INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




# Gestão de Dados para a Produção Documental de Seguros

**PEDRO NUNO FERRAZ FERNANDES**
Setembro de 2025

POLITÉCNICO
DO PORTO

ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

# Gestão de Dados para a Produção Documental de Seguros

**Pedro Nuno Ferraz Fernandes**

**Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Engenharia Software**

**Orientador: Bruno Silva**
**Supervisor: Daniel Barciela**

Porto, 13 de setembro de 2025

# Declaração de Integridade

Certifico que este trabalho académico foi realizado com total integridade.

Declaro não ter cometido plágio, nem utilizado qualquer forma de uso indevido de informações ou falsificação de resultados durante o processo de sua elaboração.

Assim, asseguro que o trabalho apresentado neste documento é original e da minha inteira autoria, não tendo sido submetido anteriormente para qualquer outro propósito.

Adicionalmente, afirmo ter pleno conhecimento do Código de Conduta Ética do P.PORTO.

ISEP, Porto, 13 de setembro de 2025

# Resumo

Este trabalho apresenta o desenvolvimento de uma nova solução para o módulo de recolha e envio de dados de uma organização do setor segurador, concebida para superar as limitações da arquitetura anterior, caracterizada por elevado acoplamento entre componentes, custos elevados de manutenção, tempos de processamento excessivos e dificuldades de compatibilidade com múltiplos formatos de documentos.

A solução proposta assenta numa arquitetura modular e flexível, suportada por *template engines* e serviços internos otimizados, permitindo a geração de documentos em diferentes formatos e a adaptação a novos requisitos de negócio. O desenvolvimento foi suportado por uma revisão do estado da arte segundo a metodologia PRISMA, garantindo alinhamento com as melhores práticas e tecnologias relevantes, e pela adoção de um processo inspirado na metodologia Scrum, adaptado a um projeto individual.

A validação da solução envolveu testes funcionais e de desempenho realizados em ambiente controlado, comparando a nova implementação com a solução anterior. Os resultados demonstraram a manutenção da integridade e consistência dos dados gerados, bem como uma redução significativa nos tempos médios de processamento, eliminando a necessidade de chamadas remotas e melhorando a agilidade de construção de novos *templates*.

Questões éticas e regulamentares, nomeadamente a proteção de dados e a conformidade legal, foram consideradas ao longo de todo o projeto. As conclusões obtidas confirmam o cumprimento integral dos objetivos definidos, estabelecendo uma base sólida para evolução futura, que inclui a finalização de todos os *templates*, a otimização adicional das consultas à base de dados e a integração do *frontend* nas plataformas internas da organização.

A solução desenvolvida contribui para reforçar a competitividade da organização no setor segurador, proporcionando um sistema mais rápido, flexível e preparado para os desafios tecnológicos do futuro.

**Palavras-chave:** *Template Engines*, Arquitetura Flexível, Seguros, Compatibilidade de Dados, Geração de Documentos, Otimização de Desempenho

# Abstract

This work presents the development of a new solution for the data collection and transmission module of an insurance sector organization, designed to overcome the limitations of the previous architecture, which was characterized by high component coupling, elevated maintenance costs, excessive processing times, and difficulties in ensuring compatibility with multiple document formats.

The proposed solution is based on a modular and flexible architecture, supported by template engines and optimized internal services, enabling the generation of documents in different formats and adapting to new business requirements. The development was supported by a state-of-the-art review conducted using the PRISMA methodology, ensuring alignment with best practices and relevant technologies, as well as the adoption of a process inspired by the Scrum methodology, adapted to an individual project context.

The solution's validation involved functional and performance testing carried out in a controlled environment, comparing the new implementation with the previous one. The results demonstrated the preservation of data integrity and consistency, as well as a significant reduction in average processing times, eliminating the need for remote calls and improving the agility in building new templates.

Ethical and regulatory considerations, namely data protection and legal compliance, were taken into account throughout the project. The conclusions confirm the full achievement of the defined objectives, establishing a solid foundation for future evolution, which includes the completion of all templates, further optimization of database queries, and the integration of the frontend into the organization's internal platforms.

The developed solution contributes to strengthening the organization's competitiveness in the insurance sector, providing a faster, more flexible system prepared for future technological challenges.

**Keywords:** Template Engines, Flexible Architecture, Insurance, Data Compatibility, Document Generation, Performance Optimization

# Conteúdo

# Lista de Figuras

# Lista de Tabelas

# Lista de Código

# Lista de Acrónimos

| | |
|---|---|
| ANSI | *American National Standards Institute*. |
| FTL | *FreeMarker Template Language*. |
| GenAI | *Generative Artificial Intelligence*. |
| HTML | *HyperText Markup Language*. |
| IPP | Instituto Politécnico do Porto. |
| JSON | *JavaScript Object Notation*. |
| JVM | *Java Virtual Machine*. |
| MVC | *Model-View-Controller*. |
| PRISMA | *Preferred Reporting Items for Systematic reviews and Meta-Analyses*. |
| RAC | *Real Application Clusters*. |
| RGPD | Regulamento Geral de Proteção de Dados. |
| SQL | *Structured Query Language*. |
| WBS | *Work Breakdown Structure*. |
| XML | *Extensible Markup Language*. |

# 1. Introdução

Este capítulo tem como objetivo contextualizar o projeto desenvolvido no âmbito do setor segurador, apresentando a relevância do tema, os desafios tecnológicos enfrentados pela organização e a necessidade de uma solução inovadora para a gestão documental. Serão abordados o enquadramento do problema, a apresentação da organização, a identificação detalhada das limitações do sistema atual, os objetivos definidos para o projeto, as considerações éticas seguidas durante a execução e o planeamento das atividades. Além disso, é apresentada a estrutura do documento para guiar o leitor.

## 1.1 Enquadramento

O setor de seguros desempenha um papel central na economia global, oferecendo produtos e serviços que seguram indivíduos e empresas contra uma ampla gama de riscos. A gestão eficiente de informações, contratos e outros documentos é um pilar fundamental para a operação bem-sucedida das seguradoras. Nesse contexto, a transformação digital emergiu como um diferencial competitivo, impulsionando a adoção de tecnologias avançadas para melhorar processos operacionais, aumentar a satisfação dos clientes e assegurar conformidade regulamentar.

Adicionalmente, tendências tecnológicas como a *Generative Artificial Intelligence* (GenAI) e plataformas *low-code/no-code* estão a tornar-se fundamentais no setor segurador, permitindo não só a automação de fluxos de trabalho, mas também a entrega de soluções mais personalizadas e eficazes. Estas ferramentas apresentam um potencial significativo para otimizar processos, como a emissão e renovação de apólices, satisfazendo as expectativas dos consumidores e as necessidades de negócios em constante transformação (I. A. James Ingham e Narisawa 2024).

Outro aspeto crítico é a crescente adoção de *marketplaces* e ecossistemas tecnológicos específicos, que permitem às seguradoras combinar soluções de diferentes fornecedores. Este modelo não só amplia a flexibilidade e a escalabilidade dos sistemas, mas também facilita a adaptação a novas exigências regulamentares e tecnológicas, promovendo uma maior agilidade operacional (S. G. James Ingham L. S. 2024).

Com a transformação digital, o setor de seguros enfrenta desafios crescentes por soluções tecnológicas que aumentem a flexibilidade, a personalização e a eficiência operacional. Estas soluções precisam atender a requisitos diversos, como a integração com sistemas legados, a conformidade com regulamentações em constante evolução e a capacidade de gerar informação em múltiplos formatos. Além disso, a crescente expectativa por respostas rápidas e personalizadas exige que os sistemas sejam mais flexíveis.

No entanto, a modernização de sistemas legados no setor de seguros é um desafio significativo (NTT Data Corporation 2024). Muitos sistemas foram projetados em um momento em que os requisitos tecnológicos eram diferentes, o que os torna rígidos e pouco adaptáveis às novas exigências de mercado. Este contexto reforça a necessidade de repensar as arquiteturas existentes e desenvolver soluções que equilibrem a compatibilidade com o legado e a inovação tecnológica.

Assim, este trabalho visa explorar essas questões no contexto de uma organização do setor dos seguros, focando-se num módulo específico responsável pela recolha e envio de dados para a gestão documental. Este módulo apresenta limitações que comprometem a adaptabilidade e ligação aos clientes, evidenciando a necessidade de uma reestruturação tecnológica.

## 1.2 Apresentação da Organização

A CLEVA é uma empresa dedicada à produção de *software* para seguradoras. Foi fundada em 1984 com o nome de I2S e é um líder consolidado no mercado segurador de Portugal, Angola e Moçambique, com implementações em mais de 40 seguradoras em diversos países.

Em 2019, a I2S uniu forças com o Grupo GFI, hoje em dia renomeado para INETUM, uma organização internacional, para reforçar a sua presença global e alavancar o portefólio conjunto de plataformas, incluindo CLEVA e COGIT, dirigidas ao mercado de seguros. A fusão teve como objetivo expandir a oferta de soluções tecnológicas para os seguros de Vida, Pensões e Não Vida, enquanto mantém o foco na inovação e na continuidade operacional. Esta parceria reforçou o compromisso com a qualidade e o suporte aos clientes existentes, além de preparar a expansão para novos mercados internacionais (Inetum 2024).

No sentido de uniformizar processos e visibilidade para os clientes, a empresa e o *software* também conhecido por I2S passou a ser chamado por CLEVA convergindo assim para a unidade de soluções entre as empresas de Portugal e França, no que está relacionados com o mercado segurador (Cleva 2024a).

Recentemente, a ANACAP, uma empresa europeia especializada em investimentos no setor de tecnologia financeira efetuou a aquisição da CLEVA, reforçando a sua posição no mercado global como fornecedora de plataformas inovadoras para a gestão de seguros. Com esta transação, a CLEVA pretende expandir as suas operações e continuar a oferecer soluções robustas e personalizáveis para seguradoras, consolidando a sua reputação como um dos principais atores no setor de tecnologia para seguros (Cleva 2024b).

Atualmente, a CLEVA leva mais 40 anos de experiência, mais de 300 colaboradores, mais de 100 projetos implementados e o seu *software* gere mais de 16 milhões de apólices (Cleva 2024a).

## 1.3 Problema

A gestão documental desempenha um papel essencial no setor segurador, uma vez que documentos como apólices, recibos e participações de sinistros são fundamentais para assegurar o correto funcionamento das operações. Neste contexto, os diversos módulos e aplicações que compõem o ecossistema de software da organização têm necessidades constantes de geração de documentos, nomeadamente emissões de apólices, comunicações formais, cartas e outros artefactos documentais.

Entre estes módulos, destaca-se o Core, considerado o elemento central na produção documental. Este módulo inclui um componente próprio de recolha e envio de dados, o qual, apesar de funcional, apresenta limitações significativas que comprometem a eficiência global do sistema e dificultam a capacidade de resposta da organização às exigências do negócio.

Um dos principais entraves da solução atual é a ausência de mecanismos flexíveis para customização dos *outputs* gerados. Qualquer alteração, seja na estrutura do documento, no

seu conteúdo ou no formato de saída, requer intervenção direta da equipa de desenvolvimento. Esta dependência não só aumenta o tempo necessário para implementar mudanças, como também sobrecarrega a equipa técnica, reduzindo a agilidade do negócio perante novas exigências ou outros ajustes. Na prática, alterações simples, como a adição de um novo campo, podem implicar múltiplos ciclos de desenvolvimento, testes e implantação, atrasando significativamente a resposta às necessidades operacionais.

Assim, este problema foi categorizado e analisado em vários pontos críticos, nomeadamente, arquitetura com alto acoplamento, desempenho insuficiente, custos elevados de manutenção, incompatibilidade com diversos formatos e impacto direto no tempo de resposta às solicitações de negócio.

### 1.3.1 Arquitetura com Alto Acoplamento

O módulo do Core apresenta uma arquitetura com um elevado grau de acoplamento, o que compromete a eficiência, escalabilidade e capacidade de resposta da solução. Para compreender as limitações desta arquitetura, é essencial analisar o papel de cada componente no fluxo de processamento, conforme ilustrado na Figura 1.1.

O Core é constituído por duas grandes aplicações: *batch-life-ear* e *life-ear*. Estas aplicações possuem componentes distintos, sendo a primeira vocacionada para processos *batch* e a segunda orientada para interações com o utilizador. Para além dos conteúdos específicos de cada aplicação, existe ainda um conjunto de artefactos comuns, nomeadamente a camada de negócio, designada por *business-bbl*.

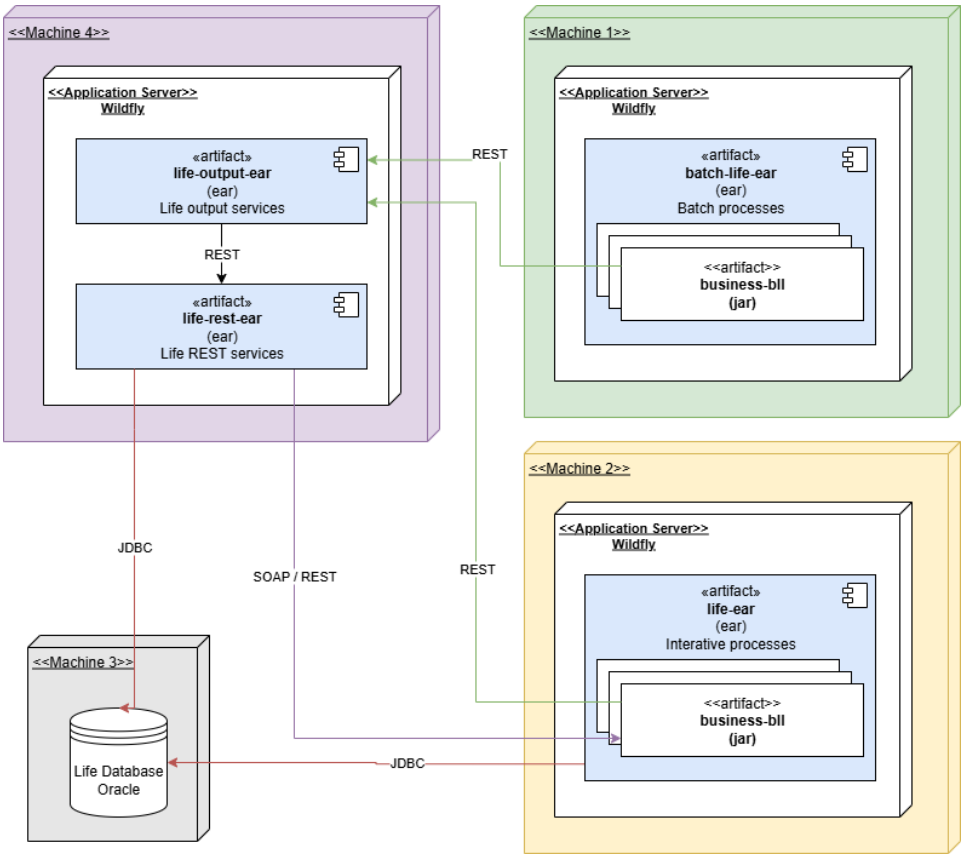



Figura 1.1: Representação da Arquitetura Atual do Sistema

Nesta arquitetura, a *Machine 1* representa uma instância física ou virtual da aplicação *batch-life-ear*, enquanto a *Machine 2* representa uma instância da *life-ear*. Existem ainda duas máquinas adicionais: a *Machine 3*, que aloja simultaneamente os componentes *life-output-ear* e *life-rest-ear*, partilhando os respetivos recursos computacionais e a *Machine 4* que contém a base de dados Oracle. Importa referir que todos os artefactos são executados em *Application Server Wildfly*.

Este *design* reflete uma abordagem em que os componentes apresentam uma forte dependência entre si, originando múltiplas interações remotas que aumentam a complexidade do sistema e reduzem significativamente a sua eficiência.

Com o intuito de apresentar uma visão detalhada do ecossistema em análise, identificam-se de seguida os principais componentes e respetivas responsabilidades na arquitetura atual:

- **life-ear:** Responsável pela gestão de diversos processos e documentos da seguradora em ambientes interativos. Sempre que é necessário despoletar a impressão de um documento, este componente envia o respetivo pedido de emissão através da camada de negócio *business-bbl*.
- **batch-life-ear:** Responsável pela gestão de processos e documentos não interativos, associados a execuções em *batch*. Tal como o *life-ear*, também este componente é responsável por desencadear pedidos de emissão de documentos através da camada de negócio *business-bbl*.
- **life-output-ear:** Tem como principal responsabilidade a construção da informação estruturada dos documentos finais, atualmente apenas no formato *Extensible Markup Language* (XML). Para tal, efetua múltiplas chamadas ao *life-rest-ear*, com o objetivo de obter as várias subpartes dos dados necessários ao processamento dos documentos. Esta dependência de múltiplas chamadas remotas contribui para um aumento significativo dos tempos de resposta e compromete a eficiência global do processo.
- **life-rest-ear:** Responsável pela recolha dos dados necessários ao processamento documental, maioritariamente através de consultas diretas à base de dados. No entanto, existem casos em que é necessário invocar serviços do componente principal (*life-ear*) para complementar a informação recolhida, o que origina uma dependência cíclica. Esta interdependência entre componentes torna o *life-rest-ear* particularmente suscetível a atrasos e falhas de comunicação.

Embora a arquitetura atual seja distribuída, o *design* não tira pleno partido das vantagens esperadas de modularidade e escalabilidade. Em vez disso, o alto acoplamento entre os componentes introduz entraves operacionais que dificultam a evolução do sistema e o alinhamento com as necessidades do negócio e evidencia várias limitações deste tipo de arquitetura. Primeiramente, existe uma dependência excessiva entre os componentes, dado que cada um depende fortemente dos outros para desempenhar as suas funções, o que aumenta o risco de falhas e compromete a autonomia do sistema. Além disso, a necessidade de múltiplas interações remotas, como chamadas *HyperText Transfer Protocol* (HTTP) e *Simple Object Access Protocol* (SOAP) entre os componentes, aumenta significativamente a latência e reduz a eficiência do sistema, especialmente em cenários de alta solicitação. Por fim, a arquitetura apresenta falta de resiliência, pois a indisponibilidade de qualquer componente ao longo do fluxo de processamento pode comprometer todo o sistema, evidenciando a sua fragilidade estrutural.

### 1.3.2 Desempenho Insuficiente

O tempo necessário para gerar e enviar documentos é inadequado para atender às exigências do mercado, especialmente durante períodos de alta solicitação. Este problema compromete diretamente a eficiência operacional e prejudica a experiência dos clientes.

A análise do processo atual demonstra que grande parte do tempo de execução é despendida na recolha de dados necessários à geração dos documentos. No caso de emissão de apólices, por exemplo, o processo limita-se essencialmente a identificar a apólice da base de dados, realizar pequenas validações de consistência e, de seguida, invocar o mecanismo de emissão. No entanto, esta recolha de dados envolve múltiplos acessos remotos e operações dispersas, o que adiciona latência significativa.

Estas limitações tornam-se particularmente evidentes em cenários de muita solicitação, onde o sistema enfrenta dificuldades para processar requisições de forma eficiente. Este problema não apenas afeta a capacidade de resposta, mas também limita a escalabilidade da solução.

Com o objetivo de medir o desempenho, foi realizada uma análise aos registos dos tempos armazenados numa base de dados amplamente utilizada durante os processos de certificação e licenciamento do *software* da empresa. Embora não seja possível medir isoladamente o tempo de recolha e envio de dados, a observação de um processo específico — emissão em massa de apólices — demonstra que, num universo de 1000 documentos, o tempo médio de processamento é de aproximadamente 2,83 segundos, sendo expectável que a maior parte deste tempo esteja associada à recolha e preparação dos dados.

### 1.3.3 Custos Elevados de Manutenção

A manutenção do sistema é complexa e onerosa, sendo significativamente influenciada por uma estrutura rígida para a produção de um documento e pela falta de automação em processos de configuração. Além disso, o alto acoplamento entre módulos contribui para um aumento nos custos operacionais e no esforço necessário para implementar alterações.

Entre as principais dificuldades, destaca-se a dependência de intervenção manual, onde as alterações no sistema frequentemente requerem envolvimento direto do departamento de desenvolvimento de *software*, aumentando o tempo necessário e os custos associados às tarefas de manutenção.

Adicionalmente, existe uma limitação importante na customização, especialmente na sintaxe associada aos atributos de cada documento. A estrutura e a hierarquia dos documentos são fixas o que impede a adaptação do sistema sem modificações no código-fonte. A ausência de mecanismos configuráveis que permitam ajustes, contribui para reduzir consideravelmente a flexibilidade do sistema.

Estes fatores combinados comprometem a evolução do sistema, tornando-o menos eficiente e dificultando a sua adaptação às necessidades em constante transformação do mercado.

### 1.3.4 Compatibilidade e Suporte a Múltiplos Formatos

A capacidade limitada de lidar com múltiplos formatos de documentos é uma barreira significativa no sistema atual. A falta de abstração dificulta o suporte a diferentes formatos de saída sem que sejam necessários ajustes manuais.

O sistema enfrenta dificuldades de integração com versões anteriores do *software*. Esta limitação não apenas aumenta a complexidade do desenvolvimento, mas também compromete a interoperabilidade com outras soluções. Além disso, a falta de flexibilidade para lidar com formatos alternativos reduz a capacidade do sistema de atender às necessidades específicas de clientes e parceiros, dificultando a customização de soluções.

### 1.3.5 Impacto no Negócio

As limitações técnicas e operacionais do sistema têm implicações diretas e significativas no desempenho da organização. A rigidez do *design* impede a adaptação rápida às necessidades dos clientes e às mudanças no mercado, comprometendo a capacidade de entrega de soluções inovadoras e personalizadas.

Estas restrições afetam a competitividade da organização, uma vez que a falta de flexibilidade dificulta a resposta eficiente às solicitações do mercado. Adicionalmente, os atrasos na geração de documentos e as falhas de compatibilidade podem resultar em penalidades legais e prejudicar a reputação da organização no mercado.

## 1.4 Objetivos

Este projeto tem como principal objetivo o desenvolvimento de uma nova solução para o módulo de recolha e envio de dados, que supere as limitações da arquitetura atual.

A proposta visa melhorar a eficiência, escalabilidade e adaptabilidade do sistema, atendendo às necessidades específicas do setor segurador e proporcionando uma base tecnológica robusta para responder às exigências atuais e futuras.

Os objetivos delineados na Tabela 1.1 não se restringem a resolver as necessidades imediatas do sistema atual, mas procuram também dotar a organização de capacidade para enfrentar desafios futuros. A implementação de uma solução sólida e flexível assegurará que a empresa acompanhe as transformações do setor segurador, reforçando a sua posição no mercado e a sua capacidade de adaptação.

Além disso, cada objetivo pretende mitigar diretamente os impactos no negócio identificados na Secção 1.3. Esses impactos incluem a limitação na resposta às necessidades dos clientes, a redução da competitividade e os riscos de não conformidade.

Tabela 1.1: Lista de Objetivos do Projeto

| Código | Descrição do Objetivo |
|---|---|
| OBJ-01 | Minimizar o custo de desenvolvimento de novas funcionalidades e o esforço necessário para suporte entre versões, contribuindo para melhorar a competitividade da organização. |
| OBJ-02 | Minimizar o tempo necessário para a geração e envio de documentos, garantindo eficiência e escalabilidade, de forma a melhorar a resposta às necessidades dos clientes e aumentar a competitividade. |
| OBJ-03 | Maximizar a compatibilidade do sistema, assegurando o suporte a múltiplos formatos de documentos e interoperabilidade entre versões, mitigando os riscos de não conformidade e melhorando a flexibilidade do sistema. |

Uma análise detalhada de cada objetivo é apresentada a seguir na Subsecção 1.4.1, com ênfase na sua relação com os problemas identificados na Secção 1.3. Essa análise procura demonstrar como cada objetivo contribui diretamente para a mitigação dos desafios enfrentados pela organização.

Complementarmente, foram definidos critérios de comparação entre tecnologias na Subsecção 1.4.2, que orientam a escolha da solução mais adequada ao projeto, garantindo alinhamento com os objetivos estabelecidos e as necessidades identificadas.

### 1.4.1 Especificidades dos Objetivos

O objetivo OBJ-01 consiste em minimizar o custo de desenvolvimento de novas funcionalidades e o esforço necessário para suporte entre versões. Este objetivo aborda diretamente as limitações da arquitetura identificadas na Secção 1.3. A arquitetura atual dificulta a introdução de novas funcionalidades e exige intervenções extensas para ajustes ou manutenção, resultando em custos elevados. Para superar esses desafios, pretende-se implementar uma arquitetura flexível, permitindo maior agilidade na adaptação a novos requisitos e redução de custos operacionais. Esta solução visa reduzir o esforço técnico necessário, promovendo um desenvolvimento mais eficiente e sustentável.

O OBJ-02 visa minimizar o tempo necessário para a geração e envio de documentos, um aspeto crítico no setor segurador devido ao elevado volume de emissões diárias. Este objetivo responde ao problema de desempenho insuficiente destacado na Secção 1.3, em que a lentidão no processamento de documentos compromete a produtividade. A resolução dos problemas relacionados ao fluxo de dados e ao envio de grandes volumes de informação permite uma melhoria significativa na eficiência operacional e na rapidez de entrega dos documentos.

Finalmente, o OBJ-03 tem como foco maximizar a compatibilidade do sistema, assegurando o suporte a múltiplos formatos de documentos e interoperabilidade entre versões. Este objetivo trata diretamente a falta de abstração para múltiplos formatos e da incompatibilidade com sistemas legados, conforme descrito na Secção 1.3, pois responde à necessidade de flexibilidade no setor segurador, onde diferentes clientes exigem documentos em formatos variados. A adoção de técnicas que abstraiam a gestão de formatos permitirá ao sistema suportar novos requisitos sem a necessidade de customizações frequentes. Além disso, a compatibilidade retroativa será tratada como uma prioridade, garantindo que a transição para a nova solução preserve a continuidade das funcionalidades existentes e atenda às expectativas dos clientes, mitigando os riscos de não conformidade.

### 1.4.2 Critérios de Comparação entre Tecnologias

Para a seleção da tecnologia mais adequada ao projeto, foi decidido estudar as ferramentas do estado da arte utilizando as seguintes métricas, alinhadas aos objetivos estabelecidos e às necessidades previamente identificadas:

- **Aprendizagem:** Este critério avalia a facilidade de adoção da tecnologia, especialmente em contextos de equipas com diferentes níveis de experiência técnica, assegurando uma curva de aprendizagem suave para os programadores.
- **Custo:** Consideram-se os custos relacionados ao esforço necessário para a implementação e manutenção da solução, incluindo o tempo investido pelas equipas de desenvolvimento e suporte.

- **Desempenho:** Mede a velocidade e eficiência da tecnologia na geração de conteúdos dinâmicos, fator essencial para cumprir as exigências de escalabilidade e qualidade.
- **Escalabilidade:** Verifica a capacidade da tecnologia de lidar com volumes crescentes de dados e requisitos mais complexos, garantindo a adaptabilidade futura da solução.
- **Flexibilidade:** Refere-se à possibilidade de personalizar e gerar resultados em múltiplos formatos, diretamente relacionado com o objetivo OBJ-03, que destaca a importância da compatibilidade e suporte a diferentes formatos.
- **Implementação:** Analisa a facilidade de integração da tecnologia ao ecossistema existente.
- **Segurança:** Avalia a capacidade da tecnologia de mitigar riscos de segurança, protegendo os dados e a integridade do sistema durante sua operação.
- **Suporte:** Considera a qualidade da documentação técnica e a existência de uma comunidade ativa, fatores essenciais para garantir apoio contínuo e acesso a boas práticas.
- **Usabilidade:** Examina a qualidade e clareza das ferramentas oferecidas para programadores, de modo a simplificar o desenvolvimento e melhorar a produtividade.

Todos os critérios foram considerados com pesos iguais, pois são igualmente relevantes para atender aos objetivos do projeto e garantir que a solução escolhida seja robusta, eficiente e escalável.

Complementarmente, e fulcral para este estudo, todas as tecnologias a ser consideradas são necessariamente *open-source*, uma vez que esta é uma imposição da empresa para evitar custos com licenciamento.

## 1.5 Considerações Éticas

A ética desempenha um papel central em todas as fases deste trabalho, orientando as decisões e práticas durante a sua execução. O projeto foi conduzido em conformidade com o Código de Conduta do Instituto Politécnico do Porto (IPP) e com o *Software Engineering Code of Ethics*, assegurando a integridade, transparência e qualidade dos resultados (DR 2020).

Conforme o Código de Conduta do IPP, foram respeitadas as normas que garantem a integridade académica, incluindo a correta citação de fontes, a originalidade do trabalho e a apresentação fiel de dados e conclusões. Este compromisso ético foi formalizado através da declaração de compromisso, assegurando que o trabalho segue os mais elevados padrões de autenticidade e transparência.

Durante a condução do projeto, foram implementadas medidas para proteger informações sensíveis, conforme regulamentações aplicáveis, como o Regulamento Geral de Proteção de Dados (RGPD). A recolha e o tratamento de dados respeitaram a privacidade dos envolvidos, sendo utilizados apenas dados simulados ou anonimizados, sempre que possível. Caso informações reais fossem imprescindíveis, teria sido obtido consentimento informado, garantindo que todas as partes interessadas compreendessem e aprovassem o uso de seus dados.

Os princípios éticos estabelecidos pelo *Software Engineering Code of Ethics* guiaram o desenvolvimento deste projeto, com destaque para a honestidade e integridade na condução das práticas, garantindo que todas as informações e resultados apresentados são verdadeiros e livres de distorções (Gotterbarn, Miller e Rogerson 1997). A comunicação clara e aberta com os *stakeholders* foi priorizada, assegurando transparência sobre os objetivos, métodos e impactos do trabalho, promovendo um ambiente de colaboração e confiança. Além disso, medidas foram tomadas para evitar conflitos de interesse, garantindo que todas as decisões e avaliações fossem conduzidas de forma imparcial, mesmo considerando a relação profissional do autor com a organização envolvida. Por fim, o projeto considerou cuidadosamente os impactos sociais, culturais e legais, alinhando-se às necessidades da sociedade e às melhores práticas da área, promovendo uma abordagem ética e responsável.

Além disso, a transparência e a acessibilidade da solução desenvolvida foram reforçadas pelo uso de tecnologias *open-source*, assegurando que o trabalho pudesse ser reproduzido e verificado por outros profissionais. Todas as ferramentas e metodologias empregadas foram documentadas, permitindo que os resultados sejam auditáveis e validados.

Ao adotar uma abordagem ética rigorosa, este projeto reflete o compromisso com os padrões de excelência académica e profissional, garantindo que o seu impacto seja positivo e alinhado às melhores práticas da engenharia de *software* e da ética aplicada.

## 1.6 Planeamento

A fase de planeamento é uma etapa essencial para o sucesso de qualquer projeto, permitindo organizar as atividades, identificar dependências e monitorizar o progresso.

No contexto deste trabalho, foram seguidas boas práticas de gestão de projetos para garantir uma execução eficiente e alinhada com os objetivos estabelecidos. Este planeamento foi conduzido em três etapas principais: elaboração do *Project Charter*, criação da Estrutura Analítica do Projeto (*Work Breakdown Structure* (WBS)) e inserção das tarefas numa ferramenta bastante conhecida no ramo da gestão de projetos *Microsoft Project*.

### 1.6.1 Metodologia de Planeamento

A primeira etapa do planeamento consistiu na elaboração do *Project Charter*, onde se identificou o âmbito, descreveu os objetivos do projeto, os principais entregáveis, os *stakeholders* envolvidos e as restrições identificadas. Este documento foi essencial para alinhar as expectativas e definir os critérios de sucesso, encontrando-se publicado, na íntegra, no Apêndice A.

Posteriormente, foi desenvolvida a WBS, onde se detalharam as atividades necessárias para a realização do projeto. Este documento auxiliar está presente no Apêndice B. Desta forma, permitiu organizar o trabalho em tarefas menores e mais geríveis, facilitando a identificação de dependências e o controlo do projeto.

Por fim, todas as tarefas identificadas na WBS foram inseridas no *Microsoft Project*, que permitiu definir o cronograma, atribuir recursos e monitorizar o estado das atividades. Este planeamento garantiu que o projeto fosse conduzido de forma estruturada e eficiente.

### 1.6.2 Cronograma do Projeto

Com base no planeamento realizado, foi possível detalhar as atividades, a sua duração e as respetivas datas de início e término. O cronograma completo está apresentado na Tabela 1.2, proporcionando uma visão clara da organização temporal do projeto.

Tabela 1.2: Cronograma do Projeto

| Fase | Atividade | Início | Término |
|---|---|---|---|
| **1** | **Planeamento** | **16/09/24** | **15/11/24** |
| 1.1 | Formalização | 16/09/24 | 16/10/24 |
| 1.1.1 | Determinação dos objetivos específicos | 16/09/24 | 16/10/24 |
| 1.1.2 | Identificação dos *stakeholders* e definição de expetativas | 16/09/24 | 16/10/24 |
| 1.1.3 | Definição dos principais entregáveis e critérios de sucesso do projeto | 16/09/24 | 16/10/24 |
| 1.2 | Planificação | 17/10/24 | 15/11/24 |
| 1.2.1 | Planeamento das principais datas e *milestones* | 17/10/24 | 15/11/24 |
| 1.2.2 | Alocação dos recursos técnicos necessários | 17/10/24 | 30/10/24 |
| **2** | **Estado da Arte e Revisão da Literatura** | **17/10/24** | **04/01/25** |
| 2.1 | Pesquisa de *frameworks*, metodologias e tecnologias relevantes para o projeto | 17/10/24 | 04/01/25 |
| 2.2 | Identificação de soluções semelhantes no mercado | 17/10/24 | 04/01/25 |
| 2.3 | Relatório intermédio | 17/10/24 | 04/01/25 |
| **3** | **Análise e Design** | **06/01/25** | **17/03/25** |
| 3.1 | Análise do Problema e Definição dos Requisitos | 06/01/25 | 17/01/25 |
| 3.1.1 | Análise detalhada dos problemas do sistema atual | 06/01/25 | 17/01/25 |
| 3.1.2 | Identificação e documentação dos requisitos funcionais e não funcionais | 06/01/25 | 17/01/25 |
| 3.2 | *Design* da Arquitetura do Sistema | 20/01/25 | 17/03/25 |
| 3.2.1 | Criação de uma estrutura modular para facilitar a escalabilidade e manutenibilidade | 20/01/25 | 03/02/25 |
| 3.2.2 | Definição da camada de recolha e envio de dados, estrutura dos módulos e fluxo de dados | 04/02/25 | 24/02/25 |
| 3.2.3 | Planeamento da interface para configuração e personalização de *templates* pelos clientes | 25/02/25 | 17/03/25 |
| **4** | **Desenvolvimento** | **18/03/25** | **19/05/25** |
| 4.1 | Desenvolvimento das funcionalidades principais | 18/03/25 | 24/04/25 |
| 4.2 | Desenvolvimento da ferramenta de migração de sistemas já existentes | 25/04/25 | 15/05/25 |
| 4.3 | Desenvolvimento da interface gráfica | 25/04/25 | 19/05/25 |
| **5** | **Testes e Validação** | **20/05/25** | **09/06/25** |
| 5.1 | Testes de coerência e funcionalidade | 20/05/25 | 02/06/25 |
| 5.2 | Testes de performance | 20/05/25 | 02/06/25 |
| 5.3 | Recolha de *feedback* de *stakeholders* | 03/06/25 | 09/06/25 |
| **6** | **Conclusões** | **10/06/25** | **23/06/25** |
| 6.1 | Compilação documentação técnica, testes e resultados | 10/06/25 | 16/06/25 |
| 6.2 | Identificação de melhorias futuras | 10/06/25 | 16/06/25 |
| 6.3 | Revisão e conclusão do relatório | 17/06/25 | 23/06/25 |

| Fase | Atividade | Início | Término |
|---|---|---|---|
| **7** | **Relatório / Entrega Final** | **06/01/25** | **30/06/25** |
| 7.1 | Relatório final | 06/01/25 | 27/06/25 |
| 7.2 | Demonstração e passagem de conhecimento interno | 30/06/25 | 30/06/25 |

Importa ainda salientar que todas as fases foram desenvolvidas de forma sequencial, com exceção da elaboração do relatório. Durante a fase de Estado da Arte e Revisão da Literatura foi produzido um relatório intermédio, enquanto a preparação do relatório final teve início em paralelo com a fase de Análise e *Design*, prolongando-se até ao final.

Como complemento, este cronograma também está representado no formato de *Gantt Chart*, permitindo uma visualização clara da sequência temporal das atividades e das relações de precedência entre elas. Devido à sua extensão, o *Gantt Chart* completo está incluído no Apêndice C.

### 1.6.3 Riscos do Projeto

No planeamento do projeto, foram identificados os principais riscos que poderiam impactar negativamente os seus resultados. Esses riscos foram analisados e foram estabelecidas estratégias específicas para mitigá-los ou evitá-los. A Tabela 1.3 apresenta um resumo desses riscos, incluindo as suas probabilidades, impactos e estratégias de resposta. O detalhe desta análise está no Apêndice D.

Tabela 1.3: Resumo dos Principais Riscos do Projeto.

| Código | Risco | Probabilidade | Impacto | Estratégia de Resposta |
|---|---|---|---|---|
| RIS-01 | Atraso no cronograma | Alta | Alta | Repriotizar tarefas, retirar tarefas menos importantes do âmbito e replanear cronograma |
| RIS-02 | Indisponibilidade de recursos técnicos | Média | Muito Alta | Implementar redundâncias |
| RIS-03 | Problemas de desempenho | Alta | Alta | Investir na otimização de arquitetura e monitorização |
| RIS-04 | Risco de segurança | Alta | Muito Alta | Realizar auditorias regulares através de ferramentas automáticas |
| RIS-05 | Dificuldades com testes de compatibilidade | Baixa | Média | Priorizar a resolução de falhas críticas |

O risco identificado como RIS-01 refere-se ao atraso no cronograma, geralmente causado por mudanças nos requisitos ou dificuldades técnicas ao longo do desenvolvimento. Esse atraso pode comprometer a entrega final do projeto e impactar os prazos acordados. Para mitigar esse risco, foi definido que seria feita uma redefinição das prioridades, identificando tarefas que possam sair deste âmbito. Consequentemente, seria feito um replaneamento do cronograma.

Já o risco RIS-02 trata da indisponibilidade de recursos técnicos, que pode ser causada por falhas ou problemas de disponibilidade na infraestrutura. Os recursos alvo deste risco são a base de dados, o código fonte que é disponibilizado pela empresa ou falta de licenças em *software* usado para os testes. Esse risco pode levar a interrupções no desenvolvimento ou atrasos nos testes, afetando o cronograma do projeto. Para mitigá-lo, optou-se por implementar redundâncias dos recursos acima mencionados: *backup* local da base de dados e do código fonte.

O risco RIS-03 refere-se a problemas de desempenho, que podem ser causados por uma arquitetura não otimizada ou sobrecarga do sistema. Esses problemas podem impactar a eficiência geral do sistema e reduzir a satisfação dos utilizadores. A estratégia de resposta envolve investir na otimização da arquitetura e no uso de ferramentas de monitorização para evitar esses impactos.

O risco identificado como RIS-04 está associado à segurança dos dados, em função de vulnerabilidades no sistema. Este risco pode levar a violações de dados e consequências legais severas. Para mitigá-lo, serão realizadas auditorias regulares de segurança, através de ferramentas automáticas de análise de código e de dependências. SonarQube e OWASP Dependency-Check são exemplos de ferramentas a serem usadas.

Por fim, o risco RIS-05 está relacionado a dificuldades com testes de compatibilidade, causadas pela diversidade de formatos e versões que precisam ser suportados. Esse problema pode afetar a experiência do utilizador final devido a falhas de compatibilidade. A decisão passa por garantir que o *software* é configurável ao ponto de permitir ativar a solução anterior e que desta forma exista tempo para solucionar eventuais problemas.

Com a identificação e tratamento adequado desses riscos, o projeto está mais preparado para enfrentar adversidades e garantir que os objetivos sejam alcançados com sucesso.

### 1.6.4 Conclusão do Planeamento

O planeamento detalhado apresentado neste capítulo permitiu organizar as atividades e definir prazos realistas para cada etapa do projeto. Com base na metodologia aplicada, é possível garantir um controlo eficiente sobre a execução do projeto, mitigar riscos e assegurar que os objetivos definidos sejam alcançados no prazo estipulado. O uso de ferramentas como o *Microsoft Project* foi essencial para acompanhar o progresso e ajustar eventuais desvios no cronograma, garantindo a entrega de uma solução robusta.

## 1.7 Abordagem de Desenvolvimento

A abordagem de desenvolvimento adotada para este projeto baseia-se na estratégia *Design and Creation*, combinada com a metodologia Scrum, uma *framework* ágil amplamente utilizada no desenvolvimento de *software*.

### 1.7.1 Design and Creation

A metodologia *Design and Creation* foi escolhida como estratégia para definir como a solução deveria ser projetada e desenvolvida. Saltuk e Kosan (2014) fundamentam-se na construção sistemática de um artefacto com o objetivo de resolver problemas específicos e gerar novos conhecimentos. No contexto deste projeto, estabeleceu-se algumas etapas, nomeadamente as descritas abaixo:

- **Levantamento de requisitos**:
  - O processo começa com o desenvolvimento de um conceito para o problema baseado nos requisitos e problemas identificados no sistema atual.
  - Definição de uma arquitetura projetada para resolver as limitações de flexibilidade, escalabilidade e compatibilidade, usando princípios de *design* centrado no utilizador.
- **Conceção**:
  - A implementação da solução segue o *design* proposto, resultando na construção de um artefacto.
  - O desenvolvimento realizado de forma incremental, com entregas frequentes de versões parciais do sistema, promovendo a validação contínua.
- **Validação**:
  - A avaliação incluiu a observação do desempenho da solução através de testes técnicos.
  - Dados obtidos durante a avaliação usados para ajustar a solução e garantir que os objetivos definidos são atendidos.
- **Conclusões**:
  - Os resultados da avaliação sofrem uma análise detalhada para entender como o artefacto contribuiu para resolver os problemas identificados.
  - As conclusões também abordam como os resultados suportaram ou refutaram as premissas iniciais da solução proposta.

### 1.7.2 Framework Scrum

O Scrum permite uma organização estruturada e iterativa, garantindo que o progresso seja monitorizado de forma contínua e que o sistema seja entregue de forma incremental (Scrum-Guides 2024). A figura 1.2 (ONES 2024) representa o funcionamento genérico desta metodologia.

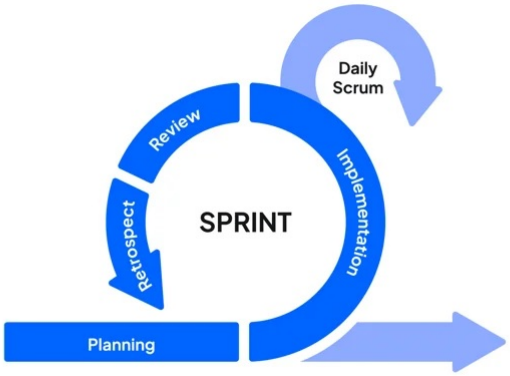



Figura 1.2: Scrum

A escolha do Scrum como metodologia de trabalho baseia-se no facto de esta ser amplamente utilizada na empresa, o que facilita a adaptação às práticas e ferramentas já estabelecidas no ambiente organizacional.

Apesar de o Scrum ser tipicamente implementado em equipas, os seus princípios e práticas foram adaptados para um cenário em que o desenvolvimento será conduzido individualmente. O desenvolvimento está estruturado em ciclos iterativos e incrementais, denominados por *sprints*, cada um com a duração de duas semanas. Cada *sprint* é composto pelo planeamento, onde são definidas as tarefas a desenvolver, com base numa lista priorizada de requisitos. No final de cada iteração, o progresso será avaliado e serão realizados os devidos ajustes ao plano, promovendo uma adaptação contínua às necessidades do projeto.

### 1.7.3 Ferramentas e Controlo de Versões

O controlo de versões e o armazenamento do código são realizados utilizando ferramentas disponibilizadas pela empresa. As submissões são efetuadas de forma regular, acompanhadas por mensagens descritivas, como a implementação de funcionalidades e a correção de erros. Esta abordagem assegura a rastreabilidade de todas as alterações.

A aplicação de uma metodologia ágil facilita a organização e otimização do fluxo de trabalho, promovendo entregas incrementais e garantindo que os objetivos são atingidos, mantendo alinhamento contínuo com as necessidades identificadas ao longo do desenvolvimento.

## 1.8 Gestão de Competências

A gestão de competências é um fator determinante para o sucesso de projetos de desenvolvimento de *software*, especialmente em contextos complexos, como o setor segurador. A identificação e o alinhamento das competências técnicas e organizacionais necessárias para a execução deste projeto são essenciais para garantir a entrega de resultados de alta qualidade dentro dos prazos estabelecidos.

Este projeto, executado por um único elemento, exige uma ampla gama de competências técnicas e de gestão. No domínio técnico, é fundamental o domínio das linguagens de programação que serão alvo do desenvolvimento, bem como das ferramentas de desenvolvimento mais recentes e modernas. Além disso, conhecimentos sólidos em arquitetura de *software* e/ou integração de sistemas são essenciais para atender aos requisitos de flexibilidade, escalabilidade e compatibilidade identificados.

Adicionalmente, foi adotada uma metodologia ágil, baseada no Scrum que força a definição de ciclos curtos de trabalho com duração de duas semanas, promovendo o planeamento quinzenal e a realização de retrospetiva no final de cada ciclo. Esta abordagem garante a reflexão constante sobre os progressos realizados, identificando áreas de melhoria e adaptando o planeamento às mudanças de requisitos ou prioridades. A combinação das competências técnicas com a gestão ágil permite enfrentar os desafios do desenvolvimento e assegurar o sucesso do projeto.

Com base nas necessidades identificadas, foi elaborado um plano de ação para priorizar as competências essenciais ao projeto. Este plano foca-se no desenvolvimento de práticas organizacionais, alinhadas com a metodologia ágil e os objetivos do projeto. A Tabela 1.4 apresenta as principais competências identificadas e as ações correspondentes.

Tabela 1.4: Plano de Ação para Gestão de Competências

| Código | Competência | Ação |
|---|---|---|
| A-01 | Planeamento e Organização | Utilizar ferramentas de gestão de projetos, para monitorizar tarefas, prazos e priorizar atividades, dentro dos ciclos. |
| A-02 | Resolução de Problemas | Implementar estratégias para identificar e mitigar riscos técnicos e organizacionais, promovendo soluções eficientes durante os ciclos quinzenais. |
| A-03 | Melhoria Contínua | Garantir que é feita a retrospetiva no final de cada *sprint* para identificar os pontos a melhorar e implementar mudanças nos ciclos seguintes. |

O plano de ação apresentado reflete o alinhamento entre as competências necessárias e as ações práticas para a sua implementação. A utilização de ferramentas de planeamento, aliada a estratégias de resolução de problemas e práticas de melhoria contínua, assegura uma abordagem estruturada e eficaz. Este plano não só facilita a gestão das tarefas e dos prazos, como também promove a adaptação às mudanças ao longo do desenvolvimento, contribuindo para a entrega contínua de valor e o cumprimento dos objetivos definidos.

## 1.9 Estrutura do Documento

Este documento está estruturado em seis capítulos, organizados de forma a refletir as etapas de investigação, desenvolvimento, validação e conclusão deste trabalho.

No Capítulo 1, **Introdução**, é apresentado o enquadramento geral do projeto, incluindo a descrição da problemática, a relevância do tema, os objetivos propostos, as considerações éticas seguidas e o planeamento inicial das atividades. Este capítulo estabelece o contexto e a importância do trabalho realizado.

O Capítulo 2, **Estado da Arte**, aborda os conceitos teóricos, as metodologias e tecnologias relevantes, bem como as ferramentas atualmente utilizadas na organização. Este capítulo tem como objetivo contextualizar o trabalho no estado atual do conhecimento, identificar práticas e ferramentas aplicáveis ao projeto.

O Capítulo 3, **Análise e Design**, inicia-se com a engenharia de requisitos, onde são identificados os atores do sistema, definidos os requisitos funcionais e não funcionais, estabelecido o mapeamento entre objetivos e requisitos e ainda são apresentadas as restrições e dependências. Segue-se a caracterização da arquitetura, incluindo a solução implementada, uma alternativa analisada, a justificação da decisão tomada, a comparação entre arquiteturas, os padrões arquiteturais aplicados, a extensibilidade e manutenção futura, e um resumo crítico da solução. Por fim, a secção de design detalha a estratégia seguida, o mecanismo de *template* e geração, a testabilidade e estratégia de validação e a interface de simulação de documentos.

O Capítulo 4, **Implementação da Solução**, descreve em detalhe a estrutura base, incluindo o serviço de orquestração, a configuração da camada de serviços, a conversão personalizada de tipos, a persistência de *templates* e um exemplo de *template* para apólices. Apresenta

ainda o processo de criação e edição de *templates* e a funcionalidade de visualização dos resultados.

O Capítulo 5, **Testes e Validação**, documenta a preparação do ambiente de testes, a execução dos testes funcionais e de desempenho, bem como a validação dos objetivos definidos face aos resultados obtidos.

Finalmente, o Capítulo 6, **Conclusões**, sintetiza os principais resultados, discute os desafios e limitações encontrados, propõe linhas de trabalho futuro e apresenta uma apreciação final sobre o impacto e relevância da solução implementada.

Os **Apêndices** incluem informações complementares, nomeadamente documentos relevantes que suportam o trabalho descrito nos capítulos principais.

# 2. Estado da Arte

O estado da arte é um processo essencial para compreender e fundamentar os aspetos teóricos e técnicos que sustentam o desenvolvimento de um projeto. Este capítulo começa por apresentar a metodologia de pesquisa aplicada, fundamentada no protocolo *Preferred Reporting Items for Systematic reviews and Meta-Analyses* (PRISMA), garantindo uma revisão sistemática rigorosa e confiável das fontes analisadas.

Em seguida, são detalhadas as ferramentas utilizadas na organização e as suas especificidades. Posteriormente, são explorados temas centrais, como a redução de dependências em arquiteturas distribuídas, a otimização no acesso a bases de dados e a contextualização do conceito de *template engine*.

Por fim, é realizada uma análise das principais tecnologias disponíveis no mercado, estabelecendo as bases para a formulação de hipóteses e a definição de tecnologias adequadas ao desenvolvimento da solução proposta.

## 2.1 Metodologia de Pesquisa

A metodologia utilizada neste estudo é baseada no PRISMA, amplamente reconhecido como um padrão internacional para a condução de revisões sistemáticas e meta-análises. O PRISMA fornece um conjunto de diretrizes para identificar, selecionar, avaliar e sintetizar estudos relevantes de forma rigorosa e transparente (PRISMA 2024). Este estudo tenta garantir que a pesquisa bibliográfica seja conduzida de maneira metódica e que os resultados sejam replicáveis e confiáveis.

O foco principal do uso do PRISMA neste trabalho foi organizar a revisão de literatura sobre soluções flexíveis e escaláveis no contexto de *engines* de *template* em Java, garantindo que todos os estudos relevantes fossem adequadamente analisados e que a abordagem estivesse alinhada com as melhores práticas da área.

### 2.1.1 Questões de Pesquisa

As questões de pesquisa foram desenvolvidas para abordar os principais desafios relacionados à flexibilidade, escalabilidade e compatibilidade das soluções de recolha e envio de dados. Estas questões guiaram todo o processo de revisão sistemática e foram estruturadas conforme a Tabela 2.1.

Tabela 2.1: Questões de Pesquisa

| **Código** | **Questão de Pesquisa** |
|---|---|
| QT-01 | Como tornar a solução flexível e extensível ao ponto de permitir a configuração do *output* através de um *template*, tornando dinâmico o resultado? |
| QT-02 | Qual é o melhor *design* ou quais são as melhores tecnologias para a implementação da nova solução? |
| QT-03 | Como manter a compatibilidade do *software* atual com a nova solução e aumentar os ganhos de desempenho? |

### 2.1.2 Hipóteses

Com base nas questões de pesquisa apresentadas na Tabela 2.1, foram formuladas hipóteses que visam propor soluções ou resultados esperados para os desafios levantados. A Tabela 2.2 apresenta as hipóteses associadas a cada questão de pesquisa.

Tabela 2.2: Hipóteses

| Hipótese / Questão | Descrição |
|---|---|
| HP-01 / QT-01 | Uma arquitetura modularizada e com suporte a *templates* dinâmicos permitirá maior flexibilidade e extensibilidade na configuração da forma de recolha e envio de dados, reduzindo a necessidade de alterações manuais em cada implementação. |
| HP-02 / QT-02 | A aplicação de padrões de *design*, como *Factory* e *Strategy*, juntamente com a utilização de *frameworks* modernas e reconhecidas pela comunidade, resultará numa solução escalável, eficiente e adaptável a múltiplos cenários. |
| HP-03 / QT-03 | A adoção de técnicas de compatibilidade retroativa, como abstrações de formatos e uso de *cache*, garantirá a integração eficiente com sistemas existentes, ao mesmo tempo que otimiza o desempenho do processamento de dados. |

As hipóteses descritas estão diretamente relacionadas às questões de pesquisa e servem como base para a avaliação e validação das soluções propostas neste trabalho. Cada hipótese reflete uma abordagem técnica para resolver os problemas identificados e orienta o desenvolvimento da análise.

### 2.1.3 Fontes de Informação

As fontes de informação utilizadas nesta revisão sistemática foram selecionadas com base na sua relevância, credibilidade e capacidade de fornecer conteúdos científicos e técnicos de alta qualidade. A Tabela 2.3 apresenta as principais fontes utilizadas, incluindo bases de dados académicas de renome, bibliotecas digitais e documentações oficiais, que em conjunto proporcionaram uma base sólida para a análise e desenvolvimento deste trabalho.

Tabela 2.3: Fontes de Informação

| Código | Fonte de Informação |
|---|---|
| FI-01 | Web of Science |
| FI-02 | IEEE Xplore |
| FI-03 | ACM Digital Library |
| FI-04 | SpringerLink |
| FI-05 | Documentação Oficial das *Frameworks* |

Estas fontes representam um conjunto diversificado de materiais, desde bases de dados académicas amplamente reconhecidas até documentações oficiais de tecnologias específicas. A combinação de fontes académicas e técnicas proporcionou uma visão abrangente e fundamentada, permitindo uma análise completa das soluções e tecnologias avaliadas nesta investigação.

### 2.1.4 Termos de Pesquisa

Os termos de pesquisa utilizados foram definidos para abordar as principais áreas de interesse desta investigação, incluindo desempenho, escalabilidade, flexibilidade e compatibilidade de *engines* de *template* em Java. A Tabela 2.4 apresenta os termos utilizados.

Tabela 2.4: Termos de Pesquisa Utilizados

| Código | Termo de Pesquisa |
|---|---|
| TP-01 | "Template engine"' |
| TP-02 | "Java" |
| TP-03 | "JSON" |
| TP-04 | "Java template scalability" |
| TP-05 | "Dynamic templates Java" |
| TP-06 | "Best practices templates" |

Estes termos foram aplicados nas fontes de informação referidas na subsecção 2.1.3. Para obter resultados relevantes, utilizou-se a combinação de palavras-chave com operadores booleanos (`AND`, `OR`) e filtros temporais.

### 2.1.5 Critérios de Inclusão e Exclusão

Os critérios de inclusão e exclusão foram estabelecidos para garantir que apenas estudos relevantes e de alta qualidade fossem selecionados nesta revisão sistemática. As Tabelas 2.5 e 2.6 apresentam, respetivamente, os critérios de inclusão e exclusão definidos para este trabalho.

Tabela 2.5: Critérios de Inclusão

| Código | Descrição |
|---|---|
| CI-01 | Estudos publicados em *peer-reviewed journals* ou apresentados em conferências científicas. |
| CI-02 | Trabalhos que analisam *template engines* como Mustache, Thymeleaf e FreeMarker no contexto de aplicações Java. |
| CI-03 | Artigos que apresentam métricas relacionadas a desempenho, escalabilidade, flexibilidade e/ou compatibilidade retroativa. |
| CI-04 | Estudos publicados nos últimos 5 anos, garantindo a atualidade dos dados. |
| CI-05 | Publicações com metodologias bem definidas e avaliação empírica. |

Tabela 2.6: Critérios de Exclusão

| Código | Descrição |
|---|---|
| CE-01 | Trabalhos sem avaliação empírica ou com metodologia insuficientemente descrita. |
| CE-02 | Estudos que analisam tecnologias desatualizadas ou fora do contexto das *template engines*. |
| CE-03 | Artigos que não apresentam métricas aplicáveis ou análises relevantes para o objetivo deste estudo. |
| CE-04 | Documentos inacessíveis em texto completo nas bases de dados utilizadas. |

Os critérios de inclusão garantem que os estudos selecionados estão alinhados aos objetivos da pesquisa, enquanto os critérios de exclusão filtram trabalhos que não atendem aos requisitos de qualidade e relevância. Esta abordagem assegurou que apenas estudos de alta qualidade fossem utilizados para sustentar as decisões de implementação deste trabalho.

### 2.1.6 Diagrama de Fluxo PRISMA

O processo de seleção dos estudos foi documentado pelo Diagrama de Fluxo PRISMA. Esta abordagem fornece uma estrutura padronizada para relatar revisões sistemáticas, permitindo uma visualização clara e objetiva das etapas realizadas, desde a identificação inicial dos estudos até a inclusão final na análise. O uso deste diagrama garante transparência, aplicabilidade e rigor metodológico.

A Figura 2.1 apresenta o diagrama de fluxo PRISMA, que detalha as seguintes etapas:

- **Identificação:** Estudos foram recolhidos nas fontes de informação (Tabela 2.3) através dos termos de pesquisa (Tabela 2.4) previamente definidos. Foram recolhidos 60 estudos de 5 bases de dados, incluindo artigos e outros registos relevantes. Após esta etapa, foram removidos 22 registos, sendo 9 por duplicação, 10 por serem irrelevantes e 3 por outras razões. Assim, 38 estudos seguiram para a etapa de triagem.
- **Triagem:** Nesta etapa, os 38 documentos de estudo foram avaliados de acordo com os critérios de inclusão e exclusão apresentados nas Tabelas 2.5 e 2.6. Um total de 10 estudos foram excluídos por não atender aos critérios mínimos. As razões mais comuns para exclusão incluíram falta de alinhamento com os objetivos da pesquisa e metodologias insuficientemente descritas. Um total de 5 estudos foram retirados por motivos diversos, incluindo dados insuficientes para análise ou falta de acesso completo aos textos. No final, 23 estudos foram incluídos na revisão sistemática, cumprindo integralmente os critérios de inclusão.
- **Inclusão:** Apenas os estudos que atenderam integralmente aos critérios de inclusão foram considerados para a análise final.

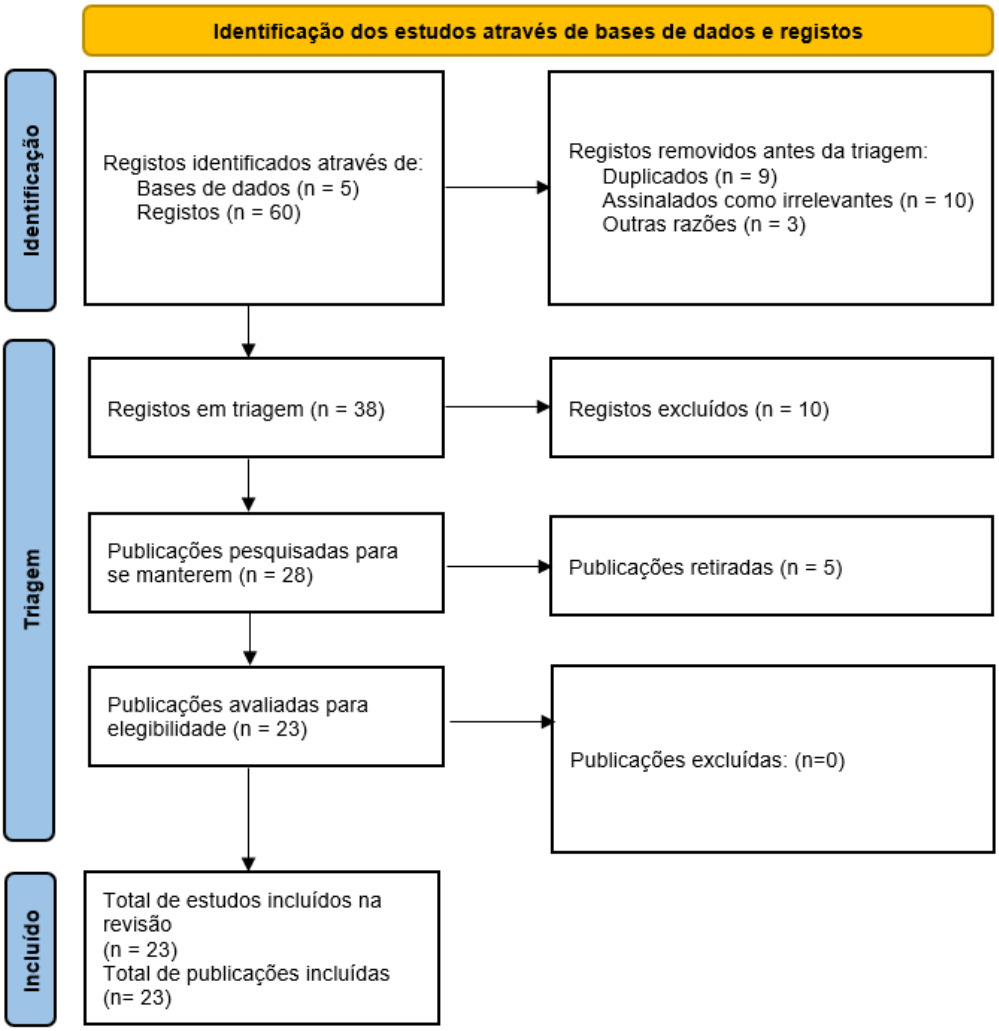



Figura 2.1: Diagrama de Fluxo PRISMA

A utilização deste diagrama assegurou que o processo de seleção dos estudos fosse conduzido com rigor metodológico, facilitando a comunicação clara e transparente das etapas realizadas.

## 2.2 Ferramentas Utilizadas na Organização

No contexto da organização em estudo, são utilizadas diversas ferramentas tecnológicas cruciais para o desenvolvimento de *software*, destacando-se as bases de dados Oracle e as linguagens de programação Java e Angular.

### 2.2.1 Oracle Database

O Oracle Database, desenvolvido pela Oracle Corporation, é amplamente utilizado em ambientes empresariais, destacando-se como um sistema de gestão de bases de dados relacional conhecido pela sua segurança, fiabilidade e robustez. Este sistema de gestão de base de dados apresenta uma arquitetura convergente capaz de lidar com diferentes tipos de dados

e cargas de trabalho, incluindo dados relacionais, gráficos, documentos e semi-estruturados (Oracle Corporation 2024h).

As principais características são:

- **Conformidade com Padrões Structured Query Language (SQL):** O Oracle SQL é compatível com os padrões *American National Standards Institute* (ANSI) SQL, proporcionando uma interface eficiente para acesso, definição e manutenção de dados (Oracle Corporation 2024l).
- **Alta Disponibilidade e Recuperação de Desastres:** Recursos como o Oracle Data Guard e o *Real Application Clusters* (RAC) garantem alta disponibilidade e recuperação eficiente em casos de falhas (Oracle Corporation 2024d).
- **Segurança Avançada:** Inclui funcionalidades como criptografia de dados, controlo de acesso rigoroso e auditorias detalhadas para proteger informações sensíveis (Oracle Corporation 2024k).
- **Desempenho e Escalabilidade:** Oferece recursos como particionamento de tabelas e otimização de consultas para lidar com grandes volumes de dados e múltiplas transações simultâneas (Oracle Corporation 2024i).
- **Suporte a Diversas Linguagens de Programação:** Compatível com linguagens como Java, Python, .NET, entre outras, facilitando a integração com diversas aplicações (Oracle Corporation 2024j).

A Oracle disponibiliza uma documentação abrangente para o Oracle Database, que permite aos utilizadores aprofundarem o conhecimento sobre esta tecnologia. Esta documentação cobre desde os conceitos mais básicos até funcionalidades avançadas, proporcionando um recurso valioso para profissionais que pretendem maximizar a utilização desta ferramenta (Oracle Corporation 2024h).

Ao usar o Oracle Database, é essencial ter em consideração aspetos cruciais como o planeamento de capacidade, a definição de estratégias eficazes de cópias de segurança e recuperação, bem como a adoção de políticas de segurança rigorosas. Seguindo estes elementos é possível garantir a integridade dos dados e assegurar a sua disponibilidade em cenários críticos.

De forma geral, o Oracle Database destaca-se como uma solução robusta e poderosa para a gestão de dados nas organizações. A sua ampla gama de recursos é capaz de atender às exigências de aplicações modernas, tornando-o uma escolha confiável para ambientes empresariais.

### 2.2.2 Java

A linguagem de programação Java é amplamente utilizada no desenvolvimento de aplicações empresariais devido à sua portabilidade, robustez e segurança. Desenvolvida inicialmente pela Sun Microsystems e atualmente mantida pela Oracle Corporation, o Java segue o princípio "escreva uma vez, execute em qualquer lugar"(WORA - *Write Once, Run Anywhere*), graças à *Java Virtual Machine* (JVM) (Oracle Corporation 2024a).

Um das principais pontos fortes do Java é a sua vasta coleção de bibliotecas e *frameworks*, como Spring, Hibernate e JavaFX, que facilitam o desenvolvimento de aplicações de diferentes tipos, desde aplicações *web* até sistemas distribuídos. Além disso, o Java oferece

suporte à programação orientada a objetos, permitindo a criação de sistemas modulares, reutilizáveis e fáceis de manter (Oracle Corporation 2024m).

Outra característica relevante é o foco na segurança e no desempenho. O Java inclui funcionalidades como gestão automática de memória, tratamento de exceções e mecanismos de segurança integrados, tornando-o uma escolha ideal para aplicações críticas (Oracle Corporation 2024c).

Devido à sua compatibilidade com múltiplas plataformas e à ampla utilização em setores como a banca, telecomunicações e comércio eletrónico, o Java continua a ser uma das linguagens mais populares e confiáveis para o desenvolvimento de *software*. Esta versatilidade torna-o uma escolha sólida para organizações que procuram construir sistemas modernos e escaláveis (Oracle Corporation 2024b).

### 2.2.3 Angular

O Angular é uma plataforma de desenvolvimento de código aberto, mantida pelo Google, que facilita a criação de aplicações *web* dinâmicas e interativas. Baseado em TypeScript, o Angular permite a construção de interfaces de utilizador ricas através de uma abordagem modular e orientada a componentes (Google LLC 2024d).

Uma das principais características do Angular é a sua arquitetura baseada em componentes, que promove a reutilização de código e a organização eficiente de projetos. Cada componente encapsula a sua lógica, *template* e estilos, permitindo o desenvolvimento de interfaces visuais modulares e de fácil manutenção (Google LLC 2024a).

O Angular também oferece um conjunto abrangente de ferramentas e bibliotecas oficiais, como o Angular Material, que fornece componentes de interface de utilizador seguindo as diretrizes do *Material Design*. Além disso, o *framework* inclui funcionalidades integradas para gestão de formulários, mapeamento de rotas, injeção de dependências e comunicação com serviços *web*, facilitando o desenvolvimento de aplicações complexas (Google LLC 2024b).

A documentação oficial do Angular disponibiliza tutoriais, guias e referências detalhadas que auxiliam os programadores em todas as etapas do desenvolvimento, desde a configuração inicial até a otimização de aplicações para produção (Google LLC 2024d).

Devido à sua robustez, escalabilidade e suporte contínuo da comunidade, o Angular é amplamente adotado no desenvolvimento de aplicações *web* empresariais, proporcionando uma base sólida para a criação de soluções modernas e eficientes (Google LLC 2024c).

## 2.3 Redução de Dependências em Arquiteturas Distribuídas

A eficiência em arquiteturas distribuídas depende diretamente da minimização das dependências entre os componentes. Sistemas com alta interdependência geram custos elevados em termos de comunicação, latência e escalabilidade.

Uma estratégia abordada por (Badmus 2023) é a introdução de mecanismos otimizados para comunicação entre serviços, com foco na redução de chamadas redundantes através da integração com protocolos mais eficientes, como *Google Remote Procedure Call* (gRPC), em vez do tradicional *Representational State Transfer* (REST). Segundo o estudo, usar gRPC pode reduzir significativamente a latência e o *overhead*, devido à sua capacidade de compressão e transmissão binária.

De acordo com (Newman 2015), uma das práticas mais eficazes para minimizar dependências entre serviços é a implementação de *API Gateways*, que consolidam múltiplas chamadas em uma única interação, funcionando como uma camada intermediária entre clientes e serviços. Este padrão não apenas reduz o número de interações diretas entre os serviços, mas também melhora a segurança e a flexibilidade ao adicionar funcionalidades como autenticação e validação de dados no próprio *gateway*.

Outra abordagem comum, destacada tanto por (Badmus 2023) como por (Newman 2015), é o uso de eventos assíncronos ou filas de mensagens para desacoplar os serviços. Sistemas baseados em eventos, como Kafka ou RabbitMQ, permitem que os componentes se comuniquem de forma indireta e resiliente, evitando bloqueios e aumentando a tolerância a falhas.

Além disso, (Badmus 2023) recomenda a implementação de mecanismos de *cache* distribuída para dados frequentemente obtidos, minimizando interações diretas entre os serviços e reduzindo o impacto de chamadas remotas. Esta prática melhora os tempos de resposta e alivia a carga dos sistemas centrais, proporcionando maior eficiência global.

Por fim, (Newman 2015) também enfatiza a importância de projetar contratos bem definidos entre os serviços, com uso de versões explicitamente documentadas para evitar problemas de compatibilidade. Esta prática reduz dependências acidentais e facilita a manutenção de arquiteturas distribuídas ao longo do tempo.

Estas práticas são fundamentais para arquiteturas que lidam com alto volume de dados e comunicação frequente entre componentes, como o sistema em análise neste projeto.

## 2.4 Otimização em Acessos a Base de Dados

O acesso eficiente a bases de dados é crucial para o desempenho de qualquer sistema. Entre as principais práticas, destaca-se o uso de índices, que são fundamentais para acelerar a recuperação de dados. A criação de índices em colunas frequentemente utilizadas em cláusulas `WHERE` ou em junções (`JOINs`) pode melhorar significativamente o desempenho das consultas, reduzindo o tempo de execução (Oracle Corporation 2024f).

Outra prática recomendada é o particionamento de tabelas, que consiste em dividir grandes tabelas em partições menores. Este método permite que operações de leitura e escrita sejam realizadas de forma mais eficiente, especialmente em ambientes que lidam com grandes volumes de dados. O particionamento, para além de melhorar o desempenho, também facilita a gestão dos dados armazenados (Oracle Corporation 2024f).

Além disso, a otimização de consultas SQL desempenha um papel essencial. Escrever consultas (`SELECT`) de forma eficiente, evitando operações desnecessárias, é uma estratégia chave para reduzir o tempo de execução e o consumo de recursos. Como exemplo, a Oracle fornece ferramentas específicas, como o *SQL Tuning Advisor*, que auxiliam na identificação de melhorias nas consultas, sugerindo alterações que podem aumentar a eficiência (Oracle Corporation 2024g).

O uso de *bind variables* também é amplamente recomendado. Esta técnica reduz a sobrecarga de compilação de consultas e promove a reutilização de planos de execução, resultando num melhor desempenho e menor consumo de recursos. Este recurso é útil em sistemas que processam grandes volumes de transações com consultas semelhantes (Oracle Corporation 2024f).

Por outro lado, a implementação de mecanismos de *caching*, como o *Database Smart Flash Cache*, é uma estratégia eficaz para acelerar o acesso a dados frequentemente consultados. Essa abordagem reduz a carga sobre o sistema de armazenamento principal, melhorando os tempos de resposta e a eficiência geral do sistema (Oracle Corporation 2024f).

Uma técnica adicional que merece destaque é o uso de *materialized views*. As *materialized views* armazenam fisicamente os resultados de uma consulta complexa, permitindo que os dados sejam recuperados de forma muito mais rápida em consultas subsequentes. Estas *views* são especialmente úteis em ambientes de *data warehousing* e aplicações analíticas, onde grandes volumes de dados são processados e consultas repetitivas são comuns. Por exemplo, ao pré-calcular e armazenar agregações ou junções de tabelas, as *materialized views* podem reduzir significativamente a carga sobre a base de dados e os tempos de resposta (Oracle Corporation 2024e).

Adicionalmente, as *materialized views* podem ser configuradas para serem atualizadas automaticamente com base em alterações nos dados subjacentes (*fast refresh*), garantindo que as informações estejam sempre atualizadas sem a necessidade de recalcular a cada consulta. Esta funcionalidade é particularmente valiosa para sistemas que precisam equilibrar a frescura dos dados e o desempenho das consultas (Oracle Corporation 2024e).

A adoção destas práticas, contribui significativamente para a otimização dos acessos a bases de dados, minimizando os tempos de resposta e maximizando o desempenho global, especialmente em contextos com grande volume de dados e consultas frequentes.

## 2.5 Template Engines

*Template Engines* são ferramentas projetadas para gerar documentos dinâmicos, combinando modelos predefinidos (*templates*) com dados fornecidos em tempo de execução. Um *template* é essencialmente um documento que contém uma estrutura fixa e marcadores de posição (*placeholders*), os quais são substituídos pelos dados dinâmicos durante o processamento (Mittapalli e Arthur 2021). Além disso, são suportadas operações lógicas básicas, como condições, ciclos de repetição e filtros para formatação de dados.

O principal objetivo dos *template engines* é separar a lógica da aplicação da apresentação, promovendo maior organização e facilidade de manutenção no código. Eles são amplamente utilizados em aplicações *web*, mas também podem ser aplicados para gerar outros formatos, como *JavaScript Object Notation* (JSON), XML e ficheiros de texto (Harrand et al. 2021).

Os *template engines* também possuem compatibilidade multiplataforma, com implementações disponíveis em diversas linguagens de programação, incluindo Java, Python, JavaScript, PHP e Ruby. Esta característica torna-os versáteis, permitindo a integração em diferentes tecnologias e contextos de desenvolvimento (Sun et al. 2020).

O funcionamento básico dos *template engines* pode ser resumido em três etapas. Inicialmente, um *template* é definido contendo *placeholders* que indicam onde os dados dinâmicos serão inseridos. Em seguida, a *template engine* processa os dados fornecidos, substituindo os *placeholders* pelos valores correspondentes. Por fim, o resultado é um documento renderizado no formato desejado, como *HyperText Markup Language* (HTML), JSON ou XML, contendo os dados já integrados (Dhalla 2020).

Utilizando como exemplo um *template engine* em JSON, este pode ser definido da seguinte forma:

```
{
  "name": "{{myName}}",
  "age": "{{myAge}}"
}
```

Figura 2.2: Exemplo de *Template* JSON

Quando combinado com os dados myName = "Pedro", myAge = 37, o resultado gerado será:

```
{
  "name": "Pedro",
  "age": 37
}
```

Figura 2.3: Exemplo de *Template* - Resultado Gerado

Com esta abordagem, os *template engines* tornam-se ferramentas fundamentais para a construção de sistemas dinâmicos, eficientes e organizados, facilitando a geração de saídas estruturadas e separando claramente as camadas de dados e apresentação.

## 2.6 Tecnologias Existentes no Mercado

Nesta secção, são abordadas as principais tecnologias de *template engines* disponíveis no mercado, focando nas suas características, pontos fortes, desvantagens e contextos de uso. Estas ferramentas fazem a geração de conteúdos dinâmicos ao combinarem modelos predefinidos com dados em tempo de execução. A sua utilização estende-se a diversas áreas, como o desenvolvimento de aplicações *web* e a produção de documentos estruturados em formatos como HTML, XML e JSON (Mittapalli e Arthur 2021).

### 2.6.1 Apache Velocity

O Apache Velocity é um *template engine* orientado a texto, amplamente utilizado no ecossistema Java para gerar conteúdos dinâmicos. Através da sua sintaxe simples e eficiente, esta ferramenta permite a separação entre a lógica da aplicação e a camada de apresentação, facilitando o desenvolvimento colaborativo entre programadores e *designers*. A arquitetura do Velocity baseia-se no conceito de *Model-View-Controller* (MVC), onde o código Java é separado dos *templates*, promovendo uma manutenção mais eficaz e uma melhor organização do código (Apache Software Foundation 2024e).

Uma das principais características do Velocity é a sua leveza e desempenho, tornando-o uma solução adequada para aplicações em que a simplicidade e a rapidez de processamento são essenciais. No entanto, a falta de funcionalidades avançadas pode ser uma limitação para aplicações mais complexas que exigem manipulações sofisticadas de dados ou suporte a macros (F. Carvalho e Duarte 2019). Apesar destas limitações, o Apache Velocity continua a ser amplamente utilizado em sistemas legados e em projetos que priorizam a simplicidade.

Para avaliar o Apache Velocity de forma abrangente, foram considerados os seguintes critérios:

- **Aprendizagem:** É considerado fácil de aprender, sendo adequado tanto para iniciantes quanto para utilizadores avançados (Apache Software Foundation 2024f).
- **Custo:** Além de ser uma ferramenta *open-source*, apresenta custos reduzidos de implementação e manutenção (Apache Software Foundation 2024f).
- **Desempenho:** Possui desempenho médio, eficiente para sistemas simples (Apache Software Foundation 2024f).
- **Escalabilidade:** É limitado em projetos de grande escala (Apache Software Foundation 2024f).
- **Flexibilidade:** Oferece pouca flexibilidade em cenários que exigem operações mais avançadas (Apache Software Foundation 2024f).
- **Segurança:** Requer atenção extra para implementar boas práticas de segurança (Apache Software Foundation 2024f).
- **Implementação:** Simples e rápida no ecossistema Java (Apache Software Foundation 2024f).
- **Suporte:** Documentação suficiente, mas comunidade menos ativa comparada a outras tecnologias (Apache Software Foundation 2024f).
- **Usabilidade:** Simples e intuitivo para uso em projetos pequenos e médios (Apache Software Foundation 2024f).

### 2.6.2 Apache FreeMarker

O Apache FreeMarker destaca-se como um *template engine* robusto e altamente flexível, projetado para criar conteúdos dinâmicos em vários formatos, como HTML, XML e JSON. Esta ferramenta suporta operações avançadas, incluindo ciclos, condições e macros, tornando-o ideal para a geração de documentos complexos e estruturados. A sua arquitetura permite a integração com diversas bibliotecas Java, facilitando a manipulação de dados complexos através de uma linguagem de *templates* especializada, o *FreeMarker Template Language* (FTL) (Apache Software Foundation 2024d).

O grande ponto forte do Apache FreeMarker é a sua capacidade de lidar com dados complexos e integrar-se de forma nativa em aplicações Java modernas, especialmente em *frameworks* como o Spring. Contudo, esta flexibilidade traz consigo uma curva de aprendizagem mais elevada, o que pode dificultar a sua adoção por utilizadores menos experientes. Ainda assim, é amplamente adotado em contextos que requerem um alto nível de personalização e performance (Mittapalli e Arthur 2021).

Os tópicos seguintes analisam o Apache FreeMarker:

- **Aprendizagem:** A curva de aprendizagem é moderada devido ao FTL ser uma linguagem especializada (Apache Software Foundation 2024b).
- **Custo:** Custo reduzido de implementação e manutenção, pois não possui dependências obrigatórias e é *open-source* (Apache Software Foundation 2024a).
- **Desempenho:** Elevado, especialmente em aplicações complexas (Apache Software Foundation 2024b).

- **Escalabilidade:** Altamente escalável, adequado para sistemas de grande complexidade (Apache Software Foundation 2024b).
- **Flexibilidade:** Flexível, com suporte para diversos formatos e manipulação avançada de dados (Apache Software Foundation 2024b).
- **Segurança:** Oferece boas práticas nativas, especialmente em ambientes que requerem proteção de dados (Apache Software Foundation 2024d).
- **Implementação:** Requer algum esforço inicial, mas é altamente integrável com Spring Boot e bibliotecas Java (Apache Software Foundation 2024d).
- **Suporte:** Excelente documentação e uma comunidade ativa (Apache Software Foundation 2024d).
- **Usabilidade:** Tem por base uma sintaxe intuitiva e recursos avançados e é adequado para projetos robustos. Além disso, oferece um simulador de *templates online*, facilitando o desenvolvimento e a validação de *templates* (Apache Software Foundation 2024c).

### 2.6.3 Thymeleaf

O Thymeleaf é um *template engine* moderno e poderoso, especialmente popular no ecossistema Spring. Através do conceito de *natural templates*, permite que os *templates* sejam visualizados diretamente em navegadores, sem necessidade de processamento pelo servidor, facilitando o desenvolvimento e a colaboração entre as equipas. A arquitetura do Thymeleaf é altamente extensível, permitindo a criação de dialetos personalizados que podem adicionar novas funcionalidades ou modificar o comportamento padrão da *engine* (Thymeleaf Team 2024).

Entre as suas principais características, destacam-se o suporte a ciclos, condições, formatação de dados e internacionalização. Contudo, a sua complexidade pode torná-lo mais pesado e menos eficiente para aplicações simples que não exigem recursos avançados (Fernando Carvalho e Fialho 2023).

Para uma avaliação detalhada, são apresentados os seguintes critérios:

- **Aprendizagem:** Fácil de aprender, especialmente para utilizadores do Spring Boot (Thymeleaf Project 2024a).
- **Custo:** Ferramenta *open-source*, com um custo baixo de implementação pela sua integração nativa com o ecossistema Spring, reduzindo assim o tempo necessários das equipas de desenvolvimento e suporte (Thymeleaf Project 2024b).
- **Desempenho:** Médio, com algumas limitações para projetos de alta performance (Thymeleaf Team 2024).
- **Escalabilidade:** Escalável para aplicações *web*, mas limitado para outros contextos (Thymeleaf Team 2024).
- **Flexibilidade:** Muito bom em aplicações *web*, mas com limitações fora deste domínio (Thymeleaf Team 2024).
- **Segurança:** Oferece boas práticas de segurança, especialmente em aplicações baseadas na web (Thymeleaf Project 2024b).

- **Implementação:** Simples em projetos Spring Boot, mas mais complexo para outros contextos (Thymeleaf Project 2024b).
- **Suporte:** Documentação completa e comunidade ativa (Thymeleaf Team 2024).
- **Usabilidade:** Ideal para aplicações *web* dinâmicas e permite a integração eficiente entre o código Java e a camada de apresentação (Rocketseat Blog 2024).

### 2.6.4 Mustache

Mustache é um *template engine* minimalista que segue uma abordagem *logic-less*, promovendo a separação completa entre a lógica da aplicação e a camada de apresentação. A sua sintaxe simples e multiplataforma faz do Mustache uma solução versátil para aplicações que priorizam clareza e facilidade de manutenção. A arquitetura do Mustache é agnóstica em relação ao *framework*, permitindo a sua implementação em diversas linguagens de programação, o que facilita a sua adoção em diferentes ambientes de desenvolvimento (Mustache Community 2024).

Um dos seus maiores pontos fortes é a simplicidade, permitindo aos programadores utilizarem *templates* reutilizáveis sem introduzir complexidade desnecessária. No entanto, esta simplicidade também representa uma limitação em casos que exigem manipulação avançada de dados diretamente nos *templates*. Apesar disso, o Mustache continua a ser amplamente utilizado em aplicações *web* e projetos que beneficiam de uma estrutura limpa e de fácil compreensão (Ardakany et al. 2020).

A análise seguinte avalia o Mustache em relação aos critérios também analisados nas outras tecnologias:

- **Aprendizagem:** Extremamente fácil de aprender, devido à sua sintaxe simples e ausência de lógica complexa (Baeldung 2024).
- **Custo:** Além de ser *open-source*, o custo muito baixo devido à sua simplicidade e rápida integração, o que reduz substancialmente o tempo investido pelas equipas de desenvolvimento e suporte (Baeldung 2024).
- **Desempenho:** Elevado em aplicações simples, proporcionando renderização eficiente de *templates* (Baeldung 2024).
- **Escalabilidade:** Limitado em cenários mais complexos, devido à sua natureza *logic-less* que pode exigir lógica adicional fora dos *templates* (SpringHow 2024).
- **Flexibilidade:** Baixa, devido à sua abordagem *logic-less*, o que limita a capacidade de incorporar lógica nos templates (Baeldung 2024).
- **Segurança:** Básica, com necessidade de medidas complementares para lidar com aspetos como a prevenção de injeção de código (Baeldung 2024).
- **Implementação:** Muito rápida e intuitiva, com suporte a diversas linguagens de programação (Mustache Community 2024).
- **Suporte:** Documentação básica e uma comunidade ativa, mas menor em comparação com outras tecnologias (Mustache Community 2024).
- **Usabilidade:** Simples e claro para casos de uso básicos, ideal para projetos que requerem *templates* sem lógica embutida (Baeldung 2024).

### 2.6.5 Handlebars

O Handlebars é uma extensão do Mustache que introduz funcionalidades adicionais, como o suporte a *helpers* personalizados e blocos condicionais. Estas funcionalidades permitem que os programadores realizem manipulações avançadas nos *templates*, mantendo ao mesmo tempo a simplicidade e a clareza da sintaxe. A arquitetura do Handlebars permite a pré-compilação dos *templates* em funções JavaScript, o que melhora o desempenho ao reduzir o tempo de processamento do lado do cliente (Handlebars Community 2024).

Um dos maiores pontos fortes do Handlebars é a sua flexibilidade, permitindo a reutilização de código e a criação de *templates* mais dinâmicos e robustos. No entanto, o uso excessivo de *helpers* pode complicar a manutenção dos *templates*, especialmente em projetos de grande escala. O Handlebars é amplamente utilizado em aplicações *web* modernas que necessitam de um *template engine* leve, mas funcional (Mardan 2018).

Os critérios de avaliação incluem os seguintes tópicos:

- **Aprendizagem:** Fácil de aprender, com sintaxe clara e direta (Escola de Tecnologia 2024).
- **Custo:** *Open-source*, apresenta baixo custo devido à sua simplicidade e rápida integração, reduzindo significativamente o tempo investido pelas equipas de desenvolvimento e suporte (Portal Micilini 2024).
- **Desempenho:** Elevado, especialmente em aplicações com *templates* pré-compilados (Handlebars Community 2024).
- **Escalabilidade:** Moderado, adequado para aplicações de porte médio e pode ser necessário complementar com outras ferramentas ou *frameworks* (Handlebars.js 2024).
- **Flexibilidade:** Boa flexibilidade devido ao suporte a *helpers* personalizados, permitindo a extensão das funcionalidades dos *templates* (Handlebars Community 2024).
- **Segurança:** Requer medidas adicionais para cenários críticos, como a implementação de validações e sanitizações para prevenir vulnerabilidades (Handlebars.js 2024)
- **Implementação:** Relativamente simples e compatível com diversos *frameworks* (Handlebars Community 2024).
- **Suporte:** Documentação sólida e comunidade ativa (Handlebars Community 2024).
- **Usabilidade:** Excelente para aplicações *web* modernas, permitindo a criação de interfaces dinâmicas e interativas de forma organizada (Credited Tecnologia 2024).

### 2.6.6 Comparação entre Tecnologias

Para a seleção da tecnologia mais adequada ao projeto, foram considerados os critérios previamente definidos na Subsecção 1.4.2 do Capítulo 1, que incluem aprendizagem, custo, desempenho, escalabilidade, flexibilidade, implementação, segurança, suporte e usabilidade.

A Tabela 2.7 apresenta a avaliação das tecnologias candidatas, com pontuações atribuídas com base nos critérios estabelecidos.

Tabela 2.7: Comparação de Tecnologias de *Template Engines*

| Critério | Velocity | FreeMarker | Thymeleaf | Mustache | Handlebars |
|---|---|---|---|---|---|
| **Aprendizagem** | 5<br>Muito Fácil | 4<br>Fácil | 4<br>Fácil | 5<br>Muito Fácil | 4<br>Fácil |
| **Custo** | 5<br>Muito Baixo | 5<br>Muito Baixo | 4<br>Baixo | 5<br>Muito Baixo | 5<br>Muito Baixo |
| **Desempenho** | 3<br>Médio | 5<br>Elevado | 3<br>Médio | 5<br>Elevado | 5<br>Elevado |
| **Escalabilidade** | 3<br>Limitada | 5<br>Elevada | 3<br>Limitada | 3<br>Limitada | 3<br>Limitada |
| **Flexibilidade** | 3<br>Limitada | 5<br>Muito Alta | 3<br>Limitada | 3<br>Limitada | 4<br>Boa |
| **Implementação** | 5<br>Muito Rápida | 4<br>Rápida | 4<br>Rápida | 5<br>Muito Rápida | 4<br>Rápida |
| **Segurança** | 3<br>Limitada | 5<br>Elevada | 5<br>Elevada | 3<br>Limitada | 3<br>Limitada |
| **Suporte** | 3<br>Básico | 5<br>Excelente | 3<br>Básico | 3<br>Básico | 4<br>Bom |
| **Usabilidade** | 5<br>Muito Fácil | 4<br>Fácil | 4<br>Fácil | 5<br>Muito Fácil | 4<br>Fácil |
| **Total** | **35** | **42** | **33** | **36** | **37** |

*Nota: A escala varia de 1 a 5, onde 1 representa o cenário menos favorável (ex.: aprendizagem difícil, custo elevado) e 5 representa o cenário mais favorável (ex.: aprendizagem muito fácil, custo muito baixo).*

Com base na Tabela 2.7, foi possível avaliar as cinco tecnologias candidatas de acordo com os critérios previamente definidos. A solução com maior pontuação total foi o Apache FreeMarker, que alcançou 42 pontos. No entanto, é importante apresentar uma análise crítica da escolha, considerando os pontos fortes e fracos das tecnologias avaliadas.

O Apache FreeMarker destacou-se em vários critérios importantes para este projeto, como desempenho, escalabilidade, flexibilidade e suporte. A sua capacidade de manipular formatos complexos, como JSON e XML, além da integração eficiente com *frameworks* Java como Spring Boot, contribuiu para a sua vantagem competitiva. Estes fatores tornam-no particularmente adequado para sistemas que exigem geração dinâmica de documentos em diferentes formatos, o que é uma necessidade crítica para este projeto.

Por outro lado, é importante reconhecer as limitações do FreeMarker em critérios como aprendizagem e implementação. Embora tenha uma curva de aprendizagem mais acentuada do que tecnologias como Mustache e Thymeleaf, a sua extensa documentação e comunidade ativa ajudam a mitigar esta dificuldade inicial. Adicionalmente, a implementação, embora não tão rápida quanto a de soluções mais simples como Mustache ou Velocity, ainda é eficiente e adaptável a contextos mais complexos.

Outras tecnologias também apresentaram características atrativas. O Mustache, a título de exemplo, possui uma curva de aprendizagem extremamente fácil e um custo muito baixo, destacando-se em simplicidade e rapidez de implementação. Contudo, a sua abordagem *logic-less* limita a flexibilidade, o que representa um desafio em projetos que exigem manipulação avançada de dados ou maior escalabilidade.

O Thymeleaf é uma solução altamente intuitiva para aplicações web baseadas em Java e oferece um excelente suporte no que respeita à segurança. Todavia, a sua escalabilidade e flexibilidade são limitadas quando comparadas ao Apache FreeMarker, especialmente em contextos que vão além da renderização de páginas HTML.

Já o Apache Velocity, embora muito rápido em termos de implementação e com custo reduzido, apresenta limitações em critérios como desempenho, escalabilidade e flexibilidade, o que o torna menos adequado para aplicações de maior complexidade ou que envolvem manipulação de formatos dinâmicos.

Por fim, o Handlebars mostrou-se uma alternativa interessante devido ao bom desempenho e à flexibilidade proporcionada pelos *helpers* personalizados. No entanto, em comparação ao Apache FreeMarker, apresenta menor escalabilidade e uma documentação menos detalhada, o que pode impactar negativamente o suporte e a curva de aprendizagem em projetos mais exigentes.

A escolha do Apache FreeMarker foi fundamentada na sua capacidade de atender aos requisitos técnicos deste projeto, que incluem a manipulação de dados dinâmicos e a geração de diferentes formatos de saída. Ainda que algumas tecnologias, como o Mustache ou o Thymeleaf, sejam mais simples em termos de aprendizagem e implementação, estas não conseguem oferecer o mesmo nível de flexibilidade e escalabilidade necessário para este contexto.

Assim, o Apache FreeMarker representa a melhor solução pela sua robustez e capacidade de adaptação às necessidades do projeto, mesmo que exija maior esforço inicial em termos de aprendizagem e implementação. A sua ampla adoção no mercado e o suporte da comunidade também garantem uma base confiável para o desenvolvimento e manutenção do sistema.

## 2.7 Desafios

A implementação da solução proposta, que consiste em utilizar Java para manipulação de *templates* em JSON e geração dinâmica de saídas em XML ou JSON, enfrenta diversos desafios técnicos e organizacionais. Estes desafios podem ser agrupados em categorias como manipulação de dados, integração de tecnologias, desempenho e manutenibilidade, conforme descrito de seguida.

### 2.7.1 Manipulação de Dados

A manipulação de dados representa um dos principais desafios da solução, devido à complexidade inerente ao uso de *templates* dinâmicos e dados heterogéneos. Entre os principais pontos de atenção estão:

- **Definição e análise:** Garantir que os *templates* em JSON criados pelo Angular sejam definidos de forma consistente e possam ser interpretados corretamente pela aplicação Java.
- **Obtenção dos dados:** A realização de *queries* eficientes à base de dados Oracle para extrair os dados definidos nos *templates*, minimizando o impacto no desempenho.
- **Transformação de dados:** A conversão dos dados da base de dados para o formato especificado definido no *template*, garantindo conformidade com o resultado final em XML ou JSON, conforme o que estiver configurado.

### 2.7.2 Integração de Tecnologias

A solução propõe a utilização de várias tecnologias que necessitam de integração harmoniosa para o funcionamento correto. Os desafios incluem:

- **Compatibilidade com Oracle Database:** Garantir que as consultas SQL e operações de manipulação de dados sejam otimizadas e compatíveis com a estrutura e características da base de dados Oracle.
- **Interação entre Angular e Java:** Estabelecer um fluxo confiável entre o Angular, que cria os *templates* e o *backend* Java, que os utiliza para gerar as saídas de dados dinâmicas.
- **Geração de saídas de dados dinâmicas:** Implementar uma lógica robusta que permita a escolha entre XML ou JSON como formato de saída, dependendo das configurações estabelecidas numa tabela da base de dados.

### 2.7.3 Desempenho

Outro desafio relevante é garantir que a solução seja capaz de lidar com altos volumes de dados sem comprometer o desempenho. Os pontos críticos incluem:

- **Processamento do template em tempo de execução:** A utilização de *template engines* em tempo de execução pode introduzir atrasos significativos, especialmente com *templates* complexos ou grandes volumes de dados.
- **Otimização de consultas:** As *queries* à base de dados devem ser otimizadas para evitar perda do desempenho em operações de leitura e escrita.

### 2.7.4 Manutenibilidade

A necessidade de manter e expandir a solução ao longo do tempo também apresenta desafios. Entre os principais, destacam-se:

- **Gestão de alterações nos templates:** A flexibilidade dos *templates* JSON pode dificultar a manutenção da solução, especialmente quando surgem mudanças frequentes nos requisitos.
- **Código modular e reutilizável:** Garantir que a aplicação Java seja projetada com modularidade, facilitando a adição de novas funcionalidades sem a necessidade de refatorizações extensas.
- **Documentação e padrões:** Manter a documentação atualizada e aplicar boas práticas de programação para facilitar o suporte ou a necessidade de efetuar novas alterações.

### 2.7.5 Qualidade e Segurança

A qualidade da solução também depende de como os desafios de segurança e validação de dados são enfrentados:

- **Validação dos dados:** Garantir que os dados extraídos da base de dados estejam corretos e que os *templates* JSON estejam bem formatados antes de serem utilizados.

- **Segurança dos dados:** Proteger as informações manipuladas pela solução contra acessos não autorizados ou exposições acidentais, especialmente ao trabalhar com dados sensíveis.
- **Testes automatizados:** Implementar uma bateria de testes automáticos que cubra diferentes cenários, garantindo o correto funcionamento da solução sob condições diversas.

Em suma, os desafios identificados abrangem desde a manipulação e transformação eficiente de dados, a integração harmoniosa de tecnologias, a garantia de desempenho, manutenibilidade e segurança, exigindo um cumprimento rígido do planeamento e uma implementação robusta para assegurar o sucesso da solução proposta.

## 2.8 Avaliação e Escolha das Tecnologias

A revisão do estado da arte, conduzida com base no protocolo PRISMA, permitiu não apenas identificar metodologias e práticas relevantes, mas sobretudo fundamentar as escolhas tecnológicas que suportam a solução desenvolvida.

No domínio da geração dinâmica de documentos, a análise comparativa entre diferentes *template engines* demonstrou que o Apache FreeMarker é a tecnologia mais adequada ao projeto. A sua elevada flexibilidade, o bom desempenho em contextos complexos e a integração nativa com o ecossistema Java foram fatores determinantes para a sua seleção, em detrimento de alternativas como Mustache, Thymeleaf ou Velocity.

Para o desenvolvimento do *backend*, foi escolhida a linguagem Java em conjunto com a *framework* Spring, garantindo robustez, escalabilidade e um alinhamento direto com a infraestrutura tecnológica já existente na organização. Esta escolha permite ainda simplificar a integração com outros módulos do sistema, assegurando consistência e manutenibilidade.

Na vertente de interface com o utilizador, a opção recaiu sobre o Angular, uma vez que a sua arquitetura modular e orientada a componentes facilita a criação de um *frontend* dinâmico e interativo. Esta escolha responde à necessidade de disponibilizar ferramentas de configuração de *templates* de forma intuitiva e adaptável às equipas funcionais.

Por fim, para a camada de dados, a utilização da base de dados Oracle mantém-se como opção estratégica, aproveitando a infraestrutura já consolidada e tirando partido de técnicas de otimização como índices, *materialized views* e mecanismos de *caching*, essenciais para lidar com grandes volumes de informação.

Em síntese, as escolhas tecnológicas recaíram sobre **Apache FreeMarker, Java/Spring, Angular e Oracle**, formando um ecossistema coerente e robusto que responde de forma eficaz aos requisitos identificados e às necessidades do setor segurador.

# 3. Análise e Design

Este capítulo apresenta a fase de análise e *design* da solução de emissão documental.

Na componente de análise, são identificados os atores envolvidos no sistema, definidos os requisitos funcionais e não funcionais e descritas as principais restrições e dependências técnicas. Adicionalmente, é realizado o mapeamento entre os objetivos estratégicos definidos e os requisitos identificados, garantindo alinhamento entre a visão do sistema e a sua concretização.

Na vertente de *design*, são detalhadas as decisões arquiteturais adotadas, a estrutura modular da solução, os componentes principais e os diagramas que ilustram o comportamento e a organização do sistema. Esta fase é importante para assegurar que a solução proposta responde de forma eficaz às necessidades da organização, promovendo robustez, desempenho e manutenibilidade.

## 3.1 Engenharia de Requisitos

A definição dos requisitos constitui uma etapa fundamental no processo de desenvolvimento de *software*, permitindo estabelecer claramente as funcionalidades esperadas, os intervenientes no sistema e os objetivos a alcançar. A solução proposta tem como objetivo principal otimizar a geração de documentos no contexto dos sistemas Core, eliminando a forte dependência entre aplicações e reduzindo os tempos de resposta associados à emissão documental. Para tal, pretende-se introduzir uma abordagem centrada num módulo de emissão que recolhe, processa e emite documentos com base em *templates* pré-definidos e dados estruturados. Nesta secção são apresentados os elementos que orientaram a construção da solução, com base nos problemas identificados na análise do sistema atual (Capítulo 2).

### 3.1.1 Atores do Sistema

A identificação dos atores do sistema é um passo inicial na fase de análise, pois permite compreender quem interage com a solução e quais são os seus interesses, responsabilidades e necessidades. Esta caracterização é essencial para garantir que os requisitos definidos refletem corretamente a realidade operacional e os fluxos aplicacionais esperados. A Tabela 3.1 apresenta os principais atores identificados, bem como uma breve descrição do seu papel no contexto da emissão documental.

Tabela 3.1: Atores do Sistema

| Ator | Descrição |
|---|---|
| *Backoffice* | Utilizador que, através do módulo *life-ear*, interage com o sistema para executar operações manuais que envolvem a emissão de documentos (ex: reimpressão de apólices, geração de cartas). |
| Processo *Batch* | Componente automatizado que, através do *batch-life-ear*, executa tarefas em lote (ex: envio de cartas em massa, emissão de recibos), acionando automaticamente o mecanismo de geração documental. |
| Motor Documental (Sistema) | Componente central responsável por gerar documentos, presente tanto na solução anterior como na atual. Este sistema interpreta os pedidos de emissão, recolhe os dados necessários, aplica os *templates* e produz o documento final. Na nova solução, este componente foi otimizado para reduzir dependências remotas e melhorar o desempenho. |
| Configurador de *templates* | Responsável pela criação e manutenção dos *templates* documentais utilizados na emissão. Este assegura que os modelos de documento refletem corretamente as regras de negócio e a estrutura dos produtos da seguradora. |

### 3.1.2 Requisitos Funcionais

Com base na análise das necessidades dos atores identificados, foram definidos os requisitos funcionais da solução. Estes requisitos descrevem as funcionalidades que o sistema deverá suportar, respondendo diretamente às operações que os utilizadores esperam realizar. A Tabela 3.2 apresenta os requisitos funcionais enumerados e descritos de forma clara e objetiva, de modo a orientar as fases subsequentes de *design* e implementação.

Tabela 3.2: Requisitos Funcionais

| Código | Descrição |
|---|---|
| RF-01 | Permitir a geração de documentos em diferentes formatos, nomeadamente *XML* e *JSON*. |
| RF-02 | Permitir a emissão de documentos a partir de diferentes origens de dados, tanto em contexto interativo como *batch*. |
| RF-03 | Validar a estrutura e conteúdo dos dados antes da geração do documento, garantindo a conformidade com o modelo esperado. |
| RF-04 | Permitir a definição de *templates* reutilizáveis e parametrizáveis para os diferentes tipos de documentos. |
| RF-05 | Disponibilizar uma interface de comunicação com os módulos Core (*life-ear* e *batch-life-ear*) para receção dos pedidos de emissão. |

### 3.1.3 Requisitos Não Funcionais

Para além das funcionalidades esperadas, é igualmente necessário definir os requisitos não funcionais, que dizem respeito a aspetos de qualidade, desempenho, segurança, manutenção e usabilidade. Estes requisitos estabelecem critérios técnicos e operacionais que a solução

deve cumprir para garantir o seu sucesso em ambiente real. A Tabela 3.3 sintetiza os principais requisitos não funcionais identificados nesta análise.

Tabela 3.3: Requisitos Não Funcionais

| Código | Descrição |
|---|---|
| RNF-01 | A solução deve priorizar o desempenho, minimizando o tempo de geração dos documentos mesmo sob elevada carga. |
| RNF-02 | A comunicação entre os diferentes componentes da aplicação deve ser realizada de forma interna e eficiente, evitando o uso de interfaces web (como REST), de modo a melhorar o desempenho global da solução. |
| RNF-03 | A solução deve ser suficientemente modular para permitir a substituição de componentes específicos (ex: motor de *templates*) sem reescrever todo o sistema. |
| RNF-04 | O sistema deve assegurar a integridade e consistência dos dados durante todo o processo de emissão documental. |
| RNF-05 | A aplicação deve estar preparada para funcionar em ambiente de execução controlado (ex: servidores JBoss/Wildfly), mas com autonomia suficiente para se manter desacoplada do ciclo de vida de outras aplicações. |

### 3.1.4 Mapeamento entre Objetivos e Requisitos

A definição dos objetivos do projeto (Capítulo 1) permitiu orientar o desenvolvimento de uma solução alinhada com as necessidades identificadas. Por sua vez, os requisitos funcionais e não funcionais apresentados nesta secção foram elaborados de modo a garantir que esses objetivos possam ser concretizados de forma sistemática e mensurável.

A tabela 3.4 apresenta o mapeamento entre os objetivos do projeto e os requisitos definidos, distinguindo entre requisitos funcionais (RF) e não funcionais (RNF). Esta correspondência evidencia como cada requisito contribui diretamente para a concretização de um ou mais objetivos estratégicos da solução.

Tabela 3.4: Mapeamento entre Objetivos e os Requisitos do Sistema

| Código | Descrição do Objetivo | Requisitos Funcionais | Requisitos Não Funcionais |
|---|---|---|---|
| OBJ-01 | Minimizar o custo de desenvolvimento de novas funcionalidades e o esforço necessário para suporte entre versões. | RF-04 | RNF-03 |
| OBJ-02 | Minimizar o tempo necessário para a geração e envio de documentos, garantindo eficiência e escalabilidade. | RF-01, RF-02 | RNF-01, RNF-02 |
| OBJ-03 | Maximizar a compatibilidade do sistema, assegurando o suporte a múltiplos formatos e interoperabilidade entre versões. | RF-01, RF-05 | RNF-03, RNF-05 |

### 3.1.5 Restrições e Dependências

A identificação de restrições e dependências é essencial para garantir que os requisitos definidos são realistas e viáveis no contexto técnico e organizacional do projeto. Estas restrições impõem limites à liberdade de conceção e implementação, enquanto as dependências representam ligações a elementos externos ao sistema que influenciam diretamente o seu funcionamento. Ignorar estas condicionantes pode comprometer a implementação ou dificultar a manutenção futura da solução.

A Tabela 3.5 apresenta as principais restrições e dependências identificadas durante a fase de levantamento de requisitos.

Tabela 3.5: Restrições e Dependências da Solução

| Código | Descrição |
|---|---|
| RD-01 | A solução deve ser compatível com a infraestrutura já existente, nomeadamente com o servidor de aplicacional Wildfly. |
| RD-02 | A base de dados utilizada continuará a ser Oracle, não sendo permitida a alteração do modelo de dados base. |
| RD-03 | Os módulos *life-ear* e *batch-life-ear* mantêm-se como emissores de pedidos, pelo que os contratos de integração atuais não podem ser alterados. |
| RD-04 | A geração de documentos deve respeitar os fluxos operacionais existentes, sem provocar ruturas nos processos de negócio em produção. |

## 3.2 Arquitetura

A arquitetura da solução proposta foi definida com base na necessidade de melhorar o desempenho, reduzir o acoplamento entre componentes e eliminar a dependência de chamadas remotas REST no processo de emissão documental. Esta secção apresenta, em primeiro lugar, a arquitetura que foi efetivamente implementada, seguida da arquitetura alternativa que foi analisada durante o processo de conceção, incluindo a justificação para a sua não adoção.

### 3.2.1 Arquitetura Implementada

A solução implementada consistiu na criação de um novo componente, *life-output-bll*, integrado diretamente na camada comum de código das aplicações *life-ear* e *batch-life-ear*. Este componente encapsula a responsabilidade de geração documental, incluindo:

- Implementação da *template engine* FreeMarker, identificada como a mais adequada no Capítulo 2 - Estado da Arte.
- Acesso direto, via Spring, à lógica de negócio já existente - *business-bll*.
- Eliminação das chamadas REST previamente existentes entre componentes distribuídos.

Com esta abordagem, o processo de emissão documental passa a ocorrer localmente dentro do mesmo servidor da aplicação, recorrendo a invocação direta de métodos em vez de comunicação remota. Esta mudança permitiu ganhos significativos de desempenho, eliminando a latência de rede, simplificando o fluxo de chamadas e reduzindo o acoplamento entre artefactos.

A Figura 3.1 ilustra a nova arquitetura da solução implementada.

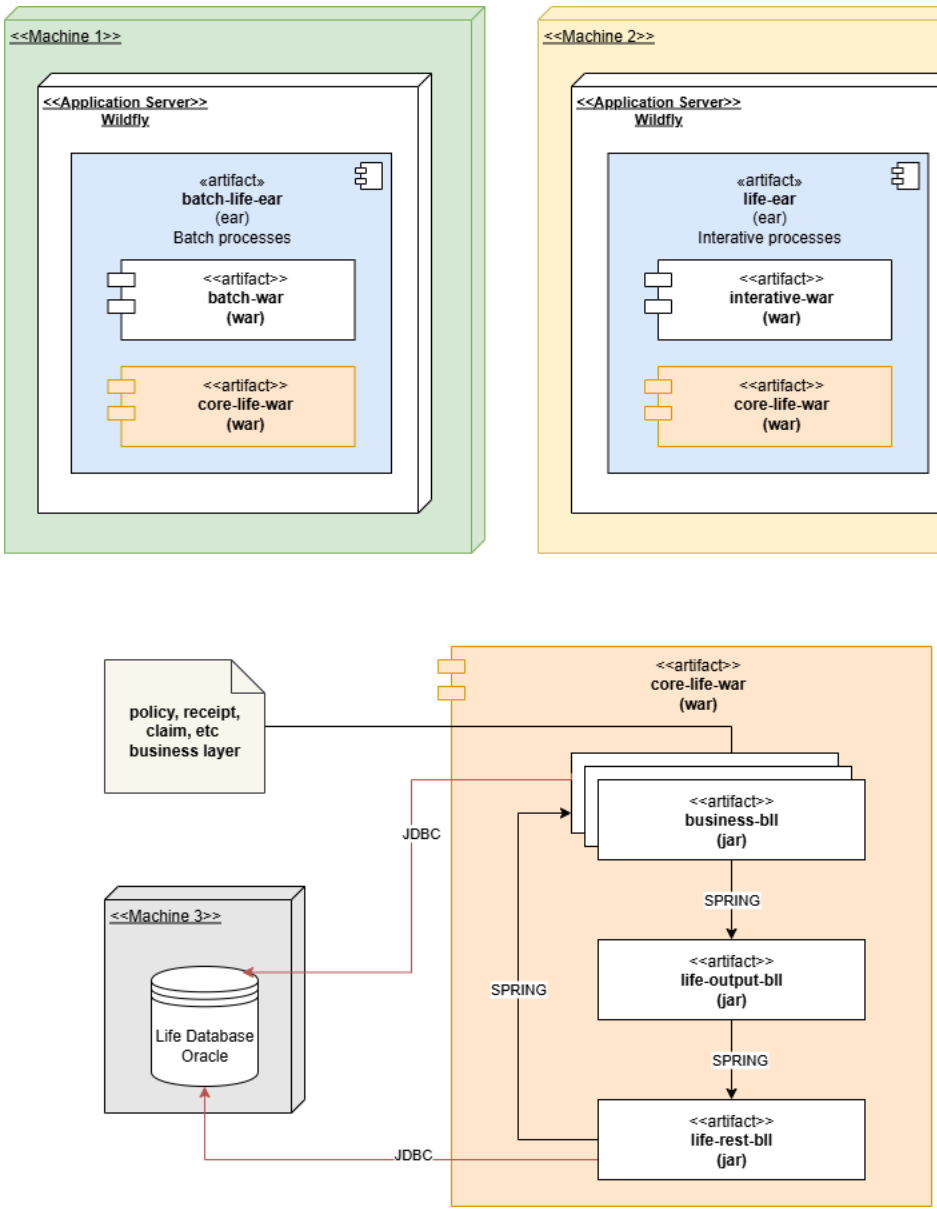



Figura 3.1: Arquitetura Implementada

### 3.2.2 Arquitetura Alternativa Analisada

Durante a fase de análise, foi considerada uma abordagem alternativa com o objetivo de maximizar a escalabilidade e isolar completamente o motor documental (sistema) do Core. Nesta solução, a componente de emissão seria isolada como um módulo externo - *life-output-ear* - e a recolha de dados seria feita diretamente através da base de dados Oracle, eliminando assim qualquer tipo de chamada REST ou SOAP entre aplicações.

A principal motivação desta abordagem era garantir que o motor documental (sistema) pudesse ser reutilizado por diferentes aplicações da organização sem depender de integrações diretas com código legado.

No entanto, esta abordagem apresentava um conjunto de limitações críticas que levaram à sua exclusão:

- **Alto risco e esforço de manutenção:** transformar regras de negócio complexas, antes executadas por serviços Java, em *queries* SQL diretas, implica risco elevado e custo de manutenção acrescido.
- **Desvio da responsabilidade do domínio:** a lógica de construção dos dados passaria para a camada de dados (SQL), o que viola o princípio de separação de responsabilidades.
- **Latência de rede persistente:** apesar da eliminação de REST entre os módulos principais, o motor documental (sistema) continuaria a ser invocado remotamente, mantendo a latência de rede.

A Figura 3.2 representa esta arquitetura alternativa considerada, mas não adotada.

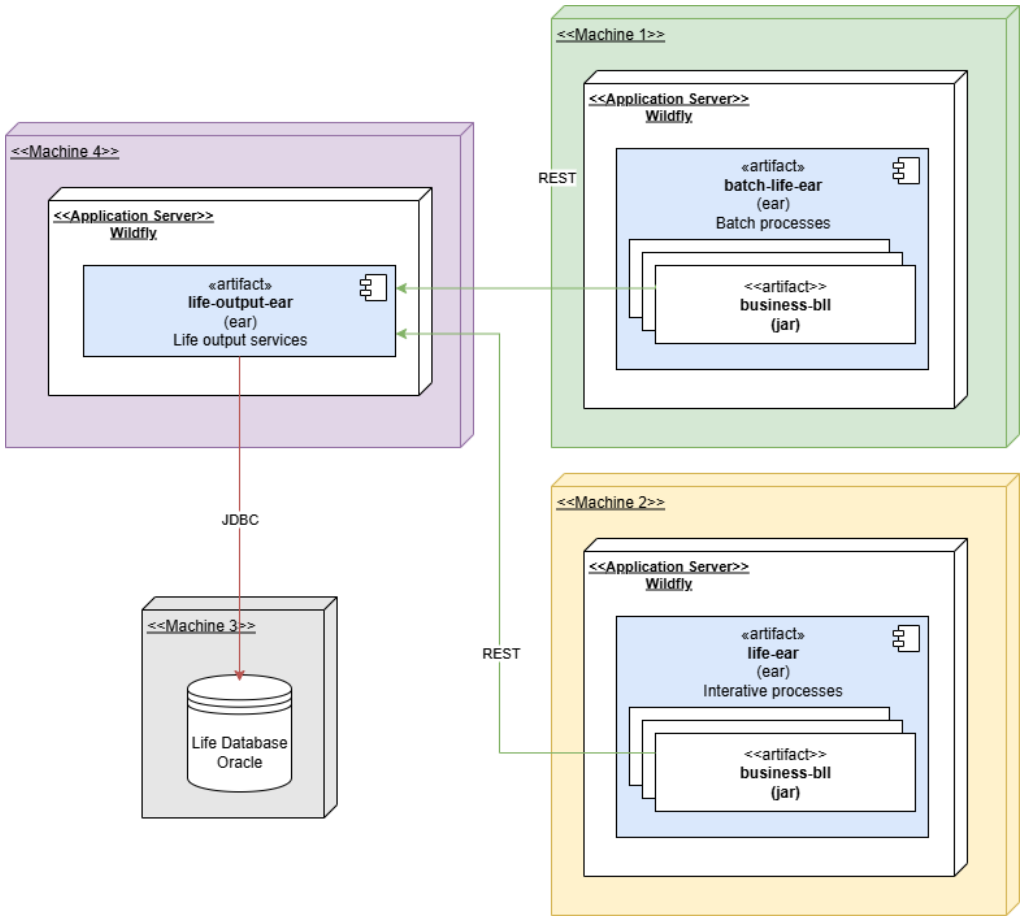



Figura 3.2: Arquitetura Alternativa

### 3.2.3 Justificação da Decisão Arquitetural

A escolha da arquitetura implementada foi fundamentada na necessidade de garantir elevada performance, reduzir a complexidade da solução e facilitar a integração com os módulos Core existentes. Ao optar por um componente integrado e reutilizável via Spring, foi possível

manter os fluxos funcionais existentes e melhorar significativamente o tempo de resposta na emissão de documentos, sem comprometer a robustez ou aumentar a complexidade técnica.

Adicionalmente, a solução implementada permitiu preservar os princípios de modularidade, reutilização de código e alinhamento com a infraestrutura atual da organização (Wildfly + Oracle), garantindo um equilíbrio eficaz entre evolução tecnológica e viabilidade operacional.

### 3.2.4 Comparação entre Arquiteturas

A escolha da arquitetura final foi precedida por uma análise comparativa entre duas abordagens viáveis: uma solução alternativa baseada em consumo direto de dados via base de dados Oracle e a solução implementada, que reutiliza os serviços existentes através de invocação interna por Spring.

Para estruturar esta decisão, foram considerados critérios técnicos e funcionais com impacto direto na performance, manutenibilidade e risco da solução. A Tabela 3.6 sintetiza essa comparação, tendo em conta aspetos como latência, complexidade de implementação e alinhamento com a arquitetura atual da organização.

Tabela 3.6: Comparação entre Arquitetura Implementada e Alternativa

| **Critério** | **Arquitetura Implementada** | **Arquitetura Alternativa** |
|---|---|---|
| Latência de rede | Eliminada com invocação local (Spring) | Persistente entre módulos (REST) |
| Desempenho geral | Elevado | Mediano |
| Esforço de implementação | Moderado (integração Spring) | Elevado (reescrita em SQL) |
| Risco técnico | Baixo | Elevado |
| Reutilização futura | Facilitada via bibliotecas comuns | Facilitada como módulo isolado |
| Aderência à infraestrutura atual | Total | Parcial (com complexidade acrescida) |
| Separação de responsabilidades | Preservada | Comprometida (negócio na base de dados) |

Como evidenciado na Tabela 3.6, a solução implementada apresenta vantagens significativas em termos de desempenho (**RNF-01**), simplicidade de comunicação entre componentes (**RNF-02**) e redução de risco técnico. Ao eliminar a latência associada a chamadas REST e evitar a complexidade de reescrever lógica de negócio em SQL, a arquitetura final assegura um equilíbrio eficaz entre performance e sustentabilidade.

Adicionalmente, esta decisão responde diretamente ao requisito de modularidade (**RNF-03**), uma vez que permite evoluir ou substituir o motor de *templates* sem impacto nos restantes serviços. Também contribui para o cumprimento do requisito funcional de disponibilizar uma interface de comunicação com os módulos Core (**RF-05**), assegurando compatibilidade retroativa com o ecossistema existente.

A arquitetura alternativa, embora teoricamente mais escalável, implicaria uma transformação profunda do modelo atual e um esforço elevado de reimplementação, colocando em causa a consistência funcional dos processos (**RNF-04**) e elevando o custo de manutenção. Assim, a

opção pela invocação direta via Spring representou a abordagem mais pragmática e alinhada com os requisitos e objetivos do projeto.

De forma complementar, esta decisão contribuiu também para o cumprimento dos objetivos definidos na Secção 1.4: reduzir o custo e esforço de manutenção (**OBJ-01**), melhorar o desempenho global e reduzir o tempo de processamento (**OBJ-02**) assim como assegurou a compatibilidade futura com diferentes formatos e evoluções tecnológicas (**OBJ-03**). Deste modo, a arquitetura selecionada não só responde às limitações identificadas como também estabelece uma base sólida para evolução do sistema.

### 3.2.5 Padrões Arquiteturais Aplicados

A arquitetura da solução adotada incorpora diversos padrões e boas práticas da engenharia de software que contribuem para a qualidade técnica, manutenibilidade e robustez do sistema. Estes padrões são especialmente relevantes num contexto empresarial com múltiplos módulos, equipas e exigências de desempenho e fiabilidade.

- **Modularidade:** A separação entre componentes permite que cada artefacto seja desenvolvido, testado e mantido de forma autónoma. Isto facilita a evolução do sistema ao longo do tempo, permitindo atualizações incrementais sem comprometer o funcionamento global.
- **Dependency Injection (DI):** Com o recurso à *framework* Spring, o sistema adota o padrão de Injeção de Dependências, delegando ao *container* a responsabilidade de instanciar e gerir os objetos. Esta abordagem reduz o acoplamento entre classes, promove a reutilização de código e facilita a criação de testes automatizados.
- **Single Responsibility Principle (SRP):** Cada componente tem uma responsabilidade bem definida, sendo que os módulos Core enviam pedidos, o motor documental (sistema) trata da composição e o *template engine* da apresentação. Este princípio está alinhado com o modelo de arquitetura em camadas (camada de apresentação, de negócio e de dados).
- **Desacoplamento Funcional:** A substituição de chamadas REST por chamadas locais elimina a dependência tecnológica entre módulos distribuídos. Isto não só reduz a latência como diminui o risco de falhas na comunicação.
- **Encapsulamento de Serviços:** O *life-output-bll* serve como fachada interna para todos os serviços de geração documental. Este padrão permite proteger a complexidade interna do sistema, expondo apenas as interfaces necessárias para os restantes módulos.

A aplicação destes padrões resultou numa arquitetura robusta, com baixo acoplamento, elevada coesão e alinhamento com os princípios e boas práticas de programação, essenciais em sistemas empresariais complexos.

### 3.2.6 Extensibilidade e Manutenção Futura

Um dos objetivos fundamentais da arquitetura foi garantir que a solução pudesse evoluir sem ruturas nem dependências rígidas. A modularidade e o encapsulamento dos componentes contribuem decisivamente para a sua extensibilidade futura, permitindo acomodar novas necessidades da organização.

**Do ponto de vista funcional**, a arquitetura permite:

- Adicionar novos tipos de documentos ou processos de negócio, reutilizando a mesma infraestrutura de emissão;
- Criar novas lógicas de composição de dados com base em regras de negócio ajustadas, sem necessidade de reescrever o sistema central;
- Substituir ou evoluir o *template engine* (por exemplo, passar de FreeMarker para Thymeleaf ou outra) de forma isolada.

**Do ponto de vista técnico e operacional**, a manutenção é facilitada por:

- Utilização de chamadas Spring diretas, facilmente testáveis e rastreáveis em tempo de execução;
- Separação de artefactos em bibliotecas reutilizáveis, como o *life-output-bll*, que pode ser evoluído independentemente;
- Integração controlada com a infraestrutura existente (Oracle, Wildfly), o que evita dependências externas que possam comprometer a operação em produção.

Esta abordagem assegura que a solução não só responde às necessidades atuais, como se encontra preparada para adaptações futuras com custo reduzido e risco controlado.

### 3.2.7 Resumo Crítico da Solução Arquitetural

A decisão arquitetural final assentou num conjunto de fatores que foram cuidadosamente ponderados, tendo em conta os objetivos funcionais e não funcionais do projeto, bem como os constrangimentos técnicos e organizacionais existentes.

A arquitetura implementada representa uma evolução pragmática da arquitetura anterior, com ganhos objetivos ao nível da performance e manutenibilidade, sem comprometer a estabilidade da solução nem a integração com os módulos Core. A adoção de um componente comum, reutilizável e leve - *life-output-bll* - mostrou-se eficaz em:

- Reduzir o tempo de resposta no processo de emissão documental;
- Eliminar pontos de falha associados a chamadas remotas;
- Reutilizar a lógica de negócio e *templates* existentes, minimizando o esforço de desenvolvimento;
- Manter a compatibilidade com o ambiente de execução atual da organização.

Por outro lado, a arquitetura alternativa, embora atrativa do ponto de vista da escalabilidade futura, apresentava riscos técnicos e operacionais demasiado elevados, nomeadamente ao nível da reimplementação da lógica de negócio em SQL.

Assim, a solução adotada reflete um compromisso equilibrado entre inovação, desempenho e viabilidade, posicionando-se como uma base sólida para futuras evoluções do sistema de emissão documental da organização.

## 3.3 Design

A fase de *design* teve como objetivo desenhar a arquitetura proposta e os requisitos definidos numa solução concreta, modular e robusta, adequada ao contexto organizacional e técnico do sistema de emissão documental. Esta secção apresenta as principais decisões de *design*, os componentes envolvidos, documentação arquitetural de apoio à compreensão da estrutura da solução e as considerações associadas à testabilidade e validação.

### 3.3.1 Estratégia de Design

A abordagem adotada privilegiou a modularidade, reutilização de serviços existentes e comunicação eficiente entre componentes. O novo componente *life-output-bll*, incorporado na camada comum das aplicações *life-ear* e *batch-life-ear*, foi desenhado como ponto central de orquestração da emissão documental.

Este componente tem como responsabilidades:

- Gerir a carga de *templates* e aplicação do motor de *templates* **FreeMarker**;
- Recolher os dados necessários via invocação direta de serviços Spring.
- Preparar o modelo de dados e invocar o processo de renderização.
- Produzir o documento final no formato pretendido (XML, JSON, etc.).
- Facilitar a integração com os módulos de negócio existentes de forma transparente para o utilizador final.

A decisão de centralizar esta lógica num único componente permitiu isolar responsabilidades e aplicar o princípio do SRP, promovendo uma arquitetura mais limpa e fácil de manter. Além disso, ao evitar dependências com APIs externas (REST ou SOAP), foi possível garantir maior controlo sobre o desempenho e a previsibilidade do sistema.

### 3.3.2 Diagrama de Sequência

A Figura 3.3 apresenta o diagrama de sequência correspondente ao processo de emissão de uma apólice, utilizado como exemplo representativo do funcionamento genérico da solução. Este caso ilustra a interação entre os principais componentes do sistema, desde o momento em que o pedido de emissão é iniciado até à geração efetiva do documento.

Apesar de existirem diferentes tipos de documentos como recibos, cartas ou participações de sinistro, todos seguem um padrão de processamento semelhante, variando apenas nas regras de negócio aplicadas e nos dados envolvidos. O caso específico da apólice, aqui representado, destaca-se pela necessidade de invocar processos de negócio adicionais para obter dados calculados em tempo real, como é o caso do saldo da apólice. Este valor resulta de uma lógica de cálculo composta, que deve ser executada no momento da geração do documento, garantindo a consistência da informação.

A elaboração deste diagrama permitiu validar a sequência de chamadas entre componentes, identificar possíveis pontos de otimização e garantir uma comunicação eficiente e sem redundâncias, contribuindo assim para o desempenho e robustez da solução.

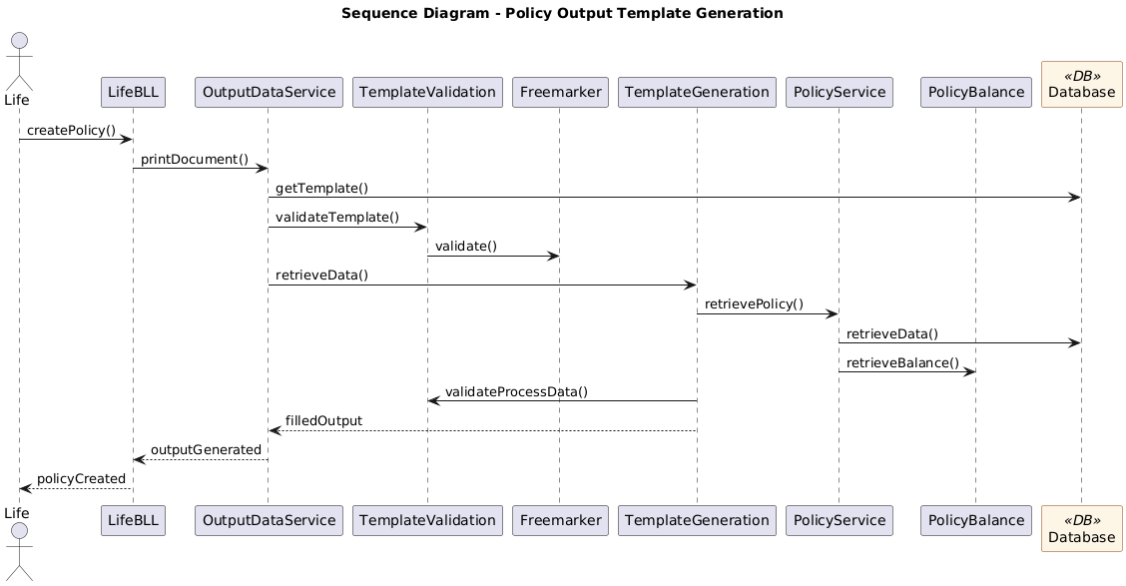



Figura 3.3: Diagrama de Sequência do Processo de Emissão Documental

### 3.3.3 Mecanismo de Template e Geração

O processo de emissão documental baseia-se na renderização de *templates* FreeMarker com dados dinâmicos fornecidos pelos serviços do Core. Para garantir a flexibilidade e a consistência da solução, o componente *life-output-bll* é responsável por preparar o modelo de dados, executar a renderização e devolver a estrutura final. As principais funcionalidades deste mecanismo incluem:

- *Templates* mantidos em formato JSON na base de dados e configuráveis pela equipa funcional.
- Validação automática com dados *dummy* antes da execução real, permitindo identificar erros de lógica e de dados em tempo de desenvolvimento.
- Suporte para múltiplos formatos (XML e JSON), dependendo do tipo de saída configurado para cada *template*.
- Capacidade de utilizar estruturas condicionais, ciclos e macros para lidar com documentos complexos e variáveis.

Desta forma, o mecanismo proporciona flexibilidade e mantém a lógica de apresentação desacoplada da lógica de negócio, alinhando-se com as boas práticas de separação de responsabilidades.

### 3.3.4 Testabilidade e Estratégia de Validação

O *design* da solução teve em conta a necessidade de facilitar a validação e os testes automatizados, assegurando a qualidade e estabilidade do sistema mesmo em cenários adversos. Para tal, foram implementados mecanismos como:

- Validação de *templates* com dados simulados (*dummy*) antes da ativação, evitando erros de execução em ambiente produtivo.

- Suporte a *mocks* e injeção de dependências via Spring (DI), facilitando a criação de testes independentes e controlados.
- Mecanismos de *log* e auditoria que facilitam a deteção e análise de erros de *template* ou dados.

Estas abordagens contribuem para a robustez e previsibilidade do sistema, permitindo detetar falhas precocemente e reduzir significativamente o tempo de correção.

### 3.3.5 Interface de Simulação de Documentos

Foi desenvolvido um *frontend* em Angular que permite criar, editar e simular a emissão de documentos em ambiente de teste. Esta interface tem como objetivo principal acelerar o ciclo de validação dos *templates* e dos dados, bem como aumentar a autonomia da equipa funcional. As principais funcionalidades disponibilizadas são:

- Permitir selecionar tipos de documento (apólice, recibo, indemnização, etc.);
- Carregar *templates* em formato JSON, permitindo simular casos específicos e validar a renderização.
- Apresentar os documentos gerados em tempo real, com destaque para erros de sintaxe ou inconsistências nos dados fornecidos.
- Possibilitar a exportação dos resultados gerados, facilitando a partilha e análise entre equipas.

Esta ferramenta revela-se essencial para as equipas funcionais e de desenvolvimento, ao permitir validações rápidas, exploratórias e iterativas durante o desenvolvimento de novos *templates*. A sua existência permite reduzir a dependência do ciclo de desenvolvimento do *backend* e antecipar erros críticos ainda em fase de configuração funcional.

# 4. Implementação da Solução

Este capítulo descreve a concretização técnica da solução de emissão documental, detalhando as principais decisões, adaptações arquiteturais e elementos implementados ao longo desta fase. A implementação abrangeu duas vertentes complementares:

- No **backend**, foi criado o novo módulo `life-output-bll`, responsável por orquestrar a geração de documentos com base em *templates* dinâmicos. Este módulo centraliza a lógica de negócio associada à obtenção de dados, validação de *templates* e renderização do conteúdo utilizando o motor FreeMarker. Foram ainda definidos os *beans* Spring necessários, bem como *wrappers* personalizados para controlo de formatação de tipos como datas ou valores numéricos.
- No **frontend**, foi desenvolvida uma aplicação Angular que permite criar, editar e validar *templates* de forma autónoma. Esta aplicação oferece funcionalidades como destaque de sintaxe, ajuda contextual com *placeholders* disponíveis, validação e simulação completa da execução de documentos a partir de identificadores reais.

Adicionalmente, foram realizadas alterações na estrutura de persistência, com a criação de uma tabela dedicada ao armazenamento de *templates* em base de dados Oracle, facilitando a gestão centralizada e o versionamento dos artefactos. A fase de implementação foi guiada pelos princípios definidos no *design* da solução e manteve o foco na modularidade, extensibilidade e testabilidade, culminando numa solução funcional, reutilizável e alinhada com os objetivos da organização.

## 4.1 Estrutura Base

A implementação da solução implicou uma reorganização da estrutura dos projetos existentes, nomeadamente a introdução do novo módulo `life-output-bll`, integrado na camada comum das aplicações `life-ear` e `batch-life-ear`. Este módulo centraliza a lógica de geração documental, passando a invocar diretamente os serviços internos existentes através de mecanismos Spring, eliminando a necessidade de comunicação remota via REST.

Foi ainda necessário adaptar a estrutura de empacotamento e dependências Maven para garantir a correta partilha de componentes entre aplicações, assim como definir pontos de extensão para a integração com o motor de *templates* FreeMarker. No que respeita à persistência dos *templates*, foi criada uma tabela específica para guardar-los com o seu identificador de *template*.

Esta reorganização da estrutura contribuiu para reduzir a duplicação de código, aumentar a manutenibilidade e facilitar a aplicação de testes unitários e de integração.

### 4.1.1 Serviço de Orquestração

O núcleo funcional da geração de documentos foi implementado na classe `OutputDataServiceImpl`, que concretiza a interface `OutputDataService`. Esta classe coordena todas as etapas do

processo: obtenção do *template*, validação, recolha dos dados e renderização final do documento. No Código 4.1 apresenta um excerto representativo da implementação deste serviço, evidenciando a orquestração dos diferentes componentes envolvidos.

```
1  @Override
2  public Pair<Output, Map<String, String>> generateOutput(String
     templateName, String format, OutputBusinessKey businessKey) throws
     Exception {
3
4      Outputcfg outputcfg = outputcfgRepo.findByIndex(templateName);
5
6      ContextType contextType = ContextType.fromString(outputcfg.
     getTemplateContext());
7
8      Pair<Template, Map<String, Object>> pair = outputTemplateValidation.
     validateTemplate(outputcfg.getTemplate(), false);
9
10     Map<String, Object> dados = outputTemplateGeneration.retrieveData(
     contextType, businessKey, pair.getSecond());
11
12     String filledTemplate = outputTemplateValidation.validateProcessData
     (pair.getFirst(), dados, format);
13
14     return Pair.of(new Output(templateName, filledTemplate),
     defineMetadata(dados));
15 }
```

Código 4.1: Excerto Classe Orquestradora

### 4.1.2 Configuração da Camada de Serviços

Toda a configuração de dependências e inicialização de serviços foi centralizada na classe `LifeOutputsConfiguration`, permitindo registar manualmente os componentes necessários à execução da lógica documental. Esta configuração, garante um controlo total sobre a criação de *beans*, facilitando a testabilidade e a integração entre os vários serviços. A título de exemplo, no Código 4.2 podemos observar a configuração do FreeMarker.

```
1  @Bean
2  public freemarker.template.Configuration freemarkerConfig() {
3      Configuration cfg = new Configuration(Configuration.VERSION_2_3_33);
4      cfg.setDefaultEncoding("UTF-8");
5      cfg.setObjectWrapper(new OutputTemplateWrapper(Configuration.
     VERSION_2_3_33));
6      cfg.setTemplateExceptionHandler(TemplateExceptionHandler.
     RETHROW_HANDLER);
7      cfg.setClassicCompatible(true); // ignora campos null
8      return cfg;
9  }
```

Código 4.2: Configuração FreeMarker

### 4.1.3 Conversão Personalizada de Tipos

A representação consistente dos dados no *template* — como datas, inteiros e valores monetários — foi assegurada através da implementação da classe `OutputTemplateWrapper`, uma extensão do `DefaultObjectWrapper` do motor FreeMarker. Esta classe personaliza o

processo de conversão de objetos Java para o modelo interno de *templates*, garantindo formatos padronizados e compatíveis com as necessidades da solução. O Código 4.3 apresenta a implementação correspondente.

```
@Override
public TemplateModel wrap(Object obj) throws TemplateModelException {
  if (obj instanceof BigDecimal bd) {
    return new SimpleScalar(bd.toPlainString());
  }
  if (obj instanceof LocalDate localDate) {
    return new SimpleScalar(localDate.format(DATE_FORMATTER));
  }
  if (obj instanceof Date legacyDate) {
    return new SimpleScalar(
    DATETIME_FORMATTER.format(legacyDate.toInstant().atZone(ZoneId.
  systemDefault()).toLocalDateTime()));
  }
  return super.wrap(obj);
}
```

Código 4.3: Excerto Classe Conversão Personalizada de Tipos

### 4.1.4 Persistência de Templates

O armazenamento dos *templates* é realizado numa tabela dedicada da base de dados Oracle, o que possibilita a centralização da configuração e o controlo de versões. A tabela `OUTPUTCFG` contém, para cada *template*, o nome identificador, o contexto de negócio associado e o conteúdo em formato `CLOB`. A definição SQL desta estrutura pode ser consultada abaixo 4.4.

```
CREATE TABLE "OUTPUTCFG" (
  "TEMPLATE_NAME" VARCHAR2(20 CHAR) NOT NULL,
  "TEMPLATE_CONTEXT" VARCHAR2(20 CHAR) NOT NULL,
  "TEMPLATE" CLOB NOT NULL
);
```

Código 4.4: *Script* Tabela de Persistência

### 4.1.5 Exemplo de Template Apólice

O Código 4.5 apresenta um excerto de um *template* real no formato `JSON` utilizado para a geração de uma apólice. Este exemplo evidencia o uso de instruções de iteração com a diretiva `<#list>` e o acesso direto às variáveis do modelo de dados, refletindo a flexibilidade e expressividade proporcionadas pelo motor FreeMarker.

```
{
  "policy": {
    "policyNumber": "${policy.policyNumber}",
    "coverages": [
      <#list policyCoverageList as policyCoverage>
      {
        "capital": "${policyCoverage.capital}",
        "description": "${policyCoverage.description}"
      }
      <#if policyCoverage_has_next>,</#if>
      </#list>
```

```
12        ],
13        "policyState": "${policy.policyState}"
14      }
15    }
```

Código 4.5: Excerto *Template* FreeMarker

## 4.2 Criação e Edição dos Templates

Com o objetivo de garantir a flexibilidade e autonomia das equipas funcionais, foi desenvolvido um módulo *frontend* em Angular que permite a criação e edição de *templates* em ambiente gráfico e controlado. Esta interface possibilita a edição dos *templates*, que são carregados da base de dados em formato JSON, destaques de sintaxe, verificação básica de estrutura e validação com dados de teste.

Esta funcionalidade torna-se um grande acelerador para construção de *templates* pois reduz a dependência da equipa técnica de desenvolvimento nas tarefas de configuração, permitindo um funcionamento mais ágil e colaborativo.

A Figura 4.1 apresenta a interface de listagem dos *templates*, permitindo à equipa funcional aceder rapidamente à configuração existente.

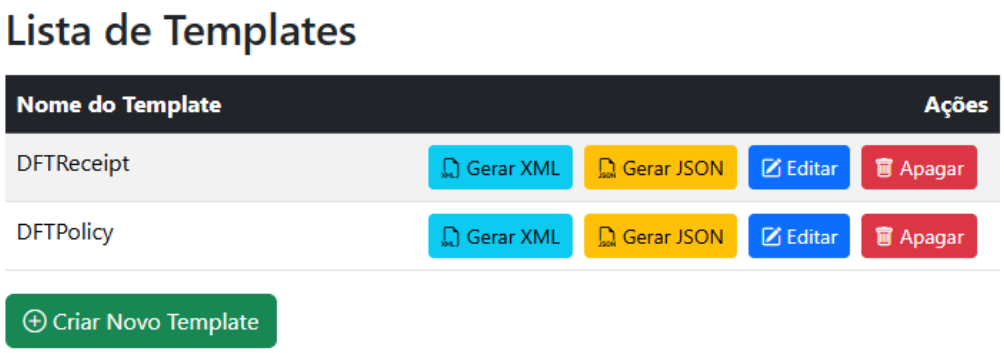



Figura 4.1: Listagem de *Templates*

A criação e edição de cada *template* é realizada através do ecrã ilustrado na Figura 4.2, o qual suporta destaques de sintaxe, carregamento automático do conteúdo a partir da base de dados no caso da edição e disponibiliza uma área lateral com a listagem contextual dos *placeholders* disponíveis. Esta funcionalidade atua como um guia para o utilizador, facilitando a configuração correta dos campos e reduzindo significativamente o risco de erros na estrutura do *template*.

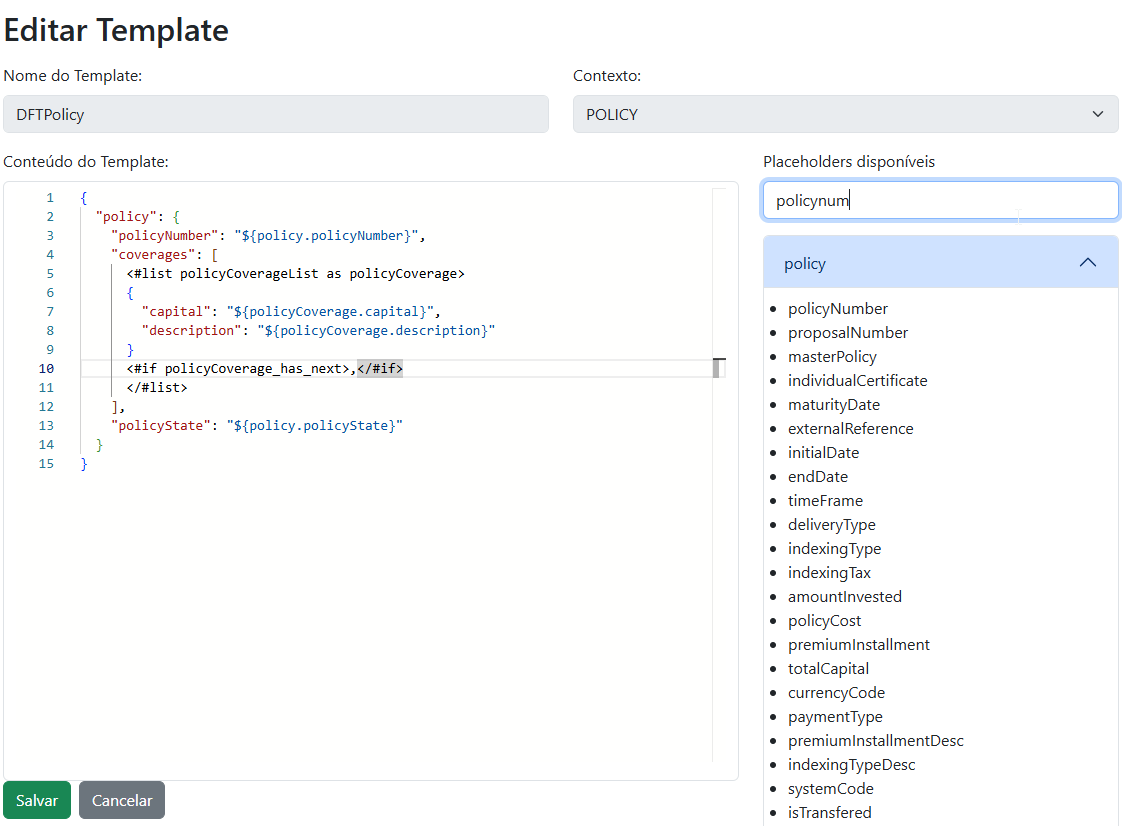



Figura 4.2: Criação e Edição de *Templates*

## 4.3 Visualização dos Resultados

Como complemento à funcionalidade de edição de *templates*, foi desenvolvido um ecrã de simulação que permite testar, em tempo real, a geração de documentos com base nos identificadores dos documentos de negócio (como apólices, recibos ou participações de sinistro).

Esta interface de simulação revela-se essencial para validar o comportamento dos *templates* antes da sua disponibilização em ambiente produção, contribuindo para a deteção precoce de erros e para a melhoria contínua da qualidade dos documentos gerados.

A área de simulação disponibiliza as seguintes funcionalidades:

- Seleção de um *template* previamente criado e registado na base de dados.
- Introdução de dados de teste reais, através das chaves de negócio.
- Geração do documento resultante no formato (XML ou JSON), com visualização em tempo real.

A Figura 4.3 ilustra esta interface, onde é possível introduzir dados de teste, visualizar a renderização final e detetar problemas de forma interativa e segura. Este mecanismo de simulação permite reduzir significativamente a dependência da equipa de desenvolvimento, aumentando a autonomia da equipa funcional na validação e construção de novos *templates*.

**Template:**

```
{
  "policy": {
    "policyNumber": "${policy.policyNumber}",
    "coverages": [
      <#list policyCoverageList as policyCoverage>
      {
        "capital": "${policyCoverage.capital}",
        "description": "${policyCoverage.description}"
      }
      <#if policyCoverage_has_next>,</#if>
      </#list>
    ],
    "policyState": "${policy.policyState}"
  }
}
```

**Output Gerado (JSON):**

```
{
  "policy": {
    "policyNumber": "30500",
    "coverages": [
      {
        "capital": "955608.00",
        "description": "Morte por acidente circulação"
      },
      {
        "capital": "955608.00",
        "description": "Morte por acidente"
      },
      {
        "capital": "955608.00",
        "description": "Invalidez absoluta definitiva"
      }
    ],
    "policyState": "0"
  }
}
```

Figura 4.3: Simulação de Geração de Documentos

# 5. Testes e Validação

Este capítulo descreve as atividades realizadas para assegurar que a solução de emissão documental cumpre os requisitos funcionais, mantém a compatibilidade com a versão anterior do sistema e apresenta melhorias significativas de desempenho. Assim, foi preparado um ambiente de testes, executados testes funcionais e efetuadas medições de desempenho para validar as otimizações introduzidas.

## 5.1 Preparação do Ambiente de Testes

Com o objetivo de assegurar a capacidade de repetição e consistência dos ensaios, foi preparado um ambiente de testes totalmente controlado. Para tal, obteve-se um *backup* de uma base de dados Oracle de testes da empresa, contendo dados e configurações de alguns produtos semelhantes ao usado por um cliente real. Com este ponto de partida, foram criados *scripts* de *baseline* e de restauro, bem como configurados cenários de negócio que permitissem a execução de testes reais e representativos.

A base de dados foi preparada para suportar *flashback*, permitindo regressar a um estado inicial definido antes de cada conjunto de testes. Para isso, foram criados dois *scripts* principais: um para restaurar a base de dados ao ponto de *baseline* e outro para criar um novo ponto de restauro.

### 5.1.1 Script de Restauro da Base de Dados

O *script* apresentado abaixo 5.1 executa o *flashback* para o ponto de restauro `baseline`, reabrindo a base de dados pronta para utilização nos testes.

```
whenever oserror exit failure
whenever sqlerror exit failure
set time on
set timing on
set echo on
spool restore.log
select to_char(sysdate, 'yyyy-mm-dd hh24:mi:ss') "Flashback database
  on:" from dual;
shutdown immediate;
startup mount;
flashback database to restore point baseline;
alter database open resetlogs;
drop restore point baseline;
create restore point baseline guarantee flashback database;
alter pluggable database all open;
spool off
exit success
```

Código 5.1: *Script* de Restauro

### 5.1.2 Script de Criação do Ponto de Restauro

O *script* 5.2 apresenta as instruções utilizadas para criar o ponto de restauro `baseline`, que serve de referência para regressar ao estado inicial antes de cada execução de testes.

```
shutdown immediate;
startup mount;
create restore point baseline guarantee flashback database;
alter database open;
alter pluggable database all open;
```

Código 5.2: *Script* de Criação do Ponto de Restauro

### 5.1.3 Verificação dos Pontos de Restauro

Para validar que o ponto de restauro foi corretamente criado, executa-se a consulta apresentada abaixo 5.3.

```
SELECT database_incarnation#, name, time, guarantee_flashback_database
FROM v$restore_point
ORDER BY time;
```

Código 5.3: Consulta Pontos de Restauro

A Figura 5.1 apresenta uma evidência obtida, confirmando que o ponto `BASELINE` foi criado com *flashback* garantido.

```
DATABASE_INCARNATION#|NAME     |TIME                    |GUARANTEE_FLASHBACK_DATABASE
---------------------+---------+------------------------+---------------------------
                   4|BASELINE |2025-07-26 19:55:59.000|YES
```

Figura 5.1: Evidência Criação de Ponto de Restauro

## 5.2 Testes Funcionais

Os testes funcionais tiveram como objetivo verificar a consistência dos dados produzidos pela nova solução em comparação com o sistema existente, assegurando que a recolha, processamento e compilação da informação resultam em *outputs* equivalentes.

Estes testes dividiram-se em duas vertentes principais:

- Testar individualmente diferentes tipos de documentos, como apólices, recibos e indemnizações.
- Comparação sistemática de resultados gerados a partir das mesmas apólices, garantindo que as diferenças se limitam a variações residuais.

Para a comparação dos resultados foi utilizado um *plugin* específico no Notepad++, que permitiu identificar rapidamente diferenças na estrutura e nos valores gerados. O processo de validação foi conduzido de forma estruturada, seguindo as etapas abaixo:

- Verificação dos dados gerais da apólice.
- Validação das sub-estruturas, incluindo:
  - Coberturas.

- Beneficiários.
- Tomador.
- Restantes elementos relevantes do documento.

A Figura 5.2 apresenta um exemplo visual de comparação entre os resultados obtidos nas duas soluções, evidenciando que as diferenças identificadas foram mínimas e, na sua maioria, irrelevantes para o contexto funcional.

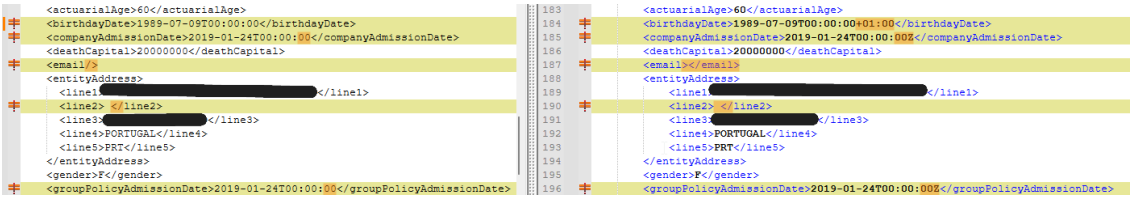


Figura 5.2: Comparação Parcial Solução Existente com a Nova Solução

Dado o tempo disponível para a fase de implementação, apenas o *template* da apólice foi totalmente desenvolvido e validado, garantindo que a sua estrutura e conteúdo para um determinado produto coincidiam com os resultados do sistema existente. O *template* de recibo foi parcialmente implementado, ficando a sua conclusão e a criação dos restantes *templates* como trabalho futuro.

## 5.3 Testes de Desempenho

Os testes de desempenho tiveram como objetivo avaliar comparativamente a solução anterior e a nova implementação, garantindo não apenas a manutenção da integridade dos dados gerados, mas também a obtenção de ganhos significativos nos tempos de processamento.

O processo de negócio escolhido para estes testes corresponde à emissão de apólices, que inclui as validações necessárias, mas cujo tempo de execução é maioritariamente consumido na recolha e compilação de dados para a geração dos *outputs*. Assim, os resultados obtidos refletem diretamente a eficiência da nova abordagem na otimização desta etapa crítica.

Para assegurar a fiabilidade da medição, foram definidos cenários controlados com dados e condições idênticas para ambas as soluções. Em cada uma das **5 execuções realizadas para cada versão** do *software*, foi processado o mesmo conjunto de **2000 emissões de apólices**, evitando qualquer variação provocada por diferenças de dados ou carga no sistema.

Todas as execuções foram efetuadas na mesma máquina e sob as mesmas condições, garantindo a ausência de processos paralelos que pudessem introduzir sobrecarga ou distorção nos tempos registados.

A Figura 5.3 apresenta a comparação dos tempos médios obtidos: a solução anterior registou um tempo médio de **14,71 minutos**, enquanto a nova solução obteve **5,07 minutos**, representando uma melhoria substancial no desempenho global, com uma redução aproximada de **65,5%** no tempo médio de processamento. Esta melhoria deve-se, principalmente, à eliminação de chamadas remotas para recolha de dados, substituídas por invocações diretas a serviços internos, bem como à otimização do processo de preparação e compilação dos dados para geração dos *outputs*.

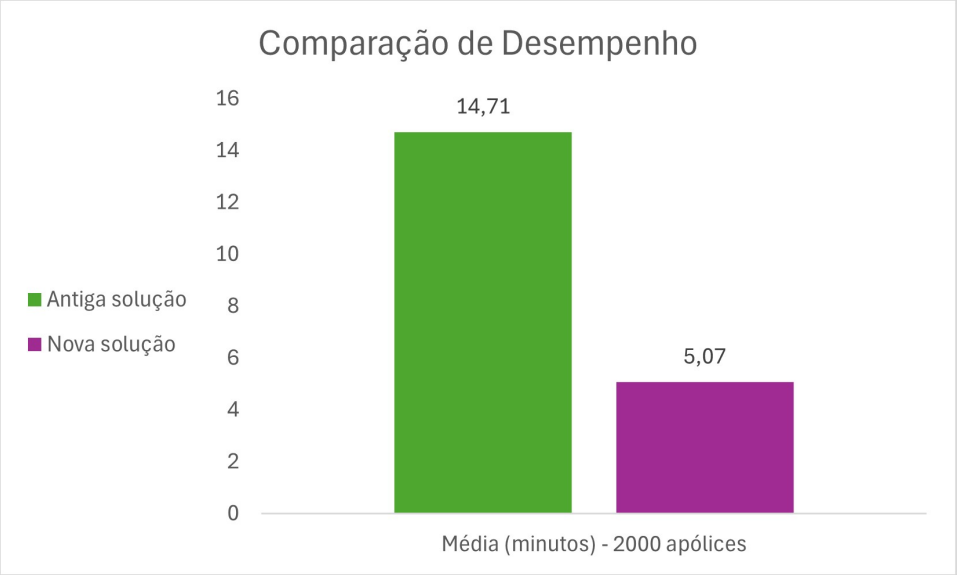



Figura 5.3: Comparação de Tempos Médios Solução Existente com a Nova Solução

A melhoria obtida representa um avanço substancial no desempenho, possibilitando que, mesmo em períodos de elevada carga, a solução consiga processar um número muito maior de documentos no mesmo espaço de tempo, minimizando pontos de estrangulamento e aumentando a agilidade na resposta às exigências operacionais.

## 5.4 Avaliação dos Requisitos Cumpridos

Os objetivos definidos na Secção 1.4 foram traduzidos em requisitos funcionais e não funcionais (Capítulo 3), que orientaram o desenho e a implementação da solução. Nesta fase, os testes funcionais (Secção 5.2) e de desempenho (Secção 5.3) permitiram verificar de forma objetiva se esses requisitos foram cumpridos, estabelecendo a ligação entre a conceção inicial e os resultados obtidos.

A Tabela 5.1 apresenta uma síntese dessa avaliação, evidenciando para cada requisito as evidências recolhidas durante os testes e o respetivo grau de cumprimento.

Tabela 5.1: Síntese de Validação dos Requisitos

| **Requisito** | **Evidências de Validação** | **Resultado** |
|---|---|---|
| RF-01 | Geração de documentos em JSON e XML<br>Estrutura preparada para extensão a novos formatos | Cumprido |
| RF-02 | Emissão confirmada em contexto interativo (via *frontend*)<br>Execução em processos *batch* controlados | Cumprido |
| RF-03 | Validação de estrutura realizada no momento da geração<br>Simulações com dados de teste confirmaram conformidade | Cumprido |
| RF-04 | Criação e edição de *templates* no *frontend*<br>Reutilização e parametrização confirmadas | Cumprido |
| RF-05 | Integração validada com *life-ear* e *batch-life-ear*<br>Eliminação de dependências remotas | Cumprido |

| Requisito | Evidências de Validação | Resultado |
|---|---|---|
| RNF-01 | Testes com elevada carga - 2000 emissões apólice<br>Resultados indicam redução de 65,5% do tempo | Cumprido |
| RNF-02 | Comunicação realizada via invocação interna Spring<br>Dependência de REST eliminada | Cumprido |
| RNF-03 | Modularidade assegurada pelo módulo `life-output-bll`<br>Flexibilidade para futura substituição de componentes | Cumprido |
| RNF-04 | Consistência dos dados verificada nos testes funcionais<br>Integridade assegurada durante todo o processo | Cumprido |
| RNF-05 | Execução validada em JBoss/Wildfly<br>Desacoplado do ciclo de vida das restantes aplicações | Cumprido |

A análise evidencia que todos os requisitos funcionais e não funcionais foram integralmente cumpridos, confirmando que a solução implementada responde de forma eficaz às necessidades técnicas e funcionais inicialmente levantadas.

# 6. Conclusões

Este trabalho teve como objetivo conceber e implementar uma nova solução para o módulo de recolha e envio de dados de uma organização do setor segurador, procurando ultrapassar as limitações detetadas na arquitetura anterior. A análise inicial evidenciou problemas como elevado acoplamento entre componentes, tempos de processamento excessivos, custos de manutenção elevados e dificuldade em lidar com múltiplos formatos de documentos. Estes fatores comprometiam a eficiência operacional e a capacidade de resposta da organização perante as exigências do mercado.

A solução desenvolvida adotou uma arquitetura modular e extensível, suportando diferentes formatos de documentos e otimizando o fluxo de recolha e compilação de dados. Foi ainda criada uma interface *frontend* que, para além da edição de *templates*, inclui mecanismos de simulação e validação em tempo real, conferindo maior autonomia às equipas funcionais e reduzindo a dependência direta da equipa técnica. A fundamentação técnica seguiu uma metodologia baseada no PRISMA, permitindo identificar as tecnologias e práticas mais adequadas, como o uso de *template engines* e padrões arquiteturais flexíveis, assegurando que a solução proposta estivesse alinhada com as tendências e requisitos atuais do setor segurador.

Ao longo do projeto, foram igualmente consideradas questões éticas e de conformidade legal, nomeadamente o cumprimento das regulamentações de privacidade e proteção de dados pessoais, como o RGPD. Todas as decisões de implementação tiveram em conta a minimização da exposição de informação sensível, garantindo a confidencialidade, integridade e disponibilidade dos dados processados, bem como a transparência na sua utilização.

## 6.1 Objetivos Alcançados

A validação da solução foi realizada através de testes funcionais (Secção 5.2) e de desempenho (Secção 5.3), conduzidos em ambiente controlado e com métricas consistentes. Estes testes permitiram verificar o cumprimento dos requisitos definidos no Capítulo 3, assegurando não apenas a robustez técnica da implementação, mas também a satisfação dos objetivos estratégicos apresentados na Secção 1.4.

A Tabela 6.1 apresenta uma síntese dos resultados obtidos, relacionando cada objetivo com as evidências recolhidas durante os testes e validações.

Tabela 6.1: Validação Cruzada entre Objetivos e Requisitos

| Objetivo | Evidências de Validação | Resultado |
|---|---|---|
| OBJ-01 | Arquitetura modular<br>Redução da duplicação de código<br>Criação e manutenção de *templates* mais ágil<br>Menor dependência técnica | Cumprido |
| OBJ-02 | Testes de desempenho com 5 execuções de 2000 apólices cada<br>Redução significativa do tempo médio de processamento<br>Eliminação de chamadas remotas | Cumprido |
| OBJ-03 | Suporte a JSON e XML<br>Estrutura extensível para novos formatos<br>Compatibilidade retroativa prevista | Cumprido |

No que respeita ao OBJ-01, relativo à redução do custo de desenvolvimento e manutenção, verificou-se que a arquitetura modular centralizada no módulo de geração documental e a flexibilidade do editor de *templates* permitem efetuar alterações de forma mais ágil e com menor dependência técnica. Esta evidência foi suportada pelos requisitos RF-04, RF-05 e RNF-03, diretamente validados nos testes.

O OBJ-02, que visava a redução do tempo de recolha e compilação de dados para geração de documentos, foi comprovado através dos testes de desempenho descritos na Secção 5.3, onde se realizaram cinco execuções controladas com 2000 emissões de apólices cada. Os ganhos registados em tempo médio de processamento, aliados à eliminação de chamadas remotas e à utilização de serviços internos via Spring, confirmaram o cumprimento dos requisitos RF-02, RF-03 e RNF-01.

Por último, o OBJ-03, centrado na maximização da compatibilidade e no suporte a múltiplos formatos, foi igualmente cumprido. A solução atualmente gera JSON e XML, estando preparada para a integração de novos formatos, garantindo ainda compatibilidade retroativa. Este objetivo foi validado através dos requisitos RF-01 e RNF-04.

## 6.2 Desafios e Limitações

O desenvolvimento do projeto apresentou alguns desafios significativos, principalmente na integração com sistemas e módulos legados, o que exigiu especial cuidado para garantir a compatibilidade. A criação de um ambiente de testes local com base de dados Oracle, incluindo scripts de *baseline* e restauro, revelou-se fundamental para assegurar a repetibilidade e consistência dos testes.

O *template* da apólice foi concluído e validado com sucesso, dado tratar-se do elemento com maior complexidade. Por sua vez, o *template* do recibo, considerado como segunda prioridade, ficou apenas parcialmente desenvolvido, enquanto os restantes documentos foram adiados para fases posteriores. Estas limitações não comprometem a arquitetura geral, mas representam áreas onde a cobertura funcional poderá e deverá ser alargada.

## 6.3 Trabalho Futuro

Como evolução futura, recomenda-se a conclusão e validação de todos os *templates* necessários, de forma a garantir cobertura total dos conceitos de negócio exigidos no mercado segurador.

No que respeita à vertente de desempenho, recomenda-se uma revisão aprofundada das *queries* de recolha de dados, com o objetivo de otimizar ao máximo o processo de extração e minimizar tempos de resposta. Esta otimização deverá incluir a análise de índices, reestruturação de junções e, sempre que possível, a redução do volume de dados transferidos, garantindo assim maior eficiência e escalabilidade para cenários de elevada carga.

Adicionalmente, o módulo *frontend* desenvolvido deverá ser integrado nos projetos internos da empresa, sofrendo as adaptações necessárias para cumprir as *guidelines* visuais e funcionais definidas pela organização.

Por fim, deverá ser avaliada a possibilidade de reutilização do *software* produzido em outros módulos da organização, potenciando a uniformização de processos e a redução de duplicação de esforço no desenvolvimento.

## 6.4 Apreciação Final

A solução desenvolvida representa um passo decisivo na modernização do processo de recolha e compilação de dados no contexto segurador. A implementação resultou num sistema mais rápido, flexível e sustentável, que não só resolve as limitações previamente identificadas, como estabelece uma base sólida para evolução tecnológica contínua. Os ganhos obtidos em desempenho e compatibilidade reforçam a competitividade da organização e a sua capacidade de adaptação num setor caracterizado por exigências elevadas. Assim, este trabalho demonstra que a adoção de uma abordagem arquitetural moderna, aliada a boas práticas de desenvolvimento e validação rigorosa, constitui um caminho sólido para garantir sistemas mais eficientes, adaptáveis e preparados para os desafios futuros.

# Bibliografia


Apache Software Foundation (2024a). *Apache FreeMarker - Downloads*. [Online; acedido 29-Dezembro-2024]. url: https://freemarker.apache.org/freemarkerdownload.html.

– (2024b). *Apache FreeMarker - Home*. [Online; acedido 29-Dezembro-2024]. url: https://freemarker.apache.org/.

– (2024c). *Apache FreeMarker - Try FreeMarker Online*. [Online; acedido 29-Dezembro-2024]. url: https://try.freemarker.apache.org/.

– (2024d). *Apache FreeMarker Manual*. [Online; acedido 16-Dezembro-2024]. url: https://freemarker.apache.org/docs/index.html.

– (2024e). *Apache Velocity Engine Overview*. [Online; acedido 16-Dezembro-2024]. url: https://velocity.apache.org/engine/devel/overview.html.

– (2024f). *Apache Velocity Engine Overview*. [Online; acedido296-Dezembro-2024]. url: https://velocity.apache.org/engine/devel/developer-guide.html.

Ardakany, A. R. et al. (2020). «Mustache: Multi-scale detection of chromatin loops from Hi-C and Micro-C maps using scale-space representation». Em: *Genome Biology*. [Online; acedido 16-Dezembro-2024]. doi: 10.1101/2020.02.24.963579. url: https://doi.org/10.1101/2020.02.24.963579.

Badmus, Emmanuel (2023). «Optimized Strategy for Inter-Service Communication in Microservices». Em: *ResearchGate*. [Online; acedido 30-Dezembro-2024]. url: https://www.researchgate.net/publication/369052693_Optimized_Strategy_for_Inter-Service_Communication_in_Microservices.

Baeldung (2024). *Introduction to Mustache*. [Online; acedido 29-Dezembro-2024]. url: https://www.baeldung.com/mustache.

Carvalho, F. e Luís Duarte (2019). «HoT: Unleash Web Views with Higher-order Templates». Em: *Proceedings of the International Conference on Web Engineering*. [Online; acedido 16-Dezembro-2024], pp. 118–129. doi: 10.5220/0008167701180129. url: https://doi.org/10.5220/0008167701180129.

Carvalho, Fernando e Pedro Fialho (2023). «Enhancing SSR in Low-Thread Web Servers: A Comprehensive Approach for Progressive Server-Side Rendering with any Asynchronous API and Multiple Data Models». Em: *Proceedings of the International Conference on Web Systems*. [Online; acedido 16-Dezembro-2024], pp. 37–48. doi: 10.5220/0012165300003584. url: https://doi.org/10.5220/0012165300003584.

Cleva (2024a). *Cleva Solutions*. Online report. [Online; acedido 25-Novembro-2024]. url: https://cleva-solutions.com/.

– (2024b). *PR e Notícias - A Anacap entra em negociações de exclusividade para adquirir a Cleva, um fornecedor líder de soluções de software para seguros da Inetum*. Online report. [Online; acedido 13-Dezembro-2024]. url: https://cleva-solutions.com/pt-pt/a-anacap-entra-em-negociacoes-de-exclusividade-para-adquirir-a-cleva-um-fornecedor-lider-de-solucoes-de-software-para-seguros-da-inetum/.

Credited Tecnologia (2024). *O que é: Handlebars - Entenda a Biblioteca de Templates*. [Online; acedido 29-Dezembro-2024]. url: https://tecnologia.credited.com.br/glossario/o-que-e-handlebars-biblioteca-de-templates/.

Dhalla, Hardeep Kaur (2020). «A Performance Analysis of Native JSON Parsers in Java, Python, MS.NET Core, JavaScript, and PHP». Em: *2020 16th International Conference*

*on Network and Service Management (CNSM)*. [Online; acedido 16-Dezembro-2024], pp. 1–5. doi: 10.23919/CNSM50824.2020.9269101. url: https://doi.org/10.23919/CNSM50824.2020.9269101.

DR (2020). *Regulamento do Código de Boas Práticas e de Conduta do Instituto Politécnico do Porto*. Online report. [Online; acedido 27-Novembro-2024]. url: https://diariodarepublica.pt/dr/detalhe/despacho/11171-2020-148327696.

Escola de Tecnologia (2024). *Handlebars: Framework de Templates HTMLs*. [Online; acedido 29-Dezembro-2024]. url: https://otecnico.org/aprendendo-handlebars-framework-de-templates-htmls/.

Google LLC (2024a). *Anatomy of Components - Angular*. [Online; acedido 21-Dezembro-2024]. url: https://angular.dev/guide/components.

– (2024b). *Angular Material UI Component Library*. [Online; acedido 21-Dezembro-2024]. url: https://material.angular.io/.

– (2024c). *Home - Angular*. [Online; acedido 21-Dezembro-2024]. url: https://angular.dev/.

– (2024d). *Introduction to the Angular Documentation*. [Online; acedido 21-Dezembro-2024]. url: https://angular.io/docs.

Gotterbarn, Don, Keith Miller e Simon Rogerson (nov. de 1997). «Software engineering code of ethics». Em: *Commun. ACM* 40.11, pp. 110–118. issn: 0001-0782. doi: 10.1145/265684.265699. url: https://doi.org/10.1145/265684.265699.

Handlebars Community (2024). *Handlebars Documentation*. [Online; acedido 16-Dezembro-2024]. url: https://handlebarsjs.com/guide.

Handlebars.js (2024). *When (not) to use Handlebars?* [Online; acedido 29-Dezembro-2024]. url: https://handlebarsjs.com/installation/when-to-use-handlebars.html.

Harrand, Nicolas et al. (2021). «The Behavioral Diversity of Java JSON Libraries». Em: *2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE)*. [Online; acedido 16-Dezembro-2024], pp. 412–422. doi: 10.1109/ISSRE52982.2021.00050. url: https://doi.org/10.1109/ISSRE52982.2021.00050.

Inetum (2024). *Gfi adquire i2S para construir uma solução ponta a ponta completa para seguradoras*. Online report. Accessed: 25 Nov 2024. url: https://www.inetum.com/pt-pt/portugal/noticias/gfi-adquire-i2s-para-construir-uma-solucao-ponta-ponta-completa-para-seguradoras.

James Ingham, Inna Agamirzian e Rika Narisawa (nov. de 2024). «Forecast: Enterprise IT Spending for the Insurance Market, Worldwide, 2022-2028, 3Q24 Update». Em: *Gartner Research*, pp. 1–7.

James Ingham Laurie Shotton, Sham Gill (2024). «Top Technology Trends in Insurance for Vendors to Plan for in 2025». Em: *Gartner Research*, pp. 1–20.

Mardan, A. (2018). «Template Engines: Pug and Handlebars». Em: *Pro Express.js: Master Express.js—The Node.js Framework For Your Web Development*. [Online; acedido 16-Dezembro-2024], pp. 113–114. doi: 10.1007/978-1-4842-3039-8_4. url: https://doi.org/10.1007/978-1-4842-3039-8_4.

Mittapalli, Jishnu Saurav e Menaka Pushpa Arthur (2021). «Survey on Template Engines in Java». Em: *ITM Web of Conferences* 37. [Online; acedido 16-Dezembro-2024], p. 01007. doi: 10.1051/itmconf/20213701007. url: https://doi.org/10.1051/itmconf/20213701007.

Mustache Community (2024). *Mustache Documentation*. [Online; acedido 16-Dezembro-2024]. url: https://mustache.github.io/docs.

Newman, Sam (2015). *Building Microservices: Designing Fine-Grained Systems*. [Online; acedido 30-Dezembro-2024]. Sebastopol, CA, USA: O'Reilly Media, Inc. isbn: 978-1491950357.

url: https://github.com/rootusercop/Free-DevOps-Books-1/blob/master/book/Building%20Microservices%20-%20Designing%20Fine-Grained%20Systems.pdf.

NTT Data Corporation (2024). *Modernizing Legacy Systems in Insurance*. [Online; acedido 10-Agosto-2025]. url: https://insurance.nttdata.com/post/modernizing-legacy-systems-in-insurance/.

ONES (2024). *What Techniques Can A Scrum Master Use?* Online report. [Online; acedido 14-Dezembro-2024]. url: https://ones.com/blog/what-techniques-can-a-scrum-master-use.

Oracle Corporation (2024a). *Java Documentation - Get Started*. [Online; acedido 21-Dezembro-2024]. url: https://docs.oracle.com/en/java/.

– (2024b). *Java Platform, Standard Edition Documentation - Releases*. [Online; acedido 21-Dezembro-2024]. url: https://docs.oracle.com/en/java/javase/index.html.

– (2024c). *Java SE - Documentation*. [Online; acedido 21-Dezembro-2024]. url: https://www.oracle.com/java/technologies/javase-documentation.html.

– (2024d). *Oracle Data Guard and RAC*. [Online; acedido 21-Dezembro-2024]. url: https://www.oracle.com/technetwork/pt/database/enterprise-edition/documentation/database-100705-ptb.html.

– (2024e). *Oracle Database 12c: Data Warehousing Guide - Basic Materialized Views*. [Online; acedido 02-Janeiro-2025]. url: https://docs.oracle.com/en/database/oracle/oracle-database/12.2/dwhsg/basic-materialized-views.html.

– (2024f). *Oracle Database 19c: Performance Tuning Guide*. [Online; acedido 30-Dezembro-2024]. url: https://docs.oracle.com/en/database/oracle/oracle-database/19/tgdba/index.html.

– (2024g). *Oracle Database 19c: SQL Tuning Guide*. [Online; acedido 30-Dezembro-2024]. url: https://docs.oracle.com/en/database/oracle/oracle-database/19/tgsql/index.html.

– (2024h). *Oracle Database Documentation*. [Online; acedido 21-Dezembro-2024]. url: https://docs.oracle.com/en/database/oracle/oracle-database.

– (2024i). *Oracle Performance and Scalability*. [Online; acedido 21-Dezembro-2024]. url: https://www.oracle.com/technetwork/pt/database/enterprise-edition/documentation/database-100705-ptb.html.

– (2024j). *Oracle Programming Languages Integration*. [Online; acedido 21-Dezembro-2024]. url: https://www.oracle.com/br/database/technologies/appdev/sql.html.

– (2024k). *Oracle Security Features*. [Online; acedido 21-Dezembro-2024]. url: https://www.oracle.com/technetwork/pt/database/enterprise-edition/documentation/database-100705-ptb.html.

– (2024l). *Oracle SQL Features*. [Online; acedido 21-Dezembro-2024]. url: https://www.oracle.com/br/database/technologies/appdev/sql.html.

– (2024m). *The Java™ Tutorials*. [Online; acedido 21-Dezembro-2024]. url: https://docs.oracle.com/javase/tutorial/.

Portal Micilini (2024). *HandleBars*. [Online; acedido 29-Dezembro-2024]. url: https://micilini.com/conteudos/nodejs/handlebars.

PRISMA (2024). *Welcome to the Preferred Reporting Items for Systematic reviews and Meta-Analyses (PRISMA) website*. Online report. [Online; acedido 14-Dezembro-2024]. url: https://www.prisma-statement.org.

Rocketseat Blog (2024). *Construindo sua primeira página com Thymeleaf*. [Online; acedido 29-Dezembro-2024]. url: https://www.rocketseat.com.br/blog/artigos/post/java-construindo-sua-primeira-pagina-com-thymeleaf.

Saltuk, O. e I. Kosan (2014). «Design and creation». Em: *Presentation, Ludwig Maximilian University of Munich*. [Online; acedido 19-Dezembro-2024]. url: https://www.medien.ifi.lmu.de/lehre/ss14/swal/presentations/topic2-saltuk_kosan-DesignAndCreation.pdf.

ScrumGuides (2024). *The 2020 Scrum Guide*. Online report. [Online; acedido 14-Dezembro-2024]. url: https://scrumguides.org/scrum-guide.html.

SpringHow (2024). *FreeMarker vs Groovy vs Mustache vs Thymeleaf*. [Online; acedido 29-Dezembro-2024]. url: https://springhow.com/spring-boot-template-engines-comparison/.

Sun, Changxia et al. (2020). «Research and Application of Data Exchange Based on JSON». Em: *2020 Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC)*. [Online; acedido 16-Dezembro-2024], pp. 349–355. doi: 10.1109/IPEC49694.2020.9115155. url: https://doi.org/10.1109/IPEC49694.2020.9115155.

Thymeleaf Project (2024a). *Thymeleaf - Home*. [Online; acedido 29-Dezembro-2024]. url: https://www.thymeleaf.org/.

– (2024b). *Thymeleaf Integration with Spring Boot*. [Online; acedido 29-Dezembro-2024]. url: https://www.thymeleaf.org/doc/tutorials/3.0/usingthymeleaf.html.

Thymeleaf Team (2024). *Thymeleaf Documentation*. [Online; acedido 16-Dezembro-2024]. url: https://www.thymeleaf.org/documentation.html.

# A. Project Charter

O *Project Charter* estabelece a visão geral do projeto, incluindo os seus objetivos, âmbito, principais entregáveis, *stakeholders* e restrições. Este documento é essencial para garantir o alinhamento entre todas as partes interessadas e para definir os critérios de sucesso do projeto.

O *Project Charter* completo encontra-se neste apêndice e pode ser consultado na página seguinte.

# PROJECT CHARTER

| 1. Informação geral | |
|---|---|
| **Nome do Projeto:** | **Desenvolvimento de uma Nova Solução de Recolha e Envio de Dados para a Gestão Documental** |
| **Sponsor:** | **Isabel Azevedo, Bruno Silva, Daniel Barciela** |
| **Departamento** | CLEVA – Product Delivery |

**2. Equipa do Projeto**

| Cargo | Nome | Departamento | Contacto Tel | E-mail |
|---|---|---|---|---|
| Orientador | Bruno Silva | ISEP – Informática | | bms@isep.ipp.pt |
| Supervisor | Daniel Barciela | CLEVA – Product Delivery | | daniel.barciela@cleva-solutions.com |
| | | | | |

**3. Stakeholders**

| Nome | Poder | Interesse |
|---|---|---|
| Clientes | Consumidor final | Diminuir tempo de espera e custo com alterações solicitadas pelos clientes |
| Equipa suporte | Customização | Tornar a customização da solução independente do departamento de engenharia. |
| Programadores | Implementação | Diminiução do tempo e custo de desenvolvimento quando é necessário uma alteração a pedido do cliente. |
| Daniel Barciela (Supervisor) | Responsável pelo software | Melhorar o componente de software |

**4. Âmbito**

**Problema / justificação**

O módulo atual, responsável pela geração e envio de documentos críticos para o negócio, enfrenta desafios significativos de manutenção e evolução devido à sua arquitetura com alto acoplamento.

Cada alteração realizada no módulo pode comprometer a compatibilidade entre versões, impedindo um versionamento eficaz e tornando difícil a adaptação a novos requisitos do mercado.

Além disso, a falta de flexibilidade no design estrutural limita a capacidade de resposta do sistema, resultando em atrasos e dificuldades na entrega de documentos em tempo útil.

Essa situação impacta negativamente tanto a eficiência operacional quanto a satisfação dos clientes, reforçando a necessidade urgente de reestruturação arquitetural que possibilite uma evolução mais ágil e sustentável, com compatibilidade aprimorada e suporte para múltiplos formatos de documento.

**Objetivos do projeto**

- **Otimização do Processo de Desenvolvimento e Suporte**
  - **Objetivo:** Minimizar o custo de desenvolvimento de novas funcionalidades e suporte entre versões.
  - **Questão de Pesquisa:** Como é possível reestruturar a arquitetura do módulo para reduzir o custo de desenvolvimento e manutenção?
  - **Hipótese:** Uma arquitetura modularizada permitirá desenvolvimento e suporte mais eficientes, reduzindo o custo e complexidade de cada nova implementação.
- **Redução do Tempo de Geração e Envio de Documentos**
  - **Objetivo:** Minimizar o tempo necessário para a geração de documentos (recolha de dados e envio).
  - **Questão de Pesquisa:** Quais mudanças na camada de recolha e envio de dados impactam diretamente a eficiência do sistema?
  - **Hipótese:** Redução de chamadas remotas entre aplicações para recolher dados de um único documento.
- **Compatibilidade e Interoperabilidade Multiformato**
  - **Objetivo:** Maximizar a compatibilidade do sistema, garantindo o suporte a múltiplos formatos de documentos.
  - **Questão de Pesquisa:** Quais são as melhores práticas para garantir compatibilidade entre versões e múltiplos formatos de documento?
  - **Hipótese:** Adotar um modelo de abstração de formatos permitirá ao sistema suportar diversos formatos sem a necessidade de customizações significativas em cada versão.

**Benefícios**

- **Flexibilidade e Customização por Cliente**
  - **Descrição:** Permitir que cada cliente possa configurar templates personalizados para o envio de dados nos documentos de negócio.
  - **Benefício:** Isso possibilita que diferentes clientes adaptem o sistema às suas necessidades específicas, aumentando a satisfação do usuário final e reduzindo a necessidade de modificações no código-base para cada cliente.
- **Facilitação da Manutenibilidade do Software**
  - **Descrição:** Simplificar a estrutura do sistema para facilitar a identificação e correção de problemas, além de permitir atualizações e modificações com impacto mínimo.
  - **Benefício:** Com uma arquitetura mais simplificada, o tempo e o esforço necessários para manutenção e suporte diminuem, o que reduz o custo operacional e melhora a confiabilidade do sistema.
- **Redução de Retrabalho**
  - **Descrição:** Implementar uma camada de configuração que permita ajustes sem necessidade de alterações frequentes no código.
  - **Benefício:** Reduzir o retrabalho e o tempo necessário para ajustes, permitindo um fluxo de desenvolvimento mais ágil e com menos falhas.
- **Aprimoramento da Compatibilidade e Interoperabilidade**
  - **Descrição:** Facilitar a integração do sistema com outros softwares e garantir que ele possa trabalhar com diversos formatos de dados.
  - **Benefício:** Aumentar a interoperabilidade do sistema, permitindo a troca de informações com outros sistemas e o uso de múltiplos formatos de documento sem necessidade de modificações significativas.
- **Escalabilidade**
  - **Descrição:** Estruturar o sistema para que ele possa ser facilmente escalado conforme o crescimento das necessidades dos clientes.
  - **Benefício:** Tornar o sistema preparado para expansão futura, permitindo que novos clientes e funcionalidades sejam adicionados com impacto mínimo na performance e na arquitetura atual.

**Entregáveis**

- **Relatório PREPD**
  - Relatório para a disciplina PREPD, onde tem de constar a revisão literária, estado da arte e planeamento.
- **Relatório DIMEI**
  - Relatório para a disciplina DIMEI, onde deve ter por base o relatório de PREPD, adicionando os detalhes técnicos, evidências dos resultados e conclusão.
  - Descrição detalhada da arquitetura, especificações dos módulos, configurações e dependências.
  - Descrição dos cenários de teste, os casos específicos e os resultados obtidos para validação da coerência e performance.
  - Evidências de melhorias e benefícios alcançados em comparação ao sistema original.
- **Apresentação e Demonstração do Software**
  - Apresentação prática das funcionalidades e do uso do sistema, evidenciando o funcionamento dos novos recursos implementados.
- **Software desenvolvido**
  - Entrega do software desenvolvido

## 5. Tempo

**Milestones / Datas**

- **Planeamento**
  - 15/Nov/2024
- **Relatório PREPD**
  - 04/Jan/2025
- **Análise e design**
  - 17/Mar/2025
- **Desenvolvimento**
  - 19/Mai/2025
- **Testes de coerência e testes performance**
  - 09/Jun/2025
- **Entrega final**
  - 30/Jun/2025

## 6. Custo

**Fontes de custo**

## 7. Pressupostos

- Disponibilidade contínua a todos os recursos técnicos (software, hardware, ferramentas de desenvolvimento e teste)
- Acesso a base de dados para efetuar os testes
- O projeto poderá seguir normas e padrões de documentação e de segurança aplicáveis ao setor, sem necessidade de adaptações específicas.

## 8. Restrições

- O software deve cumprir rigorosamente as políticas de segurança e privacidade de dados, o que pode limitar o acesso e o armazenamento de dados sensíveis durante os testes.
- O sistema precisa ser compatível com versões antigas de software e diferentes formatos de documentos, restringindo o uso de tecnologias mais recentes.

## 9. Riscos

**Riscos identificados**

| Descrição | Causa | Efeito |
|---|---|---|
| Atraso no cronograma | Mudanças nos requisitos ou dificuldades técnicas | Atraso na entrega final e possíveis custos adicionais |
| Incompatibilidade com sistemas legados | Diferença nas versões de software e falta de flexibilidade | Necessidade de reestruturação do software e maior custo |
| Indisponibilidade de recursos técnicos | Falha ou indisponibilidade | Interrupção ou atrasos no desenvolvimento e/ou testes |
| Problemas de desempenho | Arquitetura não otimizada ou sobrecarga de sistema | Diminuição da eficiência e impacto na satisfação do cliente |
| Risco de segurança e privacidade de dados | Vulnerabilidades no sistema ou não conformidade | Possível violação de dados e consequências legais |
| Dificuldades com testes de compatibilidade | Diversidade de formatos e versões a serem suportados | Possíveis falhas de compatibilidade e impacto no utilizador final |

## 10. Aprovação

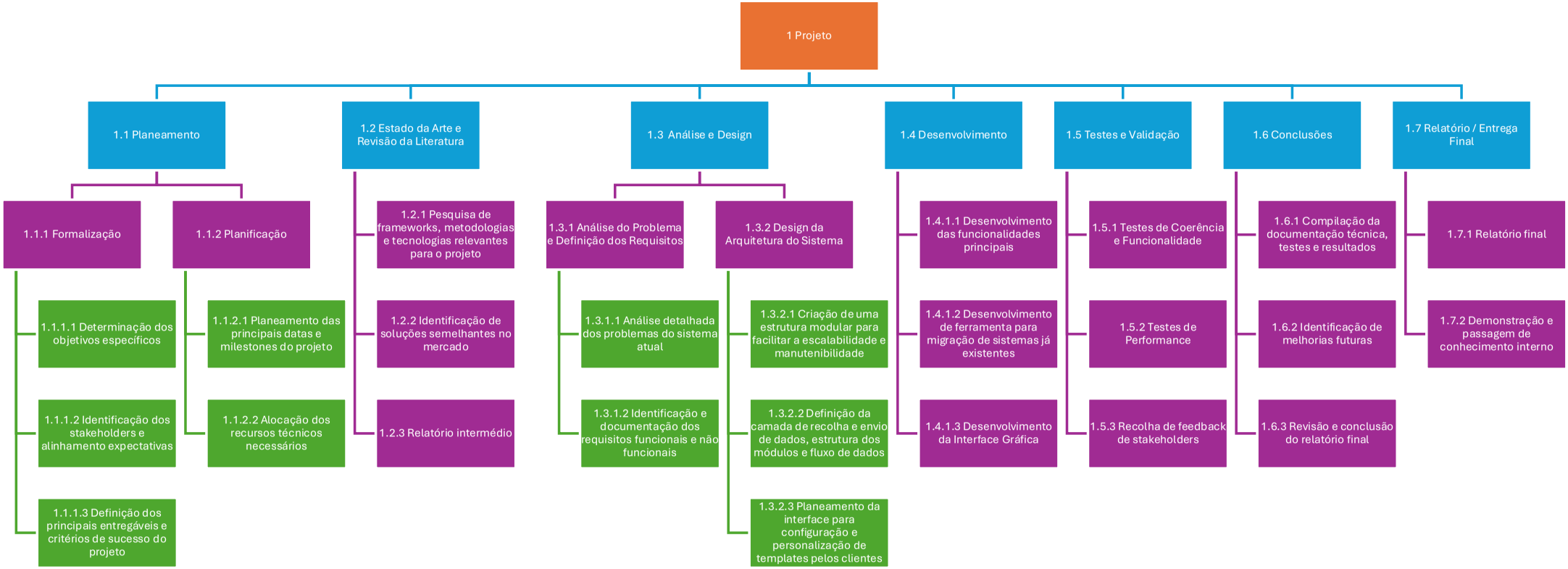
| | Nome | Assinatura | Data (DD/MM/YYYY) |
|---|---|---|---|
| **Orientador** | Bruno Silva | | |
| **Supervisor** | Daniel Barciela | | |
| **Juri** | | | |

**Notas**

# B. Work Breakdown Structure

O WBS organiza as atividades do projeto em uma estrutura hierárquica, facilitando o planeamento, a alocação de recursos e o acompanhamento das tarefas.

O documento completo da WBS encontra-se incluído no apêndice e pode ser consultado na página seguinte.


1 Projeto
1.1 Planeamento
1.1.1 Formalização
1.1.1.1 Determinação dos objetivos específicos
1.1.1.2 Identificação dos stakeholders e alinhamento expectativas
1.1.1.3 Definição dos principais entregáveis e critérios de sucesso do projeto
1.1.2 Planificação
1.1.2.1 Planeamento das principais datas e milestones do projeto
1.1.2.2 Alocação dos recursos técnicos necessários
1.2 Estado da Arte e Revisão da Literatura
1.2.1 Pesquisa de frameworks, metodologias e tecnologias relevantes para o projeto
1.2.2 Identificação de soluções semelhantes no mercado
1.2.3 Relatório intermédio
1.3 Análise e Design
1.3.1 Análise do Problema e Definição dos Requisitos
1.3.1.1 Análise detalhada dos problemas do sistema atual
1.3.1.2 Identificação e documentação dos requisitos funcionais e não funcionais
1.3.2 Design da Arquitetura do Sistema
1.3.2.1 Criação de uma estrutura modular para facilitar a escalabilidade e manutenibilidade
1.3.2.2 Definição da camada de recolha e envio de dados, estrutura dos módulos e fluxo de dados
1.3.2.3 Planeamento da interface para configuração e personalização de templates pelos clientes
1.4 Desenvolvimento
1.4.1.1 Desenvolvimento das funcionalidades principais
1.4.1.2 Desenvolvimento de ferramenta para migração de sistemas já existentes
1.4.1.3 Desenvolvimento da Interface Gráfica
1.5 Testes e Validação
1.5.1 Testes de Coerência e Funcionalidade
1.5.2 Testes de Performance
1.5.3 Recolha de feedback de stakeholders
1.6 Conclusões
1.6.1 Compilação da documentação técnica, testes e resultados
1.6.2 Identificação de melhorias futuras
1.6.3 Revisão e conclusão do relatório final
1.7 Relatório / Entrega Final
1.7.1 Relatório final
1.7.2 Demonstração e passagem de conhecimento interno

# C. Planeamento

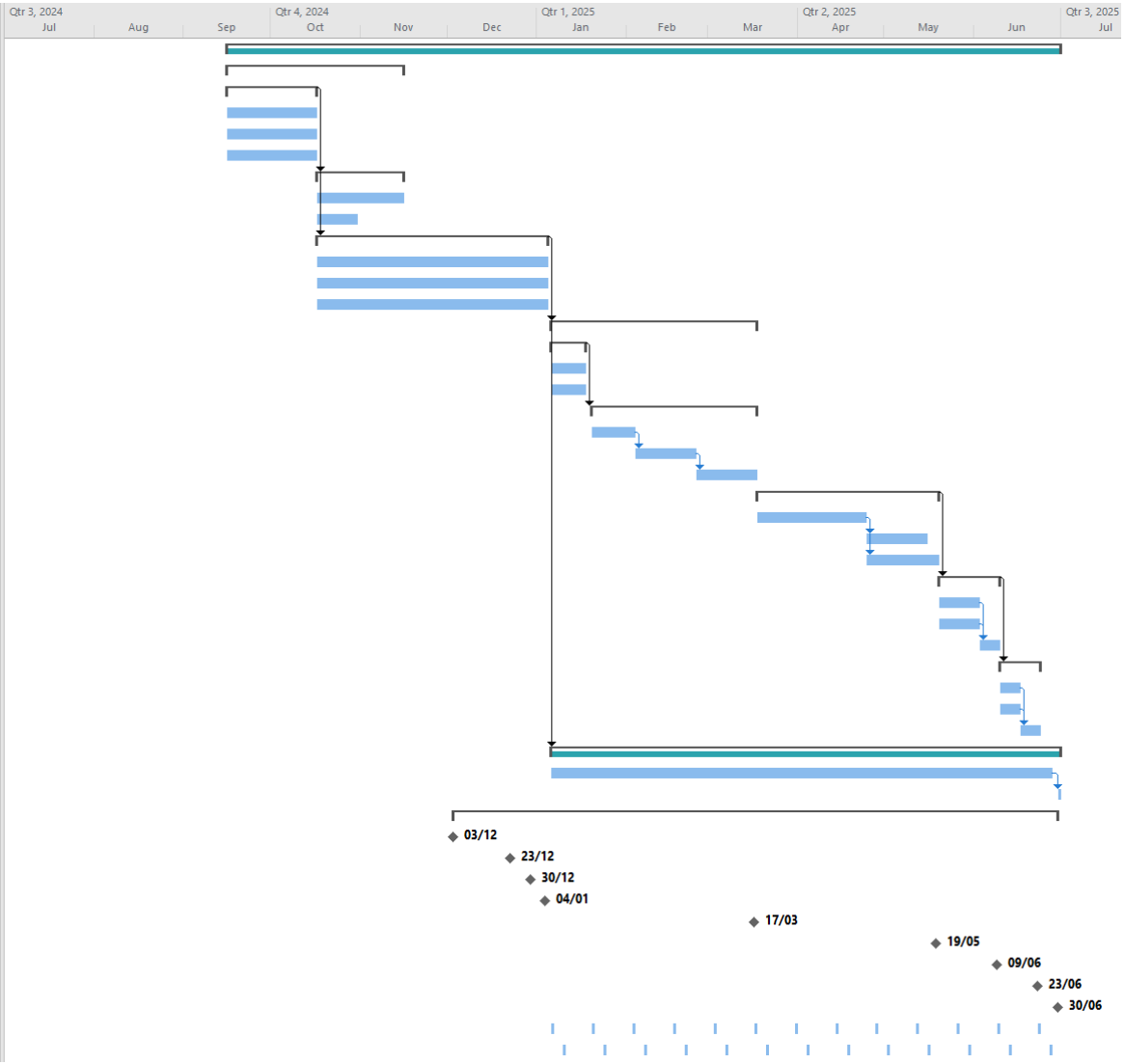
No que respeita ao planeamento, este foi desenvolvido através da ferramenta Microsoft Project. Esta ferramenta, entre outras coisas, disponibiliza alguns suportes visuais interessantes para análise, nomeadamente a *Timeline* e o *Gantt Chart*.

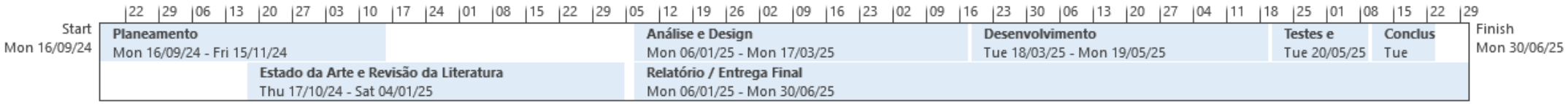
O *Gantt Chart* detalha o cronograma do projeto, apresentando as atividades planeadas, as suas durações e as dependências entre tarefas.

A *Timeline* apresenta as grandes atividades que serão feitas, contextualizadas no tempo.

O gráfico completo e a *Timeline* encontram-se nas páginas seguintes.

| Task Name | Duration | Start | Finish | Predece |
|---|---|---|---|---|
| **Projeto** | **208 days?** | **Mon 16/09/24** | **Mon 30/06/25** | |
| **Planeamento** | **45 days** | **Mon 16/09/24** | **Fri 15/11/24** | |
| **Formalização** | **23 days** | **Mon 16/09/24** | **Wed 16/10/24** | |
| Determinação dos objetivos específicos | 23 days | Mon 16/09/24 | Wed 16/10/24 | |
| Identificação dos stakeholders e alinhamento expectativas | 23 days | Mon 16/09/24 | Wed 16/10/24 | |
| Definição dos principais entregáveis e critérios de sucesso do projeto | 23 days | Mon 16/09/24 | Wed 16/10/24 | |
| **Planificação** | **22 days** | **Thu 17/10/24** | **Fri 15/11/24** | **3** |
| Planeamento das principais datas e milestones do projeto | 22 days | Thu 17/10/24 | Fri 15/11/24 | |
| Alocação dos recursos técnicos necessários | 10 days | Thu 17/10/24 | Wed 30/10/24 | |
| **Estado da Arte e Revisão da Literatura** | **59 days** | **Thu 17/10/24** | **Sat 04/01/25** | **3** |
| Pesquisa de frameworks, metodologias e tecnologias relevantes para o projeto | 59 days | Thu 17/10/24 | Sat 04/01/25 | |
| Identificação de soluções semelhantes no mercado | 59 days | Thu 17/10/24 | Sat 04/01/25 | |
| Relatório intermédio | 59 days | Thu 17/10/24 | Sat 04/01/25 | |
| **Análise e Design** | **51 days** | **Mon 06/01/25** | **Mon 17/03/25** | **10** |
| **Análise do Problema e Definição dos Requisitos** | **10 days** | **Mon 06/01/25** | **Fri 17/01/25** | |
| Análise detalhada dos problemas do sistema atual | 10 days | Mon 06/01/25 | Fri 17/01/25 | |
| Identificação e documentação dos requisitos funcionais e não funcionais | 10 days | Mon 06/01/25 | Fri 17/01/25 | |
| **Design da Arquitetura do Sistema** | **41 days** | **Mon 20/01/25** | **Mon 17/03/25** | **15** |
| Criação de uma estrutura modular para facilitar a escalabilidade e manutenibilidade | 11 days | Mon 20/01/25 | Mon 03/02/25 | |
| Definição da camada de recolha e envio de dados, estrutura dos módulos e fluxo de dados | 15 days | Tue 04/02/25 | Mon 24/02/25 | 19 |
| Planeamento da interface para configuração e personalização de templates pelos clientes | 15 days | Tue 25/02/25 | Mon 17/03/25 | 20 |
| **Desenvolvimento** | **45 days** | **Tue 18/03/25** | **Mon 19/05/25** | |
| Desenvolvimento das funcionalidades principais | 28 days | Tue 18/03/25 | Thu 24/04/25 | |
| Desenvolvimento da ferramenta de migração de sistemas já existentes | 15 days | Fri 25/04/25 | Thu 15/05/25 | 23 |
| Desenvolvimento da interface gráfica | 17 days | Fri 25/04/25 | Mon 19/05/25 | 23 |
| **Testes e Validação** | **15 days** | **Tue 20/05/25** | **Mon 09/06/25** | **22** |
| Testes de coerência e funcionalidade | 10 days | Tue 20/05/25 | Mon 02/06/25 | |
| Testes de performance | 10 days | Tue 20/05/25 | Mon 02/06/25 | |
| Recolha de feedback de stakeholders | 5 days | Tue 03/06/25 | Mon 09/06/25 | 27;28 |
| **Conclusões** | **10 days** | **Tue 10/06/25** | **Mon 23/06/25** | **26** |
| Compilação da documentação técnica, testes e resultados | 5 days | Tue 10/06/25 | Mon 16/06/25 | |
| Identifcação de melhorias futuras | 5 days | Tue 10/06/25 | Mon 16/06/25 | |
| Revisão e conclusão do relatório | 5 days | Tue 17/06/25 | Mon 23/06/25 | 31;32 |
| **Relatório / Entrega Final** | **126 days** | **Mon 06/01/25** | **Mon 30/06/25** | **10** |
| Relatório final | 125 days | Mon 06/01/25 | Fri 27/06/25 | |
| Demonstração e passagem de conhecimento interno | 1 day | Mon 30/06/25 | Mon 30/06/25 | 35 |
| **Milestones** | **150 days** | **Tue 03/12/24** | **Mon 30/06/25** | |
| Rever estrutura relatório | 0 days | Tue 03/12/24 | Tue 03/12/24 | |
| Rever Estado da Arte | 0 days | Mon 23/12/24 | Mon 23/12/24 | |
| Revisão final PREPD | 0 days | Mon 30/12/24 | Mon 30/12/24 | |
| Entrega PREPD | 0 days | Sat 04/01/25 | Sat 04/01/25 | |
| Entrega Análise e Design | 0 days | Mon 17/03/25 | Mon 17/03/25 | |
| Entrega Desenvolvimento | 0 days | Mon 19/05/25 | Mon 19/05/25 | |
| Entrega Resultados Testes | 0 days | Mon 09/06/25 | Mon 09/06/25 | |
| Revisão Relatório | 0 days | Mon 23/06/25 | Mon 23/06/25 | |
| Entrega DIMEI | 0 days | Mon 30/06/25 | Mon 30/06/25 | |
| **Planificação semanal - 30 min. no início de cada quinzena** | **121 days** | **Mon 06/01/25** | **Mon 23/06/25** | |
| **Retrospetiva semanal - 15 min. no final de cada quinzena** | **121 days** | **Fri 10/01/25** | **Fri 27/06/25** | |




22 29 06 13 20 27 03 10 17 24 01 08 15 22 29 05 12 19 26 02 09 16 23 02 09 16 23 30 06 13 20 27 04 11 18 25 01 08 15 22 29
Start
Mon 16/09/24
Planeamento
Mon 16/09/24 - Fri 15/11/24
Análise e Design
Mon 06/01/25 - Mon 17/03/25
Desenvolvimento
Tue 18/03/25 - Mon 19/05/25
Testes e
Tue 20/05/25
Conclus
Tue
Finish
Mon 30/06/25
Estado da Arte e Revisão da Literatura
Thu 17/10/24 - Sat 04/01/25
Relatório / Entrega Final
Mon 06/01/25 - Mon 30/06/25

# D. Riscos do Projeto

Os riscos do projetos estão detalhados no seguinte documento.

# Project X Risk Register

| | | | | |
|---|---|---|---|---|
| Positive Risk Response Options | Exploit | Share | Enhance | Accept |
| Negative Risk Response Options | Avoid | Transfer | Mitigate | Accept |
| Alternate Response Options | Contingency | | | |

| Risk ID | Description | Cause | Effect | Risk Owner | Probability (1-5) | Impact (1-5) | PI Score | Expected Result, No Action | Risk Response Type | Response description |
|---|---|---|---|---|---|---|---|---|---|---|
| | Description of the risk | Cause of the risk | Effect on the project | Name of person who monitors the risk | Group sourced rough estimate of how likely this is to occur | Rough estimate of how significant the impact of this risk | Probability multiplied by Impact | What will happen if the risk becomes an issue and no action is taken | Decision made by group on how to respond to this risk (see above in blue) | How do you know it is time to put the response into play |
| 1 | Atraso no cronograma | Mudanças nos requisitos ou dificuldades técnicas | Atraso na entrega final | Pedro Fernandes | 4 | 4 | 16 | Atraso no cronograma do projeto | Mitigate | Repriotizar tarefas, retirar tarefas menos importantes do âmbito e replanear cronograma. |
| 2 | Indisponibilidade de recursos técnicos | Falha ou indisponibilidade | Interrupção ou atrasos no desenvolvimento e/ou testes | Pedro Fernandes | 3 | 5 | 15 | Paralisação do projeto e impactos na entrega | Mitigate | Aumentar redundâncias em recursos críticos. |
| 3 | Problemas de desempenho | Arquitetura não otimizada ou sobrecarga de sistema | Diminuição da eficiência e impacto na satisfação do cliente | Pedro Fernandes | 4 | 4 | 16 | Perda de confiança do cliente | Avoid | Investir em otimização de arquitetura e ferramentas de monitorazação |
| 4 | Risco de segurança | Vulnerabilidades no sistema ou não conformidade | Possível violação de dados e consequências legais | Pedro Fernandes | 4 | 5 | 20 | Multas e danos de reputação | Mitigate | Implementar auditorias regulares de segurança. |
| 5 | Dificuldades com testes de compatibilidade | Diversidade de formatos e versões a serem suportados | Possíveis falhas de compatibilidade e impacto no utilizador final | Pedro Fernandes | 2 | 3 | 6 | Problemas na adoção da solução pelos utilizadores | Accept | Garantir que o software é configurável para permitir trocar que a solução anterior continue a funcionar. |