico

Categoria	Tecnologia	Versão
Linguagem	Python	3.11+
Framework Web	Streamlit	1.46.1
AI Framework	CrewAI	0.140.0
LLM Orchestration	LangChain	0.3.12
Base de Dados	SQLite + SQLAlchemy	2.0.35
XML Processing	lxml + xmldict	5.3.0 + 0.14.2
Data Processing	Pandas + NumPy	2.2.3 + 2.1.3
Containerização	Docker	Latest
Cloud Platform	Railway	Production

3.2 Modelos LLM

O sistema utiliza dois provedores de LLM para otimizar desempenho e custo:

- **Groq Cloud - Llama 3.3 70B:** Processamento rápido e gratuito, ideal para extração e análise
- **OpenAI - GPT-4o Mini:** Validações complexas e reasoning avançado

4. Funcionalidades Implementadas

4.1 Processamento de NFe

- Upload de arquivos XML individuais ou em lote
- Validação de estrutura XML e conformidade NFe PT
- Extração automática de dados fiscais e comerciais
- Detecção e tratamento de erros
- Histórico completo de processamento

4.2 Análise Inteligente

- Validação automática de conformidade fiscal
- Verificação de consistência de dados
- Identificação de anomalias e alertas
- Geração de insights e recomendações
- Análise de padrões e tendências

4.3 Gestão de Dados

- Duas tabelas principais: docs_para_erp e registro_resultados
- Visualização interativa com filtros e ordenação
- Exportação para CSV e Excel (XLSX)
- Pesquisa e filtragem avançada
- Dashboard com estatísticas em tempo real

4.4 Interface do Utilizador

- Design responsivo e moderno
- Navegação intuitiva multi-página
- Feedback visual durante processamento
- Mensagens de erro e sucesso claras
- Acessível via browser sem instalação

5. Infraestrutura e Deployment

5.1 Ambiente de Desenvolvimento

- **Sistema Operativo:** Windows 11
- **IDE:** Visual Studio Code
- **Controlo de Versão:** Git + GitHub
- **Repositório:** github.com/josefeneto/fiscalia

5.2 Containerização Docker

O sistema foi containerizado usando Docker para garantir portabilidade e consistência entre ambientes. O Dockerfile implementa:

- Imagem base Python 3.11-slim otimizada
- Multi-stage build para reduzir tamanho
- Instalação de dependências do sistema
- Cache de layers para builds rápidos
- Script de inicialização start.sh

5.3 Deploy no Railway

O deployment foi realizado com sucesso na plataforma Railway Cloud:

- **URL de Produção:** <https://fiscalia.up.railway.app/>
- Deploy automático via GitHub push
- Variáveis de ambiente configuradas
- Health checks ativos
- Volumes persistentes para dados
- Logging e monitorização ativos

5.4 Configuração de Ambiente

Variáveis de ambiente configuradas no Railway:

- GROQ_API_KEY - API key do Groq Cloud
- OPENAI_API_KEY - API key da OpenAI
- PORT - Porta do servidor (auto-configurada)
- LOG_LEVEL - Nível de logging

6. Modelo de Dados

6.1 Tabela docs_para_erp

Tabela principal contendo os documentos processados para integração ERP:

Campo	Tipo	Descrição
id	Integer	Chave primária auto-incrementada
numero_documento	String(50)	Número único da NFe
data_emissao	Date	Data de emissão do documento
fornecedor_nome	String(200)	Nome do fornecedor
fornecedor_nif	String(20)	NIF do fornecedor
valor_total	Decimal	Valor total do documento
valor_iva	Decimal	Valor do IVA
created_at	DateTime	Data de registo no sistema

6.2 Tabela registo_resultados

Tabela de auditoria contendo o histórico completo de processamento:

Campo	Tipo	Descrição
id	Integer	Chave primária auto-incrementada
nome_arquivo	String(255)	Nome do arquivo XML processado
status	String(20)	Status do processamento (sucesso/erro)
mensagem	Text	Mensagem detalhada do resultado
data_processamento	DateTime	Data e hora do processamento
tempo_processamento	Float	Tempo de processamento em segundos
insights_ai	Text	Análises e recomendações dos agentes AI

7. Testes e Validação

7.1 Ambiente de Testes

O sistema foi testado extensivamente em dois ambientes:

- **Desenvolvimento Local:** Windows 11 com Python 3.11
- **Produção:** Railway Cloud Platform com Docker

7.2 Casos de Teste

Foram realizados testes abrangentes incluindo:

- Processamento de 5 arquivos XML reais de NFe PT
- Validação de estrutura e conformidade fiscal
- Testes de upload individual e em lote
- Verificação de persistência de dados
- Testes de exportação CSV e XLSX
- Validação de análise AI e insights
- Testes de interface e usabilidade

7.3 Resultados

Todos os testes foram concluídos com sucesso:

- 100% de sucesso no processamento de arquivos válidos
- Detecção correta de arquivos inválidos ou corrompidos
- Extração precisa de todos os campos obrigatórios
- Validação fiscal funcionando corretamente
- Agentes AI fornecendo insights relevantes
- Performance adequada mesmo com múltiplos arquivos

8. Segurança e Conformidade

8.1 Medidas de Segurança

- Parsing seguro de XML com defusedxml
- Validação de input para prevenir injeção
- Gestão segura de API keys via variáveis de ambiente
- Isolamento de dados por sessão
- Logs de auditoria de todas as operações

8.2 Conformidade Fiscal

- Suporte completo ao formato NFe PT
- Validação de campos obrigatórios conforme legislação
- Verificação de cálculos de IVA
- Registo de histórico para auditoria

8.3 Proteção de Dados

- Dados armazenados localmente em SQLite
- Sem partilha de dados sensíveis com terceiros
- API keys não expostas no código
- Conformidade com RGPD

9. Estrutura de Pastas e Arquivos

Estrutura completa do projeto organizada de forma modular:

```
fiscalia/
├── streamlit_app/
│   ├── app.py                # Aplicação principal
│   └── pages/                # Páginas multi-funcionalidade
│       ├── 1_📁_Upload.py    # Página de upload
│       ├── 2_📊_Visualizar.py # Visualização de dados
│       └── 3_🤖_Análise_AI.py # Análise com AI
│   └── components/          # Componentes reutilizáveis
├── src/                    # Código fonte principal
│   ├── __init__.py
│   ├── agents/             # Agentes CrewAI
│   │   ├── __init__.py
│   │   ├── extrator_agent.py # Agente extrator
│   │   ├── validador_agent.py # Agente validador
│   │   └── analisador_agent.py # Agente analisador
│   ├── crews/              # Crews e tarefas
│   │   ├── __init__.py
│   │   ├── crew.py         # Configuração da crew
│   │   └── tasks.py        # Definição de tarefas
│   ├── models/             # Modelos de dados
│   │   ├── __init__.py
│   │   └── database_models.py # Modelos SQLAlchemy
│   ├── services/           # Serviços de negócio
│   │   ├── __init__.py
│   │   ├── xml_processor.py # Processamento XML
│   │   └── validator.py     # Validações
│   ├── database/           # Camada de dados
│   │   ├── __init__.py
│   │   └── db_manager.py   # Gestor de BD
│   ├── utils/              # Utilidades
│   │   ├── __init__.py
│   │   ├── logger.py       # Sistema de logging
│   │   └── config.py        # Configurações
├── arquivos/              # Ficheiros processados
│   ├── entrados/           # XMLs recebidos
│   ├── processados/        # XMLs processados com sucesso
│   └── rejeitados/         # XMLs com erros
├── data/                  # Base de dados e temporários
│   ├── fiscalia.db         # Base de dados SQLite
│   └── temp/               # Ficheiros temporários
└── tests/                 # Testes automatizados
    ├── __init__.py
    ├── test_xml_processor.py
    └── test_agents.py
```

```
|
|— .env                                # Variáveis de ambiente (não
versionado)
|— .env.example                       # Template de variáveis
|— .gitignore                         # Arquivos ignorados pelo Git
|— Dockerfile                         # Configuração Docker
|— .dockerignore                     # Arquivos ignorados pelo Docker
|— railway.json                      # Configuração Railway
|— start.sh                          # Script de inicialização
|— requirements.txt                  # Dependências Python
|— README.md                         # Documentação do projeto
|— LICENSE                           # Licença do projeto
```

10. Conclusões e Próximos Passos

10.1 Conquistas do Projeto

O projeto Fiscalia alcançou todos os objetivos estabelecidos, entregando um sistema completo e funcional de processamento inteligente de Notas Fiscais Eletrônicas. Os principais resultados incluem:

- Sistema totalmente operacional em produção
- Arquitetura modular e escalável
- Integração bem-sucedida de múltiplas tecnologias AI
- Interface intuitiva e profissional
- Deploy em cloud com CI/CD automático

10.2 Melhorias Futuras

Oportunidades identificadas para evolução do sistema:

Curto Prazo (1-3 meses)

- Dashboard analytics avançado com gráficos
- Exportação de relatórios em PDF
- Sistema de notificações por email
- Autenticação de utilizadores

Médio Prazo (3-6 meses)

- API REST para integração externa
- Suporte multi-empresa
- Integração com ERPs (SAP, PHC, Primavera)
- Migração para PostgreSQL
- Sistema de workflows aprovações

Longo Prazo (6-12 meses)

- OCR para digitalização de faturas em papel
- Machine Learning para detecção de anomalias
- Previsão de despesas com AI
- Aplicação mobile (iOS e Android)
- Chatbot para consultas em linguagem natural

10.3 Considerações Finais

O Fiscalia representa uma solução moderna e eficiente para o processamento automatizado de documentos fiscais, combinando as mais recentes tecnologias de Inteligência Artificial com arquitetura cloud-native. O sistema demonstrou excelente performance e confiabilidade, estando pronto para uso em ambiente de produção.

A arquitetura modular permite fácil manutenção e extensão de funcionalidades, garantindo a longevidade do projeto. O uso de tecnologias open-source e cloud platforms minimiza custos operacionais enquanto maximiza a escalabilidade.

O sistema está completamente funcional e acessível
em: <https://fiscalia.up.railway.app/>

- Fim do Relatório -