Desenvolvimento de Documentação e Implementação de Sistema de Informação para Auxílio na Aprendizagem de Qualidade de Software

Vinícius B. Bruscagini[1], **Carlos R. S. Júnior**[1]

[1]Instituto Federal de Educação, Ciência e Tecnologia do Estado de São Paulo
Campus Hortolândia (IFSP-HTO)

vinicius.bruscagini@aluno.ifsp.edu.br, carlos.rsantos@ifsp.edu.br

Abstract. *This paper has the objective to create an App using famous technologies and tools avaliable on the market. Applying software quality and project management concepts. A documentation covering all the aspects of the system will be released to help students and developers understand these concepts in a practical way.*

Resumo. *Este trabalho possui o objetivo de criar um sistema de informação utilizando tecnologias e ferramentas famosas no mercado, aplicando conceitos de qualidade de software e gerenciamento de projetos. Será disponibilizado um material que documentará todos os aspectos desse sistema, de forma a ajudar estudantes e desenvolvidores a entenderem como funcionam conceitos de qualidade de software na prática.*

1. Introdução

A área de TI, incluindo desenvolvimento de software, é uma área que evoluiu muito nas ultimas décadas (PACHECO; TAIT, 1) e continua evoluindo rapidamente até hoje. Novas Tecnologias e ferramentas surgem em rítmo acelerado e muitas delas ganham popularidade, demandando assim, novas habilidades para os profissionais de TI. Com essa evolução, os cursos oferecidos nessa área não conseguem oferecer disciplinas sobre tecnologias usadas no mercado. O ensino, então, não propicia o aprendizado suficiente para os profissionais de TI, fazendo com que muitos desses busquem cursos ou certificações para aprimorar seus conhecimentos. (MACEDO, 2011). Além disso, de acordo com uma pesquisa realizada por (MENDES et al., 2019) os alunos de cursos de Engenharia de Software se preocupam em possuir um aprendizado voltado para prática e sempre ter convivência com conceitos e métodos importantes para o mercado de trabalho.

Baseando-se nessas questões, a produção desse trabalho tem a proposta de desenvolver um sistema com tecnologias e ferramentas populares que possuem demanda por profissionais, com o objetivo principal de construir uma documentação que abrange todos os aspectos desse sistema e que explique com clareza como o software funciona, quais tecnologias foram usadas e como se deu o seu desenvolvimento. O código fonte da aplicação e a documentação serão publicados e então esses materiais poderiam ser consultados por qualquer pessoa, assim ajudando elas a entenderem alguns conceitos de Desenvolvimento de Software e como se dão suas aplicações na prática.

2. Fundamentação Teórica

Para atingirmos um software conforme descrito na seção anterior, precisamos entender alguns conceitos importantes dessa área.

2.1. Framework

Um conceito importante no desenvolvimento de sistemas é o 'Framework', de acordo com (HOSTGATOR, 2020) um framework é como um template que conta com diversas funcionalidades. Ele possui o objetivo principal de resolver problemas recorrentes no desenvolvimento e acelerar esse processo. O Projeto Pedagógico do Curso de Análise e Desenvolvimento de Sistemas do IFSP - Campus Hortolândia contém uma disciplina que ensina o uso de um framework para o desenvolvimento. No entanto, a bibliografia básica que aborda esse framework é do ano de 2010 e faz uso da tecnologia JSF, que hoje em dia é muito pouco usada. (WEB TECH SURVEY, 2021)

2.2. Qualidade de Software

A definição de qualidade no contexto da computação nem sempre é um consenso, existem diferentes terminologias, as quais podem causar problemas para pessoas que não possuem conhecimento sobre. (DUARTE; FALBO, 2000)

Um software pode ser considerado de qualidade quando ele atende a todas as necessidades, explicitas e implícitas para qual ele foi feito. (DUARTE; FALBO, 2000)

Enquanto ao código, que será o foco do artigo, existem padrões que devem ser seguidos para um código possuir qualidade. Para isso devemos pontuar que para um código ter qualidade ele deve funcionar, e principalmente, ser légivel.

> "Muitos programadores iniciantes acham que um bom código é aquele que é correto, ou seja, faz o que tem que fazer." (SIQUEIRA, 2018).

Um código, para funcionar deve possuir ao menos duas características, ele deve ser **Correto** e **Eficiente**, esses pontos são o que fazem um código funcionar. Porém isso não é o suficiente para ser considerado um código de qualidade. No mundo real, códigos são criados, alterados e revistos, geralmente por diferentes pessoas, por isso existem duas propriedades adicionais que um código deve possuir, ele deve ser **Elegante** e **Testável** como descritos por (SIQUEIRA, 2018).

Um código elegante e testável dá qualidade ao nosso código pois isso facilita, e muito, sua manutenção.

2.3. Controle de Versão

Controle de versão é a prática de rastrear e gerenciar mudanças no código do software (ATLASSIAN, 2020). Essas ferramentas gerenciam o código fonte de projetos ao longo do tempo, e possuem informações detalhadas de todas as modificações que foram realizadas em todos os arquivos do código fonte. Alguns dos benefícios que essas ferramentas proporcionam são rastreabilidade (quem fez o quê), ajudam na resolução de conflitos e aceleram o desenvolvimento. A ferramenta mais famosa para controle de versão atualmente é o Git.

2.3.1. Git

O Git é a ferramenta mais usada para controle de versão. De acordo com seu website, ele funciona com o conceito de *commits* e *branches*, em que um *commit* é uma alteração realizada no código, e uma *branch* é uma ramificação do código, elas são exemplificadas na figura 1.

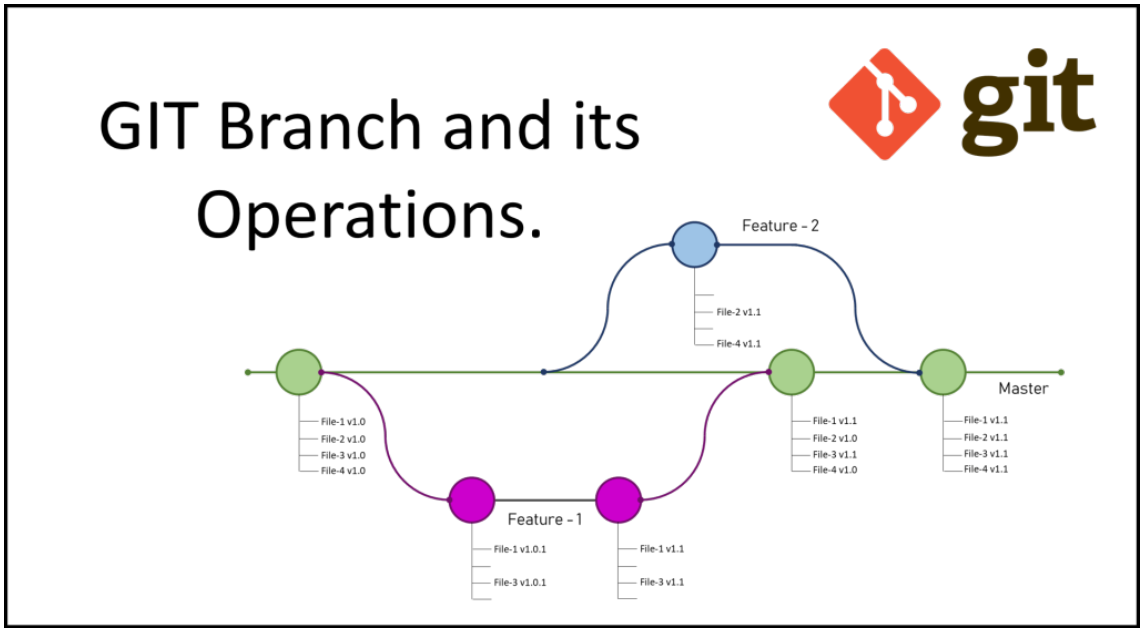

Figura 1. Exemplos de branches em um repositório Git

A figura 1 mostra três *branches* de um repositório, uma chamada *master*, outra de *Feature 1* e uma outra chamada *Feature 2*.

Durante o desenvolvimento de um software, essas ramificações acontecem frequentemente. Nesse exemplo, há duas ramificações que foram criadas para desenvolver uma funcionalidade no software, também chamado de *feature*, ou seja, a partir de uma branch principal *Master*, foram criadas duas *branches* onde o desenvolvimento de uma funcionalidade foi feita, e ao final desse desenvolvimento, o código feito nessa *branch* foi unido junto com a *branch* principal.

Com o Git instalado em um computador podemos usar o comando **git** para realizarmos operações. Um típico fluxo de um desenvolvimento usando Git é mostrado na figura 2.

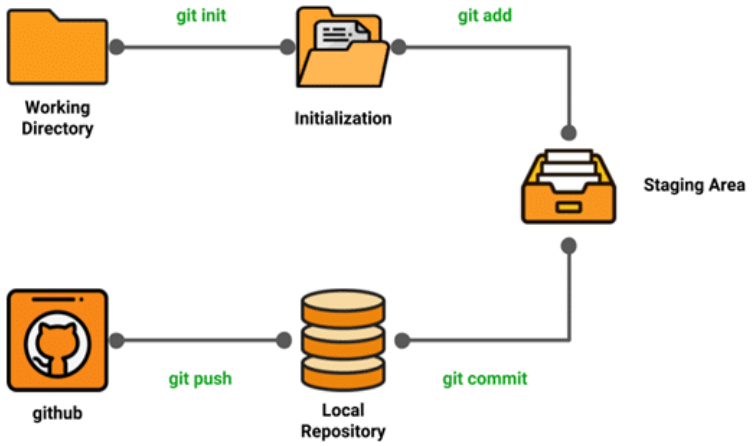

Figura 2. Típico ciclo de desenvolvimento com Git

O Git funciona em repositórios, que é uma pasta onde estão os arquivos de código de um sistema, a partir desse repositório, modificações são feitas no código, as quais são adicionadas em um *commit* e esse *commit* é enviado para um repositório remoto, onde outras pessoas da equipe podem obter as modificações feitas.

2.4. Documentação de Software

De acordo com (FORWARD, 2002), a documentação de um software é qualquer artefato que possui como finalidade informar sobre ele, em qualquer um de seus aspectos.

(COELHO, 2009) nos mostra dois tipos de documentações, são esses tipos

- Documentação Técnica
- Documentação de Uso

As documentações técnicas de um sistema é toda aquela documentação voltada ao desenvolvedor e é ela que informa como o sistema funciona internamente, sua arquitetura, tecnologias usadas e outros detalhes de implementação.

As documentações de uso são documentos que são focados nos usuários finais do sistema e as vezes também para administratores do sistema, essas documentações geralmente mostram como usar o sistema.

Ambos os tipos de documentações são muito importantes no processo de desenvolvimento e entrega de softwares. A falta ou baixo nível de qualidade de documentações atrapalham na compreensão do sistema e podem apresentar riscos para sua manutenção (SOUZA et al.,).

2.5. Metodologias Ágeis

As metodologias ágeis surgiram na década de 80 para melhorar a área de desenvolvimento de software, que era algo muito rigoroso naquela época, muitos projetos possuiam problemas e as metodologias ágeis foram então criadas para tentar solucioná-los. (SANTOS; MARINHO, 2018)

2.5.1. Kanban

O Kanban é uma metodologia que possui cinco princípios de acordo com (AGILE, 2013)

- Visualização
- Limitar o desenvolvimento em progresso
- Medir e gerenciar a sequencia de trabalho
- Explicitar o processo
- Reconhecimento de oportunidades

O Kanban funciona com quadros, onde em cada quadro temos várias seções. Nessas seções podemos agrupar tarefas.

Em um quadro Kanban podemos agrupar tarefas, definir datas e responsáveis, entre vários outros detalhes, o que facilita a organização e visão das tarefas do projeto.

Existem várias ferramentas que implementam essa metodologia. Uma delas é o *GitHub Projects*, que usaremos neste trabalho. É usada para gerenciamento de projetos com a metodologia Kanban. A interface desse programa é mostrada na Figura 3.

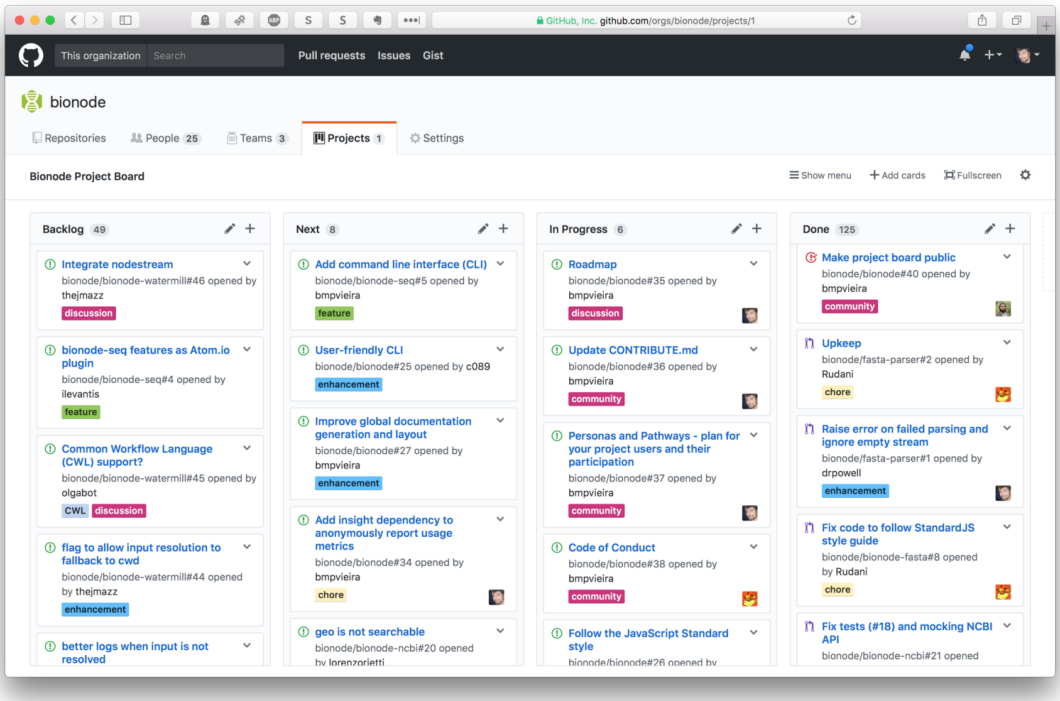

Figura 3. Tela de um quadro Kanban do *GitHub Projects*

3. Metodologia

Para início do desenvolvimento será realizada a definição das tecnologias do sistema, para isso, serão feitas pesquisas bibliográficas em websites como *GitHub* e *Stackoverflow*. Estes websites são famosos na área de desenvolvimento, além de que eles podem nos fornecer informações interessantes sobre quais são as tecnologias mais ativas, mais

comentadas e outros dados que podemos usar como base para realizamos a escolha das tecnologias.

Com a definição das tecnologias, partiremos para a análise e documentação de requisitos do sistema, e a partir dos requisitos serão criados os diagramas de classe e casos de uso.

Para o desenvolvimento, usaremos os recursos oferecidos pela plataforma GitHub, esses recursos serão

- Repositório Git
- Quadro Kanban para gerenciamento de projetos
- Actions para processos de *Pipelines*

4. Desenvolvimento do Trabalho

Em Maio de 2021 foi realizada pelo site (STACKOVERFLOW, 2021), uma pesquisa com desenvolvedores de todo o mundo para coletar dados geográficos, sociais e de uso de tecnologias de seus usuários. Esses dados nos permitem ter informações importantes sobre o uso de tecnologias em geral. A pesquisa nos dá três conjuntos de informações relevantes para usarmos no trabalho

- Frameworks para Web mais utilizados
- Frameworks gerais mais utilizados
- Ferramentas para deploy e gerenciamento mais usadas
- Banco de Dados mais utilizados

Os resultados dessas categorias, entre desenvolvedores profissionais, são exibidos nas Figuras 4, 5, 6 e 7, respectivamente.

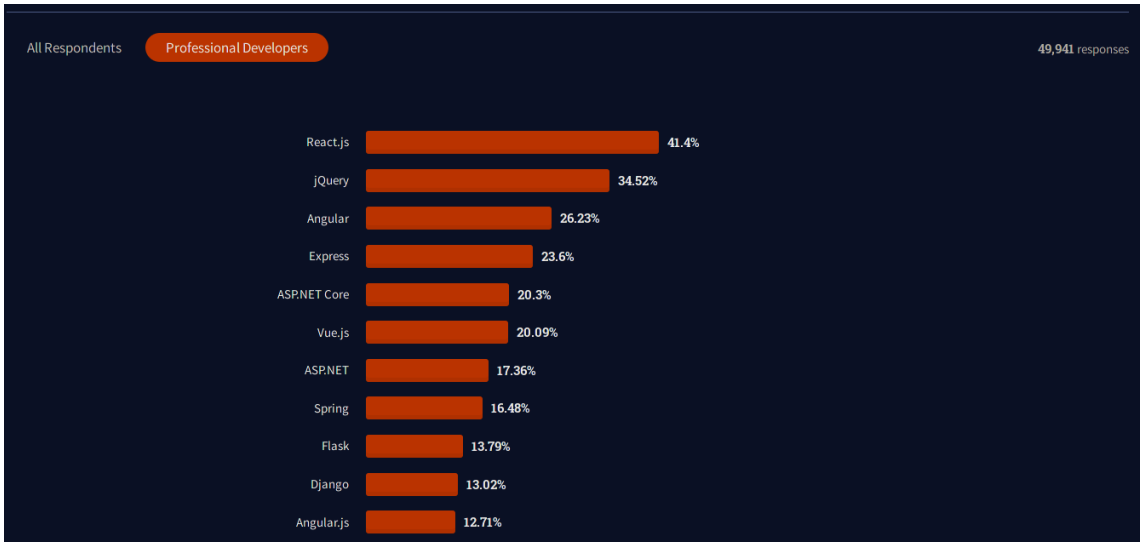

Figura 4. Gráfico mostrando os frameworks Web mais usados

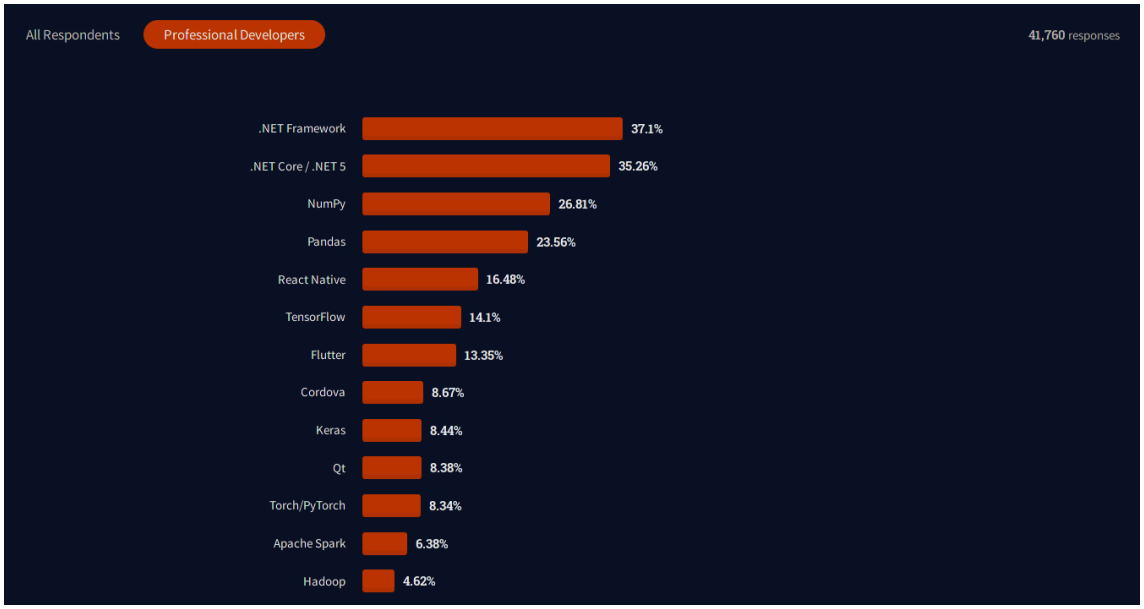

Figura 5. Gráfico mostrando os frameworks gerais mais usados

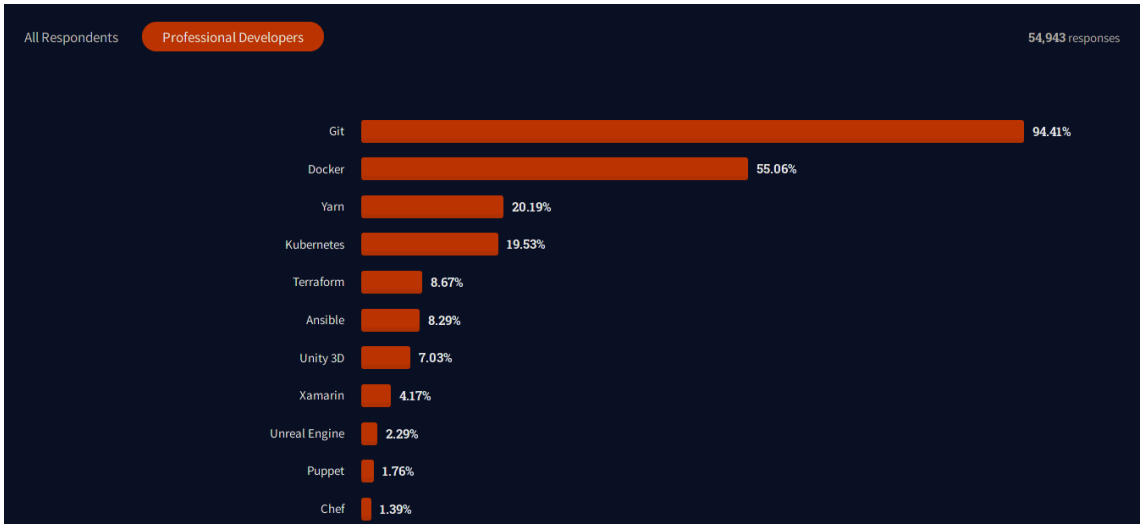

Figura 6. Gráfico mostrando as ferramentas mais usadas

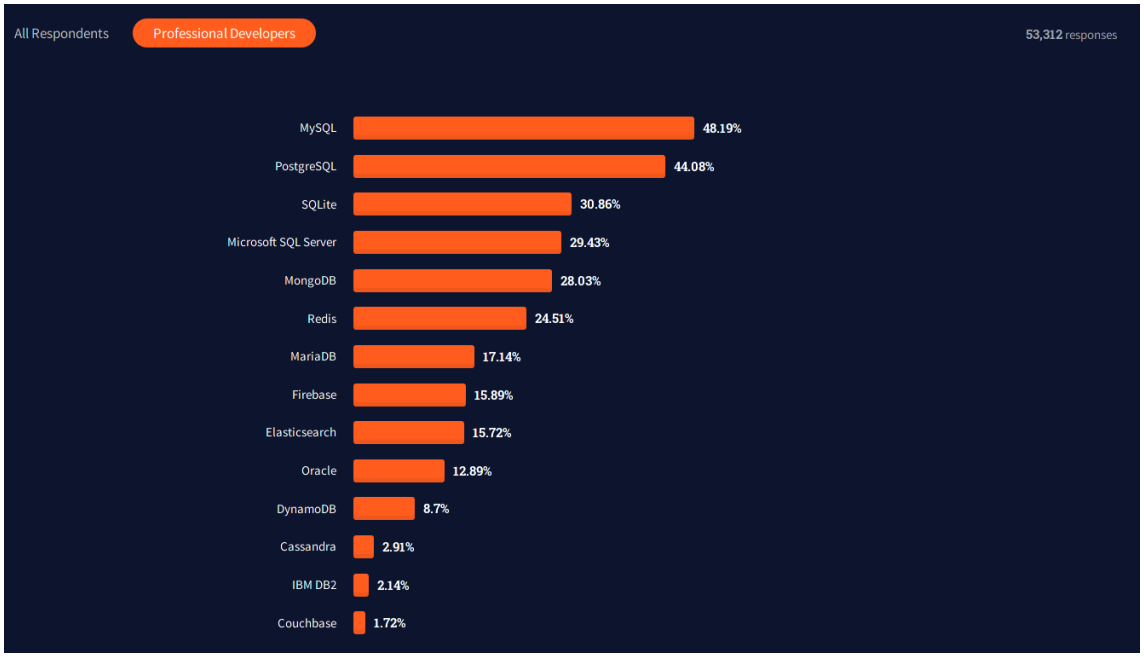

Figura 7. Gráfico mostrando os SGBDs mais usados

Baseando-se nesses dados de utilização e na experiência do autor em certas tecnologias e ferramentas, foi optado o uso das tecnologias e arquitetura apresentadas a seguir.

4.1. Arquitetura

O sistema terá duas aplicações principais, uma sendo o lado do servidor (Back-End) e outra sendo a aplicação que será usada pelos usuários (Front-End).

O Back-End será um WebService RESTFul, que será desenvolvido com o ASP.NET Core junto com a ferramenta de ORM, Entity Framework Core.

o Front-End será uma aplicação Web que será desenvolvida com o Framework Vue.js.

4.2. WebService

Um WebService é um serviço que é oferecido e é por onde aplicações se comunicam com o servidor. WebServices fazem parte da arquitetura orientada a serviços (SOA).

Um tipo de WebService muito utilizado é o REST, que significa *Representational State Transfer* ele funciona servindo requisições de clientes para certos serviços. Por exemplo, suponha que um WebService REST tenha uma URL que fornece informações de funcionários

```
GET http://meuwebservice/funcionarios
```

GET é o método HTTP usado para requisitar as informações que queremos, então se enviarmos uma requisição para a URL com o método esperado, o WebService irá responder a requisição com dados em formato JSON com as informações que queremos, por exemplo:

```
[
  {
    "nome": "João",
    "departamento": "Contabilidade"
  },
  {
    "nome": "Bianca",
    "departamento": "Engenharia"
  }
]
```

Um WebService REST pode ter vários URLs, também chamadas de EndPoints, além de obter informações, podemos realizar diferentes operações em diferentes End-Points, como inserir um objeto no sistema, por exemplo.

4.3. ASP.NET Core

O ASP.NET Core faz parte da família de tecnologias .NET que são desenvolvidas pela Microsoft. Ele nos permite criar qualquer tipo de aplicação web.

A Plataforma .NET possui uma longa história, por grande parte dessa história, suas tecnologias foram exclusivas para o sistema operacional Windows. Em 2014 a Microsoft recriou essa plataforma do zero, hoje toda a plataforma é *open-source* e funciona em qualquer sistema operacional.

4.4. Entity Framework Core

O Entity Framework Core é um framework ORM que é usado em conjunto com o .NET. Como explicitado por (O'NEIL, 2008)

> 'Um framework ORM provê uma metodologia e mecanismo para sistemas orientados a objetos manterem seus dados em um banco de dados de maneira segura, com controle de transações, e tudo isso expressado em código orientado a objeto'.

De maneira prática, um ORM permite manipularmos entidades em um banco de daddos sem precisarmos mexer com SQL. Eles geralmente funcionam mapeando as classes que criamos em nosso código, e com essas informações ele cria o banco de dados já com os relacionamentos entre as tabelas e tudo mais o que definimos nas classes.

O Entity Framework é o ORM oficial do .NET, possui amplo suporte e uso. Ele será usado nesse projeto para facilitar a manipulação dos dados no banco de dados.

4.5. Vue.js

O Vue é um framework para desenvolvimento de aplicações Web, usando a linguagem JavaScript, ele nos permite o desenvolvimento de aplicações reativas e possibilita o reuso de código através de componentes. Abaixo um pequeno código HTML de um arquivo vue

```
<div>
  <p v-if="seen">Agora você me viu</p>
</div>
```

Esse trecho de código exibe o texto ‘Agora você me viu’ caso a variável ‘seen’ seja verdadeira.

O Vue nos dá a opção de usar a linguagem TypeScript ao invês do JavaScript, ele possui várias bibliotecas que melhoram a legibilidade e arquitetura de um componente vue. Abaixo um código de um componente Vue usando TypeScript

```
<template>
  <p>Texto</p>
</template>

<script lang="ts">
import Vue from "vue";
import Component from "vue-class-component";

@Component
export default class About extends Vue {

}
</script>
```

O Vue possui um ecosistema de bibliotecas adicionais para ajudar no desenvolvimento de aplicações. Uma dessas bibliotecas é o *Vuetify*, ele fornece para o desenvolvedor componentes para interface como botões, cards, campos de textos, etc.

4.6. Gerenciamento

Durante o desenvolvimento do projeto, serão usadas as funcionalidades da plataforma GitHub, isso inclui o repositório Git para controle de versão e o quadro Kanban para gerenciamento de Projetos.

4.7. Definição de Requisitos

Com o processo e as tecnologias definidas. Agora será decidido o domínio do sistema a ser desenvolvido e seus requisitos.

Como o foco do trabalho não é o domínio da aplicação, foi optado por copiar o funcionamento de uma aplicação existente, a aplicação escolhida é o Reddit.

O Reddit, de acordo com seu website é:

> ‘A casa de milhares de comunidades, conversas sem limite e interação humana autentica.’

O Reddit consiste de comunidades que tratram de determinado assunto, nessas comunidades, usuários podem se inscrever e realizar postagens, as quais outros usuários dessa comunidade virão e podem dar um voto positivo ou negativo para essa postagem.

Foram criados dois diagramas de casos de uso que representam as funcionalidades esperadas no sistema, eles são apresentados nas figuras 8 e 9

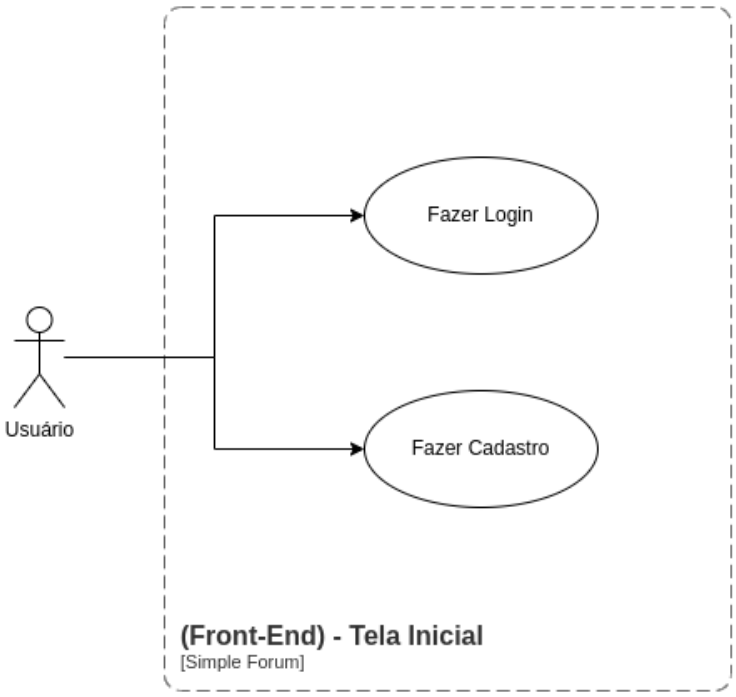

Figura 8. Diagrama de caso de uso da tela inicial

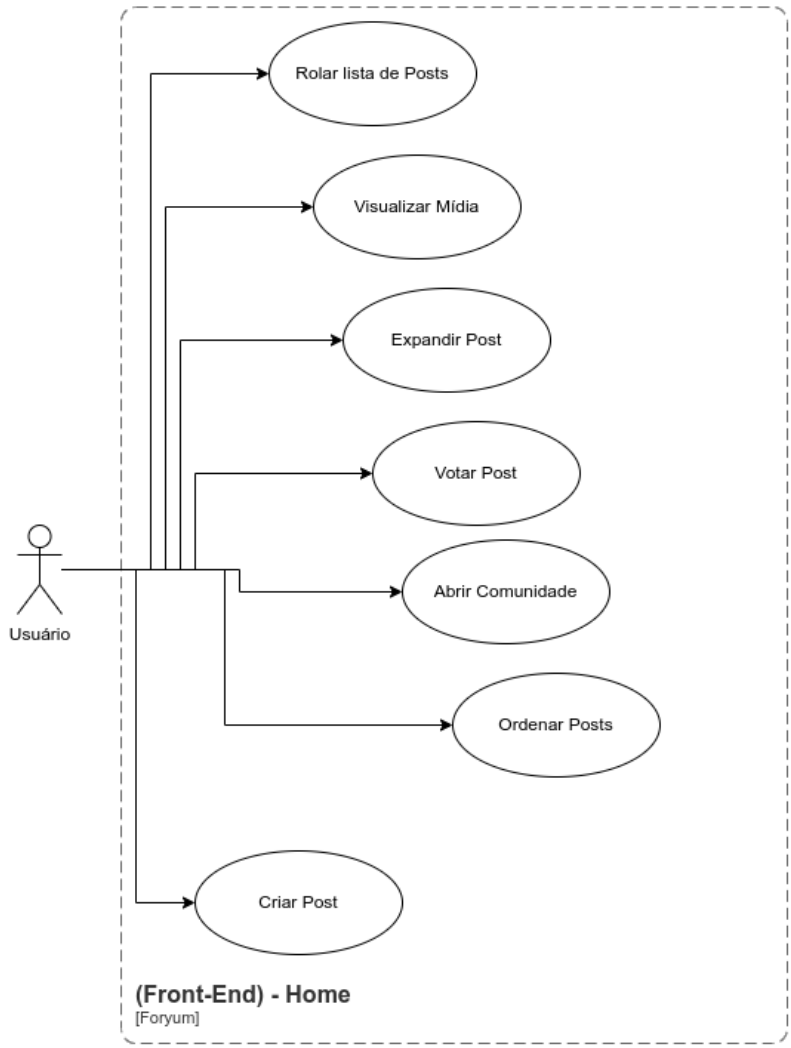

Figura 9. Diagrama de caso de uso da tela *home*

Junto com os casos de uso foram definidos as propiedades das entidades do sistema a partir da criação de um diagrama de classe, o diagrama resultante é exibido na figura 10

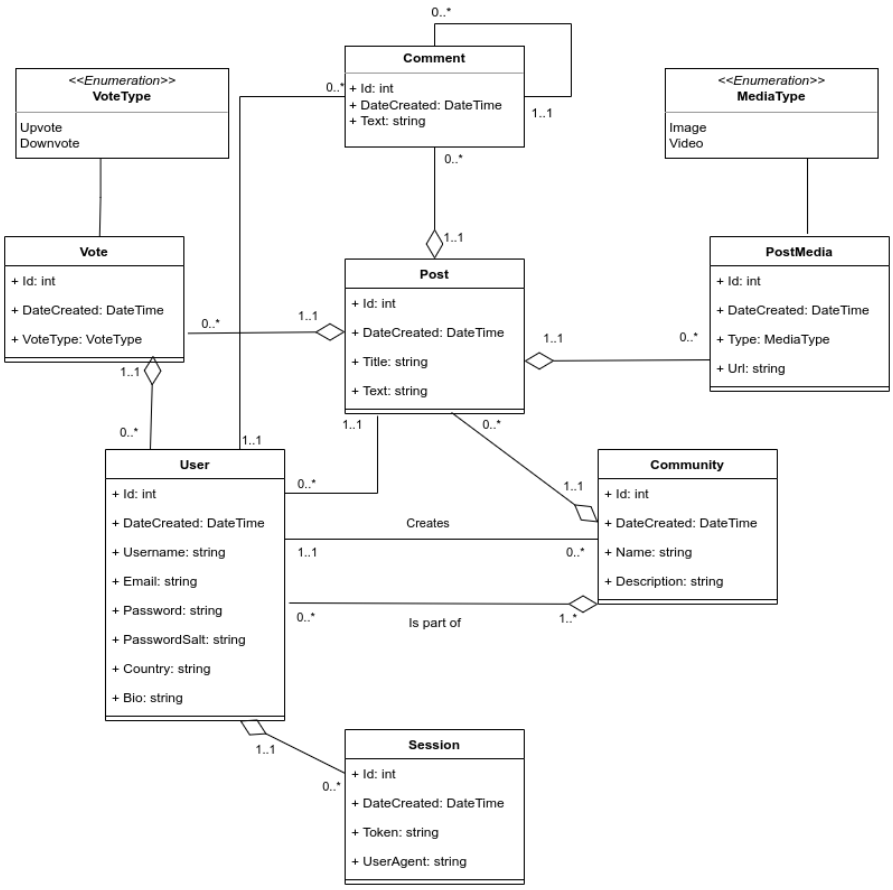

Figura 10. Diagrama de classes do sistema

4.8. Ambiente de Desenvolvimento

A ferramenta de desenvolvimento escolhida para implementação das aplicações foi o *Visual Studio Code* em conjunto com extensões disponíveis no seu repositório para facilitar seu uso com as tecnologias escolhidas para o projeto. Além da própria ferramenta de desenvolvimento foi necessário a instalação dos kits de desenvolvimento (SDK) das tecnologias que foram utitlizadas. Foram eles o *dotnet* para a implementação do Back-End, em conjunto com o *Node.js* e *Vue CLI* para implementação do Front-End.

Foi utilizado, como auxílio para o desenvolvimento do Back-End que é uma API REST, o *software Insomnia REST Client* para realização de testes.

Para a criação do banco de dados MySQL foi utilizado a ferramenta *Docker* para seu gerenciamento via contâiner, e para acesso e manipulação desse MySQL foi utilizado o software *Dbeaver*. Além dessas ferramentas, foi utlilizado o site *mockaroo* que oferece scripts para preenchimento de banco de dados com informações de testes. Facilitando o processo de desenvolvimento.

4.9. Implementação

Como mencionado anteriormente, foi utilizado a metodologia Kanban para gerenciamento de tarefas, o desenvolvimento seguiu de maneira confortável tarefa por tarefa cri-

ada no quadro. Algumas mudanças foram realizadas no escopo conforme o necessário durante o desenvolvimento.

4.10. Documentação

Além da implementação do sisema será criada uma documentação técnica do sistema em uma linguagem simples para pessoas nova da área de TI entenderem como funciona o sistema.

5. Resultados

No estado atual do trabalho, o desenvolvimento está em rítimo acelerado e alguns dos requisitos propostos foram implementados, dois *prints* do sistemas são exibidos nas figuras 11 e 12

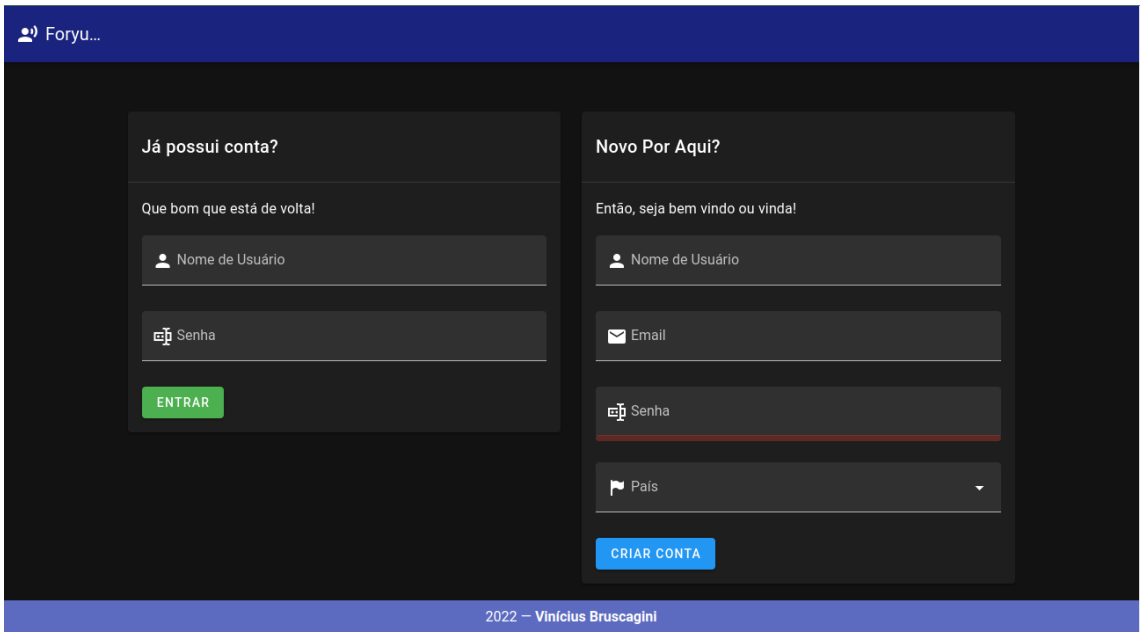

Figura 11. Tela de bem-vindo

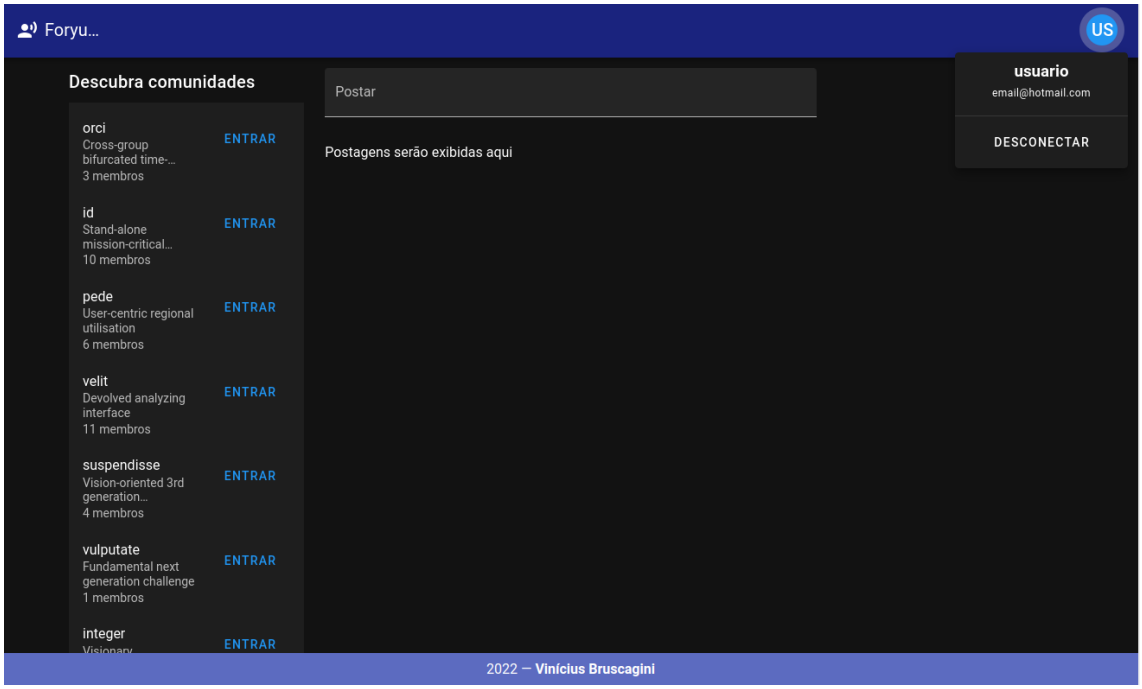

Figura 12. Tela *home*

6. Conclusão

O artigo mostrou problemas existentes no ensino de desenvolvimento de software e propôs a criação de um sistema com uma documentação abrangente para ajudar estudandes sobre alguns dos conceitos e tecnologias usadas no desenvolvimento de software, além de prover uma abordagem prática sobre o assunto. O Trabalho também explicou sobre conceitos de qualidade e sobre as ferramentas que serão usadas no desenvolvimento do software proposto no trabalho.

Referências

AGILE. *ITNOW*, v. 55, n. 2, p. 6–8, 05 2013. ISSN 1746-5702. Disponível em: ⟨https://doi.org/10.1093/itnow/bwt002⟩.

ATLASSIAN. *What is version control?* 2020. Disponível em: ⟨https://www.atlassian.com/git/tutorials/what-is-version-control⟩.

COELHO, H. S. Documentação de software: uma necessidade. *Texto Livre: Linguagem e Tecnologia*, v. 2, n. 1, p. 17–21, jun. 2009. Disponível em: ⟨https://periodicos.ufmg.br/index.php/textolivre/article/view/16562⟩.

DUARTE, K. C.; FALBO, R. d. A. Uma ontologia de qualidade de software. In: *Workshop de Qualidade de Software, João Pessoa*. [S.l.: s.n.], 2000. p. 275–285.

FORWARD, A. *Software Documentation – Building and Maintaining Artefacts of Communication*. [S.l.], 2002.

HOSTGATOR. *Framework o que é, quais utilizar e como eles funcionam!* 2020. Disponível em: ⟨https://www.hostgator.com.br/blog/frameworks-na-programacao/⟩.

MACEDO, M. C. B. O mercado de trabalho em tecnologia de informação: a inser ção profissional dos desenvolvedores de software. 2011.

MENDES, J. et al. Identificação das expectativas e dificuldades de alunos de graduação no ensino de engenharia de software. In: *Anais do XXVII Workshop sobre Educação em Computação*. Porto Alegre, RS, Brasil: SBC, 2019. p. 334–347. ISSN 2595-6175. Disponível em: ⟨https://sol.sbc.org.br/index.php/wei/article/view/6640⟩.

O'NEIL, E. J. Object/relational mapping 2008: Hibernate and the entity data model (edm). In: *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA: Association for Computing Machinery, 2008. (SIGMOD '08), p. 1351–1356. ISBN 9781605581026. Disponível em: ⟨https://doi.org/10.1145/1376616.1376773⟩.

PACHECO, R.; TAIT, T. Tecnologia de informaÇÃo: EvoluÇÃo e aplicaÇÕes. *Revista Teoria e Evidência Econômica*, v. 8, n. 14, 1 1. Disponível em: ⟨http://seer.upf.br/index.php/rtee/article/view/4816⟩.

SANTOS, E. V.; MARINHO, G. T. A implantação de metodologias ágeis para o desenvolvimento de software: Dificuldades e recomendações. 10 2018. Disponível em: ⟨https://www.sncticet.ufam.edu.br/2018/anais/download/artigos/ARTIGO_4.pdf⟩.

SIQUEIRA, F. L. Qualidade de código. 03 2018. Disponível em: ⟨www.levysiqueira.com.br/artigos/qualidade-de-codigo.pdf⟩.

SOUZA, S. C. B. de et al. Investigação da documentação de maior importância para manutenção de software.

STACKOVERFLOW. *Stack Overflow Developer Survey*. 2021. Disponível em: ⟨https://insights.stackoverflow.com/survey/2021⟩.

WEB TECH SURVEY. *JavaServer Faces*. 2021. Disponível em: ⟨https://webtechsurvey.com/technology/javaserver-faces⟩.