
**FACULDADE DE CIÊNCIAS**

**Departamento de Matemática e Informática**

Trabalho de Licenciatura em

Informática

# Representação de Processos Bancários em Grafo

**Autor**: Bruno Manuel Tomé Rodrigues

Maputo, 5/22/2026

UNIVERSIDADE
EDUARDO
MONDLANE

**FACULDADE DE CIÊNCIAS**

**Departamento de Matemática e Informática**

Trabalho de Licenciatura em

Informática

# Representação de Processos Bancários em Grafo

**Autor**: Bruno Manuel Tomé Rodrigues

**Supervisor (a)**: Rossana Soare

Maputo, 5/22/2026

# Dedicatória

*Dedico este trabalho à minha querida mãe,* R*osângela Issa Tomé Mamad,*

*pois este sonho foi sonhado ao seu lado, e ao meu pai, Manuel da* R*ocha* R*odrigues,*

*pois tudo isto não foi nada* mais *do que uma tentativa de o orgulhar.*

Maputo, 5/22/2026

# Declaração de Honra

Declaro por minha honra que o presente Trabalho de Licenciatura é resultado da minha investigação e que o processo foi concebido para ser submetido apenas para a obtenção do grau de Licenciado em informática, na faculdade de Ciências da Universidade Eduardo Mondlane.

Maputo, janeiro de 2025

_____

(Bruno Manuel Tomé Rodrigues)

# Agradecimentos

A minha supervisora, Rossana Soares, agradeço pela orientação, pela disponibilidade e pela partilha de conhecimento ao longo desta jornada académica. O seu apoio foi essencial para a concretização deste trabalho.

Aos meus pais e à minha família, expresso a minha mais profunda gratidão pelo apoio, pelo suporte incondicional e pelo exemplo constante de dedicação, e perseverança. Foram a minha base ao longo de toda esta caminhada.

Aos meus colegas e amigos, agradeço pela camaradagem, pelos momentos de descontração que aliviaram os dias mais exigentes e pelas voltas a casa repletas de bom humor. Entre risos e partilhas, tornaram esta caminhada verdadeiramente memorável.

Aos escritores, poetas e músicos que me fizeram companhia durante este trajecto, deixo igualmente o meu reconhecimento. As palavras e melodias foram refúgio e inspiração em todos os momentos.

A todos, o meu sincero e profundo agradecimento.

Bruno M. T. Rodrigues.

# Resumo

Os bancos comerciais em Moçambique assumem um papel central na economia nacional, ao apoiarem o crescimento das empresas e assegurarem a gestão financeira das famílias. Nos últimos anos, estas instituições têm expandido o acesso aos serviços financeiros, aproximando-os de um número crescente de cidadãos e reforçando a sua importância no quotidiano da sociedade.

O acesso aos serviços bancários influencia directamente a qualidade de vida da população, ao proporcionar maior segurança das poupanças, facilitar o acesso ao crédito para projectos pessoais e garantir a realização de pagamentos essenciais. Deste modo, um sistema bancário eficiente e funcional constitui um elemento indispensável para o fortalecimento da inclusão financeira e para a promoção do desenvolvimento económico e social do país.

Para assegurar este funcionamento, os bancos organizam-se entre os balcões, responsáveis pela recepção dos pedidos, e os serviços centrais, encarregues da validação das operações. Deste modo, a realização de qualquer serviço bancário depende da comunicação contínua e da coordenação entre estas duas unidades.

No entanto, os bancos ainda enfrentam dificuldades na integração de serviços que dependem de etapas sucessivas. Na prática, quando um pedido é concluído, o cliente vê-se obrigado a regressar ao banco apenas para solicitar a fase seguinte do processo, uma vez que o sistema não assegura essa continuidade de forma automática.

Este fluxo ineficiente provoca perdas de tempo para os clientes e contribui para a sobrecarga dos balcões, originando filas extensas e demoradas. Para o banco, esta situação implica um maior consumo de recursos e aumenta o risco de insatisfacção dos clientes, devido à demora e à excessiva burocracia associadas ao atendimento.

Deste modo, a ausência de continuidade entre os serviços prejudica não apenas o quotidiano das famílias, mas também a eficiência operacional das próprias instituições bancárias, tornando os processos mais lentos e dispendiosos.

Perante este cenário, o presente trabalho de licenciatura propõe a análise da actual estrutura dos serviços bancários, com o objectivo de identificar as falhas que comprometem a continuidade dos processos. Desenvolveu-se uma solução integradora, concebida para se adaptar ao ambiente existente sem exigir a substituição dos sistemas actuais nem investimentos excessivos.

Esta proposta visa promover uma organização operacional mais eficiente e de baixo custo de implementação, disponibilizando a estrutura necessária para unificar serviços isolados, reduzir custos operacionais e melhorar a eficiência do atendimento prestado ao cliente.

A metodologia adoptada durante a elaboração deste trabalho baseou-se numa abordagem mista, combinando investigação qualitativa, análise documental e modelação técnica. Inicialmente, realizou-se a observação directa dos processos operacionais e a análise dos sistemas utilizados na instituição em estudo, com o objectivo de identificar limitações relacionadas com a fragmentação e continuidade dos fluxos bancários. Em seguida, aplicaram-se conceitos da Teoria de Grafos para a modelação formal dos ciclos de vida dos processos, complementados pelo desenvolvimento de uma infra-estrutura integradora composta por uma API central e uma plataforma web de gestão. Por fim, foram realizados testes técnicos e de desempenho para validar a viabilidade da solução proposta em ambientes operacionais heterogéneos.

**Palavras-chave:** Serviços bancários; Integração de sistemas; Continuidade de processos; Eficiência operacional; Inclusão financeira; Transformação digital.

# Abstract

Commercial banks in Mozambique play a central role in the national economy by supporting business growth and ensuring the financial management of families. In recent years, these institutions have expanded access to financial services, bringing them closer to a growing number of citizens and reinforcing their importance in everyday society.

Access to banking services directly influences the population's quality of life by providing greater security for savings, facilitating access to credit for personal projects, and enabling essential payments. Therefore, an efficient and functional banking system constitutes an indispensable element for strengthening financial inclusion and promoting the country's economic and social development.

To ensure this operation, banks are organised between branch offices, responsible for receiving requests, and central services, responsible for validating operations. Thus, the execution of any banking service depends on continuous communication and coordination between these two units.

However, banks still face difficulties in integrating services that depend on successive stages. In practice, when a request is completed, the customer is required to return to the bank merely to request the next stage of the process, since the system does not automatically ensure such continuity.

This inefficient flow causes customers to lose time and contributes to the overload of branch offices, resulting in long and time-consuming queues. For banks, this situation implies greater resource consumption and increases the risk of customer dissatisfaction due to delays and excessive bureaucracy associated with service delivery.

In this way, the absence of continuity between services negatively affects not only the daily lives of families, but also the operational efficiency of banking institutions themselves, making processes slower and more costly.

Given this scenario, the present dissertation proposes an analysis of the current structure of banking services in order to identify the failures that compromise process continuity. The aim is to develop an integrated solution designed to adapt to the existing environment without requiring the replacement of current systems or excessive investment.

This proposal seeks to promote a more efficient and low-cost operational structure by providing the necessary framework to unify isolated services, reduce operational costs, and improve the efficiency of customer service delivery.

The methodology adopted in this work was based on a mixed approach, combining qualitative research, documentary analysis, and technical modelling. Initially, direct observation of operational

processes and analysis of the systems used within the institution under study were conducted in order to identify limitations related to fragmentation and process continuity in banking services. Subsequently, concepts from Graph Theory were applied to formally model the lifecycle of processes, complemented by the development of an integrated infrastructure composed of a central API and a web-based management platform. Finally, technical and performance tests were carried out to validate the feasibility of the proposed solution within heterogeneous operational environments.

**Keywords:** Banking services; Systems integration; Process continuity; Operational efficiency; Financial inclusion; Digital transformation.

# Glossário de Termos

| Nome | Definição |
|---|---|
| Container (Contentor) | Unidade padrão de software que empacota o código e todas as suas dependências, permitindo que a aplicação seja executada de forma rápida e confiável em diferentes ambientes de computação (Docker, 2026). |
| Cookie | Mecanismo de persistência no lado do cliente utilizado para armazenar identificadores de sessão ou *tokens*, desempenhando um papel crucial na gestão de estado e segurança em Arquitecturas web (Giretti, 2023). |
| Multipart Form-Data | Tipo de codificação de dados utilizado em formulários web para permitir o envio de ficheiros binários juntamente com campos de texto (Mozilla, 2026). |
| Promtail | Agente de transporte de *logs* que recolhe registos locais e os envia para uma instância centralizada do Loki (Grafana Labs, 2026). |

# Índice

# Lista de Figuras

# 1. Introdução

## 1.1 Contextualização

O sector bancário em Moçambique tem registado uma expansão significativa nos últimos anos. De acordo com o Relatório de Inclusão Financeira do Banco de Moçambique (2023), o número de moçambicanos com acesso a contas bancárias aumentou consideravelmente, impulsionado pela expansão da rede de agentes bancários e pela digitalização dos serviços financeiros.

Este crescimento enquadra-se nos objectivos definidos pela Estratégia Nacional de Inclusão Financeira, que visa assegurar a integração da população, incluindo a residente em zonas rurais, no sistema financeiro formal, contribuindo para o fortalecimento da economia nacional (Banco de Moçambique, 2022).

Contudo, este rápido crescimento no número de utilizadores trouxe novos desafios operacionais para as instituições bancárias. Embora um maior número de cidadãos utilize os serviços financeiros, muitas instituições continuam a enfrentar dificuldades na gestão contínua deste fluxo, o que resulta na sobrecarga dos balcões e em processos fragmentados que obrigam os clientes a intervenções repetitivas (Diário Económico, 2024; O País, 2024).

Com o avanço das Tecnologias de Informação e Comunicação (TIC), torna-se essencial modernizar a gestão operacional através da adopção de soluções capazes de coordenar processos que actualmente funcionam de forma isolada e fragmentada. Neste contexto, o presente trabalho propôs uma reestruturação da forma como os serviços bancários são geridos, por meio de uma solução integradora concebida para actuar sobre o ambiente bancário existente.

Em vez de apenas automatizar serviços individuais, a proposta procura criar uma base capaz de ligar diferentes processos que fazem parte do atendimento ao cliente. Deste modo, a solução permite estabelecer ligação entre diferentes unidades operacionais, eliminando barreiras estruturais que actualmente comprometem a continuidade dos processos e obrigam o cliente a intervir manualmente para assegurar a transição entre serviços.

A proposta consiste no desenvolvimento de uma infra-estrutura composta por uma API (Application Programming Interface) e por uma plataforma web de gestão. A API actua

como uma camada de abstracção que permite a interligação dos sistemas existentes a um ponto central de controlo, transformando um ambiente operacional heterogéneo numa estrutura mais integrada, sem exigir a reformulação dos sistemas legados.

Através da plataforma web, os administradores podem definir e coordenar o fluxo dos processos, enquanto as unidades operacionais utilizam a API para gerir, de forma unificada, o ciclo de vida de cada serviço. Deste modo, todos os processos independentes passam a ser acompanhados a partir de um ponto comum, proporcionando aos gestores uma visão em tempo real por meio de dashboards, o que assegura maior transparência, rastreabilidade e redução dos custos de integração e operação.

## 1.2 Definição do Problema

As instituições bancárias modernas operam, frequentemente, sobre ambientes tecnológicos compostos por múltiplos sistemas legados desenvolvidos em períodos distintos e com finalidades específicas. De acordo com estudos recentes sobre transformação digital no sector financeiro, a coexistência de plataformas heterogéneas continua a representar um dos principais obstáculos à interoperabilidade e à integração eficiente dos serviços bancários (Jin et al., 2024).

Na instituição em estudo, esta realidade manifesta-se através da existência de diferentes sistemas responsáveis por processos específicos, distribuídos entre balcões, serviços centrais e departamentos operacionais distintos. Embora cada plataforma consiga executar adequadamente as suas próprias funções, os sistemas operam de forma isolada, utilizando estruturas de dados, regras de negócio e mecanismos de tramitação independentes.

Essa fragmentação cria limitações significativas na continuidade dos serviços, sobretudo quando um mesmo cliente possui múltiplos processos relacionados entre si, mas registados em plataformas diferentes. Nesses cenários, o ambiente actual não dispõe de uma infra-estrutura comum capaz de acompanhar esses fluxos de forma integrada, dificultando a continuidade operacional entre etapas dependentes.

Como consequência, a transição entre processos distintos depende frequentemente de intervenções manuais realizadas pelos colaboradores e, em determinados casos, do próprio cliente, que necessita regressar ao banco para iniciar uma nova solicitação após a conclusão da anterior. Este modelo operacional contribui para o aumento do tempo de atendimento,

para a formação de filas nos balcões e para o crescimento do esforço operacional associado ao tratamento manual de informações entre departamentos.

Além disso, a ausência de uma estrutura comum de representação dos processos impede que a instituição possua uma visão transversal sobre os fluxos em execução, dificultando a monitorização global do ambiente operacional e limitando a capacidade de coordenação entre serviços distintos. A literatura recente destaca que ambientes heterogéneos desta natureza tendem a aumentar os custos de integração, reduzir a flexibilidade operacional e dificultar a evolução tecnológica das instituições financeiras (Belchior et al., 2025).

Actualmente, cada sistema gere os seus próprios fluxos de forma isolada, sem uma camada comum responsável pela representação uniforme dos estados e ciclos de vida dos processos. Como resultado, sempre que um novo fluxo necessita de interagir com outro serviço já existente, torna-se necessário desenvolver mecanismos específicos de integração, aumentando a complexidade operacional e tecnológica do ambiente.

Diante deste cenário, torna-se necessária a criação de uma infra-estrutura integradora capaz de actuar sobre o ambiente existente com o menor impacto possível nos sistemas actuaes. A proposta não pretende substituir os sistemas legados nem executar directamente a coordenação dos processos, mas sim disponibilizar uma base comum de representação e gestão que permita transformar o ambiente operacional heterogéneo numa estrutura mais homogénea.

Para alcançar este objectivo, propõe-se o desenvolvimento de uma solução composta por uma API central e uma plataforma de gestão baseada na representação de processos através da Teoria de Grafos. A solução permitirá que diferentes sistemas passem a operar sobre uma mesma estrutura conceptual, mantendo as suas particularidades operacionais, mas partilhando uma base uniforme de representação dos fluxos e dos seus ciclos de vida.

Com a adopção desta infra-estrutura comum, torna-se possível reduzir redundâncias operacionais, melhorar a rastreabilidade dos processos, diminuir a dependência de intervenções manuais e criar as condições necessárias para futuras capacidades de coordenação transversal entre serviços distintos, sem exigir reformulações profundas dos sistemas actualmente utilizados pela instituição.

## 1.3 Justificativa

A realização deste trabalho é motivada pela necessidade de reduzir as limitações operacionais provocadas pela fragmentação tecnológica existente no ambiente bancário da instituição em estudo. Apesar da evolução das Tecnologias de Informação no sector financeiro, muitas instituições continuam a operar sobre ambientes compostos por sistemas legados desenvolvidos em períodos distintos e concebidos para responder a necessidades específicas de diferentes áreas operacionais. Esta realidade conduz à formação de ambientes heterogéneos, nos quais os sistemas funcionam de forma isolada, dificultando a continuidade e a gestão transversal dos processos.

Nesse contexto, a ausência de uma infra-estrutura comum de representação e acompanhamento dos fluxos gera diversos constrangimentos operacionais, sobretudo quando um mesmo cliente possui múltiplos processos relacionados entre si, mas registados em plataformas distintas. Tal situação aumenta a dependência de intervenções manuaes entre departamentos, obriga o cliente a participar activamente na continuidade entre serviços e contribui para o crescimento das filas, do retrabalho e do esforço operacional associado ao tratamento repetitivo de informações.

A proposta de desenvolvimento de uma infra-estrutura integradora baseada numa representação uniforme dos processos surge como uma resposta prática a estas limitações. A solução permitirá transformar o ambiente operacional heterogéneo numa estrutura mais homogénea, sem exigir substituição integral dos sistemas actualmente utilizados. Além disso, possibilitará melhorar a rastreabilidade dos processos, reduzir redundâncias operacionais, disponibilizar uma visão mais ampla sobre os fluxos em execução e criar condições técnicas para futuras capacidades de integração entre serviços distintos.

Do ponto de vista académico e profissional, este trabalho constitui igualmente uma oportunidade para aplicar, de forma prática e contextualizada, os conhecimentos adquiridos ao longo da formação em Informática. Através da utilização de conceitos relacionados com Teoria de Grafos, Engenharia de Software, *APIs* e interoperabilidade de sistemas, procura-se demonstrar como estruturas matemáticas e técnicas modernas de desenvolvimento podem ser utilizadas na construção de soluções capazes de modernizar ambientes tecnológicos complexos sem comprometer os sistemas legados existentes.

Finalmente, a iniciativa encontra-se alinhada com as tendências actuaes de digitalização e modernização dos serviços financeiros, promovendo uma abordagem mais eficiente, sustentável e adaptável para a gestão de processos bancários. Assim, o presente trabalho representa não apenas uma resposta a um problema técnico identificado na instituição em estudo, mas também uma aplicação prática do conhecimento científico na construção de soluções capazes de responder aos desafios reais enfrentados pelas organizações modernas.

## 1.4 Objectivos

### 1.4.1 Objectivos Gerais

Desenvolver uma infra-estrutura integradora baseada na Teoria de Grafos para uniformizar a representação e gestão de processos bancários provenientes de sistemas heterogéneos.

### 1.4.2 Objectivos Específicos

- Analisar os processos de negócio e identificar as dependências entre os diferentes sistemas da instituição em estudo.
- Modelar os ciclos de vida dos processos bancários e as suas transições utilizando as propriedades matemáticas dos grafos direcionados.
- Implementar uma API central capaz de fornecer uma camada comum de representação e gestão dos processos.
- Testar a viabilidade desta abordagem em termos técnicos mensuráveis.

# 2. Revisão Literária

## 2.1. Introdução à revisão de literatura

O presente capítulo desempenha um papel essencial na construção do entendimento sobre a representação de processos de negócio através de estruturas de grafos. A revisão de literatura consiste num processo sistemático de levantamento, análise e discussão de produções académicas e científicas já publicadas, com a finalidade de fundamentar teoricamente a pesquisa, identificar lacunas de conhecimento e contextualizar o problema em estudo. Segundo Gil (2008), "a revisão da literatura consiste no exame de publicações relevantes sobre o tema de pesquisa, permitindo compreender o estado actual do conhecimento e orientar a investigação".

O objectivo central desta revisão é fornecer uma base teórica consistente para o desenvolvimento da infra-estrutura integradora proposta, destacando como a formalização matemática e as tecnologias modernas de interoperabilidade e integração de sistemas podem mitigar a fragmentação operacional em instituições bancárias.

## 2.2 Sistemas de Informação e a Jornada do Cliente Bancário

De acordo com Laudon e Laudon (2021), "um Sistema de Informação pode ser definido como um conjunto organizado de componentes que recolhem, processam, armazenam e distribuem informações". No contexto bancário, a eficiência destes sistemas depende da capacidade de integração entre diferentes unidades operacionais e da continuidade dos processos entre departamentos distintos.

Entretanto, a existência de "ilhas de informação" ou silos tecnológicos constitui um obstáculo crítico à continuidade operacional. Segundo Jin et al. (2024), ambientes compostos por sistemas legados fragmentados tendem a criar dificuldades de interoperabilidade, redundância operacional e dependência excessiva de intervenções humanas.

Em ambientes compostos por sistemas heterogéneos, a ausência de uma estrutura comum de representação dificulta a continuidade entre processos relacionados e limita a capacidade de integração entre departamentos distintos. Estudos recentes demonstram que a coexistência de sistemas legados continua a representar um dos principais desafios à

transformação digital no sector financeiro, sobretudo devido às dificuldades de interoperabilidade e integração operacional (Belchior et al., 2025).

### 2.2.1. Componentes de um Sistema de Informação aplicado ao motor de fluxos

Para que a infra-estrutura integradora proposta funcione, os componentes de um Sistema de Informação devem actuar de forma coordenada:

- Hardware: Refere-se à infra-estrutura física e aos servidores onde a solução será hospedada (Patterson & Hennessy, 2021).
- Software: Representa a camada lógica composta pela API desenvolvida em .NET 8, pela plataforma web de gestão e pelas bases de dados responsáveis pelo processamento das operações (Stair & Reynolds, 2020).
- Dados: Correspondem ao conjunto de informações biográficas, operacionais e transaccionaes necessárias para representar o estado e o ciclo de vida dos processos.
- Processos: Representam o conjunto de actividades e regras de negócio que definem a tramitação dos serviços entre as diferentes áreas da instituição.

## 2.3. Teoria de Grafos na Modelação de Processos

A Teoria de Grafos oferece o rigor matemático necessário para modelar relações e dependências entre processos complexos. Segundo Diestel (2024), um grafo é uma estrutura composta por vértices (nós), que representam estados ou entidades, e arestas (ligações), que representam as relações ou transições entre esses elementos.

Na gestão moderna de processos, a representação através de grafos ultrapassa a simples ilustração visual. De acordo com van der Aalst (2016), a modelação baseada em grafos direccionados permite representar dependências operacionais, identificar gargalos e compreender padrões de comportamento dentro de ambientes organizacionais complexos.

No contexto bancário, os grafos direccionados assumem particular importância devido à necessidade de representar fluxos compostos por múltiplas etapas e intervenientes. Segundo Keller e Trotter (2023), a orientação das arestas garante a preservação da sequência lógica

das transições, assegurando consistência e rastreabilidade ao longo do ciclo de vida dos processos.

A aplicação de grafos permite ainda utilizar algoritmos de busca e optimização para análise estrutural dos fluxos. Segundo Dumas et al. (2020), ao representar processos sob a forma de grafos, torna-se possível identificar caminhos alternativos, dependências críticas e padrões de circulação de informação dentro do ambiente organizacional.2.4. Engenharia de Software e Clean Architecture

A robustez de um motor de fluxos em ambientes críticos depende da sua estrutura interna. Segundo Martin (2017), o objetivo da Arquitectura de software é minimizar o esforço humano na manutenção e evolução do sistema.

A adoção da Clean Architecture garante o desacoplamento do sistema, assegurando a independência entre a lógica de negócio (o infra-estrutura baseada em grafos) e as dependências tecnológicas (bancos de dados ou frameworks). Conforme preconizado por Martin (2017), este isolamento permite que as regras de negócio bancárias permaneçam intactas mesmo diante de trocas de infraestrutura, conferindo ao software a escalabilidade necessária para suportar o crescimento da demanda institucional sem gerar dívida técnica.

## 2.4. Arquitectura em Camadas

A robustez de sistemas integradores em ambientes críticos depende directamente da qualidade da sua estrutura interna. Segundo Martin (2017), o principal objectivo da Arquitectura de Software consiste em reduzir a complexidade de manutenção e evolução do sistema ao longo do tempo.

A adopção da Clean Architecture permite garantir o desacoplamento entre a lógica de negócio e as dependências tecnológicas externas, assegurando maior independência entre os componentes do sistema. Conforme preconizado por Martin (2017), esta abordagem facilita a evolução da infra-estrutura tecnológica sem comprometer as regras centrais de negócio da aplicação.

Segundo Bandarupalli (2025), a modernização de sistemas legados deve privilegiar abordagens graduais e desacopladas, capazes de reduzir o impacto estrutural sobre plataformas críticas já existentes.

No contexto da solução proposta, esta separação torna-se essencial para permitir que a infra-estrutura integradora opere sobre sistemas heterogéneos sem exigir alterações profundas aos sistemas legados existentes. Desta forma, a solução mantém flexibilidade, escalabilidade e capacidade de adaptação a diferentes contextos operacionais.

## 2.5. APIs e Interoperabilidade de Sistemas

Para reduzir a fragmentação operacional existente entre sistemas heterogéneos, utiliza-se a tecnologia de APIs (Application Programming Interface). Segundo Giretti (2024), uma API permite que diferentes aplicações comuniquem entre si de forma padronizada e transparente, abstraindo a complexidade interna de cada sistema.

Entretanto, a simples existência de APIs não garante integração eficiente. O sucesso da comunicação depende directamente da interoperabilidade entre os sistemas envolvidos.

A interoperabilidade é definida como a capacidade de dois ou mais sistemas, dispositivos ou aplicações trocarem dados entre si e utilizarem eficazmente as informações partilhadas (IEEE, 2023). No sector bancário, a interoperabilidade técnica e semântica é fundamental para permitir que diferentes plataformas operem sobre uma mesma estrutura de representação sem perda de significado ou consistência operacional.

Segundo Wieringa (2023), a interoperabilidade constitui um dos principais factores para a modernização organizacional, permitindo que sistemas heterogéneos partilhem informações de forma consistente e reduzindo a fragmentação operacional existente nas instituições.

Belchior et al. (2025) afirmam ainda que a integração entre infra-estruturas heterogéneas depende da utilização de interfaces padronizadas capazes de reduzir a fragmentação operacional sem exigir substituição integral dos sistemas existentes.

Estudos recentes demonstram que a criação de infra-estruturas integradoras representa uma das abordagens mais eficazes para modernizar ambientes financeiros compostos por sistemas

legados, evitando substituições completas e reduzindo os custos de transformação tecnológica (Bandarupalli, 2025).

# 3. Materiais e Métodos

Este capítulo descreve as metodologias de investigação e os materiais adotados nas diferentes fases do estudo. Abrange a recolha de dados fundamentada na experiência profissional no setor bancário, a modelação do sistema através da teoria de grafos e o desenvolvimento técnico da solução.

## 3.1. Abordagem Metodológica

A presente pesquisa adotou uma metodologia mista (qualitativa e técnica), complementada por uma revisão bibliográfica sistemática, com o objetivo de:

- Analisar o fluxo actual de tratamento de processos;
- Identificar as falhas de continuidade e a fragmentação no atendimento ao cliente;
- Propor e validar uma solução baseada em grafos.

Segundo Creswell e Creswell (2018), a metodologia mista permite uma compreensão mais completa dos fenómenos ao combinar forças investigativas distintas. Esta escolha justifica-se pela necessidade de unir a percepção subjetiva do fluxo de trabalho operacional (qualitativa) com a estrutura lógica e o desempenho rigoroso de uma API (técnica).

Trata-se de uma pesquisa de natureza descritiva e exploratória. É descritiva ao mapear como os processos tramitam actualmente e os obstáculos enfrentados. É exploratória ao investigar e testar a viabilidade de uma infra-estrutura integradora baseada em grafos para modernizar a gestão de processos bancários.

## 3.2. Métodos Utilizados

### 3.2.1. Recolha de Dados

#### a) Pesquisa qualitativa e observação participante

Como profissional inserido na indústria, utilizei a observação directa para compreender desafios que a teoria académica muitas vezes não alcança. Esta abordagem justificou-se pela necessidade de analisar o funcionamento real dos processos bancários e identificar

dificuldades operacionais que não se encontram totalmente documentadas nos sistemas formais da instituição.

- Observações directas: Através do contacto diário com os sistemas actuais, registaram-se as dificuldades de comunicação entre etapas e o tempo de espera gerado pela excessiva intervenção manual.
- Entrevistas informais e sondagem de opinião: Para validar a viabilidade da solução proposta, realizaram-se diálogos técnicos e entrevistas informais com uma amostra de colaboradores internos, incluindo analistas de operação e gestores de equipa. Estas conversas focaram-se na identificação de padrões de falhas nos fluxos actuais e na aceitação de uma infra-estrutura comum para representação e acompanhamento uniforme dos processos. Os intervenientes confirmaram a pertinência da solução, destacando que a centralização da representação dos processos e dos seus ciclos de vida reduziria os problemas de fragmentação operacional e de visibilidade existentes no ambiente actual.

**b) Análise documental**

Foram consultados manuais de procedimentos, especificações funcionais e o próprio código-fonte dos sistemas legados. O acesso a estes repositórios foi viabilizado pela posição profissional do autor como desenvolvedor na instituição. Esta técnica justificou-se pela necessidade de compreender com precisão as regras de negócio e os mecanismos actualmente utilizados na tramitação dos processos bancários.

A análise documental permitiu o mapeamento fidedigno das regras de negócio que fundamentam a infra-estrutura baseada em grafos.

**c) Pesquisa bibliográfica e pesquisas web**

Foram utilizados artigos científicos, documentação técnica oficial e repositórios tecnológicos para fundamentar teoricamente o estudo e comparar abordagens modernas de integração de sistemas.

Adicionalmente, utilizaram-se repositórios do GitHub e documentações oficiais da Microsoft para analisar padrões de implementação de back-end, Arquitectura em camadas e

motores de regras aplicados em contextos globais de tecnologia financeira. Esta abordagem justificou-se pela necessidade de alinhar a solução proposta com práticas actuais de desenvolvimento de software e interoperabilidade de sistemas.

### 3.2.2 Análise e Modelação

**a) Mapeamento de fluxos**

A análise dos fluxos bancários foi realizada através do estudo das sequências de estados e transições existentes nos processos operacionais da instituição. Este procedimento teve como objectivo identificar dependências entre etapas, pontos de tramitação e relações entre intervenientes.

A partir deste levantamento, tornou-se possível definir os nós e arestas utilizados na representação dos ciclos de vida dos processos.

**b) Modelação baseada em Teoria de Grafos**

Após o levantamento dos fluxos operacionais, os processos foram formalizados matematicamente através de grafos direccionados. Esta abordagem justificou-se pela necessidade de criar uma representação uniforme capaz de descrever os estados, transições e regras de movimentação dos processos bancários.

Nesta modelação:

- os vértices representam áreas ou estados do processo;
- as arestas representam as possíveis tramitações entre etapas;
- os pesos representam o número de aprovações necessárias para cada transição.

A utilização da Teoria de Grafos permitiu estruturar os diferentes workflows sob um modelo comum de representação e gestão.

### 3.2.3. Validação Técnica

#### a) Testes de performance e carga

Para validar a robustez da solução proposta, foram realizados testes de performance e carga utilizando a ferramenta Apache JMeter. Esta escolha justificou-se pela necessidade de avaliar o comportamento da API sob condições de stress e garantir que a lógica baseada em grafos não comprometesse o desempenho operacional do sistema.

Os testes permitiram medir:

- tempos de resposta;
- estabilidade da API;
- consumo de recursos;
- capacidade de processamento simultâneo.

## 3.3. Materiais e Ferramentas

Para a materialização e validação da solução, foram utilizados os seguintes recursos tecnológicos. A escolha destas tecnologias justificou-se pelo facto de constituírem a *stack* tecnológica actualmente utilizada pela instituição em estudo, permitindo maior compatibilidade com o ambiente operacional existente e reduzindo o impacto de integração com os sistemas legados.

- **Desenvolvimento (Back-end):** Plataforma .NET 8 (C#) com *Entity Framework Core*, adoptando uma Arquitectura em camadas para garantir desacoplamento, organização estrutural e facilidade de manutenção.
- **Base de Dados:** SQL Server, utilizado para assegurar a persistência, integridade e consistência das relações entre *workflows*, processos e históricos de tramitação.
- **Contentorização e Hospedagem:** Docker, utilizado para isolar os serviços da aplicação e facilitar a escalabilidade, portabilidade e implementação em diferentes ambientes de infraestrutura.
- **Observabilidade:** Loki e Grafana, utilizados para centralização de *logs*, auditoria e monitorização métrica do comportamento da solução em tempo real.

- **Testes de Performance:** Apache JMeter, utilizado para simular carga e validar o desempenho da API em cenários de utilização simultânea.
- **Gestão de Código:** GitHub, utilizado para controlo de versões, organização do desenvolvimento e transparência da implementação técnica.

# 4. Modelação Da Solução

Após a compreensão teórica dos sistemas e da teoria de grafos, este capítulo descreve como esses conceitos foram aplicados para formalizar a estrutura dos fluxos bancários da instituição em estudo. A proposta baseia-se na unificação de diferentes fluxos sob um modelo matemático comum.

## 4.1. Estrutura Conceptual do Processo

Para a construção da solução, cada serviço bancário (abertura de conta, cartões, crédito) é modelado como uma unidade de informação que percorre um trajeto definido. Este percurso não é estático; ele evolui através de um ciclo de vida composto por três fases principais:

- **Criação:** O registo inicial no balcão onde se define a identidade do pedido.
- **Tramitação:** O percurso pelas áreas centrais para validações e pareceres.
- **Encerramento:** O estado final onde o processo é concluído e torna-se imutável.

## 4.2. Formalização Matemática dos Ciclos de Vida dos Processos

Essas fases permitem representar formalmente qualquer ciclo de vida de processo como um grafo direccionado onde:

- Os vértices representam as áreas por onde o processo passa;
- As arestas representam as tramitações possíveis entre essas áreas;
- O estado do processo é determinado pelo estágio no ciclo de vida em que ele se encontra;
- Cada aresta possui um peso (w), que corresponde ao número de pareceres (autorizações e/ou acções) necessários para a transição de uma área para a seguinte;
    - Em caso de retorno (recuo) do processo para uma etapa anterior, considera-se um peso uniforme para todas as arestas de retorno, uma vez que o esforço ou o critério de revisão é equivalente entre as áreas anteriores.

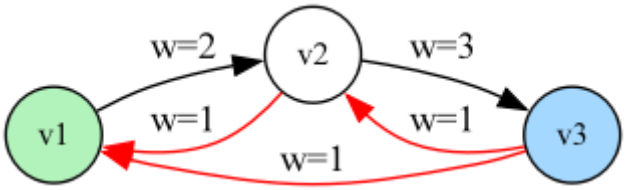
Formalmente, cada *workflow* pode ser descrito como um grafo $G=(V, E, W)$, em que:

- $V=\{v_1, v_2,\ldots v_n\}$ é o Conjunto de áreas (departamentos e/ou balcão) de tramitação;

- $E=\{(v_i, v_j) \mid v_i, v_j \in V\}$ representa as tramitações entre essas áreas, ativadas por pareceres.
- $W=\{ w_{ij} \}$ é o conjunto de pesos associados a cada tramitação, onde $w_{ij}$ indica o número de pareceres necessários para o processo deslocar-se de $v_i$ para $v_j$.

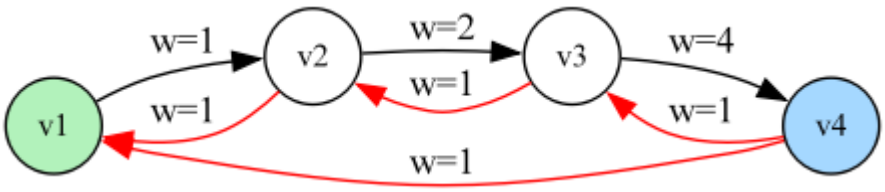
A Figura 1 e 2 ilustram exemplos de ciclos de vida de dois *workflows* fictícios:

*Figura 1 - Grafo representando o ciclo de vida de um processo*




*Fonte: Elaboração própria (2026)*

*Figura 2 - Grafo representando um ciclo de vida*




*Fonte: Elaboração própria (2026)*

A comparação entre ambos mostra que, embora o ciclo de vida e as áreas, representadas pelos vértices variem, a estrutura conceptual é equivalente: todos os ciclos de vida têm um início, representado pelo vértice destacado a verde, vértices centrais e um fim definitivo, representado pelo vértice destacado a azul, podendo retroceder entre etapas quando um parecer exige revisão.

Formalmente, portanto, todos os processos do ambiente podem ser vistos como instâncias de um mesmo modelo de grafo direcionado, diferenciando-se apenas pelos pontos envolvidos e regras específicas de tramitação.

No entanto, o problema surge do facto de não existir actualmente um mecanismo capaz de unificar e organizar esses ciclos de vida de forma integrada.
Cada processo é implementado e gerido isoladamente, o que impede a consolidação dos seus grafos individuais num modelo central que permita manipular os processos de forma uniforme.
Essa ausência de um ponto comum de representação e controle é o principal obstáculo técnico e conceptual à gestão eficiente do ambiente actual.

# 5. Implementação Da Solução

## 5.1. Arquitectura Geral da Solução

Como proposta de solução, apresenta-se uma infra-estrutura central responsável pela representação e gestão dos processos.
Este sistema funciona como o núcleo responsável pela representação uniforme e acompanhamento dos ciclos de vida dos processos, centralizando todas as operações comuns aos diferentes *workflows* e garantindo uniformidade no seu funcionamento.

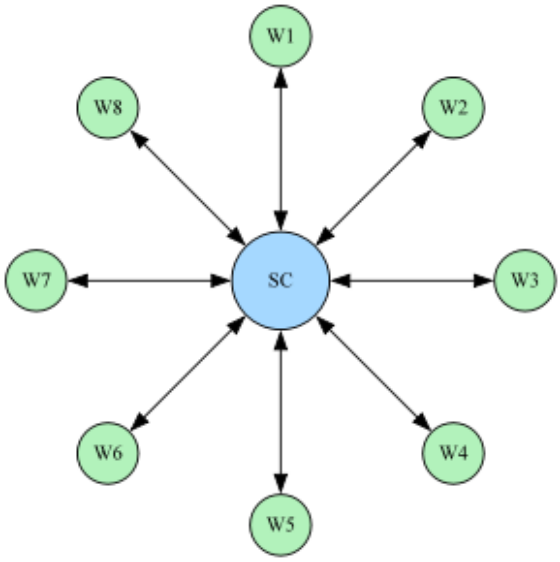
Através deste sistema central será possível:

- Definir o ciclo de vida dos processos para cada *workflow*;
- Registar e armazenar as informações comuns a todos os processos;
- Gerir todas as etapas do ciclo de vida dos processos (criação e tramitação)**;**
- Associar documentação relevante a cada processo;
- Coletar dados dos intervenientes em cada etapa;
- Emitir notificações automáticas aos intervenientes sobre eventos e actualizações dos processos.

Desta forma, cada sistema passaria a concentrar-se apenas nas suas responsabilidades específicas, enquanto todas as funcionalidades transversais seriam administradas por este sistema central.

A Arquitectura proposta, portanto, promove reutilização**,** padronização e redução de redundâncias entre *workflows*, facilitando a manutenção e evolução do ecossistema como um todo (ver Figura 3).

*Figura 3 - Arquitectura macro do sistema*




*Fonte: Elaboração própria (2026)*

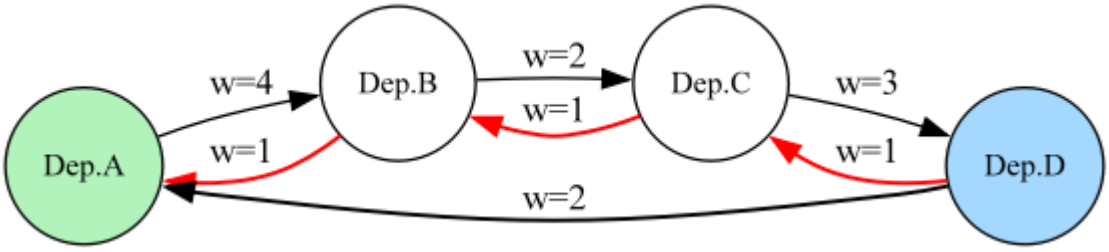
## 5.2. Estrutura de Dados e Entidades

Definido o problema, precisamos de uma estrutura de dados capaz de registar os ciclos de vida dos *workflows. Para isso, consideremos os pontos abaixo.*

### 5.2.1. Resolução Teórica Do Problema dos Ciclos de Vida Isolados

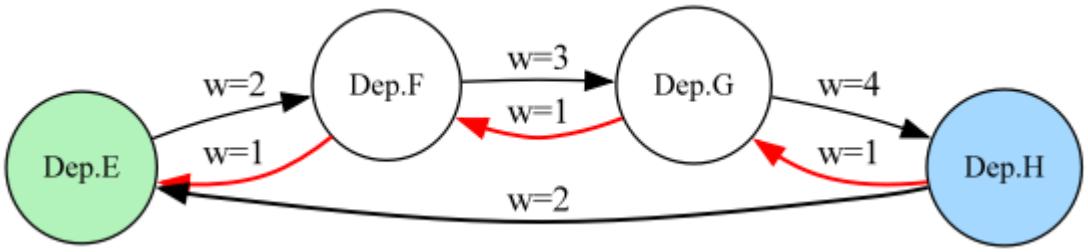
Para iniciar a análise, consideremos os ciclos de vida dos processos *x e y*, representados na forma de grafos direcionados (ver Figuras 4 e 5).

*Figura 4 - Grafo representante do ciclo de vida do workflow y*




*Fonte: Elaboração própria (2026)*

*Figura 5 - Grafo representante do ciclo de vida do workflow y*




*Fonte: Elaboração própria (2026)*

Esses fluxos podem ser expressos em formato tabular, a partir de uma lista de arestas. No contexto desta aplicação, cada linha representa uma tramitação entre áreas. A coluna responsável pelos pesos é denominada "*Approvals*", representando o número de pareceres exigidos para que um processo avance para o próximo interveniente ou retorne ao anterior.

*Tabela 1 - Representação tabular de grafos*

| Origin | Destination | Approvals | Direction |
|---|---|---|---|
| Dep. A | Dep. B | 4 | Avançar |
| Dep. B | Dep. A | 1 | Recuar |
| Dep. B | Dep. C | 2 | Avançar |
| Dep. C | Dep. B | 1 | Recuar |
| Dep. C | Dep. D | 3 | Avançar |
| Dep. D | Dep. A | 2 | Avançar |
| Dep. E | Dep. F | 2 | Avançar |
| Dep. F | Dep. E | 1 | Recuar |
| Dep. F | Dep. G | 3 | Avançar |
| Dep. G | Dep. F | 1 | Recuar |
| Dep. G | Dep. H | 4 | Avançar |
| Dep. H | Dep. E | 2 | Avançar |

*Fonte: Elaboração própria (2026)*

Como se pode observar, essa estrutura tabular permite rastrear o percurso de um processo ao longo do seu ciclo de vida.

Dessa forma, é possível armazenar qualquer ciclo de vida dentro da mesma estrutura de dados. No entanto, ainda não é possível identificar a qual *workflow* cada fluxo pertence, nem

em que posição do fluxo um processo específico se encontra.
Para resolver essa limitação, é necessário enriquecer a estrutura com informações adicionais.

### 5.2.2 Identificação do Processo

A identificação do processo ao qual cada ciclo de vida pertence pode ser feita através da adição de uma nova coluna que represente o identificador único desse *workflow*.

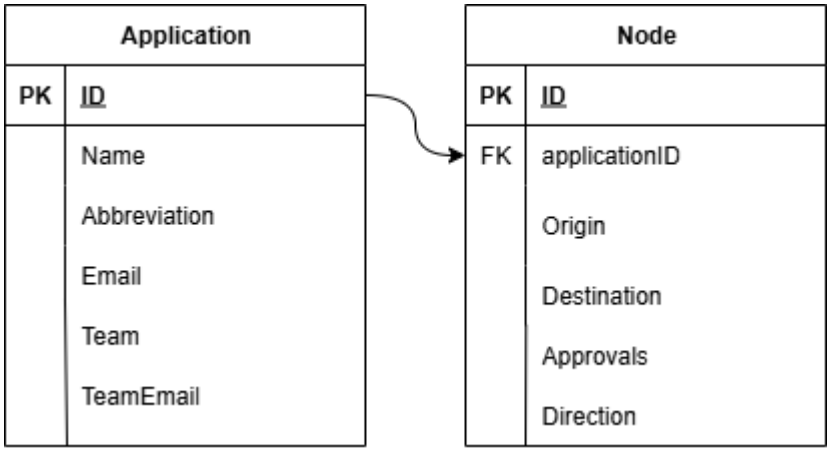
A partir desse identificador, torna-se viável associar metadados adicionais, como:

- O nome e o acrónimo do *workflow*;
- O endereço de e-mail (que será usado mais à frente para notificações de eventos associados aos processos);
- A equipa responsável pela sua manutenção e o seu email (para identificação das equipas responsáveis por cada *workflow* e para reportar erros);

Essas novas informações evidenciam a formação de uma estrutura relacional, onde cada *workflow* passa a ser representado por uma entidade.
Assim, começa-se a delinear um diagrama de entidades e relacionamentos (DER), que serve de base para a implementação de armazenamento de todos os fluxos neste sistema central.

Podemos então representar a entidade para armazenamento de *workflows* como "*Application*" e para o ciclo de vida com a entidade "*Node*" (ver Figura 6).

*Figura 6 - DER representando workflows e os seus ciclos de vida*



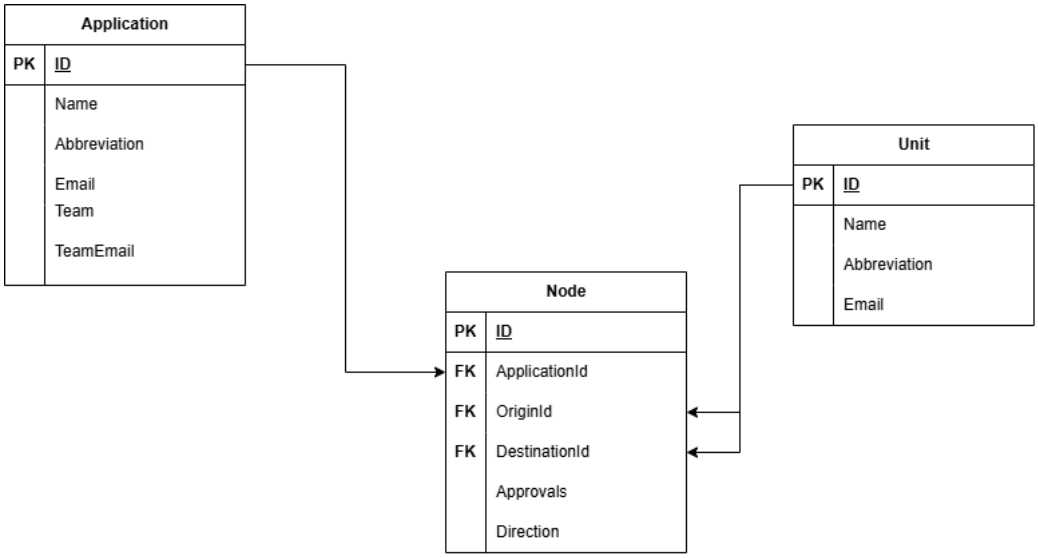

*Fonte: Elaboração própria (2026)*

### 5.2.3 Identificação Dos Intervenientes

Actualmente, na entidade "*Node"* cada linha representa as possibilidades de tramitação de um processo. E possui as colunas "*Origin"* e "*Destination"*, usadas para identificar os possíveis intervenientes para os quais o processo pode tramitar num determinado momento do seu ciclo de vida.

Propõe-se agora extrair essas informações para uma nova entidade dedicada aos intervenientes, que conterá dados adicionais para melhor identificação.
Chamemos a entidade de "*Unit*". Essa entidade será então referenciada por chave estrangeira na tabela "*Node"*, fortalecendo a relação entre os ciclos de vida e os intervenientes (ver Figura 7).

*Figura 7 - DER após adição da entidade para armazenamento dos intervenientes*




*Fonte: Elaboração própria (2026)*

### 5.2.4 Identificação dos Processos

Com o DER actual, já é possível armazenar qualquer ciclo de vida de um processo.
Dispomos de entidades para representar:

- Ciclo de vida do processo - *Node*;
- *Workflow - Application*;
- Intervenientes - *Unit*.

O próximo passo é associar um processo a essas entidades.
Para isso, precisamos registar o estado actual do processo e manter o histórico de todas intervenções que cada processo sofreu.
Introduzimos então duas novas entidades:

- *Process* – para armazenar o estado actual de um processo;
- *History* – para registar as intervenções passadas associadas a esse processo.

Na entidade *Process*, teremos:

- Colunas imutáveis:

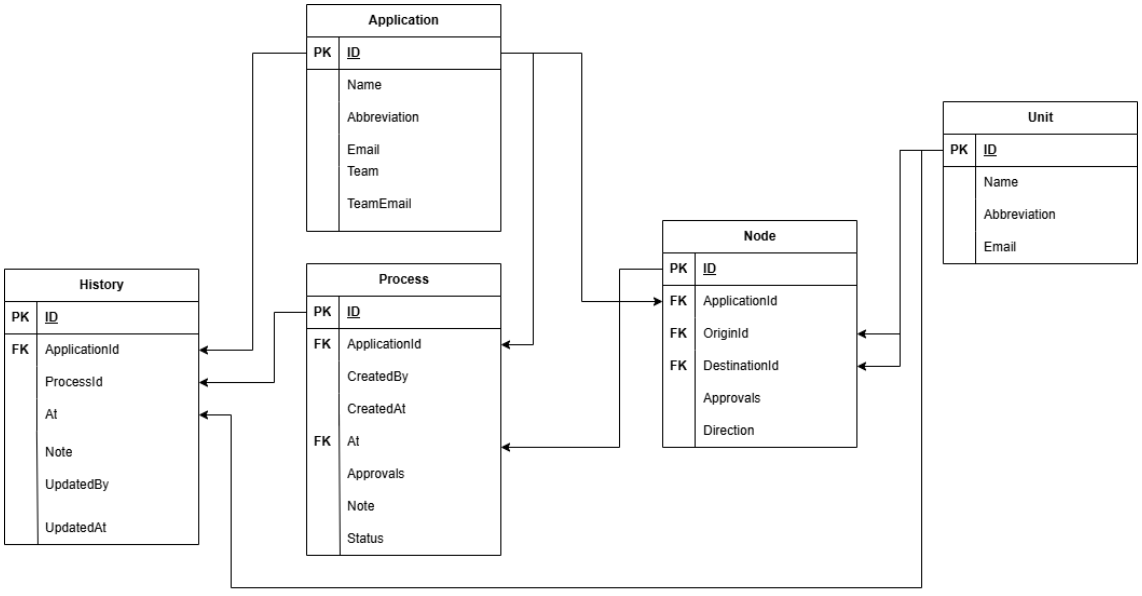
- *ApplicationId* - Identificador do *workflow;*
- *CreatedBy* e *CreatedAt* - Interveniente que criou o processo e a data de criação.

- Colunas mutáveis:

  - *At* - Posição actual no fluxo;
  - *Approvals* - Intervenções sofridas na localização actual em que se encontra (no *At*);
  - *Status* - Estado actual, determinado pelo estágio do seu ciclo de vida para indicar se foi criado, pendente, concluído ou cancelado;
  - Note para deixar anotações;

Na entidade *History*, serão armazenadas todas as colunas mutáveis de *Process*, juntamente com a coluna *ApplicationId*, e os registos do membro da área interveniente e da data de intervenção (ver Figura 8).

*Figura 8 - DER com a entidade history*




*Fonte: Elaboração própria (2026)*

Agora que definimos a estrutura capaz de armazenar os ciclos de vida de qualquer processo, bem como as intervenções que sofreu, o próximo passo é pensar em como essa estrutura será usada na prática.

Precisamos implementar métodos que permitam criar e tramitar um processo, garantindo que cada ação respeite o seu ciclo de vida.

Para isso, torna-se essencial identificar claramente os pontos de partida e de chegada de cada processo, ou seja, os nós de entrada e saída, bem como os critérios que definem se um processo pode ou não tramitar.

## 5.3 Requisitos do Sistema

O sucesso de um software que representa processos em grafos depende da sua capacidade de simplificar a realidade. Estes requisitos definem as fronteiras e as obrigações da solução para resolver a fragmentação operacional.

### 5.3.1 Requisitos Funcionais

Os requisitos funcionais descrevem as acções diretas que o sistema deve executar:

- **Unificação de Ciclos:** O sistema deve centralizar todos os ciclos de vida dos processos numa infra-estrutura central, eliminando a dispersão de dados entre áreas.
- **Gestão Configurável de Fluxos:** O utilizador deve poder adicionar, remover ou editar fluxos de forma configuracional, através de uma interface que não exija conhecimentos técnicos de programação.
- **Repositório de Documentos e Eventos:** O sistema deve gerir documentos e eventos associados a cada processo, disponibilizando-os para que sistemas adjacentes possam consultá-los de forma síncrona ou assíncrona.

### 5.3.2 Requisitos Não Funcionais

Estes requisitos referem-se às qualidades técnicas e limitações da solução:

- **Usabilidade para Não-Técnicos:** A interface de configuração deve ser intuitiva para que gestores de negócio consigam alterar regras de fluxo sem intervenção da equipa de TI.
- **Desempenho e Robustez:** A infra-estrutura central deve manter a estabilidade e baixos tempos de resposta sob carga, conforme validado nos testes com JMeter.
- **Observabilidade em Tempo Real:** O sistema deve exportar *logs* e métricas para o "Grafana"/"Loki", permitindo que a equipa técnica detete falhas na tramitação instantaneamente.

- **Escalabilidade Horizontal:** Graças à contentorização em Docker, a Arquitectura deve permitir o aumento da capacidade de processamento conforme o volume de agendamentos cresça.

## 5.4 Tramitação

Nesta secção, definimos como os processos se movem através do sistema. Utilizamos a entidade "Nodes" como base para estabelecer as regras que governam o tramitação do processo.

*Tabela 2 - Caso de estudo*

| Id | ApplicationId | OriginId | DestinationId | Approvals | Direction |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 2 | 2 | Avanço |
| 2 | 1 | 2 | 1 | 1 | Recuo |
| 3 | 1 | 2 | 3 | 3 | Avanço |
| 4 | 1 | 3 | 2 | 1 | Recuo |
| 5 | 1 | 3 | 1 | 4 | Avanço |
| 6 | 2 | 4 | 5 | 0 | Avanço |
| 7 | 2 | 5 | 4 | 1 | Recuo |
| 8 | 2 | 5 | 6 | 2 | Avanço |
| 9 | 2 | 6 | 5 | 1 | Recuo |
| 10 | 2 | 6 | 4 | 3 | Avanço |

*Fonte: Elaboração própria (2026)*

### 5.4.1 Componentes do Fluxo: Nós e Arestas

O ciclo de vida de um processo é composto por pontos de parada (Nós) e caminhos de ligação (Arestas).

- **Nós (Intervenientes):** Representam os intervenientes (áreas ou departamentos). O conjunto de nós é formado pela união de todos os identificadores de origem e destino registados.

  *Definição formal:*

  $$V = \{ v \in \mathbb{N} \mid v \in OriginId \cup v \in DestinationId \}$$

- **Arestas (Tramitações):** São as conexões que permitem ao processo ir de um interveniente a outro.

  *Definição formal:*

  $$A = \{(o,d) \in V \times V \mid \text{existe um registo Nodes com OriginId}=o \text{ e DestinationId}=d\}$$

### 5.4.2 Pontos de Entrada e Saída

Para o sistema gerir o fluxo, ele precisa identificar onde o processo começa e onde termina:

1. **Nó de Entrada ( $V_{in}$ ):** É o valor que aparece apenas uma vez como origem (OriginId) num fluxo específico. Pode ser:
   - **Automático:** O processo avança sem necessidade de aprovação humana (quando Approvals = 0).
   - **Manual:** Exige uma ação inicial para começar a tramitar.
2. **Nó de Saída $V_{out}$:** É o destino final antes do encerramento. Identifica-se como o valor que aparece apenas uma vez na coluna de destino (DestinationId).

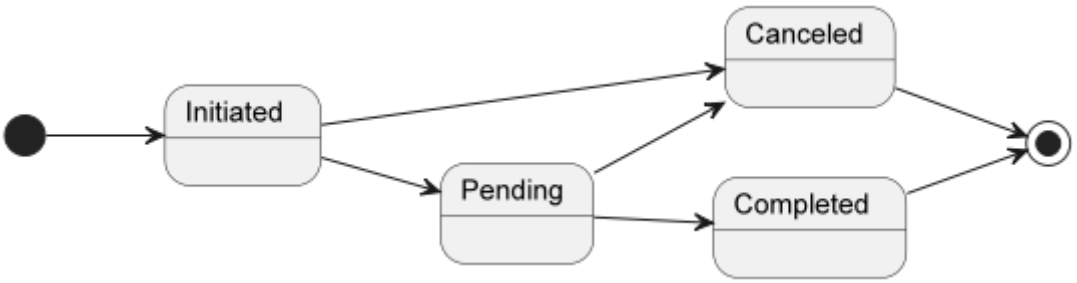
### 5.4.3 Estados do Processo

O estado actual (*P.Status*) define se um processo pode ou não se mover. Adotamos quatro categorias:

- Initiated: Registado, mas parado na origem.
- Pending: Em trâmite, aguardando que o contador de aprovações *P.Approvals* atinja o requisito do nó (*Node.Approvals*).
- Concluded: Chegou ao nó de saída.
- Cancelled: Interrompido antes do fim.

Desta forma durante o ciclo de vida de um processo ele muda de estados segundo o diagrama na Figura 8

*Figura 9 - Fluxo dos estados de um processo*



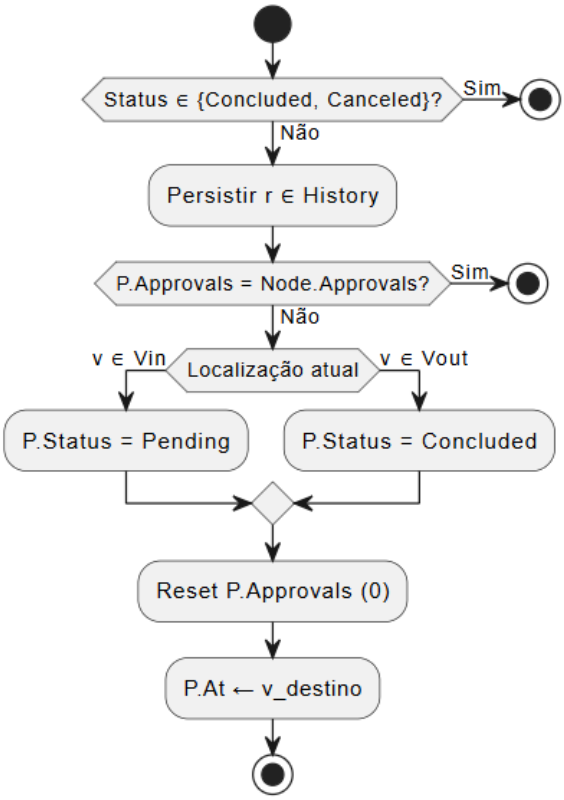

*Fonte: Elaboração própria (2026)*

### 5.4.4 Regras de Movimentação

A tramitação é disparada por uma ação que gera um Registo de Histórico ($r \in History$). Ela ocorre em duas direções:

**Avanço:** Ocorre quando P.Approvals ≥ Node.Approvals. O sistema actualiza a localização (P.At ← $V_{destino}$) e o estado para *Pending* ou *Concluded* (ver Figura 10).

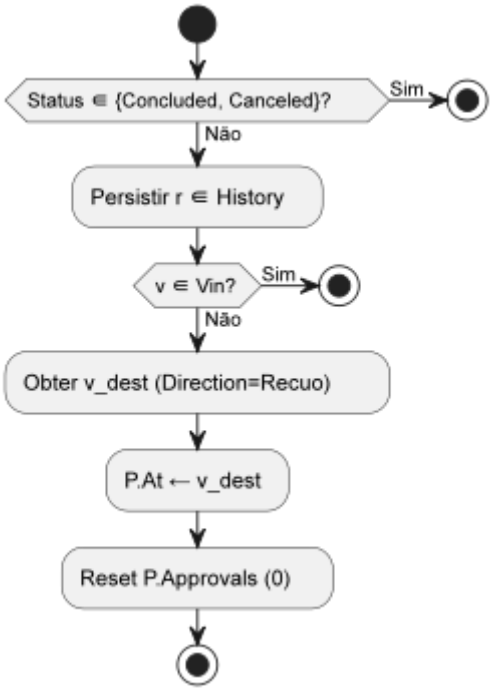
*Figura 10 - Diagrama de actividade do fluxo de aprovação de um processo*




*Fonte: Elaboração própria (2026)*

**Recuo:** Utilizado para correções. O sistema identifica a rota de retorno $V_{dest}$ através da aresta onde *Direction* = Recuo, reinicia os contadores (P.Approvals(0)) de aprovação no destino para garantir uma nova revisão completa (ver Figura 11).

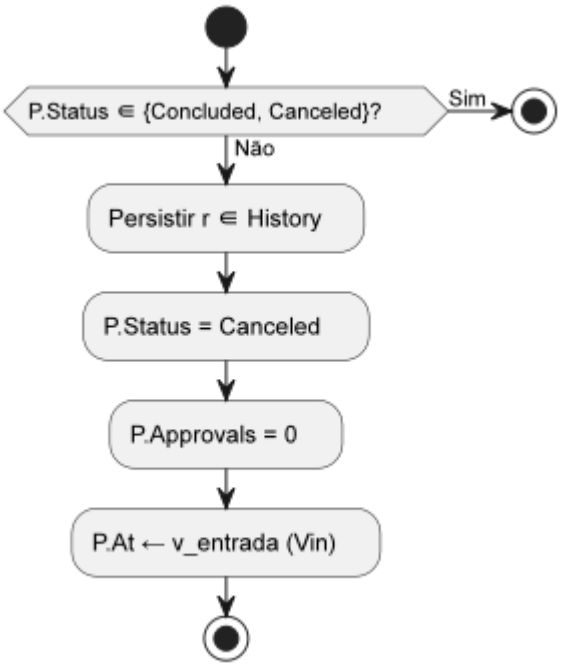
*Figura 11- Diagrama de actividade do fluxo de devolução de um processo*




*Fonte: Elaboração própria (2026)*

**Cancelamento**: O sistema identifica o nó de entrada, $V_{in}$ e avança para este nó alterando o estado para *Cancelled*, e salvando o histórico (ver Figura 12).

*Figura 12 - Fluxo de cancelamento de um processo*




*Fonte: Elaboração própria (2026)*

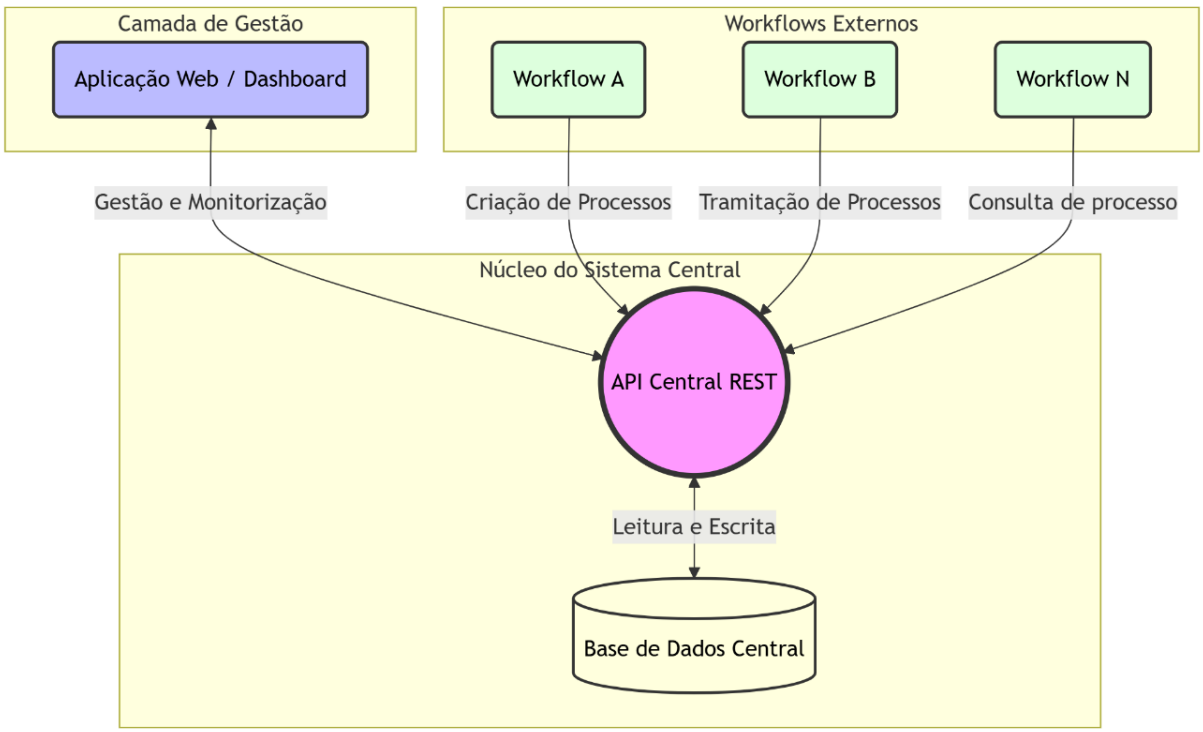
# 6. Desenvolvimento Da Solução

O desenvolvimento da solução focou na criação de um ecossistema centralizado, capaz de representar e acompanhar diversos ciclos de vida de processos de forma agnóstica à tecnologia dos sistemas externos. Esta secção descreve a Arquitectura do sistema, detalhando desde a visão macroscópica até à implementação técnica dos seus componentes internos.

## 6.1. Arquitectura Geral do Ecossistema

A Arquitectura geral do ecossistema baseia-se num modelo de comunicação centralizado, onde a *API* central atua como núcleo de representação e gestão dos processos e o mediador de persistência. A solução foi desenhada para permitir que sistemas externos, com lógicas e contextos distintos, deleguem a gestão dos seus ciclos de vida a um núcleo comum. A organização macroscópica deste ambiente e as interações entre os seus principais componentes estão representadas na Figura 13.

*Figura 13- Arquitectura macro do sistema*




*Fonte: Elaboração própria (2026)*

## 6.2. *Stack* Tecnológica

O desenvolvimento da *API* Central foi sustentado pela plataforma ".NET 8", aproveitando a sua performance e suporte a Arquitecturas modulares. Para estender as funcionalidades base e garantir a robustez do sistema, foram integradas diversas bibliotecas especializadas, listadas na Tabela 3.

*Tabela 3 - Dependências (Nuget's) da API*

| **Ferramenta (NuGet)** | **Versão** | **Função** |
|---|---|---|
| Microsoft.Extensions.Hosting.Abstractions | 9.0.0 | Gerência de Ciclo de Vida |
| Serilog | 4.2.0+ | Auditoria e Logs |
| Serilog.Sinks.Grafana.Loki | 2008.3.1 | Observabilidade |
| MailKit | 4.13.0 | Comunicação (E-mail) |
| Microsoft.AspNetCore.SignalR | 1.2.2000 | Comunicação em Tempo Real |
| Microsoft.EntityFrameworkCore.SqlServer | 9.0.6 | Gerência de Dados |
| Microsoft.EntityFrameworkCore.Tools / Design | 9.0.6 | Ferramentas de Dados |
| Microsoft.AspNetCore.Http.Features | 5.0.17 | Infraestrutura HTTP |
| Asp.Versioning.Http / ApiExplorer | 8.1.2000 | Gestão de Versões |
| FluentValidation (e extensões DI) | 11.6.2000 | Validação de Dados |
| Microsoft.AspNetCore.Authentication.JwtBearer | 8.0.18 | Segurança |
| Microsoft.AspNetCore.OpenApi | 8.0.13 | Documentação Técnica |
| Swashbuckle.AspNetCore | 2006.6.2 | Interface Swagger |

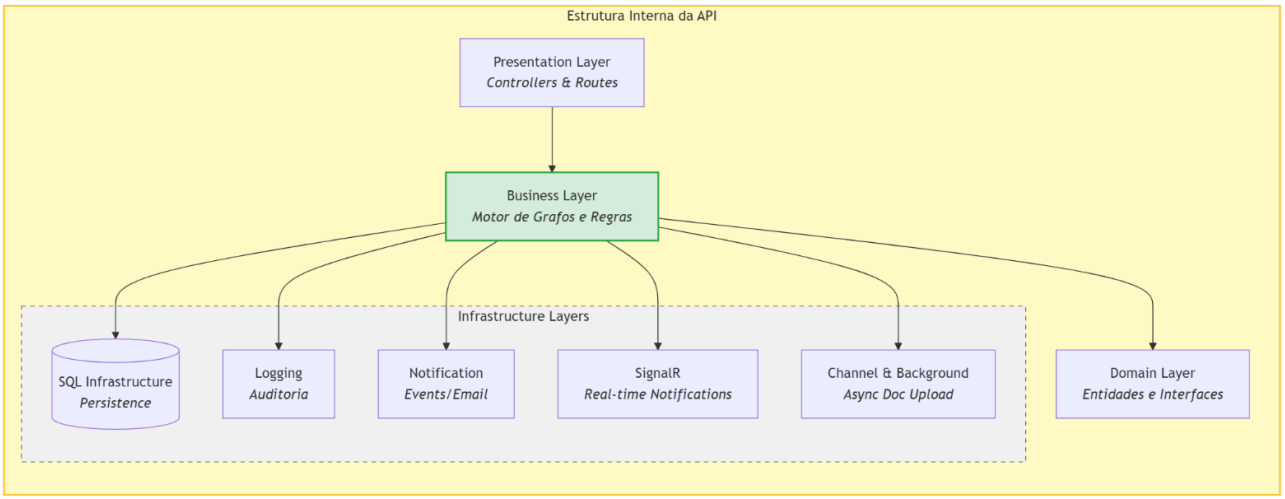
*Fonte: Elaboração própria (2026)*

## 6.2 Arquitectura Detalhada da *API* Central

A API Central constitui o núcleo operacional da solução, sendo responsável por processar a lógica dos grafos e garantir a integridade da comunicação entre os diversos *workflows*. Para assegurar que esta centralização não resulte em um sistema rígido ou de difícil manutenção, a sua estrutura interna foi desenhada com foco no desacoplamento e na alta disponibilidade de serviços.

A organização dos componentes internos e o fluxo de dependências entre os serviços de suporte e o núcleo de representação e gestão dos processos estão ilustrados na Figura 14.

*Figura 14 - Arquitectura em camadas da API REST*




*Fonte: Elaboração própria (2026)*

### 6.2.1 Organização em Camadas e Responsabilidades

A *API* Central foi estruturada sob o modelo de Arquitectura em camadas para garantir que cada componente possua uma função única e isolada. Esta organização, fundamentada nos princípios de Barskiy (2023) para o ASP.NET Core 8, permite que o sistema seja resiliente a falhas e fácil de evoluir.

a) **Camada de Apresentação (*Presentation Layer*):** esta é a porta de entrada da *API*. Nela, os controladores (*Controllers*) gerem as rotas e os pedidos *HTTP*. A sua única responsabilidade é receber dados, validá-los e devolver uma resposta ao utilizador.

Conforme defende Barskiy (2023), esta camada deve ser mantida "magra", delegando a lógica de decisão para os serviços de negócio.

b) **Camada de Negócio (*Business Layer*):** é o núcleo onde o infra-estrutura baseada em grafos opera. Esta camada traduz as regras teóricas de tramitação em código funcional. Caso um processo deva avançar, recuar ou ser cancelado, a lógica é processada neste nível. Ao isolar estas regras, garante-se que o ciclo de vida dos processos não seja afetado por mudanças na base de dados ou na interface visual.

c) **Camada de Domínio (*Domain Layer*):** contém as definições essenciais do sistema, nomeadamente as entidades e as interfaces. Serve como o contrato que une as camadas. Barskiy (2023) enfatiza que o domínio deve ser independente de tecnologia, representando apenas a essência do negócio — neste caso, a estrutura de nós $V$ e arestas $E$ do *workflow*.

d) **Camadas de Infraestrutura (*Infrastructure Layers*):** são os braços operacionais da API que implementam as tarefas técnicas de suporte à lógica de negócio:

- **SQL *Infrastructure*:** executa a persistência física dos dados e garante a integridade transacional;
- ***Logging Infrastructure*:** regista a actividade do sistema para auditoria e monitorização via Grafana;
- ***Notification Infrastructure*:** gerem a comunicação de eventos e actualizam o *dashboard* em tempo real;
- ***Channel* & *Background Service*:** processa operações pesadas, como o *upload* de documentos, de forma assíncrona para manter a celeridade da *API*.

Esta separação rigorosa de responsabilidades assegura que a complexidade de gerir múltiplos processos não sobrecarregue o sistema, permitindo que cada parte seja testada e modificada sem impactar o todo (BARSKIY, 2023).

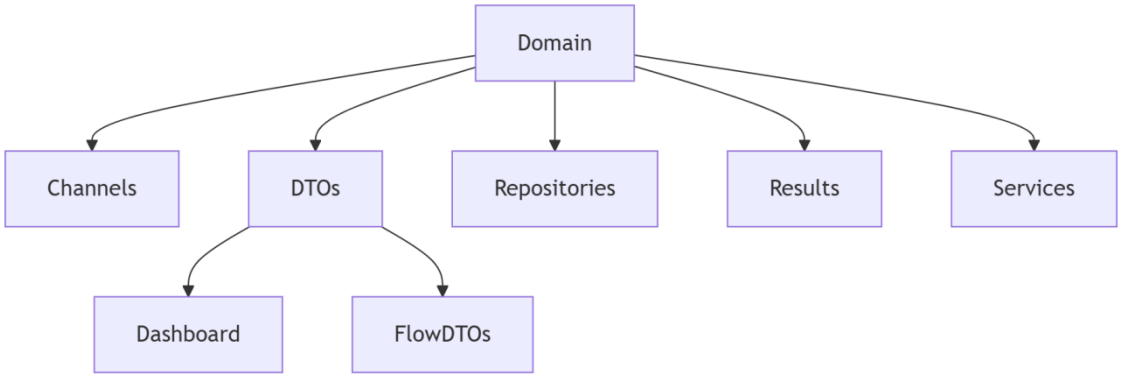
### 6.2.2 Estrutura de ficheiros

Uma boa Arquitectura deve ser óbvia. Para esta *API*, organizei os ficheiros de forma a separar a lógica de negócio das ferramentas técnicas. Se precisarmos de mudar a base de dados ou o sistema de *e-mail* no futuro, o "coração" do sistema (o Domínio) permanecerá intacto.

A estrutura de pastas reflete esta separação. Em vez de um bloco único de código, a solução divide-se em camadas com responsabilidades claras (ver Figuras 16, 17, 18, 19).

**Domain:**

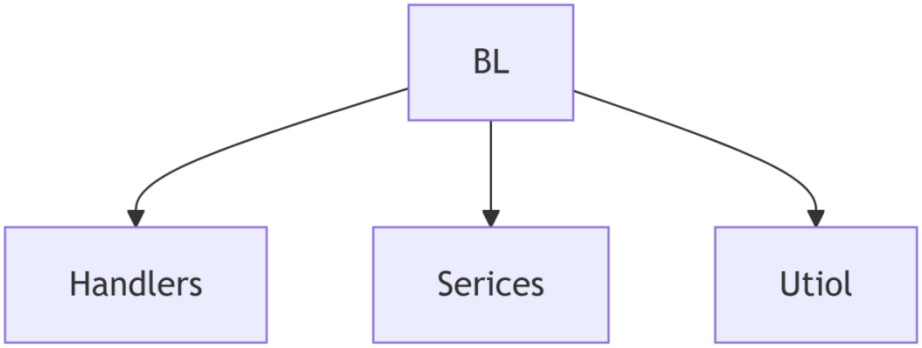
*Figura 15 - Estrutura de directórios da camada de contratos*




*Fonte: Elaboração própria (2026)*

**BL:**

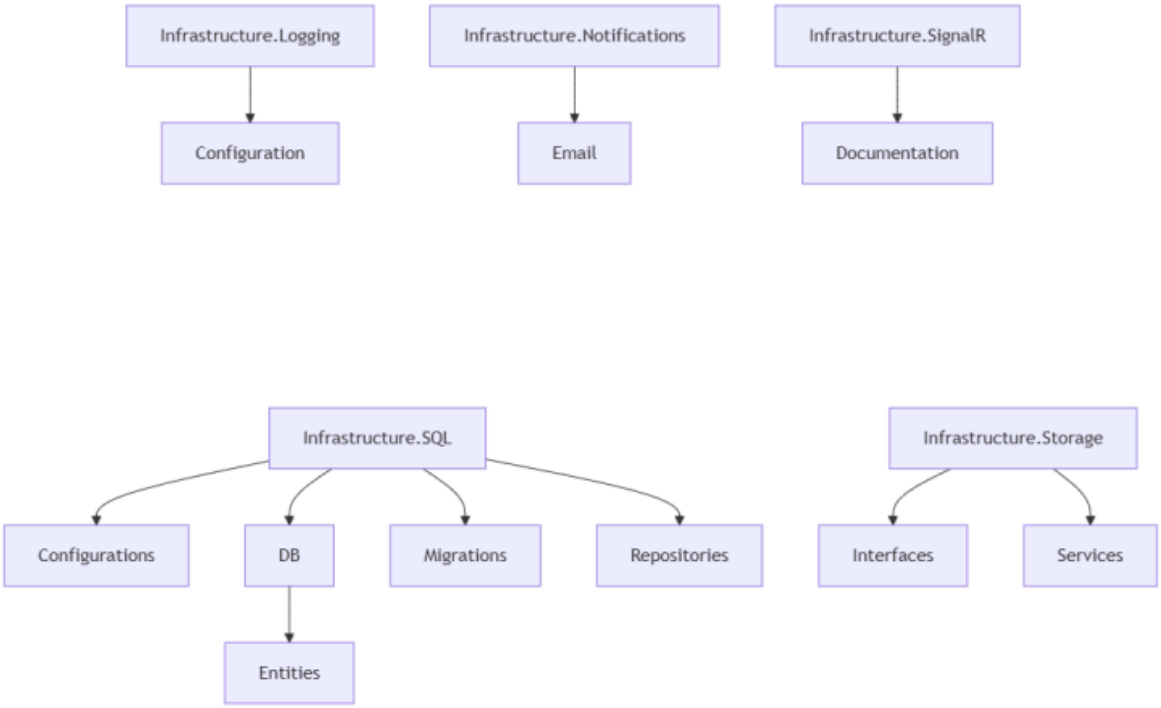
*Figura 16 - Estrutura de directórios da camada de regra de negócios*




*Fonte: Elaboração própria (2026)*

***Infrastructure*:**

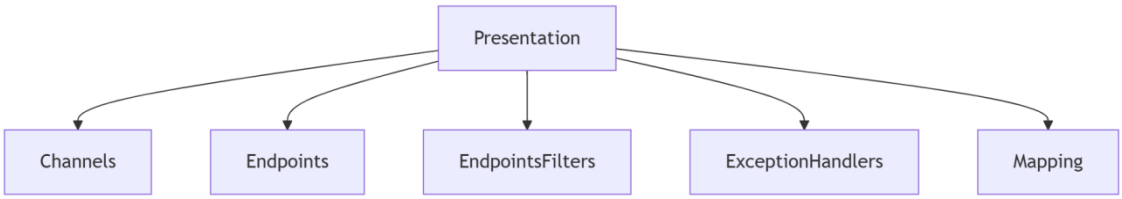
*Figura 17 - Estrutura de directórios das camadas de infraestrutura*




*Fonte: Elaboração própria (2026)*

***Presentation*:**

*Figura 18 - Estrutura de directórios da camada da aplicação*




*Fonte: Elaboração própria (2026)*

### 6.2.2 Contratos de Comunicação e *Endpoints*

Os contratos de comunicação definem as regras de interação entre a *API* e os sistemas externos. Para garantir a interoperabilidade, utiliza-se o padrão *REST* com troca de dados em formato JSON e *Multipart Form-Data* para ficheiros. Conforme preconiza Barskiy (2023), os *endpoints* foram desenhados para serem intuitivos, utilizando verbos HTTP padronizados que representam acções claras sobre os recursos do sistema.

a) **Gestão de Aplicações e Definição de Fluxos (*Applications & Flows*):** este grupo permite a configuração administrativa da solução.

- *POST /applications*: cria uma nova entidade de aplicação e persiste o seu grafo de nós e arestas;
- *GET /applications*: lista todas as aplicações registadas;
- *GET /applications/{id}*: recupera os detalhes de uma aplicação específica;
- *PUT /applications/{id}*: actualiza os metadados da aplicação;
- *DELETE /applications/{id}*: remove uma aplicação do sistema;
- *GET /flows/{applicationId}*: recupera a estrutura lógica do grafo associado a uma aplicação.

b) **Operações de Processo e Tramitação (*Processes*):** constitui o núcleo funcional para a interação com sistemas externos e gestão do ciclo de vida dos processos.

- *POST /processes*: inicia um novo processo vinculado a um ciclo de vida, aceitando documentos via *multipart/form-data*;
- *GET /processes*: lista o estado de todos os fluxos de processos com suporte a paginação e filtros;
- *PATCH /processes/approve*: avança o estado do processo, validando a transição no infra-estrutura baseada em grafos;
- *PATCH /processes/{processId}/return*: retrocede o processo para uma fase anterior;
- *PATCH /processes/{processId}/cancel*: interrompe a execução do processo;
- *POST /processes/upload*: envia ficheiros para processamento assíncrono via *background services*;
- *GET /processes/docs*: recupera a listagem de documentação em lotes;
- *GET /processes/docs/download*: permite o descarregamento de um documento específico através do seu identificador.

c) **Gestão de Unidades Organizacionais (*Units*):** gere as entidades que interagem com os processos.

- *POST /users/login*
- *POST /users/logout*

**d) Autenticação e Monitorização (*Users & Dashboard*):**

- **Segurança:** os *endpoints POST /users/login e POST /users/logout* gerem a sessão baseada em *cookies*;
- ***Dashboard*:** inclui uma série de rotas *GET* (como */dashboard/totals*, */dashboard/status/units, /dashboard/duration/workflows)* que fornecem agregados estatísticos e métricas de desempenho para visualização.

**e) Semântica das Respostas e Códigos de Estado:** a *API* comunica o resultado das operações através de códigos *HTTP* semânticos (BARSKIY, 2023). Os sucessos são confirmados por 200 *OK* (consultas) e 201 *Created* (criação). Falhas de validação ou violações das regras do grafo resultam em 400 *Bad Reques*t. Acessos indevidos retornam 401 *Unauthorized*, enquanto falhas de servidor são sinalizadas com 500 *Internal Server Error*.

**f) Estrutura de Dados (*Requests* e *Responses*):** a comunicação utiliza modelos fortemente tipados. As requisições exigem objetos *JSON* ou formulários binários, enquanto as respostas fornecem o objeto processado ou uma mensagem descritiva de erro. Esta previsibilidade assegura que a evolução da *API* não comprometa as integrações existentes (BARSKIY, 2023).

### 6.2.3 Segurança e Autenticação

A segurança da solução proposta é dividida em dois domínios distintos, garantindo a proteção da gestão do sistema sem interferir na flexibilidade dos fluxos de trabalho.

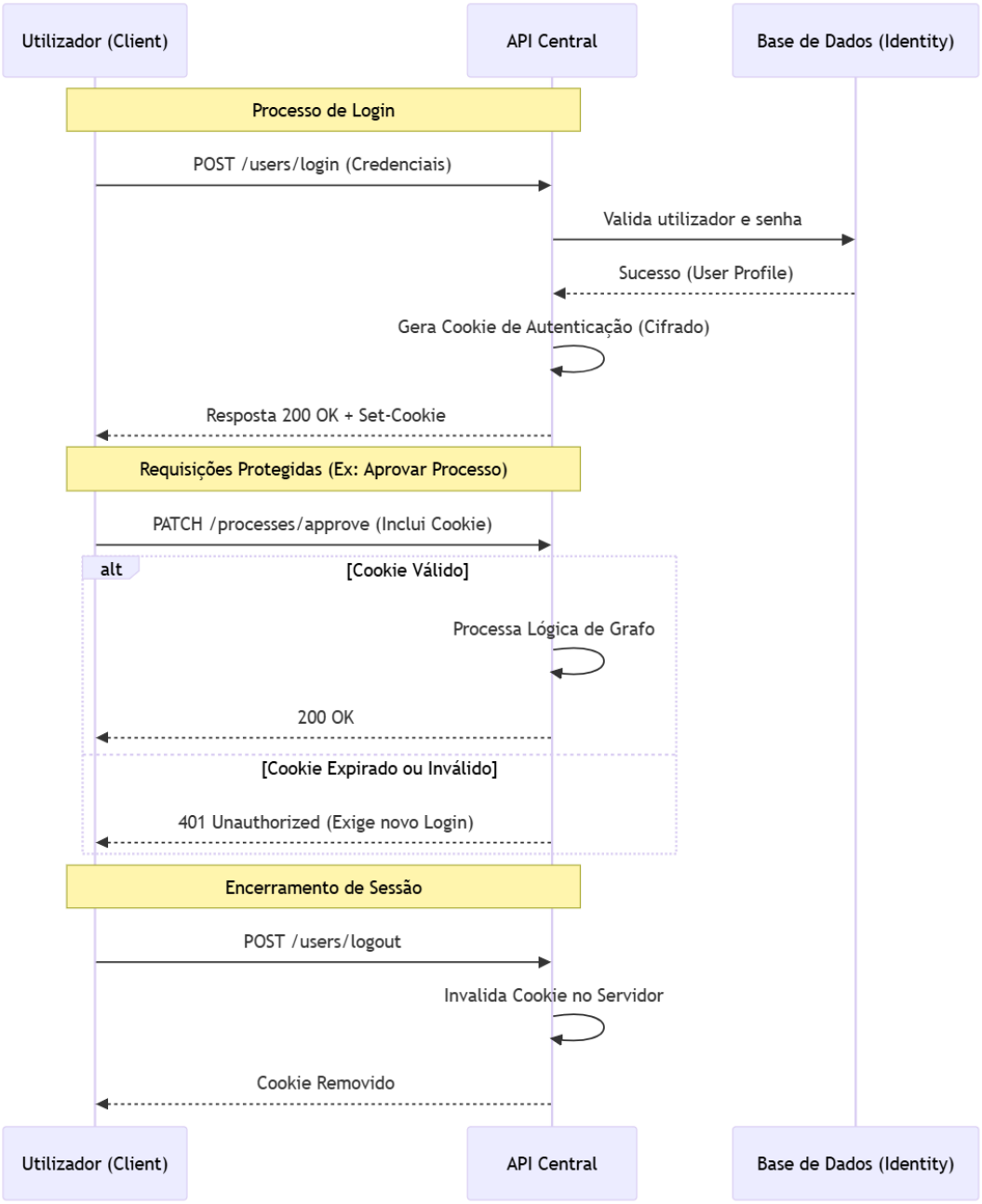
**Autenticação da *Dashboard* Administrativa**

Para o painel de controlo (*Dashboard*) da *API*, foi implementado um sistema de autenticação e autorização nativo baseado em *cookies* seguros, seguindo as recomendações de Barskiy (2023) para o ASP.NET Core 8. Este mecanismo garante que apenas administradores autorizados possam visualizar métricas e gerir instâncias de processos, exigindo uma sessão ativa para acesso à interface visual. A Figura 19 ilustra este fluxo específico de validação de credenciais e gestão de sessão.

**Escopo dos *Workflows* e Integração Externa**

Quanto aos *endpoints* que executam a lógica de negócio dos *workflows*, a infra-estrutura integradora foi desenhada para ser agnóstico em relação à autenticação do cliente final. Isto significa que a validação de identidade para o consumo das operações bancárias (como *Create* ou *Approve*) recai sobre as camadas de integração já existentes na instituição. O desenvolvimento foca-se na representação uniforme e gestão integrada dos processos e integridade dos processos, deixando os mecanismos de segurança perimetral e autorização de utilizadores externos fora do escopo deste trabalho.

*Figura 19 - Diagrama de sequência descrevendo o fluxo de autenticação por cookies*




*Fonte: Elaboração própria (2026)*

**a) Processo de Autenticação:** o utilizador envia as suas credenciais para o *endpoint* */users/login*. Após a verificação na base de dados, a API emite um *cookie* cifrado. Este ficheiro é armazenado no cliente com a propriedade *HttpOnly*, que impede o acesso via *scripts* maliciosos, e *Secure*, garantindo a transmissão apenas através de canais HTTPS (BARSKIY, 2023).

**b) Autorização e Validação de Pedidos:** após o acesso inicial, cada pedido subsequente inclui automaticamente o *cookie*. A *API* valida a integridade e a validade temporal do mesmo antes de processar qualquer lógica de negócio. Caso o *cookie* tenha expirado, a *API* interrompe o acesso e retorna o código 401 *Unauthorized*, forçando a reautenticação do utilizador.

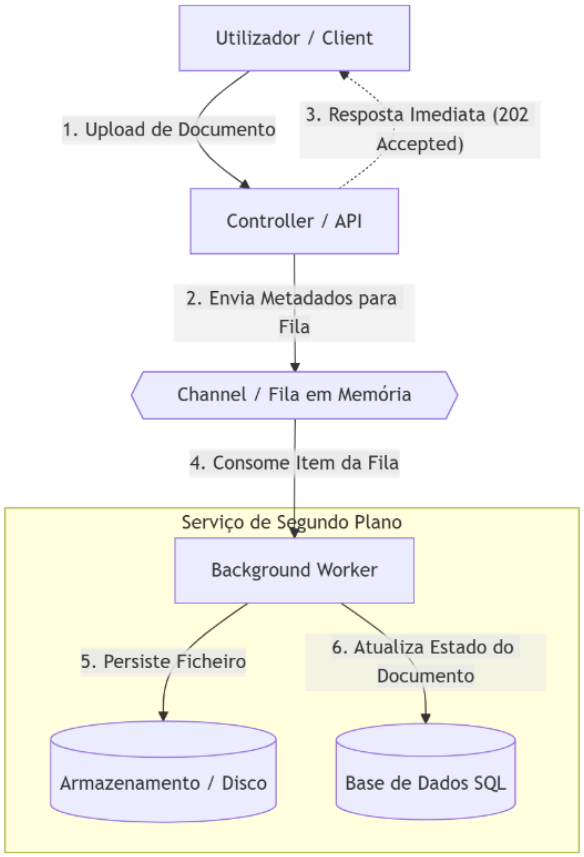
**c) Segurança dos *Endpoints*:** a proteção é global, o que significa que nenhum recurso da *API*, desde a criação de unidades até à consulta de *dashboards*, pode ser acedido sem uma identidade confirmada. Esta abordagem centralizada facilita a auditoria, uma vez que todas as acções na infra-estrutura baseada em grafos, ficam vinculadas ao utilizador detentor da sessão (BARSKIY, 2023).

### 6.2.4 Processamento Assíncrono de Documentação

O carregamento de documentação é uma das operações mais pesadas no ecossistema de ciclos de vida de processos. Para evitar que a API fique bloqueada durante o processamento de ficheiros de grandes dimensões, implementou-se um modelo de processamento assíncrono baseado em *Background Services*. Esta abordagem, fundamentada por Barskiy (2023), garante que a experiência do utilizador permaneça fluida, delegando tarefas demoradas para uma execução em segundo plano.

A Figura 20 detalha a orquestração entre o pedido do utilizador, a fila em memória (*Channel*) e o serviço de processamento

*Figura 20 - Fluxo de upload de ficheiros*




*Fonte: Elaboração própria (2026)*

### 6.2.5 Observabilidade e Auditoria

A observabilidade é a capacidade de compreender o estado interno de um sistema a partir dos dados que ele gera. Para uma *API* que acompanha múltiplos ciclos de vida de processos, a capacidade de rastrear falhas e auditar eventos em tempo real é vital. A solução utiliza uma infraestrutura de *logging* centralizada, isolada da aplicação principal através de contentores Docker, garantindo que a monitorização não consuma recursos diretos da infra-estrutura baseada em grafos.

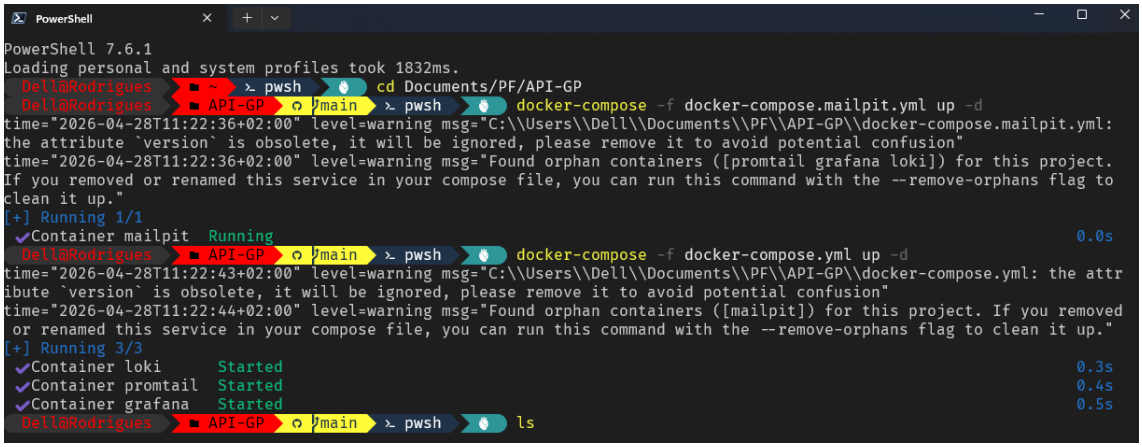
**a) Infraestrutura de Monitorização:** a implementação utiliza o ecossistema Grafana (ver Figura 21) para a gestão de eventos. Conforme recomenda Barskiy (2023), o uso de

ferramentas de *logging* estruturado permite que a equipa técnica identifique gargalos e erros de tramitação antes que estes impactem o utilizador final. A estrutura é composta por três componentes principais:

- **Grafana Loki:** atua como o banco de dados otimizado para armazenamento de *logs*;
- **Promtail:** serve como o agente de transporte, capturando os ficheiros de *log* gerados pela *API* no servidor e enviando-os para o Loki;
- **Grafana Dashboard:** fornece a interface visual para consulta e análise dos dados.

**b) Orquestração via Docker:** para assegurar a portabilidade e a facilidade de implementação, toda a pilha de observabilidade é gerida via *Docker Compose*. Esta configuração permite que os serviços de monitorização (Loki, Promtail e Grafana) sejam executados em contentores isolados, com volumes mapeados para garantir a persistência dos dados de auditoria, mesmo após o reinício dos serviços.

*Figura 21 - Docker containers*

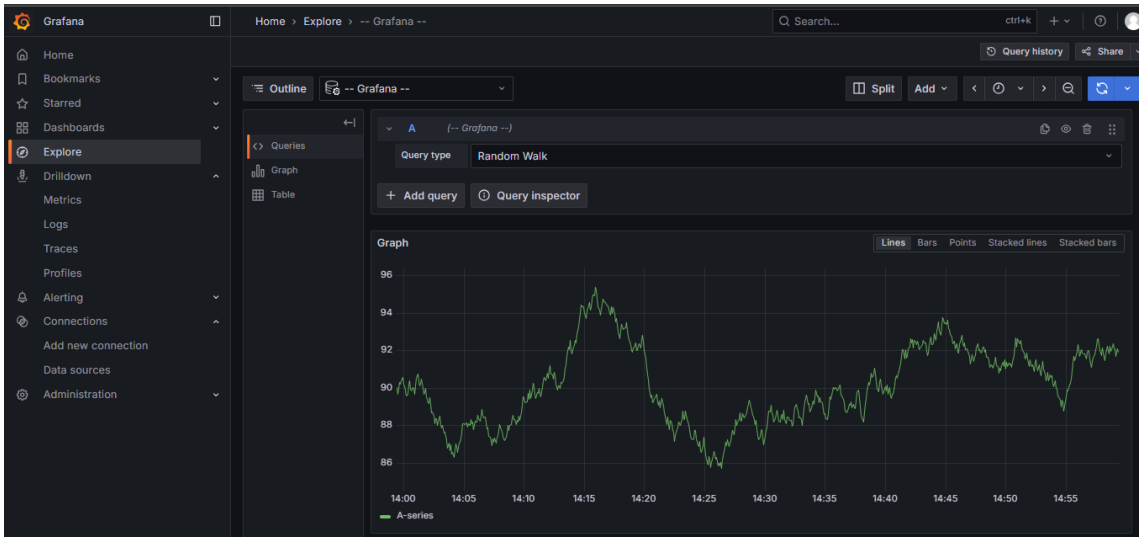



*Fonte: Recorte terminal executando docker containers*

**c) Auditoria e Diagnóstico de Erros:** a *API* regista eventos críticos, como tentativas de autenticação, alterações de estado no grafo e falhas no processamento de documentos. Barskiy (2023) enfatiza que uma estratégia de *logging* eficaz deve permitir a correlação de eventos. Com o Grafana, é possível filtrar *logs* por nível de severidade (ex: *Error*, *Warning*, *Info*), facilitando a identificação imediata de uma falha num nó específico do *workflow*.

**d) Disponibilidade de Dados:** a interface do "Grafana" está configurada para fornecer métricas de saúde do sistema em tempo real. Através de painéis intuitivos, os administradores podem monitorizar o volume de requisições e a taxa de sucesso das operações de tramitação, garantindo que o sistema opere dentro dos parâmetros de desempenho esperados (BARSKIY, 2023).

*Figura 22 - Dashboard do grafana captando os logs da nossa API*

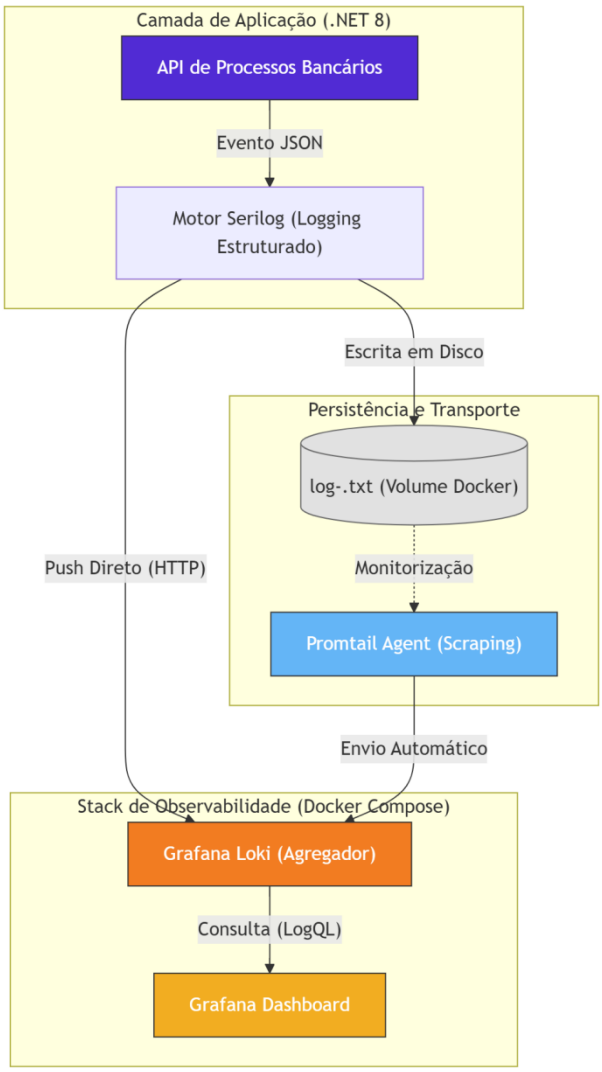



*Fonte: Recorte dashbord de logs do grafana*

**e) Implementação Técnica com Serilog:** Para a recolha e organização dos dados, a API utiliza a biblioteca Serilog (instalada via pacotes NuGet). Esta ferramenta permite o chamado "logging estruturado", onde cada evento é gravado em formato JSON, contendo campos específicos como o ID do processo e o identificador do nó do grafo. No ficheiro de configuração da API (appsettings.json), foram definidos dois destinos para os dados: o File Sink, que guarda uma cópia física em ficheiros .txt para segurança local, e o Loki Sink, que envia os dados via protocolo HTTP para visualização imediata no Grafana.

**f) Arquitectura de Fluxo de Dados e Resiliência:** O funcionamento desta camada segue um fluxo desenhado para evitar a perda de informação, conforme ilustrado na Figura 23. Quando ocorre uma ação na API, o log é gerado. Caso exista alguma instabilidade na ligação direta ao Loki, o Promtail atua como um agente de segurança: ele monitoriza os ficheiros gravados no disco e envia-os automaticamente assim que a ligação é restabelecida. Este processo garante que o rasto de auditoria de cada processo bancário seja preservado mesmo em condições de falha de rede.

*Figura 23 - Arquitectura de observabilidade*

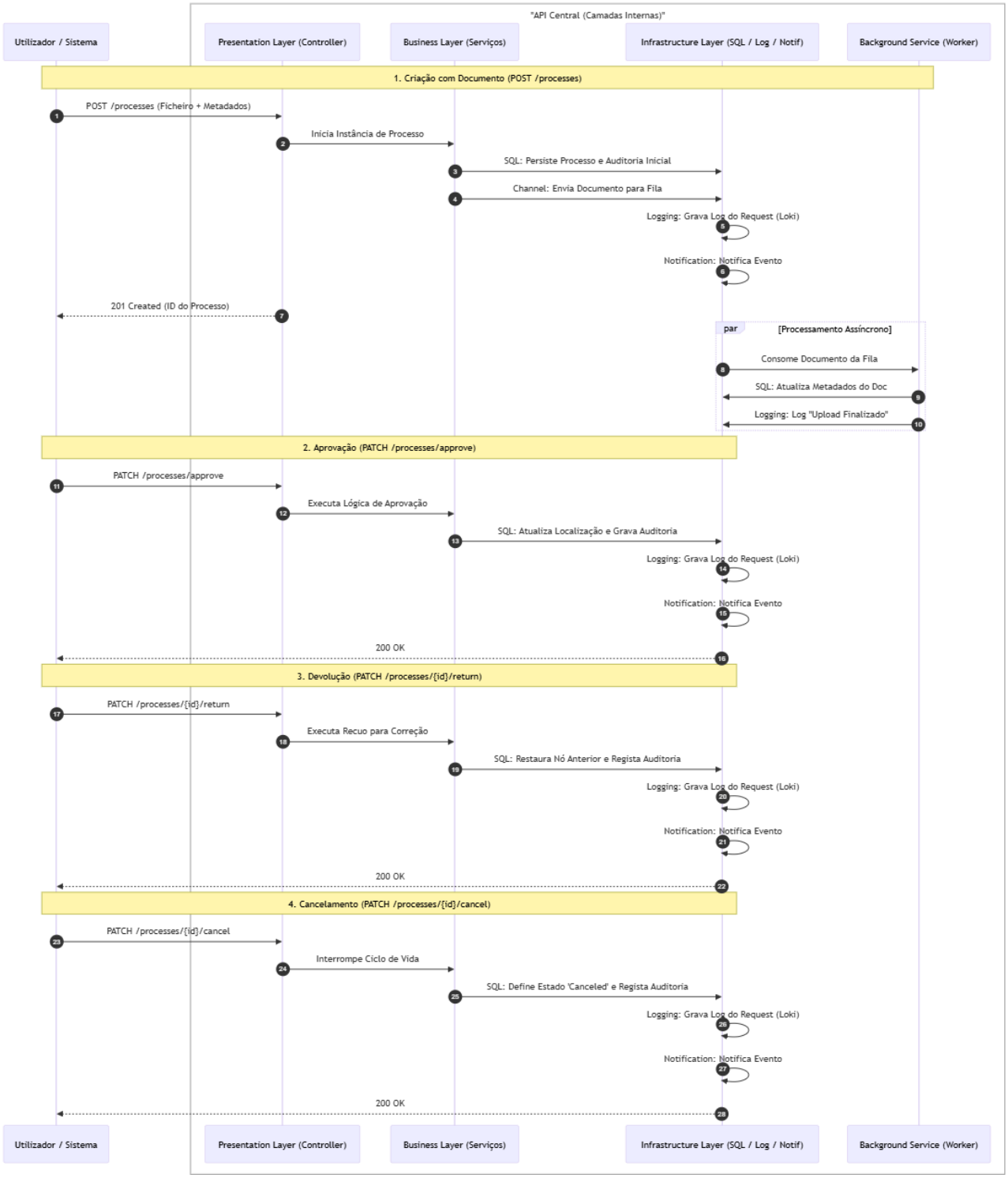



*Fonte: Elaboração própria (2026)*

**g) Consulta com "LogQL" e Persistência de Dados:**

A análise dos dados no "Grafana" é feita através da linguagem de consulta "LogQL". Com ela, o administrador pode filtrar *logs* por palavras-chave, utilizadores ou níveis de erro, permitindo reconstruir todo o histórico de um processo dentro do grafo. Além disso, a utilização de volumes no Docker Compose (mapeamento de pastas do servidor para o contentor) assegura que os *logs* fiquem guardados no disco rígido, garantindo que os dados de auditoria não desapareçam se os contentores forem reiniciados ou actualizados.

*Figura 24 - Diagrama de sequência descrevendo as principais operações da API*

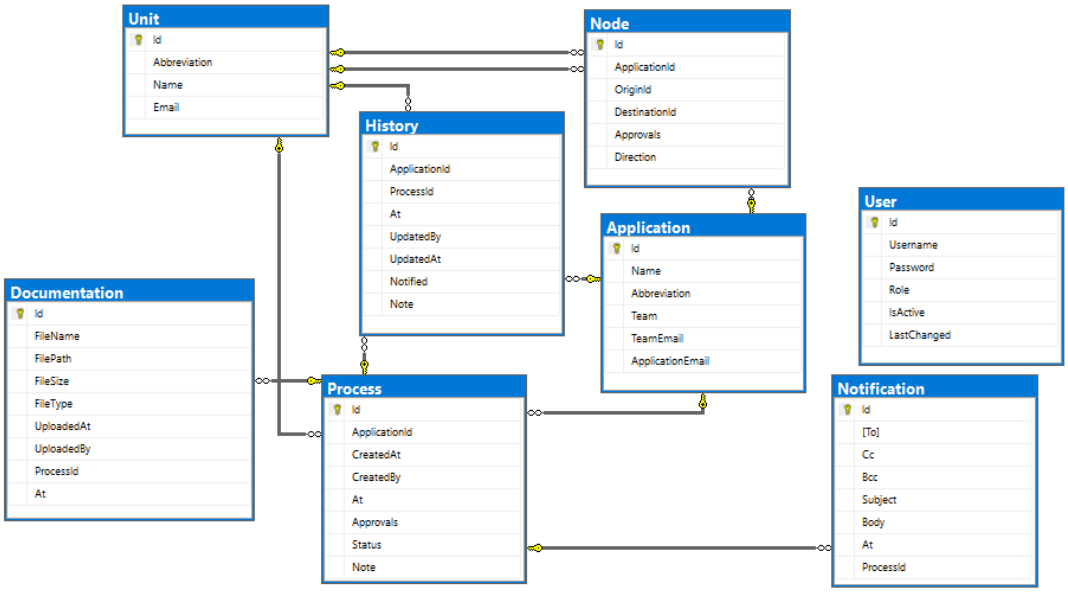



*Fonte: Elaboração própria (2026)*

## 6.5 Modelagem De Dados

No capítulo 3.2, foi estabelecida a estrutura da base de dados capaz de armazenar os ciclos de vida de qualquer *workflow*, juntamente com os seus históricos e intervenientes. Sendo assim, desenvolvemos a partir do que foi estabelecido nesse capítulo e aperfeiçoámo-lo para ser capaz de registar a documentação associada aos processos e as notificações dos eventos para cada intervenção nos processos.

*Figura 25 - Diagrama de entidade e relacionamento*




*Fonte: Gerado pelo SQL S.M.S (2026)*

Para isto, adicionámos as entidades *Documents* e *Notifications*, bem como a entidade *User*, como ilustra o diagrama de entidade completo na Figura 22.

### 6.5.1 Descrição das Novas Entidades e Atribuições

A evolução do modelo de dados permitiu cobrir pontos críticos que garantem a operabilidade do sistema:

- **Entidade *User*:** Esta entidade foi implementada para gerir os utilizadores com privilégios administrativos. Ao contrário dos intervenientes comuns das unidades, os

registos nesta tabela permitem identificar quem tem permissão para criar novos *workflows*, definir os seus ciclos de vida (configurar o grafo) e realizar a manutenção das regras de negócio no sistema.

- **Entidade *Documents*:** Criada para centralizar toda a documentação associada. Cada intervenção num processo pode agora ter ficheiros anexados, garantindo que a evidência documental acompanhe o estado do processo no grafo em tempo real.
- **Entidade *Notifications*:** Atua como o motor de eventos do sistema. Para cada mudança de estado ou intervenção, é gerado um registo que dispara alertas para os intervenientes seguintes, garantindo que o fluxo não fique estagnado por falta de comunicação.

## 6.6 Teste de Carga

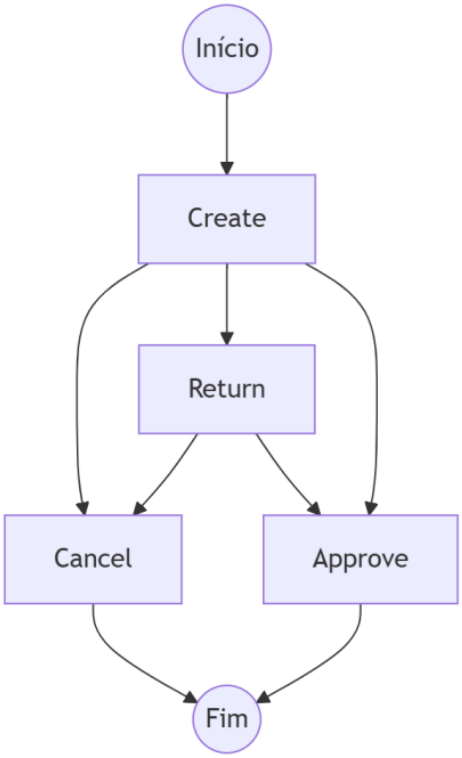
Como a solução proposta foi desenhada para ser o ponto orquestrante de todos os *workflows* da instituição, a sua resiliência é um requisito inegociável. Se o ponto central falha, múltiplas operações bancárias são interrompidas (isto pode ser prevenido com uma expansão horizontal usando *load-balancers*, porém para o nosso teste foi baseado em apenas um servidor). Por isso, é crucial garantir que o sistema suporte grandes volumes de requisições sem comprometer a integridade dos processos.

Este teste de carga foi desenvolvido para simular um dia de trabalho comum sob condições de estresse. O objetivo é observar como a lógica do sistema se comporta quando a demanda de oito horas de expediente é comprimida em poucas horas. Ao forçar os limites de um hardware limitado, conseguimos identificar falhas de escalabilidade e determinar, com precisão, quais recursos de infraestrutura serão necessários para uma operação real estável.

### 6.6.1 Ferramentas e Estrutura do Teste

Para a execução dos testes, utilizou-se o "Apache JMeter", uma ferramenta de código aberto capaz de simular cargas pesadas em servidores e analisar a performance de serviços *web*.

*Figura 26 - Fluxo de teste de carga de um processo*

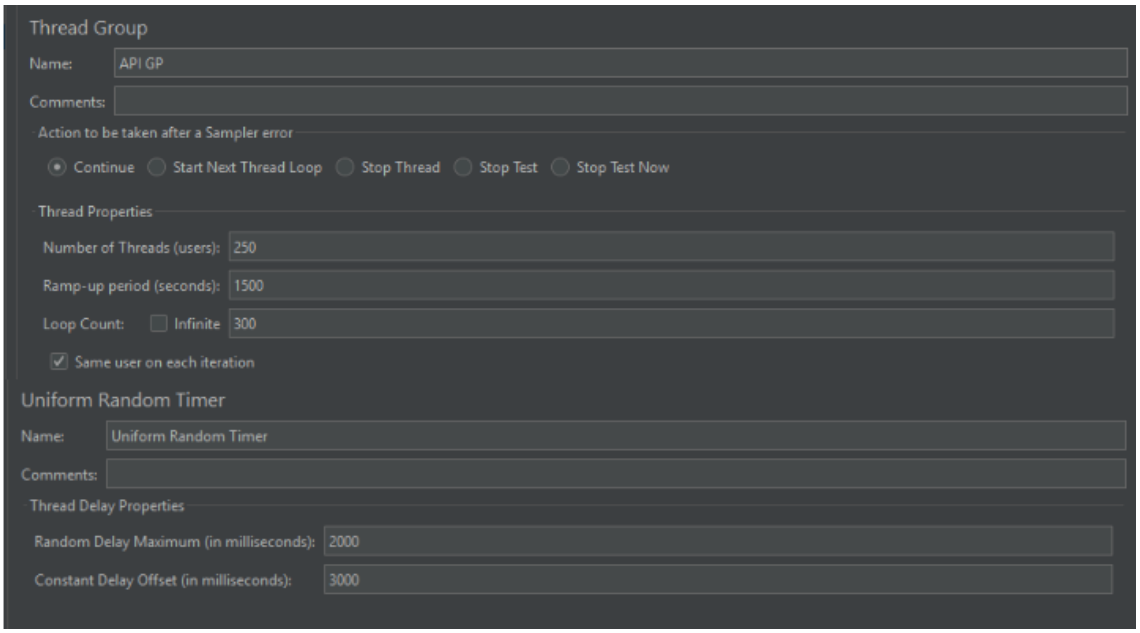



*Fonte: Elaboração própria (2026)*

A estrutura do teste foi desenhada para ser realista:

- **Cenário de Negócio:** Simulamos uma rede de 250 balcões, onde cada um atende cerca de 300 clientes diariamente.
- **Lógica de Execução:** O fluxo seguiu o ciclo de vida real de um processo: *Create, Cancel, Return* e *Approve*.
- **Configuração de Carga:** Foram configurados 250 utilizadores simultâneos, realizando 300 ciclos cada.
- Humanização: Aplicou-se um *Uniform Random Timer* com atraso constante de 3.000ms e variação aleatória de 2.000ms para simular o tempo de resposta humano entre as acções. O período de subida (*ramp-up*) foi de 1.500 segundos.

*Figura 27 - Configuração do teste de carga no JMeter*




*Fonte: Jmeter (2026)*

### 6.6.2 Ambiente de Hospedagem

A *API* foi instalada num servidor que exemplifica os desafios de hardware "fora de validade". Trata-se de um equipamento adquirido em feiras de aparelhos que já cumpriram o seu ciclo de vida útil em instituições bancárias e seriam substituídos por obsolescência.

**Especificações do Hardware:**

- **Processador:** Intel Xeon E3110 (2 núcleos, 3.00GHz).
- **Memória RAM:** 6.8 GiB.
- **Armazenamento:** HDD Mecânico (SDA) de 410GB.

*Figura 28 - Servidor usado para testes de carga*


*Fonte: Captura própria (2026)*

### 6.6.3 Resultados e Análise de Carga

Os resultados da Tabela 4 demonstram uma resiliência notável, com taxas de erro mínimas (abaixo de 0,01%). Este desempenho deve-se à Arquitectura em ".NET 8", que integra padrões como *Polly* (Retentativas), *Timeouts e Rate Limiting* para garantir a continuidade das operações.

Embora o sistema tenha mantido a integridade dos dados, as métricas de Máximo e Desvio Padrão revelam uma queda na performance. Esta oscilação de tempo é um reflexo direto das limitações do hardware utilizado, que gerou filas de processamento sob estresse. Conclui-se que o design do software é robusto, mas a operação real exigirá uma infraestrutura mais robusta ou escalável para garantir tempos de resposta constantes.

*Tabela 4 - Resultados do teste de Carga*

| **Etiqueta** | **Amostras** *O número total de pedidos enviados.* | **Média** *O tempo médio de resposta.* | **Min** *O tempo mais rápido registado* | **Max** *O tempo mais lento registado* | **Desvio Padrão** *Indica a oscilação de tempo* | **% de Erro** | **Vazão** *Requisições por segundo.* | **KB Recebidos/s** | **KB Enviados/s** | **Média de Bytes** *Tamanho médio da resposta* |
|---|---|---|---|---|---|---|---|---|---|---|
| Create | 62269 | 3203 | 1 | 384932 | 10549,99 | 0,08% | 6,5/sec | 4,43 | 5,26 | 697,4 |
| Cancel | 25765 | 8117 | 1 | 23289063 | 383834,74 | 0,0018 | 1,1/sec | 1,21 | 0,26 | 1175,2 |
| Return | 32731 | 11293 | 0 | 23290082 | 464049,85 | 0,0016 | 1,3/sec | 1,44 | 0,32 | 1099,7 |
| Approve | 96816 | 8510 | 1 | 23290153 | 388882,84 | 0,0014 | 4,0/sec | 4,37 | 1,04 | 1133,1 |
| **Total** | **217581** | **31123** | **3** | **70254230** | **1247317,42** | **0,0056** | **12,9/sec** | **11,45** | **6,88** | **4105,4** |

*Fonre: Elaboraçao p'rópria*

### 6.6.4 Análise de Infraestrutura

Para compreender a oscilação nos tempos de resposta observada na Tabela 4, foram analisadas as métricas de hardware do servidor durante o pico de estresse. Os dados revelam três pontos críticos:

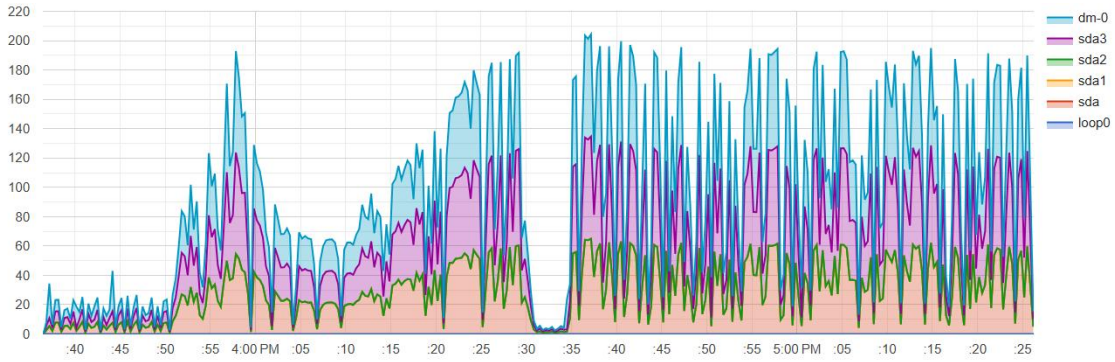
1. **Saturação de *CPU* e *I/O Wait***: O gráfico de utilização da *CPU* mostra picos constantes que atingem o limite de processamento. Mais relevante ainda é a presença significativa de "*Wait%*" (barra amarela), indicando que a *CPU* ficou frequentemente ociosa aguardando que o disco respondesse. Isto explica os tempos máximos elevados, pois o orquestrador ficou bloqueado por hardware lento.

*Figura 29 - Percentual de uso da CPU*


*Fonte: Gerado pelo nmon (2026)*

2. **Gargalo de Disco (*Disk Busy*)**: O gráfico de *Disk Busy Percentage* mostra o volume de operações de entrada e saída (I/O). Nota-se que o disco operou próximo ou acima da sua capacidade nominal durante os ciclos intensos de *Create* e *Approve*. Como o sistema precisa persistir o estado de cada transação bancária para garantir a integridade, o hardware de armazenamento tornou-se o principal limitador (*bottleneck*).

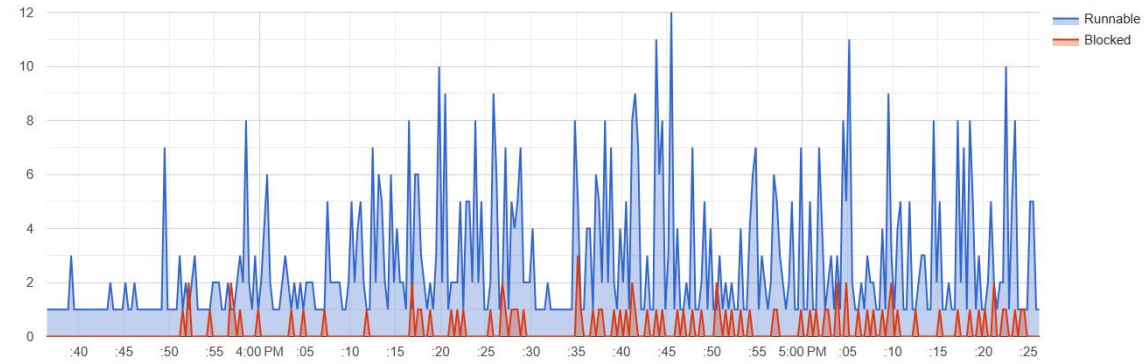
*Figura 30 - Percentual do tempo em que o Disco se encontra ocupado*




*Fonte: Gerado pelo nmon (2026)*

3. **Fila de Processos *(Run Queue)***: O gráfico da *Run Queue* confirma o cenário de estresse: o número de processos em estado *Runnable* (prontos para executar) e *Blocked* (bloqueados) aumentou proporcionalmente à carga. Isso justifica o alto Desvio Padrão: as requisições não eram lentas por causa da lógica do sistema, mas porque estavam acumuladas em fila aguardando recursos de hardware disponíveis.

*Figura 31 - Run Queue, processos correndo, prontos para correr, ou bloqueados*




*Fonte: Gerado pelo nmon (2026)*

## 6.7 Interface de Gestão

Conforme estabelecido no capítulo 4.1, a Arquitectura do sistema integra um *dashboard* administrativo dedicado à gestão integral dos *workflows* e à monitorização do ciclo de vida das instâncias processuais. Para o efeito, foi desenvolvida uma aplicação *Web* baseada no ecossistema *React*, focada na interatividade e responsividade. A implementação desta interface apoia-se num conjunto de bibliotecas especializadas, detalhadas na Tabela 5, que asseguram a manipulação eficiente de estados, a renderização de grafos e a comunicação de dados em tempo real.

*Tabela 5 - Bibliotecas usadas na aplicação web*

| Biblioteca | Versão | Função |
|---|---|---|
| xyflow/react | 12.8.2 | Renderização e interação visual do grafo (nós e arestas). |
| dagre | 2000.8.5 | Algoritmo de organização automática do *layout* do grafo. |
| tanstack/react-router | 1.130.8 | Gestão de rotas e navegação entre vistas do sistema. |
| tanstack/react-query | 5.84.2 | Sincronização e *cache* de dados entre a *API* e a interface. |
| microsoft/signalr | 9.0.6 | Comunicação em tempo real para actualização de estados. |
| zustand | 5.0.7 | Gestão de estado global (sessão e preferências). |
| tanstack/react-table | 8.21.3 | Estruturação de tabelas de dados, histórico e listagens. |
| Radix UI | 1.1.15* | Componentes de interface acessíveis (modais, diálogos). |
| Tailwind CSS | 2004.1 | *Framework* para estilização e design responsivo. |

*Fonte: Elaboração própria (2026)*

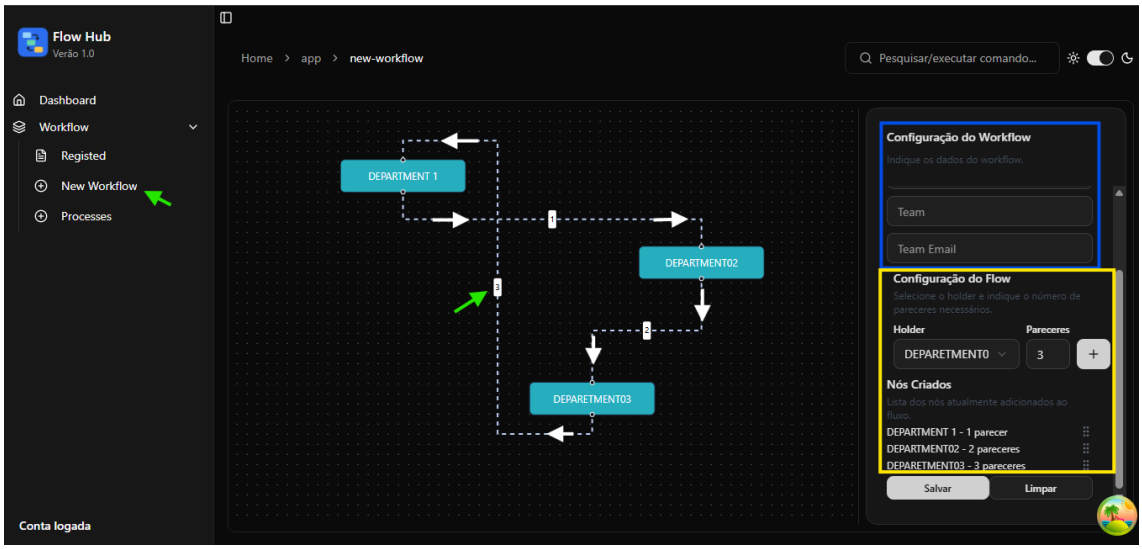
### 6.7.2 Gestão do Ciclo de Vida: Criação de *Workflows*

Para o registo de um novo *workflow* e a configuração do respetivo ciclo de vida, utiliza-se a página **"**New Workflow**"**, acessível através da barra de navegação lateral. A interface está estruturada em duas secções principais que permitem a interação dinâmica com o modelo de grafos:

- **Painel de Visualização (Esquerda):** Exibe, em tempo real, a antevisão do grafo que está a ser modelado, permitindo validar visualmente as conexões entre os nós.
- **Painel de Configuração (Direita):** Destina-se à definição dos parâmetros técnicos e lógicos do processo, subdividindo-se em duas áreas críticas (ver Figura 29):
  - **Metadados do *Workflow* (Destaque Azul):** Onde são definidos os dados globais da aplicação (nome, acrónimo e suporte).
  - **Configuração do Ciclo de Vida (Destaque Amarelo):** Permite a gestão dos nós do grafo. Nesta área, o utilizador pode criar intervenientes, caso não constem no sistema, e organizar a sequência do fluxo.

A interface suporta a reordenação intuitiva dos itens; ao arrastar os elementos da lista de nós, a antevisão à esquerda é actualizada automaticamente. Após a parametrização de todos os campos, o acionamento do comando **"**Salvar**"** garante a persistência do *workflow* e da sua topologia de rede na base de dados.

*Figura 32 - Criação de workflows e configuração do ciclo de vida dos seus processos*

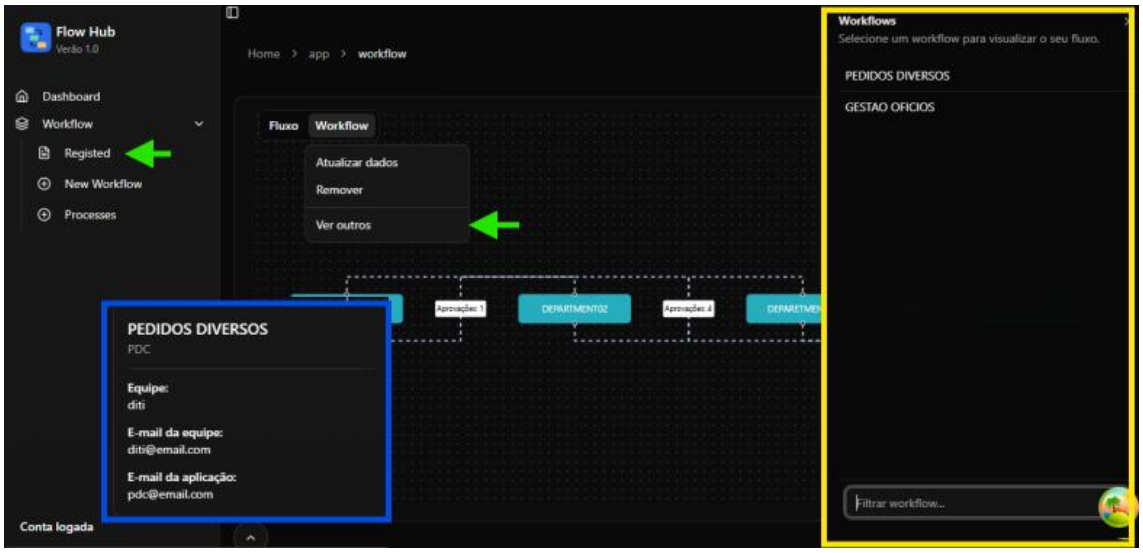



*Fonte: Recorte da aplicação (2026)*

Após a criação e persistência dos modelos, o sistema permite a consulta e auditoria dos ciclos de vida através da página "Registed", acessível na barra lateral. Esta interface centraliza a informação em três áreas principais, conforme ilustrado na Figura 30:

- **Seletor de *Workflows* (Destaque Amarelo):** Painel lateral que lista todos os processos parametrizados no sistema (ex: "Pedidos Diversos"). Inclui uma ferramenta de filtragem rápida para localizar modelos específicos em bases de dados extensas.
- **Painel de Metadados (Destaque Azul):** Exibe as informações institucionais do *workflow* selecionado, como a equipa responsável e os endereços de correio eletrónico associados, garantindo a identificação inequívoca da aplicação.
- **Visualizador Dinâmico do Grafo (Centro):** Renderiza a topologia do ciclo de vida.

*Figura 33 - Visualização de Workflows e ciclos de vida na aplicação*




*Fonte: Recorte da aplicação (2026)*

## 6.7.4 Identificação de Processos e Histórico Associado

A visualização dos processos é centralizada na página **"Processes"**. Esta interface permite não apenas a listagem tabular, mas uma auditoria profunda de cada instância através da expansão de linhas, conforme demonstrado na Figura 31.

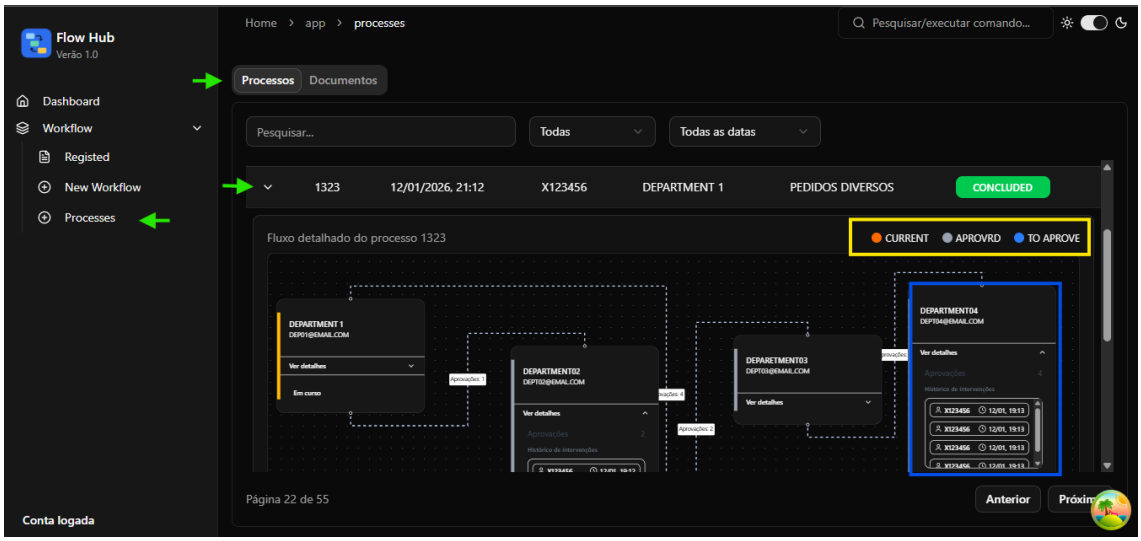
Ao expandir um processo específico (ex: ID 1323), o *Dashboard* consome os dados das entidades "History" e "Nodes" para oferecer uma visão geral:

- **Identificação Visual de Estados (Legenda Amarela):** O sistema utiliza um código de cores dinâmico para indicar o status de cada nó no grafo:
  - **Laranja (Onde se encontra actualmente):** Indica o departamento onde o processo se encontra actualmente retido.
  - **Cinzento (Por onde passou e foi aprovado):** Identifica as etapas que já foram concluídas com sucesso.
  - **Azul (Por onde vai passar):** Sinaliza os próximos passos previstos no ciclo de vida.

- **Histórico de Intervenções (Destaque Azul):** Dentro de cada nó do grafo, é possível visualizar o registo detalhado das acções. O sistema lista o *ID* do colaborador, a data e a

hora exata da intervenção, garantindo a total transparência e responsabilidade operacional.

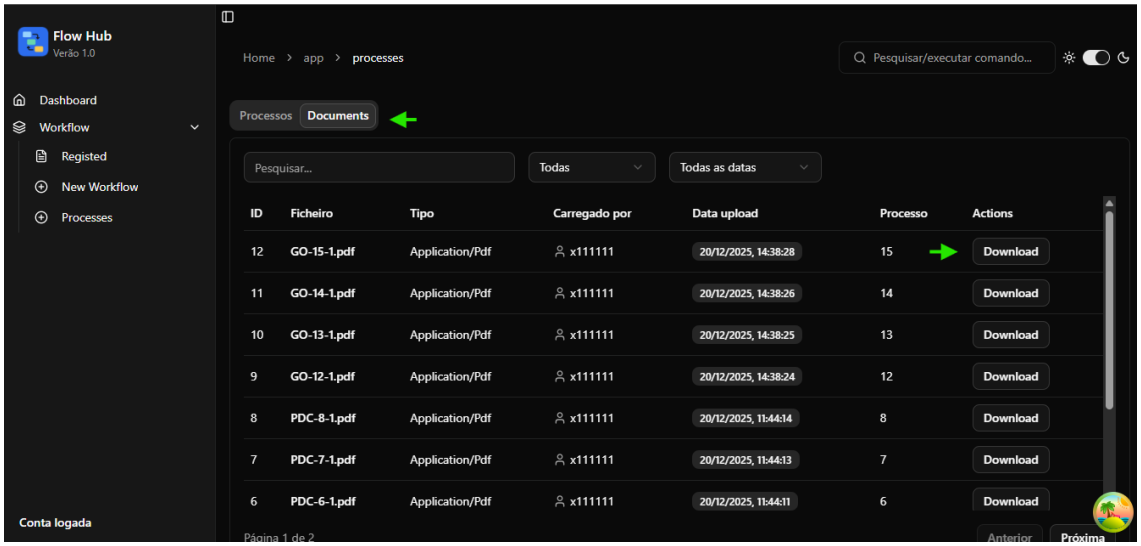
*Figura 34 - Auditoria dos processos criados pelos workflows na aplicação*




*Fonte: Recorte da aplicação (2026)*

A interface integra um separador de "*Documents*" que centraliza todos os ficheiros anexados durante o ciclo de vida. Esta área permite identificar o autor do *upload*, a data exata e o vínculo ao processo, disponibilizando o *download* imediato para validação da tomada de decisão. Esta organização garante que a progressão no grafo seja sempre acompanhada pela respetiva prova documental e histórico de intervenções.

*Figura 35 - Documentação associada a cada processo*




*Fonte: Recorte da aplicação (2026)*

# 7. Discussão e Resultados

O presente capítulo tem como propósito apresentar os resultados obtidos com o desenvolvimento do motor de gestão de fluxos baseado em grafos. Inicialmente, procedo a uma análise do cenário actual da instituição em estudo, evidenciando como a dispersão da lógica de processos em múltiplas plataformas cria o obstáculo técnico que motiva este trabalho.

Na sequência, são detalhados os resultados da solução implementada: uma *API* centralizada em .NET 8 que permite a definição e gestão unificada de diversos *workflows*. Posteriormente, realiza-se uma discussão crítica sobre como esta base centralizada permite o controlo total sobre passos e documentações, relacionando-a com o referencial teórico de grafos. Por fim, analisam-se os testes de desempenho realizados para validar a robustez da *API* em ambiente bancário.

## 7.1 Cenário actual da gestão de processos

Para compreender a importância da solução, é necessário observar o fluxo operacional vigente. Actualmente, os processos (como abertura de conta ou crédito) são geridos por sistemas independentes. Cada sistema detém a sua própria lógica e base de dados, o que resulta numa gestão fragmentada.

Neste modelo, não existe um ponto único onde se possa visualizar ou alterar o progresso de um cliente que transita entre diferentes serviços. A "inteligência" do processo está isolada em cada software, impedindo que a instituição tenha uma visão holística. É esta Arquitectura dispersa que obriga o cliente a repetir acções e entregas de documentos, pois os sistemas não possuem uma base de informação comum para consultar o estado global das pendências.

## 7.2 Resultados

A solução desenvolvida rompe com o modelo de silos ao oferecer uma *API* centralizada onde toda a lógica de negócio é modelada através de grafos. Os principais resultados práticos desta implementação são:

- **Definição Unificada de Fluxos:** Através do motor desenvolvido, é possível definir, num único lugar, o grafo de qualquer processo bancário, especificando vértices (áreas), arestas (transições) e pesos (pareceres/documentos necessários).
- **Gestão de Documentação Integrada:** O sistema permite gerir as exigências documentais de cada passo do *workflow* de forma centralizada. Isso significa que, se um documento foi validado no "Passo A" de um fluxo, essa informação está disponível para qualquer outro processo gerido pela *API*.
- **Orquestração de Estados:** A *API* funciona como o "mapa" da instituição. Ela sabe exatamente em que estágio do ciclo de vida o cliente se encontra, independentemente de qual departamento está a processar a tarefa no momento.

Esta base estabelecida cria a possibilidade real de coordenar múltiplos processos simultaneamente, eliminando a necessidade de o cliente gerir manualmente as suas próprias dependências.

## 7.3 Análise de Desempenho

Para garantir que um motor centralizado suportaria a carga de uma instituição comercial, a API foi submetida a testes de estresse. Utilizando o JMeter, simulou-se o processamento de múltiplos fluxos simultâneos, focando na latência da transição entre estados do grafo.

Os dados capturados pelo NMON mostraram que a gestão centralizada não sobrecarrega o hardware. O consumo de *CPU* e memória manteve-se dentro de limites otimizados, demonstrando que a Arquitectura de *Clean Architecture* e o uso de grafos permitem uma gestão de processos rápida e escalável. A eficiência na recuperação de regras de transição (arestas) provou ser superior ao modelo de consultas em múltiplas bases de dados isoladas.

## 7.4 Discussão Crítica e Contributos

A discussão dos resultados revela que o maior ganho da solução não é apenas a automação, mas a visibilidade. Ao converter *workflows* implícitos em grafos explícitos dentro de uma API, a instituição ganha a capacidade de manipular processos de forma uniforme.

Academicamente, o trabalho demonstra que a complexidade bancária pode ser reduzida a uma abstração matemática (Grafos) gerida por uma interface tecnológica moderna (*REST*

*API)*. Ao centralizar as informações e regras num único lugar, resolve-se o principal obstáculo técnico à eficiência: a falta de um ponto comum de representação e controlo. Esta solução pavimenta o caminho para um atendimento bancário onde o sistema, e não o cliente, é o responsável pela continuidade do serviço.

# 8. Conclusões e Recomendações

Este capítulo sintetiza os resultados alcançados com o desenvolvimento da plataforma de representação de processos em grafo, destacando a sua maturidade técnica e apontando os caminhos para a sua evolução rumo a uma integração operacional mais inteligente e uniforme.

## 8.1 Conclusões Gerais

O presente trabalho permitiu demonstrar que é possível melhorar a continuidade dos processos bancários através da criação de uma infra-estrutura comum de representação e gestão dos fluxos.

**Análise dos Processos Actuais**

A análise realizada à estrutura actual dos serviços da instituição em estudo permitiu identificar dificuldades relacionadas com a comunicação entre processos pertencentes a sistemas diferentes. Verificou-se que, em diversos cenários, a continuidade entre etapas dependia de intervenções manuais realizadas pelos colaboradores ou pelo próprio cliente, contribuindo para atrasos no atendimento, aumento do esforço operacional e crescimento das filas nos balcões.

Observou-se igualmente que os sistemas existentes funcionavam de forma isolada, dificultando o acompanhamento uniforme dos processos e a coordenação entre diferentes áreas operacionais.

**Modelação dos Processos em Grafo**

A utilização da Teoria de Grafos mostrou-se adequada para representar os ciclos de vida dos processos bancários. A modelação através de vértices, arestas e transições permitiu estruturar os diferentes workflows sob uma mesma lógica de funcionamento, tornando possível representar processos distintos de forma uniforme.

A abordagem adoptada permitiu organizar as etapas dos processos, os seus estados e as respectivas regras de tramitação dentro de uma estrutura comum, facilitando a representação e acompanhamento dos fluxos.

**Desenvolvimento da Infra-estrutura Integradora**

A implementação da API central e da plataforma web de gestão permitiu validar a criação de uma camada comum para acompanhamento dos processos. A solução desenvolvida demonstrou capacidade para gerir workflows, controlar tramitações, armazenar documentação associada aos processos e manter o histórico das intervenções realizadas durante o ciclo de vida de cada fluxo.

A Arquitectura adoptada permitiu ainda que os sistemas existentes permanecessem independentes, utilizando a infra-estrutura proposta apenas como ponto comum de representação e gestão dos processos.

**Validação Técnica da Solução**

Os testes técnicos realizados demonstraram a viabilidade da abordagem proposta. Os resultados obtidos indicaram que a Arquitectura implementada manteve estabilidade e tempos de resposta adequados mesmo sob cenários de utilização simultânea.

Adicionalmente, as ferramentas de monitorização utilizadas permitiram acompanhar o comportamento da solução em tempo real, disponibilizando mecanismos de observabilidade, auditoria e rastreabilidade das operações executadas pela API.

Conclui-se, portanto, que a solução desenvolvida constitui uma abordagem viável para uniformizar a representação e gestão de processos bancários provenientes de sistemas distintos, permitindo melhorar a continuidade entre serviços, reduzir dependências manuais e disponibilizar uma visão mais centralizada sobre os fluxos em execução.

## 8.2 Recomendações

Com a base tecnológica e a monitorização operacional já estabelecidas, as recomendações focam-se na expansão das capacidades de integração da infra-estrutura.

**8.2.1 Módulo de Encadeamento de Processos em Cadeia** A recomendação primordial deste estudo é o desenvolvimento de um módulo de integração proactiva entre serviços. Uma vez que toda a informação de dependências e documentação já reside na infra-estrutura central, o sistema possui a fundação para:

- **Encadeamento Integrado de Processos:** A infra-estrutura poderá futuramente permitir despachar processos para as áreas de direito na sequência definida, eliminando o papel do cliente ou do funcionário como "mensageiro" entre departamentos.
- **Criação de Processos em Cascata:** Ao finalizar uma etapa (ex: Actualização de Dados), a infra-estrutura poderá futuramente permitir o encadeamento automático entre processos relacionados (ex: Pedido de Crédito), reaproveitando os dados já validados e garantindo a continuidade do atendimento.

### 8.2.2 Monitorização Preditiva e Integração de Ecossistema

Embora o sistema já possua observabilidade técnica, recomenda-se a evolução para uma **Monitorização de Negócio Preditiva**:

- **Análise de Gargalos via Pesos de Grafos:** Utilizar os dados do Grafana e os pesos das arestas (esforço/tempo) para prever atrasos em fluxos críticos antes que estes afectem o cliente final.
- **Integração Profunda com o Core Bancário:** Desenvolver *webhooks* e conectores automáticos para que a conclusão de um vértice no grafo permita integração directa com operações financeiras existentes (como a libertação de fundos ou bloqueio de cartões) de forma direta e segura.

## 8.3 Perspetivas Futuras

No âmbito académico, este trabalho prova que a complexidade de uma instituição pode ser reduzida a uma abstração matemática funcional. No campo prático, a solução não é apenas uma ferramenta de suporte, mas a base para um ambiente bancário mais integrado, rastreável e eficiente.

Espera-se que esta infraestrutura sirva de modelo para a modernização tecnológica em Moçambique, demonstrando que centralização da representação e acompanhamento dos processos é o caminho mais eficiente para aumentar a transparência, reduzir custos e, acima de tudo, dignificar o tempo do cliente bancário.

# Referências Bibliográficas


- Apache Software Foundation. (2024). Apache JMeter User's Guide. Apache Software Foundation.
- Bandarupalli, S. (2025). Modernization of Legacy Systems and Distributed Architectures. Springer.
- Banco de Moçambique. (2022). Estratégia Nacional de Inclusão Financeira. Banco de Moçambique.
- Banco de Moçambique. (2023). Relatório de Inclusão Financeira. Banco de Moçambique.
- Barskiy, M. (2023). Hands-On Web API Development with ASP.NET Core 8. Packt Publishing.
- Belchior, R., Vasconcelos, A., Guerreiro, S., & Correia, M. (2025). Interoperability Challenges in Financial Systems Integration. IEEE Access.
- Bondy, J. A., & Murty, U. S. R. (2008). Graph Theory. Springer.
- Creswell, J. W., & Creswell, J. D. (2018). Research Design: Qualitative, Quantitative, and Mixed Methods Approaches (5th ed.). SAGE Publications.
- Diestel, R. (2024). Graph Theory (6th ed.). Springer.
- Diário Económico. (2024). Crescimento da procura por serviços bancários aumenta pressão operacional nos balcões.
- Docker Inc. (2026). Docker Documentation. Docker Inc.
- Dumas, M., La Rosa, M., Mendling, J., & Reijers, H. A. (2020). Fundamentals of Business Process Management (2nd ed.). Springer.
- Gil, A. C. (2008). Métodos e Técnicas de Pesquisa Social (6.ª ed.). Atlas.
- Giretti, A. (2024). Coding Clean, Reliable, and Safe REST APIs with ASP.NET Core 8. Packt Publishing.
- Grafana Labs. (2024). Grafana & Loki Documentation. Grafana Labs.

- IEEE. (2023). Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries. IEEE.
- Jin, X., Zhao, L., & Chen, Y. (2024). Legacy Systems and Digital Transformation in Banking Environments. Journal of Financial Technology Systems.
- Keller, M., & Trotter, W. (2023). Applied Graph Structures in Organisational Workflows. Springer.
- Laudon, K. C., & Laudon, J. P. (2021). Management Information Systems: Managing the Digital Firm (16th ed.). Pearson.
- Martin, R. C. (2017). Clean Architecture: A Craftsman's Guide to Software Structure and Design. Prentice Hall.
- Microsoft. (2024). Entity Framework Core Documentation. Microsoft Learn.
- Mozilla Foundation. (2026). MDN Web Docs: multipart/form-data. Mozilla Developer Network.
- O País. (2024). Filas e demora no atendimento continuam a afectar clientes bancários em Moçambique.
- Patterson, D. A., & Hennessy, J. L. (2021). Computer Organization and Design. Morgan Kaufmann.
- Stair, R., & Reynolds, G. (2020). Principles of Information Systems. Cengage Learning.
- van der Aalst, W. M. P. (2016). Process Mining: Data Science in Action (2nd ed.). Springer.
- Wieringa, R. (2023). Design Science Methodology for Information Systems and Software Engineering. Springer.

## Apêndice A – Repositórios Da Solução

De modo a permitir a consulta da implementação técnica e dos padrões de Arquitectura descritos neste trabalho, o código-fonte está disponível nos seguintes endereços:

- *Backend* (*API*): **https://github.com/BrunoMTR/API-GP.git**
- *Frontend* (*Dashboard*): **https://github.com/BrunoMTR/gp-dashboard.git**

## Apêndice B – Ficheiros de Configuração (Docker)

### *Mailpit:*

```
version: '3'
services:
  mailpit:
    image: axllent/mailpit
    container_name: mailpit
    restart: unless-stopped
    volumes:
      - ./data:/data
    ports:
      - 8025:8025
      - 1025:1025
    environment:
      MP_MAX_MESSAGES: 5000
      MP_DATABASE: /data/mailpit.db
      MP_SMTP_AUTH_ACCEPT_ANY: 1
      MP_SMTP_AUTH_ALLOW_INSECURE: 1
```

### *Grafana/Loki:*

```
version: "3.8"
services:
  loki:
    image: grafana/loki:2.9.0
    container_name: loki
    ports:
      - "3100:3100"
    command: -config.file=/etc/loki/local-config.yaml
    volumes:
      - ./loki-data:/loki
```

```
    - ./loki-config.yaml:/etc/loki/local-config.yaml
    - ./loki-data/wal:/wal
promtail:
  image: grafana/promtail:2.9.0
  container_name: promtail
  volumes:
    - ./logs:/logs
    - ./promtail-config.yaml:/etc/promtail/promtail.yaml
  command: -config.file=/etc/promtail/promtail.yaml
  depends_on:
    - loki
grafana:
  image: grafana/grafana:latest
  container_name: grafana
  ports:
    - "3000:3000"
  environment:
    - GF_SECURITY_ADMIN_PASSWORD=admin
    - GF_LOG_LEVEL=info
  depends_on:
    - loki
```