

ISPGAYA

instituto superior politécnico

Licenciatura em Engenharia Informática
Projeto de Engenharia Informática em Contexto Empresarial
2023-2024

Pedro Miguel Tomé Rego

RELATÓRIO DE ESTÁGIO CURRICULAR **DESENVOLVIMENTO SOFTWARE .NET E REACT**

**Relatório de estágio orientado pelo Professor Doutor Mário Jorge Dias Lousã e
apresentada à Escola Superior de Ciência e Tecnologia**

Vila Nova de Gaia, julho de 2024

Dedicatória

Dedico este relatório a todos os que contribuíram para a
minha jornada académica. À minha família, amigos,
professores e colegas de trabalho.

Agradecimentos

O presente relatório de estágio não seria possível sem a ajuda de várias pessoas que de uma maneira ou de outra contribuíram para a realização do meu estágio curricular.

Quero, em primeiro lugar, agradecer à ITSector pela oportunidade que me deu durante este período de aprender e evoluir, agradecer também a todos os colaboradores que tive o prazer de conhecer, em especial o meu orientador Carlos Afonso, por todos os conselhos, paciência e disponibilidade que demonstrou para comigo desde o início deste processo.

À instituição do ISPGAYA, a todos os professores que me lecionaram unidades curriculares, em especial ao meu orientador de estágio Eng. Mário Lousã e coordenador de curso Eng. Jorge Simões por todo o conhecimento transmitido ao longo destes anos de vida académica.

Por fim, agradecer à minha família por todo o carinho e amor que me permitem lutar pelos meus sonhos.

Resumo

O presente relatório tem como principal objetivo caracterizar as atividades desenvolvidas no contexto do estágio curricular na ITSector, descrevendo o trabalho realizado durante a duração do mesmo. O estágio iniciou-se com a realização de uma academia, no caso a Academia .Net com duração de três semanas, para obter conhecimentos e experiência para uma maior facilidade de integração nos projetos futuros.

Após a conclusão da Academia .Net iniciaram-se os trabalhos num projeto ligado a um banco internacional e posteriormente um projeto interno da empresa que será descrito e desenvolvido mais à frente neste documento.

Serve o presente relatório para descrever as ferramentas e tecnologias usadas durante o processo de estágio, bem como para apresentar a informação necessária para o desenvolvimento dos projetos e relatar a experiência vivenciada, culminando com a descrição das competências adquiridas neste período.

Neste documento encontram-se também informações relevantes sobre a ITSector, as principais atividades desenvolvidas e um pouco da história desta organização. No final, no âmbito das conclusões, é realizada uma reflexão crítica dividida por aprendizagens e dificuldades enfrentadas durante o estágio, destacando-se as competências e as melhorias implementadas nos projetos.

Abstract

The present report aims to characterize the activities carried out within the context of the curricular internship at ITSector, describing the work performed during its duration. The internship began with participation in an academy, specifically the .Net Academy lasting three weeks, aimed at acquiring knowledge and experience to facilitate integration into future projects.

Following the conclusion of the .Net Academy, work commenced on a project connected to an international bank, followed by an internal company project, which will be further described and developed in this document.

This report serves to describe the tools and technologies used during the internship process, as well as to provide the necessary information for project development and report on the experience gained, culminating with the description of the competencies acquired during this period.

This document also includes relevant information about ITSector, the main activities carried out, and a brief history of the organization. Lastly, the conclusions section features a critical reflection divided into learnings and challenges encountered during the internship, highlighting the skills developed and improvements implemented in the projects.

Lista de abreviaturas e siglas

SO	Sistema Operativo
LINQ	Language Integrated Query.
DP's	Design Patterns.
SQL	Structured Query Language.
BD	Base Dados.
CSS	Cascading Style Sheets.
HTML	Hypertext Markup Language.
CRUD	Create, Read, Update, Delete.
UI	User Interface.
MVC	Model View Controller.
SOAP	Simple Object Access Protocol.
XML	eXtensible Markup Language.
JWT	JSON Web Tokens.
HTTP	HyperText Transfer Protocol.
API	Application Programming Interface.
JSON	JavaScript Object Notation.

Índice

Dedicatória.....	ii
Agradecimentos	iii
Resumo	iv
Abstract.....	v
Lista de abreviaturas e siglas	vi
Índice	vii
Introdução	xi
1. Contexto empresarial – a ITSector.....	12
2. Academia .NET.....	13
2.1. Enquadramento	13
2.2 Trabalhos desenvolvidos.....	13
2.2.1 C#	13
2.2.2 <i>Design Patterns</i>	14
2.2.3 <i>SQL</i>	15
2.2.4 <i>Bootstrap</i>	16
2.2.5 <i>ASP.NET Web Forms</i>	16
2.2.6 <i>ASP.NET MVC</i>	17
2.2.7 <i>Web Services (WCF)</i>	18
2.3 Projeto final	19
2.3.1 <i>Enquadramento do projeto final</i>	19
2.3.2 <i>Requisitos do projeto final</i>	20
3. Projeto Document Digital Service	21
3.1 Contexto do projeto	21
3.2 Tecnologias Utilizadas.....	22
3.2.1 <i>Visual Studio Code</i>	22
3.2.2 <i>Azure DevOps</i>	23
3.2.3 <i>React</i>	24
3.2.4 <i>Axios</i>	24
3.3 Planeamento do projeto	26
3.3.1 <i>Requisitos funcionais</i>	26
3.3.2 <i>Requisitos não funcionais</i>	27

3.4 Página <i>login</i>	27
3.5 Ligação ao <i>backend</i>	30
3.6 Access tokens.....	32
3.7 User Roles.....	34
3.8 NSWAG.....	35
Conclusão.....	37
Bibliografia	38

Lista de Figuras

Figura 1. Layout página de login	28
Figura 2. Tentativa de login com credenciais erradas.....	29
Figura 3. Página Principal.....	29
Figura 4. <i>Logout</i>	30
Figura 5. Configuração Axis.....	31
Figura 6. Metodo Post com Axios	31
Figura 7. Armazenar tokens.....	32
Figura 8. Token no Navegador	33
Figura 9 . Configuração User Roles.....	34
Figura 10. Código NSwag.....	36
Figura 11. Implementação código NSwag.....	36

Lista de Tabelas

Tabela 1 - Requisitos Funcionais	26
--	----

Introdução

O presente relatório descreve o estágio curricular realizado na empresa ITSector, uma empresa de desenvolvimento de software especializada em soluções tecnológicas inovadoras no setor financeiro. Este estágio está integrado na unidade curricular de Projeto de Engenharia Informática em Contexto Empresarial do curso de Licenciatura em Engenharia Informática.

O estágio teve início com participação na Academia .Net, com duração de três semanas, onde foram abordadas diversas tecnologias e ferramentas para o desenvolvimento de software. Esta formação teve como objetivo preparar os estagiários para a integração eficiente em projetos futuros da empresa.

Os principais objetivos deste estágio foram:

- Adquirir e aplicar conhecimentos em tecnologias .Net e React
- Desenvolver projetos práticos desde aplicações de consola em C# até sistemas web complexos usando ASP.NET e React
- Implementar boas práticas de desenvolvimento de software
- Adquirir conhecimento e experiência do mercado de trabalho.

Este documento está organizado em três secções principais:

- Secção 1 – ITSector: contextualiza a empresa, a sua história, valores e áreas de atuação
- Secção 2 – Academia .Net: descreve as atividades e projetos desenvolvidos durante a formação inicial, abordando diversas tecnologias
- Secção 3 – Document Digital Service: apresenta o projeto interno desenvolvido durante o estágio, focado na criação de uma aplicação frontend em React com autenticação de utilizadores e gestão de permissões.

1.Contexto empresarial – a ITSector

Fundada em 2005, a ITSector é uma empresa de desenvolvimento de software especializada em soluções tecnológicas inovadoras no setor financeiro, criada com o objetivo de oferecer ao mercado soluções de sistemas de informação de alto nível.

Tendo como base os dados recolhidos a partir do site da empresa (www.itsector.pt), a grande valência da ITSector são os serviços de Outsourcing, que a partir de localizações estratégicas em Portugal, desenvolve software para mais de 20 países, como por exemplo: Angola, Moçambique, Macau, África do Sul, Canadá, Reino Unido, França, entre outros.

Na fundação da empresa estão os valores que todos os colaboradores devem seguir, sendo esses a integridade, a responsabilidade, o compromisso, a inovação, o dinamismo e a ambição, sendo que o principal valor é a paixão pelo que se faz.

A sede da ITSector é o seu escritório no Porto, onde se encontra a maioria dos seus colaboradores, sendo que tem ainda o seu espaço em Lisboa, Braga, Aveiro, Bragança e Castelo Branco conseguindo assim atrair talentos de qualquer parte do país.

Estruturalmente, a ITSector divide os colaboradores pelas seguintes áreas:

- Área da Inovação
- Área da Tecnologia
- Área de Segurança TI
- Área de Finanças
- Área Legal
- Área de Aquisição de talento.

2. Academia .NET

2.1 Enquadramento

Como referido anteriormente, o presente estágio iniciou-se com uma academia cujo objetivo passava pela integração e formação de novos colaboradores às tecnologias .NET.

Segundo a Amazon Web Services(n.d.a),” .NET é uma plataforma de código aberto para criação de aplicações de desktop, web e móveis que podem ser executadas nativamente em qualquer Sistema Operativo (SO)”. O sistema .NET inclui ferramentas, bibliotecas e linguagens que oferecem suporte ao desenvolvimento de software moderno, escalável e de alta performance.

Nesta academia foram introduzidos conceitos básicos como tipos primitivos, estruturas de controle, funções, classes e objetos, assim como conceitos como herança, polimorfismo e encapsulamento. Os estagiários aprenderam também sobre *frameworks* de desenvolvimento e ferramentas de colaboração e manutenção de código. Esta formação foi orientada por colaboradores da ITSector e funcionou da seguinte maneira:

- Self training através de conteúdos disponibilizados pela Microsoft – 75%
- Acompanhamento pelo tutor dos trabalhos em curso – 15%
- Exposições teórico/práticas – 5%
- Reuniões de acompanhamento – 5%

2.2 Trabalhos Desenvolvidos

2.2.1 C#

A primeira tarefa pedida pelos orientadores foi o desenvolvimento de uma aplicação de consola em C#, cujo objetivo era obter uma lista de valores introduzidos pelo utilizador, onde fosse possível realizar as seguintes ações:

- Filtrar números pares

- Filtrar números ímpares
- Obter os três primeiros valores
- Obter o maior valor
- Inverter a ordem
- Ordenar de forma ascendente.

Este projeto serviu como introdução à linguagem *C#* e à tecnologia Language Integrated Query (LINQ). Segundo a Microsoft (2024), ”*C#* é uma linguagem versátil e poderosa, amplamente utilizada no desenvolvimento de aplicações para a plataforma .NET”. A LINQ permite realizar consultas a coleções de dados diretamente no código *C#*, facilitando assim a manipulação de dados, tornando as operações como filtragem, projeção e agrupamento eficientes, melhorando a produtividade e legibilidade do código.

2.2.2 Design Patterns

O segundo projeto consistiu numa introdução aos *Design Patterns* (DP), ou padrões de design. Esses padrões são soluções reutilizáveis para problemas comuns encontrados no desenvolvimento de software. Utilizar *DP's* melhora a estrutura do código, tornando-o mais modular, flexível e fácil de manter.

O objetivo deste exercício era desenvolver uma aplicação que simulasse o funcionamento das transações bancárias realizadas entre clientes e bancos, tendo em conta as seguintes necessidades:

- Desenvolver uma fábrica de construção de instâncias de sistemas interbancários, usando o *Factory Method*
- Criar várias entidades responsáveis por transferências e pagamentos com o padrão *Singleton* que gera entidades não replicáveis
- Desenvolver uma interface de comunicação entre os clientes e os serviços através do padrão *Facade*

- Uso do *Observer* para que exista uma entidade responsável pelas comunicações entre clientes e entidades bancárias.

2.2.3 SQL

O terceiro projeto foi uma oportunidade para aprofundar conhecimentos sobre *Structured Query Language* (SQL), mais especificamente em SQL Server. Segundo a Amazon Web Services(n.d.b), “SQL é uma linguagem de programação para armazenar e processar informações num banco de dados relacional.Você pode usar intruções SQL para armazenar, atualizar, remover, pesquisar e recuperar informações da base de dados.Também pode usar SQL para manter e otimizar a performance da base de dados.”

O SQL Server é um sistema de gestão de BD relacional, desenvolvido pela Microsoft que oferece um conjunto de ferramentas para armazenar e processar dados, facilitando a gestão de grandes volumes de informação.

O exercício proposto para este tema era executar *queries* utilizando tabelas de uma BD de teste da empresa. Para o efeito, usou-se o Microsoft Management Studio (MMS2022) e as *queries* a implementar eram as seguintes:

- Devolver o número total de processos
- Agrupar processos por tipo
- Ordenar processos de forma crescente e decrescente pela data de criação
- Devolver o nome dos clientes com processos abertos
- Devolver uma lista de processos que contenham as letras “VE”
- Devolver uma lista de processos que contenham o número de processo “28” e “31”
- Criar um Stored Procedure que devolva todos os processos existentes.

2.2.4 Bootstrap

Após a introdução a ferramentas *backend*, o foco virou-se para *frontend*, começando por criar duas páginas web, conectadas, utilizando a ferramenta Bootstrap. “Bootstrap é um framework front-end que fornece estruturas de CSS para a criação de sites e aplicações responsivas de forma rápida e simples. Além disso, pode lidar com sites de desktop e páginas de dispositivos móveis da mesma forma.”, segundo a Alura(2023).

Neste projeto, foi possível explorar alguns dos recursos fornecidos pelo Bootstrap, como por exemplo o sistema de *grid*, que permite organizar o layout da página de forma que se adapte ao tamanho do ecrã. Foi também possível usar componentes de estilização, como, por exemplo, botões, *navbars*, *cards* e *modals* para construir uma interface mais limpa e funcional sem a necessidade de escrever CSS personalizado de raiz.

Este projeto serviu para relembrar linguagens como HTML e CSS, como também para mostrar que ferramentas como o Bootstrap podem aumentar a produtividade e ao mesmo tempo garantir consistência visual aos projetos web.

2.2.5 ASP.NET Web Forms

Este projeto em ASP.NET Web Forms consistia em desenvolver uma aplicação *Create,Read,Update,Delete* (CRUD). A ASP.NET Web Forms é uma tecnologia da Microsoft (2024, June 6) que segundo a mesma, “facilita a criação de aplicações web dinâmicas permitindo o uso de controles de servidor para gerar HTML dinamicamente”.

Assim, foi criada uma tabela interativa onde os utilizadores podiam inserir, atualizar, excluir e visualizar dados. Os conceitos usados no desenvolvimento desta aplicação foram:

- GridView – Controle que exibir dados em formato de tabela e fornece a edição e exclusão de registos diretamente na interface
- SqlDataSource – Controle que conecta a tabela à BD SQLServer. Facilita o CRUD pois conecta-se diretamente com o GridView

- Controles de Eventos – Uso de Textboxes, Dropdownlists e Botões para melhorar a UI.

Este projeto foi importante, pois o ASP.NET é uma ferramenta antiga, mas que continua a ser usada por muitas empresas devido à possibilidade de manipulação de dados de maneira eficiente e segura.

2.2.6 ASP.NET MVC

Para o projeto seguinte o enunciado era o mesmo, apenas mudou a arquitetura onde se ia desenvolver a aplicação. O ASP.NET MVC (Model,View,Controller) é uma *framework* da Microsoft que separa a aplicação em três componentes principais: Modelo, Visão e Controlador promovendo uma estrutura organizada e facilitando o desenvolvimento, teste e manutenção do código.

A estrutura da aplicação foi a seguinte:

- Model – classes do modelo que representam a estrutura dos dados, contendo lógica da aplicação
- Controller – desenvolvimento de controladores para gerir a lógica e responder às solicitações do utilizador.
- View – responsável pela renderização da *UI*, mais concretamente a exibição dos dados do modelo e fornecer formulários para entrada de dados.

De referir que em ambos o projeto foi usada a ferramenta Entity Framework para o mapeamento de objetos da aplicação para as tabelas da BD e a execução de consultas de forma eficiente.

Este projeto proporcionou uma compreensão aprofundada de como criar aplicações web robustas com a arquitetura MVC. A separação entre modelo, visão e controlador promoveu uma melhor organização e leitura do código e facilitou a manutenção do mesmo.

2.2.7 Web Services (WCF)

Antes de avançar para o projeto final, foi proposta a criação de uma aplicação que convertesse unidades de temperatura (Celsius, Fahrenheit, Kelvin) utilizando um serviço SOAP. A Simple Object Access Protocol (SOAP) “é um formato de mensagem XML usado nas interações de serviços web”, segundo a IBM(2022, August 24), que permite a troca de informações estruturadas entre sistemas distribuídos.

Os passos usados para completar a tarefa foram os seguintes, tendo em conta que o serviço foi fornecido pelos orientadores:

- Consumir serviço, usando ferramentas como WCF (Windows Communication Foundation) para gerar *proxies* de cliente;
- Implementação de métodos para realizar as conversões de temperatura. Cada método recebe o valor de temperatura e retorna o valor convertido na unidade desejada
- Desenvolver UI onde os utilizadores podiam inserir os valores e a temperatura e posteriormente a unidade na qual pretendiam fazer a conversão. A UI foi desenvolvida com a tecnologia ASP.NET Web Forms;

Este projeto foi importante pois permitiu conhecer a maneira como sistemas diferentes se comunicam. A utilização do SOAP garante uma comunicação robusta e segura baseada em padrões.

2.3 Projeto final

2.3.1 Enquadramento do projeto final

O projeto final que coincidia com o final da academia, tinha como objetivo consolidar conhecimentos adquiridos até ao momento nas tecnologias de C#, ASP.NET, JavaScript, Web API's, BootStrap, CSS e HTML.

Os objetivos gerais do projeto final eram os seguintes:

- Desenvolvimento funcional/técnico
- Seguir linhas básicas do layout fornecido
- Seguir boas práticas para produção de código
- Aplicar conceitos e tecnologias apreendidas
- Sistema resistente a falhas.

O projeto proposto era uma réplica de uma seguradora(automóvel/viagens), que incorporava as seguintes áreas funcionais:

- *Login/logout*
- Dados Pessoais
- *Dashboard*
- Criação/Atualização de apólices
- Consulta de apólices.

2.3.2 Requisitos do projeto final

De maneira geral, pretendia-se criar uma aplicação web, desenvolvida em .NET Core com o objetivo de gerir diversos clientes de uma seguradora. A aplicação seguia os seguintes requisitos:

- No ecrã de login:
 - Os campos *username/ password* não podiam estar vazios.
 - O campo *username* tem de ser um email.
 - A *password* não deve ser visível ao utilizador.
 - Se o login for realizado com sucesso, devia ser apresentada uma mensagem ao *user* e redirecioná-lo para o ecrã de *Dashboard*.
 - Quando o botão “*Enter*” for pressionado, a ação de login deve ser invocada.
- No menu deve ter as seguintes entradas:
 - Início.
 - Dados Pessoais.
 - Apólices.
- No ecrã de *Dashboard*:
 - Apresentar um carrossel/*slider* que permitisse navegar entre as apólices do cliente,devendo conter a informação mencionada no protótipo(apresentado mais à frente).
 - No caso de seleção de uma das apólices, redireciona o *user* para a página de detalhes da apólice.
 - Apresentar um controlo com as últimas operações do utilizador.
- No ecrã de Dados Pessoais:
 - Possibilitar a consulta dos dados pessoais do utilizador.
 - Permitir alterar foto de perfil do utilizador.

- No ecrã de consulta de apólices:
 - Apresentar um *slider*/carrossel que permitisse navegar entre as diferentes apólices.
 - A informação devia ser organizada em três tabuladores: dados gerais, condições e documentos.
- Controlo de logout:
 - Devia ser apresentado apenas quando o utilizador estivesse autenticado.
 - Devia ser apresentada a fotografia de perfil do utilizador.
 - Devia permitir a realização de *logout*(terminar sessão).

3. Projeto Document Digital Service

3.1 Contexto do projeto

Esta secção tem como objetivo apresentar o desenvolvimento de um projeto Frontend em React focado na criação de uma página de login com autenticação de utilizadores e gestão de permissões. O projeto já se encontrava em desenvolvimento, mas foi necessário implementar um sistema de login seguro para proteger informações sensíveis e garantir o acesso adequado a diferentes níveis de utilizador.

Para garantir a segurança e eficiência das aplicações torna-se imperativo usar tecnologias recentes que permitam uma gestão de autenticação robusta e desenvolvimento de uma interface com o utilizador simples e intuitiva.

Os principais objetivos do projeto incluíam:

- Desenvolver página de login.
- Implementar a autenticação de utilizadores usando tokens JWT.
- Armazenar token de autenticação na Session Storage.
- Implementar funcionalidades de User Roles.

- Gerir o redirecionamento da aplicação.
- Garantir a segurança da aplicação.

Nas próximas secções serão desenvolvidos os objetivos, as tecnologias usadas, evidenciando os resultados obtidos, desafios enfrentados e aprendizagens obtidas.

3.2 Tecnologias Utilizadas

3.2.1 Visual Studio Code

A IDE escolhida para desenvolvimento deste projeto foi o VSCode, “um editor de código multiplataforma que suporta uma ampla gama de linguagens de programação”, segundo o site Pplware, como por exemplo JavaScript, Python, C++, Java, entre outras. Este suporte é ampliado ainda mais através da sua arquitetura de extensões, permitindo assim aos seus utilizadores instalar plugins e extensões para adicionar funcionalidades aos seus projetos.

O VSCode possui várias características que o diferenciam de outros editores de código, tornando-o a escolha mais popular entre os desenvolvedores. Abaixo estão algumas das principais diferenças:

- Extensibilidade e personalização: possui um vasto Marketplace de extensões e permite que os utilizadores personalizem praticamente todos os aspetos do ambiente de desenvolvimento.
- Funcionalidade embutidas: oferece sugestões de código, contém depurador embutido permitindo fazer *debug* da aplicação e um terminal integrado que permite executar comandos sem sair do editor.
- Suporte a Git: possui a integração nativa com Git, uma das ferramentas de gestão e versionamento mais usadas do mercado.

- Colaboração em tempo real: Oferece extensão chamada Live Share, que permite a colaboração em tempo real entre desenvolvedores. Múltiplos utilizadores podem editar e correr código simultaneamente, facilitando o trabalho em equipa.

Por estas razões o VSCode foi o editor de código escolhido para desenvolver a plataforma de login deste projeto.

3.2.2 Azure DevOps

Azure DevOps é uma plataforma integrada da Microsoft(n.d.b) que segundo a mesma, “oferece um conjunto de ferramentas para o desenvolvimento de software. Facilita a colaboração entre equipas, permitindo a gestão do ciclo de vida do desenvolvimento de aplicações de forma eficiente”.

Esta ferramenta é extremamente útil para as equipas de desenvolvimento pois ajuda no planeamento e acompanhamento de tarefas pelo gestor de projeto, consegue gerir o código-fonte colaborativamente, o que facilita o versionamento e histórico de alterações. Com a funcionalidade dos pipelines, o Azure DevOps automatiza a construção, teste e implementação de software, garantindo assim uma entrega rápida e eficaz.

A ferramenta é usada por uma grande quantidade de empresas pois engloba uma série de serviços que incluem:

- Azure Boards: gestão de projetos e acompanhamento de tarefas.
- Azure Repos: repositórios Git para controle de versão.
- Azure pipelines: integração e entregas contínuas.
- Azure Test Plans: gestão de testes.

No desenvolvimento deste projeto foi usada esta ferramenta pois otimiza processos, melhora a colaboração e garante entregas ágeis e de maior qualidade. Acresce que, o Azure DevOps é usado por toda a empresa nos vários projetos que desenvolvem.

3.2.3 React

Segundo o Alura(2023,January 17), “React é uma biblioteca JavaScript de código aberto criada pelo Facebook(atual Meta) para a construção de interfaces de utilizador. É popular por ser fácil de usar, altamente flexível e escalável, e é usado por muitas empresas de tecnologia, incluindo o Facebook,Instagram e Airbnb”.

O React permite criar componentes reutilizáveis, facilitando a manutenção e escalabilidade de aplicações grandes. Além disso,tem várias características que a distingue de outras *frameworks*, como por exemplo o facto de permitir a construção de *interfaces* a partir de componentes independentes e reutilizáveis, o uso de um Virtual DOM (Document Object Model) para atualizar apenas os componentes que realmente mudam, possui, ainda uma ampla comunidade ativa e uma quantidade significativa de bibliotecas e ferramentas que ampliam as funcionalidades dos projetos.

A grande inovação que esta ferramenta trouxe ao mercado foi permitir escrever HTML dentro do JavaScript, tornado o código mais legível e expressivo. A esta inovação deu-se o nome de JSX (JavaScript XML).

3.2.4 Axios

A Axios é uma biblioteca JavaScript usada para fazer requisições HTTP a partir do navegador e do Node.js, criada com o objetivo de simplificar a comunicação entre o *frontend* e o *backend*.

Esta ferramenta oferece uma API que suporta a realização de requisições assíncronas, tornando a gestão de chamadas HTTP mais eficiente.

A Axios é baseada em Promises, o que significa que é possível escrever código assíncrono(*async/await*) mais linear e fácil de entender. Ao fazer uma requisição HTTP,

a função devolve uma Promise, depois pode-se usar o método `.then()` ou `.catch()` para dar uma resposta positiva ou para tratamentos de erro, respetivamente.

Outra das vantagens do Axios que foi usada no desenvolvimento do projeto, é o facto de permitir a configuração de interceptors usados para modificar ou inspecionar requisições ou respostas antes que elas sejam concluídas. Axios suporta também configuração de valores padrões para cabeçalhos HTTP, tempo limite de requisição, simplificando a gestão das requisições.

O papel da Axios neste projeto traduziu-se:

- No envio de credenciais de login do *user* ao *backend* e espera a resposta.
- No uso de interceptors para garantir que apenas *users* autenticados pudessem aceder a certas rotas e recursos da aplicação.
- Na gestão de roles.

3.3 Planeamento do Projeto

3.3.1 Requisitos Funcionais

Na tabela 1 estão especificados os requisitos funcionais do projeto. São especificações detalhadas do comportamento que o sistema deve exibir para cumprir com as funções principais. Descrevem o que o sistema deve fazer, incluindo ações, tarefas, dados a serem processados, interações com utilizadores e outras partes do sistema. Especificar os requisitos é fundamental para o desenvolvimento de software, pois guiam a equipa de desenvolvimento na criação de funcionalidades que atendam às necessidades dos utilizadores.

Tabela 1 - Requisitos Funcionais

Requisito	Objetivo
Autenticação do Utilizador	A aplicação deve permitir que os utilizadores façam login usando as suas credenciais
Verificar Credenciais	As credenciais devem ser verificadas no <i>backend</i> , e em caso de sucesso, uma <i>token</i> JWT deve ser gerada e enviada para o <i>frontend</i>
Armazenamento de <i>tokens</i>	A <i>token</i> JWT deve ser armazenada na <i>session storage</i> do navegador após autenticação bem-sucedida
Requisições do <i>backend</i>	A <i>token</i> deve ser usada para autenticar todas as requisições subsequentes do <i>backend</i>
Gerir User Roles	A aplicação deve identificar o papel do utilizador a partir da <i>token</i>
Níveis de Acesso	Diferentes roles devem ter diferentes níveis de acesso e permissões na aplicação
Redireccionamento	Após a autenticação, os <i>users</i> devem ser redirecionados à página principal da aplicação

3.3.2 Requisitos não funcionais

No contexto do desenvolvimento de software, os requisitos não funcionais garantem que o sistema atende às necessidades de desempenho, usabilidade e segurança.

Estes requisitos definem as características de qualidade do sistema, assegurando que ele opere de forma eficiente e confiável em diversas condições.

Para o projeto em questão foram definidos os seguintes requisitos não funcionais:

- Desempenho rápido e responsivo.
- Interface intuitiva e amigável.
- Segurança dos dados do utilizador.
- Atualizações regulares.
- Manutenção eficiente.

3.4 Página Login

Na fase de desenvolvimento de Frontend, uma das principais funcionalidades implementadas foi a página de login. Esta página foi projetada para oferecer uma interface intuitiva e fácil de usar, permitindo que os utilizadores acessem ao sistema de forma segura.

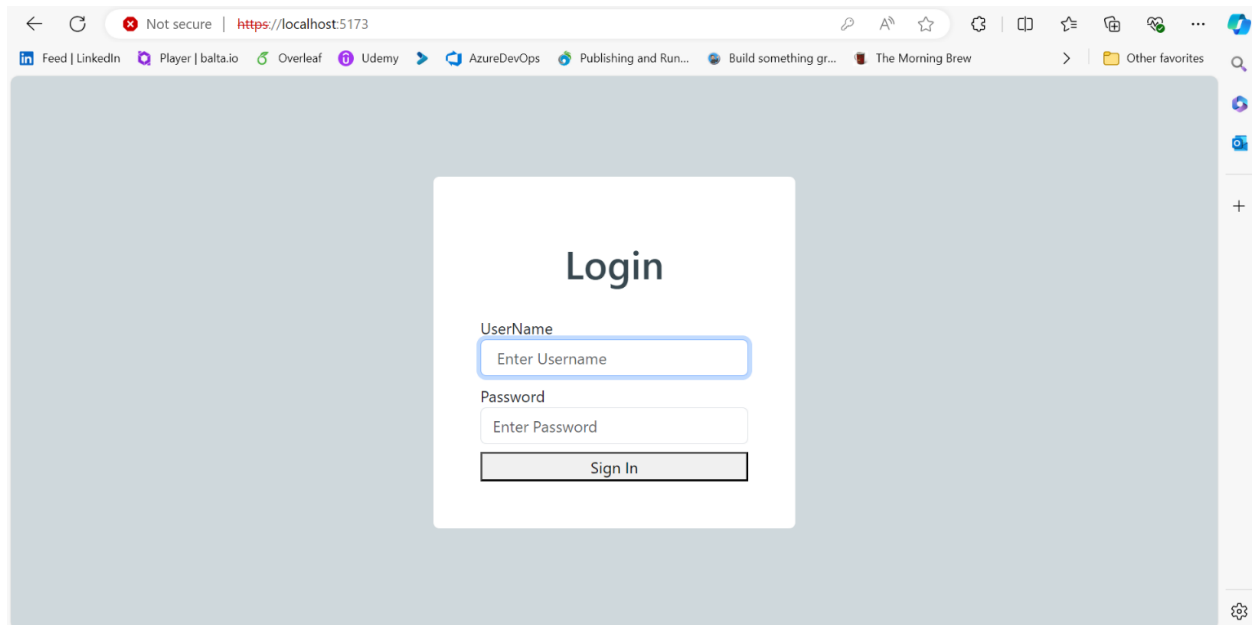
A interface incluía campos para a inserção de credenciais, como o nome de utilizador e senha, além de elementos visuais que orientam o *user* durante o processo de autenticação. O design responsivo garantia que a página fosse acessível em dispositivos de tamanhos diferentes.

Abaixo, são apresentados alguns “*print screens*” da página de login, ilustrando o layout e disposição dos elementos, bem como as mensagens de feedback de sucesso ou erro no login.

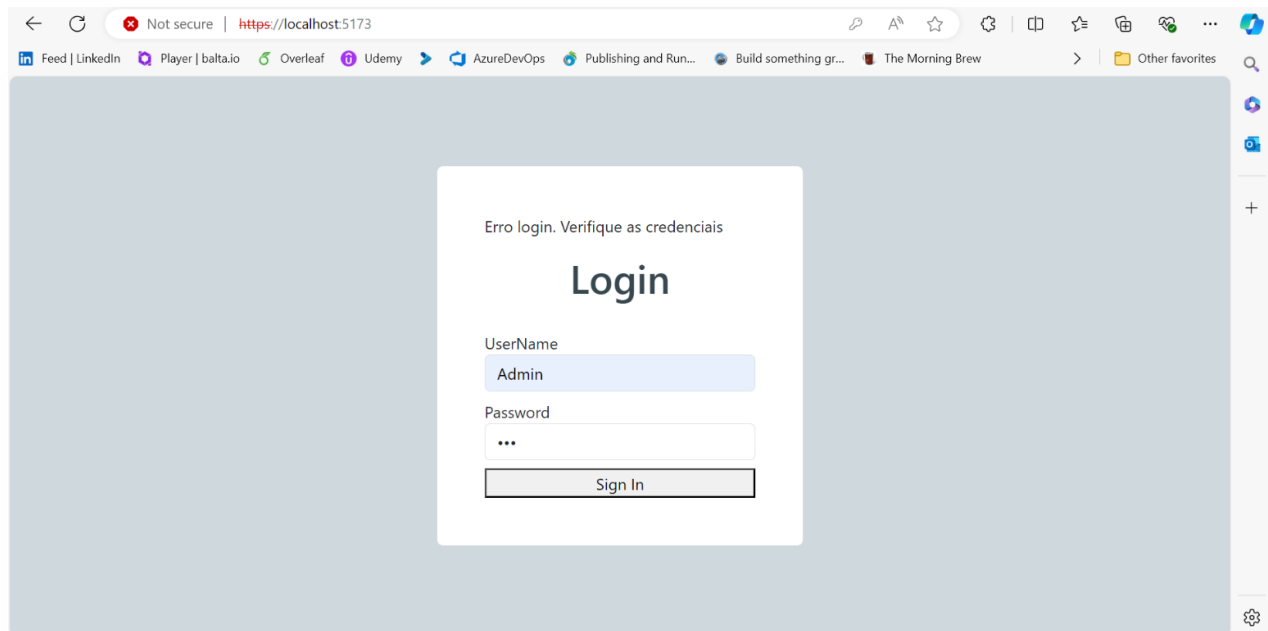
Na figura 1 é ilustrada a página de login criada em React Native, aqui é usada a ferramenta Bootstrap para a melhoria do design, proporcionando uma boa experiência ao

utilizador. É usado um formulário com dois *input labels* (*UserName* e *Password*) e um botão “*Sign In*” que quando os dados são válidos remetem o *user* para a página principal.

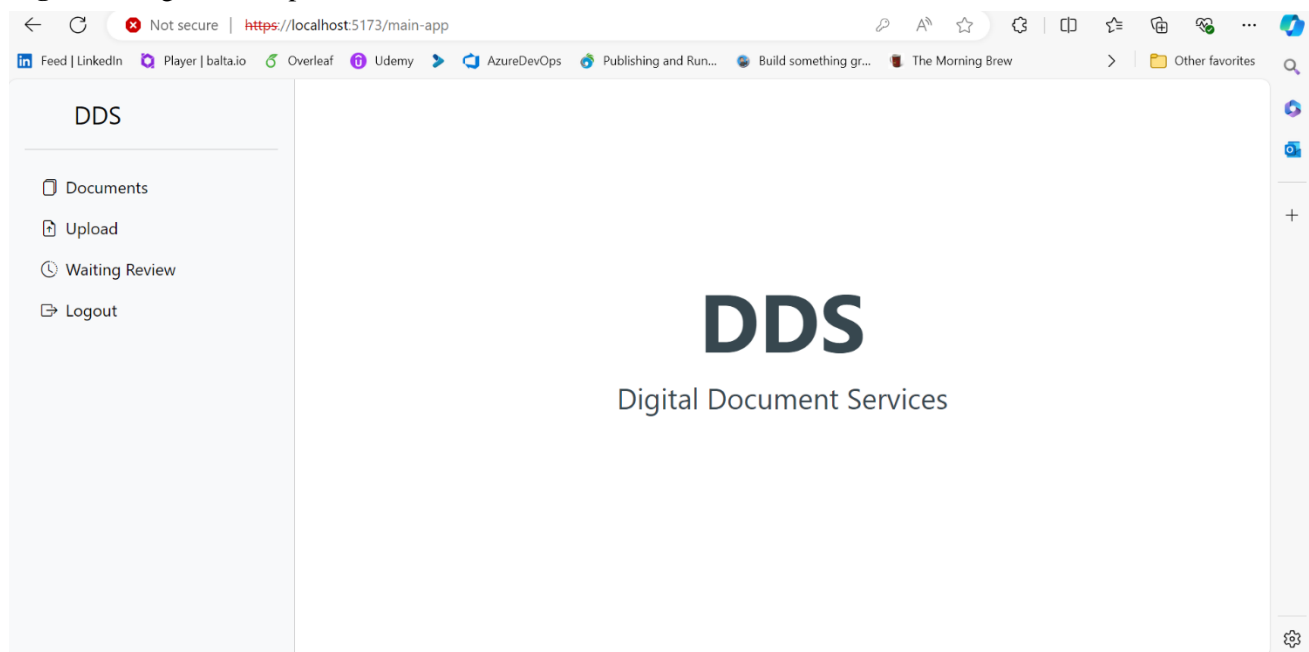
Figura 1. Layout página de login



Na figura 2 observa-se que o sistema está pronto para enviar uma mensagem de erro ao utilizador, caso as credenciais de login estejam erradas. Esta funcionalidade permite que o sistema fique protegido contra utilizadores maliciosos.


Figura 2. Tentativa de login com credenciais erradas

Na figura acima (Figura 3) é apresentada a página principal que o utilizador vê quando o login é bem-sucedido. A página já estava previamente desenvolvida.

Figura 3. Página Principal

Na Figura 4 está ilustrado como foi feita a implementação de *logout*, onde são removidas as tokens e os *user roles* e o usuário será depois redirecionado para a página de login. Tanto o tema das *access tokens* e as *user roles* serão analisados mais à frente.

Figura 4. Logout



```
src > authentication > logout.tsx > logout
1  export const logout = () => {
2      // Remove tokens e dados do sessionStorage
3      sessionStorage.removeItem('refreshToken');
4      sessionStorage.removeItem('accessToken');
5      sessionStorage.removeItem('TokenTime');
6      sessionStorage.removeItem('Role');
7  }
8  }
```

3.5 Ligação ao backend

A ligação com o *backend* foi realizada através do Axios, biblioteca JavaScript que permite fazer requisições HTTP a partir do navegador.

A configuração do Axios vai ser mostrada e explicada nas seguintes figuras.

Assim, na figura 5 é criada uma instância de Axios que permite configurar uma URL base e cabeçalhos que serão aplicados a todas as requisições feitas por essa instância. Esta configuração permite que o *frontend* se comunique com o *backend* de maneira eficiente e segura. O uso de uma instância personalizada facilita a manutenção e reutilização do código, garantindo que todas as requisições sigam um padrão definido.

Figura 5. Configuração Axios

```
const axiosServer = axios.create({
  baseURL: import.meta.env.VITE_API_URL,
  headers: {
    'Access-Control-Allow-Origin': '*',
    'Content-Type': 'application/json',
  }
});
```

Na figura 6 é ilustrado a função de login que envia credenciais ao *backend* para autenticação. Esta função é integrada num componente React que captura as credenciais do utilizador e chama a função login, lidando com erros e sucessos.

Figura 6. Metodo Post com Axios

```
const login = async (credentials: Credentials) =>{
  try{
    const loginResponse = await axiosServer.post('/Auth/Login',
      {
        email : credentials.userName,
        password : credentials.psw,
        twoFactorCode : '',
        twoFactorRecoveryCode: ''
      }
    );
  }
```

3.6 Access tokens

Na implementação da página de login, foram usadas tokens JWT (JSON Web Tokens) para gerir a autenticação do utilizador. Os tokens JWT contêm informações codificadas (JSON) sobre o utilizador, sendo assinados digitalmente, permitindo que o sistema verifique a autenticidade do token sem precisar de consultar o *backend* a cada requisição.

Depois de uma autenticação bem-sucedida, o *token* é gerado pelo *backend* e é armazenado no “*Session Storage*” do navegador web. Isso garante que a sessão do utilizador permaneça ativa enquanto o navegador está aberto, oferecendo assim uma camada adicional de segurança.

A escolha do “*Session Storage*” justifica-se pela segurança, já que os dados são removidos automaticamente quando a janela do navegador é fechada, minimizando assim os riscos de exposição. Além disso, essa técnica melhora a experiência do utilizador, pois proporciona acesso contínuo e sem interrupções durante a sessão.

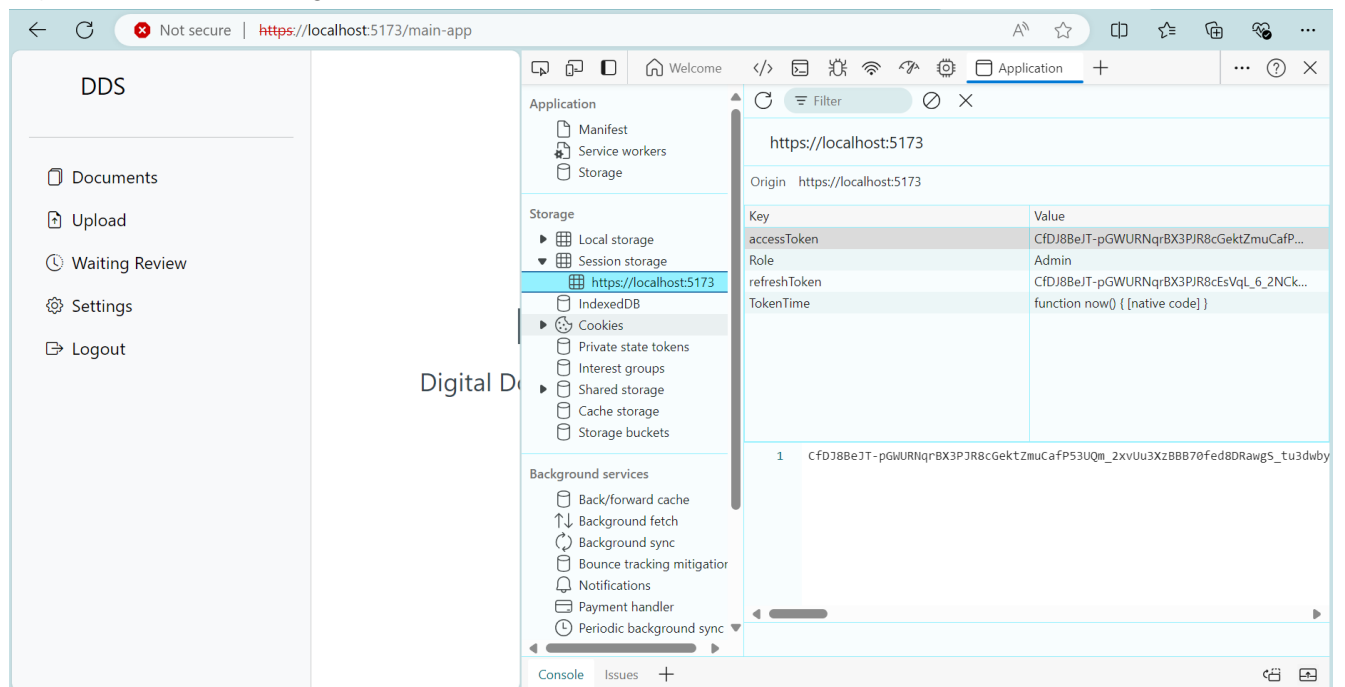
Na figura 7 está o código usado dentro da função de login que armazena, *access token* e *refresh token* na *Session Storage*.

Figura 7. Armazenar tokens

```
//Armazena token na sessions storage  
sessionStorage.setItem('refreshToken',refreshToken);  
sessionStorage.setItem('accessToken',token);  
sessionStorage.setItem('TokenTime',Date.now.toString());
```


Após um login bem-sucedido e abrindo o “Inspecionar”, na aba “Application” é possível comprovar que na Session Storage estão armazenados os tokens de acesso como é identificado o role do utilizador. Os *user roles* serão analisados na próxima subsecção.

Figura 8. Token no Navegador



3.7 User Roles

No desenvolvimento de sistemas de autenticação e autorização, a gestão de user roles é essencial para controlar o acesso a diferentes partes e funcionalidades da aplicação.

Neste projeto, foi implementada a funcionalidade de *user roles* através do JSON fornecido pelo backend. Esta abordagem permite incorporar informações sobre os papéis do utilizador diretamente no *token* de autenticação, facilitando a verificação e o controlo de acesso no *frontend* e no *backend*.

No *backend*, a geração do token JWT inclui informações sobre os papéis do utilizador. No exemplo a *token* inclui um *claim* role com o valor “Admin” e esse *token* é então devolvido ao cliente e armazenado o nome da role na *Session Storage*.

Estas *claims* são úteis pois permitem ao *frontend* utilizar esse *token* para ajustar a interface conforme as permissões que o cliente queira dar a cada um dos utilizadores. No caso deste projeto, existe uma página denominada de “waiting review” onde apenas o administrador pode ter acesso. Com isso é possível alterar as rotas e proteger componentes para que utilizadores que não tenham o papel de administrador não possam ter acesso a elas.

Figura 9 . Configuração User Roles

```
// Faz a chamada para obter os claims detalhados
const claimsResponse = await axiosServer.post('/Auth/DetailedClaims', {}, {
  headers: {
    Authorization: `Bearer ${token}`
  }
});

const userClaims = claimsResponse.data;

//Extraí Nome da Role
const role = userClaims["http://schemas.microsoft.com/ws/2008/06/identity/claims/role"][0];

sessionStorage.setItem('Role', role);
```

3.8 NSWAG

Já numa fase final de projeto foi usada uma ferramenta para geração de clientes e documentação de APIs a partir de especificações OpenAPI(conhecida como Swagger). Esta ferramenta tem o nome de NSwag e permite que desenvolvedores automatizem a criação de código cliente em várias linguagens a partir da definição de API fornecida pelo *backend*. A especificação OpenAPI é um formato padrão para descrever APIs RESTful, que inclui informações sobre *endpoints*, métodos HTTP suportados, parâmetros e esquemas de resposta.

Os passos usados para usar esta ferramenta são os seguintes:

- O JSON gerado pelo Swagger no *backend* é introduzido no NSWAG Studio.
- Configuração das opções de geração de código.
- Geração de código automaticamente.
- Importar código gerado para o *frontend*.

Ao integrar o código gerado no projeto React, as chamadas à API ficam simplificadas e tipificadas, reduzindo a chance de erros, inconsistências e diminuindo o tempo gasto na geração de código manualmente.

De maneira a explicar o que foi dito anteriormente, apresenta-se a seguir um exemplo de função gerada pelo NSwag e de seguida a implementação no projeto.

A figura 10 representa o código de uma das muitas funções geradas pelo NSwag. No caso é uma função que encontra todos os documentos listados na base de dados, no endpoint “MyDocuments” na linha 2 do código.

Figura 10. Código NSwag

```
ddsWebEndpointsDocumentEndpointsGetSelfList( cancelToken?: CancelToken): Promise<DocumentDTO[]> {  
    let url_ = this.baseUrl + "/MyDocuments";  
    url_ = url_.replace(/[?&]$/, "");  
  
    let options_: AxiosRequestConfig = {  
        method: "GET",  
        url: url_,  
        headers: {  
            "Accept": "application/json"  
        },  
        cancelToken  
    };  
  
    return this.transformOptions(options_).then(transformedOptions_ => {  
        return this.instance.request(transformedOptions_);  
    }).catch((_error: any) => {  
        if (isAxiosError(_error) && _error.response) {  
            return _error.response;  
        } else {  
            throw _error;  
        }  
    }).then((_response: AxiosResponse) => {  
        return this.processDDSWebEndpointsDocumentEndpointsGetSelfList(_response);  
    });  
}
```

Na figura 11 é criada uma variável GetDocuments() que usa a função da figura 10, permitindo assim a automatização e melhoria de produtividade.

GetDocuments() é usado numa componente que mostra a tabela de todos os documentos da lista.

Figura 11. Implementação código NSwag

```
export const GetDocuments = async () => {  
    return new DDSAPIService("http://localhost:5180", axiosServer).ddsWebEndpointsDocumentEndpointsGetSelfList()  
};
```

Conclusão

Dado por concluído o conteúdo do estágio, importa mencionar algumas considerações tanto a nível dos objetivos propostos como a nível das experiências que permitiram uma evolução significativa.

No que diz respeito aos objetivos propostos na Academia .Net pode afirmar-se que foi concluída com sucesso., ao serem desenvolvidos todos os trabalhos propostos e dentro dos prazos pedidos. Relativamente ao projeto interno considera-se também que foi realizado com sucesso, pois o objetivo seria desenvolver uma página de autenticação para um projeto que já existia, adicionando as funcionalidades de *tokens* e *user* roles que no final, ficaram implementadas com sucesso.

Pessoalmente, o estágio realizado na ITSector fez-me adquirir conhecimentos tanto a nível pessoal, como sobretudo a nível técnico, que são os dois pilares para a minha carreira profissional. Aprendi que um programador tem de ser muito mais do que um desenvolvedor de software, tem de saber relacionar com os colegas, saber trabalhar em equipa e saber trabalhar sob pressão.

Considero que este estágio foi bastante adequado para o meu percurso na Licenciatura em Engenharia Informática, ao possibilitar em contexto real de trabalho, aplicar os conceitos assimilados ao longo do meu percurso académico. Foi também importante para esclarecer o que eu gosto realmente de fazer, pois permitiu-me explorar diferentes áreas do desenvolvimento de software e adquirir uma maior experiência a nível de programação e melhorar em termos de código eficiente e organizado.

Bibliografia

- Fowler, M. (2006). *Continuous integration*. Retrieved from <https://martinfowler.com/articles/continuousIntegration.html>
- Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design patterns: Elements of reusable object-oriented software*. Addison-Wesley Professional.
- Microsoft. (2024). *C# documentation*. Retrieved from <https://docs.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core>
- Microsoft. (2024, June 6). *Introduction to ASP.NET Core*. Retrieved from <https://docs.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core>
- Microsoft. (n.d.a). *Visual Studio Code documentation*. Retrieved from <https://code.visualstudio.com/docs>
- Microsoft. (n.d.b). *Azure DevOps Services*. Retrieved from [Azure DevOps Services | Microsoft Azure](#)
- Amazon Web Services. (n.d.a). O que é o .Net? Retrieved from <https://aws.amazon.com/pt/what-is/net/>
- Amazon Web Services. (n.d.b). O que é SQL? Retrieved from <https://aws.amazon.com/pt/what-is/sql/>
- Alura. (2023, September 18). Bootstrap: O que é, Documentação, como e quando usar. Retrieved from [Bootstrap: o que é, como usar, documentação e exemplos | Alura](#)
- Alura(2023,January 17). *React: o que é, como funciona e um guia dessa popular ferramenta JS*. Alura. Retrieved from <https://www.alura.com.br/artigos/react-js>
- BM Integration Bus 10.0.0. (2022,August 24). Retrieved from <https://www.ibm.com/docs/pt-br/integration-bus/10.0?topic=services-what-is-soap>
- Pinto, P. (2024, March 16). Visual Studio Code: É este o melhor IDE para programadores? *Pplware*. <https://pplware.sapo.pt/software/visual-studio-code-e-este-o-melhor-ide-para-programadores/>