TrokaTroka

Pedro A. S. Ribeiro[1], Robério P. da S. Badu [1], Rodrigo C. de Souza[1], Yuri M. Marques[1]

[1]Universidade Evangélica de Goiás - UNIEVANGÉLICA
Caixa Postal 122 e 901 – 75083-515 – Anápolis – GO – Brasil

[2]Engenharia de Software – UniEvangélica
Anápolis, GO.

{pedro.ribeiro, rodrigo.costa, yuri.marques}@aluno.unievangelica.edu.br,

roberio.pb@gmail.com

Abstract. *One of the reasons for the decrease in book reading in Brazil is the increase in activities such as social media, influenced by cultural and economic factors, including high book prices and inflation. To encourage reading, a software will be developed for book exchange, using Kanban and Scrum methodologies, and its feasibility will be evaluated through an online questionnaire. Two applications will be created, a website and a mobile app, using microservices as the architecture for the back-end. The goal is to contribute to book reading in Brazil.*

Resumo. *Um dos motivos da diminuição da leitura de livros no Brasil é o aumento de atividades como redes sociais, influenciado por fatores culturais e econômicos, incluindo preços altos de livros e inflação. Para incentivar a leitura, um software será desenvolvido para troca de livros, usando as metodologias Kanban e Scrum, e será avaliado por um questionário online. Duas aplicações serão criadas, um site e um aplicativo móvel, usando microsserviços como arquitetura para o back-end. O objetivo é contribuir para a leitura de livros no Brasil.*

1. Introdução

O sistema criado no projeto Trokatroka tem o intuito de realizar trocas de livros entre usuários de maneira mais simples e conveniente, utilizando um sistema de pontos. Os pontos para a troca de livros são adquiridos no cadastro de uma nova conta e posteriormente com trocas realizadas dentro do sistema. No cadastro do livro o usuário faz uma definição de preço (em pontos) de quanto quer no livro ofertado, e outro usuário usa seus pontos para realizar essa troca. As negociações são feitas de maneira direta entre usuários, assim como as trocas também.

Assim, o projeto visa produzir um sistema web e responsivo que pode ser acessado de qualquer dispositivo através de um *Progressive Web App* (PWA), garantindo uma experiência de trocas de livros conveniente e flexível. O PWA possui uma grande vantagem contra os aplicativos tradicionais, os aplicativos tradicionais precisão ser instalados nos dispositivos, porém o PWA possui uma instalação através da web, podendo acessar o sistema pelo navegador.

A importância de ter uma abordagem mobile, através do PWA está ligada diretamente ao aumento no uso de dispositivos móveis. Hoje em dia, as pessoas usam seus smartphones e tablets para praticamente tudo, desde fazer compras *online* até trabalhar remotamente e se comunicar com amigos e familiares. Os aplicativos móveis tornaram-se uma parte essencial do cotidiano.

Além disso, ter uma abordagem no desenvolvimento mobile pode ajudar o projeto a coletar e analisar dados do usuário, o que pode ser usado para melhorar a experiência do usuário e personalizar as ofertas de produtos e serviços. Isso pode levar a um aumento na fidelidade do cliente e a um melhor desenvolvimento do projeto como um todo.

Dessa maneira, a parte mobile é essencial para o progresso do projeto como um todo. Assim, com essa tecnologia o grupo pode melhorar a vida dos consumidores, aumentar a eficiência do que foi proposto e solucionar os problemas elencados. Com o aumento contínuo no uso de dispositivos móveis, o desenvolvimento mobile será importante para a propagação do projeto.

1.1. Contextualização

Segundo Pressman, ”[...] Software de computador é o produto que profissionais de software desenvolvem e ao qual dão suporte no longo prazo. [...]”. Dessa forma, o desenvolvimento de software se tornou uma atividade que está presente nas mais diversas áreas da sociedade, os softwares desenvolvidos estão inseridos no cotidiano brasileiro.

A leitura é uma atividade comumente presente no dia a dia dos brasileiros. No entanto, uma pesquisa conduzida pelo Instituto Pró-Livro (IPL), em parceria com o Itaú Cultural e a Inteligência em Pesquisa e Consultoria Estratégica (IBOPE), em sua 5ª edição, demonstra que a leitura tem sido cada vez mais negligenciada, especialmente na faixa etária de 14 a 24 anos, em comparação com o uso da internet, por exemplo.

Portanto, é fundamental reestabelecer o hábito da leitura, sobretudo entre os jovens, uma vez que a leitura é uma ferramenta poderosa para transformar a sociedade, conforme mencionado por Henry David Thoreau em seu livro Walden: ”Muitos homens iniciaram uma nova fase em suas vidas a partir da leitura de um livro”.

1.2. Justificativa

Porém, como mostrado na pesquisa, realizada pela Confederação Nacional do Comércio de Bens, Serviços e Turismo (CNC), 91% das profissões tiveram um reajuste abaixo da inflação. Dessa forma, o poder de compra do brasileiro teve um decréscimo nos últimos anos, em decorrência da alta inflação. Entretanto, de acordo com uma pesquisa, divulgada em agosto de 2022 pelo Sindicato Nacional dos Editores de Livros (SNEL), houve um aumento de 3,36% no preço médio dos livros em agosto de 2022, comparado com o mesmo período de 2021.

Sendo assim, este aumento contribui com que haja cada vez menos leitores, uma vez que, segundo dados estatísticos do Instituto Pró-Livro (IPL) Itaú Cultural e IBOPE Inteligência mostrados em sua 5ª edição de pesquisa Retratos da leitura no Brasil (2019), existem cerca 100 milhões de leitores, compondo 52% da população sendo ele 4% menor que sua análise anterior (2015).

Além disso, é possível observar que a popularização de soluções para leitura de ebooks tem impactado significativamente no mercado de livros físicos. Com a facilidade

de acesso a uma grande variedade de títulos em formato digital, muitas pessoas têm optado por essa opção de leitura, em detrimento dos livros físicos.

No entanto, é importante ressaltar que a adoção de soluções como o Kindle, por exemplo, ainda é pouco acessível para a camada mais pobre da sociedade. O alto custo desses dispositivos é um dos fatores que tornam difícil o acesso a esse tipo de tecnologia para uma parcela significativa da população.

Dessa forma, apesar das vantagens da leitura em formato digital, ainda há desafios a serem enfrentados para garantir que todas as pessoas tenham acesso a essa possibilidade de leitura. É necessário buscar soluções que tornem essas tecnologias mais acessíveis e inclusivas, para que mais pessoas possam desfrutar dos benefícios que a leitura pode oferecer.

Como mostrado anteriormente, todos esses fatores contribuem com a falta de acesso a livros, diante dessa problemática, torna-se necessário a proposição de soluções. Como solução, foi idealizado e projetado um sistema, visando o desenvolvimento de uma aplicação que possa contribuir para a leitura no Brasil.

Uma vez que, o desenvolvimento de software é uma área que vem crescendo nos últimos anos, já que só em 2021 cresceu cerca de 6,5%, se comparado com o ano anterior, totalizando um valor aproximado a 53,3 bilhões de dólares, como publicado pelo Ministério da Ciência, Tecnologia e Inovação (MCTI) no relatório Indústria de Software e Serviços de TIC no Brasil: caracterização e trajetória recente.

1.3. Problema

Nos últimos anos, o Brasil tem enfrentado uma diminuição no número de leitores, resultado de diversos fatores socioeconômicos, incluindo a inflação e o aumento do preço médio dos livros. Essa realidade tem consequências graves, uma vez que, de acordo com a Pesquisa Nacional por Amostra de Domicílio (PNAD) Contínua Educação, realizada pelo IBGE em 2019, quase um terço da população brasileira se enquadra na definição de analfabeto funcional.

É evidente que a leitura de livros é uma atividade fundamental para o desenvolvimento crítico e cognitivo do indivíduo. Dessa forma, é crucial que sejam encontradas soluções para aumentar o número de leitores no país. Isso não só ajudará no combate ao analfabetismo funcional, mas também contribuirá significativamente para o desenvolvimento intelectual da população.

Entretanto, atualmente as soluções disponíveis no mercado para a troca de livros não conseguem satisfazer essa demanda e falham em contribuir efetivamente para o desenvolvimento da leitura no Brasil. Dessa forma, surge a seguinte questão: como o desenvolvimento de software pode contribuir para o aumento no número de leitores?

2. Objetivos

O propósito deste projeto é criar aplicações para a troca de livros utilizando tecnologias voltadas para o desenvolvimento de aplicações *web* e mobile. Para atingir esse objetivo principal, o projeto tem objetivos específicos, tais como: conduzir uma pesquisa descritiva para avaliar a viabilidade do projeto, aplicar metodologias ágeis como Kanban e *Scrum* no

desenvolvimento das aplicações necessárias para a solução, compreender os processos envolvidos na prática de desenvolvimento de software ágil e documentar os procedimentos do desenvolvimento de aplicações *web* e mobile.

Neste documento, você encontrará informações sobre a construção do projeto, que teve como base a arquitetura de microsserviços e utilizou diversas linguagens de programação para desenvolver a plataforma. Também será apresentado o gerenciamento dos bancos de dados através de SGBDs, além de informações sobre o controle de versão e versionamento do código. Serão detalhados os *Frameworks* utilizados ao longo do projeto e o motivo pelo qual foram escolhidos, e por fim, será apresentada a ferramenta de testes utilizada para garantir a qualidade do código, bem como a integração de machine learning para fornecer recomendações de livros.

3. Referencial Teórico

Neste pode-se encontrar a maneira que o projeto foi construído, tendo como base os microsserviços seguido das linguagens de programação utilizadas para o desenvolvimento da plataforma. Os SGBDs utilizados para fazer o gerenciamento dos bancos de dados, o que foi utilizado para manter o controle de versões e como foi feito o versionamento do código. *Frameworks* utilizados no decorrer do projeto e por qual motivo os mesmos tiveram que ser utilizados, finalizando com a ferramenta de testes que foi utilizada para garantir a qualidade do código, juntamente com a machine learning para indicações de livros.

3.1. Tecnologias e Ferramentas

A escolha da linguagem de programação é um aspecto importante no desenvolvimento do projeto. Para o *back-end*, optou-se pela utilização de duas linguagens: C# e Java. Essas linguagens foram escolhidas porque os desenvolvedores já possuíam experiência nelas. O C# é uma linguagem orientada a objetos e fortemente tipada (FENNER, Pierre. 2018).

Por sua vez, Java é uma linguagem orientada a objetos altamente portável, o que significa que os softwares desenvolvidos em Java podem ser executados em diversos dispositivos graças à Java Virtual Machine (FENNER, Pierre. 2018).

O sistema gerenciador de banco de dados (SGBD) é um software utilizado para gerenciar um ou vários bancos de dados. Com ele, é possível criar relações entre tabelas de dados, eliminar, copiar, alterar, criar, gerir e fazer consultas, além de permitir importação e exportação de dados. Para este projeto, foi decidido utilizar o SQL Server e o MongoDB (KREUTZ, D. L., 2022).

O SQL Server é um sistema gerenciador de banco de dados desenvolvido pela Microsoft, escolhido pela sua facilidade de análise dos dados, flexibilidade, armazenamento em nuvem e processamento de consulta inteligente (MISTRY, R.; MISNER, S. 2014.). O MongoDB é um banco de dados não relacional de código aberto, escolhido por trabalhar com dados heterogêneos, oferecer grande flexibilidade nos documentos e não exigir a projeção de um esquema de banco de dados para trabalhar com ele (MONGOBD, 2022). Ambos os SGBDs serão utilizados no sistema para gerar relatórios e análises rápidas da quantidade de usuários e informações cadastradas no banco de dados, além de permitir manipulação e armazenamento de dados sigilosos usando uma infraestrutura em nuvem e consultas dinâmicas aos livros cadastrados (KREUTZ, D. L., 2022).

Durante o desenvolvimento do sistema, o controle de versão foi mantido através do Git, uma ferramenta indispensável para gerenciar as diversas alterações realizadas ao longo do processo. Com o Git, foi possível registrar as modificações em cada arquivo, além de possibilitar o resgate de versões anteriores, quando necessário. (BELOKI, U. H., 2022). O Git é uma ferramenta de controle de versão que permite registrar e gerenciar as mudanças realizadas em um projeto em desenvolvimento. Além de manter um histórico detalhado de todas as alterações, o Git permite que os desenvolvedores trabalhem em paralelo em diferentes versões do código. (BELOKI, U. H., 2022).

Para o controle de versão do código desenvolvido, foi utilizado o Azure Repos, uma ferramenta que oferece duas opções de controle de versão: o GIT e o TFVC. Com o Azure Repos, foi possível organizar e gerenciar as alterações feitas ao longo do tempo, permitindo que diferentes equipes trabalhem simultaneamente no código, de forma colaborativa e organizada. (CHANDRASEKARA, C.; HERATH, P., 2020).

A interface do usuário é a parte gráfica de um software que permite a interação do usuário com um site, programa ou aplicativo. Para o desenvolvimento das interfaces do usuário, os desenvolvedores utilizam diferentes tipos de *Frameworks* que auxiliam e aceleram o processo de desenvolvimento. Um *framework* é um conjunto de códigos genéricos que facilita e otimiza o processo de desenvolvimento de sistemas. Existem diferentes tipos de *Frameworks*, como os de backend e os de *front-end*. Os *Frameworks* de *back-end* são usados para implementar as regras de negócio, onde o usuário não interage diretamente com o sistema. Por outro lado, os *Frameworks* de *front-end* são usados na parte em que o usuário interage com o sistema. Algumas das bibliotecas de *front-end* mais conhecidas são React, Angular e Vue.js.

De acordo com a documentação oficial do React, os principais benefícios da utilização dessa biblioteca são a reutilização de componentes, a criação de Single Page Applications (SPA), a flexibilidade e a facilidade de uso (BOCZKOWSKI, K.; PAŃCZYK, B., 2020). No React, um componente é uma parte específica da interface que pode ser reutilizada em outras interfaces, tornando o desenvolvimento mais padronizado e acelerando o processo. Além disso, o React permite criar SPAs, que carregam toda a interface de uma vez só, melhorando significativamente a velocidade de navegação do usuário (BOCZKOWSKI, K.; PAŃCZYK, B., 2020).

O React é um *framework* de fácil aprendizado, especialmente para desenvolvedores com experiência em JavaScript e HTML, que são as tecnologias utilizadas pelo framework. Além disso, é flexível e permite a utilização de outras bibliotecas e *Frameworks* (BOCZKOWSKI, K.; PAŃCZYK, B., 2020).

A qualidade do código é um aspecto crucial em qualquer projeto de desenvolvimento de software. A SonarQube é uma plataforma de código aberto desenvolvida pela SonarSource que permite uma análise direta do código desenvolvido. Com ela, é possível avaliar a qualidade do código, identificar bugs e identificar más práticas de programação, como code smells. Essa plataforma é extremamente útil para melhorar a qualidade e a segurança do código, além de ajudar a detectar possíveis problemas antes mesmo de eles se tornarem críticos (LENARDUZZI, V. et al, 2020).

3.2. Microsserviços

A arquitetura de microsserviços, descrita por Susan Rigetti em seu livro ”Microsserviços prontos para a produção: Construindo sistemas padronizados em uma organização de engenharia de software”, é uma abordagem arquitetural que se baseia em dividir a aplicação em pequenos serviços independentes. Essa abordagem permite uma maior escalabilidade, disponibilidade e tolerância a falhas, já que cada serviço é responsável por uma única atividade. Além disso, a arquitetura de microsserviços possibilita que a camada de serviço da aplicação seja desenvolvida em diferentes tecnologias (RIGETTI, S. J. F., 2022).

Neste contexto, O Azure DevOps é um produto desenvolvido e gerenciado pela Microsoft oferecem diversos serviços que contribuem para o processo de desenvolvimento como um todo. No projeto em questão, foram aplicadas diversas funções oferecidas pelo Azure DevOps, como o controle de versão do andamento do projeto, o gerenciamento de requisitos, a geração de relatórios e a gerência de projetos (KRIEF, M. 2019.).

O *back-end* é a parte da aplicação responsável pelas regras de negócio, validações e pela comunicação com o banco de dados. Essa comunicação será realizada por meio de *APIs*, que são mecanismos que permitem que dois componentes se comuniquem através de um conjunto de definições e protocolos (THIBAULT, S., 2022)

Em aplicações com arquitetura baseada em microsserviços, é essencial definir um formato para a troca de dados. Nesse projeto, será utilizado o formato JavaScript Object Notation (JSON), uma vez que ele é leve, fácil de entender e baseado em um subconjunto da linguagem JavaScript. O JSON é uma excelente opção para troca de dados em microsserviços, já que a sua simplicidade facilita a comunicação entre os serviços (BASSET, L., 2022).

Além disso, outro conceito importante em aplicações baseadas em microsserviços é a mensageria, que é um sistema de troca de mensagens e eventos gerenciado por um servidor. Para implementar esse conceito no projeto, será utilizado o RabbitMQ, um software livre e de código aberto. O RabbitMQ é uma excelente opção para implementar a mensageria em microsserviços, já que ele é escalável, robusto e confiável (SURHONE, L. M. TIMPLEDOM, M. T.; MARSEKEN, S. F., 2010).

4. Metodologia

O presente estudo realizou uma pesquisa descritiva quantitativa por meio de um questionário estruturado, aplicado a 80 participantes entre 02 de setembro e 09 de setembro de 2021. O objetivo da pesquisa foi validar a viabilidade do desenvolvimento de uma aplicação para facilitar a troca de livros e avaliar a sua aceitação. O questionário foi realizado *online* por meio da ferramenta Google Forms para garantir agilidade e oferecer relatórios das respostas obtidas. As perguntas do questionário focaram na troca de livros e leitura, a fim de avaliar se as pessoas que possuem o hábito de ler realizam trocas de livros e qual a frequência dessa prática. Com base nas respostas obtidas, foram iniciados os trabalhos de desenvolvimento da aplicação, utilizando o *framework Scrum* para o gerenciamento do projeto e dividindo-o em sprints de 15 dias.

A organização das tarefas foi feita por meio da ferramenta Jira e o método Kanban foi utilizado para a visualização do fluxo de trabalho e do estágio de cada tarefa. O código-fonte foi armazenado no Azure Repos, que utiliza o sistema Git para o controle de versões

distribuído.

A arquitetura do sistema foi definida como microsserviços, o que permite escalabilidade e o uso de mais de uma linguagem de programação. A hospedagem em nuvem das *APIs* foi realizada por meio do Azure DevOps, que oferece segurança, flexibilidade e escalabilidade. O RabbitMQ foi utilizado como sistema de mensageria para permitir a comunicação entre os sistemas distribuídos, sendo compatível com as linguagens de programação C# e Java, utilizadas no desenvolvimento do *back-end* da aplicação. A maior parte dos microsserviços foi desenvolvida em C#, utilizando a plataforma .NET e o Entity Framework, enquanto o microsserviço referente ao chat foi desenvolvido em Java.

Para o desenvolvimento do *front-end*, foi utilizada a biblioteca React. Além disso, foram empregadas tecnologias que visam garantir a portabilidade da aplicação para diferentes plataformas. Entre elas, destacam-se as Aplicações *web* Progressivas (PWA), que permitem que sites *web* funcionem como aplicativos nativos por meio do navegador. Também foi utilizada a tecnologia Atividade *web* Confiável (TWA), que possibilita a geração de aplicativos nativos para diferentes dispositivos.

Para o processo de *CI/CD*, o Azure *pipeline* foi utilizado para criar novas releases e realizar o *deploy*, integrado com o SonarQube para inspecionar a qualidade do código-fonte e garantir a agilidade e qualidade na criação de novas funcionalidades. O SQL Server foi escolhido como sistema gerenciador de banco de dados e o Azure DevOps e Vercel foram utilizados para a hospedagem em nuvem do *back-end* e do *front-end web*, respectivamente

5. Resultados

Com base nos dados coletados em um formulário (pesquisa de viabilidade) com 80 respostas válidas, pode-se constatar a viabilidade do projeto. Dos entrevistados, 72% expressaram interesse em trocar seus livros já lidos por outros, enquanto aproximadamente 73,3% afirmaram que realizariam trocas de livros pelo menos uma vez por mês. Dentre aqueles que desejavam trocar seus livros, 96,2% indicaram que utilizariam um aplicativo para realizar as trocas. Além disso, 74,7% dos entrevistados manifestaram disposição em utilizar um aplicativo para trocar outros itens. Foi considerada uma amostra de 75 respostas para análise, uma vez que algumas respostas estavam duplicadas. Esses resultados indicam uma demanda significativa e um potencial de utilização de um aplicativo para facilitar as trocas de livros e outros itens.

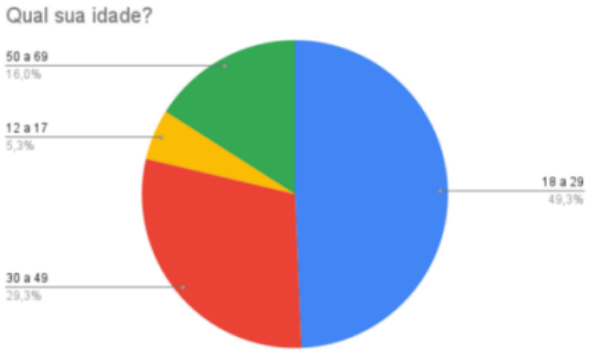

Figura 1. Idade dos entrevistados.

Através do questionário, foi possível obter informações sobre a quantidade de livros que os entrevistados possuíam na data da entrevista. Observou-se que aproximadamente 62,5% dos entrevistados possuem 15 ou mais livros. Esses dados permitem verificar uma possível relação entre a quantidade de livros possuídos e a disposição dos entrevistados em realizar trocas de livros.

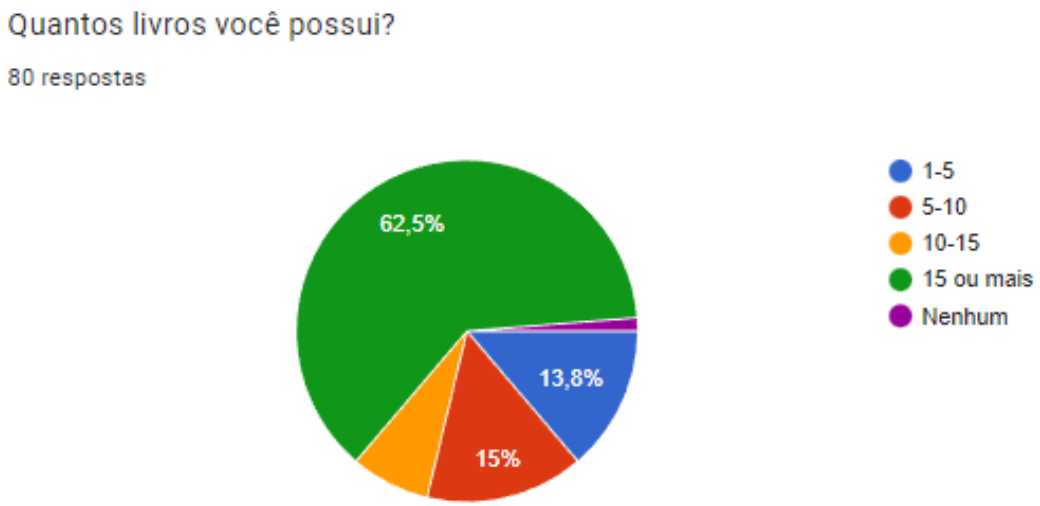

Figura 2. Quantidade de livros.

Os dados revelaram que 55% dos entrevistados são do gênero feminino, 43,8% se identificaram como do sexo masculino, enquanto cerca de 1% das pessoas preferiram não responder. Essas informações permitem identificar com mais precisão o público-alvo do projeto, que é predominantemente composto por pessoas do gênero feminino. Essa informação será útil na construção da persona, auxiliando na identificação do perfil e características dos usuários que serão atendidos pela aplicação desenvolvida neste trabalho, o que possibilita que a aplicação atenda ao público-alvo de maneira eficaz.

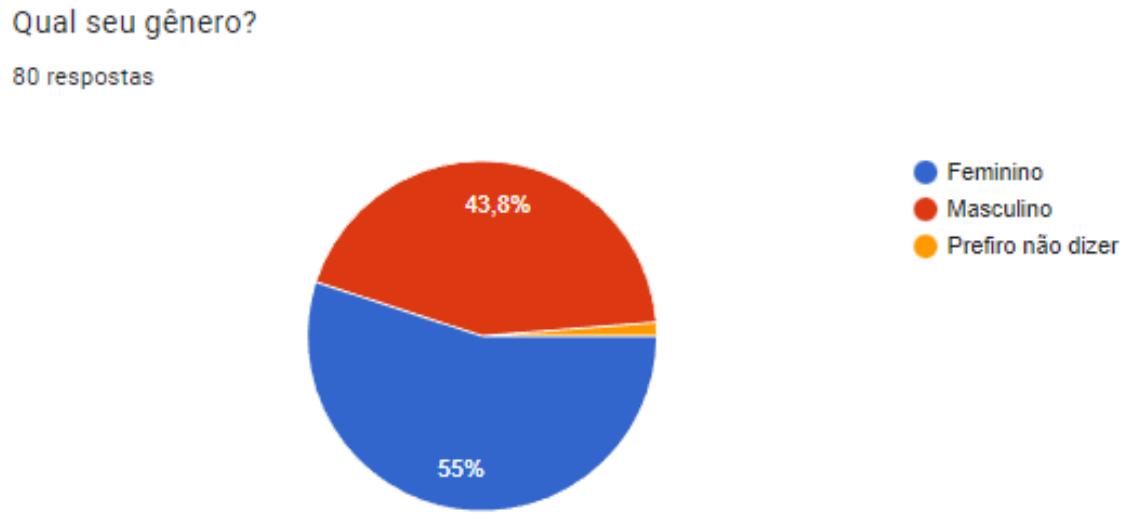

Figura 3. Gênero dos entrevistados.

Na pesquisa realizada, que questionou sobre a possibilidade de utilizar um aplicativo para facilitar as trocas de livros, o resultado foi altamente favorável. A grande maioria das respostas se concretizou em favor do uso de um aplicativo para a troca de livros, o que demonstra um alto nível de aceitação e interesse por parte dos entrevistados. Esses dados indicam um potencial promissor para o desenvolvimento e implementação de um aplicativo de trocas de livros, atendendo às necessidades e preferências do público pesquisado. Essa resposta positiva dos entrevistados reforça a viabilidade do projeto e a relevância de uma plataforma digital para facilitar a troca de livros entre os usuários.

Com base no questionário realizado, foi possível identificar as personas do sistema, o que permitiu o desenvolvimento da aplicação com foco no público-alvo que irá utilizar a plataforma *web*. Essa abordagem é essencial para direcionar os esforços de desenvolvimento e obter resultados satisfatórios na plataforma proposta neste projeto. Ao identificar o tipo de usuário a ser focado, é possível adaptar a aplicação às necessidades e preferências desse público específico, garantindo uma melhor experiência de uso e maximizando os benefícios da plataforma *web*.

A tela de cadastro de novos usuários requer o preenchimento dos seguintes campos: nome, CPF, e-mail e senha. Essas informações são necessárias para criar uma conta de usuário no sistema. A figura a seguir ilustra essa tela de cadastro de novos usuários.

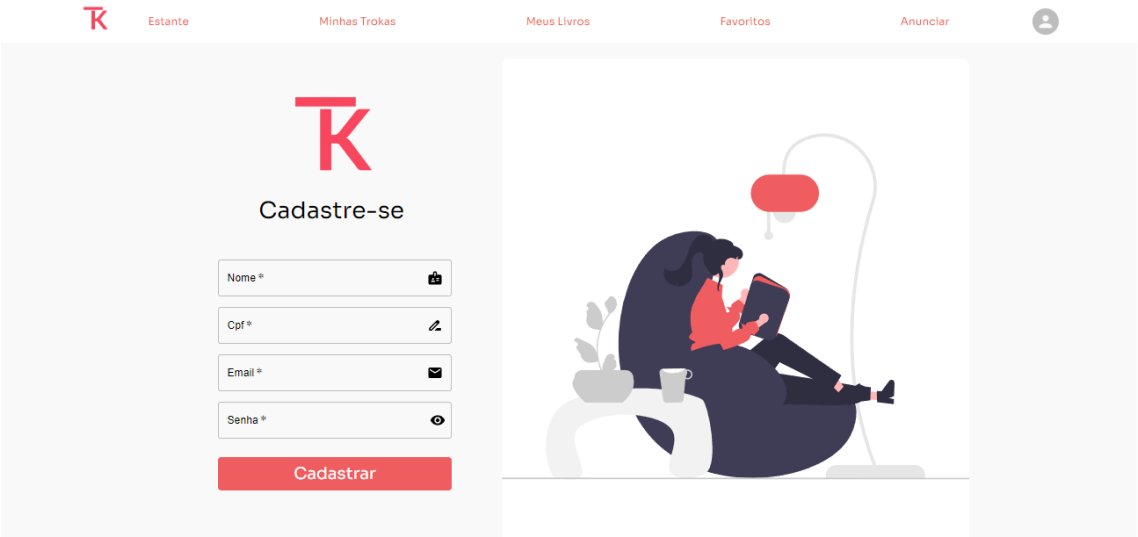

Figura 4. Tela de cadastro.

Na tela de cadastro de livro, os usuários têm a opção de inserir um novo livro, desde que preencham os campos necessários. Os campos obrigatórios incluem uma foto do livro, título, autor, ISBN (Padrão Internacional de Numeração de Livro), editora, razão da troca, categoria, tipo de capa (normal ou dura), idioma e preço de compra (opcional), além da data da compra do livro (opcional). Esses campos garantem que as informações essenciais sobre o livro sejam registradas no sistema. A figura a seguir exemplifica essa tela de cadastro de livro.

A tela de login do sistema requer o preenchimento do e-mail e senha para acessar. Além disso, a tela apresenta botões para recuperar a senha caso seja esquecida, bem como para criar uma nova conta. Esses recursos proporcionam aos usuários a possibili-

dade de realizar o login de forma segura e oferecem opções para resolver problemas de acesso, como esquecimento de senha, além de permitir que novos usuários se registrem no sistema.

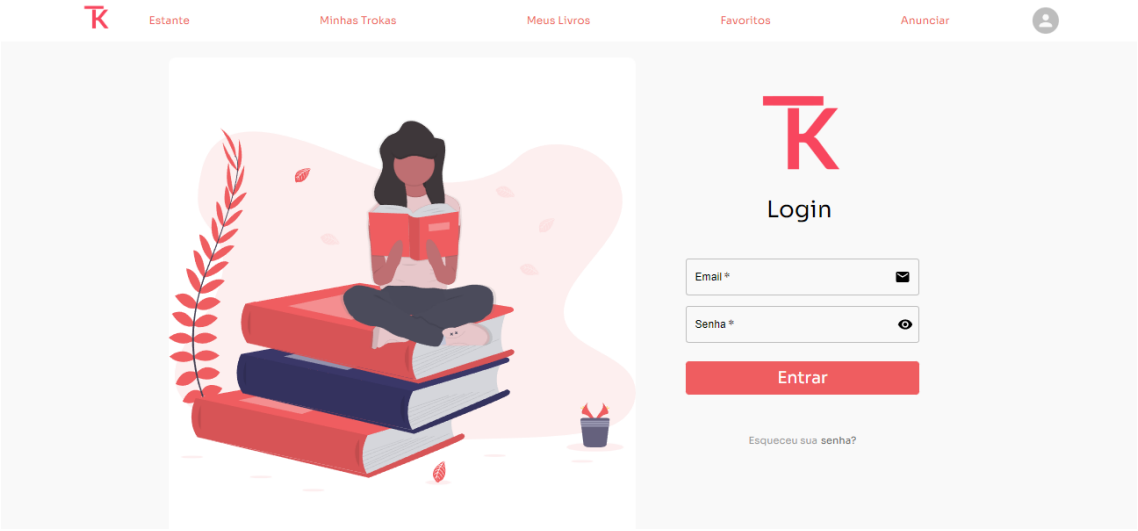

Figura 5. Tela de entrar.

O *front-end* da parte *web* foi desenvolvido na forma de um site, visando atender aos usuários que optem por não baixar ou utilizar o aplicativo mobile. O site possui uma tela de listagem de livros, chamada de ”estante”, que permite aos usuários pesquisar livros por nome e aplicar filtros, como por categoria, por exemplo. A tela exibe todos os livros disponíveis no site, oferecendo também a opção de ajustar a quantidade máxima de livros listados por página. Essa funcionalidade proporciona aos usuários uma experiência conveniente e flexível na busca e visualização dos livros disponíveis no site. O sistema *web* de troca de livros, denominado TrokaTroka, foi pensado e idealizado como uma forma simples, direta e segura para a troca eficaz de livros. Pensando assim, a equipe trabalhou em conjunto para conseguir se organizar e desenvolver algumas documentações do projeto no intuito de facilitar na construção do projeto como um todo.

Dessa maneira, o projeto correu bem possuindo diversos documentos para clarear o que estava sendo desenvolvido pelo grupo. Dos documentos mencionados no parágrafo anterior foram criados diversos documentos baseando-se em metodologias ágeis, entre eles está presente a jornada de usuário.

Um documento dedicado exclusivamente ao comportamento do usuário dentro do nosso sistema, nele é possível encontrar pontos iniciais, desde seu cadastro e discorrendo entre os principais passos que o usuário fará para realizar um anúncio ou solicitação de troca. Passando pelo login no sistema, identificações necessárias, ao cadastrar um livro, juntamente com a visualização de algum produto desejado, seguida da solicitação de troca e da sua confirmação, assim um local é marcado entre os usuários para realização da troca.

A figura a seguir mostra o código fonte da entidade Rating, responsável por armazenar os dados de uma avaliação. Essa entidade possui os seguintes atributos: nota, comentário, identificador do avaliador, identificador do avaliado e usuário avaliado. Esses atributos são essenciais para registrar e gerenciar as informações relacionadas às avaliações realizadas pelos usuários. O código fonte da entidade Rating permite a

manipulação e o acesso aos dados dessa avaliação, facilitando a implementação de funcionalidades e análises relacionadas ao sistema de avaliação.

```
namespace TT.Book.Domain.Entities;

11 referências
public class Rating : BaseEntity
{
    1 referência
    public Rating(int grade, string comment, Guid idRater, Guid idRated)
    {
        Grade = grade;
        Comment = comment;
        IdRater = idRater;
        IdRated = idRated;
    }

    4 referências
    public int Grade { get; private set; }
    2 referências
    public string Comment { get; private set; }
    3 referências
    public Guid IdRater { get; private set; }
    4 referências
    public Guid IdRated { get; private set; }

    1 referência
    public User RatedUser { get; set; }

    0 referências
    public Rating()
    { }
}
```

Figura 6. Rating.

A arquitetura utilizada foi a de microsserviços, foi produzido os artefatos para a documentação do desenvolvimento tanto do *back-end* quanto do *front-end*, abaixo é possível ver o documento arquitetural, na qual constam as linguagens de programação, bibliotecas, *frameworks*, plataformas e ferramentas utilizadas para a programação da aplicação.

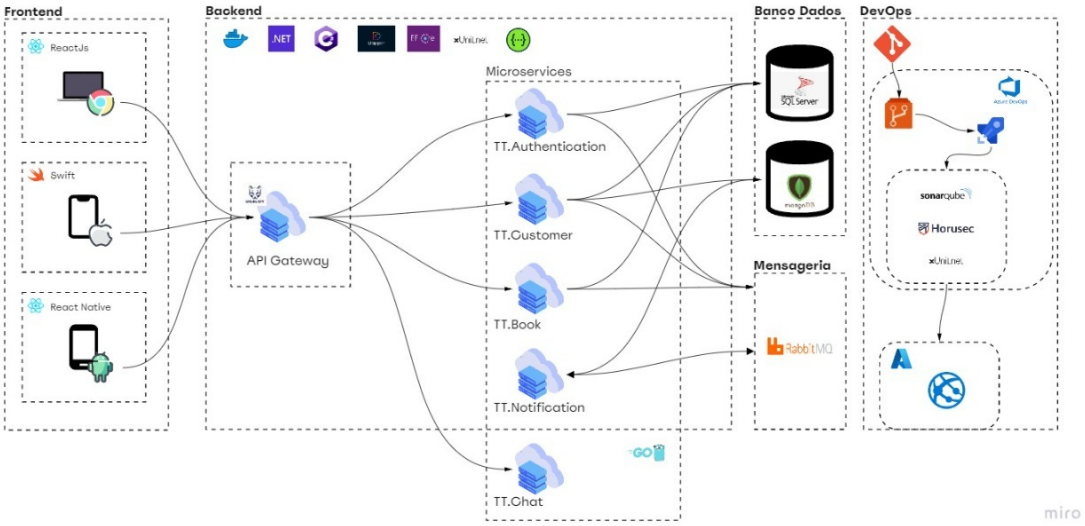

Figura 7. Arquitetura do sistema

Após a definição da arquitetura, foi realizada a programação das *APIs* que compõem a aplicação, foi desenvolvido as *APIs* para usuário, livros, trocas, autenticação, notificação e chat; foi escolhido um padrão de projeto na qual foi separado nas seguintes camadas: entidade, serviço, controle e repositório.

A camada controle recebe a requisição e direciona ao serviço responsável, enquanto a de serviço é responsável pelas regras de negócio de cada serviço, caso seja necessário persistir no banco de dados, a camada repositório é responsável pelo acesso ao banco de dados, e a entidade define os atributos, e faz o carregamento dos dados do objeto.

Portanto, o modelo lógico é uma das partes necessárias dentro do sistema como um todo, pois é a partir dele que o banco de dados será construído. Nele a equipe pensou nas diferentes interações dentro do sistema, dividindo o sistema em tabelas, cujo um atributo identificador foi atrelado a cada tabela dentro do sistema. Além da estrutura das tabelas e seus atributos identificadores, são definidas as relações entre as tabelas por meio de chaves estrangeiras. Elas são utilizadas para estabelecer a integridade e a consistência dos dados.

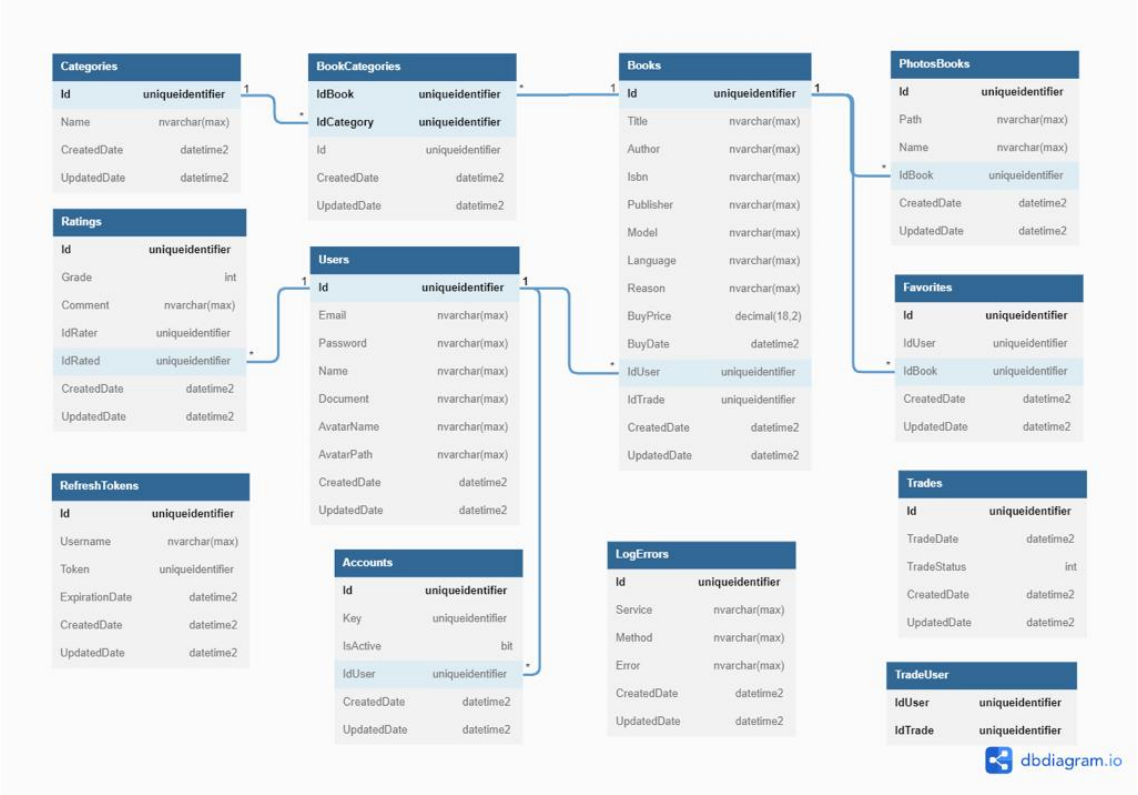

Figura 8. Modelo relacional do banco de dados.

Neste modelo podemos encontra a tabela Categories que possui uma relação de 1:N com a tabela CategoriaDeLivros onde pode ser acessado o delete o upload dos livros no sistema, por sua vez a tabela CategoriaDeLivros se comunica com a tabela Livros, cuja sua relação e N:1, logo Livros se comunica 1:N com outras 2 tabelas que são FotosLivros e Favoritos e possui uma comunicação N:1 com a tabela Usuários. A tabela Usuários se comunica com Avaliações e com Contas ambas com relações 1:N, e neste modelo achamos também tabelas independentes como Trocas, TrocasUsuário, LogErros e RefreshTokens.

No projeto foi acrescentado também um sistema de pontos, o seu objetivo é proporcionar uma experiência dinâmica e gratificante quando usado o nosso sistema de troca,

além de proporcionar um ambiente mais sustentável e econômico para os usuários. Assim que o cadastro de novos usuários é feito em nosso sistema os pontos são direcionados a eles de maneira automática como uma recompensa e incentivo para começar as trocas de livros. Esses pontos iniciais são extremamente úteis, pois é através deles que a troca será efetuada pelos usuários.

Ademais, há outras maneiras dos usuários conseguirem pontos no sistema, que será através de trocas realizadas. O sistema funcionará como um resgate de livros, o usuário colocará o livro no sistema e pedirá um valor (em pontos) nesse livro, assim outro usuário avaliará se o preço é justo e trocará o livro pelos pontos que tem. Os pontos não possuem validade, podendo assim ser acumulados e usados posteriormente. O sistema tem o intuito de ser bem simples e direto, deixar a leitura de maneira mais acessível a todos.

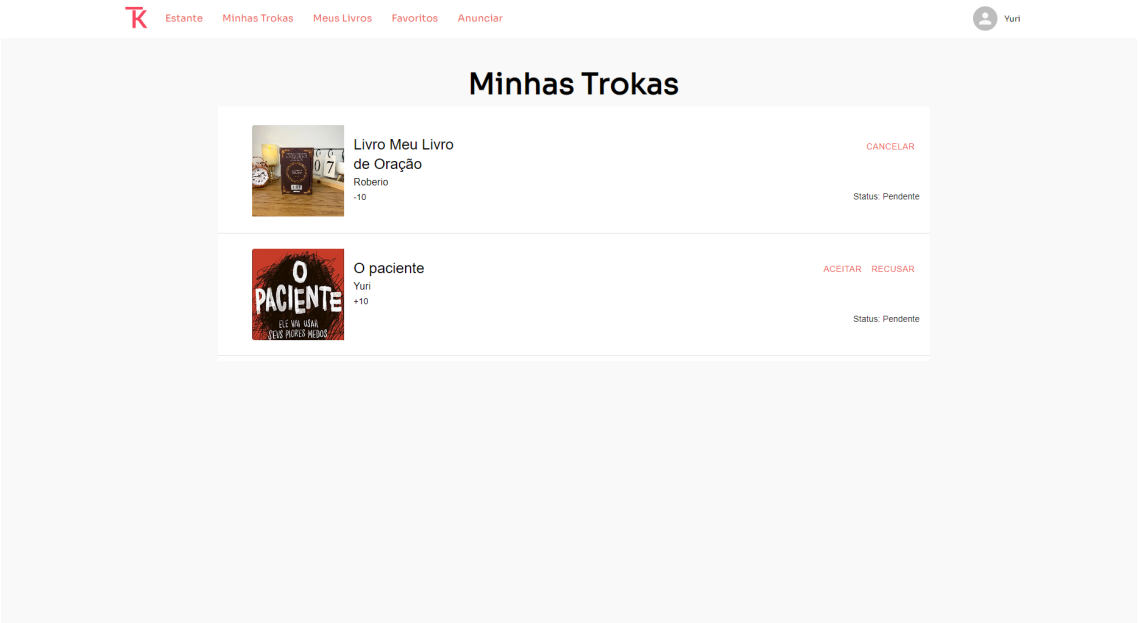

Figura 9. Listagem de trocas.

E para garantir a segurança uma das medidas de segurança adotadas durante o desenvolvimento do projeto é a utilização de um *gateway* para proteger os microserviços, evitando a exposição direta da infraestrutura interna. Isso ajuda a prevenir ataques diretos aos serviços, pois os usuários interagem com o sistema através do *gateway*, que atua como uma camada de proteção. Ademais, as senhas dos usuários são criptografadas antes de serem armazenadas no banco de dados. Isso garante que, mesmo em caso de violação do banco de dados, as senhas permaneçam protegidas e não possam ser facilmente decifradas. A criptografia é uma técnica essencial para garantir a confidencialidade dos dados sensíveis.

Além das medidas mencionadas, o sistema trokatroka também utiliza tokens de autenticação para validar a identidade dos usuários. Esses tokens são gerados durante o processo de autenticação e são necessários para acessar as funcionalidades do sistema. Também é importante destacar que durante o registro de uma conta, é necessário fornecer um e-mail válido e confirmá-lo para criar uma conta. Esses processos adicionais de

autenticação ajudam a garantir a segurança das contas dos usuários.

Portanto, uma medida que ajuda a aumentar a segurança é manter o software sempre atualizado, corrigindo possíveis vulnerabilidades de segurança. Para garantir a entrega contínua de um software, utilizou-se o Azure *pipeline*s, um serviço fornecido pela Microsoft. Por meio do Azure *pipeline*s, é possível automatizar o processo de construção, testes e implantação do aplicativo. Dessa forma, é criado um fluxo de trabalho automatizado que permite a integração e entrega contínua do software em desenvolvimento.

No contexto da entrega contínua utilizando o Azure *pipeline*s, é possível acompanhar o estado do *deploy* de cada release para verificar se a entrega foi bem sucedida. Além disso, há a opção de desfazer uma release caso seja necessário. A lista de releases inclui a entrega específica da release 32 (exemplificada abaixo), cujo estado pode ser verificado para avaliar o sucesso da implantação ou tomar medidas corretivas, se necessário.

Foi implementada uma *pipeline* para integração contínua e entrega contínua (CI/CD) no Azure DevOps, na qual segue o seguinte fluxo: baixa o código-fonte mais recente do repositório do projeto, restaura os pacotes NuGet necessários, se houver, compila o projeto e gera os arquivos binários, executa os testes de unidade e verifica se eles obtiveram êxito, publica os arquivos gerados (binários, *logs*, *test results*, entre outros) para análise posterior e no SonarQube, emite notificações para a equipe de desenvolvimento sobre o status da build em caso de falha, os desenvolvedores são notificados imediatamente para que possam corrigir o problema o mais rápido possível.

Referências

Pressman, R.; Maxin, B.(2016) Engenharia de Software. Porto Alegre: Grupo A. Sutherland, J. (2014) *Scrum*: The art of doing twice the work in half the time. São Paulo: Texto Editores Ltda.

Rigetti, S. J. F. (2019) Microsserviços prontos para a produção: Construindo sistemas padronizados em uma organização de engenharia de software. São Paulo: Novatec Editora.

Surhone, L. M.; Timpledom, M. T.; Marseken, S. F. (2010) RabbitMQ. Betascript Publishing.

Martin, R. C. Código Limpo. Rio de Janeiro: (2011) Alta Books. Anderson, D. J. Kanban: Mudança Evolucionária de Sucesso para seu Negócio de Tecnologia. Blue Hole Press.

Villaça, L. H.; Pimenta JR, A. F.; Azevedo, L. G. (2018) Construindo aplicações distribuídas com microsserviços. Tópicos em Sistemas de Informação: Minicursos XV Simpósio Brasileiro de Sistemas de Informação. SBC.

De Almeida, G. P.; Martins, E. A. (2019) Desenvolvimento de uma *web* API de controle de banco de dados para software financeiro de microempresas. In: VIII JORNACITEC-Jornada Científica e Tecnológica.

Sousa, N. T. S. (2022) Sistematização do desenvolvimento de interfaces *web*. Tese de Doutorado.

Hron, M.; Obwegeser, N. (2022) Why and how is *Scrum* being adapted in practice: A systematic review. Journal of Systems and Software, v. 183, p. 111110.

Bassett, L. (2015) Introdução ao JSON: Um guia para JSON que vai direto ao ponto. São Paulo: Novatec Editora.

Microsoft. (2022) Um tour pela linguagem C#. Microsoft. Disponível em: https://learn.microsoft.com/pt-br/dotnet/csharp/tour-of-csharp/. Acesso em: 15 nov. 2022.

Microsoft, Documentação do Microsoft SQL .Disponível em: https://learn.microsoft.com/pt-br/sql/?view=sql-server-ver16 .Acesso em: 15 nov. 2022.

MongoDB, Welcome to the MongoDB Documentation .Disponível em: https://www.mongodb.com/docs/.Acesso em: 15 nov. 2022.

Kreutz, D. L. et al. (2002) Análise de Desempenho de um SGBD para Aglomerado de Computadores.

Meta Plataforms. Documentação React. Disponível em : https://pt-br.reactjs.org/docs/getting-started.html. Acesso em: 15 nov. 2022

Beloki, U. H. (2022) Automation to Enable SRE with GitHub Actions/Azure DevOps/Azure Automation. In: The Art of Site Reliability Engineering (SRE) with Azure. Apress, Berkeley, CA. p. 91-142.

Thibault, Simon et al. (2022) NIRPS *back-end*: design and performance. In: Ground-based and Airborne Instrumentation for Astronomy IX. SPIE. p. 1652-1659.

Fenner, P. et al. (2014) A ORIENTAÇÃO A OBJETOS EM JAVA, C++ EC#: COMPARATIVO DE IMPLEMENTAÇÕES1. MISTRY, R.; MISNER, S. Introducing Microsoft SQL Server 2014. Microsoft Press,

Boczkowski, K.; Panczyk, B. (2020) Comparison of the performance of tools for creating a SPA application interface-React and Vue. js. Journal of Computer Sciences Institute, v. 14, p. 73-77.

Krief, M. (2019) Learning DevOps: The Complete Guide to Accelerate Collaboration with Jenkins, Kubernetes, Terraform and Azure DevOps. Packt Publishing Ltd,

Chandrasekara, C.; Herath, P. (2020) Getting Started with Azure Git Repos. In: Hands-on Azure Repos. Apress, Berkeley, CA. p. 139-170

Lenarduzzi, V. et al. (2020) Are sonarqube rules inducing bugs? In: 2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER). IEEE. p. 501-511.

Gotardo, Reginaldo Aparecido. Uma abordagem de sistema de recomendação orientada pelo aprendizado sem fim. 2014.