Desenvolvimento de documentação e implementação de sistema de informação para contribuir na formação do perfil de analista de sistemas

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. *Regarding difficulties in universities for adapting emerging technologies into curriculum, this paper has the objective to create a technical software documentation based in a developed software that uses consolidated market technologies. Containing simple and comprehensive language to contribute with the formation of IT students.*

Resumo. *Com a dificuldade de adequação das instituições de ensino em relação a novas tecnologias, este trabalho possui como principal objetivo disponibilizar uma documentação técnica de um software desenvolvido com tecnologias consolidadas no mercado. Com uma linguagem simples e compreensível para colaborar na formação de alunos dos cursos de Análise e Desenvolvimento de Sistemas.*

1. Introdução

A área de TI (Tecnologia da Informação), é uma área que evoluiu muito nas ultimas décadas [Pacheco and Tait 1] e continua evoluindo rapidamente. 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, há uma dificuldade por parte das instituições de ensino para ofertar conteúdos sobre essas habilidades para estudantes, 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 como 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. Assim ajudando estudantes a entenderem alguns conceitos de desenvolvimento de software e como se dão suas aplicações na prática.

Esse artigo possui a seguinte organização: A seção 2 nomeada Fundamentação Teórica, evidencia em subseções os conceitos empregados durante o desenvolvimento deste trabalho. Na sequencia, a seção 3 nomeada Metodologia, descreve o processo realizado para esse desenvolvimento. A seção 4, Desenvolvimento do Trabalho, mostra as

etapas de desenvolvimento para chegar ao objetivo do trabalho, divididos em subseções. A seção 5 de Resultados, mostra o que foi implementado e documentado de acordo com o objetivo do trabalho. Por último a seção 6, Conclusão, resume todo o artigo e mostra possíveis trabalhos futuros.

2. Fundamentação Teórica

Nessa seção são apresentados conceitos relacionados ao desenvolvimento de *software* e gerenciamento de projetos. Esses conceitos foram empregados para realizar o desenvolvimento do *software* conforme descrito na seção anterior.

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 (*Java Server Pages*), que hoje em dia é muito pouco usada. [Survey 2021]

2.2. 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": "Aline",
    "departamento": "Engenharia"
  },
  {
    "nome": "João",
    "departamento": "Finanças"
  }
]
```

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

2.3. 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 and 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 and 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.

2.4. 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*.

2.4.1. 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.

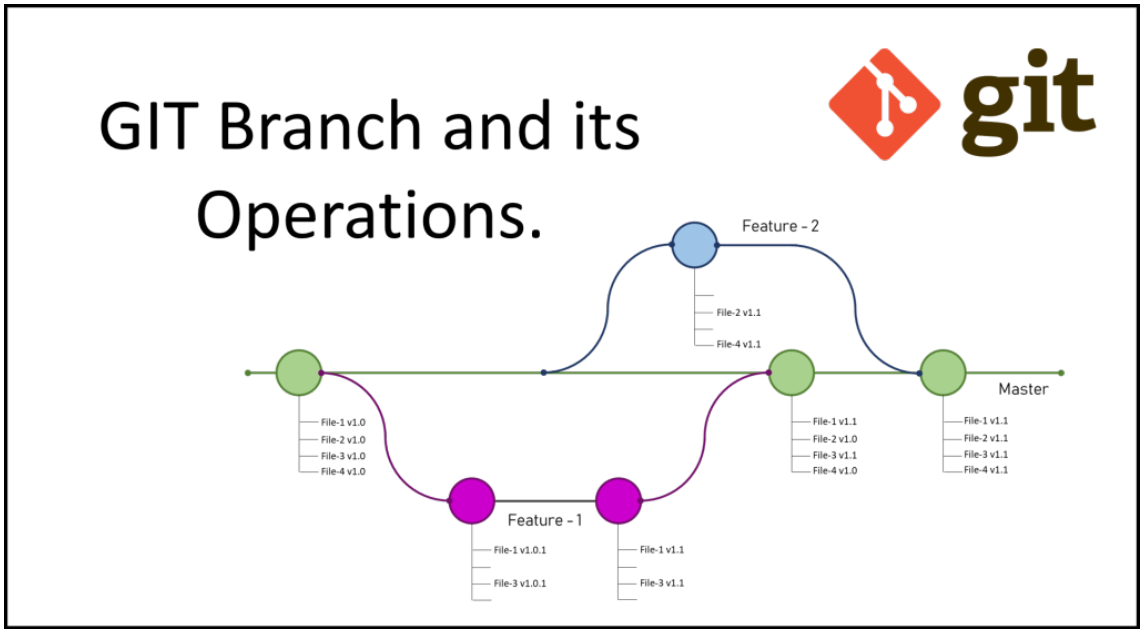

Figure 1. Exemplos de branches em um repositório Git

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 *software* 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.

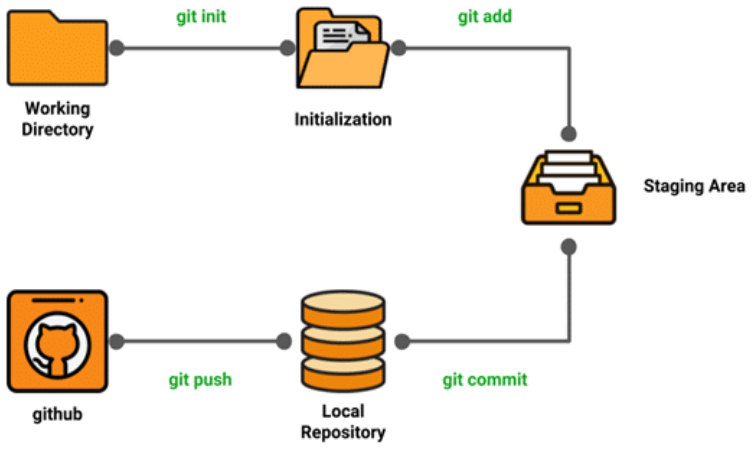

Figure 2. Típico ciclo de desenvolvimento com Git

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.

2.5. 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 [de Souza et al.].

2.6. 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 and Marinho 2018]

2.6.1. 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.

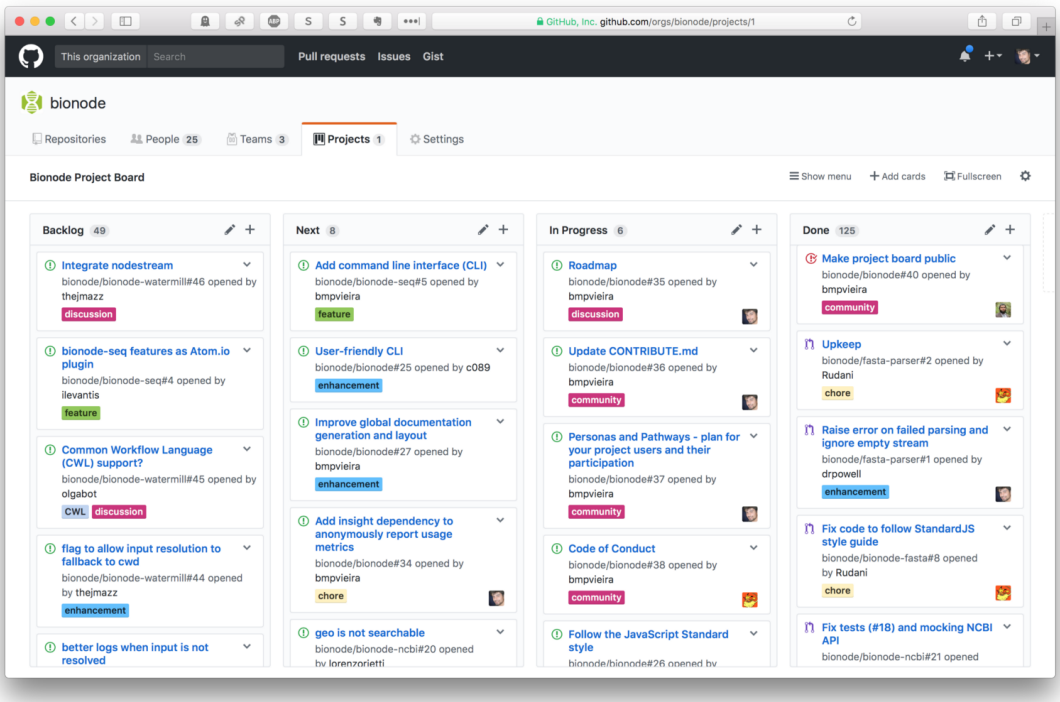

Figure 3. Tela de um quadro Kanban do *GitHub Projects*

3. Metodologia

Para início do desenvolvimento foi realizada a definição das tecnologias do sistema, para isso, foram feitas pesquisas bibliográficas em *WebSites* como *GitHub* e *Stackoverflow*. Estes *WebSites* são famosos na área de desenvolvimento, eles forneceram informações importantes sobre quais são as tecnologias mais ativas, mais comentadas e outros dados que foram usados para a escolha das tecnologias usadas no desenvolvimento do sistema.

Com as tecnologias definidas, o próximo passo foi a análise e documentação de requisitos do sistema, e com esses requisitos, foram criados diagramas de classes e casos de uso.

Para o desenvolvimento, foram usados recursos oferecidos pela plataforma *GitHub*, esses recursos são:

- Repositório *Git*
- Quadro *Kanban* para gerenciamento de projetos
- *Wiki* onde será criada a documentação técnica do sistema

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.

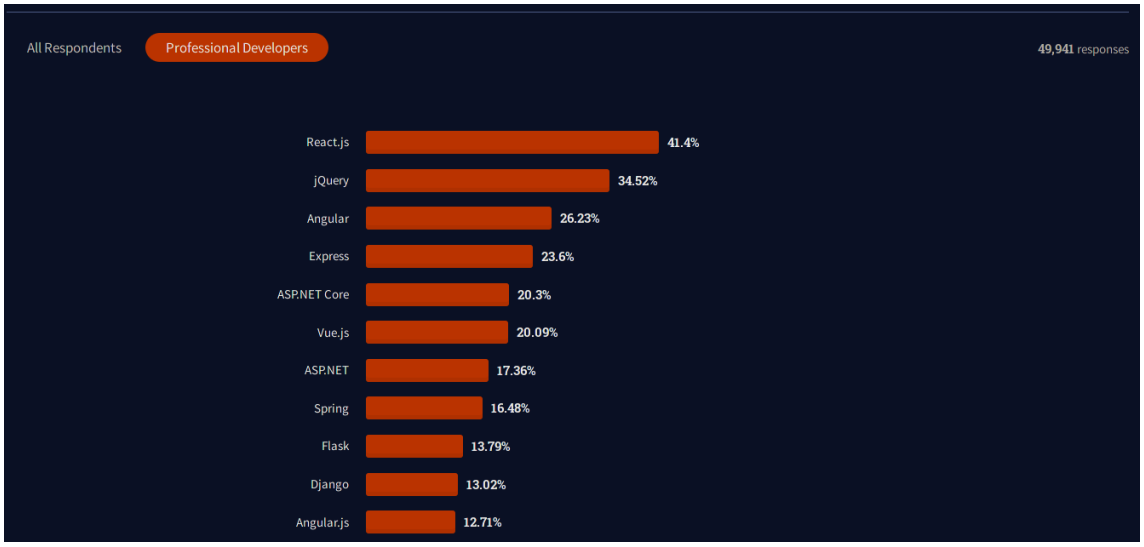

Figure 4. Gráfico mostrando os frameworks Web mais usados

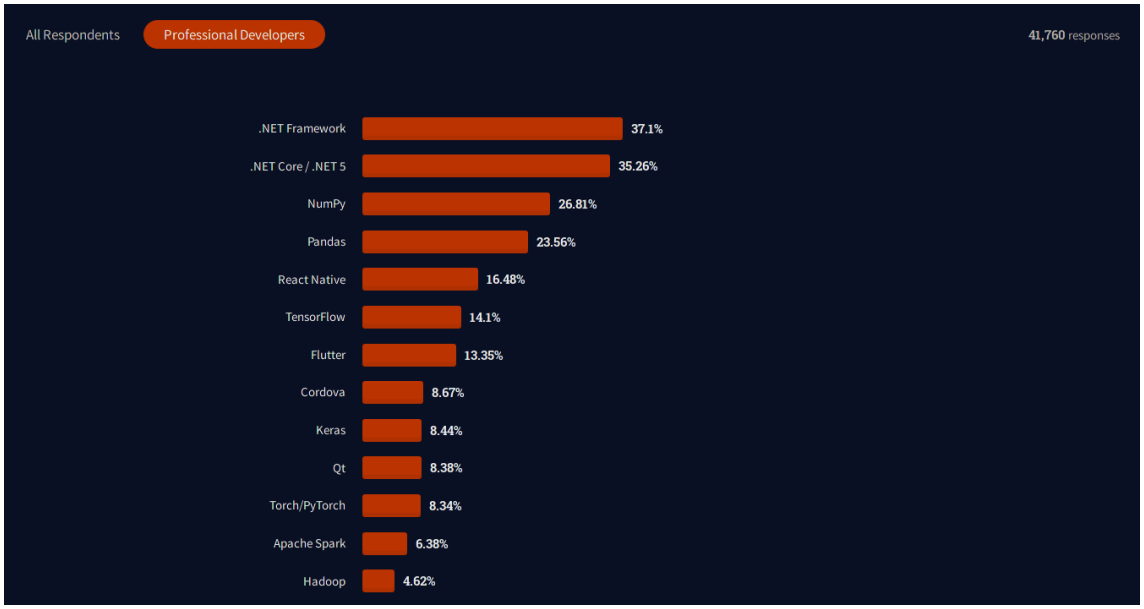

Figure 5. Gráfico mostrando os frameworks gerais mais usados

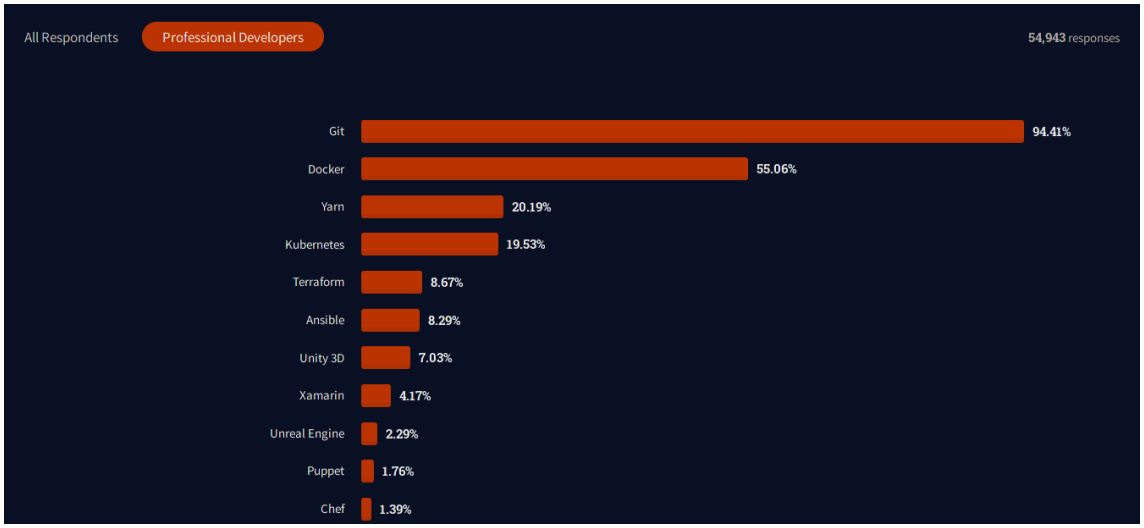

Figure 6. Gráfico mostrando as ferramentas mais usadas

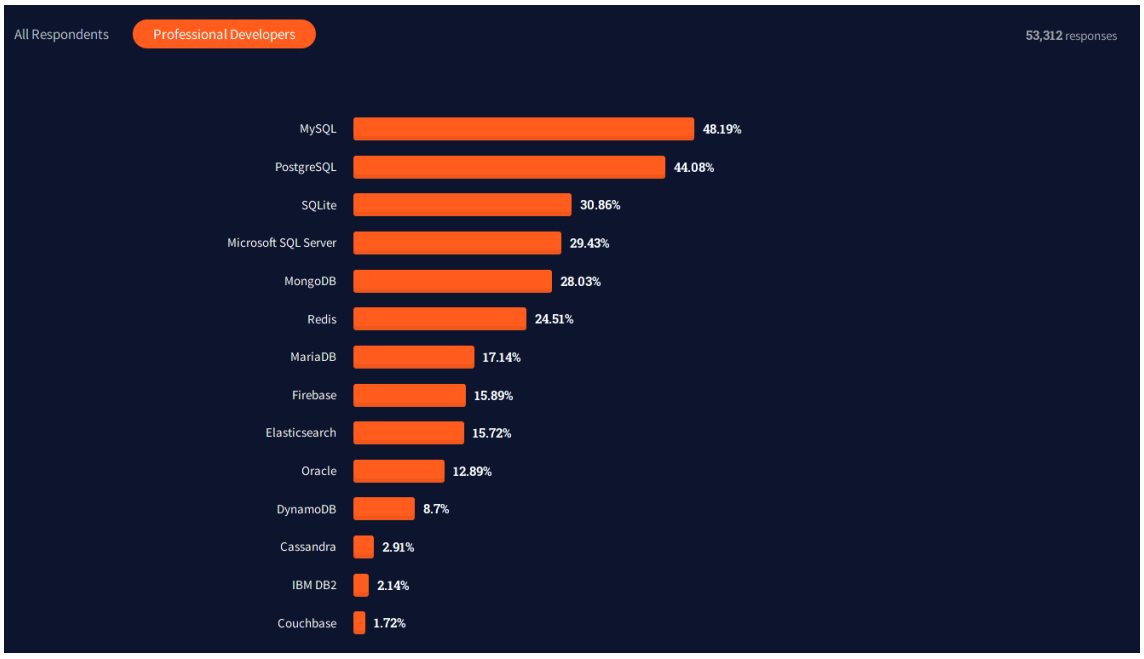

Figure 7. Gráfico mostrando os SGBDs mais usados

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 nas próximas seções.

4.1. Arquitetura e Tecnologias

A arquitetura escolhida para o sistema foi um *WebService REST*, como mostrado na seção 2.2. Além do *WebService* foi decidido a criação de uma aplicação cliente que acesse o *WebService*.

As tecnologias que foram usadas para a implementação dessas aplicações são apresentadas nas próximas seções.

4.1.1. 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.

4.1.2. 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 dados 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.

4.1.3. 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.

4.2. 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 tratam 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.

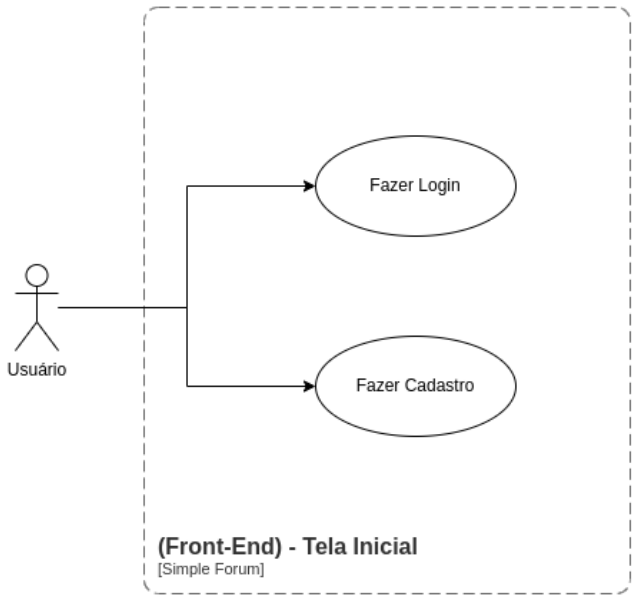

Figure 8. Diagrama de caso de uso da tela inicial

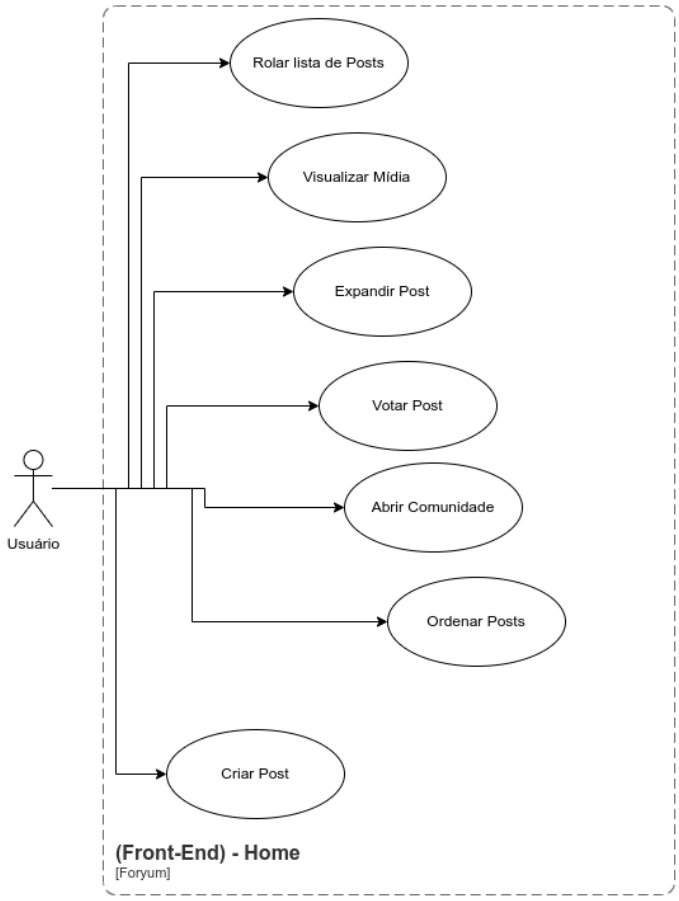

Figure 9. Diagrama de caso de uso da tela *home*

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.

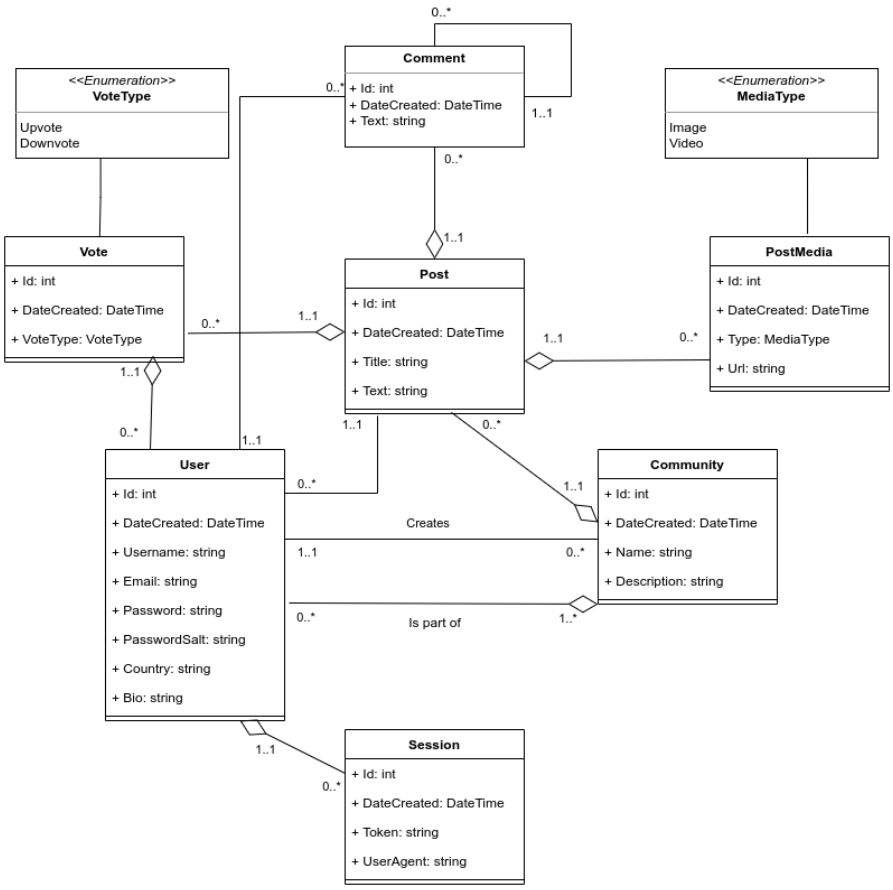

Figure 10. Diagrama de classes do sistema

4.3. 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 SDKs (Kit de Desenvolvimento) das tecnologias que foram utitlizadas. Foram eles o *dotnet* para a implementação do *WebService*, em conjunto com o *Node.js* e *Vue CLI* para implementação do cliente *web*.

Foi utilizado, como auxílio para o desenvolvimento do *WebService* 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.

4.4. Implementação

Como mencionado anteriormente, foi utilizado a metodologia *Kanban* para gerenciamento de tarefas, a implementação dos sistemas foram realizadas baseados nessas tarefas. Para implementação de algumas dessas tarefas houveram atrasos devido a necessi-

dade de aprendizagem para a resolução daquele problema técnico. Como por exemplo a implementação de sistema de segurança nas aplicações.

4.5. Documentação

Após a finalização da implementação das aplicações, foi construída uma documentação dos *softwares*, essa documentação foi feita na plataforma *GitHub* na funcionalidade de *Wiki*. Foi criada uma *Wiki* em cada repositório da aplicação, uma para a aplicação em *Vue,js* e outra para a API em *.NET*

5. Resultados

As duas aplicações foram implementadas junto com todas as funcionalidades propostas. Dois *prints* da aplicação *web* são exibidos nas Figuras 11 e 12.

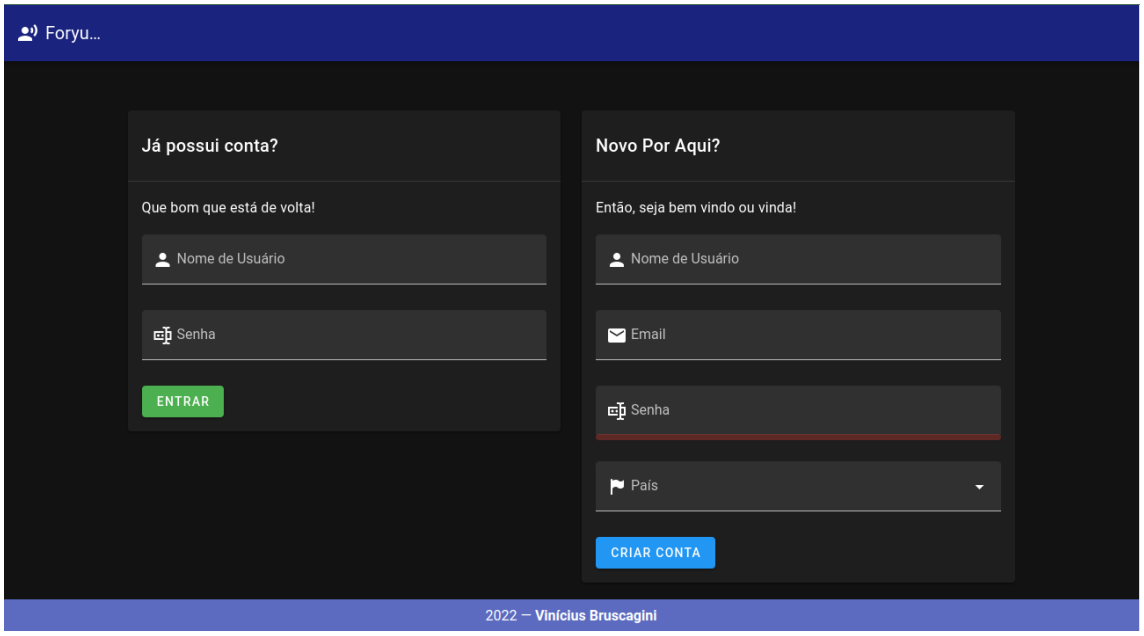

Figure 11. Tela de bem-vindo

Essa é a primeira tela da aplicação, onde o usuário pode fazer seu *Login* ou se cadastrar.

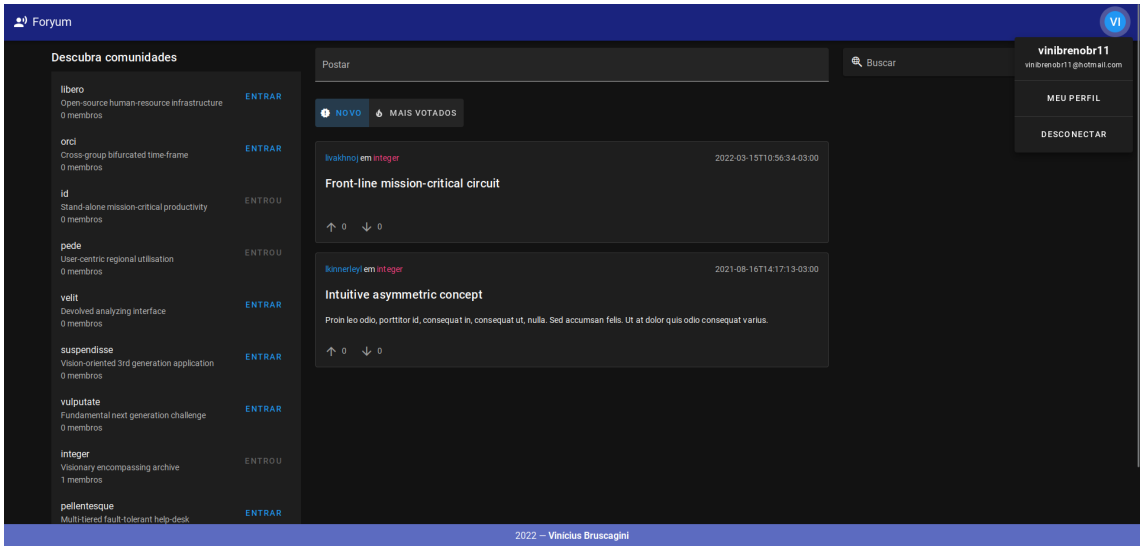

Figure 12. Tela *home*

Essa tela é a *Home*, é onde o usuário vê as postagens de outros usuários em comunidades. A partir dessa tela o usuário pode também entrar em comunidades, fazer postagens, ou realizar uma pesquisa.

O Desenvolvimento da API em *.NET* foi finalizadada e todas as funcionalidades usadas pela aplicação *Web* foram implementadas com restrições de acesso e esquemas de segurança, um *print* dos endpoints disponíveis nessa *API* é exibido na Figura 13 através da biblioteca *Swagger*.

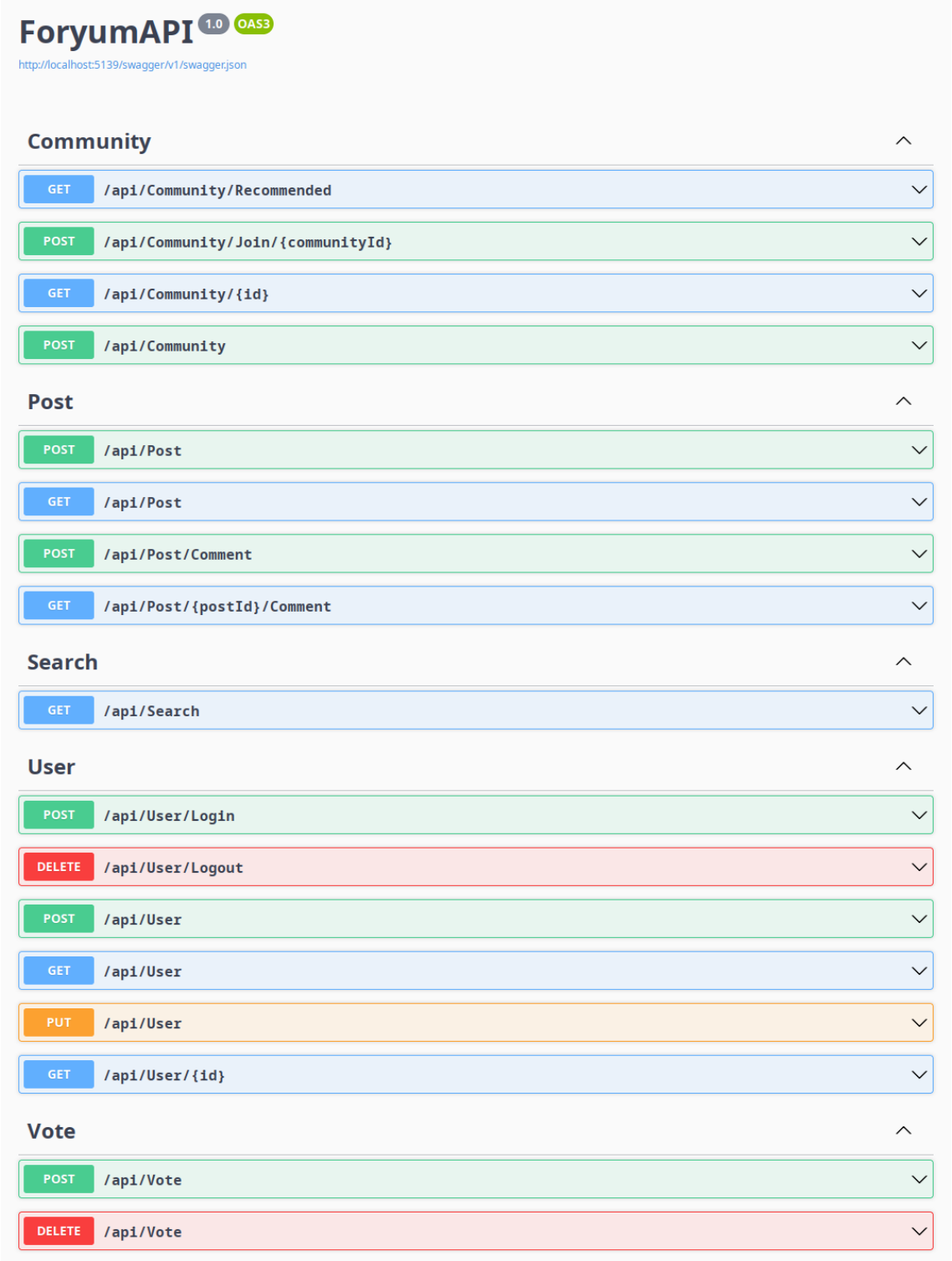

Figure 13. Print da tela do Swagger

Além do desenvolvimento, foi criada a documentação técnica dessas aplicações, elas estão na *Wiki* dos repositórios *Git*. Além dos arquivos *README* que mostram uma introdução das aplicações na tela do repositório.

Os repositórios estão localizados nos links especificados a seguir:

- https://github.com/V11-0/ForyumServer (WebService)
- https://github.com/V11-0/ForyumWeb (Aplicação Web)

6. Conclusão

O artigo mostrou a dificuldade do ensino em relação as novas tecnologias no desenvolvimento de *software* e propôs a criação de um sistema com uma documentação técnica para ajudar estudandes sobre alguns dos conceitos e tecnologias usadas no desenvolvimento de *software*, além de prover uma abordagem prática sobre o assunto. Foram implementados duas aplicações sendo um *WebService* com tecnologias *.NET* e uma aplicação *web* com o *framework Vue*. O código fonte e a documentação técnica dessas duas aplicações estão disponíveis *online* na plataforma *GitHub*.

Como trabalhos futuros, uma pesquisa de opinião poderia ser feita para estudantes de TI em relação ao material disponibilizado por este trabalho. Além da documentação, outros conceitos que não foram abrangidos nesse trabalho poderiam ser explorados como questão de *Pipelines* para realização de entrega contínua (*CI/CD*). Outra recomendação poderia ser o aprimoramento e criação de novos recursos para *software* criado nesse trabalho.

References

Agile (2013). Agile. *ITNOW*, 55(2):6–8.

Atlassian (2020). What is version control? https://www.atlassian.com/git/tutorials/what-is-version-control.

Coelho, H. S. (2009). Documentação de software: uma necessidade. *Texto Livre: Linguagem e Tecnologia*, 2(1):17–21.

de Souza, S. C. B., das Neves, W. C. G., de Oliveira, K. M., Anquetil, N., and Figueiredo, R. Investigação da documentação de maior importância para manutenção de software.

Duarte, K. C. and Falbo, R. d. A. (2000). Uma ontologia de qualidade de software. In *Workshop de Qualidade de Software, João Pessoa*, pages 275–285.

Forward, A. (2002). Software documentation – building and maintaining artefacts of communication. Technical report.

HostGator (2020). Framework o que é, quais utilizar e como eles funcionam! https://www.hostgator.com.br/blog/frameworks-na-programacao/.

Macedo, M. C. B. (2011). O mercado de trabalho em tecnologia de informação: a inser ção profissional dos desenvolvedores de software. Master's thesis, Universidade Federal do Rio Grande do Sul.

Mendes, J., Costa, Y., Frazão, K., Santos, R., Santos, D., and Rivero, L. (2019). Identificação das expectativas e dificuldades de alunos de graduação no ensino de engenharia de software. In *Anais do XXVII Workshop sobre Educação em Computação*, pages 334–347, Porto Alegre, RS, Brasil. SBC.

O'Neil, E. J. (2008). Object/relational mapping 2008: Hibernate and the entity data model (edm). In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, page 1351–1356, New York, NY, USA. Association for Computing Machinery.

Pacheco, R. and Tait, T. (1). Tecnologia de informação: Evolução e aplicações. *Revista Teoria e Evidência Econômica*, 8(14).

Santos, E. V. and Marinho, G. T. (2018). A implantação de metodologias ágeis para o desenvolvimento de software: Dificuldades e recomendações. *Anais da XII Semana Nacional de Ciência e Tecnologia*.

Siqueira, F. L. (2018). Qualidade de código. *http://www.levysiqueira.com.br*.

StackOverflow (2021). Stack overflow developer survey. https://insights.stackoverflow.com/survey/2021.

Survey, W. T. (2021). Javaserver faces. https://webtechsurvey.com/technology/javaserver-faces.