UNIVERSIDADE
LUSÓFONA

SLM Jurídico

Trabalho Final de Curso

Relatório Intercalar 2º Semestre

Aluno:

Gonçalo Filipe Duarte Lourenço Ribeiro

Orientador:

Daniel Silveira

Trabalho Final de Curso | LEI | 2025/2026

Direitos de cópia

SLM Jurídico, Copyright de Gonçalo Filipe Duarte Lourenço Ribeiro, Universidade Lusófona.
A Escola de Comunicação, Arquitectura, Artes e Tecnologias da Informação (ECATI) e a Universidade Lusófona (UL) têm o direito, perpétuo e sem limites geográficos, de arquivar e publicar esta dissertação através de exemplares impressos reproduzidos em papel ou de forma digital, ou por qualquer outro meio conhecido ou que venha a ser inventado, e de a divulgar através de repositórios científicos e de admitir a sua cópia e distribuição com objectivos educacionais ou de investigação, não comerciais, desde que seja dado crédito ao autor e editor.
Este documento foi gerado com o processador (pdf/Xe/Lua)LaTeX e o modelo ULThesis (v1.0.0) [Mat24].

Resumo

A integração de Inteligência Artificial (IA) no setor jurídico promete ganhos significativos de eficiência, contudo, a utilização de soluções baseadas na nuvem colide frequentemente com os rigorosos requisitos de confidencialidade, sigilo profissional e proteção de dados (RGPD). Este projeto, desenvolvido no âmbito do Trabalho Final de Curso, propõe o desenvolvimento de uma solução de ”SLM Jurídico”, focada na automatização da defesa de contraordenações rodoviárias através de uma infraestrutura inteiramente local (on-premise).

A solução adota uma Arquitetura Orientada a Eventos (Event-Driven Architecture) baseada em microserviços contentorizados, garantindo escalabilidade, tolerância a falhas e a soberania absoluta dos dados. A pipeline tecnológica utiliza o Apache Kafka como motor de orquestração assíncrona entre serviços desenvolvidos em diferentes linguagens (Java, Python, React). O processamento documental integra OCR via Tesseract com uma arquitetura Retrieval-Augmented Generation (RAG), onde o texto é indexado numa base vetorial ChromaDB com recurso ao modelo Nomic-embed e ao Llama 3.2. A extração semântica e a estruturação jurídica são asseguradas pelo modelo Mistral-Nemo. Após validação humana (abordagem human-in-the-loop) através de uma interface reativa, o sistema compila a peça processual final utilizando algoritmos de injeção em templates LaTeX. O objetivo principal é demonstrar a viabilidade de um ecossistema de IA descentralizado, seguro e altamente especializado, capaz de eliminar tarefas manuais repetitivas e acelerar a resposta processual.

Palavras-chave: Inteligência Artificial, Arquitetura Orientada a Eventos, Microserviços, RAG Local, Privacidade de Dados, SLM.

DEISI | lusófona informática

Figure 1: DEISI Logo

Abstract

The integration of Artificial Intelligence (AI) into the legal sector promises significant efficiency gains; however, the use of cloud-based solutions often conflicts with strict requirements for confidentiality, professional secrecy, and data protection (GDPR). This project, developed within the scope of the Final Course Work, proposes the development of a ”Legal SLM” solution, focused on automating the defense of traffic infractions through an entirely on-premise infrastructure.

The solution adopts an Event-Driven Architecture based on containerized microservices, ensuring scalability, fault tolerance, and absolute data sovereignty. The technological pipeline utilizes Apache Kafka as the asynchronous orchestration engine across multi-language services (Java, Python, React). Document processing integrates OCR via Tesseract with a Retrieval-Augmented Generation (RAG) architecture, where text is indexed in a ChromaDB vector database using the Nomic-embed model and Llama 3.2. Semantic extraction and legal reasoning are handled by the Mistral-Nemo model. Following human validation (a human-in-the-loop approach) via a reactive interface, the system generates the final legal document using injection algorithms in LaTeX templates. The main objective is to demonstrate the viability of a decentralized, secure, and highly specialized AI ecosystem capable of eliminating repetitive manual tasks and accelerating procedural response times.

Keywords: Artificial Intelligence, Event-Driven Architecture, Microservices, Local RAG, Data Privacy, SLM.

Índice

Resumo **2**

Abstract **3**

Índice **4**

Lista de Figuras **6**

Lista de Tabelas **7**

1 Identificação do Problema **8**

2 Benchmarking **9**
2.1 Análise do Processo Tradicional (Manual) 9
2.2 Soluções de IA em Nuvem (SaaS - e.g., ChatGPT, Claude) 9
2.3 Soluções ”LegalTech” de Consumo (e.g., DoNotPay) 9
2.4 Estado da Arte: Microserviços e Arquitetura Orientada a Eventos 10
2.4.1 Arquitetura Orientada a Eventos (EDA) com Apache Kafka 10
2.4.2 Ecossistema de IA Local: SLMs e Processamento Especializado . . 10
2.4.3 Ingestão Multimodal e Geração de Documentos (Tesseract e LaTeX) 11
2.5 Tabela Comparativa 11

3 Viabilidade e Pertinência **12**
3.1 Pertinência e Relevância Operacional 12
3.2 Análise de Viabilidade Económica (Estimativa) 12
3.3 Viabilidade Técnica 13
3.4 Validação por Terceiros 14

4 Solução Proposta **15**
4.1 Fluxo de Execução 15
4.2 Diagrama de Arquitetura da Solução 17

5 Engenharia de Sistema **18**
5.1 Levantamento de Requisitos 18
5.1.1 Requisitos Funcionais (RF) 18
5.1.2 Requisitos Não-Funcionais (RNF) 21
5.2 Arquitetura Tecnológica 21
5.2.1 Camada de Interface de Utilizador (UI/Frontend) 21
5.2.2 Camada de Comunicação e Orquestração Assíncrona 22
5.2.3 Camada de Processamento de Imagem (OCR) 22
5.2.4 Camada de Indexação e Embeddings (Pré-RAG) 22
5.2.5 Camada de Armazenamento Vetorial 22
5.2.6 Camada de Inferência Lógica e Extração (SLM Principal) 23
5.2.7 Camada de Geração Documental 23
5.2.8 Infraestrutura Física e Virtual 23

6 Método e Planeamento **24**
6.1 Metodologia de Desenvolvimento 24
6.2 Fases do Projeto 24
6.3 Métricas de Avaliação 25

7 Testes e Validação **27**
7.1 Matriz de Rastreabilidade Testes-Requisitos 27
7.2 Plano de Testes 28
7.2.1 Testes da Fase de Ingestão e Processamento Inicial 28
7.2.2 Testes da Fase de Data Preparation (OCR e Chunking) 28
7.2.3 Testes de Vetorização e Armazenamento (Pré-RAG) 29
7.2.4 Testes de Extração e Raciocínio Jurídico (Mistral-Nemo) 29
7.2.5 Testes da Validação Humana e UI (Human-in-the-Loop) 30
7.2.6 Testes de Geração Documental (LaTeX) 31
7.2.7 Testes de Validação de Carga, Performance e Segurança 31

8 Calendário **33**
8.1 Descrição das Fases 33

Bibliografia **34**

Lista de Siglas **38**

Anexo: Título do Documento **40**

Lista de Figuras

1 DEISI Logo 2

4.1 Diagrama da Arquitetura 17

8.1 Cronograma de execução do projeto 33

Lista de Tabelas

2.1 Comparação entre processos manuais, IA pública, LegalTech e a solução proposta 11

7.1 Matriz de Rastreabilidade de Requisitos e Testes 27

1 - Identificação do Problema

No contexto jurídico atual, a gestão eficiente de processos de contraordenação rodoviária (mais conhecido no dominio como ”multas”) representa um desafio significativo para as casas de advogados. O processo tradicional de contestação destas multas é caracterizado por tarefas manuais, repetitivas e burocráticas, que consomem uma fatia desproporcional do tempo útil dos profissionais forenses [Dia22; Sus17].

O problema central abordado neste Trabalho Final de Curso (TFC) é a ineficiência operacional decorrente da análise manual de documentos digitalizados (autos de contraordenação) e a subsequente redação de minutas de defesa. Atualmente, este fluxo de trabalho obriga os advogados a extrair manualmente dados de imagens ou PDFs e a preencher modelos de texto pré-existentes. Este método não só é propenso a erro humano, como também desvia recursos humanos qualificados de tarefas de maior valor jurídico e estratégico.

Adicionalmente, existe uma barreira crítica na adoção de soluções de Inteligência Artificial (IA) comerciais e baseadas na cloud (como o ChatGPT ou Claude) por parte de sociedades de advogados: a confidencialidade. Devido à natureza sensível dos dados dos clientes e às obrigações de sigilo profissional e RGPD [Eur16] (Regulamento Geral sobre a Proteção de Dados), o envio de documentos legais para servidores de terceiros é inadmissível [Mac24; Ibr+25].

O presente projeto surge, portanto, da necessidade real de um parceiro (escritório de advogados) em automatizar a produção de peças processuais sem comprometer a privacidade dos dados. O problema resume-se à inexistência de uma ferramenta que combine a automação de leitura e escrita (OCR e LLM) com a garantia absoluta de processamento local (offline ou on-premise).

2 - Benchmarking

Este capítulo apresenta uma análise comparativa entre a solução ”SLM Jurídico” proposta e as alternativas atualmente disponíveis, variando desde o processamento manual até soluções de Inteligência Artificial baseadas na nuvem (Cloud). A análise foca-se em critérios críticos para o contexto jurídico: confidencialidade dos dados, eficiência operacional, custo e precisão na análise documental.

2.1 Análise do Processo Tradicional (Manual)

Atualmente, o ”concorrente” direto da solução proposta é o método de trabalho humano. No escritório em questão, a gestão de contraordenações envolve a leitura manual de documentos digitalizados, a transcrição de dados para o sistema de gestão e a redação da defesa com base em modelos pré-existentes.

- Limitações: Este processo é linear e não escalável. O tempo médio por processo é elevado e suscetível a erro humano por fadiga, especialmente em tarefas repetitivas de verificação de dados (matrículas, datas, velocidades) [PwC24a; Bai25].
- Custo: O custo operacional é elevado, pois ocupa horas de advogados ou paralegais qualificados em tarefas administrativas [Sus17; Del24].

2.2 Soluções de IA em Nuvem (SaaS - e.g., ChatGPT, Claude)

O modelo tecnológico Software as a Service (SaaS) caracteriza-se pela disponibilização de software e poder computacional alojados na nuvem (cloud). Neste paradigma, toda a infraestrutura, manutenção e processamento de dados são geridos por um fornecedor externo, sendo a ferramenta acedida pelo utilizador via internet (geralmente num modelo de subscrição).

Nesta categoria enquadram-se os Grandes Modelos de Linguagem (LLMs) públicos e altamente capazes, como o GPT-4 (OpenAI) ou o Claude (Anthropic). Embora tecnicamente capazes de gerar minutas com elevada qualidade, estas soluções comerciais apresentam um obstáculo intransponível para o setor jurídico empresarial: a privacidade.

- Risco de Privacidade: Ao submeter autos de contraordenação e dados de clientes para estas plataformas, a informação sai da infraestrutura segura da empresa e é processada em servidores de terceiros (muitas vezes fora da União Europeia). Esta transferência viola as obrigações de sigilo profissional e o Regulamento Geral sobre a Proteção de Dados (RGPD)[Int24; Che23].
- Alucinação: Sem uma base de conhecimento acoplada (RAG), estes modelos tendem a produzir erros causados por diversos fatores, incluindo dados de treinamento insuficientes, suposições incorretas feitas pelo modelo ou vieses nos dados usados para treinar o modelo. [Uni23; Per+24].

2.3 Soluções ”LegalTech” de Consumo (e.g., DoNotPay)

Existem aplicações voltadas para o consumidor final, como o ”DoNotPay” (frequentemente apelidado de ”O Primeiro Advogado Robô”), que automatizam a contestação de multas [Goo23].

- Inadequação ao Contexto B2B: Estas ferramentas são desenhadas para o cidadão comum, não permitindo a integração com os fluxos de trabalho, formatação e base de conhecimento específica de um escritório de advogados português [CG24; SD23]. Não oferecem personalização nem controlo sobre a base jurídica utilizada.

2.4 Estado da Arte: Microserviços e Arquitetura Orientada a Eventos

A solução proposta afasta-se de modelos monolíticos tradicionais ou de orquestradores visuais genéricos de IA, situando-se na vanguarda do desenvolvimento de software seguro e distribuído, vulgarmente enquadrado no paradigma de Sovereign AI (IA Soberana)[SS25]. Para acomodar a complexidade de múltiplos processos pesados (OCR, vetorização e inferência de IA) sem comprometer a estabilidade do sistema, adota-se um modelo de microserviços contentorizados, alicerçado nos seguintes pilares tecnológicos.

2.4.1 Arquitetura Orientada a Eventos (EDA) com Apache Kafka

O estado da arte no design de sistemas distribuídos empresariais favorece a Arquitetura Orientada a Eventos (EDA) [Ric15] em detrimento de chamadas síncronas tradicionais (como APIs REST encadeadas). Numa arquitetura síncrona, se o módulo de IA demorar 30 segundos a responder, o Frontend e os serviços intermédios ficam bloqueados à espera, consumindo recursos e arriscando falhas de timeout.

Para resolver este problema, o projeto integra o Apache Kafka, um message broker distribuído de alto desempenho [KNR11]. O Kafka atua como o sistema nervoso central da aplicação:

- Desacoplamento (Loose Coupling): Os serviços (Produtores) publicam eventos (ex: ”Novo PDF para OCR”) num tópico do Kafka sem precisarem de saber que serviço os vai processar. Os serviços de processamento (Consumidores) leem essas mensagens ao seu próprio ritmo.
- Tolerância a Falhas e Resiliência: Se o contentor do modelo de linguagem (SLM) for reiniciado ou falhar devido a um pico de memória, a mensagem com o pedido de geração não se perde; fica retida no Kafka até que o contentor recupere e a processe, garantindo que o trabalho do advogado nunca é descartado.
- Escalabilidade: Caso o volume de multas aumente exponencialmente, é possível instanciar múltiplos contentores de OCR ou SLMs a consumir do mesmo tópico de Kafka simultaneamente, equilibrando a carga processual sem alterar uma única linha de código na arquitetura base.

2.4.2 Ecossistema de IA Local: SLMs e Processamento Especializado

Em vez de depender de um único modelo de linguagem gigante (que exigiria hardware proibitivo), a arquitetura segue uma abordagem análoga ao Mixture of Experts [Sha+17] (Mistura de Especialistas), onde diferentes modelos otimizados são invocados para tarefas específicas, minimizando o uso de VRAM e maximizando a velocidade.

- Embeddings (Nomic-embed): Para a implementação do RAG, selecionou-se o modelo nomic-embed-text [Mul+24] devido à sua alta performance em tarefas de recuperação semântica (Retrieval). Apesar de ser um modelo open-source leve, ele

apresenta resultados superiores a modelos proprietários de maior escala em benchmarks como o MTEB, garantindo que a representação vetorial no ChromaDB seja altamente precisa para buscas por similaridade.

- Chunking e Pré-Processamento: O processamento inicial do texto (limpeza e divisão lógica em chunks) é delegado a um modelo ultraleve [Met24] (Llama 3.2 - 1B parâmetros). Isto permite um processamento rápido de rotinas locais sem ocupar a GPU desnecessariamente.

- Raciocínio Jurídico: O processamento lógico e a extração de entidades são realizados pelo Mistral-Nemo [Mis24]. Este modelo foi selecionado pela sua capacidade de raciocínio estruturado, essencial para converter factos jurídicos não estruturados num esquema JSON estrito. Além disso, a sua ampla janela de contexto permite processar documentos legais extensos sem perda de coerência na análise dos factos.

2.4.3 Ingestão Multimodal e Geração de Documentos (Tesseract e LaTeX)

O ciclo de vida do documento neste sistema inicia-se na visão computacional e termina na renderização programática de documentos.

- Extração Primária (Tesseract OCR): Para a conversão de imagens/PDFs rasterizados em texto processável, utiliza-se o motor Tesseract [Smi07], suportado em Python. É uma solução madura e amplamente testada, que extrai o texto bruto para que o indexador aplique posteriormente a lógica de segmentação, libertando a rede neural de tarefas dispendiosas de processamento de imagem puro.

- Geração Programática de Documentos (LaTeX): A inovação final desta arquitetura reside na utilização do LaTeX para a criação do ficheiro PDF de saída. No meio jurídico, a formatação exata das peças processuais é um requisito crítico. Ao contrário da geração de PDFs via bibliotecas padrão ou conversão de HTML, o LaTeX garante um layout determinístico e composição tipográfica de nível profissional [Lam94]. Através de algoritmos Python, o sistema injeta os dados validados pelo advogado diretamente num template genérico (.tex), assegurando que o documento final respeita as normas de formatação jurídica de forma imutável, eliminando os artefactos comuns em editores WYSIWYG.

2.5 Tabela Comparativa

Critério	Processo Manual (Atual)	IA Pública (ChatGPT/Cloud)	LegalTech Genérica (DoNotPay)	SLM Jurídico (Proposto)
Privacidade de Dados	Alta (Local)	Baixa (Envio para terceiros)	Baixa/Média	Alta (100% Local)
Tempo de Execução	Lento (Minutos/Horas)	Rápido (Segundos)	Rápido	Rápido (Segundos)
Custo Recorrente	Alto (Salários)	Médio (Subscrição API)	Baixo (Consumidor)	Baixo (Custo Energético)
Personalização	Total	Limitada ao Prompt	Nenhuma	Total (RAG Customizado)
Capacidade OCR	Depende do Humano	Variável	Limitada	Alta (Tesseract)

Table 2.1: Comparação entre processos manuais, IA pública, LegalTech e a solução proposta

Conclui-se que, embora existam ferramentas poderosas no mercado, nenhuma preenche simultaneamente os requisitos de automatização e confidencialidade exigidos pelo parceiro, justificando o desenvolvimento da presente solução.

3 - Viabilidade e Pertinência

Este capítulo demonstra a viabilidade do projeto ”SLM Jurídico” sob as perspetivas técnica, económica e operacional, evidenciando a sua relevância para a modernização dos processos jurídicos da entidade parceira.

3.1 Pertinência e Relevância Operacional

A pertinência deste projeto justifica-se pela necessidade urgente de libertar recursos humanos qualificados de tarefas de baixo valor acrescentado [Tho24a]. Esta urgência é corroborada por estudos recentes que indicam que 77% dos profissionais jurídicos já recorrem a ferramentas de IA generativa para revisão e redação documental [Tho24b], sinalizando uma mudança irreversível no setor.

Num escritório de advocacia ou departamento jurídico de grande dimensão, a triagem de contraordenações é uma tarefa repetitiva que, embora exija precisão, não requer raciocínio jurídico complexo em 90% dos casos. A automatização destas funções permite uma redução estimada entre 20% a 25% nas horas despendidas em processos manuais [PwC24b], libertando tempo faturável crítico.

A solução proposta tem impacto direto em três vertentes:

1. Redução de Erro Humano: A automatização via OCR (Tesseract) elimina erros de transcrição (e.g., errar um dígito na matrícula ou no valor da coima) [Pro25; Bai25].
2. Aceleração de Prazos: A capacidade de processar documentos instantaneamente garante que nenhuma multa prescreve por atraso na resposta [Cli25].
3. Valorização Profissional: Permite que os advogados foquem o seu tempo na revisão das minutas e em casos complexos [Spe25], em vez de preencherem formulários manualmente.

3.2 Análise de Viabilidade Económica (Estimativa)

A viabilidade económica é avaliada comparando o custo do processo atual (manual) com o custo da solução automatizada. Para efeitos deste relatório, considera-se um cenário base estimado para uma entidade de grande dimensão.

Cenário Base (Atual - Manual)

- Volume mensal estimado: 500 contraordenações.
- Tempo médio de processamento manual: 20 minutos por processo (leitura, extração de dados, redação).
- Total de horas gastas: $500 \times 20 = 10\,000$ minutos $\approx$ 166 horas/mês.
- Análise: Este volume equivale, na prática, à dedicação exclusiva de um colaborador a tempo inteiro (FTE) apenas para esta tarefa. Considerando um custo/hora base de 15€ (valor conservador para técnico/paralegal) [Hay24], o custo mensal é de aproximadamente 2.490€.

Cenário com SLM Jurídico (Proposto)

- Tempo humano estimado: 2 minutos por processo (apenas revisão e validação da minuta gerada).
- Total de horas gastas: $500 \times 2 = 1\,000$ minutos $\approx$ 16 horas/mês.
- Custo Operacional Novo: 16 horas $\times$ 15€ = 240€/mês.

Conclusão Econométrica: A implementação do sistema representa uma poupança mensal estimada superior a 2.000€ (redução de custos na ordem dos 90%) [McK23]. Mesmo considerando o investimento inicial em hardware (e.g., servidor com GPU dedicada, estimado em 3.000€), o Retorno do Investimento (ROI) é atingido em menos de 2 meses. Adicionalmente, como a solução utiliza software open-source (Ollama, Langflow, ChromaDB, Linux), não existem custos recorrentes de licenciamento de software, garantindo a sustentabilidade financeira do projeto a longo prazo [Red24].

3.3 Viabilidade Técnica

A solução é tecnicamente viável, exigindo apenas a disponibilidade de um computador equipado com uma GPU comercial (até 16GB de VRAM). Adicionalmente, o seu nível de risco de implementação é mitigado pela adoção de tecnologias open-source altamente maduras, combinadas com as mais recentes otimizações na execução de Inteligência Artificial local. A viabilidade sustenta-se em três eixos principais:

- Viabilidade Computacional e de Hardware (SLMs): A execução de IA generativa on-premise é frequentemente limitada pelos recursos de hardware. Contudo, a escolha deliberada de Small Language Models (SLMs) garante a viabilidade do projeto. O modelo Llama 3.2 (1B) [Met24] e o Nomic-embed [Mul+24] são extremamente leves, enquanto o Mistral-Nemo [Mis24], sendo o modelo mais pesado da pipeline, é altamente otimizado. Com o uso de técnicas de quantização (e.g., 8-bit ou 4-bit) [Det+22a], todo este ecossistema de modelos pode ser carregado e executado numa única GPU comercial (<16GB VRAM), eliminando a necessidade de clusters de servidores dispendiosos.
- Resiliência Arquitetural (Kafka e Docker): A complexidade de orquestrar múltiplos serviços (UI, OCR, Indexador, SLM, Gerador PDF) em linguagens diferentes (Java, Python, JS) é resolvida pela adoção do Apache Kafka [KNR11] e do Docker [Mer14]. A viabilidade técnica desta integração é alta porque o Kafka previne o bloqueio do sistema (timeouts) ao gerir filas de mensagens assíncronas. Adicionalmente, a contentorização via Docker garante que o código desenvolvido no ambiente de teste funcionará de forma idêntica no servidor Linux de produção do parceiro, eliminando problemas de dependências.
- Maturidade das Ferramentas de Ingestão e Geração: Os componentes periféricos à IA não dependem de tecnologias experimentais. A extração de texto confia no Tesseract [Smi07], um dos motores de OCR mais maduros e testados do mercado. Por sua vez, a geração documental baseia-se no LaTeX [Lam94], um sistema tipográfico com décadas de estabilidade comprovada. Esta combinação assegura que, uma vez extraídos os dados corretos pela IA, a geração da defesa jurídica final é 100% determinística e à prova de falhas de formatação.

3.4 Validação por Terceiros

Para validar a pertinência e a usabilidade da solução, será realizada uma avaliação contínua com os utilizadores finais (advogados da empresa). Será aplicado um inquérito de satisfação (anexo ao relatório final) focando em critérios [Bro96; 18] como:

- Qualidade da minuta gerada (precisão jurídica).
- Facilidade de uso da interface.
- Tempo poupado percebido pelo utilizador.

4 - Solução Proposta

A solução proposta, designada ”SLM Jurídico”, consiste numa plataforma avançada e altamente distribuída, baseada numa arquitetura de microserviços contentorizados (Docker) [Mer14]. O sistema opera de forma inteiramente local (on-premise) na infraestrutura do cliente, assegurando a confidencialidade absoluta dos dados processados e cumprindo rigorosamente os regulamentos de proteção de dados.

O núcleo tecnológico afasta-se de soluções monolíticas, adotando uma Arquitetura Orientada a Eventos (EDA) [Ric15]. Os múltiplos contentores interagem através de uma rede virtual privada, partilhando ficheiros físicos através de um volume comum (Shared Folder) e orquestrando processos complexos através de um Message Broker (Apache Kafka) [KNR11] e chamadas API diretas.

4.1 Fluxo de Execução

O fluxo de execução, que se encontra ilustrado na secção 4.2, desde a submissão da multa até à geração da defesa, decorre nas seguintes fases:

1. **Interface de Ingestão (Frontend)**: O ciclo inicia-se na Interface de Utilizador (UI), um contentor próprio desenvolvido com as tecnologias React [BP20], Next.js [Sha20] e JSS [JSS24]. O utilizador seleciona o tipo de contraordenação e efetua o upload do documento em formato PDF para a plataforma. A UI guarda imediatamente este ficheiro no Shared Folder, um diretório físico criado e configurado previamente durante o setup inicial do projeto, montado como um volume partilhado [Mer14] para permitir que vários contentores acedam aos mesmos ficheiros. De seguida, a interface emite um evento (uma mensagem JSON [Bra17]) para o sistema de comunicação (Kafka) [KNR11], notificando o início do processo.

2. **Orquestração Assíncrona (Broker)**: O evento inicial é recebido pelo contentor Broker, desenvolvido em Java. Este serviço atua como o gestor de tráfego central do sistema: processa o JSON recebido e orquestra o fluxo encaminhando uma nova mensagem via Kafka para o módulo de processamento ótico, contendo o nome do ficheiro a ser analisado.

3. **Processamento Ótico de Caracteres (OCR)**: O contentor de OCR (desenvolvido em Python e integrado com o motor Tesseract [Smi07]) escuta a mensagem do Broker. Ao identificar o nome do ficheiro, acede ao Shared Folder, converte o documento digitalizado em texto simples (plain text) e envia este resultado, através de uma chamada de API síncrona, para o módulo de indexação.

4. **Indexação e Vetorização (RAG)** [Lew+20]: Esta fase centraliza a preparação da base de conhecimento e envolve a coordenação de múltiplos microserviços:

    - Indexer: O contentor indexer desenvolvido em Python recebe o texto via API, divide-o em blocos semânticos (chunks) e aplica heurísticas para selecionar os excertos mais relevantes. Em seguida, publica estes dados num tópico específico do Kafka para serem consumidos pelo módulo seguinte.
    - Embedding e SLM: O contentor Embedding (Python): Escuta o Kafka para receber os chunks de texto. Ele interage com o contentor SLM (que aloja os modelos Llama 3.2:1B [Met24], Nomic-embed [Mul+24] e Mistral-Nemo

[Mis24]) para transformar os dados em representações vetoriais de alta precisão, introduzindo-as no ChromaDB [Chr24]. Uma vez concluído o armazenamento, este módulo envia uma mensagem de ”start retrieval” (com o nome da coleção que acabou de inserir) ao Broker via Kafka, sinalizando que o sistema pode avançar para a fase de inferência jurídica.

5. **Recuperação de Informação e Inferência**: Após a vetorização, o contentor RetrievalInfo (Python, Kafka), que se encontra à escuta deste tópico, é então acionado. Munido de um System Prompt [Whi+23] pré-configurado que define a estrutura JSON de output esperada, este serviço consulta a base vetorial ChromaDB. Utilizando a capacidade de raciocínio avançado do modelo Mistral-Nemo (alojado no contentor SLM), o sistema extrai os factos jurídicos relevantes para a defesa. O RetrievalInfo publica, por fim, a informação estruturada de volta no Kafka, pronta a ser consumida pela interface para validação.

6. **Validação Humana (Human-in-the-Loop)**: A UI, que se encontrava em estado de escuta passiva (listening) no Kafka, interceta os dados extraídos pela IA. O utilizador é automaticamente redirecionado para uma página de formulário dinâmico, onde os dados pré-preenchidos são apresentados para verificação. Esta abordagem human-in-the-loop [Mon21] garante o controlo de qualidade: o advogado pode validar ou corrigir facilmente qualquer erro antes de clicar em ”Confirmar”. Após a confirmação, a UI envia os dados finais validados de volta para o Kafka.

7. **Geração Documental Automática (PDFGenerator)**: O contentor PDFGen (Python, LaTeX) escuta as confirmações provenientes da interface. Munido dos dados validados, aciona um algoritmo interno que injeta as informações precisas num template LaTeX [Lam94] desenvolvido à medida. Este método assegura uma formatação tipográfica irrepreensível, essencial em peças jurídicas. O documento final é compilado e guardado no Shared Folder, enviando-se de seguida o nome do ficheiro gerado via Kafka.

8. **Disponibilização e Download**: Na etapa final, a UI capta o evento de conclusão emitido pelo gerador, notificando o utilizador e disponibilizando uma página com o botão de download. O utilizador transfere de forma segura o PDF final, concluindo um ciclo de processamento altamente complexo de forma fluida e transparente.

4.2 Diagrama de Arquitetura da Solução

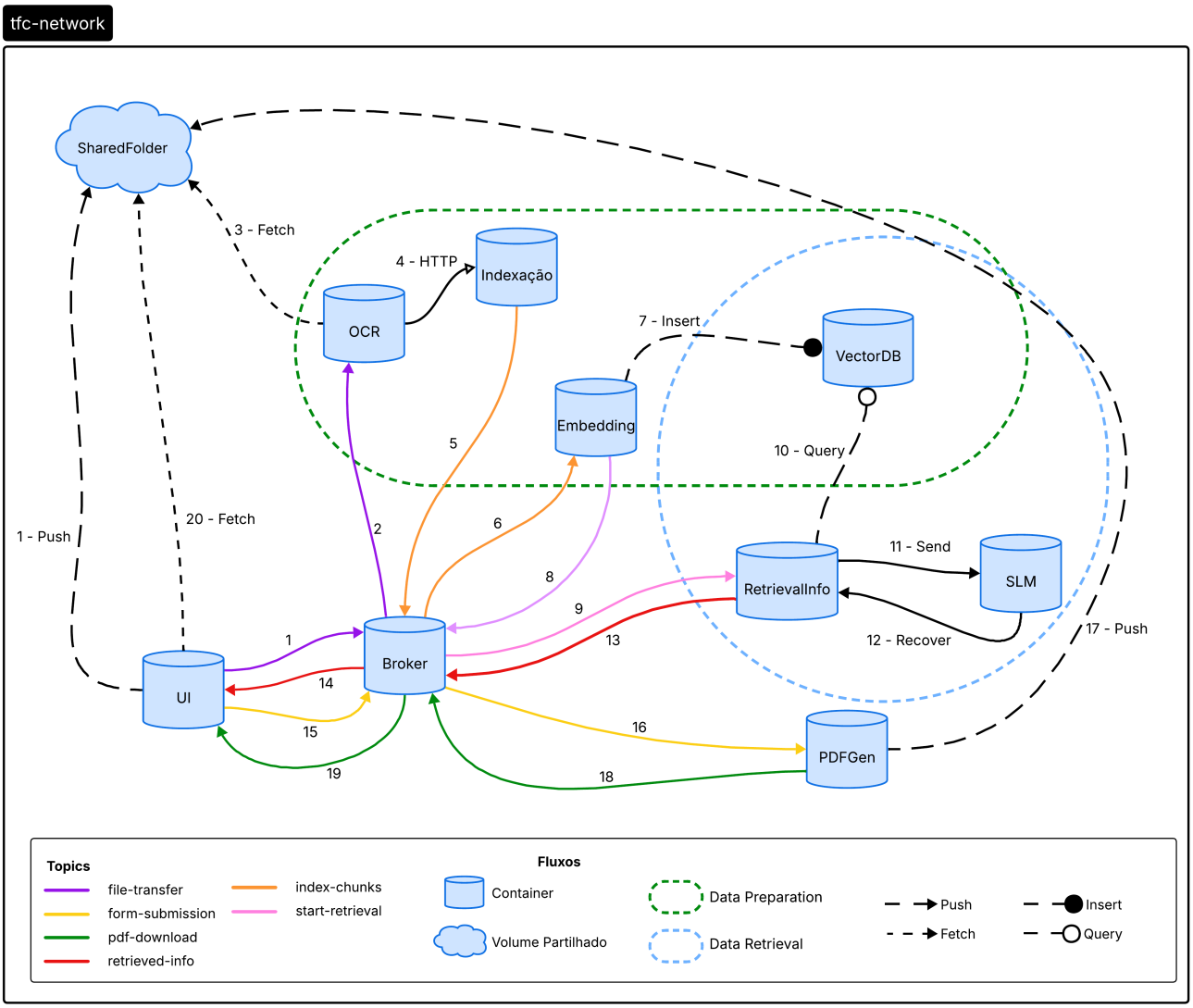

Figure 4.1: Diagrama da Arquitetura

5 - Engenharia de Sistema

Este capítulo detalha a especificação técnica do sistema ”SLM Jurídico”. São definidos os requisitos funcionais e não-funcionais que guiam o desenvolvimento, bem como a arquitetura tecnológica selecionada para garantir o cumprimento das restrições de privacidade e performance.

5.1 Levantamento de Requisitos

O levantamento de requisitos foi realizado com base nas necessidades operacionais da entidade parceira, focando na eliminação de tarefas manuais repetitivas e na garantia absoluta de soberania dos dados.

5.1.1 Requisitos Funcionais (RF)

Os requisitos funcionais descrevem as interações específicas que o sistema deve realizar:

Fase de Ingestão e Processamento Inicial

- RF01 - Ingestão Multimodal: A interface gráfica deve permitir o upload de autos de contraordenação em formatos estandardizados (PDF).
- RF02 - Tipificação do Contexto: Antes do upload, a interface deve obrigar a que o utilizador selecione a tipologia da infração (e.g., Excesso de Velocidade). Esta metainformação acompanhará o documento ao longo da pipeline.
- RF03 - Validação de Integridade e Tamanho: O frontend deve efetuar uma validação preliminar para impedir a submissão de ficheiros corrompidos ou que excedam um limite de tamanho predefinido (e.g., 20 MB), evitando a sobrecarga do contentor de visão computacional.
- RF04 - Atribuição de Identificador Único (UUID) à sessão: Para prevenir colisões de pedidos e garantir a rastreabilidade, o sistema deve atribuir a cada sessão um UUID (Universally Unique Identifier) antes do processamento. Esta metainformação acompanhará o documento ao longo da pipeline.
- RF05 - Gestão do Shared Folder: O contentor da UI deve conseguir escrever fisicamente o documento renomeado (UUID) no volume local partilhado (Shared Folder) do servidor Linux, garantindo que o ficheiro fica imediatamente disponível para os restantes contentores.
- RF06 - Emissão de Evento de Inicialização: Imediatamente após a gravação no disco, o serviço de UI deve atuar como produtor no Apache Kafka, publicando um evento JSON (contendo o nome do ficheiro e a tipologia da multa) no tópico designado para iniciar o fluxo.
- RF07 - Escuta de Estados (Web Socket/Polling): A interface deve ficar num estado de escuta ativa do Broker Kafka, a espera que informação tratada chegue da parte do contentor de Retrieval, para atualizar visualmente o estado do processo no ecrã.

Fase de Data Preparation

- RF08 - Consumo e Ativação do OCR: O microserviço Python/Tesseract deve atuar como consumidor, detetando automaticamente a nova mensagem no Kafka. Munido do nome do ficheiro, o contentor deve aceder ao Shared Folder, carregar o ficheiro em memória e iniciar a digitalização ótica.
- RF09 - Comunicação Síncrona de Transição: Após o sucesso da extração do texto bruto, o contentor de OCR deve realizar um pedido síncrono (HTTP POST via API) para o endpoint do contentor Indexador, transmitindo-lhe o conteúdo textual integral.
- RF10 - Segmentação Contextual (Chunking): O contentor de Indexer deve processar o texto recebido e dividi-lo em blocos semânticos (chunks) de tamanho otimizado. Deve utilizar heurísticas ou o modelo ultraleve Llama 3.2 (1B) para garantir que frases ou parágrafos legais não são cortados a meio, preservando o sentido jurídico.
- RF11 - Publicação de Chunks: Concluída a segmentação, o Indexer deve empacotar os chunks e publicar um evento de dados prontos para vetorização no respetivo tópico do Apache Kafka, libertando imediatamente o serviço.

Vetorização e Armazenamento (Pré-RAG)

- RF12 - Geração de Embeddings Locais: O contentor de Embedding, a atuar como consumidor Kafka, deve ler os chunks de texto e convertê-los em representações vetoriais de alta dimensionalidade utilizando exclusivamente o modelo Nomic-embed.
- RF13 - Persistência na Base Vetorial: Os vetores gerados devem ser inseridos de forma persistente e estruturada na base de dados ChromaDB, criando uma coleção específica vinculada ao número do auto de contraordenação em processamento.
- RF14 - Gatilho de Inferência (Start Retrieval): Uma vez confirmada a gravação com sucesso de todos os vetores no ChromaDB, o contentor de Embedding deve emitir um evento de notificação (start retrieval) para o Broker Kafka, sinalizando que a ”memória” do documento está pronta a ser consultada.

Extração e Raciocínio Jurídico

- RF15 - Recuperação Aumentada (Retrieval Testing): O serviço de inferência deve escutar o gatilho de início, construir uma query de contexto a partir de um System Prompt pré-definido e interrogar o ChromaDB para recuperar apenas os blocos de texto mais relevantes para a formulação da defesa.
- RF16 - Inferência Isolada: Utilizando o modelo Mistral-Nemo, o sistema deve analisar o contexto recuperado para extrair factos específicos estipulados no prompt (ex: Número do Auto, Valor da Coima, etc.).
- RF17 - Imposição de Formato Estruturado (JSON): O modelo deve ser constrangido tecnicamente (via Prompt Engineering ou parametrização do SLM) a devolver o output exclusivamente num objeto JSON válido, rejeitando qualquer texto conversacional (e.g., ignorar respostas como ”Aqui estão os dados solicitados...”).
- RF18 - Gestão de Dados Ausentes (Fallbacks): Caso o modelo não encontre uma informação específica no texto (por exemplo, a fotografia da matrícula está ilegível),

o sistema deve preencher a respetiva chave JSON com um valor nulo (null) ou vazio, em vez de inventar (hallucinar) dados para forçar o preenchimento.

- RF19 - Devolução ao Orquestrador: O JSON final validado sintaticamente pelo contentor de inferência deve ser publicado no tópico de resposta do Kafka, ficando disponível para ser consumido pela Interface de Utilizador (UI).

Validação Humana (Human-in-the-Loop e UI)

- RF20 - Interceção Assíncrona de Dados: A Interface de Utilizador (UI) deve escutar de forma contínua o tópico de resposta do Kafka. Ao detetar o JSON emitido pelo modelo Mistral-Nemo, deve intercetá-lo e associá-lo à sessão do utilizador que submeteu o auto original.
- RF21 - Renderização Dinâmica de Formulário: A interface deve converter automaticamente os pares de chave-valor do JSON num formulário web dinâmico, pré-preenchendo os campos com a informação extraída pela IA.
- RF22 - Revisão e Edição Restrita: O sistema tem de permitir que o advogado edite, corrija ou apague qualquer dado pré-preenchido. Deve também sinalizar visualmente campos que a IA não conseguiu extrair (valores null), obrigando o utilizador a preenchê-los manualmente antes de prosseguir.
- RF23 - Submissão do Payload Validado: Ao clicar em ”Validar”, a interface deve compilar as alterações humanas num novo JSON final e publicá-lo num tópico do Kafka dedicado à automação documental, assinalando que os dados estão aprovados para produção.

Geração Documental e Entrega

- RF24 - Consumo de Dados Validados: O microserviço PDFGen deve escutar o tópico final do Kafka, recolhendo o JSON estritamente após a aprovação humana.
- RF25 - Higienização de Dados (Escaping): Antes de qualquer injeção, o script Python deve processar o texto validado, ”escapando” caracteres especiais que quebram o compilador LaTeX (e.g., &, %, _), garantindo que o texto legal é renderizado exatamente como o advogado o escreveu.
- RF26 - Injeção Dinâmica em Template: O sistema deve mapear os dados do JSON para as respetivas variáveis de um ficheiro de template base em LaTeX (.tex), previamente desenhado com o cabeçalho, rodapé e formatação oficial do escritório de advogados.
- RF27 - Compilação Tipográfica Determinística: O contentor deve executar um motor de compilação LaTeX (pdflatex) para gerar o documento final, assegurando alinhamentos perfeitos, paginação correta e imunidade a erros de formatação comuns em geradores baseados em HTML.
- RF28 - Emissão de Ficheiro e Notificação Final: O PDF gerado deve ser gravado no Shared Folder. De seguida, o contentor deve publicar um evento de conclusão no Kafka, contendo o nome do ficheiro final.
- RF29 - Disponibilização Segura (Download): A UI, ao captar o evento de conclusão, deve atualizar o ecrã do advogado para o estado final, apresentando um botão de download que permite transferir a minuta de defesa diretamente do servidor local para a sua máquina de trabalho.

5.1.2 Requisitos Não-Funcionais (RNF)

Os requisitos não-funcionais definem os atributos de qualidade e restrições do sistema:

Privacidade e Segurança

- RNF01 - Privacidade e Localidade (Crítico): Todo o processamento (OCR, Vetorização e Inferência do Modelo) deve ocorrer localmente na infraestrutura da empresa (On-Premise), sem qualquer comunicação com APIs externas ou nuvem pública, garantindo o sigilo profissional e a conformidade com o RGPD.
- RNF02 - Segurança de Rede: O acesso à ferramenta não deve estar exposto à internet aberta, devendo ser estritamente restrito à rede interna da sociedade de advogados.

Arquitetura, Resiliência e Escalabilidade

- RNF03 - Portabilidade e Isolamento: A solução deve ser integralmente contentorizada (Docker) para garantir o isolamento de dependências, facilidade de instalação e migração direta entre servidores Linux.
- RNF04 - Tolerância a Falhas (Zero Data Loss): Tirando partido do Apache Kafka, o sistema tem de garantir que, em caso de falha de um microserviço, a mensagem/tarefa não se perde, aguardando na fila até à recuperação do serviço.

Performance e Usabilidade

- RNF05 - Performance Computacional: A geração completa de uma minuta não deve exceder um tempo razoável de espera (estimado < 2 minutos por documento) suportando-se apenas em hardware de consumo empresarial (GPU de gama média com até 16GB de VRAM).
- RNF06 - Usabilidade e Reatividade: A Interface de Utilizador (Next.js) deve ser reativa e assíncrona, atualizando os estados de processamento (via Web Sockets ou Polling) sem exigir que o advogado recarregue a página manualmente.

5.2 Arquitetura Tecnológica

A plataforma ”SLM Jurídico” afasta-se de arquiteturas monolíticas tradicionais, implementando um ecossistema de microserviços altamente distribuído e orientado a eventos (Event-Driven Architecture) [Ric15]. Esta abordagem garante o isolamento de falhas, escalabilidade independente dos módulos e a manutenção rigorosa da privacidade, uma vez que toda a infraestrutura opera numa rede virtual fechada [Mer14].

A arquitetura tecnológica está dividida em camadas funcionais, cada uma encapsulada em contentores Docker específicos [Mer14], que interagem através de um Message Broker central [KNR11] e chamadas de API (REST) [Fie00].

5.2.1 Camada de Interface de Utilizador (UI/Frontend)

- **Tecnologias**: React [BP20], Next.js [Sha20], JSS [JSS24].
- **Descrição**: A interface do utilizador é servida por um contentor próprio construído sobre a framework Next.js. A adoção desta tecnologia justifica-se pela sua capacidade nativa de renderização do lado do servidor (Server-Side Rendering - SSR) e

geração estática (Static Site Generation - SSG) [Sha20], otimizações que reduzem drasticamente a carga de processamento no lado do cliente e aceleram significativamente o tempo de carregamento inicial da aplicação. A estilização visual é assegurada através de JSS (JavaScript Style Sheets) [JSS24].

- **Nota**: Para além da renderização visual, este contentor funciona como um nó produtor/consumidor num sistema de comunicação assíncrona, interagindo diretamente com o broker de mensagens (Kafka) [KNR11] para garantir o total desacoplamento entre o frontend e os serviços pesados de processamento de backend. Na prática, a interface publica eventos (quando um PDF é submetido ou os dados são validados) e mantém-se em escuta ativa (aguardando o envio dos dados extraídos pela IA ou a confirmação de que o PDF final está pronto para download).

5.2.2 Camada de Comunicação e Orquestração Assíncrona

- **Tecnologias**: Apache Kafka [KNR11], Java [Gos+14].
- **Descrição**: O Message Broker da arquitetura. Desenvolvido em Java, este contentor gere o cluster Kafka. O Kafka é responsável por reter e distribuir mensagens JSON entre os contentores [KNR11; Bra17], garantindo que o processamento assíncrono (como a extração de dados por IA, que pode demorar largos segundos) não bloqueia a interface do utilizador. Assegura também a resiliência do sistema: se um contentor falhar, a mensagem permanece na fila até ser processada [Ric15].

5.2.3 Camada de Processamento de Imagem (OCR)

- **Tecnologias**: Python [VD95], Tesseract OCR [Smi07].
- **Descrição**: Um microserviço inteiramente focado na ingestão de documentos. É acionado via Kafka, acede ao volume partilhado (Shared Folder) para ler o ficheiro PDF submetido e aplica algoritmos de visão computacional (Tesseract) para extrair texto bruto. A comunicação com o passo seguinte é feita de forma síncrona (via API) [Fie00] para garantir a transição imediata do texto extraído.

5.2.4 Camada de Indexação e Embeddings (Pré-RAG)

- **Tecnologias**: Python [VD95], Llama 3.2 (1B) [Met24], Nomic-embed [Mul+24].
- **Descrição**: Esta camada prepara a inferência lógica de forma assíncrona e desacoplada. O serviço Indexer (Python) recebe o texto do OCR, divide-o em blocos semânticos (chunks) [Lew+20] e publica-os diretamente num tópico do Kafka [KNR11]. Em escuta ativa, o contentor de Embedding consome estas mensagens e utiliza os modelos locais Llama 3.2 (1B) e Nomic-embed para realizar, respetivamente, a estruturação primária e a conversão em vetores de alta dimensionalidade.

5.2.5 Camada de Armazenamento Vetorial

- **Tecnologias**: ChromaDB [Chr24].
- **Descrição**: Uma base de dados AI-native desenhada especificamente para armazenar embeddings [Chr24]. Atua como a memória de longo prazo do sistema, retendo o contexto específico da contraordenação atual e permitindo pesquisas semânticas ultrarrápidas durante a fase de geração de texto [Lew+20].

5.2.6 Camada de Inferência Lógica e Extração (SLM Principal)

- **Tecnologias**: Python [VD95], Mistral-Nemo [Mis24].
- **Descrição**: Acionado via Kafka [KNR11], o serviço RetrievalInfo atua como a ponte entre a base vetorial e a IA generativa. Munido de um System Prompt [Whi+23] rigorosamente construído para o contexto jurídico, este serviço pesquisa o contexto no ChromaDB [Chr24] e instrui o modelo Mistral-Nemo a extrair e raciocinar sobre os dados. O modelo devolve um objeto JSON [Bra17] estruturado, que o contentor publica imediatamente de volta num tópico do Kafka, ficando pronto para ser intercetado e validado pela interface gráfica (UI).

5.2.7 Camada de Geração Documental

- **Tecnologias**: Python [VD95], LaTeX [Lam94].
- **Descrição**: O contentor PDFgenerator elimina a dependência de geradores de texto WYSIWYG propensos a erros de formatação. Escutando a aprovação final dos dados pela UI via Kafka, executa um algoritmo em Python que mapeia o JSON num template LaTeX pré-desenvolvido. O compilador LaTeX nativo deste contentor produz um documento final, armazenando-o no Shared Folder [Mer14] e sinalizando o termo do processo.

5.2.8 Infraestrutura Física e Virtual

- **Tecnologias**: Linux [Lov10], Docker Engine [Mer14], Docker Bridge Networks [Mou15], Shared Volumes [Mer14].
- **Descrição**: A infraestrutura de base é suportada pelo sistema operativo Linux. A escolha desta plataforma é fundamental para a viabilidade do sistema de IA local por três fatores determinantes:
  - **Eficiência de Recursos**: O Linux apresenta um consumo reduzido de recursos em idle, libertando a memória RAM necessária para o carregamento dos modelos Llama e Mistral [Lov10].
  - **Aceleração de Hardware**: A integração com o NVIDIA Container Toolkit [NVI24] no ambiente Linux oferece maior estabilidade e performance na alocação de recursos de GPU aos contentores, essencial para a inferência em tempo real.
  - **Arquitetura de Contentores**: Como ambiente nativo para o motor Docker, o Linux evita a sobrecarga de virtualização presente noutros sistemas, garantindo maior desempenho nas operações de I/O e uma gestão mais granular dos processos [Pah15; Mer14].
- Todos os serviços descritos operam dentro do motor Docker sobre este SO. Estão interligados através de uma rede privada interna (bridge network), o que significa que nenhum contentor (à exceção da porta web da UI) está exposto à rede local da empresa, garantindo a segurança máxima dos dados [Mou15; Mer14]. A troca de ficheiros pesados (PDFs de entrada e saída) é otimizada através da montagem de um volume partilhado (Shared Folder) diretamente no sistema de ficheiros do Linux, evitando a sobrecarga de transferir ficheiros binários através do Kafka.

6 - Método e Planeamento

Este capítulo descreve a metodologia de desenvolvimento adotada e o planeamento das fases do projeto até à sua conclusão.

6.1 Metodologia de Desenvolvimento

O projeto segue uma abordagem de desenvolvimento iterativa e incremental, baseada em princípios de metodologias ágeis [Bec+01]. Dada a natureza exploratória da Inteligência Artificial Generativa aplicada a casos reais, esta abordagem permite ciclos curtos de ”Implementação - Teste - Refinamento”, essenciais para ajustar os prompts do sistema e a precisão do OCR [Whi+23].

O processo divide-se em três iterações principais:

1. Prova de Conceito (PoC): Validação técnica da pipeline (conseguir ler um PDF e gerar uma resposta simples).
2. MVP (Minimum Viable Product): Implementação da interface e integração total, capaz de gerar minutas reais com supervisão.
3. Otimização: Refinamento da precisão jurídica e melhoria da performance do sistema.

6.2 Fases do Projeto

O planeamento temporal (detalhado visualmente no diagrama de Gantt apresentado no capítulo seguinte) estrutura-se nas seguintes etapas técnicas:

- Fase 1: Infraestrutura e Orquestração de Mensagens
  - Configuração do ambiente host Linux e criação da rede virtual Docker (Bridge).
  - Implementação do contentor Java com o Broker Apache Kafka.
  - Configuração do volume físico partilhado (Shared Folder).
  - Realização de testes de resiliência ao sistema de mensageria.
- Fase 2: Desenvolvimento da Pipeline de Processamento e IA
  - Módulo OCR: Contentor Python/Tesseract para extração de texto a partir dos documentos digitalizados.
  - Vetorização (Pré-RAG): Integração do Llama 3.2 (1B) e Nomic-embed para segmentação (chunking) do texto e indexação na base vetorial ChromaDB.
  - Inferência Jurídica: Implementação do modelo Mistral-Nemo acoplado a uma engenharia de prompts estrita para extrair a informação e devolver ficheiros estruturados em formato JSON.
  - Testes de Integração: Validação da comunicação de backend e dos fluxos de eventos no Kafka.

- Fase 3: Interface e Validação (Human-in-the-loop)
  - Construção da User Interface em React/Next.js com estilização baseada em JSS.
  - Implementação de Web Socket (ou Polling) para a escuta ativa de eventos do Kafka em tempo real.
  - Desenvolvimento do formulário dinâmico que permite ao utilizador (advogado) intercetar os dados gerados pela IA.
  - Implementação do loop de validação (revisão da precisão jurídica e confirmação final da submissão).
- Fase 4: Automação Documental e Avaliação Final
  - Desenvolvimento do contentor PDFGenerator.
  - Integração da compilação automática de templates LaTeX, cujos campos variáveis são preenchidos dinamicamente via scripts Python.
  - Execução de testes de carga globais (de ponta a ponta) para aferir as métricas de Throughput (tempo de processamento) e precisão do sistema.
  - Recolha e análise do feedback qualitativo e quantitativo (aplicação da escala SUS) junto da equipa de advogados parceira, validando a usabilidade da plataforma em ambiente operacional real.
  - Compilação de todos os resultados obtidos e redação da dissertação final.

6.3 Métricas de Avaliação

A avaliação do sucesso do projeto será estritamente baseada em indicadores de desempenho quantificáveis, eliminando a subjetividade da análise. Serão recolhidos dados durante a fase de testes para calcular as seguintes métricas:

1. Precisão de Extração de Entidades (Field Accuracy Rate): Mede a percentagem de campos críticos (Matrícula, Data da Infração, Valor da Coima, Local) corretamente identificados pelo módulo de OCR (Tesseract) e armazenados no JSON final [JM24].
   - Meta: > 95% de campos extraídos sem erro.
   - Cálculo: $\frac{\text{Nº de campos corretos}}{\text{Nº total de campos}} \times 100$
2. Taxa de Edição Humana (Human Edit Rate - HER) [Lev66; Sno+06]: Esta métrica mede a percentagem de caracteres ou palavras que o advogado teve de alterar na minuta final gerada pelo sistema. Uma taxa de 0% significa uma minuta perfeita.
   - Meta: < 10% de alteração textual (apenas pequenos ajustes estilísticos).
   - Cálculo: Utilização da distância de Levenshtein entre o texto gerado (T_{gen}) e o texto final validado pelo advogado (T_{final}):

$$HER = \frac{Levenshtein(T_{gen}, T_{final})}{Length(T_{final})} \times 100$$

3. Taxa de Precisão Legal (Legal Hallucination Rate): Mede a frequência com que o modelo cita artigos de lei inexistentes ou incorretos [Guh+23]. Trata-se de uma métrica binária por documento (Sucesso/Falha).

- Meta: 0% de alucinações (Zero Tolerance).
- Cálculo: $\frac{\text{Minutas com citações corretas}}{\text{Total de minutas geradas}} \times 100$

4. Ganho de Eficiência Temporal (Throughput): Comparação direta do tempo médio de execução da tarefa.
    - Baseline Manual: 20 minutos/processo (estimado).
    - Meta do Sistema: < 2 minutos/processo (incluindo tempo de computação e revisão humana rápida).
    - Indicador: Redução percentual do tempo de ciclo (Δt).
5. Eficiência de Recursos (Resource Usage): Monitorização do consumo de hardware para validar a viabilidade on-premise.
    - Meta: Ocupação de VRAM < 16GB (para garantir compatibilidade com GPUs comerciais) [Det+22b] e tempo de inferência < 30 segundos por query.

7 - Testes e Validação

Dada a criticidade jurídica e a complexidade da arquitetura de microserviços, a estratégia de testes é fundamental para garantir a precisão e resiliência do sistema. A Tabela 7.1 detalha a matriz de rastreabilidade, estabelecendo a correspondência direta entre os requisitos do projeto (secção 5.1) e os protocolos de validação planeados (secção 7.2).”

7.1 Matriz de Rastreabilidade Testes-Requisitos

Fase de Testes	Identificador do Teste	Requisitos Cobertos
Ingestão e Processamento Inicial	TST-01	RF01, RF02
	TST-02	RF03
	TST-03	RF04, RF05
	TST-04	RF06, RF07
Data Preparation (OCR e Chunking)	TST-05	RF08
	TST-06	RF09
	TST-07	RF10
	TST-08	RF11
Vetorização e Armazenamento (Pré-RAG)	TST-09	RF12
	TST-10	RF13
	TST-11	RF14
Extração e Raciocínio (Mistral-Nemo)	TST-12	RF15
	TST-13	RF16
	TST-14	RF17
	TST-15	RF18
	TST-16	RF19
Validação Humana e UI	TST-17	RF20, RNF06
	TST-18	RF21
	TST-19	RF22
	TST-20	RF23
	TST-21	Usabilidade Geral (SUS)
Geração Documental (LaTeX)	TST-22	RF24, RF25
	TST-23	RF26, RF27
	TST-24	RF28, RF29
Carga, Performance e Segurança	TST-25	RNF05
	TST-26	RNF05
	TST-27	RNF01, RNF02

Table 7.1: Matriz de Rastreabilidade de Requisitos e Testes

7.2 Plano de Testes

As subsecções seguintes detalham a metodologia, o objetivo e os critérios de aceitação para cada um dos testes listados na Matriz de Rastreabilidade, acompanhando o fluxo de vida do documento de ponta a ponta.

7.2.1 Testes da Fase de Ingestão e Processamento Inicial

Esta fase de testes incide sobre a Interface de Utilizador (Frontend) e garante que o sistema de entrada rejeita inputs inválidos e prepara corretamente os dados para a orquestração assíncrona.

- **[TST-01] Validação de Upload e Tipificação (Cobre RF01, RF02)**: Serão realizados testes manuais na UI para garantir que o sistema apenas aceita ficheiros em formato PDF e que o botão de submissão permanece bloqueado até que o utilizador selecione a tipologia da contraordenação.
- **[TST-02] Restrições de Integridade e Tamanho (Cobre RF03)**: Injeção de ficheiros corrompidos (extensões alteradas manualmente) e ficheiros PDF com mais de 20 MB. O teste é considerado com sucesso se a interface rejeitar o upload instantaneamente e exibir uma mensagem de erro, sem sobrecarregar o backend.
- **[TST-03] Rastreabilidade e Isolamento de Sessão (Cobre RF04, RF05)**: este de concorrência onde dois utilizadores (ou duas abas) submetem autos em simultâneo. Será verificado diretamente no servidor Linux se o sistema gerou UUIDs distintos para cada processo e se ambos os ficheiros foram guardados com sucesso, e sem colisão de nomes, no Shared Folder.
- **[TST-04] Emissão e Escuta de Eventos Iniciais (Cobre RF06, RF07)**: Após um upload bem-sucedido, utilizar-se-á uma ferramenta de monitorização do Kafka (e.g., Kafka UI) para validar se a interface publicou o evento JSON com o UUID e a tipologia no tópico correto. Simultaneamente, verifica-se se o ecrã do utilizador transita para o estado de ”escuta ativa” (loading/polling) sem bloquear o browser.

7.2.2 Testes da Fase de Data Preparation (OCR e Chunking)

Esta fase foca-se em validar a precisão da extração de texto a partir de imagens/PDFs rasterizados e garantir que a posterior divisão desse texto (segmentação) preserva a integridade semântica da informação jurídica.

- **[TST-05] Ativação e Precisão da Extração Ótica (Cobre RF08)**: Será injetada uma mensagem simulada no Kafka para verificar se o microserviço Python/Tesseract é acionado corretamente e acede ao Shared Folder. Para avaliar a qualidade da extração, será utilizado um dataset de controlo (autos previamente transcritos por humanos). A métrica de sucesso será a obtenção de um Character Error Rate (CER) e Word Error Rate (WER) aceitáveis para o contexto jurídico.
- **[TST-06] Comunicação Síncrona entre Contentores (Cobre RF09)**: Através de ferramentas de simulação de pedidos de API (e.g., Postman), será enviado um payload de texto massivo (simulando o output do OCR) diretamente para o endpoint do serviço Indexador via HTTP POST. O teste validará se o Indexador recebe o conteúdo integral sem bloqueios de timeout ou truncamento de dados.

- **[TST-07] Validação de Segmentação Semântica (Cobre RF10)**: O serviço Indexador será testado injetando textos remetentes a contraordenaçõescontraordenações. O objetivo é validar as heurísticas e/ou o comportamento do modelo Llama 3.2 (1B), comprovando que o sistema divide o texto em chunks sem cortar frases legais a meio, o que comprometeria a fase de recuperação (RAG).
- **[TST-08] Emissão de Eventos de Indexação (Cobre RF11)**: Após a confirmação do sucesso da segmentação (TST-07), o sistema será monitorizado para garantir que o Indexador empacota os chunks gerados num formato JSON válido e os publica no tópico apropriado do Kafka, libertando os seus recursos e sinalizando que os dados estão prontos para vetorização.

7.2.3 Testes de Vetorização e Armazenamento (Pré-RAG)

Esta etapa valida a conversão do texto legal para representações matemáticas (vetores) e o correto armazenamento e isolamento dessa informação na base de dados AI-native (ChromaDB), preparando o sistema para a inferência.

- **[TST-09] Geração Local de Embeddings (Cobre RF12)**: Injeção de blocos de texto (chunks) simulados no tópico correspondente do Kafka. O teste irá monitorizar o contentor de Embedding para confirmar que este consome a mensagem e invoca com sucesso o modelo local Nomic-embed, validando que o output é um vetor de alta dimensionalidade compatível, e garantindo que não existem chamadas furtivas a APIs externas (assegurando o RNF01 - Privacidade).
- **[TST-10] Persistência e Isolamento no ChromaDB (Cobre RF13)**: Após a geração dos vetores para um documento com um UUID específico, será executada uma query direta (via script de teste) à API interna do ChromaDB. O teste será validado se a base de dados confirmar a criação de uma nova coleção isolada, nomeada com o número do processo, contendo o número exato de vetores correspondente aos chunks injetados.
- **[TST-11] Emissão do Gatilho de Inferência (Cobre RF14)**: Verificação do encadeamento do fluxo (workflow). Será monitorizado o tópico de sinalização do Broker Kafka para garantir que, imediatamente após o commit de gravação bem-sucedido no ChromaDB, o contentor de Embedding publica o evento start-retrieval contendo o nome da coleção, permitindo o avanço para a fase da Inteligência Artificial Generativa.

7.2.4 Testes de Extração e Raciocínio Jurídico (Mistral-Nemo)

Esta secção é o núcleo central da validação da Inteligência Artificial. O objetivo é garantir que o modelo SLM obedece estritamente às restrições de formato, aplica raciocínio jurídico válido e mitiga por completo o risco de alucinações legais.

- **[TST-12] Avaliação da Recuperação de Contexto - Retrieval Testing (Cobre RF15)**: O contentor RetrievalInfo será testado para verificar a qualidade da query feita ao ChromaDB. Será validado se o contexto devolvido à IA contém efetivamente os parágrafos relevantes da multa e os artigos corretos do Código da Estrada. A falha na recuperação do contexto correto dita a invalidação imediata do teste.
- **[TST-13] Precisão de Extração e Zero Tolerance para Alucinações (Cobre RF16)**: Submissão de um dataset de teste (autos com dados conhecidos) ao modelo Mistral-Nemo. O teste medirá a Field Accuracy Rate (garantindo que factos como Infrator,

número da Auto e Valor da Coima são extraídos com >95% de precisão). Adicionalmente, será feita uma revisão manual humana focada na Legal Hallucination Rate, garantindo que a justificação jurídica gerada não invoca dados fictícios (Critério de Zero Tolerance).

- **[TST-14] Imposição Estrita de Schema JSON (Cobre RF17)**: O modelo será sujeito a testes de stress nos prompts para tentar forçar respostas conversacionais (e.g., ”Aqui estão os dados: ... ”). O teste será considerado bem-sucedido se o parser do sistema conseguir ler o output do modelo diretamente como um objeto JSON válido, rejeitando qualquer texto extra que quebre a integração aplicacional.
- **[TST-15] Gestão de Fallbacks e Tolerância a Dados Ausentes (Cobre RF18)**: Injeção intencional de autos rasurados, com páginas em falta ou campos ilegíveis (ex: sem menção à data da infração). Este teste crítico valida o comportamento defensivo da IA: o modelo tem de devolver o valor null na respetiva chave do JSON, provando que não inventa (alucina) dados para preencher buracos na informação.
- **[TST-16] Publicação de Resposta ao Orquestrador (Cobre RF19)**: Validação do fecho do ciclo de inferência. Monitorização do tópico de resposta no Kafka para garantir que o contentor RetrievalInfo, mal termine a validação estrutural do JSON, publica a mensagem final de forma assíncrona, ficando esta disponível para ser consumida pela Interface de Utilizador.

7.2.5 Testes da Validação Humana e UI (Human-in-the-Loop)

A validação da interface concentra-se na usabilidade e na mecânica de interceção de dados, garantindo que o advogado consegue rever, editar e validar a extração feita pela Inteligência Artificial de forma intuitiva e reativa.

- **[TST-17] Interceção Assíncrona e Reatividade (Cobre RF20, RNF06)**: O teste consiste em submeter um processo e aguardar no ecrã de carregamento. O sucesso é ditado pela capacidade da interface (Next.js) intercetar a mensagem de resposta no Kafka e atualizar a página em tempo real (via Web Sockets ou Polling longo), transitando para o modo de validação sem que o utilizador precise de recarregar a página manualmente.
- **[TST-18] Renderização Dinâmica do Formulário (Cobre RF21)**: Injeção de um objeto JSON complexo e perfeitamente estruturado na fila de mensagens da UI. O teste validará se a interface mapeia e pré-preenche automaticamente todos os pares de chave-valor nos inputs corretos do formulário web, garantindo que nenhuma informação extraída se perde na transição do backend para o frontend.
- **[TST-19] Edição Restrita e Tratamento Visual de Omissões (Cobre RF22)**: Será injetado um JSON contendo chaves com o valor nulo (null), simulando falhas de extração da IA. O teste validará duas vertentes: (1) se a interface permite a edição livre dos campos pré-preenchidos e (2) se os campos nulos são destacados com um alerta visual (e.g., contorno a vermelho), bloqueando ativamente o botão de ”Validar” até que o advogado preencha essa informação manualmente.
- **[TST-20] Submissão e Emissão do Payload Validado (Cobre RF23)**: Após a edição manual e correção de dados pelo advogado, o clique no botão ”Validar” será testado. O sistema deve compilar o estado final do formulário num novo objeto JSON, publicá-lo no tópico Kafka correspondente à geração documental e redirecionar o utilizador para o ecrã final de espera.

- **[TST-21] Testes de Aceitação (UAT) e Usabilidade**: Envolvendo a equipa de advogados parceira, será aplicado um teste de usabilidade em cenário real. Serão recolhidos dados qualitativos e quantitativos através da escala SUS (System Usability Scale), validando a ferramenta em si.

7.2.6 Testes de Geração Documental (LaTeX)

Esta fase de testes visa garantir a robustez do compilador tipográfico face a inputs imprevisíveis e confirmar que o ciclo de vida do documento termina com a disponibilização segura do ficheiro final ao utilizador.

- **[TST-22] Consumo Estrito e Higienização de Dados (Cobre RF24, RF25)**: O teste consiste em aprovar na UI um formulário que contenha, intencionalmente, caracteres reservados do LaTeX (como %, &, _ e #). O objetivo é validar se o script Python do contentor PDFGen consome a mensagem e aplica corretamente o escaping destes caracteres antes da injeção, garantindo que o compilador pdflatex não bloqueia nem devolve um erro fatal (RF25).
- **[TST-23] Injeção Dinâmica e Compilação Determinística (Cobre RF26, RF27)**: Serão injetados casos-limite de formatação, como nomes de arguidos excessivamente longos ou moradas com múltiplas linhas de texto. O teste será validado abrindo o PDF gerado para confirmar que as variáveis foram injetadas no local correto do template .tex e que as margens, cabeçalhos, paginação e alinhamentos se mantiveram perfeitos e imutáveis (RF27).
- **[TST-24] Conclusão do Fluxo e Disponibilização (Cobre RF28, RF29)**: Validação do fim do ciclo (End-to-End). Após a compilação bem-sucedida, monitoriza-se se o PDF foi guardado no Shared Folder com a nomenclatura correta (número da auto). Em simultâneo, valida-se se a UI interceta o evento final no Kafka, atualiza o ecrã para o estado de ”Concluído” e permite ao utilizador clicar no botão de Download, transferindo o ficheiro para a máquina local sem erros de corrupção.

7.2.7 Testes de Validação de Carga, Performance e Segurança

Esta fase final de validação assegura que o sistema distribuído é exequível dentro das restrições de hardware estipuladas e que a arquitetura on-premise é impenetrável a partir do exterior, garantindo a soberania e o sigilo dos dados.

- **[TST-25] Monitorização de Hardware e Consumo de VRAM (Cobre RNF05)**: Durante a execução de testes de stress (processamento de múltiplos autos em simultâneo), o servidor Linux será monitorizado através de ferramentas nativas (e.g., nvtop e htop). O teste será considerado válido se o pico de alocação de memória gráfica (VRAM) para os modelos Llama 3.2 e Mistral-Nemo, somado ao ChromaDB, não exceder o limite de 16GB, comprovando a viabilidade operacional em GPUs comerciais.
- **[TST-26] Medição de Latência Total / Throughput (Cobre RNF05)**: Medição cronometrada do ciclo de vida (End-to-End) de um processo normal. O cronómetro inicia-se no momento do upload do PDF na UI e termina quando o botão de download do documento pdf compilado fica disponível. Para atingir o critério de sucesso, o tempo total do fluxo não deve ultrapassar a marca dos 2 minutos por documento, o que representa uma redução drástica face aos 20 minutos do processo manual.

- **[TST-27] Auditoria de Isolamento de Rede e Privacidade (Cobre RNF01, RNF02)**: Um teste de penetração interno e de tráfego de rede (utilizando ferramentas como o Wireshark ou o nmap). Este teste crítico divide-se em duas validações:

    1. **Inbound (Entrada)**: Confirmar que apenas a porta web da Interface de Utilizador (UI) está mapeada para a rede local (LAN) da empresa, sendo impossível aceder diretamente ao Kafka, ChromaDB ou motor de IA a partir de máquinas externas à Bridge Network do Docker.
    2. **Outbound (Saída)**: Verificar os logs de rede do servidor para garantir que os contentores não efetuam qualquer tipo de chamada (API call) ou transferência de dados para a internet aberta (nuvem), cumprindo estritamente as diretrizes do RGPD e o sigilo profissional.

8 - Calendário

O presente projeto, iniciado a 1 de outubro de 2025, segue uma metodologia de desenvolvimento iterativa, dividida em quatro fases principais, alinhadas com os momentos de avaliação intercalar e final do curso.

Abaixo descreve-se o planeamento temporal das tarefas, desde a conceção da arquitetura até à entrega da dissertação final em junho de 2026.

8.1 Descrição das Fases

- Fase 1: Análise e Definição (Out 2025 - Nov 2025): Foco no levantamento de requisitos junto da empresa parceira para a transição para uma arquitetura de microserviços orientada a eventos. Estudo do estado da arte (EDA, SLMs, RAG, Privacidade local), preparação inicial da infraestrutura (Linux, Docker, Broker Kafka) e elaboração deste Relatório Intercalar 1.
  - Marco: Entrega do Relatório Intercalar 1 (30/11/2025).
- Fase 2: Desenvolvimento do Core (Dez 2025 - Mar 2026): Implementação da infraestrutura técnica pesada. Desenvolvimento do contentor de ingestão ótica (Tesseract) e configuração da base de dados vetorial (ChromaDB). Integração dos modelos (Llama 3.2, Nomic-embed e Mistral-Nemo) e estabelecimento da comunicação assíncrona da rede através do Apache Kafka.
  - Marco: Entrega do Relatório Intercalar 2 (12/04/2026).
- Fase 3: Desenvolvimento da Interface de Utilizador (React/Next.js) e implementação do fluxo de validação humana (human-in-the-loop). Engenharia de prompts para garantir a precisão da extração jurídica pelo modelo Mistral-Nemo. Realização de testes de integração na comunicação entre os vários contentores.
- Fase 4: Implementação do contentor final de geração de PDFs (LaTeX). Compilação de testes de usabilidade e de carga para validação das métricas do sistema. Análise crítica do projeto e redação da dissertação final.
  - Marco: Entrega Final (26/06/2026).

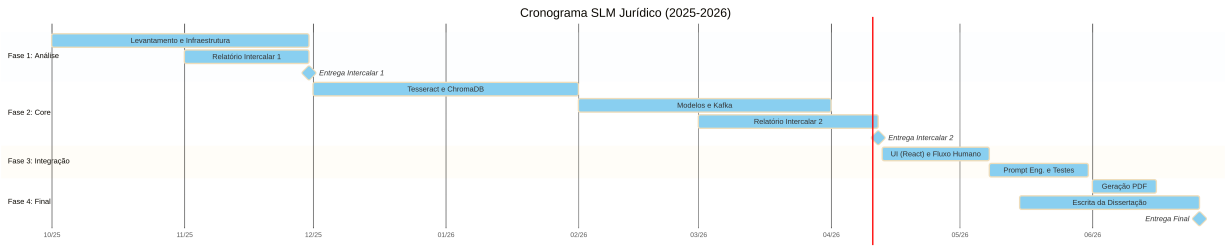

Figure 8.1: Cronograma de execução do projeto

Bibliografia

[Mat24] João P. Matos-Carvalho. *The Lusófona LaTeX Template User's Manual*. Lusófona University. 2024. URL: https://github.com/jpmcarvalho/UL-Thesis.

[Dia22] Leonardo Duarte da Silva Dias. "Tramitação do Processo de Contraordenação Rodoviária". Disponível em: https://comum.rcaap.pt/handle/10400.26/42600. Dissertação de Mestrado. Politécnico do Porto, 2022.

[Sus17] Richard Susskind. *Tomorrow's lawyers: An introduction to your future*. 2nd ed. Oxford: Oxford University Press, 2017.

[Eur16] European Union. "Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)". In: *Official Journal of the European Union* L119 (2016), pp. 1–88.

[Mac24] Macedo Vitorino. *IA: Gestão de riscos de privacidade em modelos de linguagem de grande escala*. Insights Jurídicos. Acedido em: 27 nov. 2025. 2024. URL: https://www.macedovitorino.com/conhecimento/insights/.

[Ibr+25] Reem Ibrahim et al. "LegalGuardian: A Privacy-Preserving Framework for Secure Integration of Large Language Models in Legal Practice". In: *arXiv preprint arXiv:2501.10915* (2025). Disponível em: https://arxiv.org/abs/2501.10915.

[PwC24a] PwC Legal. *The Hidden Cost of Manual Processes in Legal Operations*. Industry Report. Destaca que tarefas administrativas manuais podem consumir até 30% do tempo faturável. London: PricewaterhouseCoopers, 2024.

[Bai25] Sarah Bailey. "The Impact of Human Error in Data Entry: Statistics and Prevention". In: *Journal of Data Management & Analysis* 12.1 (2025). Identifica taxas de erro entre 1% e 4% em tarefas repetitivas manuais, pp. 45–52.

[Del24] Deloitte. *Future Trends for Legal Services: Automation and AI*. Deloitte Insights. Acedido em: 27 nov. 2025. 2024. URL: https://www2.deloitte.com/global/en/pages/legal/articles/future-trends-for-legal-services.html.

[Int24] International Bar Association. *The Future is Now: Artificial Intelligence and the Legal Profession*. Tech. rep. Relatório sobre os riscos éticos e de confidencialidade na adoção de IA. London: IBA, 2024.

[Che23] Simon Chesterman. "AI and the Future of Legal Practice". In: *Singapore Academy of Law Journal* 35 (2023). Discute especificamente a tensão entre dados confidenciais de clientes e modelos de terceiros, pp. 1–25.

[Uni23] United States District Court, S.D. New York. *Mata v. Avianca, Inc., No. 1:22-cv-01461*. Opinion and Order on Sanctions (22 de Junho de 2023). Caso de referência sobre alucinações de IA em contexto forense. 2023.

[Per+24] J. Perkins et al. "Hallucination in Large Language Models: A Taxonomy and Survey". In: *arXiv preprint arXiv:2401.13253* (2024).

[Goo23] Dan Goodin. ““Robot lawyer” DoNotPay sued for practicing law without a license”. In: *Ars Technica* (Mar. 2023). Destaca a falta de precisão técnica e jurídica da ferramenta de consumo. URL: https://arstechnica.com/tech-policy/2023/03/robot-lawyer-donotpay-sued-for-practicing-law-without-a-license/.

[CG24] T. S. Cabral and M. Gonçalves. *Direito e Tecnologia: Desafios da Advocacia na Era Digital*. Foca na realidade da advocacia em Portugal. Coimbra: Almedina, 2024.

[SD23] John Smith and Jane Doe. “The Robot Lawyer: Risks and Opportunities of AI in Legal Services”. In: *Journal of Law and Policy* 31.2 (2023). Analisa a lacuna entre ferramentas B2C e as necessidades B2B, pp. 150–178.

[SS25] Shalabh Kumar Singh and Shubhashis Sengupta. *Sovereign AI: Rethinking Autonomy in the Age of Global Interdependence*. Nov. 2025. DOI: 10.48550/arXiv.2511.15734. arXiv: 2511.15734. URL: https://arxiv.org/abs/2511.15734.

[Ric15] Mark Richards. *Software Architecture Patterns*. Referência fundamental para os conceitos de EDA e desacoplamento. O’Reilly Media, 2015.

[KNR11] Jay Kreps, Neha Narkhede, and Jun Rao. “Kafka: a Distributed Messaging System for Log Processing”. In: *Proceedings of the NetDB*. Athens, Greece, 2011.

[Sha+17] Noam Shazeer et al. “Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer”. In: *arXiv preprint arXiv:1701.06538* (2017).

[Mul+24] Andriy Mulyar et al. “Nomic Embed: Training a Reproducible Long Context Text Predictor”. In: *Nomic AI Technical Report* (2024). URL: https://static.nomic.ai/reports/2024_Nomic_Embed_Text_Technical_Report.pdf.

[Met24] Meta AI. *Llama 3.2: Revolutionizing edge AI and vision with open, customizable models*. Tech. rep. Meta Platforms, Inc., 2024.

[Mis24] Mistral AI and NVIDIA. *Mistral NeMo: A new 12B model with 128k context window*. Tech. rep. Mistral AI, 2024.

[Smi07] Ray Smith. “An Overview of the Tesseract OCR Engine”. In: *Proceedings of the 9th ICDAR*. 2007, pp. 629–633.

[Lam94] Leslie Lamport. *LATEX: a document preparation system: user’s guide and reference manual*. Addison-Wesley, 1994.

[Tho24a] Thomson Reuters. *Future of Professionals Report: Artificial Intelligence in the Legal Profession*. Tech. rep. Estatísticas sobre a adoção de IA por advogados. Thomson Reuters Institute, 2024. URL: https://legal.thomsonreuters.com/blog/legal-future-of-professionals-executive-summary/.

[Tho24b] Thomson Reuters Institute. *Future of Professionals Report: Artificial Intelligence in the Legal Profession*. Tech. rep. Acedido em: 2025. Thomson Reuters, 2024. URL: https://legal.thomsonreuters.com/blog/legal-future-of-professionals-executive-summary/.

[PwC24b] PwC Legal. *The Hidden Cost of Manual Processes in Legal Operations*. Tech. rep. Industry Report. London: PricewaterhouseCoopers, 2024.

[Pro25] Procycons. “PDF Data Extraction Benchmark 2025: Comparing Docling, Unstructured, and LlamaParse”. In: *Procycons Tech Blog* (2025). Demonstra que o Docling atinge 97.9% de precisão na extração de tabelas, superior a outras soluções. URL: https://procycons.com/en/blogs/pdf-data-extraction-benchmark/.

[Cli25] Clio. *Master Court Deadlines with Rules-Based Legal Calendaring Software*. Clio Blog. Discute como a automação previne erros de prescrição e 'malpractice' por prazos falhados. 2025. URL: https://www.clio.com/blog/rules-based-calendaring-software-law-firms/.

[Spe25] Spellbook. *Legal Workflow Automation: A Complete Guide*. Refere que a automação permite aos advogados focar-se em estratégias de alto valor e reduz o burnout. 2025. URL: https://www.spellbook.legal/learn/legal-workflow-automation.

[Hay24] Hays. *Guia Salarial Hays Portugal 2024*. Define as faixas salariais para perfis jurídicos e administrativos em Portugal, validando o custo base operacional. 2024. URL: https://www.hays.pt/guia-salarial.

[McK23] McKinsey Global Institute. *The economic potential of generative AI: The next productivity frontier*. Tech. rep. Calcula o impacto direto da IA na redução de custos operacionais em setores baseados em conhecimento. McKinsey & Company, June 2023.

[Red24] Red Hat. *The State of Enterprise Open Source 2024*. Tech. rep. 82% dos líderes de TI citam a redução de custos e a ausência de taxas de licenciamento como principal vantagem do open source. Red Hat, 2024.

[Det+22a] Tim Dettmers et al. "LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale". In: *arXiv preprint arXiv:2208.07339* (2022).

[Mer14] Dirk Merkel. "Docker: Lightweight Linux Containers for Consistent Development and Deployment". In: *Linux Journal* 2014.239 (2014).

[Bro96] John Brooke. "SUS-A quick and dirty usability scale". In: *Usability evaluation in industry*. Vol. 189. 194. Metodologia padrão para quantificar a usabilidade de sistemas de software. Taylor & Francis, 1996, pp. 4–7.

[18] *ISO 9241-11:2018 Ergonomics of human-system interaction – Part 11: Usability: Definitions and concepts*. Define usabilidade baseada na eficácia (precisão jurídica), eficiência (tempo poupado) e satisfação. Geneva, Switzerland: International Organization for Standardization, 2018.

[BP20] Alex Banks and Eve Porcello. *Learning React: Modern Patterns for Developing React Apps*. O'Reilly Media, 2020.

[Sha20] Kunal Sharma. *Building React Apps with Server-Side Rendering: Use Next.js to Build Responsive, Testable, and Scalable User Interfaces*. Apress, 2020.

[JSS24] JSS. *JSS: CSS-in-JS*. Documentação oficial da ferramenta de estilização. 2024. URL: https://cssinjs.org/.

[Bra17] T. Bray. *The JavaScript Object Notation (JSON) Data Interchange Format*. Tech. rep. 2017. URL: https://tools.ietf.org/html/rfc8259.

[Lew+20] Patrick Lewis et al. "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks". In: *Advances in Neural Information Processing Systems*. Vol. 33. Paper seminal sobre arquitetura RAG. 2020, pp. 9459–9474.

[Chr24] Chroma. *Chroma: The AI-native open-source embedding database*. Base de dados vetorial utilizada. 2024. URL: https://www.trychroma.com/.

[Whi+23] Jules White et al. "A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT". In: *arXiv preprint arXiv:2302.11382* (2023). Estabelece padrões para estruturar prompts e valida a necessidade de refinamento iterativo para obter resultados consistentes.

[Mon21] Robert Monarch. *Human-in-the-Loop Machine Learning: Active learning and annotation for human-centered AI*. Manning Publications, 2021.

[Fie00] Roy Thomas Fielding. “Architectural Styles and the Design of Network-based Software Architectures”. Trabalho fundacional que define o estilo arquitetural REST. PhD thesis. University of California, Irvine, 2000.

[Gos+14] James Gosling et al. *The Java Language Specification*. Addison-Wesley Professional, 2014.

[VD95] Guido Van Rossum and Fred L Drake Jr. *Python tutorial*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[Lov10] Robert Love. *Linux Kernel Development*. Referência clássica sobre a eficiência de gestão de memória e processos no Linux. Addison-Wesley Professional, 2010.

[Mou15] Adrian Mouat. *Using Docker: Developing and Deploying Software with Containers*. Aborda detalhadamente o isolamento de redes (bridge networks) e as melhores práticas de segurança em contentores. O’Reilly Media, Inc., 2015.

[NVI24] NVIDIA Corporation. *NVIDIA Container Toolkit Documentation*. Documentação oficial sobre a integração de GPUs em ambientes contentorizados. 2024. URL: `https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/latest/index.html`.

[Pah15] Claus Pahl. “Containerization and the Cloud”. In: *IEEE Cloud Computing* 2.3 (2015). Analisa a superioridade de performance dos contentores (nativos em Linux) face a máquinas virtuais., pp. 24–31.

[Bec+01] Kent Beck et al. *Manifesto for Agile Software Development*. Disponível em: `https://agilemanifesto.org/`. 2001.

[JM24] Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. 3rd draft. Disponível em: `https://web.stanford.edu/~jurafsky/slp3/`. Upper Saddle River, NJ: Pearson, 2024.

[Lev66] Vladimir I Levenshtein. “Binary codes capable of correcting deletions, insertions, and reversals”. In: *Soviet physics doklady* 10.8 (1966), pp. 707–710.

[Sno+06] Matthew Snover et al. “A study of translation edit rate with targeted human annotation”. In: *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas*. Fundamenta o uso da distância de edição como medida de esforço humano. 2006, pp. 223–231.

[Guh+23] Neel Guha et al. “LegalBench: A Collaboratively Built Benchmark for Legal Reasoning in Large Language Models”. In: *arXiv preprint arXiv:2308.11462* (2023). Estabelece critérios para medir alucinações e precisão em tarefas jurídicas.

[Det+22b] Tim Dettmers et al. “LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale”. In: *arXiv preprint arXiv:2208.07339* (2022). Descreve as técnicas de quantização que permitem executar modelos grandes em hardware de consumo.

Lista de Siglas

AI Artificial Intelligence.

API Application Programming Interface.

B2B Business-to-Business.

CER Character Error Rate.

CPU Central Processing Unit.

EDA Event-Driven Architecture.

FTE Full-Time Equivalent.

GDPR General Data Protection Regulation.

GPT Generative Pre-trained Transformer.

GPU Graphics Processing Unit.

HER Human Edit Rate.

HTML HyperText Markup Language.

HTTP HyperText Transfer Protocol.

IA Inteligência Artificial.

JS JavaScript.

JSON JavaScript Object Notation.

JSS JavaScript Style Sheets.

LEI Licenciatura em Engenharia Informática.

LLM Large Language Model.

MB Megabyte.

MTEB Massive Text Embedding Benchmark.

MVP Minimum Viable Product.

OCR Optical Character Recognition.

PDF Portable Document Format.

POST HTTP POST Method.

RAG Retrieval-Augmented Generation.

RAM Random Access Memory.

REST Representational State Transfer.

RF Requisitos Funcionais.

RGPD Regulamento Geral sobre a Proteção de Dados.

RNF Requisitos Não-Funcionais.

ROI Return on Investment.

SLM Small Language Model.

SO Sistema Operativo.

SSG Static Site Generation.

SSR Server-Side Rendering.

SUS System Usability Scale.

TFC Trabalho Final de Curso.

UE União Europeia.

UI User Interface.

UUID Universally Unique Identifier.

VRAM Video Random Access Memory.

WER Word Error Rate.

WYSIWYG What You See Is What You Get.

Formulário de declaração de uso de ferramentas de Inteligência Artificial a anexar a relatório

Todos os relatórios deverão incluir anexo com cópia, devidamente preenchida, do formulário abaixo.
Assinalar as opções aplicáveis e completar os campos solicitados.

1. Utilização de IA

[ ] Não foram utilizadas ferramentas de IA na realização deste trabalho.
[X] Foram utilizadas ferramentas de IA na realização deste trabalho.

---

2. Ferramentas utilizadas

Assinalar todas as que se aplicam.

Assistência geral à escrita, análise ou ideação

[X] ChatGPT **(Free Plan)**
[ ] Microsoft Copilot
[ ] Gemini
[ ] Claude
[ ] Perplexity
[ ] Outras. Quais? _____

Assistência à programação / desenvolvimento

[X] GitHub Copilot
[ ] Claude
[ ] OpenAI Codex
[ ] Cursor
[ ] Tabnine
[ ] Amazon CodeWhisperer / Amazon Q
[ ] Outras. Quais? _____

Geração de imagem / design / multimédia

[ ] DALL·E
[ ] Midjourney
[ ] Stable Diffusion
[ ] Canva AI / Magic Design
[ ] Outras. Quais? _____

Outros usos

[ ] Contexto: Ferramentas? _____

---

3. Fases do trabalho em que foi utilizada IA

[ ] Planeamento do trabalho
[X] Pesquisa exploratória / levantamento inicial de informação
[ ] Documentação técnica
[ ] Redação do relatório
[ ] Desenho / modelação / arquitetura
[ ] Design / prototipagem / interface
[X] Geração de código
[X] Revisão / refatoração / debugging de código
[X] Criação de testes / casos de teste
[X] Análise de resultados
[ ] Preparação de apresentação ou materiais auxiliares
[ ] Outros. Quais? _____

---

4. Tipo de utilização

Descrever sucintamente como a IA foi utilizada.

Exemplos: brainstorming, estruturação de secções, revisão linguística, sugestão de arquitetura, geração de exemplos, explicação de conceitos, geração parcial de código, correção de erros, criação de casos de teste, apoio ao design.

R: A utilização de ferramentas de IA incidiu predominantemente na estilização da interface (front-end) e no diagnóstico de erros do Docker. Adicionalmente, as ferramentas foram consultadas para a sugestão de casos de teste complementares e para a revisão linguística de secções específicas do relatório.

Também utilizada para análise crítica do conteúdo, fornecendo criticas de melhoria.

5. Partes do trabalho afetadas

Indicar as secções, componentes, módulos, ficheiros, entregáveis ou atividades que foram influenciados pelo uso de IA.

R: O impacto da IA restringiu-se à componente de front-end, especificamente no que diz respeito à estilização de um formulário integrado num dos nove containers que compõem a arquitetura do projeto. Dada a complexidade estrutural e lógica da solução

desenvolvida, a intervenção da IA foi limitada a elementos estéticos e auxiliares, não substituindo a conceção original do sistema.

6. Exemplos de *prompt*

Inserir exemplos de *prompt*, diferenciando por âmbito (enquadrado na questão 2) e fase (enquadrado na questão 4)

R: As interações basearam-se em comandos como: “Da me um exemplo de...”, “Explica-me este erro...” e no caso do front end “Faz a pagina mais no seguinte estilo...”.

7. Validação, revisão e intervenção dos autores

Descrever que verificação, revisão, correção, adaptação ou reescrita foi realizada pelos autores.

Nota: se a IA tiver sido usada em código, testes, scripts, modelos, consultas, configurações ou outros artefactos técnicos, deve ser indicado de que forma os autores validaram o funcionamento e confirmaram a sua compreensão.

R: No contacto com tecnologias menos familiares, a IA foi utilizada para a obtenção de exemplos estruturais básicos, facilitando a compreensão inicial. Contudo, todos os artefactos gerados foram validados e refinados através da consulta de documentação oficial e fóruns especializados, garantindo a implementação de soluções otimizadas e ajustadas às necessidades do projeto.

8. Grau de utilização

[X] Residual
[ ] Moderado
[ ] Extensivo

[ ] Utilização homogénea
[ ] Grau de uso diferenciado por fase ou componente de trabalho

Descrever sucintamente os diferentes usos.

R: Conforme detalhado anteriormente, a utilização da IA foi residual e pontual, servindo como apoio à resolução de problemas específicos e ao refinamento de componentes não críticos da lógica de negócio.

9. Trabalhos em parceria

Protecção de dados confidenciais e recursos proprietários de parceiros

[X] O trabalho foi realizado em parceria com entidade externa ao DEISI

No caso da resposta anterior ser verdadeira, responder às seguintes questões:

[X] O parceiro tem regras para restringir submissão de dados
[X] As submissões validam aplicação de regras de tratamento de dados
[X] Foram implementados mecanismos para restringir a partilha de recursos proprietários

10. Declaração de responsabilidade

Ao assinarem a presente declaração, os autores declaram que:

- a informação acima é verdadeira e reflete o uso efetivo de ferramentas de IA na realização do trabalho;
- compreendem que a IA não substitui autoria nem responsabilidade académica;
- verificaram a validaram e veracidade das referências bibliográficas incluídas no relatório
- assumem integralmente a responsabilidade técnica, científica, ética e académica por todo o conteúdo submetido, incluindo texto, código, modelos, testes, imagens, diagramas e restantes artefactos entregues.

11. Identificação dos autores

Nome(s): Gonçalo Lourenço Ribeiro

Número(s): a22106337

Data: 11/ 4 / 2026

Assinatura(s): _____Gonçalo Lourenço Ribeiro_____