# 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.*** *Given the challenges for IT Universities to adapt their curriculum in regarding to new tecnologies due to the fast-growing-market, this paper has a main objective to make avaliable a technical documentation of a developed software with consolidated technologies in the market to collaborate with the formation of IT Students.*

***Resumo.*** *Dado o desafio das instituições de ensino da área de tecnologia em adequar os conteúdos abordados às novas tecnologias devido a rápida evolução do mercado, este trabalho tem como principal objetivo disponibilizar uma documentação técnica de um software desenvolvido com tecnologias consolidadas no mercado afim de colaborar na formacão de alunos do curso de Análise e Desenvolvimento de Sistemas.*

## 1. Introdução

A área de TI (Tecnologia da Informação), é uma área que evoluiu muito nas últimas décadas [Pacheco et al. 2000] 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, surge um desafio para as instituições de ensino atualizarem seus conteúdos, o principal fator para esse problema é a burocracia para a atualização das ementas de disciplinas. Isso faz com que muitos desses busquem cursos ou certificações para aprimorar seus conhecimentos [Macedo e Borba 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 em 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 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.

Neste trabalho, então, a solução desenvolvida possui o nome de 'Foryum' e esse sistema foi desenvolvido com tecnologias escolhidas baseadas em pesquisas que são detalhadas na seção 4, foram elas *.NET* e *Vue.js*. Além da solução técnica, foi criada a

documentação dessa solução, que é importante para o entendimento do código fonte. Esses materiais estão disponíveis na plataforma *GitHub* nos *links* mostrados na seção 5 de Resultados. Com o desenvolvimento de um *software* e de sua documentação, espera-se que a disponibilização desses materiais possa colaborar com estudantes dos cursos de Análise e Desenvolvimento de Sistemas pois esse material mostra como funciona um ambiente real de *software*.

Esse artigo possui a seguinte organização: a Seção 2 nomeada Fundamentação Teórica, evidencia em subseções os conceitos empregados durante o desenvolvimento deste trabalho. Na sequência, 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

De acordo com [HostGator 2020] um *framework* é como um template que conta com diversas funcionalidades que possui o objetivo principal de resolver problemas recorrentes no desenvolvimento e acelerar esse processo.

### 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* que 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, um exemplo de uma resposta *JSON* é mostrada na Figura 1.

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 et al. 2000].


**Figura 1. Exemplo de JSON como resposta**

Um *software* pode ser considerado de qualidade quando ele atende a todas as necessidades, explicitas e implícitas para qual ele foi feito [Duarte et al. 2000]. Além disso, um *software* é composto por um código fonte. Existem padrões que devem ser seguidos para um código possuir qualidade. Devemos pontuar que para isso, ele deve funcionar, e principalmente, ser legível.

De acordo com [Siqueira 2018]: "Muitos programadores iniciantes acham que um bom código é aquele que é correto, ou seja, faz o que tem que fazer".

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

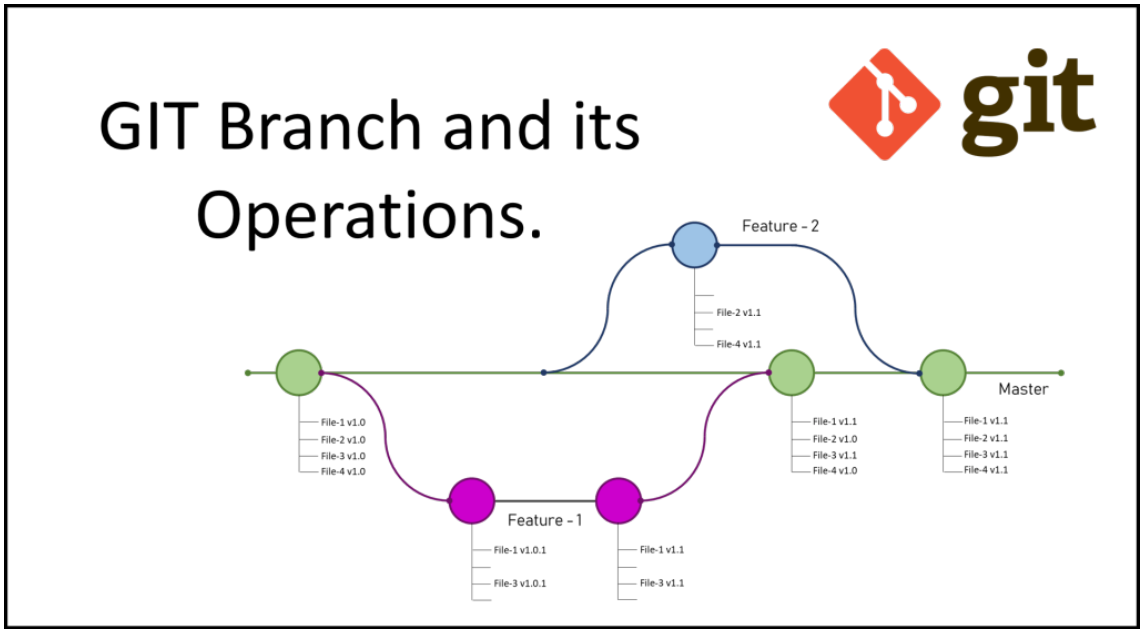


**Figura 2. Exemplos de branches em um repositório Git**

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 3.

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 e Simone 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, esse material 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 em administratores. Essas documentações geralmente mostram como usar o sistema.

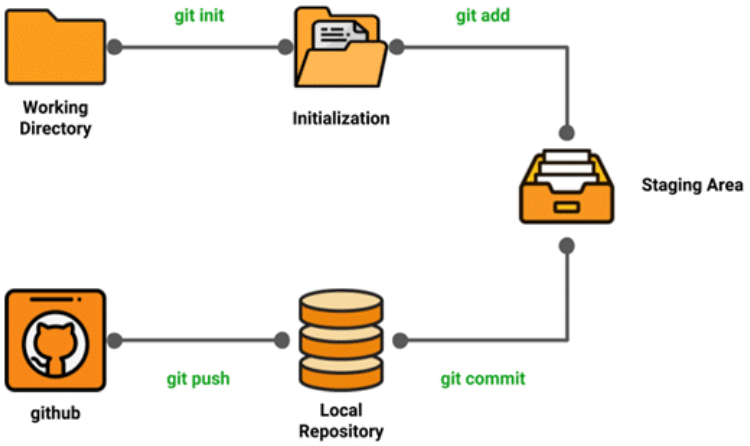


**Figura 3. Típico ciclo de desenvolvimento com Git**

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 et al. 2018].

### 2.6.1. Kanban

O *Kanban* funciona com quadros, chamados de quadro *Kanban*, onde um quadro é constituido por colunas e em cada coluna pode-se definir tarefas.

Em um quadro Kanban podemos agrupar tarefas, definir datas e responsáveis para essas tarefas, entre vários outros detalhes, o que facilita a organização e visão do trabalho a ser realizado em um projeto.

Existem várias ferramentas que oferecem um quadro *Kanban* para o gerenciamento de tarefas. Uma delas é o *GitHub Projects*, que é usada neste trabalho. A interface desse programa é mostrada na Figura 4.

# 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.

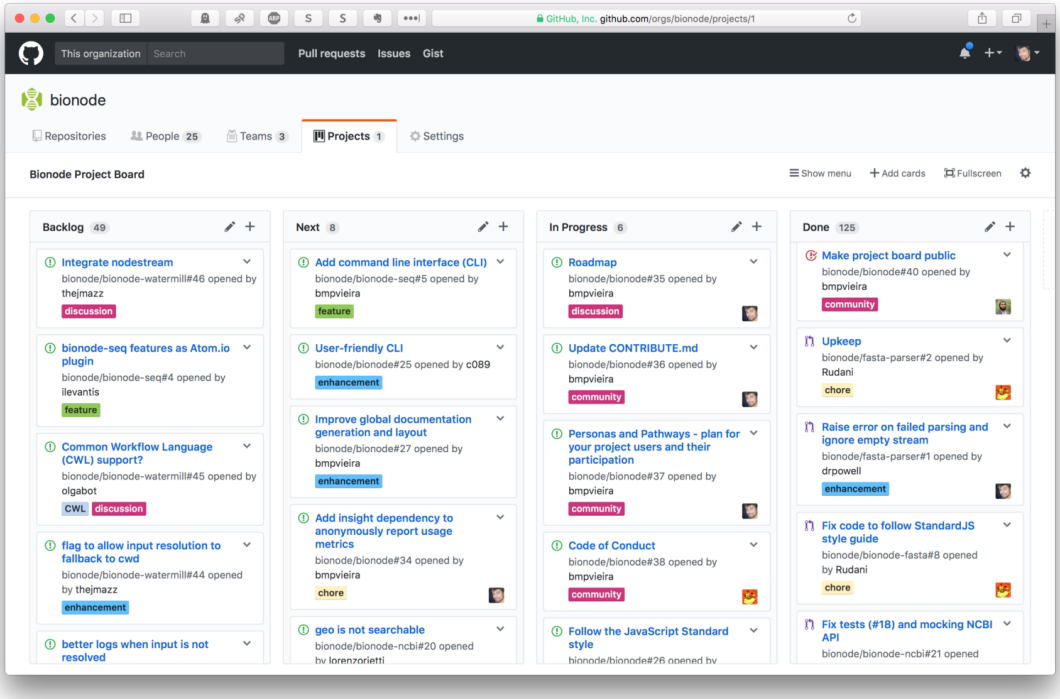


**Figura 4. Visão de um quadro *Kanban* do *GitHub Projects***

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 o gerenciamento de tarefas;
- *Wiki* onde foi 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 5, 6, 7 e 8, respectivamente.

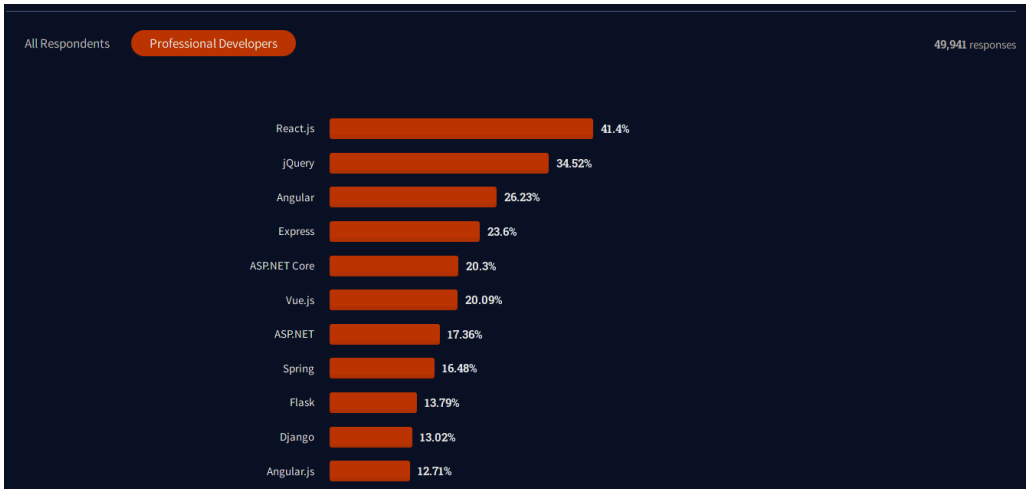


**Figura 5. Gráfico mostrando os frameworks Web mais usados**

A Figura 5 mostra as tecnologias para desenvolvimento de aplicações Web mais populares entre desenvolvedores profissionais que participaram da pesquisa, e mostra também a popularidade de *frameworks JavaScript* para desenvolvimento *Web*, como *React.js*, *Angular*, *Express* e *Vue.js*.

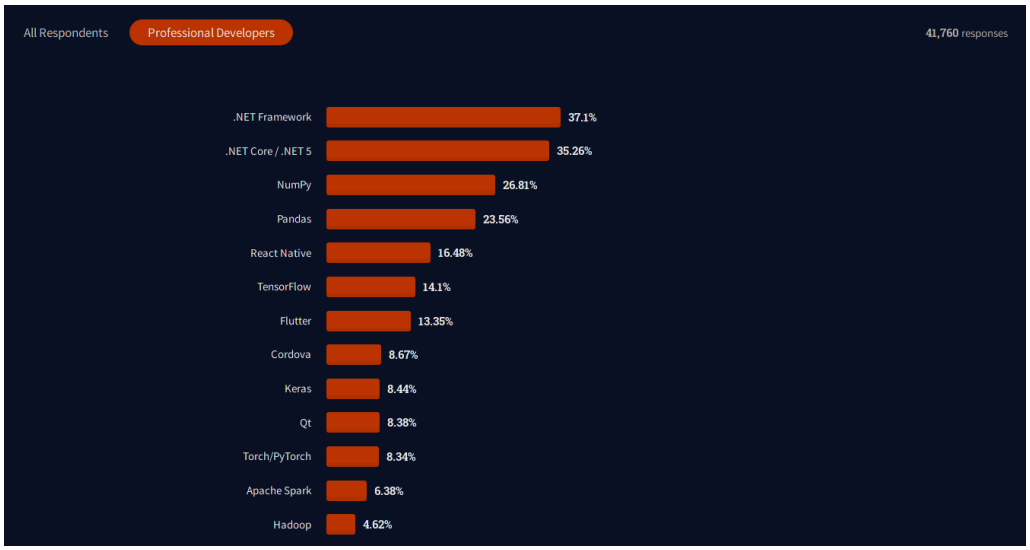


**Figura 6. Gráfico mostrando os frameworks gerais mais usados**

A Figura 6 mostra o *.NET Framework / .NET 5* como *framework* mais popular para o desenvolvimento geral de aplicações, outras tecnologias mostradas na pesquisa são específicas para algum tipo de tarefa e não são genéricas para o desenvolvimento de aplicações, como *NumPy* e *Pandas*, por exemplo, que são bibliotecas para análise de dados com a linguagem *Python* [Pratik 2017].

A Figura 7 mostra as ferramentas mais populares presentes no ambiente de desenvolvimento dos desenvolvedores que participaram da pesquisa. *Git* e *Docker* aparecem como ferramentas populares, sendo o *Git* presente na rotina de mais de 94% dos participantes, isso mostra que o *Git* é uma tecnologia bastante presente e que é importante que

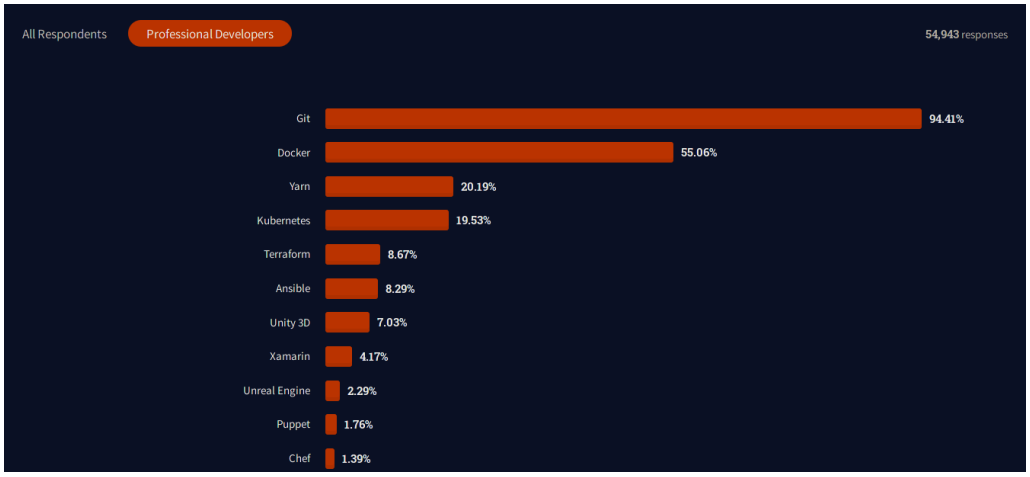


**Figura 7. Gráfico mostrando as ferramentas mais usadas**

o perfil de Analista de Sistema possua conhecimentos sobre essa ferramenta.

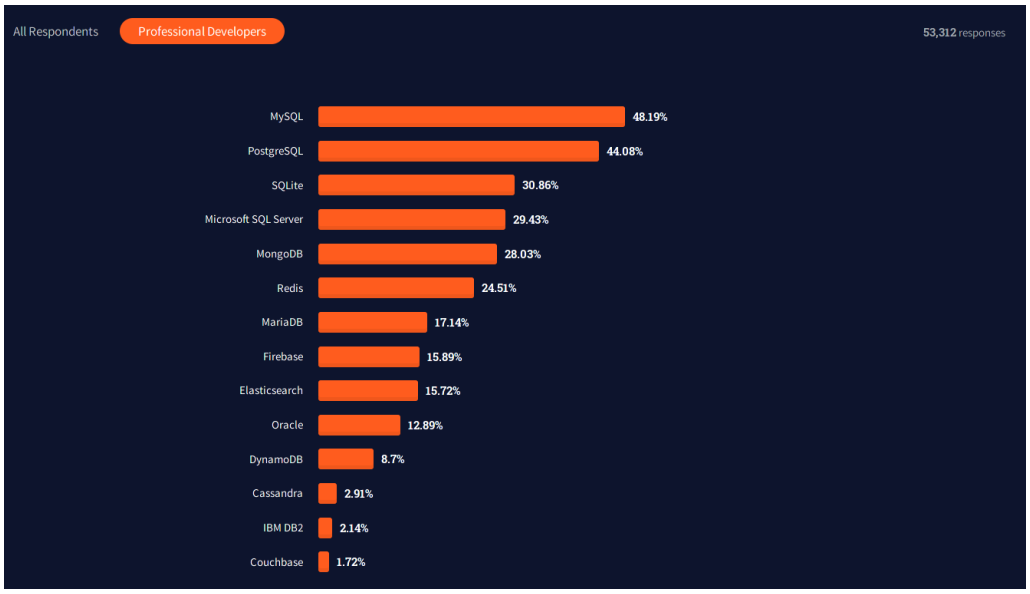


**Figura 8. Gráfico mostrando os SGBDs mais usados**

A Figura 8 mostra quais os SGBDs mais populares, sendo eles o *MySQL* e o *PostgreSQL*.

Baseando-se nos dados de popularidade indicados pela pesquisa do site *Stack Overflow* e na experiência do autor em certas tecnologias e ferramentas, foi optado o uso das tecnologias e arquitetura apresentadas na Seção 4.1.

## 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 e J. 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">
```

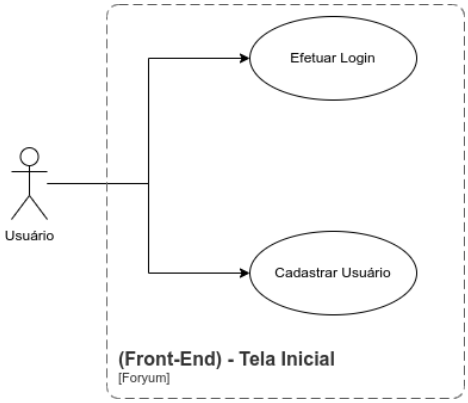


**Figura 9. Diagrama de casos de uso da tela inicial**

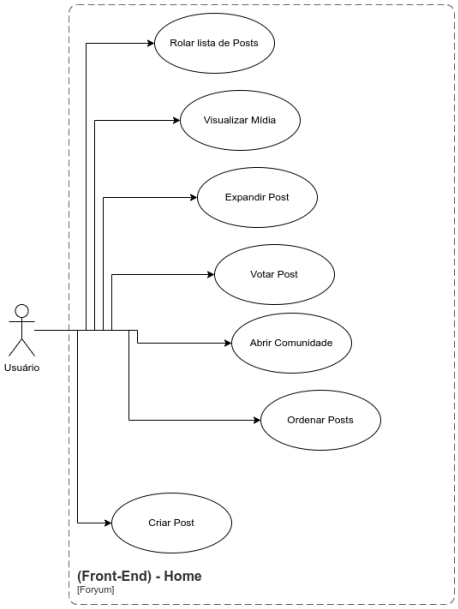


**Figura 10. Diagrama de casos de uso da tela *home***

```
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. O próximo passo foi de definir o domínio do sistema a ser desenvolvido e seus requisitos.

Os requisitos do sistema foram baseados em um sistema já existente, essa aplicação é o *Reddit*, e o nome do sistema desenvolvido nesse trabalho é 'Foryum'. 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 9 e 10.

Junto com os casos de uso, foram definidos as propiedades das entidades do sis-

tema a partir da criação de um diagrama de classe, o diagrama resultante é exibido na Figura 11.

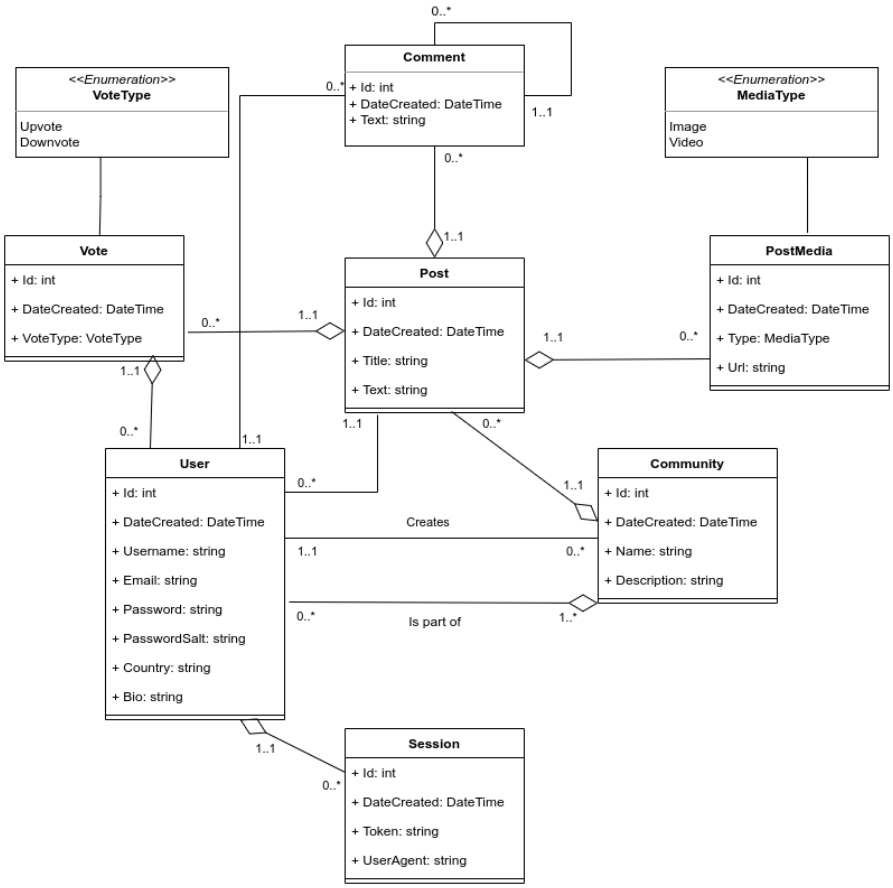


**Figura 11. Diagrama de classes do sistema**

## 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 um quadro *Kanban* para gerenciamento de tarefas e a implementação dos sistemas foi realizada com base no quadro *Kanban*. Para a implementação de algumas dessas houve a necessidade de aprendizagem para sua implementação.

Alguns dos aspectos técnicos que foram implementados no sistema são mostrados nas Seções 4.4.1 e 4.4.2.

### 4.4.1. Autenticação

O sistema impede que qualquer operação seja realizada se o usuário não estiver autenticado, salvo apenas a funcionalidade de cadastro. Para isso, existe um mecanismo de autenticação que foi implementado no sistema 'Foryum', esse mecanismo é chamado de *Bearer Token* e ele funciona de acordo com a Figura 12.

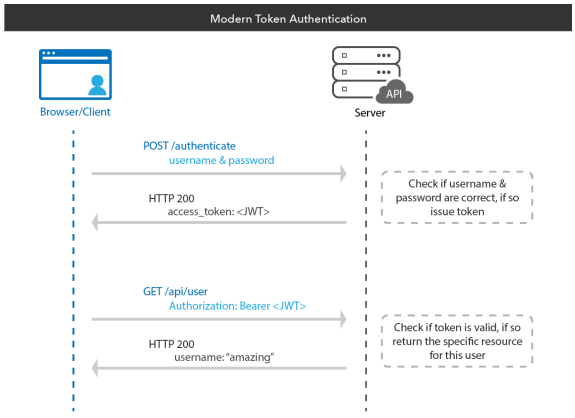


**Figura 12. Autenticação por Bearer Token**

Nesse mecanismo, o cliente após efetuar o Login no sistema, recebe um *token*. A partir desse momento o cliente passa a enviar esse token em todas as requisições subsequentes, dessa maneira o servidor sabe quem o cliente é e que está autenticado.

### 4.4.2. Armazenamento de Senhas

Outra questão relacionada a segurança é o armazenamento de senhas em banco de dados, nunca se deve armazenar senhas em texto plano. No sistema foi implementado um processo de armazenamento de senhas que é chamado de *Password Hash Salting*.

O funcionamento desse processo é visualizado na Figura 13.

Nesse processo, é concatenado uma string gerada aleatoriamente em uma senha, a partir do resultado dessa concatenação, um *Hash* é gerado e esse valor é armazenado no banco junto com o *Salt*.

Esse processo, para maior eficiência deve possuir *Salts* diferentes para cada usuário em um banco de dados. Armazenar senhas dessa maneira evita ataques conhecidos como *Rainbow Table Attack*.

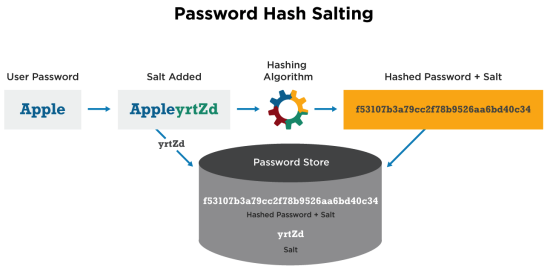


**Figura 13. Processo de *Salting* de uma senha**

### 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. As Figuras 14 e 15 mostram o resultado da aplicação *Web*, mostrando a tela de Bem-Vindo e a tela Home, respectivamente.

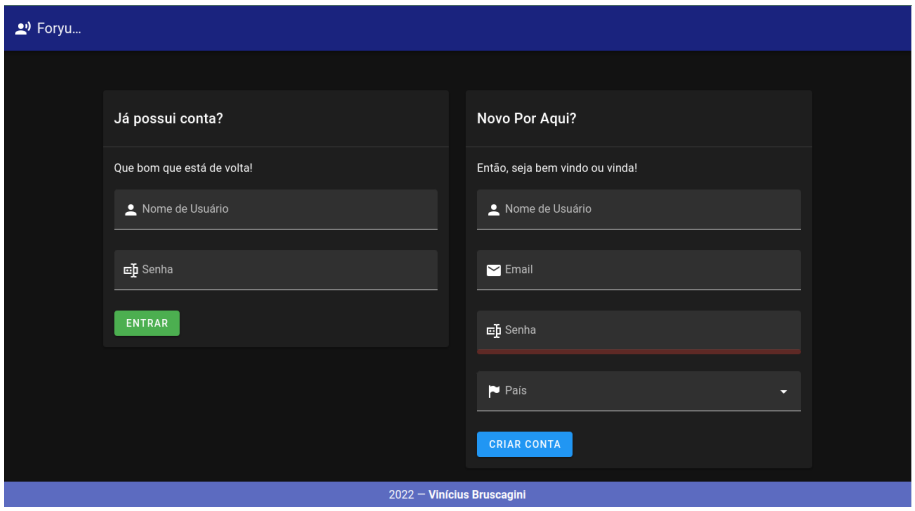


**Figura 14. Tela de bem-vindo**

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 usando o *Bearer Token*.

Além do desenvolvimento, foi criada a documentação técnica dessas aplicações que 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).

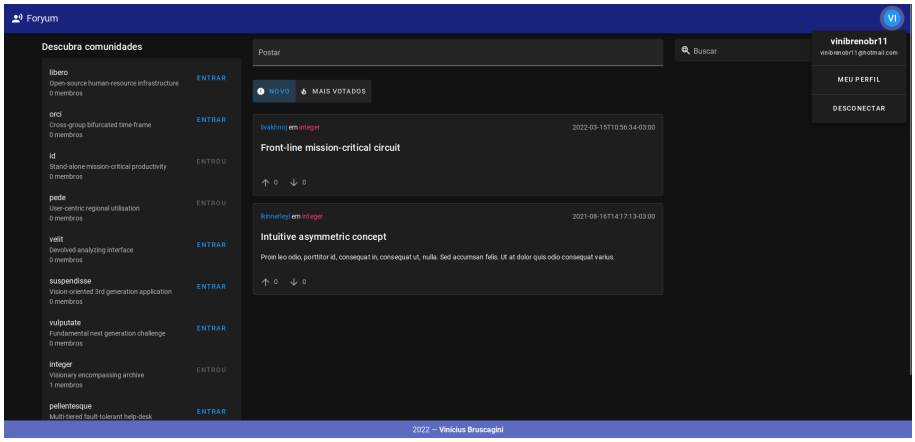


**Figura 15. Tela *home***

## 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 em relação a 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*.

Para criação desse trabalho, foram articulados conhecimentos obtidos atráves das disciplinas listadas abaixo, que são disciplinas que fazem parte da carga horária do curso de Análise e Desenvolvimento de Sistemas do Instituto Federal de São Paulo - Campus Hortolândia:

- Engenharia de Software;
- Inglês Técnico Avançado;
- Banco de Dados II;
- Interação Humano-Computador;
- Programação Orientada a Objetos;
- Arquitetura de Software;
- Qualidade de Software;
- Segurança da Informação;
- Desenvolvimento de Sistemas Web.

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 o *software* criado nesse trabalho, tais recursos poderiam ser o *upload* de mídias e gerenciamento de sessões, por exemplo.

## Referências

Atlassian (2020). What is version control? `https://www.atlassian.com/git/tutorials/what-is-version-control`. [Online; acesso em 04-maio-2022].

Coelho e Simone, H. (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., e Figueiredo, R. Investigação da documentação de maior importância para manutenção de software.

Duarte, Cristina, K., Falbo, e de Almeida, R. (2000). Uma ontologia de qualidade de software. *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/`. [Online; acesso em: 23-agosto-2021].

Macedo e Borba, M. C. (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., e 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., E. (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, Roberto, Tait, e Tania (2000). Tecnologia de informação: Evolução e aplicações. *Revista Teoria e Evidência Econômica*, 8(14).

Pratik (2017). Numpy and pandas tutorial – data analysis with python. `https://cloudxlab.com/blog/numpy-pandas-introduction/`. [Online; acesso em 02-outubro-2022].

Santos, V., E., Marinho, e T., G. (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. `www.levysiqueira.com.br/artigos/qualidade-de-codigo.pdf`. [Online; acesso em 29-setembro-2022].

StackOverflow (2021). Stack overflow developer survey. `https://insights.stackoverflow.com/survey/2021`. [Online; acesso em: 23-agosto-2021].

# Documento Digitalizado Público

## Anexo I (Artigo) - Vinícius Breno Bruscagini - HT3001474

**Assunto:** Anexo I (Artigo) - Vinícius Breno Bruscagini - HT3001474
**Assinado por:** Carlos Junior
**Tipo do Documento:** Anexo
**Situação:** Finalizado
**Nível de Acesso:** Público
**Tipo do Conferência:** Documento Digital

Documento assinado eletronicamente por:

- **Carlos Roberto dos Santos Junior**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 03/10/2022 21:26:10.

Este documento foi armazenado no SUAP em 03/10/2022. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifsp.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

**Código Verificador:** 1118055
**Código de Autenticação:** 3e5d07551e