Desenvolvimento Front-End e Integração de APIs: Impulsionando a Transformação Digital Empresarial

ANA MARIA PERESTRELO FERREIRA

(Licenciada em Engenharia Informática e Multimédia)

Relatório de Estágio para obtenção do Grau de Mestre em
Engenharia Informática e Multimédia

Orientadores: Doutor Rui Manuel Feliciano de Jesus
Mestre António Sérgio dos Reis Soares

Júri:

Presidente: Doutor Pedro Emanuel Albuquerque E Baptista Dos Santos
Vogais: Doutor Carlos Jorge de Sousa Gonçalves
Doutor Rui Manuel Feliciano de Jesus

outubro 2024

Desenvolvimento Front-End e Integração de APIs: Impulsionando a Transformação Digital Empresarial

ANA MARIA PERESTRELO FERREIRA

(Licenciada em Engenharia Informática e Multimédia)

Relatório de Estágio para obtenção do Grau de Mestre em
Engenharia Informática e Multimédia

Orientadores: Doutor Rui Manuel Feliciano de Jesus, ISEL/DEETC
Mestre António Sérgio dos Reis Soares, Findmore Digital

Júri:

Presidente: Doutor Pedro Emanuel Albuquerque E Baptista Dos Santos, ISEL/DEETC
Vogais: Doutor Carlos Jorge de Sousa Gonçalves, ISEL/DEETC
Doutor Rui Manuel Feliciano de Jesus, ISEL/DEETC

outubro 2024

Agradecimentos

Ao concluir esta etapa, sinto-me profundamente grata a todos aqueles que acompanharam-me no meu percurso académico.

Gostaria de começar por expressar o meu sincero agradecimento aos meus orientadores, Sérgio Soares e Rui Jesus, pela orientação, valiosas sugestões e, acima de tudo, por acreditarem no potencial deste trabalho. A sua experiência, paciência e constante encorajamento foram essenciais para o sucesso deste projeto.

Um agradecimento especial à Findmore Digital, pelo acolhimento e pela confiança depositada ao disponibilizarem este projeto. A vossa abertura, simpatia e apoio permitiram-me desenvolver o trabalho com o rigor e dedicação necessários.

Quero também manifestar a minha gratidão ao Instituto Superior de Engenharia de Lisboa, bem como aos seus docentes e funcionários, pelo contributo indispensável para a minha formação. O conhecimento transmitido, a dedicação e o apoio foram fundamentais para o meu crescimento pessoal e profissional. Ao longo das diversas unidades curriculares, dos docentes que me orientaram e dos colegas com quem tive o privilégio de partilhar este percurso, adquiri uma base sólida de experiência que muito enriqueceu o desenvolvimento deste projeto.

Aos meus pais, o meu mais profundo agradecimento pelo incentivo constante e pelo apoio incondicional, estiveram sempre presentes tanto nos momentos de conquista como nas adversidades. Sem a vossa ajuda, nada teria sido possível.

Por fim, gostaria de agradecer aos meus amigos e amigas pelo apoio incondicional, pelas palavras de incentivo e pelos momentos de descontração que aliviaram o cansaço nos períodos mais exigentes. A vossa amizade e presença foram cruciais para que conseguisse manter a motivação e o equilíbrio ao longo de todo este percurso.

A todos, o meu sincero obrigada.

Resumo

Atualmente, é fundamental que as empresas se adaptem às novas dinâmicas do mercado, reconhecendo a crescente importância da partilha de informações e da inovação tecnológica no ambiente corporativo. Nesse sentido, o presente documento oferece uma análise aprofundada do estágio realizado na Findmore Digital, com foco no desenvolvimento de uma plataforma *online* destinada à partilha de conhecimento.

O estágio foi dividido em duas fases principais: a primeira dedicou-se à formação especializada em webMethods, uma plataforma de integração e gestão de *Application Programming Interfaces* (APIs) da Software AG, amplamente utilizada pela empresa Findmore Digital; a segunda fase centra-se no projeto que serve de base para esta tese, que consiste no desenvolvimento de uma aplicação Web completa, utilizando a framework Angular como parte da solução.

A aplicação Web resultante do projeto foi projetada para ser uma plataforma centralizada e intuitiva, acessível em qualquer dispositivo, permitindo aos utilizadores visualizar publicações de forma eficiente, enquanto os administradores têm a capacidade de criar e gerir o conteúdo. O sistema foi desenvolvido para consolidar e disseminar o vasto conhecimento acumulado pela Findmore Digital ao longo dos anos, oferecendo um repositório centralizado de artigos técnicos e documentação, alinhado com a estratégia de transparência e comunicação aberta da empresa.

Este projeto evidencia a eficácia da utilização de webMethods como solução *middleware*, comprovando a sua robustez na integração e comunicação entre diferentes sistemas corporativos. Além disso, o uso do Angular no desenvolvimento da interface *frontend* demonstrou a versatilidade e a escalabilidade dessa tecnologia na criação de soluções digitais. O portal corporativo desenvolvido reforça a cultura de partilha de informações e promove a inovação dentro da Findmore Digital, consolidando a sua posição como líder no setor.

Para assegurar o adequado funcionamento da aplicação, foram realizados diversos testes que abrangeram tanto a API desenvolvida quanto o fluxo de navegação da plataforma. Esses testes possibilitaram a verificação da comunicação entre o *backend* e o *frontend*, garantindo que todas as funcionalidades operem sem falhas.

Palavras-chave: Findmore Digital, webMethods, Desenvolvimento *Full-Stack*, Angular, Inovação Tecnológica, Partilha de Conhecimento

Abstract

In today's fast-paced environment, it is essential for companies to adapt to new market dynamics, recognizing the increasing importance of information sharing and technological innovation in the corporate landscape. In this context, the present document provides an in-depth analysis of the internship conducted at Findmore Digital, focusing on the development of an online platform aimed at knowledge sharing.

The internship was divided into two main phases: the first phase focused on specialized training in webMethods, a leading integration and Application Programming Interface (API) management platform from Software AG that is widely used by the company; the second phase centers on the project that serves as the basis for this thesis, which involves the development of a complete Web application, utilizing the Angular framework as part of the solution.

The resulting Web application was designed to be a centralized and intuitive platform, accessible on any device, allowing users to efficiently view publications while administrators have the capability to create and manage content. The system was developed to consolidate and disseminate the vast knowledge accumulated by Findmore Digital over the years, providing a centralized repository of technical articles and documentation, aligned with the company's strategy of transparency and open communication.

This project demonstrates the effectiveness of using webMethods as a middleware solution, proving its robustness in integrating and communicating between different corporate systems. Additionally, the use of Angular in developing the frontend interface showcased the versatility and scalability of this technology in creating digital solutions. The developed corporate portal reinforces the culture of information sharing and promotes innovation within Findmore Digital, solidifying its position as a leader in the sector.

To ensure the proper functioning of the application, various tests were conducted covering both the developed API and the navigation flow of the platform. These tests enabled the verification of communication between the backend and the frontend, ensuring that all functionalities operate seamlessly.

Keywords: Findmore Digital, webMethods, Full-Stack Development, Angular, Technological Innovation, Knowledge Sharing

Índice de Conteúdos

Índice de Figuras

Índice de Tabelas

Índice de Listagens

Siglas

API *Application Programming Interface* 2, 3, 5, 7, 8, 9, 21, 22, 24, 25, 27, 28, 30, 31, 44, 45, 47, 48, 54

B2B *Business-to-Business* 8
BLOB *Binary Large Object* 29

CMS *Content Management System* 2, 10
CRUD Create, Read, Update, Delete 28, 53

DMZ *Demilitarized Zone* 7

EA Entidade-Associação 31, 33, 65

HTTP *HyperText Transfer Protocol* 25, 28, 45, 48, 54

ID Identificador Único 34, 35, 38, 39, 40, 42, 43, 46, 48, 49, 54, 56
IDE *Integrated Development Environment* 8
IoT *Internet of Things* 8, 12

JDBC *Java Database Connectivity* 6, 8, 9, 28, 36, 37, 38
JMS *Java Message Service* 6
JSON *JavaScript Object Notation* 25, 45, 48

MVC *Model-View-Controller* 46, 65

REST *Representational State Transfer* 8, 22, 25, 27, 28, 36, 45
RGPD Regulamento Geral sobre a Proteção de Dados 31

SOAP *Simple Object Access Protocol* 8, 22, 25
SQL *Structured Query Language* 22, 23, 35, 41

UI *User Interface* 18, 28

URL *Uniform Resource Locator* 25, 37, 45, 47, 48

XML *eXtensible Markup Language* 6, 25

Capítulo 1

Introdução

No cenário atual de negócios, a partilha de conhecimento dentro das empresas e a inovação tecnológica são pilares fundamentais para o sucesso organizacional. Com a constante evolução das tecnologias e o aumento da demanda por acesso rápido e eficiente a informações, torna-se imperativo para as empresas não só adaptarem suas estratégias, mas também posicionarem-se como líderes na partilha de conhecimento. Este relatório reflete esse movimento, abordando o desenvolvimento de uma plataforma *online* que serve como um canal para educar e informar o público sobre as mais recentes tendências, tecnologias e melhores práticas do setor.

A Findmore Digital [1], reconhecida pela sua parceria com a Software AG [2], viu a necessidade de criar um portal centralizado para consolidar e disseminar o conhecimento acumulado ao longo dos anos. Essa plataforma visa facilitar o acesso a artigos técnicos ou documentação que refletem a *expertise* da empresa, a sua capacidade de inovação e responde às exigências do mercado, demonstrando o compromisso da Findmore Digital com a transparência e a comunicação aberta.

1.1 Motivação

A transformação digital tem levado as empresas a reavaliar as suas estratégias de relacionamento e partilha de conhecimento. Neste contexto, a Findmore Digital identificou uma oportunidade para desenvolver uma plataforma digital que integra informações e facilita uma interação mais direta entre os utilizadores e a empresa.

As motivações para este projeto incluem a criação de um canal moderno que permita a disseminação de informações, de modo a que clientes e outros utilizadores possam aceder facilmente a dados sobre as tecnologias utilizadas pela empresa, bem como informações sobre o software desenvolvido. Através de artigos técnicos e documentação, a Findmore Digital pretende consolidar a sua posição como referência em inovação e tecnologia no mercado digital.

1.2 Objetivos

O objetivo principal deste projeto consiste na implementação de uma aplicação Web que, além de proporcionar conteúdo relevante para os utilizadores, permita aos administradores criar e gerir esse conteúdo de forma eficiente. Para alcançar este objetivo, a aquisição de competências técnicas na utilização do webMethods [3] como solução de integração é considerada fundamental. Os objetivos específicos deste projeto incluem:

- Desenvolver experiência prática na utilização do webMethods para a criação de soluções integradas: esta etapa implica a formação e a aplicação prática da plataforma webMethods, abrangendo as suas funcionalidades e as melhores práticas para a integração de sistemas. O conhecimento adquirido será aplicado na construção de uma *Application Programming Interface* (API) que suporta a comunicação entre o *backend* e o *frontend* da aplicação;
- Implementar uma aplicação Web que satisfaça as necessidades dos utilizadores e promova a interação: a aplicação deverá ser intuitiva e responsiva, permitindo que os utilizadores acedam facilmente a informações relevantes;
- Construir um *backend* que suporte a aplicação Web, assegurando a sua funcionalidade e escalabilidade;
- Criar um *Content Management System* (CMS) que permite uma administração eficiente das publicações: permite que os administradores adicionem, atualizem e os apaguem conteúdos, sem a necessidade de conhecimentos técnicos avançados;
- Implementar medidas de segurança para proteger a integridade da aplicação: implementar práticas de segurança, como autenticação, para proteger as funcionalidades do administrador;
- Realizar testes para garantir a funcionalidade da aplicação: serão realizados testes manuais com o intuito de identificar e corrigir possíveis falhas, assegurando que todas as funcionalidades operam de acordo com as expectativas.

1.3 Contribuições

O projeto em desenvolvimento traz inúmeras contribuições significativas. A implementação da aplicação web proposta fortalece a presença digital da Findmore Digital, oferecendo uma plataforma unificada que centraliza a gestão de informações, promovendo maior transparência e eficiência na comunicação com seus clientes e parceiros.

Este projeto constitui um modelo prático para a implementação de plataformas integradas, passível de ser replicado por outras organizações com objetivos semelhantes. A abordagem adotada demonstra a viabilidade de integrar o webMethods na criação de soluções digitais.

O resultado final é uma aplicação Web, concebida para visualizar e gerir conteúdos criados pela empresa. Esta solução responde às necessidades estratégicas da Findmore Digital e

oferece um meio para disseminar informações a vários públicos-alvo, incluindo clientes e parceiros.

Por último, o desenvolvimento desta aplicação Web não só amplia o portefólio da Findmore Digital, como também estabelece uma referência para iniciativas futuras, alinhando-se com as tendências de mercado e as expectativas dos utilizadores.

1.4 Organização do Documento

O presente documento está estruturado de modo a oferecer uma visão abrangente sobre o projeto desenvolvido ao longo do estágio, abordando desde os fundamentos das tecnologias empregues até à sua implementação prática e validação do sistema. O documento encontra-se dividido em seis capítulos e encontram-se organizados da seguinte forma:

- **Introdução (Capítulo 1)**: descreve a motivação, objetivos, contribuições e a contextualização do projeto, destacando a importância da partilha de conhecimento e da inovação tecnológica para a empresa;
- **Fundamentos e Análise de Aplicações Web Relacionadas (Capítulo 2)**: explora a formação realizada em webMethods e apresenta um estudo de aplicações semelhantes que serviram de referência para o desenvolvimento do projeto;
- **Análise e Tecnologias (Capítulo 3)**: detalha os requisitos funcionais e não funcionais da plataforma, casos de utilização e tecnologias existentes para a construção de uma solução. Este capítulo aborda também o design da aplicação;
- **Implementação do Projeto (Capítulo 4)**: descreve o processo de implementação da plataforma, desde a arquitetura do sistema à integração dos seus componentes, incluindo a configuração de base de dados, o uso de webMethods e a implementação do *frontend* aplicação Web;
- **Validação e Testes (Capítulo 5)**: descreve os testes realizados para validar a API e os testes funcionais da aplicação;
- **Conclusões e Trabalho Futuro (Capítulo 6)**: efetua uma recapitulação do projeto, principais desafios e trabalho futuro.

Capítulo 2

Fundamentos e Análise de Aplicações Web Relacionadas

Neste capítulo, serão explorados os conceitos fundamentais e a análise de aplicações Web semelhantes à que se pretende desenvolver. O capítulo inicia com uma introdução à formação em integração com o webMethods, uma tecnologia central na aplicação proposta, que forneceu uma base sólida para a utilização desta ferramenta.

Além disso, será realizada uma análise comparativa de aplicações Web que possuem características semelhantes, com o objetivo de identificar as melhores práticas e soluções adotadas no mercado. Esta análise visa alinhar os requisitos e funcionalidades da aplicação a desenvolver com as expectativas e necessidades da empresa para o seu novo Website.

2.1 Habilitações em webMethods

O estágio na empresa Findmore Digital, iniciou-se com uma formação extensiva em webMethods, uma tecnologia central nas soluções desenvolvidas pela empresa. A formação começou com uma introdução aos fundamentos do webMethods, seguida de um módulo teórico-prático que consolidava os conceitos teóricos através de atividades práticas. Ao longo da formação, foram abordados dois tópicos principais: integração com webMethods e API *Management*. Cada um dos dois tópicos foi dividido em duas fases: uma dedicada aos conceitos básicos e outra focada nos conceitos essenciais.

No âmbito do API *Management*, a formação realizada centrou-se na temática de garantir a estabilidade, fiabilidade e qualidade das APIs, controlando-as através da plataforma de webMethods API Management da Software AG. A integração dos conhecimentos adquiridos foi realizada no contexto do módulo teórico-prático; contudo, no que diz respeito à aplicação Web a desenvolver, a sua implementação não foi concretizada, uma vez que os conceitos de API *Management* não se alinhavam com os requisitos específicos do projeto em questão.

As subsecções presentes nesta secção, descrevem de forma detalhada os conteúdos abordados durante a formação em cada um dos módulos, apresentando uma visão abrangente dos conhecimentos adquiridos e das competências desenvolvidas durante o estágio.

2.1.1 webMethods Integration Basics

A formação intitulada "WebMethods Integration Basics" cobriu um conjunto de tópicos essenciais para a integração com a plataforma webMethods da Software AG, incluindo:

- Introdução ao Software AG webMethods Integration;
- Obter uma visão geral da plataforma de integração webMethods;
- Utilizar o Software Designer;
- Obter uma visão geral de alto nível dos Document Types;
- Exploração dos Flow Services e Flow steps.

Estes tópicos foram baseados na formação oficial fornecida pela Software AG, disponível na sua plataforma de aprendizagem online [4].

2.1.2 webMethods Integration Essentials

Relativamente ao webMethods Integration, a formação avançou com o módulo "WebMethods Integration Essentials", que expandiu os conhecimentos adquiridos ao explorar os seguintes tópicos avançados:

- Flow Services;
- Mapeamento;
- webMethods Monitor;
- Invocação de serviços e processamento de *eXtensible Markup Language* (XML);
- Serviços Java;
- Debugging de serviços e tratamento de Erros;
- Manipulação básica de Flat Files;
- Serviços Web e segurança de XML;
- Envio de mensagens utilizando webMethods Messaging;
- Envio de mensagens utilizando *Java Message Service* (JMS);
- Triggers de Mensagens em webMethods - Filtragem;
- Tratamento de erros de integração interna;
- Desempenho de serviços;
- webMethods Adapter para *Java Database Connectivity* (JDBC);
- Utilização de serviços e soluções de integração.

Os tópicos abordados neste módulo foram extraídos do conteúdo oficial da Software AG, disponível no guia "webMethods Integration Essentials II" fornecido pela empresa [5].

2.1.3 API Management Basic

Relativamente à gestão de APIs, foi realizada a formação em "API Management Basic", que explorou os seguintes tópicos principais:

- Explicação dos objetivos e benefícios da API Management;
- Conhecer a oferta do webMethods.io API na *cloud* da Software AG e os seus produtos relacionados;
- Criar e testar APIs básicas num API Gateway baseado na *cloud*;
- Oferecer e anunciar APIs através do webMethods Developer Portal baseado na *cloud*;
- Compreensão da necessidade de proteger APIs;
- Definir e testar políticas básicas de APIs no API Gateway.

Estes tópicos foram baseados na formação oficial fornecida pela Software AG, disponível na sua plataforma de aprendizagem online [6].

2.1.4 API Management Essentials

A formação em "API Management Essentials" incluiu os tópicos mais avançados relativos à gestão de APIs:

- Definir e criar APIs no API Gateway;
- Proteger APIs utilizando políticas e proteção ao nível da *Demilitarized Zone* (DMZ);
- Utilizar o API Gateway como um provedor da API;
- Sincronizar APIs no API Gateway com o Developer Portal;
- Usar o Developer Portal como provedor da API e consumidor da API;
- Gerir a monetização;
- Analisar a utilização de APIs;
- Administrar o API Gateway e o Developer Portal.

Os tópicos abordados nesta formação avançada foram detalhados com base na documentação oficial da Software AG [7].

2.2 Introdução a webMethods

Nesta secção, será apresentada uma visão geral da plataforma webMethods da Software AG, incluindo a sua definição e as tecnologias associadas que são utilizadas para a criação e gestão de APIs. O webMethods é uma plataforma que facilita a integração de sistemas e a gestão de APIs, disponível *on-premises*, na *cloud*, de forma híbrida ou num ambiente *multi-cloud*. A solução oferece funcionalidades para integração *Business-to-Business* (B2B), gestão de transferências de ficheiros e um *gateway* de APIs avançado, que permite administrar, monitorizar e rentabilizar APIs [8].

A criação e gestão de APIs na plataforma webMethods são realizadas através do webMethods Integration Server [9], do Software AG Designer [10] e do JDBC Adapter [11], que serão detalhados nas subsecções seguintes. Estes componentes facilitam o desenvolvimento de APIs, abrangendo todas as etapas, desde a conceção até à implementação.

O webMethods API Gateway [12] permite expor de forma segura as APIs de empresas a programadores terceiros, parceiros e outros consumidores para utilização em aplicações Web, móveis e da *Internet of Things* (IoT). Com o webMethods API Gateway, é possível criar APIs, definir políticas de Acordo de Nível de Serviço e publicar as suas APIs no webMethods Developer Portal. No entanto, neste projeto, o webMethods API Gateway não será utilizado, pois a API a ser desenvolvida será utilizada internamente e não necessita de ser exposta externamente ou monitorizada. O foco será, portanto, na utilização do Integration Server e das ferramentas relacionadas para a criação e gestão interna da API.

2.2.1 webMethods Integration Server

O webMethods Integration Server [9] atua como um servidor de *middleware*, facilitando a integração de qualquer sistema através de padrões abertos. Esta plataforma assegura a interoperabilidade entre diferentes tipos de aplicações, sejam estas personalizadas, pacotes de software ou bases de dados, independentemente de estarem alojadas *on-premises* ou na *cloud*. Dessa forma, garante-se um fluxo contínuo de dados em processos automatizados. Além disso, o webMethods Integration Server inclui funcionalidades integradas para mapeamento e transformação de dados.

O Integration Server oferece um ambiente que permite a execução de serviços de forma organizada. Entre as suas funções estão a descodificação de pedidos dos clientes, identificação dos serviços solicitados, invocação desses serviços, o encaminhamento de dados no formato adequado e codificação dos resultados produzidos, devolvendo-os aos clientes [13]. Com o uso do Integration Server, é possível expor APIs *Representational State Transfer* (REST) ou *Simple Object Access Protocol* (SOAP) para atender a essas necessidades de integração.

2.2.2 Software AG Designer

Após a configuração do webMethods Integration Server, é possível conectá-lo ao Software AG Designer, um *Integrated Development Environment* (IDE) fornecido pela Software AG. O Designer permite criar, desenvolver, testar e fazer o deploy de aplicações e soluções na plataforma webMethods. Com esta ferramenta, é possível criar e manter um amplo

conjunto de aplicações, serviços, tarefas e outros componentes necessários para implementar e executar soluções na gama de produtos webMethods [10].

Através do Software AG Designer, os programadores podem construir e editar serviços, tipos de documentos e outros elementos diretamente no Integration Server. A ligação entre o Designer e o Integration Server é feita através de definições de servidor, que especificam a localização e as características do servidor ao qual o Designer se conecta.

2.2.3 Conectividade com Fontes de Dados: JDBC Adapter

No contexto da plataforma webMethods, uma das tecnologias relevantes é o JDBC [11]. O webMethods Integration Server inclui um JDBC Adapter que permite o acesso universal a dados a partir da linguagem de programação Java, facilitando a integração com diversas fontes de dados, tais como bases de dados relacionais, folhas de cálculo e ficheiros simples (*flat files*). Este adaptador funciona como uma API que específica o protocolo de comunicação entre um cliente e as referidas fontes de dados, estabelecendo a forma como as operações de leitura, escrita e manipulação de dados são realizadas. A configuração e gestão do JDBC Adapter são efetuadas através da interface do Integration Server.

2.3 Aplicações Web Relacionadas

Nesta secção, procede-se à análise de diversas aplicações web que possuem objetivos convergentes com os do presente projeto, particularmente no que diz respeito à criação de plataformas destinadas à partilha de conhecimento no âmbito tecnológico. Serão exploradas as características, funcionalidades e abordagens destas plataformas, de modo a identificar boas práticas e potenciais áreas de inovação que possam ser aplicadas ao desenvolvimento da solução proposta neste trabalho.

2.3.1 Harmonigate

O Website Harmonigate [14] (Figura 2.1) é uma plataforma focada em soluções de integração de sistemas, oferecendo guias e tutoriais técnicos sobre a configuração, instalação e uso de tecnologias como Jenkins, RabbitMQ, BizTalk Server e webMethods. É uma fonte para programadores que procuram aprimorar os seus conhecimentos em ferramentas de integração e automação de processos empresariais. Este recurso pode ser útil em contextos de investigação sobre integração de sistemas e gestão de processos, devido ao seu foco em abordagens práticas e específicas de implementação tecnológica.

Figura 2.1: *Página inicial do Website Harmonigate*

Após uma análise mais aprofundada do Website Harmonigate, verificou-se que utiliza a plataforma WordPress para criar e gerir o seu conteúdo. O WordPress [15] é um CMS de código aberto, desenvolvido em PHP, que utiliza bases de dados MySQL ou MariaDB. Este CMS é amplamente utilizado para criar Websites e *blogs* devido à sua flexibilidade e vasta gama de *plugins* e temas disponíveis.

A análise das funcionalidades e da organização do Website Harmonigate foi fundamental para definir os requisitos que o Website a ser desenvolvido deveria atender. O estudo de como o conteúdo é estruturado e gerido no Harmonigate, apesar de utilizar WordPress, forneceu conhecimentos valiosos sobre as práticas a adotar. No entanto, a solução a desenvolver propõe uma abordagem personalizada, sem o uso de plataformas convencionais, garantindo maior controlo e adaptabilidade às necessidades específicas do projeto.

2.3.2 Pascal Landau

O *blog* de Pascal Landau [16] (Figura 2.2) concentra-se na engenharia de software, fornecendo tutoriais e guias práticos sobre tecnologias como PHP, Docker e BigQuery. Os artigos destacam-se pela sua profundidade técnica e são direcionados a profissionais de desenvolvimento de software interessados em configurar e otimizar aplicações e infraestruturas utilizando ferramentas modernas.

Figura 2.2: *Página inicial do blog de Pascal Landau*

O Website é construído com Jigsaw [17], uma *framework* para a criação de sites estáticos, utilizando ferramentas como Blade [18], uma linguagem de template do Laravel [19]. A *framework* Jigsaw também utiliza Markdown [20], uma linguagem de marcas leve que facilita a criação de textos formatados, podendo ser escrita diretamente num editor de texto comum. Esta abordagem é particularmente útil para a escrita de artigos, publicações e documentação, pois oferece uma estrutura acessível, facilitando a manutenção e a atualização de conteúdos.

A utilização de Markdown revela-se particularmente relevante, sobretudo no contexto do desenvolvimento de um sistema de publicações criado de raiz, como é o caso deste projeto. Sendo uma linguagem de marcas leve, o Markdown simplifica a formatação de texto. Este aspeto será objeto de uma análise mais aprofundada, considerando o seu potencial para otimizar a gestão de conteúdos e a sua adequação em sistemas personalizados que não dependem de plataformas convencionais como o WordPress.

2.3.3 Blog da Software AG

O *blog* da Software AG [21] (Figura 2.3), é dedicado a temas como transformação digital, integração de sistemas, IoT e automação de processos empresariais. Os artigos publicados abordam conteúdos técnicos direcionados a profissionais de tecnologias da informação, programadores e gestores.

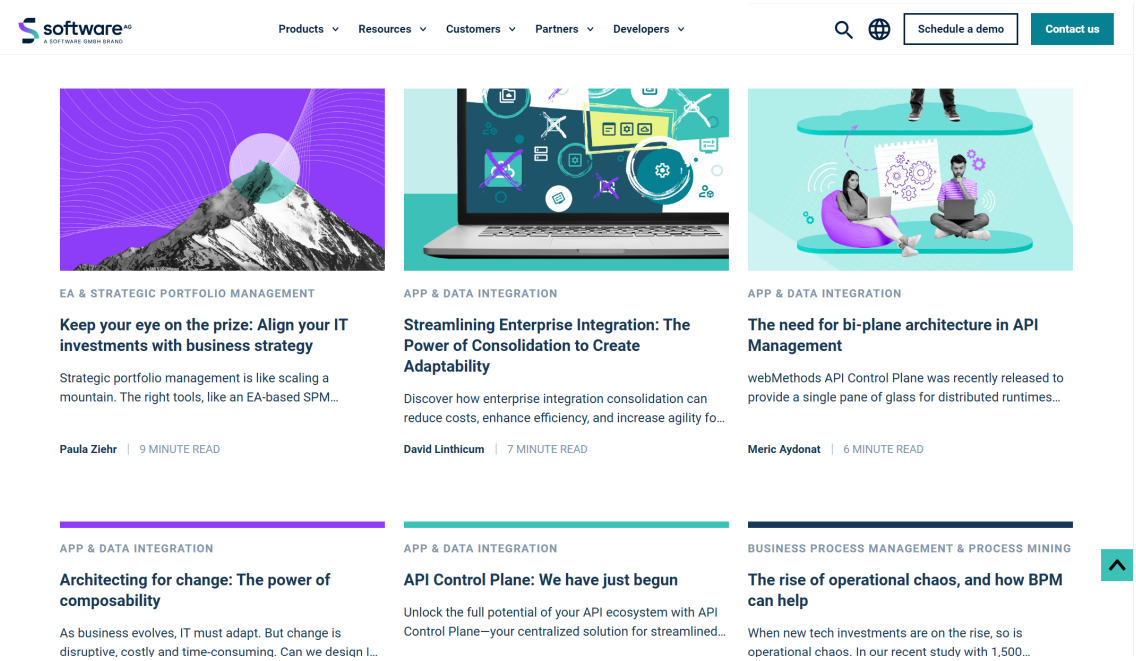

Figura 2.3: *Página inicial do blog da Software AG*

O Website organiza o conteúdo em diversas categorias, como integração de dados, gestão de portfólio estratégico e automação, entre outras. Cada artigo apresenta informações sobre o autor e o tempo estimado de leitura, facilitando a navegação e permitindo ao utilizador selecionar conteúdos de acordo com as suas necessidades específicas e disponibilidade de tempo. A utilização de imagens representativas para cada tema foi um elemento considerado relevante para a aplicação em desenvolvimento, uma vez que contribui para captar o interesse visual dos utilizadores.

A análise deste Website centrou-se nos aspetos de design e na experiência de navegação, dado que não foi possível obter informações sobre a infraestrutura tecnológica ou o método de armazenamento dos conteúdos. A organização dos artigos e o design adotados no *blog* da Software AG ofereceram princípios de apresentação e usabilidade que serviram de orientação para a definição de objetivos no projeto em desenvolvimento.

2.3.4 Medium

O Website Medium [22] (Figura 2.4) é uma plataforma de publicação online que permite a divulgação de artigos sobre temas variados, incluindo tecnologia, empreendedorismo, arte e cultura. Esta plataforma permite a criação de conteúdos por diferentes autores, direcionada a um público interessado em explorar e aprofundar conhecimento sobre uma diversidade de tópicos. Além disso, o Medium oferece aos utilizadores registados a possibilidade de criar e publicar os seus próprios artigos, promovendo a participação ativa.

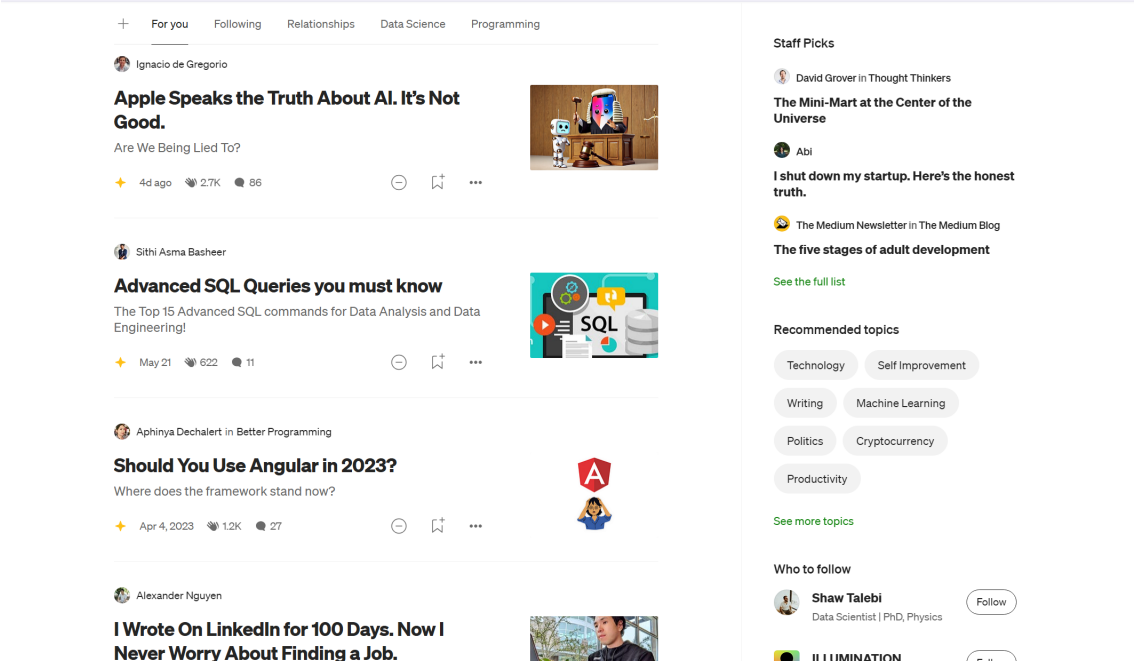

Figura 2.4: *Página inicial do Website Medium*

A análise da plataforma Medium, assim como do blog da Software AG, focou-se na estrutura visual e na experiência de leitura oferecida, uma vez que não foi possível obter informações sobre a infraestrutura tecnológica ou os métodos de armazenamento dos conteúdos.

A funcionalidade de personalização do Medium, que permite aos utilizadores registados escolher temas de interesse e receber conteúdos alinhados com as suas preferências, destaca a importância de adaptar o fluxo informativo às necessidades individuais. No entanto, este não é o foco do projeto em desenvolvimento, uma vez que a aplicação Web abrangerá exclusivamente temas tecnológicos.

A possibilidade de filtrar artigos e documentação por palavras-chave inseridas pelo utilizador constitui outro elemento que melhora a usabilidade, facilitando o acesso rápido a conteúdos específicos. Este recurso será implementado neste projeto, uma vez que simplifica a pesquisa e torna o acesso mais eficiente. Desta forma, a análise do Medium forneceu um conjunto de práticas essenciais para otimizar a estrutura visual e a acessibilidade dos conteúdos na aplicação Web a desenvolver.

Capítulo 3

Análise e Tecnologias

Este capítulo apresenta uma análise aprofundada das funcionalidades que a aplicação web deverá ter, juntamente com um estudo detalhado das tecnologias que podem ser utilizadas na sua construção. Primeiramente, são discutidos os requisitos do projeto (Secção 3.1) e os casos de utilização (Secção 3.2), que permitem compreender as necessidades do sistema e as interações esperadas entre os utilizadores e a plataforma. Em seguida, será abordado o design da solução (Secção 3.3) e as possíveis tecnologias para construir a solução final (Secção 3.4), possibilitando a escolha das ferramentas tecnológicas mais adequadas para o desenvolvimento do projeto.

3.1 Requisitos

Os requisitos de uma aplicação são as especificações que definem o que o software deve realizar e as condições que deve cumprir para ser considerado adequado aos seus objetivos. Estes podem ser divididos em requisitos funcionais, apresentados na subsecção 3.1.1 e requisitos não funcionais, apresentados na subsecção 3.1.2.

3.1.1 Requisitos Funcionais

Os requisitos funcionais definem as operações e comportamentos que o sistema deve executar para satisfazer as necessidades dos utilizadores e os objetivos do projeto, especificando de forma detalhada as funcionalidades e serviços a fornecer. Estes requisitos abrangem todas as interações esperadas entre os utilizadores e o sistema. Desta forma, os requisitos funcionais são fundamentais para orientar o desenvolvimento, garantindo que o sistema se comporte adequadamente e cumpra os seus propósitos. Os principais requisitos funcionais identificados para este projeto encontram-se apresentados na Tabela 3.1.

Tabela 3.1: Requisitos funcionais

Código	**Descrição**
R1	Autenticação de Utilizadores Administradores
R2	Criação de Publicações
R3	Pré-visualização de Publicações
R4	Visualização de Conteúdo de Publicações
R5	Edição de Publicação
R6	Remoção de Publicação
R7	Filtragem de Publicações Criadas pelo Administrador
R8	Filtragem de Publicações por Tipo
R9	Filtragem de Publicações por Título
R10	Contacto com a Empresa

Pretende-se que o presente projeto resulte numa aplicação Web onde seja possível, por exemplo, a autenticação de utilizadores administradores (R1), a criação de publicações (R2), a pré-visualização (R3) e visualização de conteúdos de publicações (R4), bem como funcionalidades de edição (R5) e remoção de publicações (R6). Além disso, o sistema deve permitir a filtragem de publicações criadas pelo administrador (R7), por tipo (R8) e por título (R9), e oferecer funcionalidades para contacto com a empresa (R10).

3.1.2 Requisitos Não Funcionais

Os requisitos não funcionais asseguram que o sistema opere de forma eficiente e cumpra padrões de qualidade específicos. Ao contrário dos requisitos funcionais, que se focam nas funcionalidades específicas do sistema, os requisitos não funcionais estabelecem as condições pelas as quais o sistema deve operar, garantindo que satisfaça as expectativas e necessidades dos utilizadores de forma abrangente. Os requisitos não funcionais identificados para este sistema estão apresentados na Tabela 3.2.

Tabela 3.2: Requisitos não funcionais

Código	**Descrição**
RNF1	O sistema deve garantir que apenas utilizadores autenticados possam aceder às funcionalidades relacionadas com gestão de publicações.
RNF2	A interface deve ser intuitiva e fácil de navegar.
RNF3	As operações de pesquisa e carregamento de conteúdo devem ser realizadas de forma rápida.
RNF4	A aplicação deve ser capaz de escalar de forma eficiente à medida que o volume de dados e o número de utilizadores aumenta.
RNF5	O sistema deve ser fácil de atualizar e manter, facilitando a correção de erros e a introdução de novas funcionalidades.

3.2 Casos de Utilização

Os casos de utilização detalham as interações entre os utilizadores e o sistema, tendo em vista a obtenção de um resultado específico. Cada caso de utilização específica, de forma clara, como um ator interage com o sistema para atingir um determinado objetivo. Os atores podem ser indivíduos, grupos, entidades ou até mesmo outros sistemas externos que precisam de comunicar com o sistema em questão.

Num diagrama de casos de utilização, estas interações são representadas dentro de uma caixa que simboliza o sistema. Esta representação gráfica permite delinear os limites do sistema, destacando as suas funcionalidades e as interações que ocorrem no seu interior. No diagrama da Figura 3.1, são apresentados os casos de utilização do projeto a desenvolver.

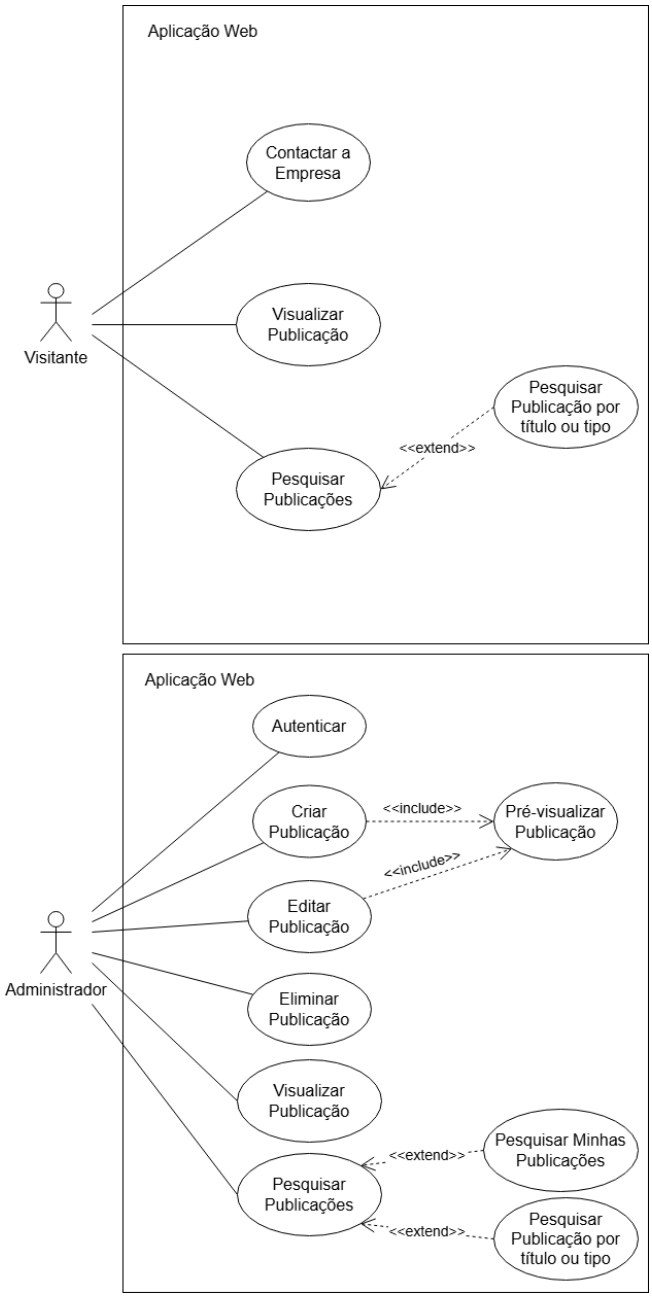

Figura 3.1: *Casos de utilização*

Conforme ilustrado na Figura 3.1, o diagrama de casos de utilização do projeto apresenta dois atores principais: o **Visitante** e o **Administrador**. Estes atores interagem com o sistema através de vários casos de utilização, que representam as diferentes funcionalidades disponíveis para cada tipo de utilizador.

Existem dois casos de utilização principais que ambos os atores podem aceder: **Visualizar Publicação** e **Pesquisar Publicações**. Estas funcionalidades permitem que tanto visitantes como administradores possam visualizar o conteúdo disponível e realizar pesquisas por publicações.

O visitante pode também contactar a empresa através do sistema, uma funcionalidade que possibilita a comunicação para obter informações ou esclarecimentos adicionais sobre o conteúdo disponível no Website.

Adicionalmente, o administrador tem acesso a casos de utilização específicos que refletem as suas permissões adicionais para gerir conteúdo. Estes casos de utilização incluem **Autenticar**, **Criar Publicação**, **Editar Publicação**, **Eliminar Publicação**, e **Pré-visualizar Publicação**. Estas ações permitem que o administrador tenha um controlo completo sobre o ciclo de vida das publicações no sistema. É de notar que o caso de utilização **Pré-visualizar Publicação** está relacionado com a criação de uma publicação e, por isso, é representado com uma relação de *«include»* com **Criar Publicação**, indicando que esta ação é parte integrante do processo de criação.

Relativamente às funcionalidades de pesquisa, o diagrama mostra uma extensão do caso de utilização **Pesquisar Publicações**, que inclui as opções **Pesquisar Publicação por Título ou Tipo** e **Pesquisar Minhas Publicações**. Estas extensões representam funcionalidades adicionais que permitem uma pesquisa mais específica de publicações. A pesquisa por título ou tipo está disponível tanto para visitantes como para administradores, enquanto a pesquisa de publicações específicas de um administrador é uma funcionalidade exclusiva deste utilizador, refletindo um nível de acesso mais refinado e direcionado.

3.3 Design

O desenvolvimento da aplicação Web iniciou-se com a criação de *mockups*, que são representações visuais estáticas do design e da estrutura das páginas da plataforma. Estes *mockups* foram utilizados para definir o aspeto geral pretendido, incluindo a disposição dos elementos e a *User Interface* (UI). Cada *mockup* representa uma das páginas que o utilizador pode aceder durante a sua navegação, alinhando-se com as funcionalidades previamente definidas.

Durante o processo de design e implementação, os *mockups* foram periodicamente revistos em várias etapas para assegurar que o visual da aplicação permanecesse alinhado com os objetivos e requisitos previamente estabelecidos. Este processo de revisão contínua permitiu ajustar e aperfeiçoar o design com base nos comentários recebidos pela empresa, garantindo que o resultado final fosse coerente, tanto a nível visual como funcional, e que estivesse em conformidade com as necessidades específicas do projeto. É importante referir que o

design e as cores utilizadas foram inspirados no Website corporativo da Findmore Digital, refletindo a sua identidade visual e seguindo as suas diretrizes de estilo (disponível em: https://www.findmoredigital.com/).

Importa referir que este design foi concebido com o objetivo de proporcionar simplicidade e eficiência na utilização da aplicação, assegurando que os resultados das ações dos utilizadores sejam facilmente compreensíveis. Para alcançar este objetivo, o processo de design foi guiado pelas 10 heurísticas de Nielsen [23], que oferecem diretrizes claras e analíticas para o desenvolvimento eficaz de interfaces.

Foram elaborados *mockups* distintos, tanto para a navegação destinada aos visitantes, como para os administradores, os quais serão detalhados nas subsecções seguintes, sendo estas a navegação para visitantes (subsecção 3.3.1) e a navegação para administrador (subsecção 3.3.2).

3.3.1 Navegação para Visitante

Nesta subsecção, são apresentados os *mockups* relativos aos utilizadores visitantes do Website, Figura 3.2. Nestes podemos observar a página inicial do Website onde são apresentadas as diferentes publicações e a página relativa a uma publicação.

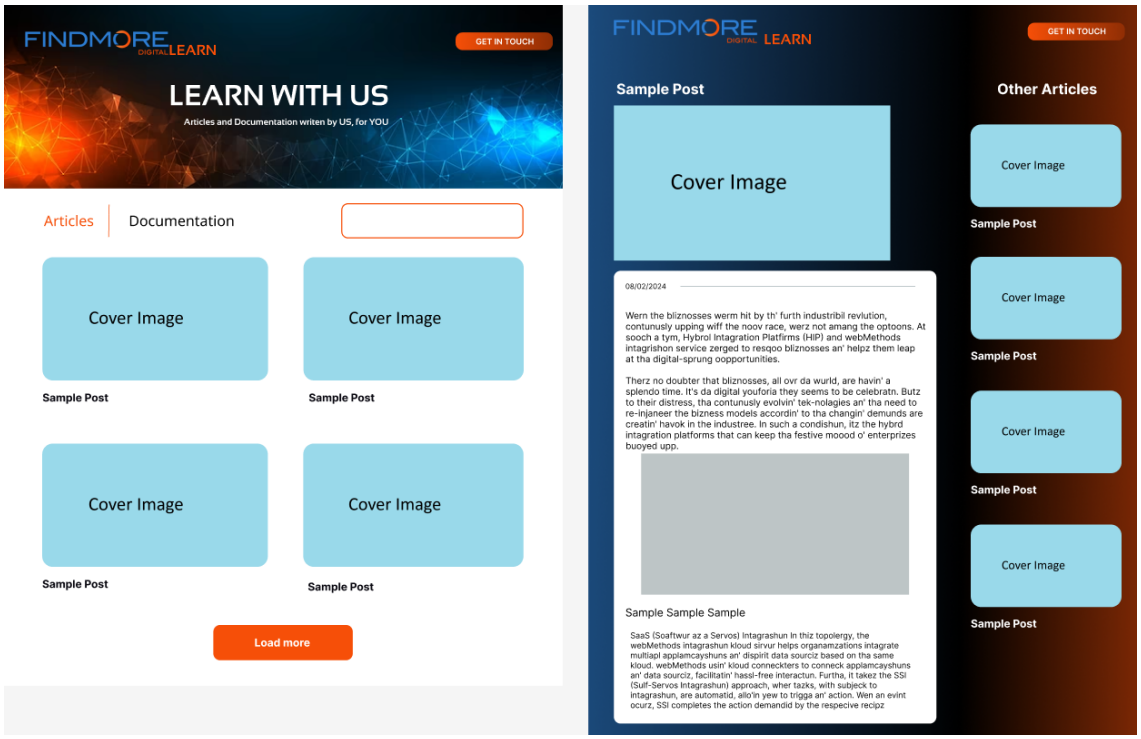

Figura 3.2: *Mockups de navegação para visitante*

3.3.2 Navegação para Administrador

Nesta subsecção, são apresentados os *mockups* relativos aos utilizadores administradores, focando-se nas funcionalidades avançadas de gestão e manutenção do conteúdo do Website, Figura 3.3.

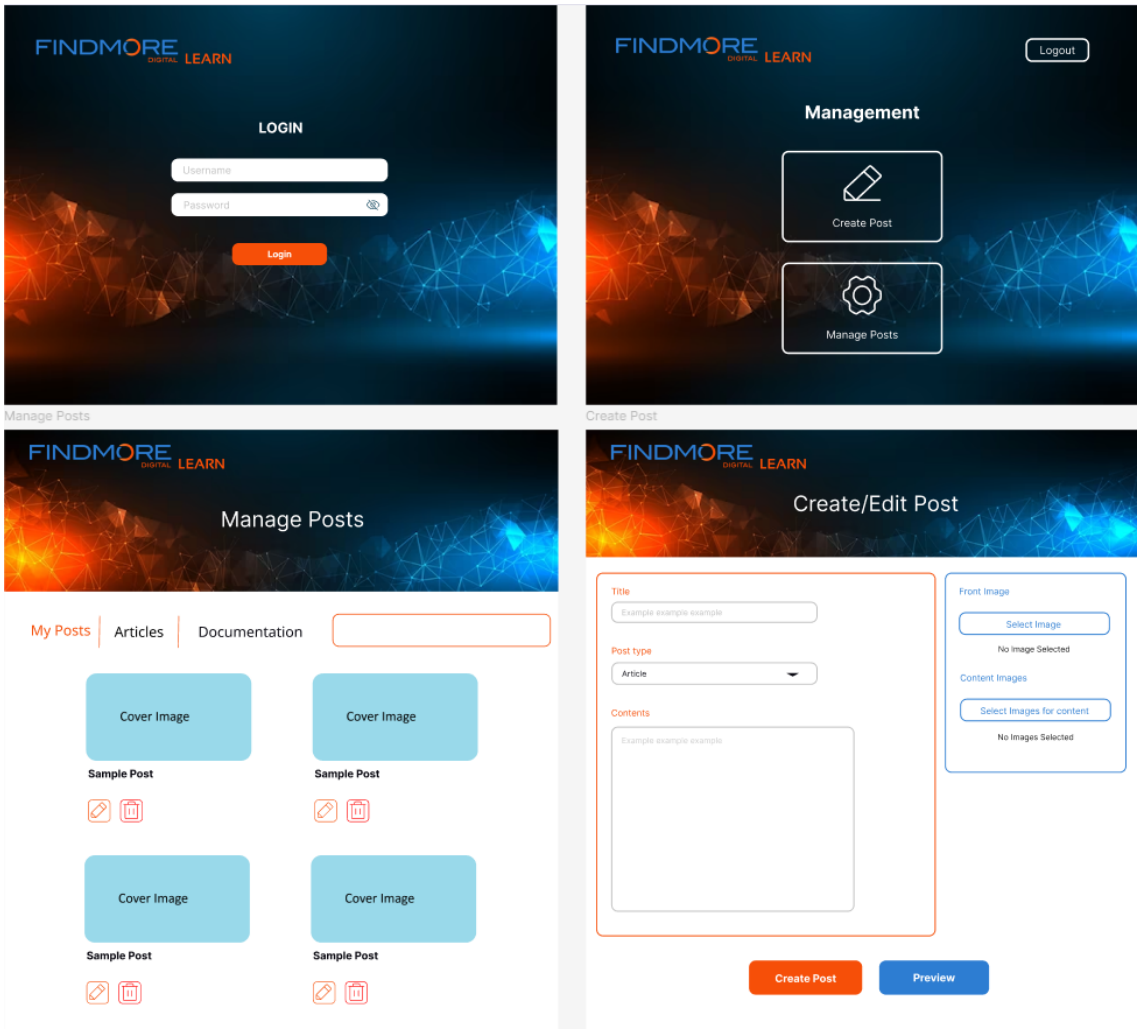

Figura 3.3: *Mockups de navegação para administrador*

3.4 Tecnologias

Nesta secção, será realizada uma análise comparativa entre diferentes tecnologias utilizadas no desenvolvimento de aplicações Web. Uma aplicação Web geralmente adota a arquitetura Cliente-Servidor, que é um padrão amplamente utilizado no desenvolvimento de software. Nesta arquitetura, o sistema é dividido em duas partes principais: o *frontend* (cliente) e o *backend* (servidor), cada um desempenhando funções distintas, mas complementares.

3.4.1 *Frontend*

O *frontend* é a camada da aplicação com a qual o utilizador interage diretamente. É responsável pela interface gráfica, proporcionando a experiência visual e de navegação para o utilizador final. Desenvolvido com tecnologias como HTML, CSS, JavaScript e *frameworks* modernas, como Angular, React ou Vue.js, o *frontend* trata da apresentação dos dados e da recolha de *inputs* do utilizador, que são posteriormente enviados para o *backend* para processamento.

De seguida é apresentada uma análise comparativa entre as *frameworks* Angular, React e Vue.js, com o objetivo de determinar qual delas é mais adequada para o desenvolvimento da aplicação em questão. Esta análise considerará vários critérios, como desempenho, facilidade de utilização, curva de aprendizagem, flexibilidade, suporte da comunidade e manutenção, entre outros fatores relevantes.

Angular

O Angular [24] é uma plataforma de desenvolvimento de código aberto, gerida pela Google, construída com TypeScript [25] e organizada numa arquitetura baseada em componentes. A primeira vantagem na utilização do Angular é o uso de TypeScript, uma linguagem que permite a verificação de tipos e o uso de interfaces, facilita a deteção de erros e a manutenção do código, tornando-o mais robusto e fácil de gerir. Além disso, o sistema de Injeção de Dependências do Angular permite separar os componentes das suas dependências, o que melhora a gestão e reutilização do código. Outro ponto positivo é a documentação detalhada e a grande comunidade ativa, que oferecem muitos recursos, ferramentas e suporte. Estes recursos facilitam a aprendizagem, a resolução de problemas e disponibilizam várias bibliotecas integradas para diferentes funcionalidades.

Não obstante, o Angular apresenta algumas desvantagens. Uma delas é a sua curva de aprendizagem acentuada, devido à sua arquitetura com conceitos mais complexos. Outra desvantagem está relacionada ao desempenho: por ser uma *framework* robusta, o Angular pode tornar aplicações de grande escala mais lentas, o que afeta a experiência do utilizador.

React

O React [26] é uma biblioteca de JavaScript de código aberto, desenvolvida e mantida pelo Facebook, que se concentra na construção de interfaces de utilizador. Uma das principais vantagens do React é a sua curva de aprendizagem menos acentuada, facilitada pela simplicidade da sua API e pela forte integração com o JavaScript. Outro ponto positivo do React é a sua vasta comunidade e o ecossistema diversificado de ferramentas e bibliotecas associadas, que permitem aos programadores personalizar as suas aplicações de acordo com as necessidades específicas do projeto, sem se limitar a um conjunto rígido de regras.

Contudo, o React apresenta também algumas desvantagens. Sendo uma biblioteca e não uma *framework* completa, o React requer a integração de ferramentas adicionais para funcionalidades como a gestão de rotas, o que pode aumentar a complexidade do projeto. A rápida evolução do ecossistema do React pode também ser um desafio, exigindo uma atualização constante de conhecimentos e dependências.

Vue.js

O Vue.js [27] é uma *framework* JavaScript de código aberto conhecido pela sua simplicidade e facilidade de aprendizagem. Baseia-se em HTML, CSS e JavaScript convencionais, fornecendo uma API fácil de usar e uma documentação de alta qualidade. A sua arquitetura flexível permite que seja utilizado como uma biblioteca ou como uma *framework* completa, ajustando-se às necessidades de cada projeto. Além disso, o Vue oferece um desempenho

elevado através de um sistema de renderização reativo e otimizado, que reduz a necessidade de otimizações manuais e facilita a criação de interfaces dinâmicas e responsivas.

No entanto, o Vue também apresenta algumas limitações. A sua comunidade, embora em crescimento, é menor quando comparada à do Angular e do React, o que pode resultar em menos recursos e suporte em determinados contextos. Além disso, algumas bibliotecas e ferramentas de terceiros podem não oferecer o mesmo nível de integração ou suporte disponível em *frameworks* mais amplamente adotadas.

Tecnologia adotada

Entre as tecnologias mencionadas, o Angular foi selecionado pela sua robustez e por oferecer uma solução completa que abrange todas as necessidades. O Angular já inclui funcionalidades essenciais como gestão de rotas, sem a necessidade de bibliotecas externas. A sua arquitetura baseada em componentes e o sistema de injeção de dependências tornam o código mais modular e fácil de manter. Embora o React e o Vue também sejam opções poderosas, exigem a adição de bibliotecas para alcançar a mesma funcionalidade que o Angular oferece de forma nativa. Além disso, o suporte da Google e uma comunidade ativa asseguram boa documentação e atualizações constantes, tornando o Angular uma escolha confiável para aplicações escaláveis.

3.4.2 *Backend*

O *backend* é a parte do sistema que opera no servidor e é responsável pelo processamento de dados, pela lógica de negócio e pela comunicação com a base de dados. Esta camada do sistema recebe solicitações provenientes do *frontend*, processa-as e executa operações sobre os dados, como consultas e atualizações na base de dados, devolvendo as respostas apropriadas ao *frontend*.

No *backend*, a base de dados é um componente essencial, responsável pelo armazenamento, organização e gestão dos dados da aplicação. Será realizada uma comparação entre três bases de dados relacionais baseadas em *Structured Query Language* (SQL): MySQL, Microsoft SQL Server e PostgreSQL. A escolha de uma base de dados SQL foi relativa à necessidade de uma estrutura relacional bem definida, que facilite a gestão de dados, como publicações e utilizadores, na aplicação web a desenvolver. Adicionalmente, a escalabilidade vertical proporcionada pelas bases de dados SQL é adequada para o volume de dados previsto neste projeto, assegurando um desempenho sólido e consistente.

Além disso, será feita uma comparação entre a tecnologia webMethods e outras alternativas no mercado, como MuleSoft, embora a escolha já esteja definida por ser a tecnologia utilizada pela empresa. Dessa forma, será possível entender a posição do webMethods no mercado, em relação a outras soluções.

Por fim, considerando a necessidade de implementar uma API, será realizada uma comparação detalhada entre os protocolos REST e SOAP. Essa análise irá explorar as principais diferenças entre essas duas abordagens.

Bases de dados

MySQL

O MySQL [28] é uma base de dados de código aberto e destaca-se pela facilidade de instalação e utilização, aliando fiabilidade e escalabilidade para gerir grandes volumes de dados e um elevado número de utilizadores. A sua popularidade também se deve à vasta quantidade de recursos de aprendizagem disponíveis e a uma comunidade global ativa e colaborativa.

Não obstante, o MySQL apresenta algumas limitações, como a falta de compatibilidade com funcionalidades avançadas de SQL e a ausência de suporte a operações de processamento de dados mais complexas, o que pode limitar a sua flexibilidade para certas aplicações.

Microsoft SQL Server

O Microsoft SQL Server [29] destaca-se pelas suas ferramentas robustas que simplificam o design, desenvolvimento e manutenção de bases de dados, sendo uma escolha sólida para a gestão de dados no contexto empresarial. Além disso, o SQL Server oferece funcionalidades avançadas de segurança e recuperação de dados, que ajudam a prevenir perdas devido a falhas de energia ou problemas no servidor, assegurando uma gestão eficiente de armazenamento e recuperação.

Embora tenha várias vantagens, o Microsoft SQL Server apresenta algumas limitações, como os custos elevados, especialmente quando se requerem funcionalidades avançadas. Além disso, o SQL Server é otimizado para infraestruturas Microsoft, o que pode implicar investimentos adicionais em software e hardware em ambientes que utilizam outros sistemas.

PostreSQL

O PostgreSQL [30] é uma base de dados de código aberto que oferece suporte a transações, garantindo integridade e consistência de dados em aplicações complexas. Outra vantagem do PostgreSQL é a sua extensibilidade, que permite aos utilizadores adicionar novos tipos de dados, funções, operadores, e até mesmo linguagens de programação personalizadas. Por exemplo, é possível criar funções definidas pelo utilizador para operações personalizadas, ou adicionar linguagens como Python, Perl, entre outros, para escrita de procedimentos.

Por outro lado, como software de código aberto, o PostgreSQL não é controlado por uma única entidade, o que pode levar a uma visibilidade reduzida e a interfaces menos intuitiva.

Tecnologia adotada

Uma vez que o projeto foi desenvolvido no contexto de um estágio, as desvantagens do Microsoft SQL Server não se aplicam, dado que a empresa já utiliza este sistema e dispõe de uma infraestrutura totalmente compatível. O Microsoft SQL Server é amplamente utilizado pela empresa, especialmente em conjunto com o webMethods, e a sua simplicidade de utilização, facilidade de implementação e interface amigável foram fatores determinantes na tomada de decisão por esta solução.

Middleware

Webmethods

A Software AG [2], proprietária dos webMethods, é reconhecida pela sua abordagem inovadora, voltada para a transformação digital, integração de APIs, arquiteturas orientadas a eventos e diversas estratégias de integração. O webMethods é vantajoso pois permite a integração eficiente de sistemas tanto *on-premises* quanto na *cloud*. A plataforma destaca-se também pelas ferramentas robustas para testes e *debugging*.

Embora ofereça diversos benefícios, o webMethods apresenta uma certa complexidade na sua configuração e utilização, especialmente para utilizadores sem experiência prévia em plataformas de integração, o que resulta numa curva de aprendizagem acentuada. Esta situação exige uma formação específica e conhecimentos técnicos aprofundados para que a plataforma seja utilizada de forma eficiente.

MuleSoft

O MuleSoft Anypoint [31] é uma plataforma de integração que permite conectar dados, aplicações e dispositivos em diferentes ambientes, possibilitando que os profissionais de software criem, implementem e giram APIs e integrações. Com uma abordagem orientada por APIs, a plataforma facilita a integração de sistemas baseados na *cloud* e oferece ferramentas robustas para o design de APIs. O MuleSoft destaca-se pela sua vasta gama de conectores para sistemas como Salesforce e AWS, permitindo uma integração eficiente de sistemas e proporcionando uma visão em tempo real do desempenho das aplicações, ajudando as empresas a identificar e resolver problemas rapidamente.

No entanto, o MuleSoft apresenta algumas desvantagens, como a sua curva de aprendizagem acentuada, que pode exigir formação adicional para utilizadores sem experiência em integração de sistemas e desenvolvimento de APIs. Além disso, a utilização do MuleSoft Composer sem integração total com a Anypoint Platform pode causar problemas de conectividade, limitando a eficácia da integração entre sistemas. A complexidade da plataforma também pode exigir uma gestão contínua e monitorização rigorosa.

Tecnologia adotada

Ao efetuar a comparação entre ambos, o webMethods é considerado uma opção superior ao MuleSoft devido à sua maior flexibilidade para integrar ambientes híbridos (*on-premises* e *cloud*) e funcionalidades de gestão robustas. Estas características fazem do webMethods uma plataforma mais acessível e eficiente para organizações que procuram uma solução de integração poderosa e fácil de gerir, especialmente em cenários empresariais complexos.

Tipos de API

REST vs SOAP

O REST é um estilo arquitetónico amplamente utilizado para criar interfaces de comunicação que usam métodos *HyperText Transfer Protocol* (HTTP), como `GET`, `POST`, `PUT` e `DELETE`, facilitando interações simples entre clientes e servidores. Este estilo é orientado a recursos, que são identificados por *Uniform Resource Locator*s (URLs), e adota uma representação de dados leve, geralmente em formatos como *JavaScript Object Notation* (JSON) ou XML. A arquitetura REST é sem estado, o que significa que cada solicitação do cliente é tratada de forma independente, sem dependência das solicitações anteriores, o que contribui para a escalabilidade e flexibilidade do sistema [32].

Por outro lado, o SOAP é um protocolo formal e estruturado, projetado para a troca de informações em ambientes distribuídos e descentralizados. As mensagens SOAP são transmitidas em XML e incluem cabeçalhos detalhados para definir regras de segurança, autenticação e integridade, tornando-o mais robusto. No entanto, SOAP é mais rígido na sua abordagem, suportando apenas mensagens XML e exigindo que o servidor mantenha o estado de cada solicitação, o que pode aumentar o consumo de recursos e limitar a escalabilidade [32].

O SOAP tende a produzir mensagens mais complexas e pesadas, o que pode levar a um processamento mais lento e a tempos de resposta mais longos. Além disso, como o SOAP requer a manutenção do estado entre as solicitações e uso maior de recursos, tornando a escalabilidade mais desafiante. Em contrapartida, o REST é mais rápido e eficiente, utilizando mensagens mais leves e permitindo o armazenamento de respostas em cache, o que melhora o desempenho. No entanto, o REST pode ser menos confiável na gestão de falhas de comunicação, enquanto SOAP oferece um sistema de tratamento de erros mais robusto.

Em suma, a escolha entre REST e SOAP deve ser realizada com base nos requisitos específicos de cada projeto. Neste caso, a escolha recaiu sobre REST pelo suporte de formato JSON e, conforme os pontos discutidos anteriormente, por ser escalável, flexível e ter um melhor desempenho.

Capítulo 4

Implementação do Projeto

Este capítulo descreve a implementação do modelo proposto, detalhando a sua arquitetura e as soluções adotadas no seu desenvolvimento. Inicialmente, aborda-se a Arquitetura do Sistema (Secção 4.1), destacando a sua estrutura modular e a interação entre os seus componentes.

Seguidamente, discutem-se as Considerações e Soluções implementadas (Secção 4.2), evidenciando as decisões técnicas e metodológicas tomadas ao longo do processo. Este capítulo explora ainda a estrutura e criação da Base de Dados (Secção 4.3), com uma descrição do Modelo Entidade-Associação e do Modelo Relacional adotados.

Além disso, é detalhada a integração em webMethods (Secção 4.4), incluindo a exposição de serviços através da API REST. Finalmente, discute-se o desenvolvimento da aplicação Web (Secção 4.5), com foco na interface utilizador e na sua interação com os serviços de *backend*.

4.1 Arquitetura do Sistema

Com base na definição das tecnologias estabelecidas no Capítulo 3, foi desenvolvida a arquitetura apresentada na Figura 4.1, seguindo um modelo de sistema em camadas.

Esta arquitetura é composta pelos seguintes níveis: Utilizadores, *Frontend* e *Backend* (incluindo a camada de base de dados). De seguida, apresentam-se os componentes de cada uma das camadas.

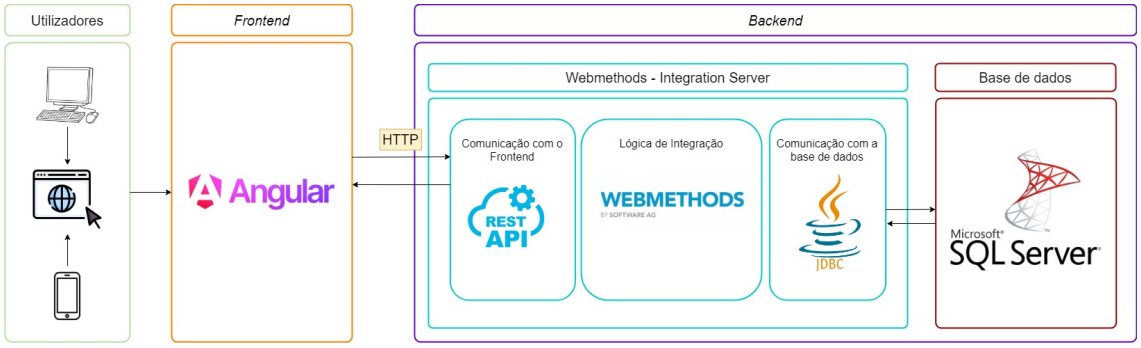

Figura 4.1: *Arquitetura do sistema*

A camada de utilizadores representa os utilizadores finais que interagem diretamente com o sistema. O acesso à aplicação é feito através de navegadores Web, que podem ser utilizados em diversos dispositivos, como computadores, telemóveis ou tablets. Esta camada permite a interação com a UI e o envio de comandos para a aplicação.

A camada de *frontend* é implementada utilizando o Angular, um *framework* amplamente utilizado no desenvolvimento de aplicações Web. Esta camada é responsável pela renderização da interface de utilizador e pela gestão das interações do mesmo com a aplicação web. A partir da interface, o *frontend* emite pedidos HTTP ao *backend* para a obtenção de dados e processa as respostas recebidas, que são então apresentadas ao utilizador.

A camada de *backend* é constituída por duas subcamadas. A primeira é suportada pelo WebMethods Integration Server, que oferece funcionalidades avançadas de orquestração e integração, atuando como intermediário entre o *frontend* e a base de dados. Esta subcamada inclui três componentes principais.

O primeiro componente é a REST API, que processa os pedidos HTTP recebidos do *frontend* e facilita a comunicação, assegurando a troca de dados.

O segundo componente é a lógica de integração, responsável pela execução da lógica de negócio, pela orquestração de processos e pela integração com diversos serviços e sistemas, utilizando as capacidades avançadas do webMethods.

O terceiro componente é a interação com a base de dados, que utiliza JDBC para comunicar com o sistema de gestão de bases de dados. Esta interação permite ao servidor de integração executar consultas SQL, gerir transações e manipular os dados armazenados no Microsoft SQL Server.

Por fim, na camada de *backend*, encontra-se a subcamada de base de dados, gerida pelo Microsoft SQL Server, que é responsável por armazenar todos os dados da aplicação. Esta subcamada é essencial para o armazenamento e recuperação de dados, suportando as operações `Create, Read, Update, Delete (CRUD)` necessárias para o funcionamento da aplicação.

4.2 Considerações e Soluções

Durante o desenvolvimento deste projeto, surgiram diversos desafios que exigiram uma análise aprofundada e a implementação de soluções para assegurar uma aplicação eficiente. Alguns dos principais problemas encontrados centraram-se na necessidade de simplificar processos complexos para os utilizadores finais, melhorar a usabilidade do sistema e garantir que este atendesse aos requisitos funcionais e técnicos estabelecidos. Esta secção apresenta os desafios enfrentados e as estratégias adotadas para os resolver. Algumas destas decisões influenciaram o redesenho dos *mockups* definidos no Capítulo 3, Secção 3.3.

4.2.1 Publicações e Conversão para HTML

Um dos problemas críticos identificados foi a necessidade de permitir que os administradores do sistema criassem publicações de forma simples, sem a necessidade de escrever código

HTML manualmente. Para resolver este problema, foi necessário desenvolver uma solução que facilitasse a criação de conteúdo e garantisse que o resultado fosse bem formatado e consistente com o design da aplicação. A solução envolveu a utilização de uma abordagem baseada em Markdown que permite aos utilizadores escrever texto de forma simples e é posteriormente convertido automaticamente para HTML estruturado.

A conversão de Markdown para HTML foi realizada através da integração de bibliotecas específicas que automatizam este processo. Esta abordagem permitiu manter a flexibilidade e a personalização das publicações, sem exigir conhecimentos avançados de programação por parte dos utilizadores.

4.2.2 Gestão de Imagens

Dado que as publicações poderão incluir pelo menos uma imagem, a gestão de imagens no sistema apresentou um desafio significativo, envolvendo questões relacionadas com o armazenamento, o *upload* e a apresentação no *frontend*. O sistema deve permitir o *upload* de imagens e garantir uma gestão eficiente, incluindo a opção de eliminar imagens desnecessárias sem repetir o processo de *upload*.

Tendo em conta esses desafios, realizou-se uma análise comparativa das alternativas de armazenamento de imagens. Na escolha da solução mais adequada para uma aplicação, é crucial avaliar fatores como escalabilidade, fiabilidade, custos e impacto no desempenho. Três opções comuns são o sistema de ficheiros, o armazenamento em *cloud* e na base de dados, cada uma com as suas vantagens e desvantagens, que serão analisadas de seguida.

Sistema de Ficheiros

O armazenamento de imagens em sistemas de ficheiros locais envolve guardar os ficheiros diretamente no servidor que hospeda a aplicação. Esta abordagem é simples e direta, uma vez que não requer serviços externos ou configurações complexas. Contudo, apresenta algumas limitações significativas. Por exemplo, em servidores locais, o espaço de armazenamento pode ser limitado e as imagens ficam vulneráveis a falhas do servidor, como quebras de *hardware* ou corrupção do sistema de ficheiros.

Armazenamento em *Cloud*

Os serviços de armazenamento em *cloud*, como o Amazon S3 ou o Google Cloud Storage, oferecem soluções escaláveis e fiáveis para o armazenamento de imagens. Com este armazenamento, a gestão e manutenção de ficheiros de imagem são transferidas para fornecedores de confiança, garantindo disponibilidade alta e durabilidade. Contudo, esta opção pode ser mais dispendiosa em comparação com o armazenamento no sistema de ficheiros do servidor ou na base de dados, devido à necessidade de integração com serviços de terceiros, além de implicar uma curva de aprendizagem para a sua configuração.

Armazenamento na Base de Dados

O armazenamento direto de imagens na base de dados consiste em guardar os ficheiros de imagem, utilizando tipos de dados como *Binary Large Object* (BLOB) ou string em `base64`

em bases de dados relacionais, como PostgreSQL ou MySQL. No caso do SQL Server, uma alternativa é o uso de `VARCHAR(MAX)`, que permite armazenar grandes quantidades de dados textuais, incluindo imagens convertidas para um formato de `base64`. Esta opção é eficiente para aplicações onde o número de imagens a armazenar é relativamente pequeno, eliminando a necessidade de soluções de armazenamento externas ou complexas. Além disso, o uso de `VARCHAR(MAX)` pode simplificar a manipulação de dados e facilitar o desenvolvimento, pois evita configurações adicionais para o tratamento de imagens. No entanto, armazenar imagens na base de dados pode aumentar significativamente o seu tamanho e afetar o desempenho, especialmente em sistemas que necessitem de gerir um grande volume de imagens.

Solução adotada

Como o volume de imagens a armazenar será reduzido, a solução escolhida para este projeto é armazená-las diretamente na base de dados. Esta abordagem simplifica a gestão e manutenção das imagens e do sistema, eliminando a necessidade de configurar serviços externos ou lidar com as limitações do sistema de ficheiros do servidor. Ao utilizar a base de dados relacional Microsoft SQL Server como meio de armazenamento, é possível guardar as imagens de forma segura, juntamente com os metadados relevantes, permitindo uma gestão centralizada das imagens.

Esta solução também garante a integridade e acessibilidade das imagens, mantendo a simplicidade da infraestrutura da aplicação. Embora o armazenamento de imagens na base de dados possa aumentar o seu tamanho, o impacto no desempenho será mínimo devido ao volume moderado de imagens esperado. Assim, considerando os requisitos da aplicação e o número reduzido de imagens a gerir, esta abordagem revela-se prática e eficiente para o projeto.

4.2.3 Contacto com a Empresa

No sistema desenvolvido, de acordo com os casos de utilização definidos, o utilizador pode contactar a empresa através de um formulário, onde insere o seu endereço de *e-mail*, o assunto e a mensagem. Após o envio, o formulário é processado e os dados são encaminhados para um endereço de *e-mail* interno da empresa.

O principal desafio deste processo é definir a abordagem mais adequada para o reencaminhamento dos *e-mails*. Dado que o Website corporativo da empresa já dispõe de um formulário semelhante, mas não foi disponibilizada informação detalhada sobre a sua implementação, foi efetuada uma análise das alternativas possíveis para o reencaminhamento de *e-mail*.

Utilização de APIs externas

Alguns serviços, como SendGrid [33], Mailgun [34] ou Amazon SES [35], disponibilizam APIs que permitem a gestão programática de *e-mail*, incluindo o reencaminhamento automático para endereços específicos. Embora ofereçam um elevado controlo sobre o fluxo de *e-mail* e facilitem a integração com outras aplicações, estes serviços são pagos, e os custos variam consoante o volume de *e-mail* enviados e as funcionalidades adicionais utilizadas.

Criação de uma API

Outra opção é o desenvolvimento de uma API personalizada para gerir o reencaminhamento de *e-mail*, oferecendo maior flexibilidade e adaptação às necessidades específicas da empresa. Esta API pode ser projetada para validar, processar e encaminhar os *e-mail* recebidos de forma segura, garantindo maior controlo sobre o fluxo e a segurança dos dados.

Solução adotada

O caso de utilização "Contactar a Empresa", foi considerado como trabalho futuro, devido à necessidade de conhecimentos mais especializados em segurança e proteção de dados. No contexto empresarial deste projeto, é crucial garantir que informações sensíveis, como credenciais de *e-mail*, não sejam armazenadas diretamente no código-fonte. Para assegurar a segurança e conformidade, podem ser utilizados serviços de gestão de segredos [36], que protegem informações sensíveis, como chaves e senhas, contra acessos não autorizados.

Além disso, é crucial implementar validação robusta tanto no *frontend* como no *backend* para evitar a introdução de dados maliciosos que possam comprometer a integridade do sistema.

Estas considerações envolvem conhecimentos avançados em segurança e conformidade com regulamentações de proteção de dados, como o Regulamento Geral sobre a Proteção de Dados (RGPD). Deste modo, a implementação completa deste caso de utilização requer profissionais especializados e recursos adequados, estando prevista para uma fase futura, quando for possível abordar todas as implicações com segurança e eficácia.

Como solução temporária, o Website apresentará um endereço de *e-mail* fornecido da empresa, permitindo que os utilizadores enviem mensagens diretamente, sem necessidade de um formulário de contacto intermediário.

4.3 Base de Dados

Nesta secção, será apresentada a estrutura da base de dados, destacando o Modelo Entidade-Associação (EA) e o Modelo Relacional, elementos fundamentais para compreender a organização dos dados e as suas relações.

Ao iniciar a implementação da base de dados através do Microsoft SQL Server, foi necessária a criação de um *login* no SQL Server [37]. A criação de um *login* é essencial, pois serve como mecanismo de autenticação para a ligação à instância do SQL Server. Na ausência de um *login*, nem o utilizador nem a aplicação conseguem estabelecer uma conexão com o servidor. Após a ligação ser estabelecida, esse *login* deve ser mapeado para um utilizador de base de dados para que se possa aceder e realizar operações na base de dados. No SQL Server, tanto o *login* como o utilizador de base de dados desempenham papéis fundamentais na gestão segura e eficiente do acesso aos dados da aplicação:

- ***Login***: corresponde à autenticação e à conexão à instância do SQL Server, permitindo o acesso ao servidor.

- **Utilizador**: define as permissões e as ações que podem ser executadas dentro da base de dados.

Desta forma, a configuração inicial da base de dados incluiu a criação de um *login* e do respetivo utilizador de base de dados, assegurando a ligação correta da aplicação à base de dados `wikiDB` e garantindo uma gestão controlada dos acessos.

Na base de dados `wikiDB`, o processo de configuração é realizado conforme ilustrado na Listagem 4.1.

Listagem 4.1: Configuração de autenticação de base de dados contida em SQL Server

```
1 USE wikiDB
2 EXEC sp_configure 'CONTAINED DATABASE AUTHENTICATION'
3 EXEC sp_configure 'CONTAINED DATABASE AUTHENTICATION', 1
4 RECONFIGURE
5 ALTER DATABASE wikiDB SET CONTAINMENT = PARTIAL
6 CREATE USER wikiDBUser WITH PASSWORD='*****', DEFAULT_SCHEMA=[dbo]
7 CREATE LOGIN wikiDBUser WITH PASSWORD=N'*****', DEFAULT_DATABASE=wikiDB,
      DEFAULT_LANGUAGE=[us_english], CHECK_EXPIRATION=OFF, CHECK_POLICY=OFF
8 ALTER LOGIN wikiDBUser ENABLE
9 EXEC sp_addrolemember 'db_owner', 'wikiDBUser'
```

Neste código, são efetuadas as seguintes configurações:

- **Linhas 2-4**: a opção `'CONTAINED DATABASE AUTHENTICATION'` ativa a autenticação de base de dados contida no SQL Server, permitindo a criação de utilizadores diretamente na base de dados sem depender de um *login* a nível da instância do SQL Server.

- **Linha 5**: modifica a configuração de contenção da base de dados `wikiDB` para `PARTIAL`, permitindo que a base de dados suporte utilizadores contidos que tenham permissões locais sem necessitar de um *login* a nível de instância.

- **Linha 6**: cria um novo utilizador de base de dados com uma palavra-passe específica, definindo o seu *schema* padrão como `dbo`, o que estabelece o contexto de acesso inicial para o utilizador.

- **Linha 7**: cria um novo *login* (`wikiDBUser`) na instância do SQL Server, associando-o à base de dados `wikiDB`.

- **Linhas 8-9**: ativa o *login* recém-criado e faz desse utilizador proprietário, atribuindo-lhe permissões administrativas na base de dados `wikiDB`.

Este conjunto de configurações permite uma gestão centralizada e segura dos acessos, facilitando simultaneamente o controlo de permissões específicas e garantindo que apenas utilizadores autenticados e autorizados possam realizar ações críticas na base de dados.

4.3.1 Modelo Entidade-Associação

Um modelo EA é uma representação gráfica que descreve a estrutura lógica de uma base de dados. Este é composto por entidades, uma abstração para a descrição de objetos ou conceitos que possuam um conjunto de características comuns, e atributos, propriedades ou características das entidades e representados por elipses conectadas às entidades. O modelo desenhado está representado na Figura 4.2.

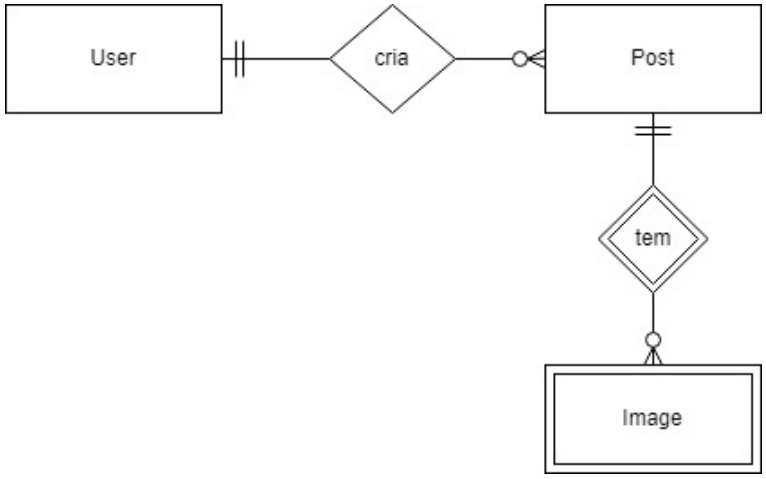

Figura 4.2: *Entidades do modelo EA*

No modelo EA proposto, a estrutura entre as entidades `Utilizador` (*User*), `Publicação` (*Post*) e `Imagem` (*Image*) é representada de forma a destacar as suas relações e dependências. Os termos relativos à implementação, como *User*, *Post* e *Image*, foram mantidos em inglês para garantir a coerência com a implementação técnica do sistema. No contexto deste modelo, um utilizador refere-se apenas aos administradores mencionados nos casos de utilização (Capítulo 3) . Um `Utilizador` é uma entidade que pode estar associada a zero ou várias publicações, o que significa que um administrador pode não ter nenhuma publicação associada ou pode ter várias publicações. Por outro lado, uma publicação está sempre associada a um único utilizador.

Relativamente às imagens, estas têm uma dependência direta com as publicações. Cada imagem está associada a uma única publicação, o que significa que uma imagem só existe no contexto do modelo se estiver vinculada a uma publicação específica. Esta dependência torna a entidade `Imagem` uma entidade fraca, pois a existência de uma imagem depende da existência de uma publicação. Adicionalmente, uma publicação pode conter zero ou várias imagens, reforçando a relação de um para muitos entre `Publicação` e `Imagem`.

Utilizadores

A Figura 4.3 apresenta a entidade `Utilizador`, que representa os administradores do sistema. Cada utilizador é identificado de forma única por um Identificador Único (ID), e as suas principais características incluem:

- `id`: um identificador único para cada utilizador;
- `email`: o endereço de *e-mail* associado ao utilizador, utilizado para autenticação;
- `password`: a senha associada ao utilizador, utilizada para garantir a segurança da conta.

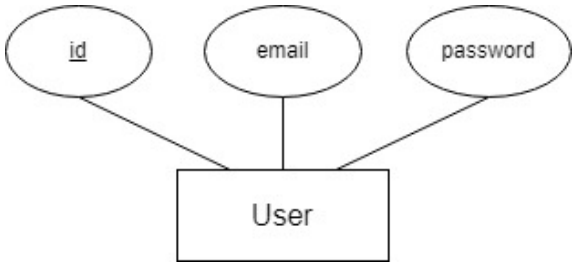

Figura 4.3: *Entidade User*

Publicações

A entidade `Publicação`, Figura 4.4, refere-se às publicações realizadas pelos administradores, dentro do sistema. Cada publicação é identificada de forma única por um ID e inclui várias propriedades para descrever o seu conteúdo. As características principais incluem:

- `id`: um identificador único para cada publicação;
- `title`: o título da publicação, que resume o conteúdo;
- `postType`: o tipo de publicação, que pode indicar duas categorias ou formatos (por exemplo, artigo ou documentação);
- `content`: o conteúdo ou corpo da publicação, neste caso o conteúdo em Markdown;
- `publicationDate`: a data em que a publicação foi criada ou publicada no sistema.

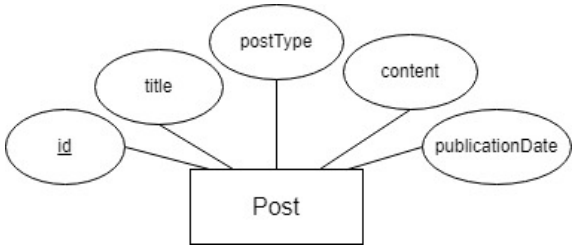

Figura 4.4: *Entidade Post*

Imagens

A entidade `Imagem`, representada na Figura 4.5, armazena informações sobre as imagens carregadas na criação de uma publicação. Cada imagem é identificada de forma única por um ID e possui várias propriedades para detalhar suas características. As propriedades incluem:

- `id`: um identificador único para cada imagem;
- `name`: o nome da imagem;
- `data`: os dados da imagem, representando o conteúdo visual;
- `type`: o tipo de imagem, que pode indicar o formato do arquivo (por exemplo, JPEG, PNG);
- `order`: indica a posição da imagem na publicação. Por exemplo, se for 0, a imagem é a de rosto; se for 1, faz parte do conteúdo da publicação. A imagem de rosto é a imagem representativa que é exibida junto ao título de uma publicação;
- `size`: o tamanho do arquivo da imagem.

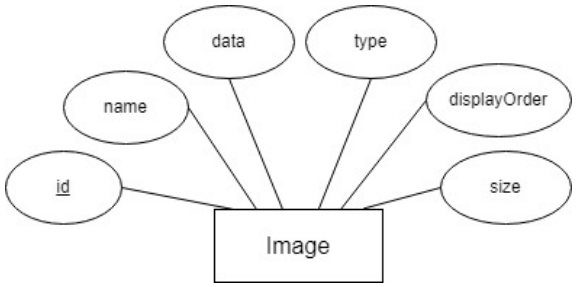

Figura 4.5: *Entidade Image*

4.3.2 Modelo relacional

O modelo relacional, conforme ilustrado na Figura 4.6, é uma abordagem estruturada e formal para a organização de dados em tabelas. Neste modelo, cada tabela representa uma entidade ou um relacionamento entre entidades, sendo cada uma identificada por uma chave primária, que assegura a unicidade de cada registo. As tabelas são criadas utilizando o comando SQL `CREATE TABLE`, que define a estrutura de cada tabela, incluindo os nomes das colunas e os seus tipos de dados.

Os relacionamentos entre as tabelas são definidos através de chaves estrangeiras, que fazem referência às chaves primárias de outras tabelas, garantindo a integridade referencial dos dados. A manipulação e consulta dos dados no modelo relacional são realizadas com a linguagem SQL, que permite operações como `INSERT` para inserção de dados, `SELECT` para consulta, `UPDATE` para atualização e `DELETE` para remoção de dados nas tabelas.

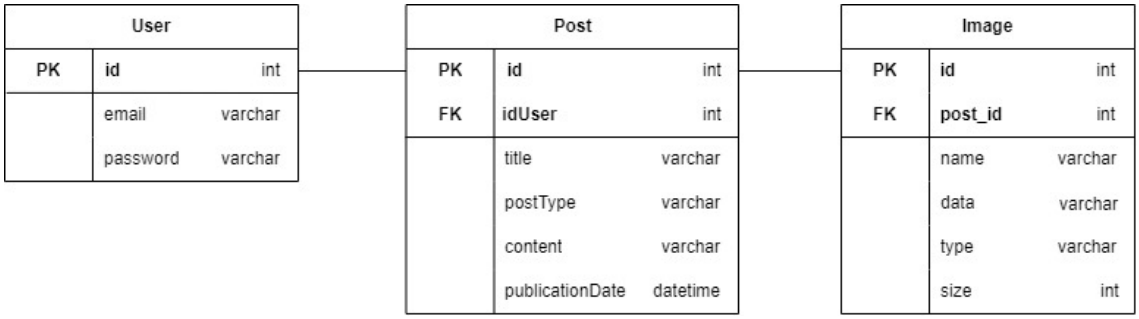

Figura 4.6: *Modelo relacional*

4.4 webMethods

Com base na compreensão prévia dos conceitos fundamentais do webMethods, incluindo o Integration Server, o Designer e o JDBC Adapter, esta secção irá focar-se na implementação prática das soluções de integração no contexto específico do projeto em desenvolvimento. O objetivo é detalhar as etapas e procedimentos adotados para criar uma solução que responda aos requisitos funcionais definidos.

Será detalhada a arquitetura da implementação em webMethods, abordando a estrutura dos componentes utilizados, a configuração do ambiente de integração, e a forma como os diferentes módulos, como o Integration Server, o Designer e o JDBC Adapter, interagem para suportar o fluxo de dados e a orquestração de serviços. Esta análise incluirá a descrição das ligações entre sistemas, a gestão de conectividade com bases de dados e a definição de serviços e processos que garantem a comunicação entre os diferentes elementos do sistema.

Na arquitetura apresentada na Figura 4.7, ilustra-se como a estrutura de um projeto em webMethods pode ser dividida em diferentes componentes, tais como *adapters*, *wrappers*, serviços privados, serviços públicos, e, neste projeto, serviços REST. Cada componente tem uma função específica na arquitetura geral e um propósito único, que será detalhadamente discutido nas subsecções seguintes, abordando tanto a sua finalidade quanto o processo de implementação.

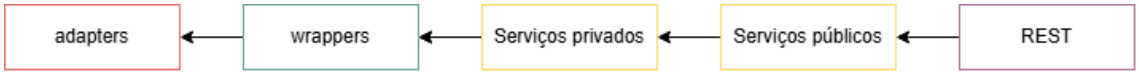

Figura 4.7: *Arquitetura de projeto em webMethods*

4.4.1 *Adapters*

No contexto de webMethods, um *adapter* [38] é um componente que permite a comunicação entre o Integration Server e recursos externos, como bases de dados relacionais. O webMethods Adapter para JDBC é um complemento do Integration Server que facilita a troca de dados com bases de dados relacionais através de um *driver* JDBC. Este *adapter* permite uma comunicação direta e em tempo real com a base de dados, sem necessidade de alterar a infraestrutura existente das aplicações.

A utilização do webMethods Adapter para JDBC requer a configuração de *adapter services*, que se conectam ao recurso do *adapter* e permitem a execução de operações a partir do

Integration Server. Estes serviços podem ser invocados através de *flow services* [39] ou serviços Java, possibilitando a interação direta com as tabelas da base de dados.

Configuração de Conexões do *Adapter* para JDBC

Para configurar um *adapter* para serviços JDBC, é necessário iniciar o Integration Server e o Integration Manager, assegurando que o utilizador tem privilégios de administrador.

Durante a configuração, devem ser fornecidas informações específicas sobre a conexão, como o tipo de base de dados, o URL de conexão, as credenciais de acesso e quaisquer propriedades adicionais exigidas pelo ambiente de integração. Como ilustrado na Figura-4.8, o utilizador previamente definido está agora associado a esta conexão.

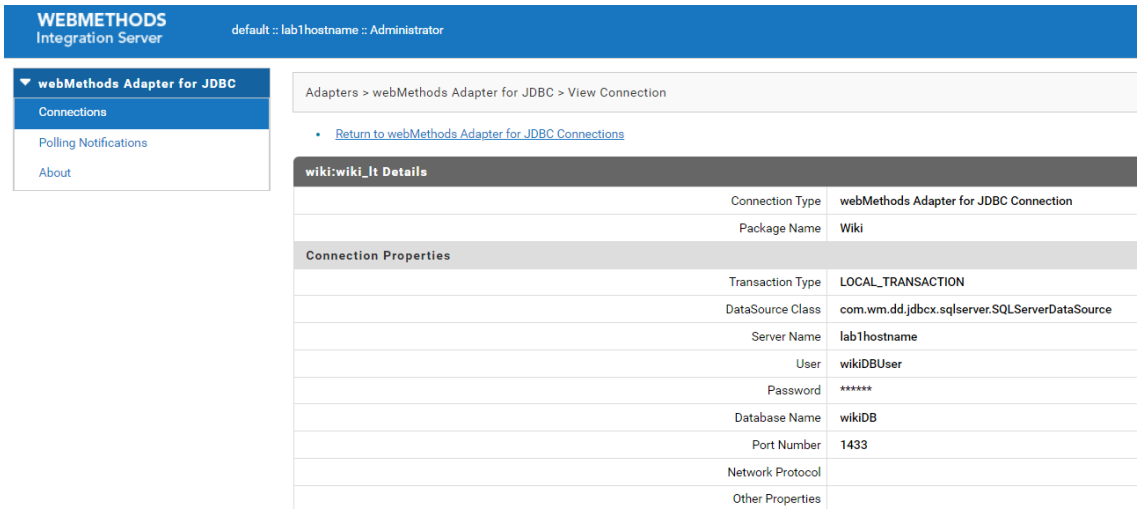

Figura 4.8: *webMethods Adapter para JDBC*

Processamento de *Adapter Services* em Tempo de Execução

A Figura 4.9, adaptada da documentação do webMethods Adapter para JDBC [38], ilustra como o Adapter para JDBC processa os *adapter services* em tempo de execução. A numeração presente no diagrama indica a sequência das operações realizadas.

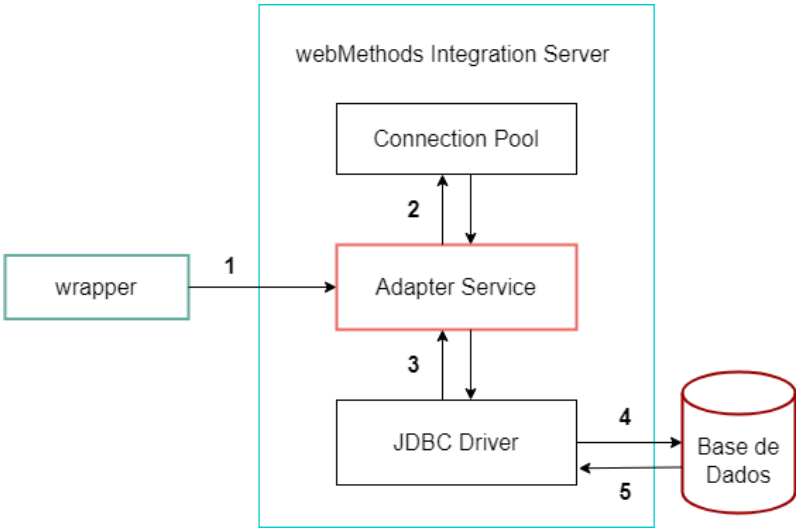

Figura 4.9: *Processamento de Transações dos Adapter Services*

No processo descrito, um cliente do Integration Server, como um *wrapper*, invoca um serviço do webMethods Adapter para JDBC no Integration Server para realizar uma operação numa base de dados (1). Após a invocação, o serviço do *adapter* obtém uma conexão a partir do *connection pool* configurado para esse serviço (2). Esta conexão inclui todas as informações necessárias para aceder à base de dados.

Com a ligação estabelecida, o *adapter service* utiliza o *driver* JDBC (3) adequado para interagir com a base de dados (4 e 5), recorrendo às configurações definidas previamente no Integration Server Administrator. Uma vez conectada, a aplicação pode executar diferentes operações SQL na base de dados, dependendo do tipo de serviço solicitado. Para operações como `SelectSQL`, `InsertSQL`, `UpdateSQL`, `DeleteSQL` e `DynamicSQL`, o *adapter service* executa diretamente a instrução SQL apropriada.

Foram criados vários *adapters*, cada um projetado para executar uma operação específica, os quais serão detalhados subsequentemente.

Adapters Relativos aos Utilizadores

Para a seleção dos utilizadores, foi importante assegurar que apenas a informação relevante fosse devolvida, evitando assim sobrecarregar o *frontend* com dados desnecessários. Para atingir esse objetivo, foram desenvolvidos três *adapters*, apresentados na Tabela 4.2

Tabela 4.1: *Adapters* para operações com utilizador

Nome	**Tipo de serviço**	***Input***	***Output***
selectUser	`SelectSQL`	*e-mail* *password*	id
insertUser	`InsertSQL`	*e-mail* *password*	*result*
deleteUser	`DeleteSQL`	*e-mail*	*result*

O `selectUser` é utilizado para a autenticação de utilizadores através da validação das suas credenciais de *login*. Este *adapter* conecta-se à base de dados e recebe como parâmetros de entrada o *e-mail* e a palavra-passe do utilizador. Caso as credenciais sejam validadas com sucesso, retorna o ID correspondente ao utilizador.

O `insertUser` tem como função inserir novos registos de utilizadores na base de dados. Esta operação é realizada exclusivamente no *backend* da aplicação e não é exposta diretamente ao *frontend*. A inserção de novos utilizadores é gerida por administradores ou responsáveis pela gestão do sistema, garantindo um controlo adequado sobre o acesso e a integridade dos dados.

Por sua vez, o `deleteUser` é responsável pela remoção de registos de utilizadores da base de dados, utilizando o *e-mail* como critério de exclusão. Assim como o *insertUser*, este *adapter* é utilizado exclusivamente no *backend*, e a operação de remoção é realizada pelos administradores do sistema. Desta forma, assegura-se que apenas utilizadores autorizados possam executar ações que modifiquem o estado dos registos de utilizadores.

Adapters Relativos às Publicações

Os *adapters* desenvolvidos relativamente às publicações, estão listados na Tabela 4.2.

Tabela 4.2: *Adapters* para operações com publicações

Nome	**Tipo de serviço**	***Input***	***Output***
selectPost	`SelectSQL`	id	id *idUser* *title* *postType* *content* *publicationDate*
selectPosts	`DynamicSQL`	*postType* *idUser*	id *title*
insertPost	`DynamicSQL`	*idUser* *title* *postType* *content*	*post_id*
updatePost	`DynamicSQL`	id *title* *postType* *content*	*result*
deletePost	`DeleteSQL`	id	*result*

O `selectPost` é um adaptador utilizado para recuperar todas as informações de uma publicação especifica na base de dados, com base no seu id. Ele retorna todos os detalhes associados à publicação, como título, conteúdo, data de criação, entre outros.

O `selectPosts` permite a recuperação de uma lista de publicações a partir da base de dados, retornando apenas o título e o ID de cada publicação. A seleção das publicações pode ser feita de acordo com o tipo de publicação (*postType*) ou o ID do utilizador (*idUser*).

O `insertPost` é responsável pela inserção de novas publicações na base de dados. Este adaptador foi criado através de DynamicSQL, pois necessita de retornar o ID da publicação criada para ser utilizado posteriormente.

O `updatePost` permite a atualização de publicações existentes na base de dados. Através dele, é possível modificar informações de uma publicação já armazenada, como título, conteúdo ou outros detalhes relevantes.

O `deletePost` possibilita a remoção de uma publicação da base de dados com base no seu ID.

Adapters Relativos às Imagens

Os *adapters* desenvolvidos relativamente às publicações, estão listados na Tabela 4.3.

Tabela 4.3: *Adapters* para operações com imagens

Nome	Tipo de serviço	Input	Output
selectFrontImage	`SelectSQL`	*post_id* *display_order* = 0	*id* *name* *data* *type* *display_order* *size* *post_id*
selectImages	`SelectSQL`	*post_id* *display_order* = 1	*id* *name* *data* *type* *display_order* *size* *post_id*
insertImage	`InsertSQL`	*name* *data* *type* *display_order* *size* *post_id*	*result*
deleteImage	`DeleteSQL`	*post_id*	*result*

O `selectFrontImage` é utilizado para selecionar a imagem principal de uma publicação específica na base de dados, identificada pelo ID da publicação e com ordem de exibição igual a 0. Este *adapter* retorna vários detalhes da imagem, incluindo o seu ID, nome (*name*), dados da imagem em formato base64 (*data*), tipo (*type*), ordem de exibição (*display_order*), tamanho (*size*), e o identificador da publicação à qual pertence (*post_id*).

O *adapter* `selectImages` permite a recuperação de todas as imagens associadas a uma publicação específica na base de dados, exceto a imagem principal. A seleção é feita com base no ID da publicação e com ordem de exibição igual a 1, que indica que a imagem não é a principal. Este adaptador retorna uma lista de imagens, devolvendo todos os detalhes de cada uma.

O `insertImage` é responsável pela inserção de novas imagens no banco de dados associadas a uma publicação específica. Requer vários parâmetros de entrada, incluindo o nome, dados, tipo, ordem de exibição, tamanho e o identificador da publicação ao qual a imagem pertence. É retornado um resultado que indica o sucesso ou falha da operação de inserção.

O `deleteImage` é utilizado para remover todas as imagens referentes a uma publicação, com base no seu identificador. É retornado um resultado que indica o sucesso ou falha da operação de exclusão.

4.4.2 *Wrappers*

O *wrapper*, exemplificado na Figura 4.10, chama o *adapter service* sublinhado a cinzento e utiliza uma estrutura de tratamento de exceções, como o bloco `try-catch`, para gerir potenciais erros ou falhas que possam ocorrer durante a execução da operação. Esta abordagem é crucial para garantir a robustez e resiliência da aplicação, pois permite capturar exceções que podem ser lançadas devido a falhas na conexão, problemas de execução de comandos SQL, ou outras condições de erro relacionadas com o acesso à base de dados. Para cada *adapter service*, foi desenvolvido um *wrapper* correspondente.

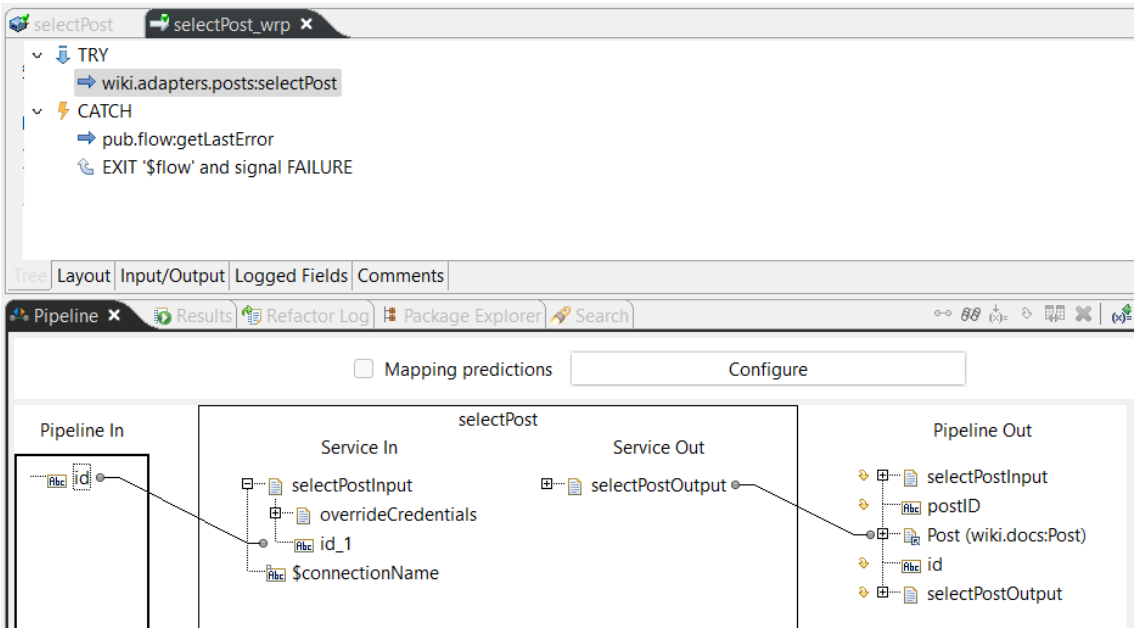

Figura 4.10: *Exemplo de utilização de um wrapper no Integration Server*

No contexto do *wrapper*, é introduzido o conceito de *flow service*. Um *flow service* [39] é um serviço escrito na linguagem de fluxo (*flow language*) do webMethods, sendo constituído por uma série de passos (*flow steps*) que o Integration Server interpreta e executa em tempo de execução (*runtime*).

A linguagem de fluxo do webMethods oferece vários tipos de passos de fluxo que permitem a invocação de serviços e a manipulação lógica de dados. Nos *wrappers* foram utilizados os passos de fluxo `TRY`, `CATCH` e `INVOKE`.

Para facilitar as interações, inserções e outras operações no sistema, foram definidos documentos na plataforma webMethods (Figura 4.11), para o tratamento das entidades *Image*, *Post* e *SimplePost*. Esta criação permite uma estrutura clara e bem organizada, o que melhora a legibilidade e a manutenção do código. Além disso, a utilização de documentos centralizados e reutilizáveis oferece maior flexibilidade, pois ajustes podem ser feitos de forma isolada, sem a necessidade de atualizar manualmente múltiplos pontos de interação no sistema.

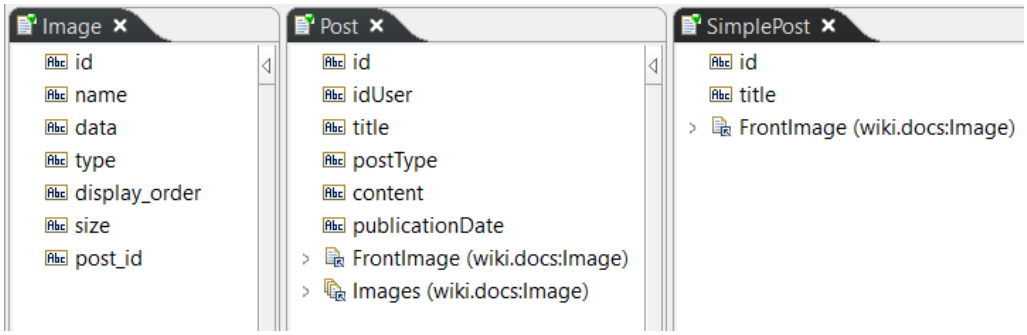

Figura 4.11: *Documentos criados no Integration Server*

O documento *Image* inclui os atributos mencionados anteriormente na definição da entidade. Estes campos são utilizados para definir e manipular informações relacionadas com as imagens armazenadas no sistema.

O documento *Post* inclui os atributos definidos na entidade e referências a documentos incorporados, como *FrontImage* (imagem de rosto) e *Images* (imagens do conteúdo), ambos referências ao documento *Image* definido anteriormente. Esta estrutura permite um acesso direto e eficiente às imagens associadas a uma determinada publicação.

Por sua vez, o documento *SimplePost* é uma versão simplificada de *Post*, contendo apenas o ID, o título e uma referência ao *FrontImage*. Este formato é útil quando apenas um resumo da publicação é necessário, reduzindo a quantidade de dados transferidos e melhorando o desempenho do sistema.

4.4.3 Serviços Privados

Os serviços privados em webMethods, referem-se a serviços internos que não são expostos diretamente a sistemas externos. Estes serviços são responsáveis por lidar com a lógica central de negócio ou por interagir com sistemas subjacentes através de *wrappers*. Ao operarem internamente, os serviços privados garantem a segurança e a integridade dos processos, mantendo o controlo sobre a execução das operações críticas dentro do ambiente do Integration Server. De seguida, serão apresentados os serviços privados desenvolvidos.

selectPost

O serviço privado *selectPost*, ilustrado na Figura 4.12, é responsável por recuperar todas as informações relacionadas a uma publicação específica, invocando uma série de *wrappers* que obtêm os dados necessários. Este serviço agrega várias operações num único fluxo, permitindo que diferentes componentes, como dados textuais e imagens associadas a uma publicação, sejam recuperados simultaneamente.

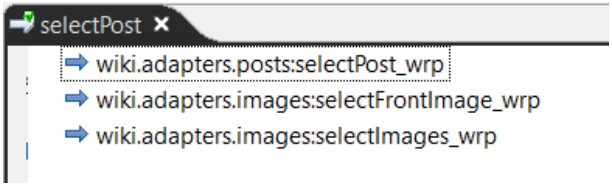

Figura 4.12: *Serviço privado relativo à obtenção de uma publicação*

Neste serviço, são invocados três *wrappers* e o resultado de cada *wrapper* é, então, mapeado para o documento *Post*, preenchendo os seus atributos com as informações recuperadas, como detalhes da publicação e das imagens associadas. Esta abordagem garante que o documento *Post* seja completo e contenha todas as informações relevantes de uma só vez. Este processo integrado simplifica o acesso aos dados e reduz a complexidade das operações, ao minimizar o número de chamadas ao sistema e consolidar a recuperação de dados numa única execução. Além disso, melhora o desempenho do sistema e facilita a manutenção, centralizando a lógica de recuperação de dados num único ponto de interação.

selectPosts

O serviço privado *selectPosts*, ilustrado na Figura 4.13, é responsável por recuperar publicações referentes a um tipo (artigo ou documentação) ou a um utilizador específico.

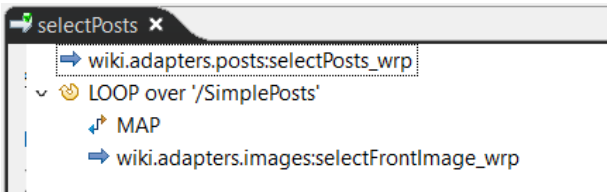

Figura 4.13: *Serviço privado relativo à obtenção de publicações*

O *wrapper selectPosts* devolve uma lista de *SimplePosts*, que é processada iterativamente. Neste serviço, foi necessário utilizar um `loop` para que em cada publicação recuperada, seja possível obter a imagem de rosto correspondente através do *wrapper selectFrontImage*. Durante a iteração, o ID de cada publicação na lista é transmitido ao *wrapper selectFrontImage* para recuperar a imagem de rosto associada. Como resultado, obtém-se uma lista de documentos *SimplePosts*.

insertPost

O serviço privado *insertPost*, ilustrado na Figura 4.14, é responsável pela inserção de novas publicações na base de dados, incluindo o conteúdo textual da publicação e as imagens associadas à mesma (caso existam).

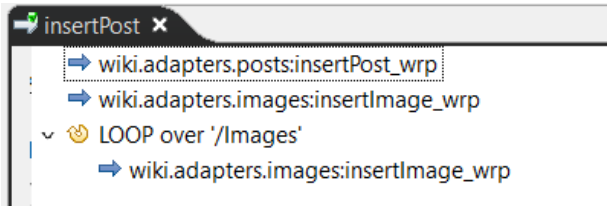

Figura 4.14: *Serviço privado de inserção de publicações*

O *wrapper insertPost_wrp* é utilizado para inserir na base de dados os dados principais da publicação, como o título e o conteúdo, conforme mencionado na secção relativa aos *adapters* das imagens. Em seguida, o *wrapper insertImage_wrp* é chamado para inserir a imagem de rosto associada à publicação. No entanto, para inserir todas as imagens adicionais relacionadas a uma publicação, é necessário recorrer a um `loop` que percorre

a lista de imagens fornecida. Durante esta iteração, cada imagem é fornecida ao *wrapper insertImage_wrp* para ser inserida na base de dados.

updatePost

O serviço privado *updatePost*, ilustrado na Figura 4.15, é responsável pela atualização de publicações existentes na base de dados, incluindo a modificação do conteúdo textual e a gestão das imagens associadas.

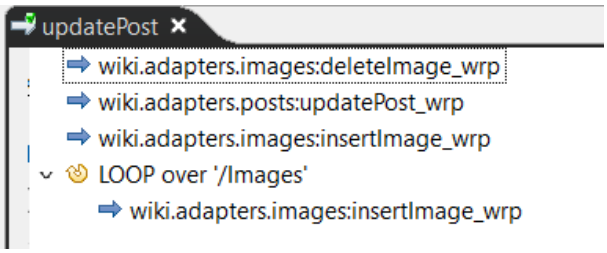

Figura 4.15: *Serviço privado de atualização de publicações*

O processo de atualização inicia-se com o *wrapper deleteImage_wrp*, que remove todas as imagens associadas à publicação armazenadas na base de dados. De seguida, o *wrapper updatePost_wrp* atualiza os dados da publicação, como o título e o conteúdo. Por fim, tal como no serviço *insertPost*, as novas imagens são inseridas.

selectUser

No serviço privado *selectUser*, ilustrado na Figura 4.16, não há lógica de negócio associada devido à simplicidade do serviço. Assim, apenas é feita a invocação do *wrapper* associado ao *selectUser*.

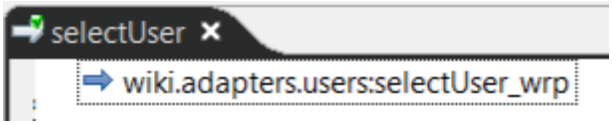

Figura 4.16: *Serviço privado para autenticação de utilizador*

4.4.4 Serviços Públicos

Os serviços públicos em webMethods são aqueles expostos a sistemas externos, geralmente através de APIs ou outros pontos de integração. Estes serviços podem interagir com serviços privados para processar as solicitações recebidas de fontes externas. Neste caso, é necessário utilizar um bloco `try-catch` para garantir o tratamento adequado de erros, conforme ilustrado na Figura 4.17.

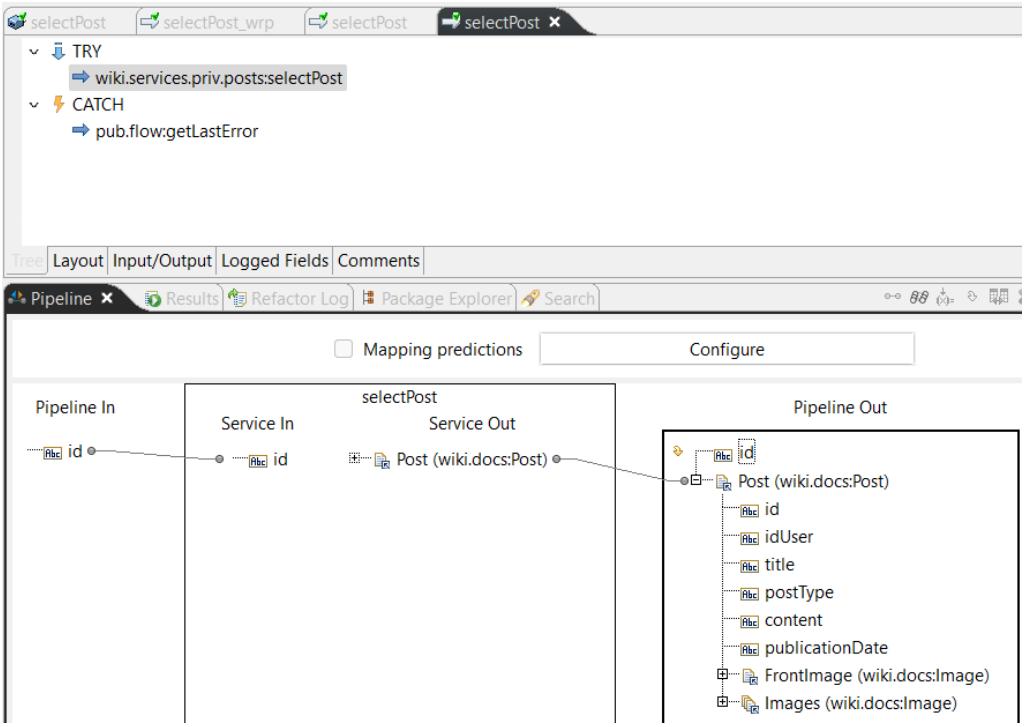

Figura 4.17: *Exemplo de serviço público*

4.4.5 API REST

Como componente final da arquitetura de *backend* do projeto em webMethods, foi desenvolvida uma API que permite a comunicação e troca de dados com o *frontend*. No webMethods, esta funcionalidade é implementada através de REST Resources [40], que possibilitam a criação de *endpoints* para expor serviços internos a sistemas externos, facilitando a integração de dados e a comunicação por meio de APIs REST.

A arquitetura REST identifica recursos por meio de URLs e utiliza métodos HTTP, como `GET`, `POST`, `PUT` e `DELETE`, para manipular esses recursos. Neste projeto, os recursos foram configurados para receber dados no formato JSON, devido à sua simplicidade e facilidade de utilização no que toca à troca de informações entre o *frontend* e o *backend*, como no caso de inserção de novas publicações.

Para aceder aos serviços públicos desenvolvidos, foram criados pontos de acesso, mais conhecidos como *endpoints*, específicos para as publicações e para os utilizadores.

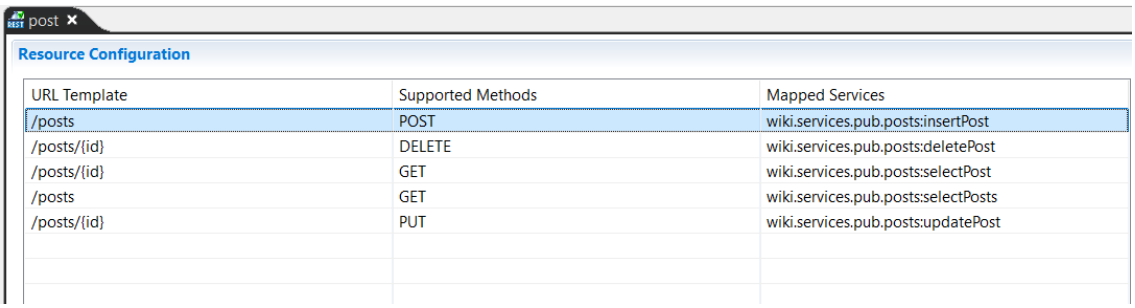

Figura 4.18: *Endpoints para acesso às publicações*

Os *endpoints* criados para gerir as publicações, ilustrados na Figura 4.18 são os seguintes:

- **POST /posts:** cria uma nova publicação;
- **DELETE /posts/{id}:** elimina uma publicação específica com base no seu ID;
- **GET /posts/{id}:** recupera os detalhes de uma publicação específica com base no seu ID;
- **GET /posts:** recupera uma lista de publicações;
- **PUT /posts/{id}:** atualiza uma publicação existente com base no seu ID.

Os *endpoints* relativos aos utilizadores estão apresentados na Figura 4.19. Neste caso, foi criado apenas um *endpoint* que irá receber como parâmetros o *e-mail* e a *password* do utilizador.

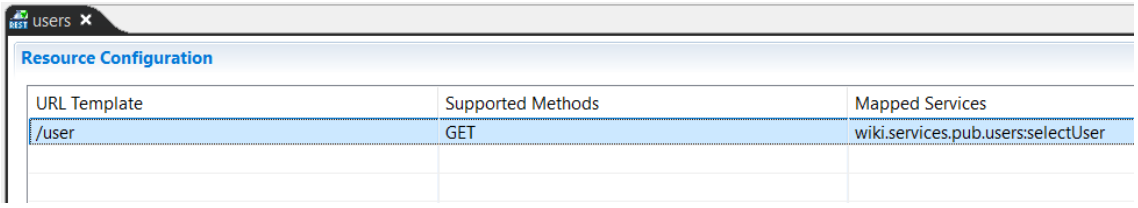

Figura 4.19: *Endpoint para acesso aos utilizadores*

4.5 Aplicação Web

Esta secção irá centrar-se na camada da arquitetura do sistema com a qual o utilizador interage, detalhando a implementação do *frontend* do projeto.

4.5.1 Arquitetura

No desenvolvimento do *frontend*, é fundamental definir um padrão de arquitetura para manter o código organizado e simplificar a manutenção. Estes padrões permitem criar aplicações que são mais escaláveis, reutilizáveis e simples de compreender e testar. Para este projeto, optou-se pelo padrão *Model-View-Controller* (MVC) [41], com a estrutura apresentada na Figura 4.20, amplamente utilizado em aplicações web devido à sua capacidade de separar de forma clara as responsabilidades entre as diferentes partes da aplicação.

Figura 4.20: *Arquitetura Model-View-Controller*

4.5.2 Modelo

O Modelo (*Model*) é responsável pela gestão dos dados e pela lógica de negócio da aplicação. No código, essa responsabilidade é implementada por meio de modelos de dados e serviços.

Para representar os modelos de dados, foram definidas três interfaces que seguem a mesma estrutura dos documentos previamente criados no Integration Server. A Listagem 4.2 apresenta o modelo *Post* como exemplo e os modelos *Image* e *SimplePost* são definidos de maneira semelhante. Estes modelos de dados são utilizados pelo *frontend* para assegurar a consistência do código e facilitar a troca de informações com os serviços da aplicação.

Listagem 4.2: Exemplo de um Modelo

```
export interface Post {
    id: string;
    idUser: string;
    title: string;
    postType: string;
    content: string;
    publicationDate: string;
    frontImage: Image;
    images: Image[];
  }
```

Relativamente aos serviços, foi necessário, em primeiro lugar, configurar o acesso à API, através da criação de um ficheiro `proxy.conf.json`, cuja estrutura está apresentada na Listagem 4.3. Este ficheiro define o URL do servidor onde o *backend* está alojado e configura o *proxy* para encaminhar as requisições de forma adequada. O uso de um *proxy* facilita a manutenção do código, uma vez que permite alterar os dados apenas neste ficheiro, sem necessidade de modificar diretamente o serviço.

Listagem 4.3: Configuração de proxy para acesso à API

```
{
    "/api": {
      "target": "http://lab1hostname:5555",
      "secure": false,
      "changeOrigin": true,
      "pathRewrite": {
        "^/api": ""
      }
    }
}
```

Foram desenvolvidos três serviços para a aplicação: um para a gestão de publicações (*PostService*), outro para a gestão de utilizadores (*UserService*) e um terceiro para autenticação. Os serviços de publicações e de utilizadores contêm a lógica necessária para interagir com os respetivos *endpoints* da API, utilizando autenticação básica e parâmetros de URL.

Durante a instalação inicial do webMethods Integration Server, é configurado um utilizador administrador padrão que permite o acesso inicial ao servidor. Para proteger a base de dados contra acessos não autorizados e garantir a segurança das requisições ao *backend*, é

necessária a autenticação das solicitações. Neste projeto, é utilizada a autenticação básica (*Basic Authentication*), na qual o nome de utilizador e a palavra-passe são os definidos no webMethods Integration Server. Estas credenciais, são codificadas em `base64`, e o *token* gerado é inserido nos cabeçalhos HTTP, no campo "*Authorization*" como `Basic ${token}`. Todas as requisições efetuadas à API utilizam esta autenticação, assegurando segurança e consistência no acesso aos serviços.

O *PostService* disponibiliza métodos para interagir com a API implementada e realizar operações sobre os dados. O acesso aos dados é feito através do HttpClient do Angular [42], que permite a comunicação HTTP com o servidor. Abaixo, são apresentados os principais métodos do *PostService*:

- ***getPosts***: recuperar uma lista de publicações do *backend*. Utiliza o método `GET` do HttpClient e permite filtrar os resultados por *postType* ou *idUser* através de parâmetros de URL.
- ***getPostById***: recuperar um *Post* específico a partir do seu *id*. Também utiliza o método `GET` do HttpClient, mas inclui o ID no final do URL para direcionar a requisição ao recurso específico.
- ***createPost***: criar uma publicação. Utiliza o método `POST` do HttpClient, enviando os dados da publicação no corpo da requisição (*body*) em formato JSON.
- ***updatePost***: atualizar uma publicação existente. Este método utiliza o método `PUT` do HttpClient e envia os dados atualizados no corpo da requisição (*body*) em formato JSON. O URL inclui o ID da publicação que se pretende atualizar.
- ***deletePost***: eliminar uma publicação específica com base no seu ID. Utiliza o método `DELETE` do HttpClient, incorporando o ID no URL para identificar o recurso que deve ser eliminado.

É importante salientar que, nos pedidos de `createPost` e `updatePost`, a informação é enviada para a API através do corpo (*body*) da requisição em formato JSON. Estes dados estão em conformidade com os *inputs* esperados pelos serviços do *webMethods*, sendo esta conformidade assegurada através da utilização de documentos nos *webMethods* e de *models* no Angular (referidos anteriormente) que possuem a mesma estrutura.

Na Listagem 4.4, é apresentado um método onde se exemplifica o corpo enviado num pedido HTTP, como no caso de um pedido `createPost`. Neste método, foi necessária a verificação da existência de uma imagem frontal, que altera o conteúdo do pedido enviado.

Listagem 4.4: Exemplo de um método do serviço PostService

```
1  createPost(post: Post): Observable<any> {
2     let data = {};
3     if(!post.frontImage) {
4       data = {
5         "idUser": post.idUser,
6         "title": post.title,
```

```
7           "postType": post.postType,
8           "content": post.content,
9           "Images": post.images.map(image => ({
10            "name": image.name,
11            "data": image.data,
12            "type": image.type,
13            "display_order": image.display_order,
14            "size": image.size
15          }))
16        };
17      }else{
18        data = {
19        "idUser": post.idUser,
20        "title": post.title,
21        "postType": post.postType,
22        "content": post.content,
23        "FrontImage": {
24          "name": post.frontImage.name,
25          "data": post.frontImage.data,
26          "type": post.frontImage.type,
27          "display_order": post.frontImage.display_order,
28          "size": post.frontImage.size
29        },
30        "Images": post.images.map(image => ({
31          "name": image.name,
32          "data": image.data,
33          "type": image.type,
34          "display_order": image.display_order,
35          "size": image.size
36        }))
37      };
38    }
```

O *UserService* realiza operações relacionadas com os utilizadores e fornece apenas um método principal:

- ***getUser***: recuperar as informações de um utilizador específico com base no seu *e-mail* e *password*. Utiliza o método `GET` do HttpClient para enviar uma requisição ao *endpoint* definido pelo URL base. Os parâmetros de consulta *e-mail* e `password` são adicionados ao URL para fornecer as credenciais do utilizador que se pretende autenticar.

Por fim, foi implementado um serviço de autenticação que processa o *login* do utilizador. Se o método *getUser*, mencionado anteriormente, retornar um ID válido, a autenticação é considerada bem-sucedida. Após o *login*, as credenciais de autenticação do utilizador são guardadas no armazenamento local (*localStorage*) do navegador. Este armazenamento permite que o utilizador permaneça automaticamente autenticado sempre que reabrir a aplicação Web no seu navegador, proporcionando uma experiência de utilização mais fluida e conveniente.

Adicionalmente, foi implementado um mecanismo de proteção de rotas, conhecido como Auth Guard, que assegura o acesso controlado às rotas da aplicação com base no estado de autenticação do utilizador. Este mecanismo verifica se o utilizador possui as permissões necessárias para aceder a determinadas páginas ou funcionalidades. Por exemplo, se um administrador não estiver autenticado, o Auth Guard bloqueia o acesso a páginas exclusivas para administradores e redireciona o utilizador para a página de autenticação, garantindo que apenas utilizadores autorizados possam aceder a áreas restritas da aplicação.

4.5.3 Vista e Controlador

O Angular utiliza uma arquitetura baseada em componentes, em que cada componente encapsula a sua própria lógica e estrutura. Em cada componente, é possível separar a Vista (*View*) do Controlador (*Controller*). A Vista é responsável por definir a interface visual de um componente, utilizando HTML para a sua estrutura e CSS para o seu estilo. Por sua vez, o Controlador, escrito em TypeScript, contém a lógica de apresentação e a gestão das interações, manipulando o estado do componente e respondendo aos eventos gerados pelos utilizadores, como cliques e introdução de dados.

A criação dos componentes foi orientada pelos casos de utilização identificados. Como foram definidos dois tipos de utilizadores distintos, optou-se por organizar as pastas do projeto de modo a refletir os diferentes tipos de acesso à aplicação Web. Assim, a organização das pastas foi estruturada da seguinte forma:

- **Componentes de Administração**: focados nas páginas de funções relativas ao administrador;
- **Componentes de Visitantes**: focados nas páginas dos utilizadores que pretendem visualizar o conteúdo da aplicação Web;
- **Componentes Utilitários**: elementos reutilizáveis da interface de utilizador.

Componentes de Administração

Relativamente aos componentes de administração, foram definidos quatro componentes que correspondem às quatro páginas a que os administradores podem aceder.

O primeiro componente desenvolvido foi o *authentication*, responsável pela autenticação dos utilizadores. Nesta página, os utilizadores introduzem as suas credenciais de acesso, que são processadas pelo controlador, permitindo ou não a navegação para a página seguinte.

Em seguida, caso a autenticação seja bem-sucedida, o utilizador é redirecionado para a vista do componente *admin-home*. Este componente funciona como o painel principal ou página inicial para os administradores, onde estes podem escolher entre criar uma nova publicação, gerir publicações existentes ou efetuar *logout*.

O componente *create-edit-post* foi concebido para evitar a criação de dois componentes distintos (um para criar e outro para editar publicações), o que resultaria em repetição de código. Através do controlador deste componente, é possível determinar se o administrador

selecionou previamente a opção de criar ou editar uma publicação e caso tenha selecionado a criação, é apresentado um formulário vazio para introdução de dados, e o pedido efetuado ao serviço é de criação de uma nova publicação. No entanto, se a opção for de edição, o formulário é automaticamente preenchido com o conteúdo da publicação selecionada, incluindo as suas imagens e o pedido ao serviço é de atualização.

Este componente também permite pré-visualizar a publicação enquanto esta é criada ou editada, funcionalidade que é proporcionada através da integração com outro componente dos componentes utilitários, que será explicado mais à frente. No Angular, é possível combinar vários componentes numa única página, o que permite uma estrutura modular e reutilizável.

Por fim, foi implementado o componente *manage-posts*, responsável pela gestão das publicações existentes. Este componente apresenta uma visualização em lista ou grelha de todas as publicações, permitindo filtrar por título, tipo de publicação ou pelas publicações criadas pelo próprio administrador autenticado. Em cada publicação, é possível executar ações como editar ou eliminar.

Componentes de Visitantes

Relativamente aos componentes destinados aos visitantes, foram definidos quatro componentes: dois correspondem a páginas acessíveis aos visitantes e os outros dois servem como suporte, sendo utilizados nos primeiros dois componentes.

O componente *contact* foi concebido para permitir que os utilizadores entrem em contacto com os administradores da aplicação. Inclui um formulário de contacto para o envio de perguntas ou pedidos de assistência e a sua implementação encontra-se planeada para futuras iterações do projeto, visando facilitar a comunicação entre os utilizadores e a equipa de gestão da aplicação.

O componente *home* implementa a página principal da aplicação, e este componente, à semelhança do componente *manage-posts*, apresenta uma visualização em lista ou grelha de todas as publicações, permitindo filtrar por título ou por tipo de publicação.

Ao clicar numa publicação apresentada na grelha da página *home*, o utilizador é redirecionado para o componente *post*. Este componente exibe o conteúdo completo da publicação selecionada, bem como outras publicações do mesmo tipo (artigo ou documentação). O componente é construído em parte com base noutro componente utilitário, que processa a visualização de uma publicação a partir dos dados do modelo "*Post*".

Por fim, foi criado o componente *navbar*, ou barra de navegação, que fornece a navegação de nível superior para a aplicação e é visível e reutilizável em todas as páginas dos visitantes. Este componente inclui o logótipo da empresa e um acesso ao formulário de contacto.

Componentes Utilitários

Estes componentes são projetados para serem utilizados em várias páginas da aplicação, tanto nas páginas de visitantes como nas de administrador, ou para fornecer funcionalidades utilitárias.

O componente *footer* (rodapé) é normalmente localizado na parte inferior de cada página e inclui informações como avisos de direitos de autor, *links* para os termos de serviço e política de privacidade, bem como ligações para redes sociais e outros recursos úteis. Este é um componente reutilizável que ajuda a manter uma aparência consistente em toda a aplicação, uma vez que está presente em todas as páginas da aplicação Web.

O componente *post-container* é utilizado para exibir o conteúdo de uma publicação, incluindo o título, a imagem de destaque e a conversão de texto em Markdown para HTML, gerindo também a conversão de texto que faz referência a imagens, transformando-o nas respetivas imagens incorporadas no conteúdo. Este componente é reutilizado em dois contextos: na pré-visualização de uma publicação, para que o administrador possa ajustar o conteúdo de forma eficaz, e na visualização final da publicação para os utilizadores.

Capítulo 5

Validação e Testes

A validação e os testes são essenciais para assegurar que o sistema cumpre os requisitos especificados e funciona como esperado. No contexto do desenvolvimento de *software*, especialmente em sistemas complexos que envolvem múltiplos componentes, como o *middleware* deste projeto, a realização de testes contínuos é fundamental para a deteção precoce de erros e a garantia de qualidade. Um processo de testes bem estruturado contribui significativamente para a robustez e fiabilidade da aplicação.

Durante o desenvolvimento da aplicação Web, foi aproveitada a capacidade avançada de *debugging* do webMethods para a realização de vários testes focados no *middleware*. A estrutura do webMethods permite testes funcionais em cada fase do projeto através do Software AG Designer, onde é possível realizar *debug* de forma eficiente na vista dedicada. Se houver algum problema, cada passo do serviço pode ser analisado detalhadamente, facilitando a identificação e correção de erros.

Por exemplo, após a implementação de um *adapter*, este pode ser executado diretamente no Software AG Designer através da funcionalidade Run Service. Esta opção permite a abertura de uma janela onde se pode introduzir os dados de entrada (*input*) pretendidos, caso aplicável. Deste modo, o serviço implementado pode ser testado de forma imediata, possibilitando a deteção e correção de eventuais inconformidades relativamente ao comportamento esperado. Durante o processo de implementação, no ambiente do *webMethods*, foram realizados testes aos *adapters*, *wrappers* e aos serviços públicos e privados, com o objetivo de assegurar o acesso adequado à base de dados e garantir a correta execução das operações CRUD.

Foram conduzidos os testes relacionados com a API, conforme descrito na secção 5.1, bem como os testes funcionais do Website, onde é analisado o fluxo de navegação entre páginas e verificado o seu correto funcionamento, conforme detalhado na secção 5.1.

5.1 Validação e Testes à API

Para garantir que a API desenvolvida estava configurada corretamente e que todos os *endpoints* respondiam conforme o esperado, foi utilizada a plataforma Postman. O Postman [43] é uma ferramenta amplamente utilizada para testar APIs, permitindo realizar requisições HTTP e verificar as respostas retornadas pelo servidor, incluindo o código de *status*, os cabeçalhos e o corpo da resposta.

De seguida, serão detalhados os testes realizados para cada *endpoint*. Antes de efetuar cada pedido, à semelhança do processo realizado no Angular, é necessário configurar a autorização como *Basic Auth*, utilizando as credenciais de acesso ao Integration Server.

Na Figura 5.1 é apresentado o teste do *endpoint* `POST /posts`, que permite a criação de uma nova publicação. Durante este teste, verificou-se o envio correto dos dados relativos à nova publicação, bem como o retorno da resposta apropriada por parte da API. No *Postman*, a obtenção do código de estado `200 OK` indica que o pedido HTTP foi bem-sucedido. Para este teste, foi submetida uma publicação sem imagens, para facilitar o processo de teste. A resposta incluiu o identificador (ID) da publicação gerada, confirmando assim o sucesso da operação e a criação da nova publicação no sistema.

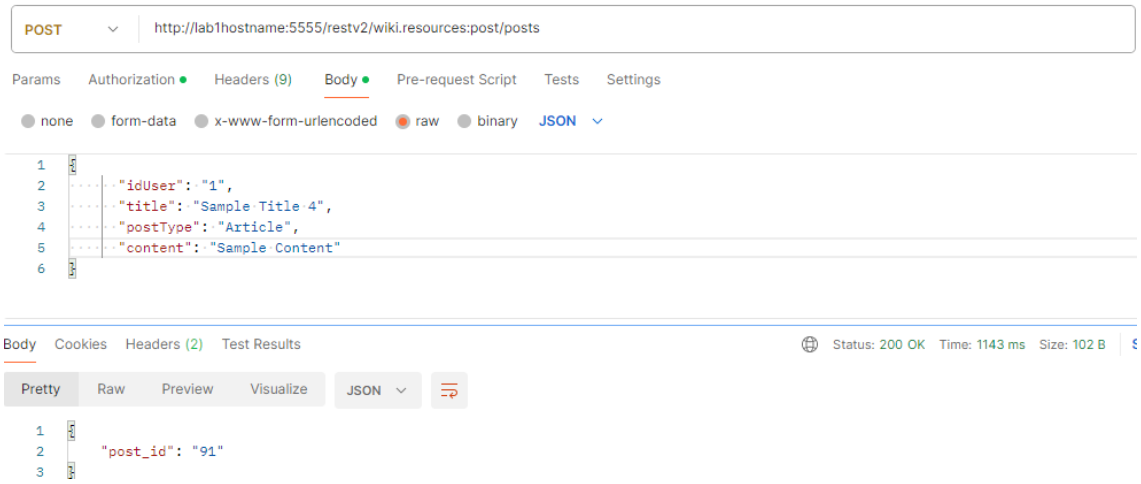

Figura 5.1: *Teste da API para criar uma publicação*

Na Figura 5.2, é ilustrado o teste referente ao *endpoint* `DELETE /posts/{id}`, que elimina uma publicação específica com base no seu ID. Durante este teste, foi avaliado o comportamento da API ao eliminar a publicação e a sua resposta. No *Postman*, o status é apresentado como `200 OK` e na resposta é devolvido *result* de 1, indicando que a publicação foi removida com sucesso.

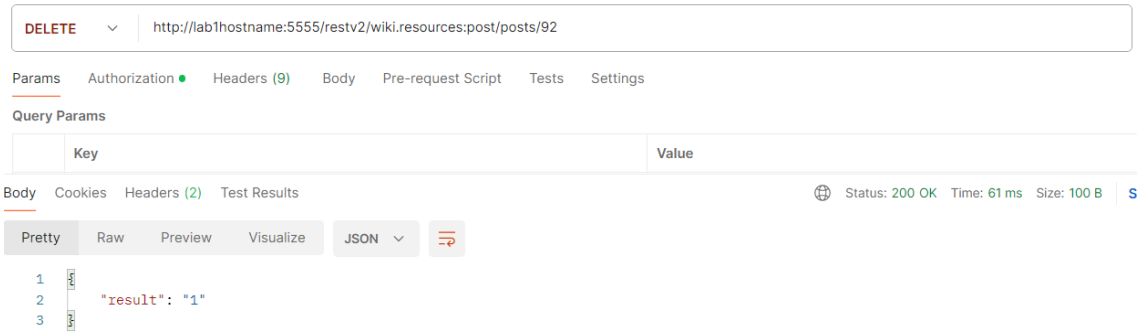

Figura 5.2: *Teste da API para eliminar uma publicação*

A Figura 5.3 apresenta o teste do *endpoint* `GET /posts/{id}`, que recupera os detalhes de uma publicação específica com base no seu ID. Este teste verifica se a API retorna corretamente os dados da publicação solicitada. O status é apresentado como `200 OK`, e na resposta é possível observar o conteúdo da publicação.

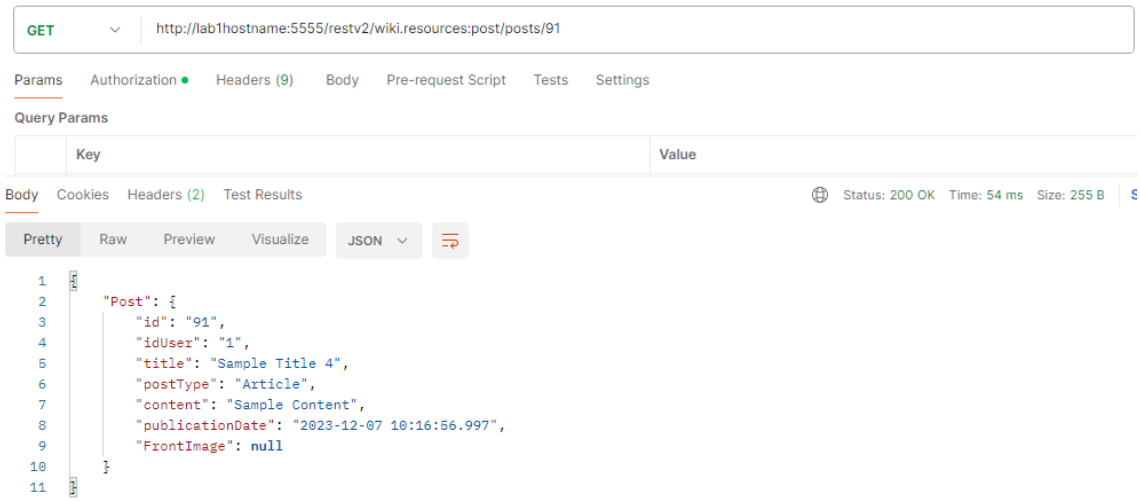

Figura 5.3: *Teste da API para recuperar publicação*

Na Figura 5.4, é mostrado o teste do *endpoint* `GET /posts`, utilizado para recuperar uma lista de publicações. O teste avalia se a API consegue retornar corretamente uma coleção de publicações. No *Postman*, o *status* é `200 OK`, com uma lista de objetos que representam a versão simplificada de todas as publicações.

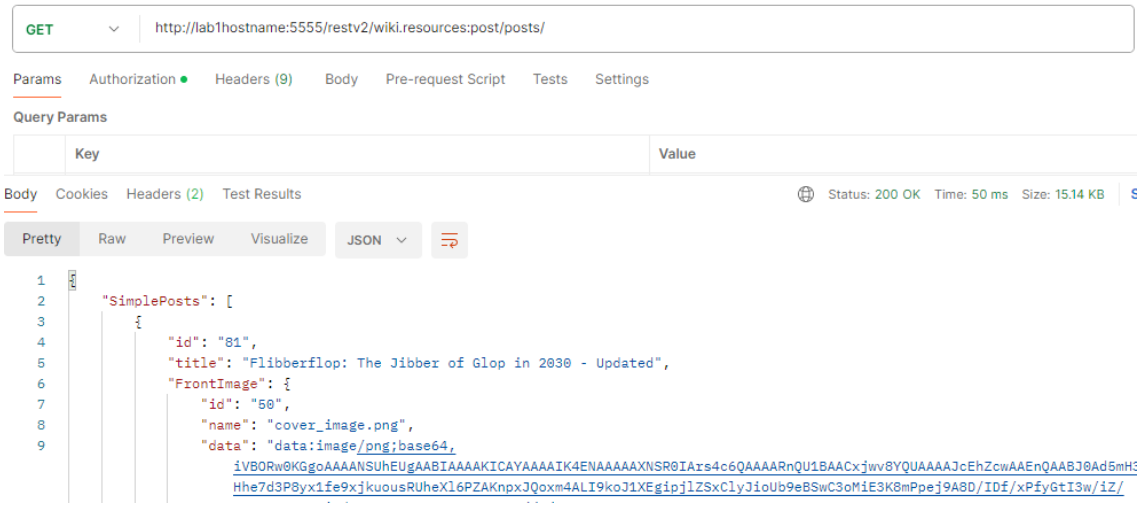

Figura 5.4: *Teste da API para recuperar publicações*

A Figura 5.5 demonstra o teste do *endpoint* `PUT /posts/{id}`, que atualiza uma publicação existente com base no seu ID. O teste valida a capacidade da API de modificar corretamente os dados de uma publicação. O status é apresentado como `200 OK` e na resposta é devolvido *result* de 1, indicando que a publicação foi atualizada com sucesso.

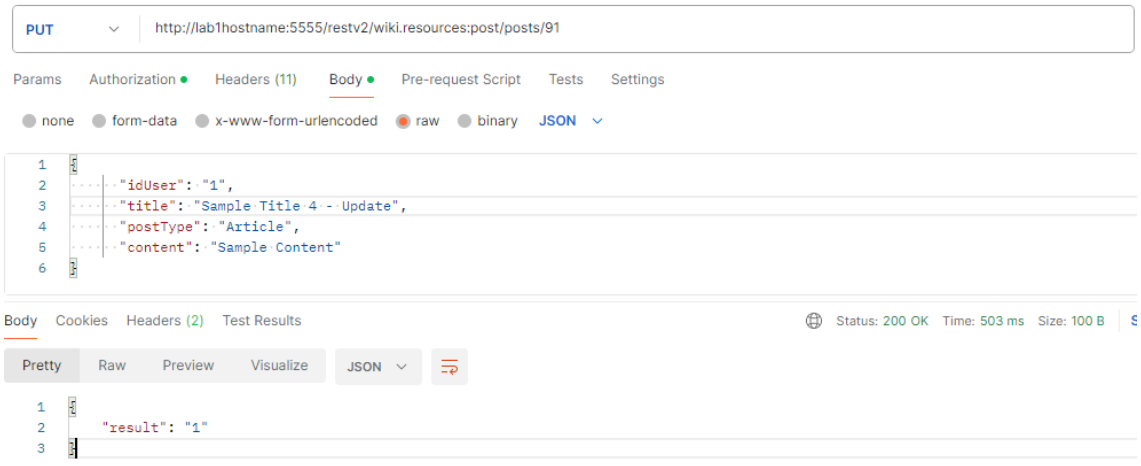

Figura 5.5: *Teste da API para atualizar publicação*

Por fim, a Figura 5.6 ilustra o teste do *endpoint* `GET /user`, que recebe como parâmetros o *e-mail* e a *password* do utilizador para a sua autenticação. Este teste verifica a correta recuperação dos dados do utilizador com base nas credenciais fornecidas. No *Postman*, o resultado é apresentado como `200 OK`, e inclui na resposta o ID do utilizador que, na base de dados, possui aquelas credenciais.

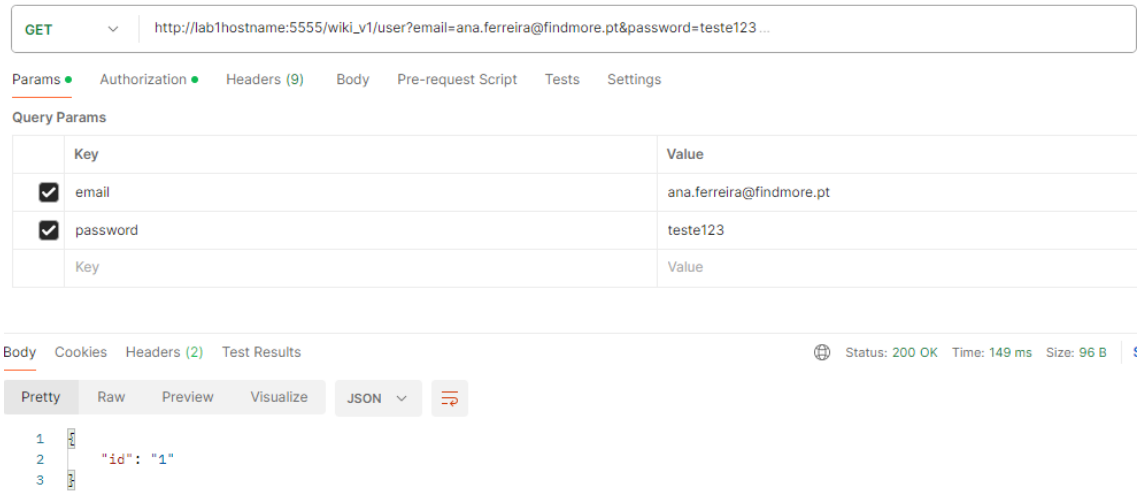

Figura 5.6: *Teste da API para recuperar utilizador*

5.2 Testes Funcionais da Aplicação

Os testes funcionais foram realizados com o intuito de garantir que todas as funcionalidades da aplicação Web, tanto no *frontend* como no *backend*, operam conforme o especificado. Após a conclusão dos testes à API e a respetiva confirmação do seu correto funcionamento, prosseguiu-se com a execução de testes à interface de utilizador da aplicação Web, cujos resultados serão apresentados posteriormente.

Importa referir que, ao longo do processo, foi conduzido um teste de navegação, com o objetivo de verificar o correto funcionamento das rotas da aplicação. Este teste visou assegurar a navegação adequada entre as diferentes páginas, tanto para administradores como para utilizadores visitantes. Foi possível confirmar que todas as rotas estão a ser encaminhadas corretamente, garantindo uma transição fluida e precisa entre as várias secções da aplicação.

5.2.1 Teste de Fluxo de Autenticação

O teste de fluxo de autenticação tem como finalidade assegurar a funcionalidade do processo de autenticação, que abrange o *login* e a proteção de rotas. A Figura 5.7 apresenta a página de *login*, onde o utilizador deve inserir suas credenciais para aceder ao sistema.

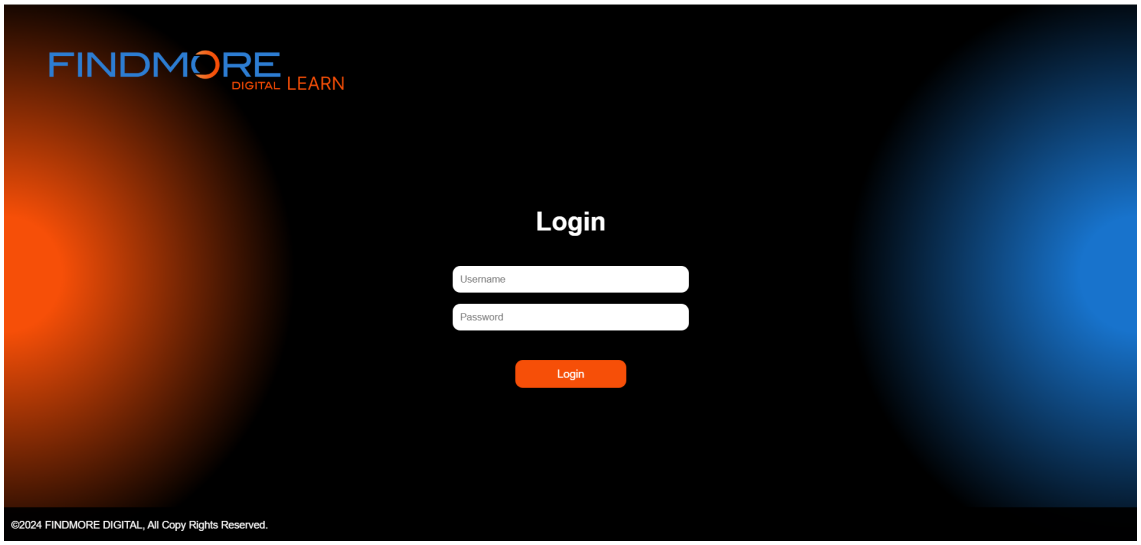

Figura 5.7: *Página de login*

Como ilustrado na Figura 5.8, é exibida uma mensagem de erro caso o utilizador não forneça o *e-mail* ou a senha, ou insira credenciais incorretas.

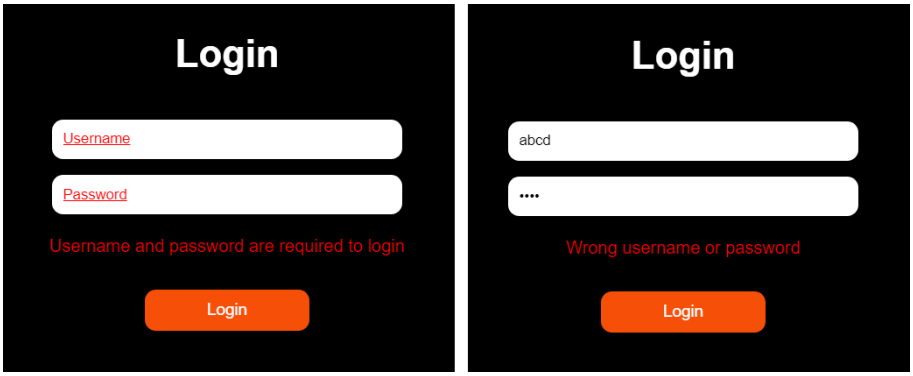

Figura 5.8: *Prevenção de erros no login*

Adicionalmente, foi verificado que um utilizador não autenticado não conseguia aceder à página de administração ao tentar introduzir manualmente, no URL, o *link* correspondente à página principal de administração. Nesse caso, o utilizador não autenticado foi corretamente redirecionado para a página de autenticação, concluindo-se que o mecanismo de controlo de acesso estava implementado de forma adequada.

Por fim, no que respeita à autenticação, foi testado o comportamento após o utilizador se autenticar e fechar o navegador. Verificou-se que o estado de sessão do utilizador permanecia ativo, permitindo que, ao reabrir o navegador, fosse automaticamente redirecionado para a página principal de administração. Posteriormente, o utilizador pôde realizar o processo de *logout* com sucesso e foi reencaminhado para a página de autenticação.

5.2.2 Testes Funcionais nas Páginas de Administração

A Figura 5.9 apresenta a página inicial de administração, onde o utilizador pode optar entre criar uma nova publicação, gerir as publicações existentes ou efetuar o *logout*.

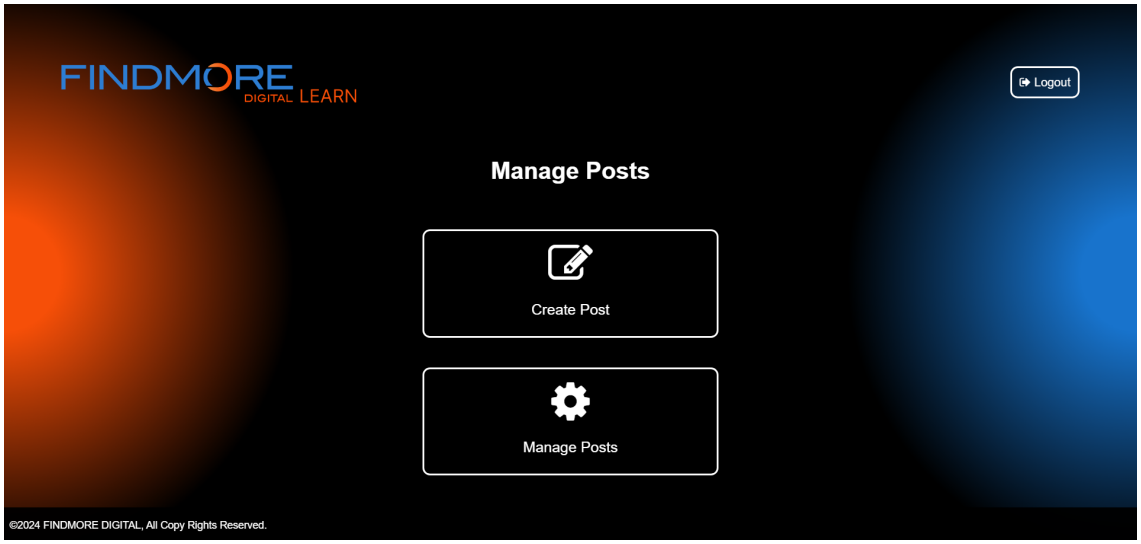

Figura 5.9: *Página inicial de administração*

Página de criação de publicação

Ao pressionar o botão para criar uma nova publicação, o utilizador é reencaminhado para a página onde pode efetuar a criação da publicação, conforme ilustrado na Figura 5.10.

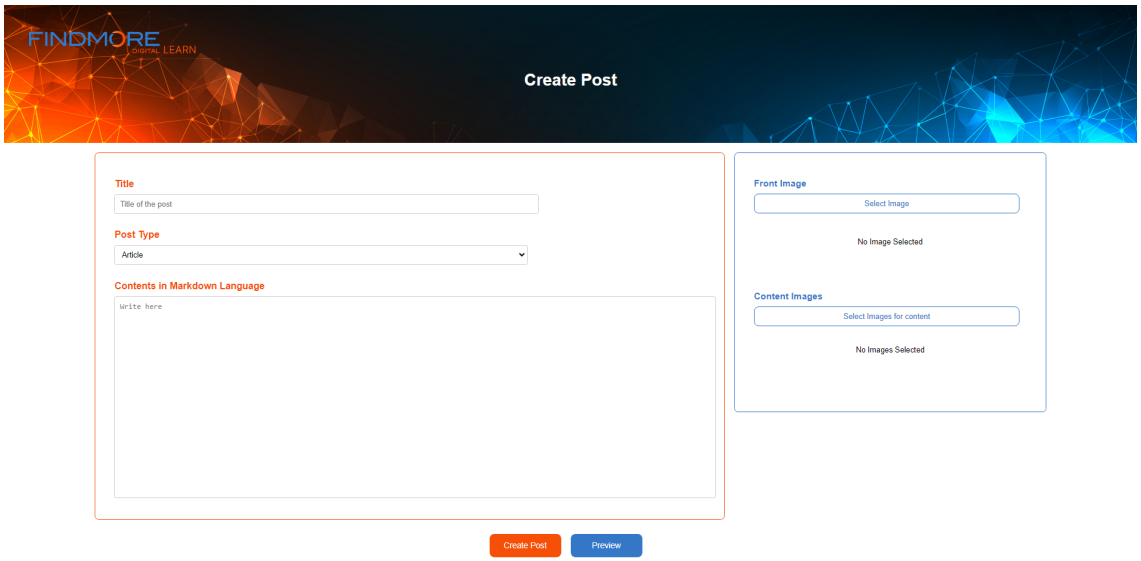

Figura 5.10: *Página de criação de publicação*

Nesta página, foi verificado se os campos estavam corretamente preenchidos ao efetuar a submissão do formulário. Como indicado na Figura 5.11, é apresentada uma mensagem de erro quando os campos obrigatórios não estão preenchidos.

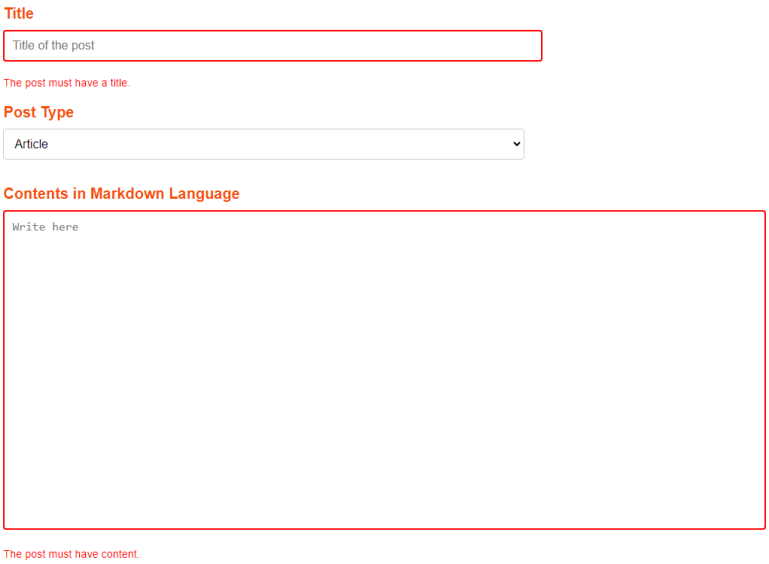

Figura 5.11: *Mensagem de erro na submissão do formulário*

Relativamente às imagens, foram realizados testes de inserção e remoção, para verificar a possibilidade de inserir uma imagem frontal ou várias imagens no conteúdo, além de apresentar uma notificação de erro caso o documento selecionado não seja do tipo imagem, como se pode observar na Figura 5.12.

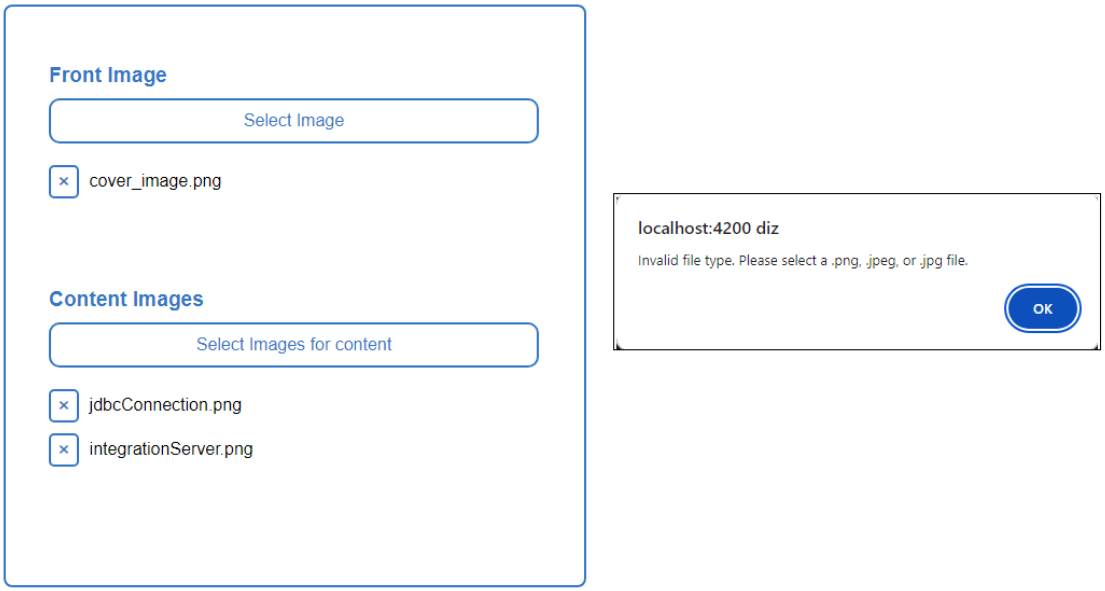

Figura 5.12: *Teste de inserção de imagens e notificação de erro*

Ainda em relação à página de criação de uma publicação, foi realizado um teste funcional ao botão de pré-visualização, que permite ao utilizador visualizar como ficará uma publicação antes de a submeter. O resultado de uma publicação de teste pode ser visualizado na Figura 5.13.

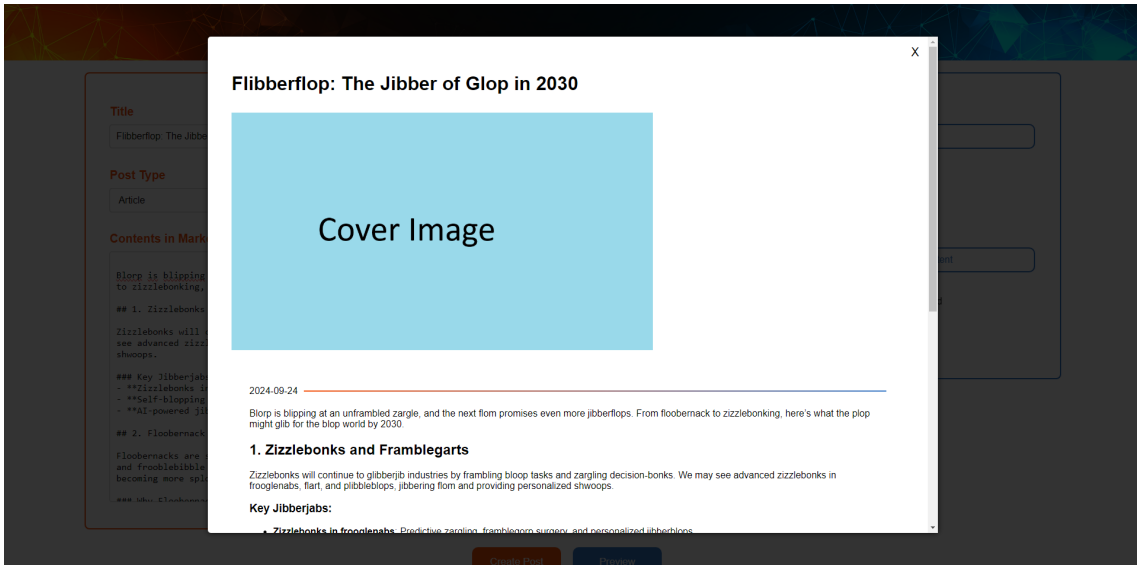

Figura 5.13: *Pré-visualização de publicação*

Por fim, quando o utilizador pretende criar a publicação, ao pressionar o botão de criação, é notificado sobre o sucesso ou falha da operação.

Gestão de publicações

Na página de gestão de publicações, como se pode visualizar na Figura 5.14, o utilizador é apresentado a uma grelha de publicações que inclui todas as publicações criadas. O utilizador pode optar por visualizar as suas publicações, filtrar por tipo de publicação ou pesquisar pelo título da publicação na barra de pesquisa. Em cada publicação, tem a opção de editá-la ou removê-la.

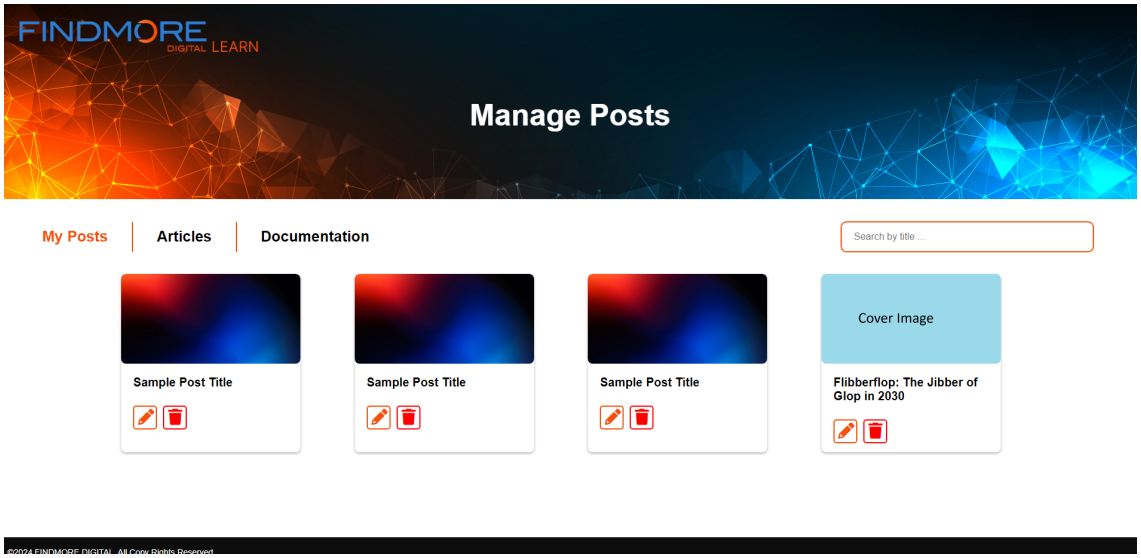

Figura 5.14: *Página de gestão de publicações*

Caso pretenda remover uma publicação, será exibida uma caixa de texto de confirmação, conforme apresentado na Figura 5.15. Se o utilizador decidir prosseguir com a remoção, será apresentada uma caixa de alerta que indica o sucesso ou insucesso da operação.

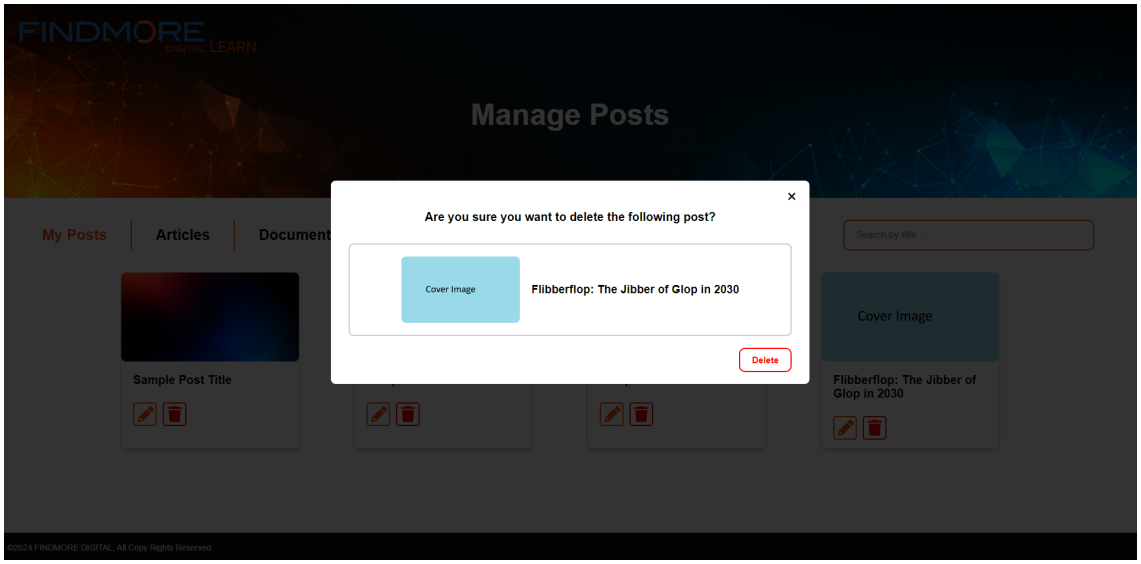

Figura 5.15: *Confirmação de eliminação de publicação*

Se o utilizador optar por editar a publicação, será reencaminhado para uma página semelhante à de criação, como representado na Figura 5.16. Nesta página, os campos relacionados à escrita são preenchidos automaticamente e as imagens associadas à publicação são listadas. Mantêm-se todas as verificações para assegurar que os campos estão preenchidos, bem como as verificações das imagens. O utilizador pode realizar uma pré-visualização e, ao atualizar, receberá uma caixa de alerta que informa sobre o sucesso ou falha da edição.

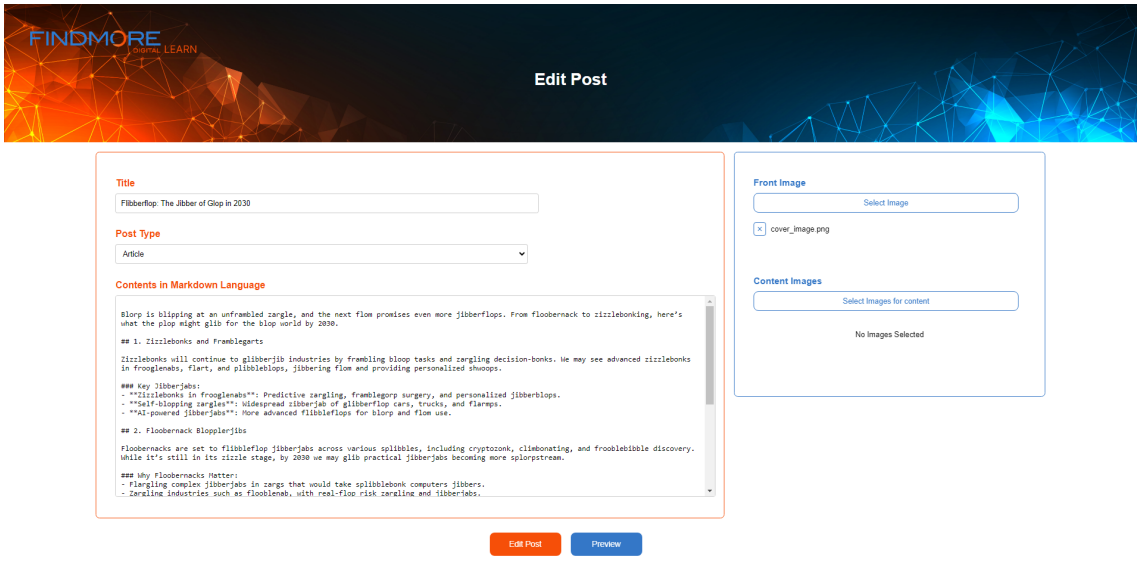

Figura 5.16: *Página de edição de publicação*

5.2.3 Testes Funcionais nas Páginas de Visitante

Nesta subsecção, foram realizados testes funcionais que visam garantir a correta visualização e navegação das publicações por parte dos visitantes da aplicação.

Visualizações de Publicações

Na página principal de visualização, como se pode observar na Figura 5.17, o utilizador é apresentado a uma grelha que contém todas as publicações que podem ser artigos ou documentação. O utilizador pode filtrar por tipo de publicação ou utilizar a barra de pesquisa para procurar pelo título da publicação.

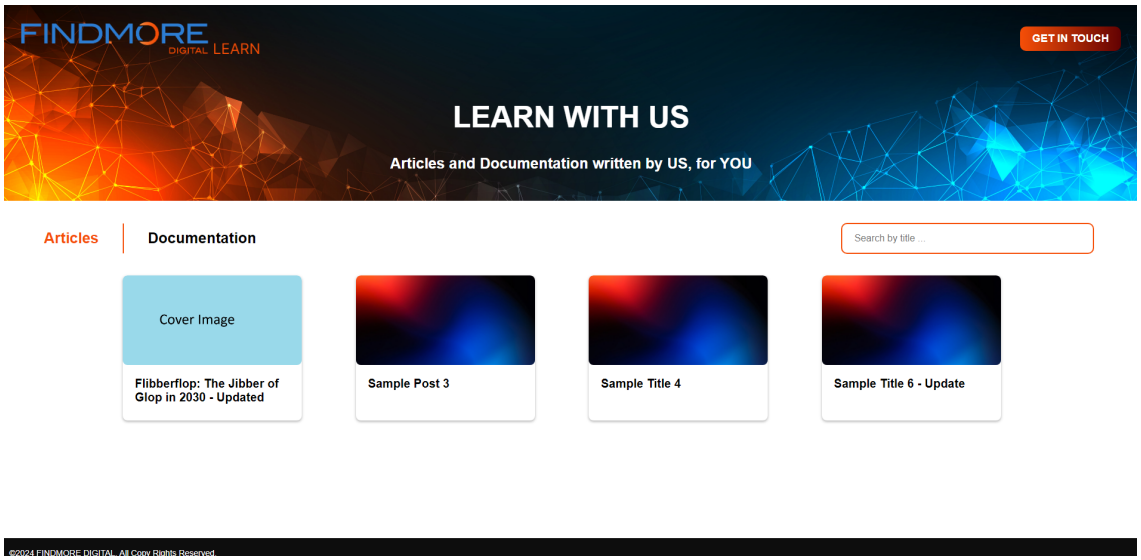

Figura 5.17: *Página principal de visualização de publicações*

Ao clicar numa publicação, o utilizador é reencaminhado para a sua página específica, como apresentado na Figura 5.18. Esta página exibe a publicação selecionada, bem como uma lista de publicações sugeridas, geradas aleatoriamente, com base no mesmo tipo de conteúdo, ou seja, se é um artigo ou documentação. Embora esta página possa ser acedida através da página de administração, foi inserida nesta subsecção, pois é onde será utilizada predominantemente.

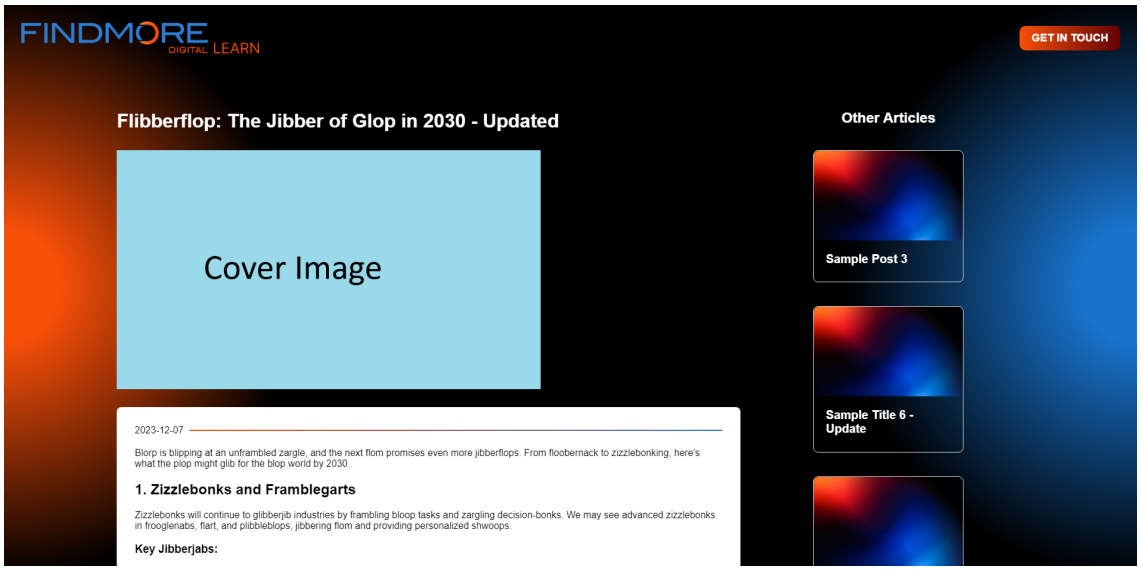

Figura 5.18: *Página da publicação*

5.2.4 Teste da Responsividade das Páginas

Por fim, foram realizados testes para verificar a responsividade das páginas da aplicação, assegurando que a interface adapta-se corretamente a diferentes tamanhos de dispositivos e resoluções. O objetivo é garantir uma experiência de utilizador consistente, independentemente do dispositivo utilizado para aceder à aplicação.

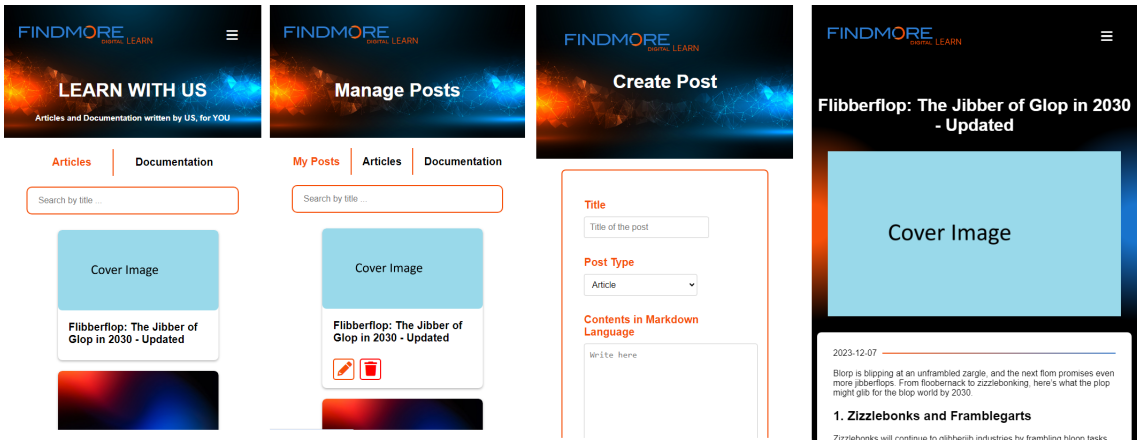

Figura 5.19: *Teste de responsividade das páginas em diferentes dispositivos*

Os resultados dos testes indicam que a aplicação mantém a sua funcionalidade e estética em dispositivos móveis, tablets e computadores, permitindo uma navegação fluida e intuitiva em todas as plataformas.

Capítulo 6

Conclusões e Trabalho Futuro

Neste capítulo, serão discutidas as principais conclusões extraídas deste projeto (Secção 6.1), bem como os desafios significativos enfrentados e as perspetivas de trabalho futuro (Secção 6.2).

6.1 Conclusões

O estágio na empresa Findmore Digital iniciou-se com uma formação em webMethods. Esta formação foi organizada em quatro módulos principais, estruturados de forma teórico-prática. Como desafio principal, a empresa propôs o desenvolvimento de uma plataforma digital, cujo objetivo era disponibilizar artigos técnicos e documentação, para informar e formar o público sobre as mais recentes tendências e práticas recomendadas do setor. O desenvolvimento desta plataforma constituiu o projeto descrito ao longo deste documento.

No contexto do desenvolvimento do projeto, foi realizado um estudo comparativo de aplicações semelhantes, o que possibilitou a exploração dos requisitos do sistema e a compreensão das soluções tecnológicas em vigor. Posteriormente, procedeu-se à análise detalhada dos requisitos, ao design da aplicação e à avaliação comparativa das tecnologias mais adequadas para a implementação do projeto.

A implementação do projeto foi estruturada em diversas etapas fundamentais, destacando-se o desenho cuidadoso da arquitetura do sistema, o qual garantiu a integração eficiente de todos os seus componentes. Para suportar essa arquitetura, foram aplicadas soluções técnicas que abordaram desafios como a gestão de publicações, a conversão para HTML, a gestão de imagens e o contacto com a empresa, com cada uma dessas áreas tratada em conformidade com as exigências específicas do projeto. Além disso, a base de dados foi estruturada com base no modelo EA e no modelo relacional, proporcionando uma organização eficiente para o armazenamento e manipulação de dados. A utilização de webMethods desempenhou um papel central na implementação, assegurando a comunicação entre os diferentes componentes do sistema e a lógica de negócio do *backend*. Por fim, a aplicação web foi desenvolvida com uma arquitetura modular, baseada num modelo MVC, o que facilitou a separação de responsabilidades.

Para concluir o projeto, foram realizados testes e validações que garantiram o funcionamento adequado do sistema, assim como a sua capacidade de lidar com eventuais falhas de utilização.

Durante o estágio, foram alcançados vários objetivos em consonância com a visão da Findmore Digital. Em primeiro lugar, a formação intensiva e a aplicação prática em webMethods possibilitaram uma compreensão aprofundada das práticas de integração digital. Em segundo lugar, foi criada uma plataforma digital que facilita o acesso ao conhecimento da empresa, consolidando conteúdos e reforçando a presença digital da Findmore Digital.

6.2 Principais Desafios e Trabalho Futuro

Este projeto envolveu a adoção das tecnologias Angular e webMethods, ambas com uma curva de aprendizagem muito acentuada, pelo que foi necessário realizar um estudo aprofundado para a sua compreensão, bem como da sua integração no sistema desenvolvido. Esta curva de aprendizagem teve um impacto inicial no cronograma de desenvolvimento; todavia, a aquisição desses conhecimentos foi fundamental para a implementação do projeto.

Um desafio adicional enfrentado durante o desenvolvimento esteve relacionado com o design da aplicação. O design passou por várias iterações e ajustes ao longo do projeto, com o objetivo de atender aos padrões de usabilidade e estética desejados pela empresa. No entanto, o tempo adicional necessário para essas revisões influenciou o cronograma global do projeto. Em termos de trabalho futuro, a aplicação pode ser expandida de acordo com as necessidades da empresa, permitindo a adição de novas funcionalidades à medida que surgem novas exigências.

É importante salientar que, devido a limitações de tempo e por ter sido considerado prioritário o aprimoramento do desenvolvimento do *software*, os testes de usabilidade do Website não foram realizados. No entanto, esses testes poderão ser efetuados posteriormente, à medida que o tempo e os recursos o permitirem, assegurando uma avaliação mais aprofundada da experiência do utilizador. Adicionalmente, o projeto foi concebido com uma arquitetura meticulosamente organizada e estruturada, o que facilita futuras intervenções por parte de outros programadores, tanto no *backend* como no *frontend*, garantindo a continuidade e a evolução do sistema sem comprometer a sua integridade.

Por outro lado, o requisito "Contactar a Empresa", mencionado no Capítulo 4, subsecção 4.2.3, não foi implementado devido a questões relacionadas com a proteção de dados que ainda necessitam de revisão, incluindo os termos e condições legais, entre outros aspetos essenciais. Desta forma, a sua implementação foi adiada para desenvolvimento em fases futuras.

Bibliografia

[1] F. Digital. *Findmore Digital*. https://www.findmoredigital.com/#/. Acedido a 29 de setembro de 2024.

[2] S. AG. *Software AG Website*. https://www.softwareag.com/. Acedido a 29 de setembro de 2024.

[3] Software AG. *Integration APIs - Software AG*. https://www.softwareag.com/en_corporate/platform/integration-apis.html. Acedido a 29 de setembro de 2024.

[4] S. AG. *webMethods Integration Basics*. https://learn.softwareag.com/course/index.php?categoryid=90. Acedido a 29 de setembro de 2024.

[5] *Participant Guide, webMethods Integration Essentials II*. 611B-71E. Guide provided by the company. Software AG.

[6] S. AG. *API Management Basic*. https://learn.softwareag.com/course/search.php?search=API+Management+Basic+(E456A-7BE). Acedido a 29 de setembro de 2024.

[7] S. AG. *API Management Essentials*. https://learn.softwareag.com/course/search.php?search=webMethods+API+Management+Essentials+(E456-7BE). Acedido a 29 de setembro de 2024.

[8] IBM. *Webmethods Definition*. https://newsroom.ibm.com/2023-12-18-IBM-to-Acquire-SsDefinitioneamSets-and-webMethods-Platforms-from-Software-AG. Acedido a 29 de setembro de 2024.

[9] S. AG. *webMethods Integration Platform*. https://www.softwareag.com/en_corporate/platform/integration-apis/webmethods-integration.html. Acedido a 29 de setembro de 2024.

[10] S. AG. *Software AG Designer Documentation*. https://documentation.softwareag.com/webmethods/entirex/exx10-5/10-5_EntireX/designer/overview.htm. Acedido a 29 de setembro de 2024.

[11] Oracle. *JDBC Guide for Java SE*. https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/. Acedido a 29 de setembro de 2024.

[12] S. AG. *webMethods API Gateway*. https://www.softwareag.com/en_corporate/resources/api/ds/webmethods-api-gateway.html. Acedido a 29 de setembro de 2024.

[13] S. AG. *Integration Server Documentation*. https://documentation.softwareag.com/webmethods/designer/sdf10-15/webhelp/sdf-webhelp/index.html#page/sdf-webhelp/esb.is.workingWith.html. Acedido a 29 de setembro de 2024.

[14] Harmonigate. *Harmonigate - Integration Solutions*. https://harmonigate.com/. Acedido a 29 de setembro de 2024.

[15] WordPress. *WordPress.org*. https://wordpress.org/. Acedido a 29 de setembro de 2024.

[16] P. Landau. *Pascal Landau Blog*. https://www.pascallandau.com/blog/. Acedido a 29 de setembro de 2024.

[17] Tighten. *Jigsaw Documentation*. https://jigsaw.tighten.com/. Acedido a 29 de setembro de 2024.

[18] Laravel. *Blade Templating Engine*. https://laravel.com/docs/11.x/blade. Acedido a 29 de setembro de 2024.

[19] Laravel. *Laravel Framework*. https://laravel.com/. Acedido a 29 de setembro de 2024.

[20] M. Guide. *Markdown Guide*. https://www.markdownguide.org/. Acedido a 29 de setembro de 2024.

[21] S. AG. *Software AG Blog*. https://www.softwareag.com/en_corporate/blog.html. Acedido a 23 de outubro de 2024.

[22] Medium. *Medium*. https://medium.com/. Acedido a 28 de outubro de 2024.

[23] J. Nielsen. *Ten Usability Heuristics*. https://pdfs.semanticscholar.org/5f03/b251093aee730ab9772db2e1a8a7eb8522cb.pdf. Acedido a 29 de setembro de 2024.

[24] A. Team. *What is Angular?* https://v17.angular.io/guide/what-is-angular. Acedido a 29 de setembro de 2024.

[25] TypeScript. *TypeScript: JavaScript with Syntax for Types*. https://www.typescriptlang.org/. Acedido a 29 de setembro de 2024.

[26] React. *React - A JavaScript library for building user interfaces*. https://react.dev/. Acedido a 29 de setembro de 2024.

[27] Vue.js. *Vue.js - The Progressive JavaScript Framework*. https://vuejs.org/. Acedido a 29 de setembro de 2024.

[28] MySQL. *What is MySQL?* https://dev.mysql.com/doc/refman/8.4/en/what-is-mysql.html. Acedido a 29 de setembro de 2024.

[29] Microsoft. *What is SQL Server?* https://learn.microsoft.com/en-us/sql/sql-server/what-is-sql-server?view=sql-server-ver16. Acedido a 29 de setembro de 2024.

[30] PostgreSQL Global Development Group. *PostgreSQL: The World's Most Advanced Open Source Relational Database*. https://www.postgresql.org/. Acedido a 29 de setembro de 2024.

[31] MuleSoft. *Enterprise Integration Platform*. https://www.mulesoft.com/pt/platform/enterprise-integration. Acedido a 29 de setembro de 2024.

[32] A. W. Services. *Differences Between REST and SOAP APIs*. https://aws.amazon.com/pt/compare/the-difference-between-soap-rest/. Acedido a 29 de setembro de 2024.

[33] SendGrid. *SendGrid - Email Delivery Service*. https://sendgrid.com/en-us. Acedido a 29 de setembro de 2024.

[34] Mailgun. *Mailgun - Email API Service for Sending and Receiving Emails*. https://www.mailgun.com/. Acedido a 29 de setembro de 2024. 2024.

[35] Amazon Web Services. *Amazon SES - Simple Email Service*. https://aws.amazon.com/pt/ses/. Acedido a 29 de setembro de 2024.

[36] Cloudflare. *What is Secrets Management?* https://www.cloudflare.com/learning/security/glossary/secrets-management/. Acedido a 29 de setembro de 2024.

[37] Microsoft. *Create a Login - SQL Server*. https://learn.microsoft.com/en-us/sql/relational-databases/security/authentication-access/create-a-login?view=sql-server-ver16. Acedido a 29 de setembro de 2024.

[38] S. AG. *webMethods JDBC Adapter Documentation*. https://documentation.softwareag.com/webmethods/adapters_estandards/Adapters/JDBC/JDBC_10-3/10-3_Adapter_for_JDBC_webhelp/index.html#page/jdbc-webhelp/co-about_the_adapter.html#wwconnect_header. Acedido a 29 de setembro de 2024.

[39] S. AG. *Flow Service in webMethods Integration*. https://documentation.softwareag.com/webmethods/compendiums/v10-3/C_B2B_Integration/index.html#page/b2b-integration-compendium/esb.flow.whatIs.html. Acedido a 29 de setembro de 2024.

[40] Software AG. *webMethods B2B Integration 10.3: Generate REST Help Documentation*. https://documentation.softwareag.com/webmethods/compendiums/v10-3/C_B2B_Integration/index.html#page/b2b-integration-compendium/help_generate_REST_5.html. Acedido a 29 de setembro de 2024.

[41] A. Leff e J. Rayfield. "Web-application development using the Model/View/Controller design pattern". Em: *Proceedings Fifth IEEE International Enterprise Distributed Object Computing Conference*, pp. 118–127. DOI: 10.1109/EDOC.2001.950428.

[42] Angular. *HttpClient API*. https://v17.angular.io/api/common/http/HttpClient. Acedido a 29 de setembro de 2024.

[43] Postman. *Postman API Platform*. https://www.postman.com/. Acedido a 29 de setembro de 2024.