Universidade Federal do Estado do Rio de Janeiro
Centro de Ciências Exatas e Tecnologia
Escola de Informática Aplicada

Plataforma Learning Curve: uma versão preliminar do suporte computacional para elaboração colaborativa da documentação do desenvolvedor

Juan Garcia Trindade e Silva

Orientador

Paulo Sérgio Medeiros dos Santos

Rio de Janeiro, RJ – Brasil
Março, 2022

Catalogação informatizada pelo autor

S586

Silva, Juan Garcia Trindade e
Plataforma Learning Curve: uma versão preliminar do suporte computacional para elaboração colaborativa da documentação do desenvolvedor / Juan Garcia Trindade e Silva. -- Rio de Janeiro, 2022.
59 f.

Orientador: Paulo Sérgio Medeiros dos Santos.
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal do Estado do Rio de Janeiro, Graduação em Sistemas de Informação, 2022.

1. auto-aprendizagem. 2. documentação. 3. desenvolvimento de software. I. Santos, Paulo Sérgio Medeiros dos, orient. II. Título.

Plataforma Learning Curve: uma versão preliminar do suporte computacional para elaboração colaborativa da documentação do desenvolvedor

Juan Garcia Trindade e Silva

Projeto de Graduação apresentado à Escola de Informática Aplicada da Universidade Federal do Estado do Rio de Janeiro (UNIRIO) para obtenção do título de Bacharel em Sistemas de Informação.

Aprovado por:

_____

Paulo Sérgio Medeiros dos Santos (UNIRIO)

_____

Jobson Luiz Massollar da Silva (UNIRIO)

_____

Victor Vidigal Ribeiro (Granbery)

Rio de Janeiro, RJ – Brasil

Março, 2022

RESUMO

Apesar da crescente demanda por desenvolvedores de *software*, ainda existe uma escassez de profissionais qualificados no mercado de trabalho. Um elemento central para esta escassez é a capacidade de formação e qualificação de profissionais para o mercado. Com o objetivo de auxiliar na qualificação desses profissionais, surge a premissa de utilizar documentações de desenvolvedores como fonte de informação e aprendizado.

Esse trabalho propõe a construção de uma nova plataforma que permita a visualização e construção dessas documentações, fornecendo um ambiente colaborativo entre estudantes e profissionais de desenvolvimento de *software*.

Para auxiliar na concepção da plataforma, foi realizada uma etapa de fundamentação teórica, buscando informações sobre aspectos que influenciam positivamente e negativamente uma documentação de software, além de uma etapa de Design Thinking para apoiar o entendimento do problema e a geração de ideias.

Como resultado deste trabalho, foi construída uma versão inicial da plataforma Learning Curve, com o objetivo de servir como apoio ferramental para trabalhos futuros explorarem o tema de forma empírica.

Palavras-chave: auto-aprendizagem, documentação, desenvolvedores de software

ABSTRACT

Despite the growing demand for software developers, there is still a shortage of qualified professionals in the job market. A central element for this shortage is the capacity to train and qualify professionals for the market. With the objective of helping in the qualification of these professionals, the premise of using developer documentation as a source of information and learning arises.

This work proposes the construction of a new platform that allows the visualization and construction of these documentations, providing a collaborative environment between students and software development professionals.

To assist in the design of the platform, a theoretical foundation stage was carried out, seeking information on aspects that positively and negatively influence software documentation, in addition to a Design Thinking stage to support the understanding of the problem and the generation of ideas.

As a result of this work, an initial version of the Learning Curve platform was built, with the objective of serving as a tool support for future works to explore the theme empirically.

Keywords: self-learning, documentation, software developers

Lista de Figuras

Figura 1: Exemplo de documentação criada através da plataforma Docusaurus……19

Figura 2: Exemplo de documentação criada através da plataforma Read the Docs……20

Figura 3: Aplicativo Mimo, um exemplo de produto examinado na fase de Imersão…22

Figura 4: Exemplos de cartões de Insight, agrupados por afinidade……...25

Figura 5: Resultado do processo de Brainstorming, com ideias sobre possíveis funcionalidades da aplicação……..……..……...26

Figura 6: Exemplo do uso do User Story Mapping para definição do escopo……27

Figura 7: Página inicial da aplicação……..…….30

Figura 8: Formulário de cadastro……..……...31

Figura 9: Página Home e suas funcionalidades……...32

Figura 10: Página de projetos……..……32

Figura 11: Detalhes de um projeto……..…….33

Figura 12: Editor de documentação…….33

Figura 13: Templates disponíveis no editor……..……...34

Figura 14: Cookbook gerado automaticamente……..…….34

Figura 15: Página de documentação……..……..35

Figura 16: Coluna com exemplo de código e botão para cópia instantânea……35

Figura 17: Comentários de uma documentação……..……36

Figura 18: Exemplo de Markdown gerado……..……36

Figura 19: Botão para contribuição em uma documentação……...37

Figura 20: Contribuição aguardando revisão……..…….37

Figura 21: Página com detalhes da contribuição…….……37

Figura 22: Resultados de uma busca na plataforma através da barra de pesquisa……..38

Figura 23: Resultados de uma busca na plataforma através da página de pesquisa detalhada……..……..……..39

Figura 24: Página Explorar……..……39

Figura 25: Perfil de um usuário na plataforma……40

Figura 26: Diagrama simplificado com a arquitetura do sistema e as tecnologias utilizadas……..……..……..……41

Figura 27: Código para upload de imagens no Cloudinary…….44

Figura 28: Classe responsável por consumir os eventos e sincronizar com o Elasticsearch……..……..……45

Figura 29: Exemplo de evento de criação de usuário……..46

Figura 30: Serviço de inserção de dados no Elasticsearch……..46

Figura 31: Modelagem do banco de dados relacional…….47

Figura 32: Possível estrutura de uma documentação……..48

Figura 33: Exemplo de documento no MongoDB……..48

Figura 34: Lista de seções da documentação analisada……..50

Figura 35: Menu lateral da documentação na plataforma……...51

Figura 36: Visualização da nova documentação na barra de pesquisa……51

Figura 37: Exemplo de código na documentação da linguagem Go……...52

Figura 38: Exemplo de formatação gerada pelo Markdown……...52

Lista de Tabelas

Tabela 1: Diretrizes para construção de documentações de desenvolvedor……16

Tabela 2: Análise de ferramentas baseado nos macrogrupos apresentados…….23

Tabela 3: Nova análise de ferramentas, focada em ferramentas para construção de documentações de software..……...24

Tabela 4: Premissas concebidas na fase de Ideação da plataforma…… 27

Tabela 5: Pontuação obtida por ação do usuário…….40

Índice

Introdução 11
- Motivação 11
- Objetivo 13
- Estrutura do texto 13

Fundamentação Teórica 14
- Contexto do trabalho 14
- Documentação de desenvolvedor 15
- Recomendações gerais para documentação 16
  - Comportamento e expectativas dos desenvolvedores 16
  - Estrutura da documentação 17
  - Conteúdo da documentação 18
  - Tratamento de erros 18
- Ferramentas similares 19
  - Docusaurus 19
  - Read the Docs 20

Concepção da Plataforma 21
- Design Thinking 21
- Metodologia 21
  - Imersão 21
  - Síntese e análise 24
  - Ideação 25
  - Prototipação 28

Visão Geral da Aplicação 29
- Principais entidades 29
- Gestão de acesso 30
- Páginas 30
  - Landing page 30
  - Autenticação 31
  - Home 31
  - Projetos 32
  - Documentações 33
  - Contribuições 37
  - Pesquisa 38
  - Explorar 39
  - Perfil do usuário 40

Arquitetura e Design do Sistema 41
- Linguagem de programação 42
- Aplicação web 42

API 43
Repositório de imagens 43
Mecanismo de busca 45
Banco de dados 46
PostgreSQL 46
MongoDB 47
Prova de Conceito 50
Análise da documentação existente 50
Conclusão 53
Considerações finais 53
Limitações e próximos passos 53
Referências 55
Apêndice A - Estórias de Usuário 58

1 Introdução

1.1 Motivação

De acordo com um relatório divulgado pela Associação das Empresas de Tecnologia da Informação e Comunicação e de Tecnologias Digitais (Brasscom), há uma estimativa de 70 mil novas oportunidades de emprego a serem criadas por ano na área da tecnologia da informação (TI) até 2024. Entretanto, a quantidade de novos profissionais qualificados será menor que a demanda, com uma expectativa média de formação de somente 46 mil trabalhadores por ano.

Segundo uma pesquisa publicada em 2019 pelo Stack Overflow[1] - um dos maiores sites relacionados ao desenvolvimento de sistemas, com mais de 4.7 milhões de usuários - não é mais estritamente necessário um diploma de bacharelado para atuar como um desenvolvedor de sistemas atualmente. Por exemplo, dentre os participantes que afirmaram trabalhar como desenvolvedores de software profissionalmente, aproximadamente 25% deles informaram que não possuem diploma de curso superior. Além disso, em torno de 90% dos participantes da pesquisa informaram que aprenderam pelo menos uma nova linguagem de programação, *framework* ou ferramenta fora da educação formal, representada pelas instituições de ensino superior no Brasil.

Diante da necessidade de qualificação de mais profissionais, a internet vem exercendo um papel fundamental no fornecimento de alternativas para educação formal. Durante os últimos anos, diversos modelos de aprendizado foram criados focados exclusivamente nos desenvolvedores de *software,* como Bootcamps (BURKE et al., 2018) e Massive Open Online Course (MOOC) (HEW; CHEUNG, 2014) (REICH; RUIPÉREZ-VALIENTE, 2019). O primeiro é um curso de curta duração destinado a preparar os participantes para o mercado de trabalho ao fim do programa, ensinando programação de forma intensiva e prática. Os MOOCs, por outro lado, são cursos online com número ilimitado de participantes, com o objetivo de promover discussões e interações entre professores e estudantes em uma larga escala. Além disso, alguns

[1] https://stackoverflow.com/

outros serviços como o StackOverflow e o Github[2], com milhões de usuários espalhados pelo mundo, estenderam seus objetivos iniciais e começaram a servir como locais para compartilhamento de conhecimento sobre desenvolvimento de software.

Segundo o StackOverflow (2019), estudantes e profissionais de desenvolvimento de *software* já estão acostumados com o autodidatismo, demonstrando um anseio por autonomia e conhecimento prático no aprendizado. É importante ressaltar que a maioria dos estudantes desejam obter um novo emprego na área ou resolver um problema específico; consequentemente, é comum que eles escolham se aprofundar no uso de ferramentas populares que possam auxiliá-los nesses objetivos.

Com o objetivo de reduzir a escassez de profissionais qualificados, surge a oportunidade de utilizar documentações de desenvolvedor como instrumento de aprendizado. Essas documentações podem ser descritas, de forma resumida, como uma espécie de manual de uma ferramenta de desenvolvimento de *software* com detalhamento técnico sobre seus principais usos, além de detalhes sobre a sua implementação. Além disso, em boa parte dos casos, elas são escritas pelos próprios desenvolvedores das ferramentas. Esse tipo de documentação será abordado com maiores detalhes no próximo capítulo.

A premissa é que a documentação associada com tecnologias de *software* (ex: *frameworks*, ferramentas, bibliotecas, técnicas e metodologias) possam ser uma fonte rica e confiável de conhecimento. Todavia, para que sejam utilizadas com o propósito de aprendizado, as documentações precisam ser organizadas para este propósito. De toda forma, estudantes de TI estão acostumados a consumir esse tipo de informação ao interagirem com tecnologias que eles não dominam.

Entretanto, infelizmente, existem diversos casos de *softwares* sem documentação ou com documentação de baixa qualidade. Além disso, há ainda uma percepção negativa de alguns desenvolvedores de que a elaboração de uma documentação de *software* é algo custoso e difícil de manter.

[2] https://github.com/

1.2 Objetivo

Baseado nessa premissa, está sendo desenvolvido um novo projeto denominado LearningCurve, visando construir uma plataforma com duplo objetivo: (i) apoiar a elaboração de documentações de desenvolvedor com maior qualidade e de forma colaborativa e (ii) auxiliar no aprendizado de estudantes de TI através da documentação de software. Este trabalho, entretanto, tem como escopo somente o primeiro objetivo: construir um apoio ferramental para a elaboração do projeto, facilitando a construção, colaboração e visualização de documentações dentro da plataforma LearningCurve. O contexto do projeto será detalhado posteriormente.

1.3 Estrutura do texto

Esse trabalho é estruturado em capítulos. Além dessa introdução, o conteúdo será organizado na seguinte estrutura:

Capítulo II: Apresentação do contexto e dos conceitos relacionados a esse estudo, apresentando e discutindo os principais trabalhos relacionados.

Capítulo III: Processo de ideação da ferramenta, explicando todos os passos por trás da concepção deste trabalho, com foco na metodologia Design Thinking.

Capítulo IV: Visão geral das funcionalidades da aplicação, apresentando suas principais funcionalidades e traçando um paralelo entre os objetivos, diretrizes e premissas propostas com o produto final desenvolvido.

Capítulo V: Detalhamento das características técnicas da plataforma, aprofundando nas decisões associadas a arquitetura de software, escolha de tecnologias e outros assuntos puramente técnicos.

Capítulo VI: Prova de conceito. Avaliação de uma documentação existente, utilizando as diretrizes discutidas anteriormente na fundamentação teórica. Neste capítulo também são apontadas eventuais melhorias fornecidas pela plataforma LearningCurve.

Capítulo VII: Considerações finais sobre o estudo, condensando as principais descobertas e contribuições para a área. Também são discutidas possibilidades para eventuais futuros trabalhos.

2 Fundamentação Teórica

Esse capítulo aprofunda tópicos fundamentais deste trabalho, como os conceitos por trás de uma documentação de desenvolvedor, as diretrizes para construção de uma documentação de qualidade e outras descobertas extraídas de trabalhos relacionados.

2.1 Contexto do trabalho

Baseado nos problemas apresentados na seção introdutória deste texto, como a escassez de profissionais qualificados e o potencial da documentação de software na aprendizagem, foi concebido um novo projeto denominado LearningCurve. A ideia é que baseando-se no estado da arte seja possível identificar quais características da documentação influenciam na sua qualidade.

O projeto busca fornecer uma plataforma para que essas documentações possam ser criadas, mantidas e utilizadas, além de auxiliar na colaboração entre autor e a comunidade de desenvolvedores. O conceito de plataforma pode ser descrito como um uma junção de elementos técnicos (software e hardware) e organizacionais (padrões e processos) (DE REUVER; SØRENSEN; BASOLE, 2018).

Esse projeto está em estágio inicial de desenvolvimento, tendo sua concepção elaborada por meio do Platform Design Toolkit[3]. Esse método pode ser definido como "um guia de criação e modelagem de plataformas digitais, definindo um passo a passo utilizando templates em formato de canvas. Canvas são ferramentas dinâmicas em formato de quadro que permitem, através do pensamento visual, analisar o modelo de negócios, no caso, da plataforma que está sendo criada. Desta forma, além de ajudarem a fundamentar a plataforma, servirão como base para seu desenvolvimento" (CAMPOS, 2021, não publicado)

A expectativa é que os estudantes da plataforma utilizem uma abordagem autodidata para o aprendizado, apoiando-se nas documentações de software como ferramenta de aprendizado. Este comportamento é baseado no conceito de “aprendizagem autodirigida” (self-directed learning) (KNOWLES, 1975), referente ao conceito dos estudantes tomarem a iniciativa e, sem a ajuda dos professores,

[3] https://platformdesigntoolkit.com/

identificarem as suas próprias necessidades, objetivos e estratégias de aprendizagem. Esses estudantes também são responsáveis por seus próprios resultados, avaliando seu progresso ao longo do tempo.

2.2 Documentação de desenvolvedor

A documentação permeia cada fase do ciclo de vida do desenvolvimento de um software (LETHBRIDGE et al., 2003). Ela pode ser descrita como um conjunto de instruções com o objetivo de orientar a interação e o manuseio de um sistema. Essas documentações podem ser classificadas em dois tipos: internas e externas (GAGGERO, 1983).

O primeiro tipo mencionado refere-se às documentações destinadas ao auxílio na manutenção de um sistema ou na instrução de novos membros dentro de um time ou organização. Alguns exemplos são os requisitos do sistema e diagramas de projeto UML (*Unified Modeling Language*).

As documentações externas, por outro lado, são mais voltadas ao problema e podem ser direcionadas a diferentes tipos de audiências. Segundo Gaggero (1983), existem 3 tipos de documentações externas: (i) documentação introdutória, voltada ao usuário final, contendo uma breve descrição do programa e de suas funcionalidades; (ii) documentação do usuário, contendo instruções de uso e guias de instalações, destinadas aos interessados nas operações e aplicações do programa; (iii) documentação de manutenção, apresentando manuais de teste e do sistema, assim como histórico de arquivos, visando auxiliar os desenvolvedores responsáveis pela manutenção de um programa. Por exemplo, uma documentação externa pode ser destinada tanto a profissionais realizando integrações com uma API comercial quanto a desenvolvedores contribuindo com projetos *open source* (código aberto).

No contexto deste trabalho, o foco são as documentações do usuário, ou mais especificamente, as documentações de desenvolvedor (DAGENAIS; ROBILLARD, 2010). Esse termo, apesar de não ser largamente utilizado, sintetiza o seu propósito: elas são escritas pelos próprios desenvolvedores com o objetivo de fornecer informações importantes sobre o domínio e o *design* de um sistema, de forma similar a um manual, relacionando-os com a implementação e demonstrando como solucionar possíveis casos de uso através da utilização do *software* em questão.

2.3 Recomendações gerais para documentação

Para compreender quais aspectos influenciam a qualidade de uma documentação de desenvolvedor, foram reunidas um conjunto de recomendações de diversos autores. É importante ressaltar que essas recomendações não são exclusivamente relacionadas a documentação de software, englobando também tópicos adjacentes como aprendizado, autodidatismo e também elaboração de manuais e instruções em geral.

Baseado nessas recomendações, foi criada uma série de diretrizes que servirão de apoio para as próximas etapas deste trabalho. A Tabela 1 contém um resumo das diretrizes identificadas, que serão detalhadas mais minuciosamente adiante.

#	Diretriz	Tema
1	Introduza a documentação com conteúdo prático ao invés de teórico	Comportamento e expectativas dos desenvolvedores
2	Forneça um mecanismo de busca robusto	Estrutura da documentação
3	Organize a documentação em três colunas	Estrutura da documentação
4	Facilite a visualização e cópia de código	Estrutura da documentação
5	Conteúdo implícito deve ser elucidado através de exemplos	Conteúdo da documentação
6	Evite dependência direta entre seções	Conteúdo da documentação
7	Adicione seções com soluções de erros frequentes	Tratamento de erros

Tabela 1: Diretrizes para construção de documentações de desenvolvedor

Comportamento e expectativas dos desenvolvedores

Meng et al. (2020) discutem alguns aspectos do comportamento de um desenvolvedor ao interagir com uma documentação de software. Eles introduzem o conceito de "aprendizagem sob medida" (*just-in-time learning*), observando o fato que a maioria dos desenvolvedores não tem intenção de entender completamente a documentação, mas sim somente às partes necessárias para atingir determinado objetivo, interagindo de forma seletiva com a mesma.

Outro aspecto relevante é o balanceamento apropriado entre teoria e prática. Van der Meij e Caroll (1995), apesar de não se referirem especificamente à documentação de software, reforçam a relutância dos leitores em assimilar e apropriar-se de informações

mais conceituais. Os autores comentam sobre a importância de não fornecer conteúdos puramente teóricos no começo de uma documentação, evitando que os leitores fiquem impacientes. Eles recomendam, por exemplo, iniciar a documentação com uma tarefa prática e introduzir a teoria posteriormente, de forma que os leitores possam ter uma visão mais clara da relação entre o conceito e o domínio (Diretriz #1).

Como mencionado, desenvolvedores tendem a valorizar mais informações práticas do que teóricas. Baseado nessa premissa, um fenômeno denominado "aprendizagem dirigida a aplicativo" (*app-directed learning)* (SILLITO; BEGGEL, 2013) vem sendo estudado. Esse fenômeno mostra que desenvolvedores definem seus tópicos de estudo baseado nas necessidades de suas tarefas, aprendendo o mínimo necessário para finalizá-las, ao invés de buscar um conhecimento profundo sobre o assunto. Uma parte considerável da documentação costuma ser ignorada por eles até eles enfrentarem algum problema relacionado ao seu objetivo. Por exemplo, em um cenário hipotético no qual um desenvolvedor precise adicionar uma funcionalidade de cache em sua aplicação, é mais provável que ele busque obter conhecimento sobre uma ferramenta popular, como o Redis[4], ao invés de pesquisar conceitos específicos sobre o assunto.

Os desenvolvedores geralmente buscam informação através de pesquisas *online*, solucionando dúvidas em fóruns de perguntas e respostas ou lendo postagens de blogs. Por conta de estarem associadas a resolução de um problema específico, as informações encontradas tendem a ser mais superficiais e acabam fornecendo um conhecimento parcial sobre o assunto. Entretanto, é importante destacar a redução substancial no tempo gasto nas tarefas, além do aumento na motivação e no sentimento de conquista por parte dos desenvolvedores.

Estrutura da documentação

A estrutura da documentação tem um papel fundamental no processo de aprendizado. Van der Meij e Caroll (1995) afirmam ser crucial fornecer mais que um conteúdo de qualidade, mas também ferramentas que permitam aos leitores navegar rapidamente por todo o conteúdo. A experiência ao interagir com a documentação impacta diretamente na velocidade e na quantidade de informação que pode ser compreendida e apropriada.

[4] https://redis.io/

Meng et al. (2020), por exemplo, recomendam que seja fornecido um mecanismo de busca robusto, para que seja possível procurar por conteúdos específicos; caso não seja possível, é aconselhado que todo o conteúdo seja concentrado em uma única página (Diretriz #2). Os autores também sugerem a organização do conteúdo em um *layout* de três colunas, separando claramente o texto do menu lateral e eventuais exemplos de código (Diretriz #3). Além disso, é importante que os códigos apresentados também sejam fáceis de serem copiados e colados (Diretriz #4).

Conteúdo da documentação

No que tange o conteúdo, Meng et al. (2020) recomendam diversas ações para garantir a simplicidade e clareza. Por exemplo, os autores sugerem que não seja explicado o óbvio: o conteúdo implícito deve ser elucidado através de exemplos práticos ao invés de mais explicações textuais (Diretriz #5). Se não for possível, é recomendado ser o mais breve possível para evitar uma experiência de aprendizado massiva e frustrante. Em relação aos exemplos, também é recomendado que qualquer tipo de teoria seja apresentada sempre de forma concreta, ilustrando a ideia por trás. Além disso, é aconselhado não criar dependências diretas entre seções (Diretriz #6): elas devem ter um começo, meio e fim bem definidos para que os leitores possam entender plenamente o assunto mesmo que não tenham lido as seções anteriores.

Tratamento de erros

Meng et al. (2020) enfatizam a importância do tratamento de erros. Caso o conteúdo da documentação possa resultar em um eventual problema, uma seção com possíveis soluções deve estar presente em seguida (Diretriz #7). Essas seções, geralmente presentes no final da documentação, são essenciais para manter os leitores engajados. Para embasar esse argumento, os autores destacam que desenvolvedores gastam entre 25% a 50% do tempo envolvidos em problemas ao estudarem uma nova tecnologia.

No entanto, mesmo quando uma seção sobre tratamento de erros é fornecida, ainda é difícil mitigar completamente as chances de um desenvolvedor se deparar com um novo problema. Levando isso em consideração, a coleta de *feedbacks* torna-se crucial para que seja possível realizar a melhoria contínua da documentação.

2.4 Ferramentas similares

Existem outras ferramentas e plataformas que fornecem funcionalidades para construção de documentações de software. A seguir serão listados as principais ferramentas similares, traçando um paralelo entre elas e a plataforma a ser desenvolvida neste trabalho.

Docusaurus

Docusaurus[5] é uma ferramenta de código aberto para gerar de websites estáticos a partir de documentações de software. O artefato gerado é uma aplicação de página única (single-page application, em inglês) em React[6], fornecendo funcionalidades como suporte a internacionalização, navegação entre o conteúdo e uma ferramenta de busca. Ela é mantida atualmente pela companhia Meta[7] (anteriormente conhecida como Facebook).

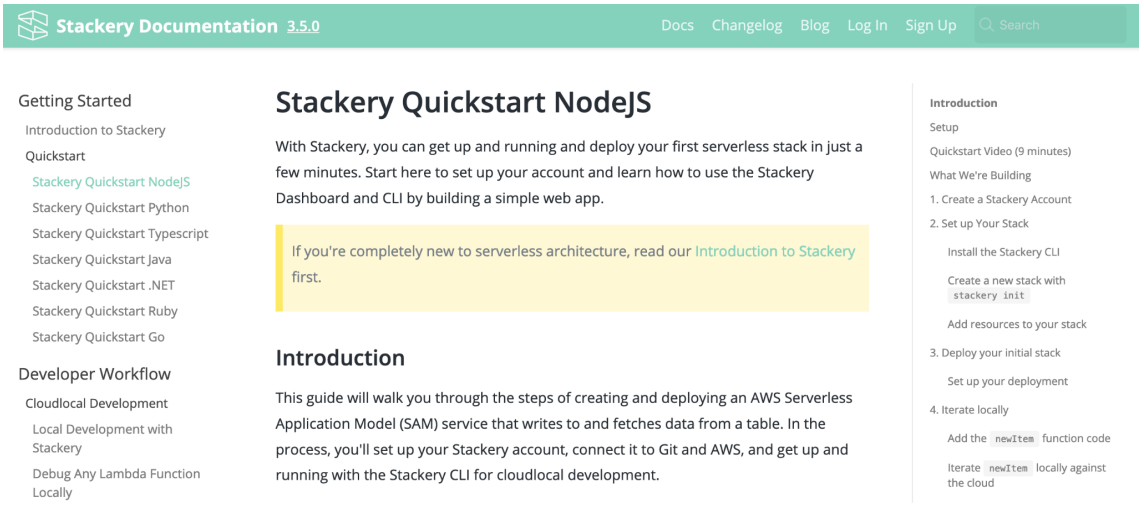

Figura 1: Exemplo de documentação criada através da plataforma Docusaurus

Existem aspectos similares entre o Docusaurus e a plataforma a ser construída neste trabalho, como utilização de Markdown[8] como linguagem de marcação para construção das documentações, versionamento do conteúdo e a disponibilização de um motor de busca para pesquisa.

Apesar dos aspectos mencionados anteriormente, o principal diferencial da plataforma proposta em relação ao Docusaurus é a construção de um ambiente

[5] https://docusaurus.io/
[6] https://reactjs.org/
[7] https://about.facebook.com/meta/
[8] https://www.markdownguide.org/

colaborativo. A ferramenta analisada necessita que cada desenvolvedor hospede sua documentação em um website próprio, além de não fornecer nenhuma funcionalidade para que os usuários sugiram modificações em uma documentação. LearningCurve, por outro lado, tem como objetivo ser um repositório central com documentações de diferentes fontes, permitindo que o desenvolvedor crie e visualize uma documentação automaticamente, além de permitir que outros usuários do site interajam com a documentação de forma colaborativa.

Read the Docs

Read the Docs[9] é uma plataforma online para construção e compartilhamento de documentações de software. Ela permite que os desenvolvedores utilizem linguagens de marcação, como Markdown, para construção das documentações, exibindo-as na própria plataforma, sem necessidade de hospedá-las em um website próprio. Apesar de conter menos funcionalidades que a ferramenta anterior, a plataforma em questão é a que mais se aproxima do LearningCurve devido ao fato de servir como um portal agregador de documentações.

A principal diferença, entretanto, é que a plataforma Read the Docs não fornece nenhuma funcionalidade para colaboração e interação entre usuários. Diferentemente da plataforma proposta neste trabalho, não é possível, por exemplo, adicionar comentários ou sugerir modificações em uma documentação.

Figura 2: Exemplo de documentação criada através da plataforma Read the Docs

[9] https://readthedocs.org/

3 Concepção da Plataforma

Para um melhor entendimento do problema e um maior aprofundamento teórico, foi utilizada a metodologia de Design Thinking, que será apresentada nos próximos tópicos.

As sessões foram realizadas ao longo de 2 meses, totalizando aproximadamente 6 sessões. Essas sessões foram compostas por 3 participantes no total: o autor deste trabalho, o professor orientador e um aluno de iniciação científica.

3.1 Design Thinking

Apesar de sua popularidade, a definição de Design Thinking ainda é tema de debate entre seus praticantes (LIEDTKA, 2015). Ele é comumente definido como uma combinação entre design de produto, inovação e solução de problemas. Segundo Tschimmel (2012), o Design Thinking também pode ser descrito como uma introdução da cultura e métodos de design em diferentes áreas.

Um dos grandes benefícios do Design Thinking é o seu repertório de estratégias para solução de problemas. *Designers* vem enfrentando diversos problemas complexos e em aberto ao longo dos anos, desenvolvendo diversas formas de solucioná-los (DORST, 2011). Ao longo do próximo tópico, as fases e principais técnicas utilizadas neste trabalho serão apresentadas e detalhadas.

3.2 Metodologia

Imersão

A fase inicial do Design Thinking é denominada imersão. Ela pode ser dividida em duas subfases: Imersão Preliminar, destinada ao entendimento inicial do problema; e Imersão Aprofundada, focada no entendimento das necessidades e oportunidades que irão guiar a geração de soluções nas fases seguintes do projeto.

Neste trabalho, para alcançar os objetivos de cada passo, uma pesquisa documental (*Desk Research*) foi proposta. Essa pesquisa consistiu na coleta de informações sobre o problema através de trabalhos relacionados, como visto no capítulo

anterior. Após essa etapa, foram analisados produtos existentes, examinando as principais funcionalidades e eventuais limitações que poderiam ser endereçadas na solução final proposta.

Nessa etapa do trabalho, o foco da pesquisa ainda era a concepção da plataforma LearningCurve. Por conta disso, os produtos analisados focam não somente na construção de documentação de desenvolvedor, mas sim no aprendizado de desenvolvimento de software como um todo. Alguns exemplos de produtos analisados foram os websites FreecodeCamp[10] e Github; e os aplicativos móveis SoloLearn[11] e Mimo[12] (Figura 3), designados exclusivamente para estudantes de ciência da computação.

Figura 3: Aplicativo Mimo, um exemplo de produto examinado na fase de Imersão

Os critérios de avaliação foram divididos em 4 macrogrupos, com o objetivo de responder às seguintes questões:

1. **Gerenciamento do Conteúdo:** Como é feita a criação, atualização e gestão do conteúdo da ferramenta? É centralizada em um indivíduo/organização ou é compartilhada entre a comunidade?
2. **Estruturação do Conteúdo:** Qual o formato do conteúdo apresentado?
3. **Preço:** Todos os usuários conseguem acessar todo o conteúdo de forma gratuita?
4. **Portabilidade:** É possível acessar a ferramenta de qualquer dispositivo?

[10] https://www.freecodecamp.org/
[11] https://www.sololearn.com/home
[12] https://getmimo.com/

Utilizando os critérios estabelecidos no passo anterior, foi construída a seguinte tabela:

	Gerenciamento do Conteúdo	**Estruturação do Conteúdo**	**Preço**	**Portabilidade**
FreecodeCamp	Centralizada	Cursos com conteúdo próprio, seguindo uma abordagem mais tradicional	Gratuito	Somente Web
Github (Roadmaps)	Compartilhada	Estruturação do caminho da aprendizagem, adicionando links para conteúdo em outras plataformas	Gratuito	Somente Web
SoloLearn	Centralizada	Conteúdo próprio em conjunto com exercícios e desafios	Freemium	Web + Aplicativo
Mimo	Centralizada	Aprendizagem baseada em exercícios / desafios	Freemium	Somente Aplicativo

Tabela 2: Análise de ferramentas baseado nos macrogrupos apresentados

Após essa análise, o principal resultado obtido foi confirmação de que a maioria das plataformas relacionadas apresenta conteúdos centralizados desenvolvidos pelas empresas responsáveis, não permitindo a colaboração entre a comunidade.

Entre as ferramentas mencionadas, somente o Github fornecia um ambiente colaborativo. Entretanto, diferentemente das outras plataformas, não há nenhuma funcionalidade voltada para auxiliar no aprendizado ou facilitar a construção de documentações.

Posteriormente, com a fase de Imersão já finalizada, foi realizada uma nova análise com ferramentas mais similares ao escopo definido para o trabalho naquele momento: elaboração colaborativa da documentação do desenvolvedor na plataforma

Learning Curve. Essas ferramentas - Docusaurus e Read the Docs, mencionadas anteriormente - são voltadas à construção de documentações de *software*.

	Ferramenta de documentação	**Disponibilizaçã o do conteúdo**	**Colaboração entre usuários**	**Preço**
Docusaurus	Barra de pesquisa Suporte à Markdown Multi-idiomas	*Self-hosted*	Não disponível	Gratuito e open-source
Read the Docs	Barra de pesquisa Suporte à Markdown	Hospedado na plataforma	Não disponível	Gratuito para indivíduos Pago para empresas

Tabela 3: Nova análise de ferramentas, focada em ferramentas para construção de documentações de *software*

Síntese e análise

Na fase de Síntese e Análise, todo o conhecimento adquirido foi sumarizado. Uma das técnicas do Design Thinking é a elaboração de Cartões de *Insight*: cada um desses cartões apresenta um fragmento de informação coletada na fase de Imersão e sua respectiva fonte. O objetivo é fornecer um repositório central de informações chaves, permitindo uma consulta rápida nas próximas etapas.

Essa etapa foi realizada em dois momentos: assíncrona e sincronamente. Primeiramente, todos os envolvidos na concepção da plataforma escreveram cartões com informações consideradas relevantes, separadamente e sem discussão prévia. Após a escrita dos cartões, foi realizado um encontro síncrono para revisão e discussão entre os colaboradores. Nesta revisão, alguns cartões foram agrupados para evitar duplicidades, ocasionando em um maior detalhamento e, consequentemente, diminuição no número total de cartões.

Posteriormente, um Diagrama de Afinidade é construído. Esse diagrama agrupa cartões similares em colunas representando diferentes temas. A ideia é identificar conexões entre assuntos e mapear oportunidades para o projeto.

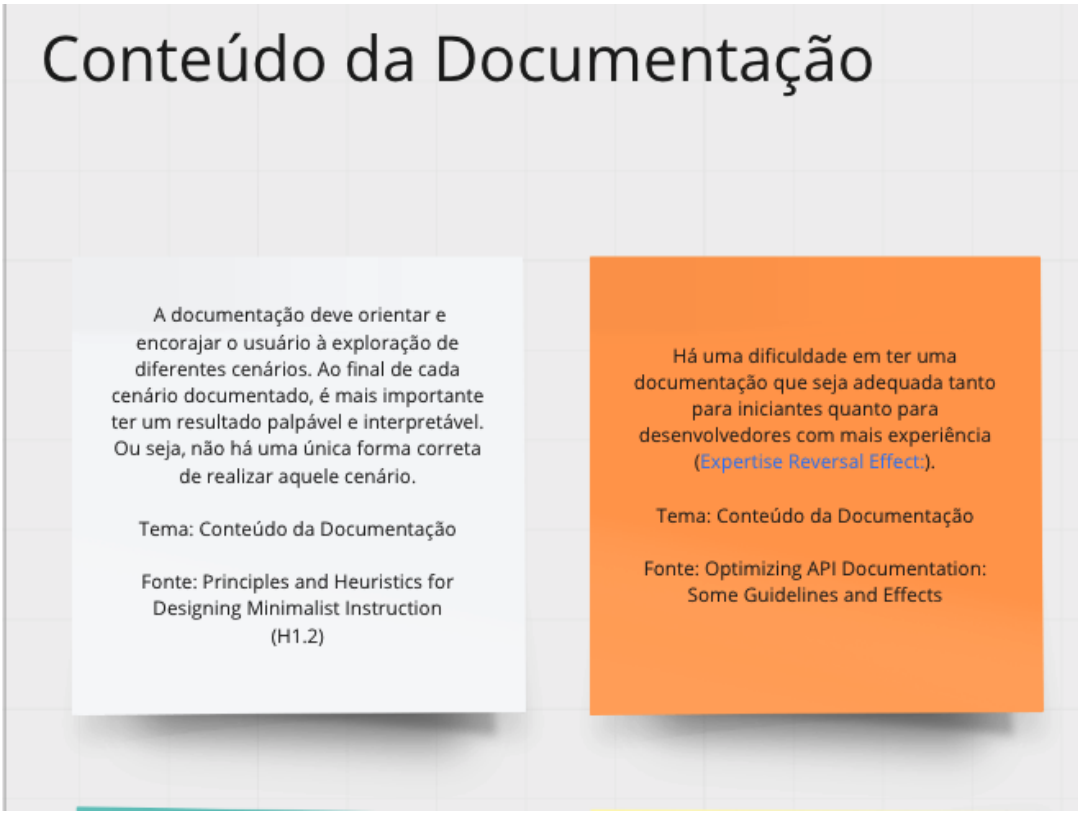

Figura 4: Exemplos de cartões de *Insight,* agrupados por afinidade

Um exemplo do trabalho resultante dessa fase pode ser visto na Figura 4, onde os cartões cinza e laranja foram agrupados sob o mesmo tema. Os dois cartões, originados de diferentes trabalhos, referem-se a informações úteis sobre o conteúdo da documentação.

Ideação

Antes da ideação da solução, o conceito de documentação de desenvolvedor foi dividido em duas subcategorias: documentação conceitual e documentação orientada a tarefas.

A primeira refere-se ao tipo mais comum, no qual a documentação apresenta uma mescla de conteúdo teórico e prático para que o leitor consiga aprender o conteúdo de forma mais ampla e generalista. O segundo tipo, entretanto, apresenta um objetivo claro, específico e conciso. A documentação orientada a tarefas é similar a um pequeno tutorial e apresenta uma lista de passos para auxiliar o leitor a alcançar determinado resultado.

A documentação de uma API empresarial pode ser categorizada como uma documentação conceitual enquanto um pequeno tutorial sobre como adicionar uma biblioteca em um projeto pode ser um exemplo de documentação orientada a tarefas.

Após o entendimento do problema e a sumarização do conhecimento, a próxima etapa refere-se à ideação da solução. Na fase de Ideação, a técnica mais comum é denominada *Brainstorming*, processo no qual todos os envolvidos reúnem-se para propor o maior número possível de ideias.

Nessa etapa, os participantes ficaram inicialmente responsáveis por sugerir ideias de forma assíncrona. Em seguida, foi realizado um encontro síncrono para discussão das ideias levantadas.

Do ponto de vista deste trabalho, as ideias foram posteriormente convertidas em estórias de usuários (user story) (Figura 5). Essas estórias são funcionalidades expressas em um baixo nível de abstração, ajudando a definir os requisitos de um projeto usando linguagem natural (WAUTELET et al., 2014). O resultado esperado é um roteiro com as definições básicas das facilidades da plataforma para apoiar o seu desenvolvimento.

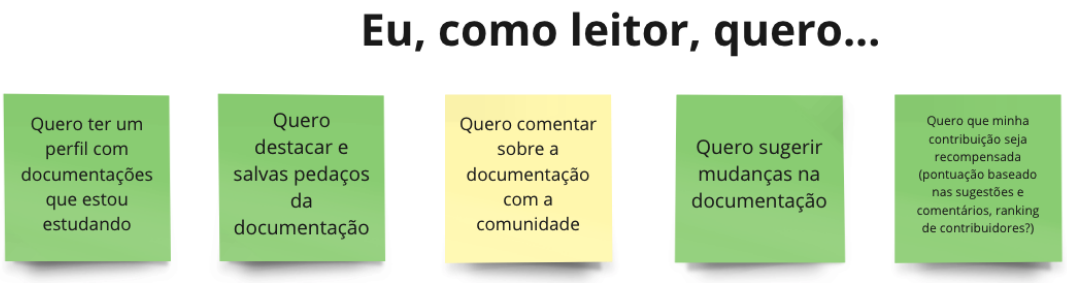

Figura 5: Resultado do processo de *Brainstorming*, com ideias sobre possíveis funcionalidades da aplicação

Para definir mais claramente o escopo do que deveria ser desenvolvido neste trabalho, foi utilizada também a técnica de *User Story Mapping*. Essa técnica permite ordenar as estórias de usuário por prioridade, organizando-as baseado em seu valor do ponto de vista do negócio e utilidade para os usuários (PATTON et al., 2010). Ela pode ser dividida basicamente em 6 etapas (DRUMMOND; ALVES, 2010): listagem das estórias, escrita em cartões, ordenação em fluxo, priorização, agrupamento por funcionalidades e, por fim, definição da primeira entrega.

Por exemplo, para a funcionalidade de visualização de métricas de uma documentação, foi definido que a entrega inicial seria composta somente pela exibição de dados consolidados (Figura 6). A visualização utilizando gráficos e filtros mais

complexos pode ser endereçada em um eventual trabalho posterior. A lista completa de estórias de usuário propostas nesta fase está presente no Apêndice A.

É importante ressaltar que a partir dessa etapa foi definido como as funcionalidades relacionadas à construção, colaboração e visualização de documentações de desenvolvedor.

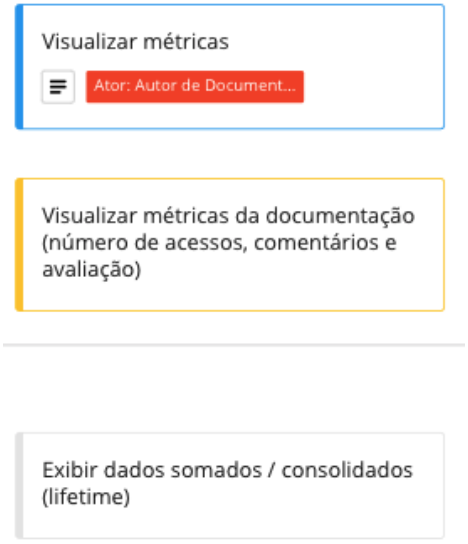

Figura 6: Exemplo do uso do *User Story Mapping* para definição do escopo

Nesta etapa também foram concebidas premissas com potencial para agregar valor à plataforma. Diferentemente das diretrizes, as premissas não foram baseadas na literatura, mas sim em percepções obtidas através da análise de ferramentas similares e também na experiência dos envolvidos neste trabalho com desenvolvimento de software. A seguir, segue um resumo das principais premissas identificadas:

#	Premissa	Confiança
1	Comunicação através de comentários impacta positivamente o engajamento dos usuários	Alta
2	Sistema de pontuação impacta positivamente o engajamento dos usuários	Alta
3	Exportação do conteúdo da documentação é importante para empresas	Baixa

Tabela 4: Premissas concebidas na fase de Ideação da plataforma

A motivação por trás da alta confiança nas duas primeiras premissas é devido ao grande número de exemplos de sucesso. A Premissa #1, referente aos comentários, é baseada na análise de sites como StackOverflow, StackExchange[13] e Quora[14], que baseiam-se na colaboração através de comentários para responder dúvidas dos usuários. Esses sites também apresentam um sistema de pontuação (*gamification*) para incentivar a colaboração (Premissa #2).

Já em relação a Premissa #3, a hipótese é que empresas tendem a preferir manter suas documentações hospedadas por conta própria ao invés de dependerem de uma ferramenta de terceiros. Por conta disso, a possibilidade de exportar a documentação permitiria que elas continuassem utilizando a plataforma como ferramenta de construção das documentações, não necessariamente para visualização. A baixa confiança deve-se ao fato de não ter nenhuma informação disponível na internet sobre o assunto especificamente.

Prototipação

Nessa fase, o objetivo é construir um protótipo para validar a solução proposta anteriormente. Esse protótipo pode ser de baixa fidelidade, como um rascunho simples para ilustrar a ideia, até um produto mínimo viável (MVP), resultando em uma versão inicial do produto final após desenvolvimento de algumas funcionalidades principais.

No caso deste trabalho, um MVP ainda em estágio preliminar foi criado para facilitar a experimentação e análise empírica da solução. Ele será apresentado em maiores detalhes nos próximos capítulos.

[13] https://stackexchange.com/
[14] https://quora.com/

4 Visão Geral da Aplicação

Neste capítulo, a aplicação desenvolvida será apresentada em maiores detalhes. As funcionalidades principais serão discutidas, traçando um paralelo com os conceitos aprendidos ao longo deste texto.

A aplicação está disponível para visualização através do link: https://learning-curve-web.vercel.app.

4.1 Principais entidades

Primeiramente, para facilitar o entendimento, é importante detalhar as seguintes entidades que serão mencionadas ao longo das próximas seções:

- **Pré-requisitos:** Conteúdo a ser estudado antes da leitura de uma documentação. Os pré-requisitos são opcionais e servem como uma recomendação, tendo como objetivo garantir ao estudante uma maior compreensão do assunto a ser tratado.
- **Documentação:** Entidade principal da plataforma. Refere-se a uma documentação de desenvolvedor. A granularidade da informação pode variar, sendo possível criar uma documentação contendo desde um passo-a-passo com a resolução de um único problema até o detalhamento técnico de todos os endpoints de uma API.
- **Projeto**: Responsável por agrupar as documentações de mesmo contexto. Eles podem ser referentes a uma biblioteca, framework ou até mesmo uma aplicação. Alguns possíveis projetos são, por exemplo, a biblioteca Lodash[15], o framework Ruby on Rails[16] e uma aplicação como IMDB API[17].
- **Contribuição:** Refere-se a um pedido de alteração de uma documentação. Essa alteração pode ser uma contribuição para correção de texto, adição de exemplos ou até mesmo a escrita de novas subseções de uma documentação.
- **Usuário**: Entidade responsável por interagir com as outras entidades mencionadas anteriormente.

[15] https://lodash.com/
[16] https://rubyonrails.org/
[17] https://imdb-api.com/

4.2 Gestão de acesso

A visualização do conteúdo da plataforma não exige autenticação, disponibilizando o acesso gratuito para todos os visitantes que tenham interesse nas informações. Entretanto, para criação de projetos, documentações e comentários, torna-se necessário a criação de uma conta na plataforma e, consequentemente, a autenticação na mesma.

É importante destacar que, apesar de conter funcionalidades destinadas a dois tipos distintos de usuários, o criador de documentação e o estudante, a plataforma permite que o usuário exerça os dois papéis utilizando apenas uma conta.

4.3 Páginas

Landing page

Caso o usuário não esteja autenticado, a *Landing Page* será exibida (Figura 7), apresentado com algumas informações básicas sobre a plataforma. A tela apresenta a contagem do número de projetos, documentações e usuários cadastrados na plataforma, com o intuito de aumentar o interesse do usuário no conteúdo disponível. A página apresenta também uma caixa de texto no qual o visitante pode inserir seu e-mail e ser redirecionado para a página de cadastro. É possível também, através do menu superior, navegar entre as páginas disponíveis e visualizar conteúdos que não exigem autenticação.

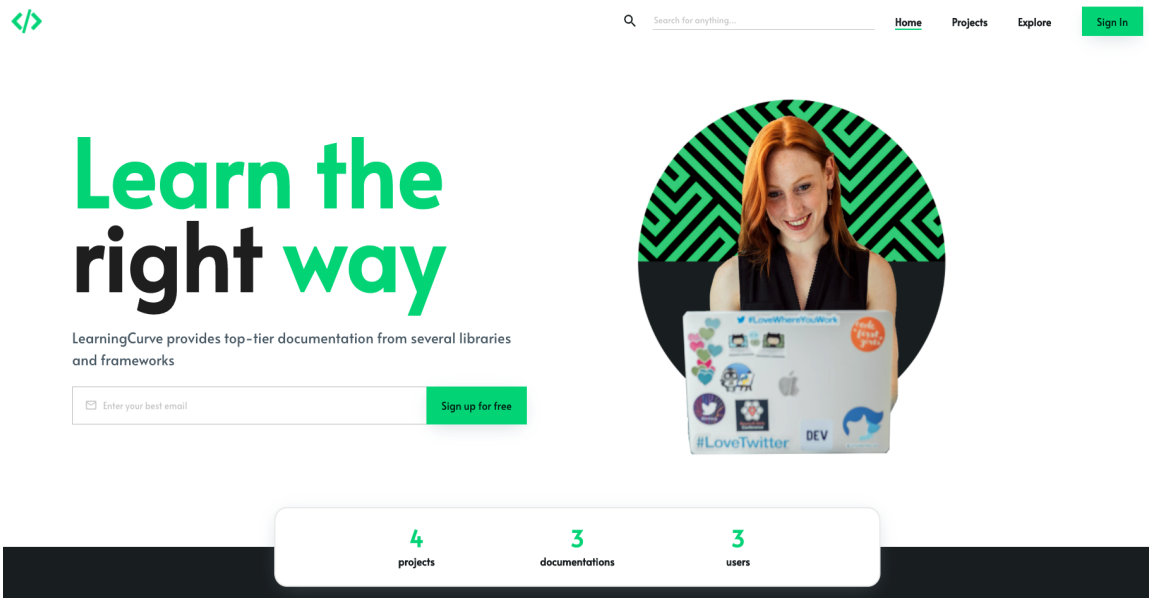

Figura 7: Página inicial da aplicação

Autenticação

A página de autenticação da plataforma apresenta a possibilidade de alternar entre dois formulários: cadastro e *login*.

No formulário de cadastro (Figura 8) , para a criação de uma nova conta, é necessário que o usuário forneça um nome de usuário único, o e-mail a ser associado e uma senha.

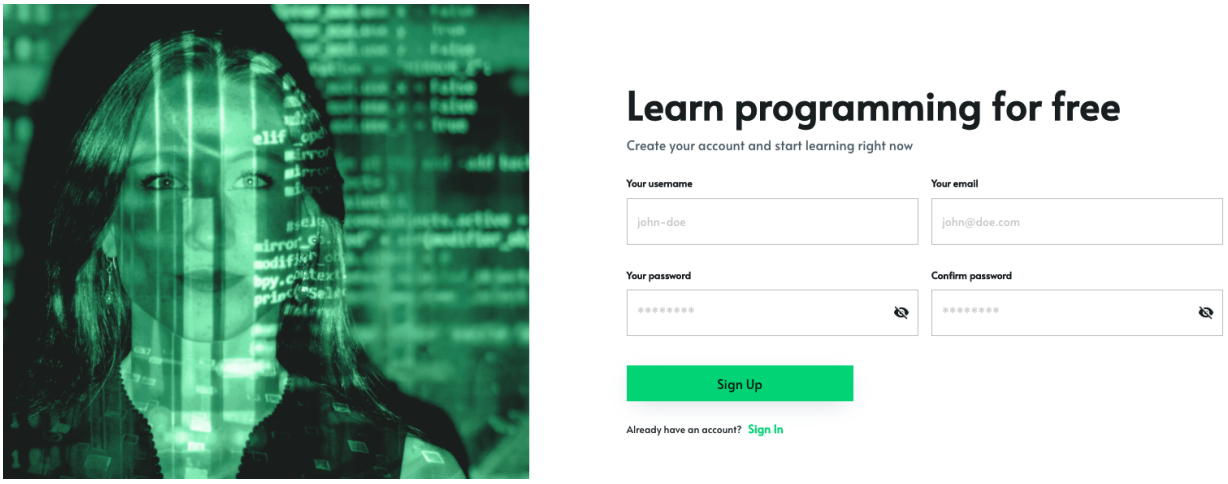

Figura 8: Formulário de cadastro

Caso o usuário já tenha uma conta, ele pode acessar o formulário de *login*. Após o preenchimento do nome de usuário e senha, ele é autenticado e redirecionado para a página *Home*.

Home

Ao acessar a plataforma autenticado, a página inicial mencionada anteriormente é ocultada e uma nova página passa a ser exibida ao usuário: a página *Home* (Figura 9). Essa página apresenta informações customizadas para o usuário autenticado, exibindo o status de suas contribuições, suas documentações favoritas e as informações referentes às suas documentações publicadas. É possível também, através desta página, aprovar ou rejeitar uma contribuição, além de acessar um atalho para criação de uma nova documentação.

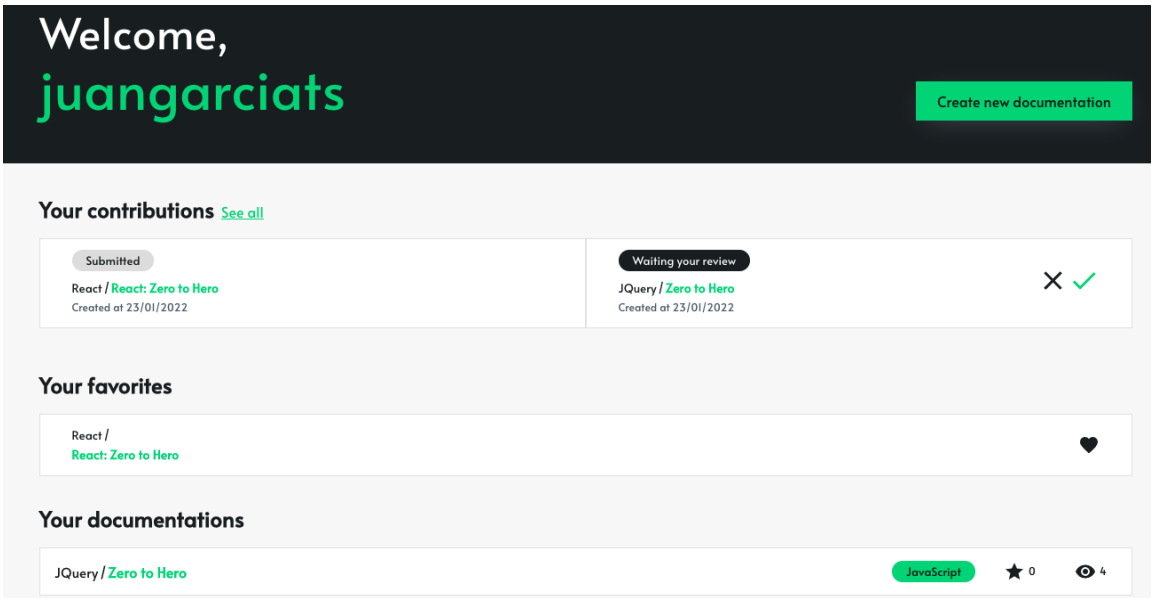

Figura 9: Página *Home* e suas funcionalidades

Projetos

Ao navegar para a página de projetos, é possível visualizar todos os registros criados na plataforma (Figura 10).

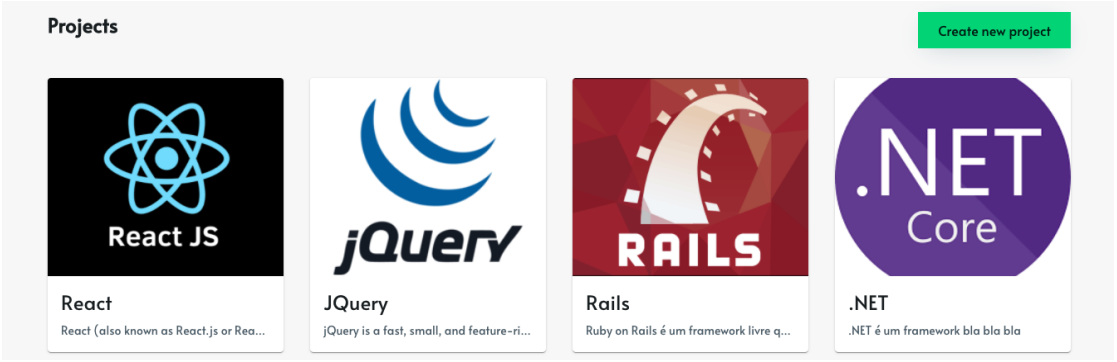

Figura 10: Página de projetos

Ao clicar em um projeto, é exibida a lista de todas as documentações do projeto selecionado (Figura 11).

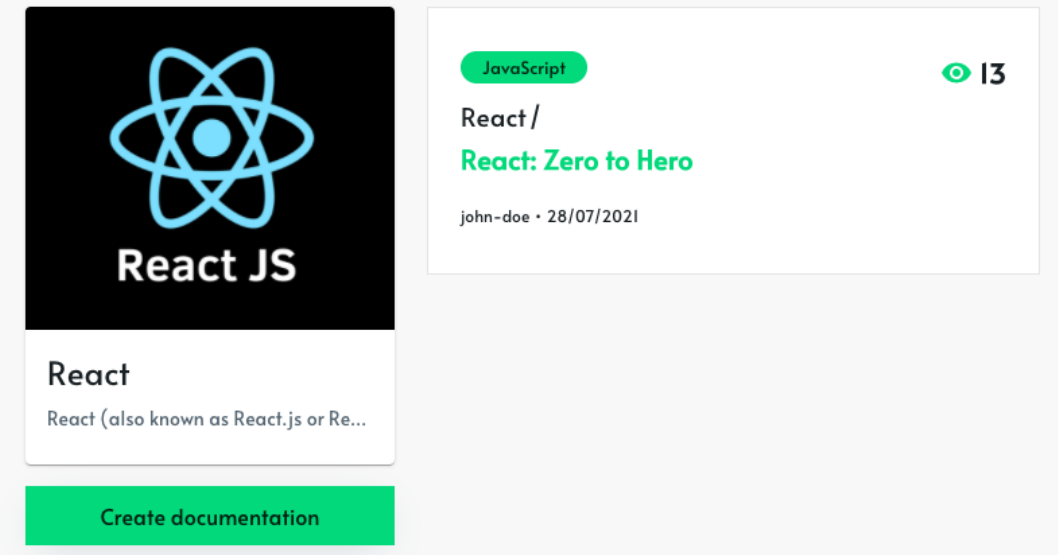

Figura 11: Detalhes de um projeto

Documentações

Para a criação, edição e contribuição de documentações, é disponibilizado um editor de linguagem Markdown, além de uma barra de ferramentas com atalhos rápidos (Figura 12).

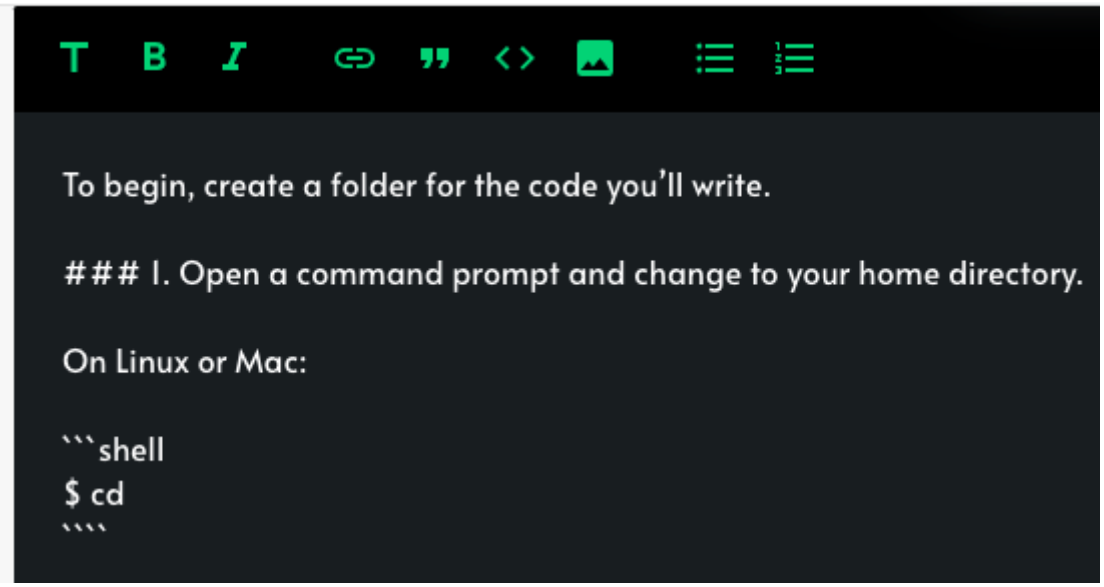

Figura 12: Editor de documentação

Também são disponibilizados *templates* para agilizar a escrita da documentação. Por exemplo, ao digitar o caractere de cifrão ($), uma lista é aberta com possíveis opções (Figura 13). Nesse primeiro momento, a única opção para criação de *cookbooks*, que podem ser classificados como uma espécie de documentação orientada a tarefas,

definida na seção 3.2. Caso a opção seja confirmada, o Markdown é inserido automaticamente no editor (Figura 14).

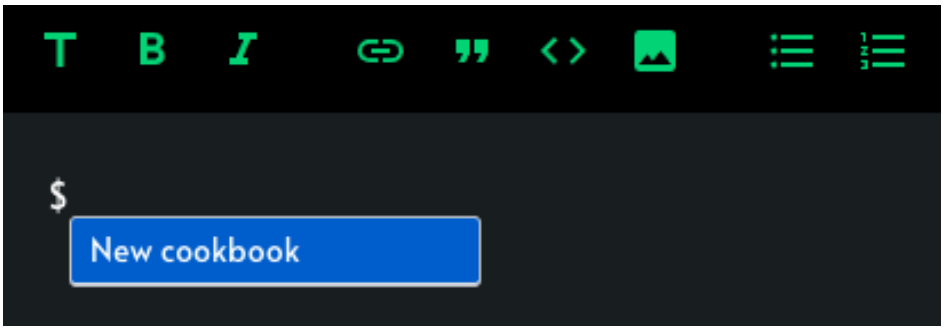

Figura 13: *Templates* disponíveis no editor

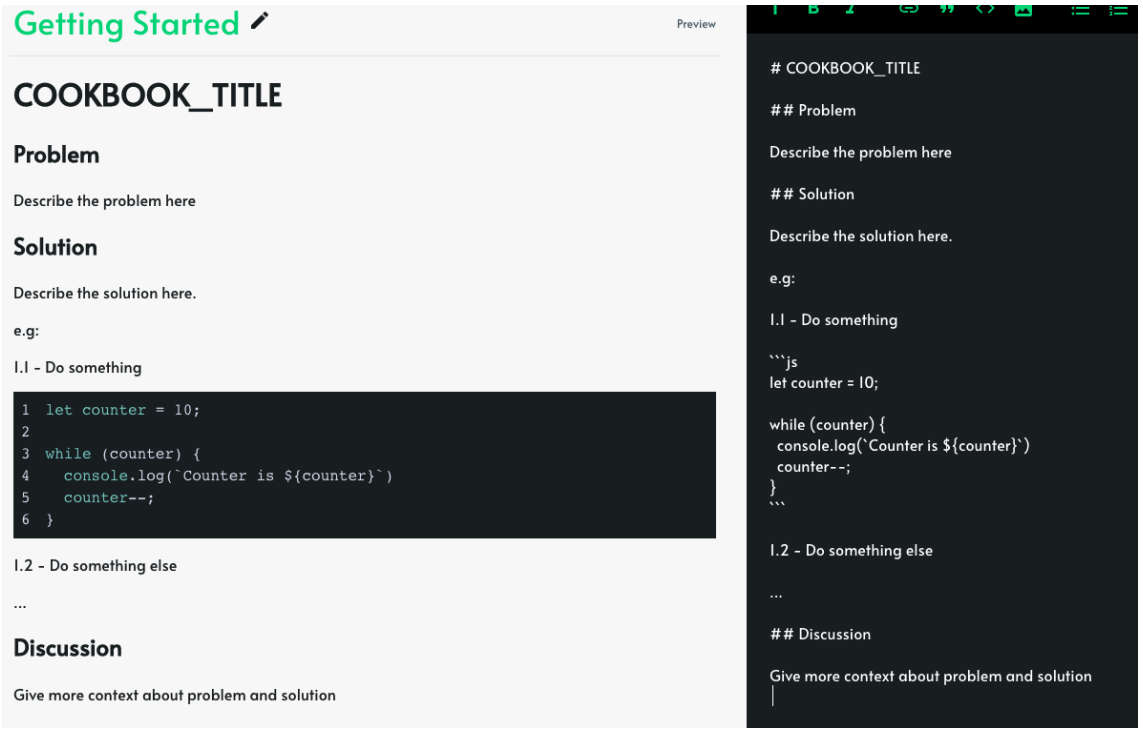

Figura 14: *Cookbook* gerado automaticamente

Em relação à visualização da documentação, cada uma delas apresenta uma página exclusiva na plataforma (Figura 15). A documentação é exposta em três colunas, dividido em menu para navegação, texto da documentação e eventuais exemplos de código, respeitando a Diretriz #3 (Capítulo 2.3).

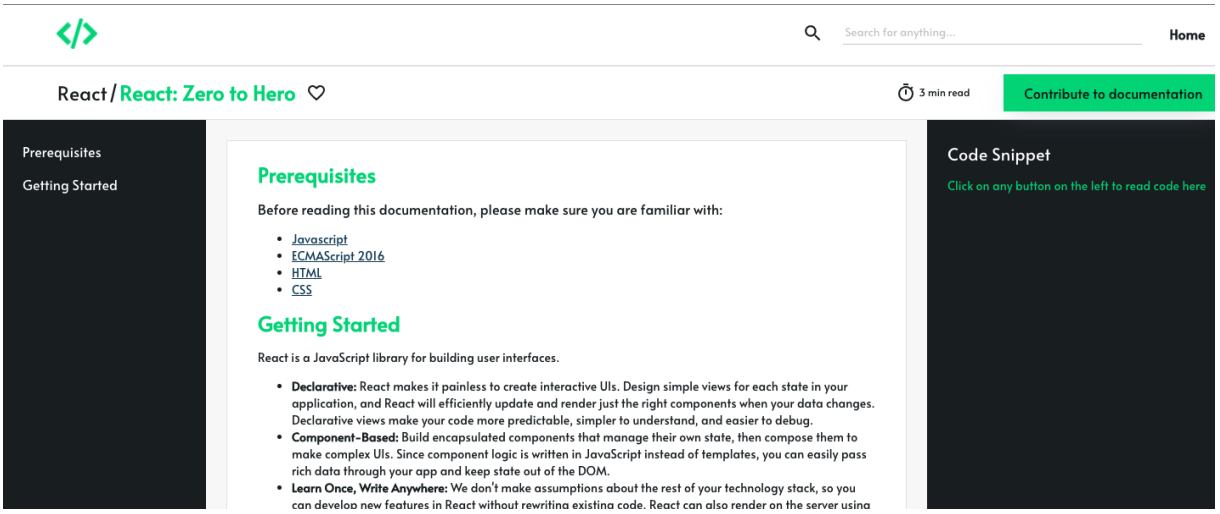

Figura 15: Página de documentação

Os códigos exibidos são formatados de acordo com a linguagem de programação desenvolvida e podem ser copiadas instantaneamente por meio de um botão localizado abaixo do título (Figura 16), respeitando a Diretriz #4.

Figura 16: Coluna com exemplo de código e botão para cópia instantânea

Ao clicar no ícone de comentários presente no menu superior, é possível interagir com outros usuários da plataforma utilizando comentários (Figura 17). Os comentários podem ser respondidos, potencializando o engajamento e a colaboração entre estudantes (Premissa #1).

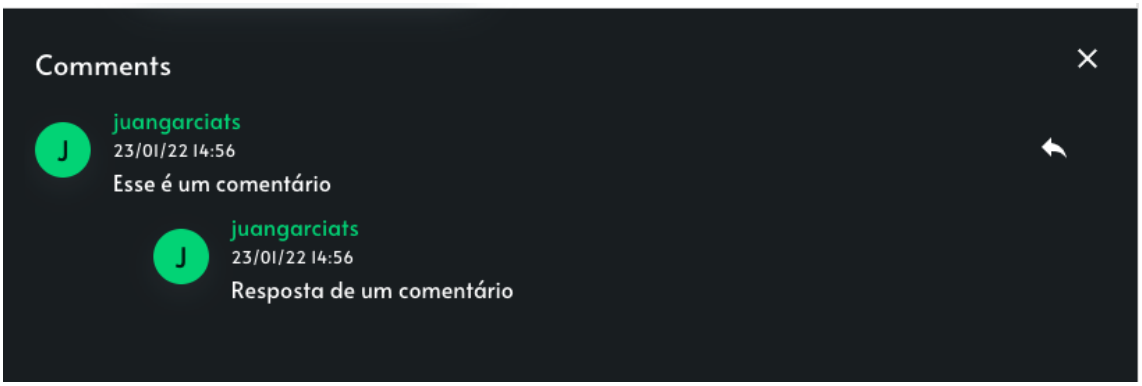

Figura 17: Comentários de uma documentação

Outra funcionalidade presente na página é a possibilidade de exportar toda a documentação para Markdown, caso o usuário tenha interesse em adicioná-la em outra plataforma, relacionada diretamente a Premissa #3 mencionada anteriormente. Um exemplo de Markdown gerado pode ser visto na Figura 18.

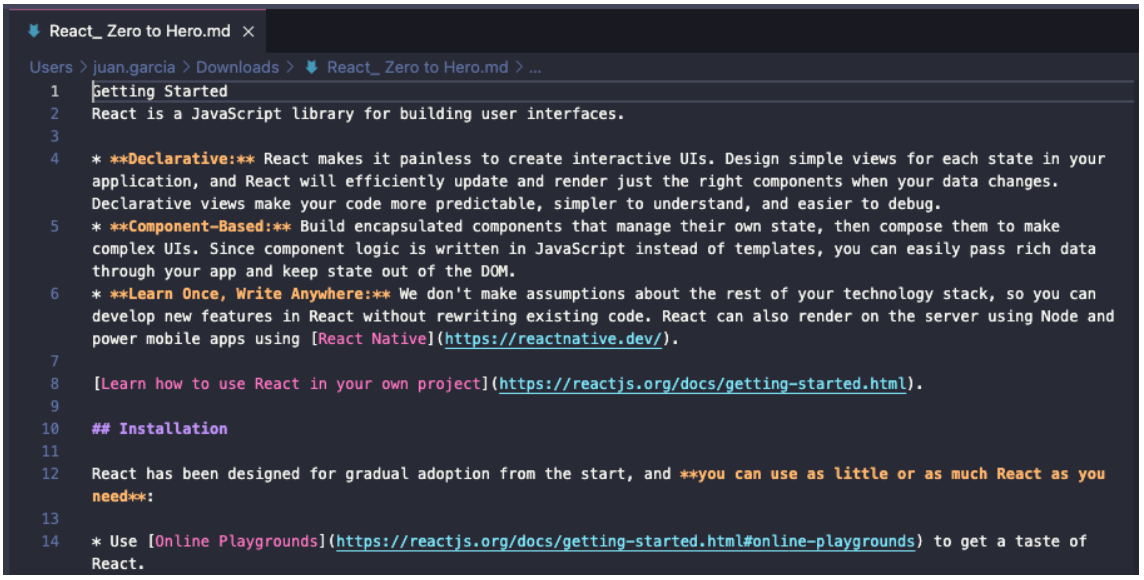

Figura 18: Exemplo de Markdown gerado

Contribuições

Caso o usuário não seja o criador da documentação, o botão de edição mencionado anteriormente é alterado para um botão de contribuição (Figura 19).

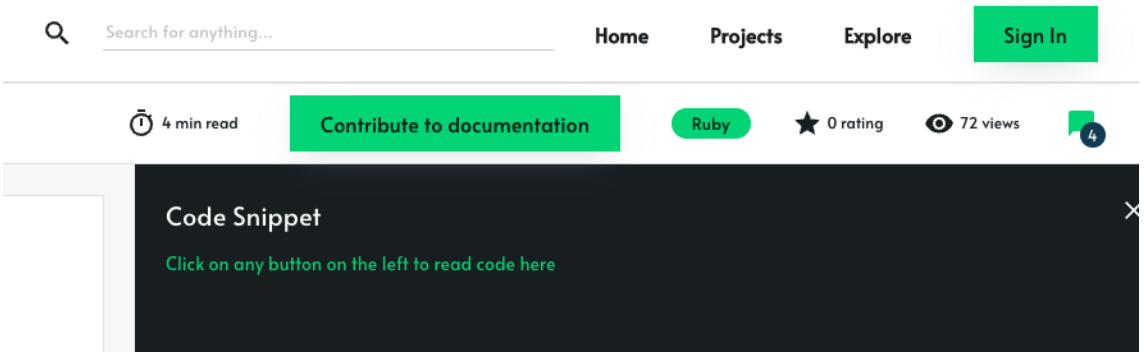

Figura 19: Botão para contribuição em uma documentação

Ao clicar no botão, o usuário é redirecionado para o editor de documentações. A principal diferença entre o modo de edição e de contribuição é que a documentação não é publicada automaticamente após a submissão.

Quando a contribuição é finalizada, é necessário que o autor original da documentação aprove ou rejeite as alterações propostas (Figura 20). As alterações podem ser visualizadas e revisadas em uma página dedicada (Figura 21).

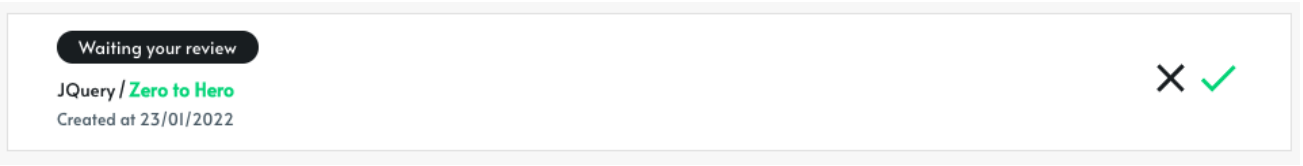

Figura 20: Contribuição aguardando revisão

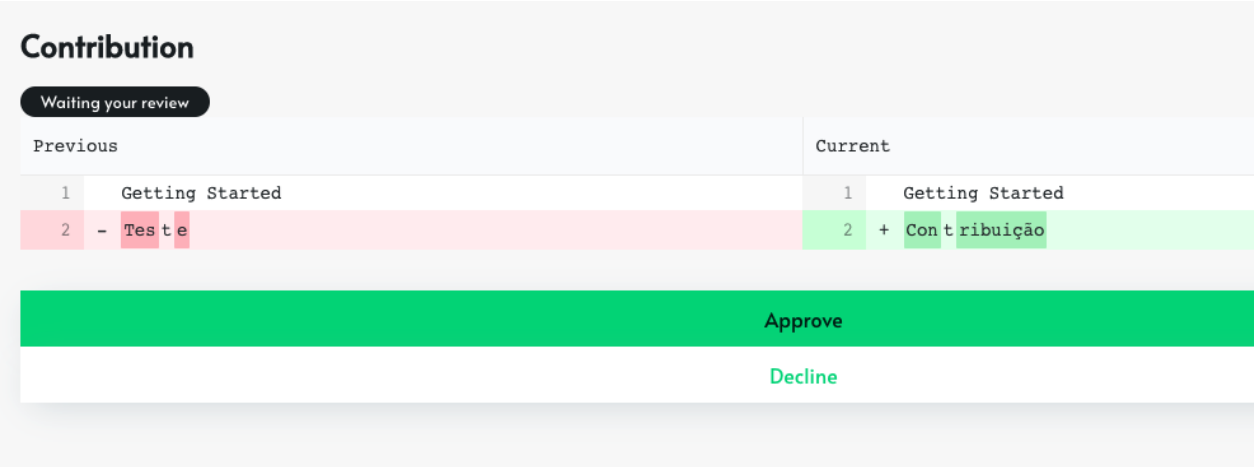

Figura 21: Página com detalhes da contribuição

Existem três possíveis status para uma contribuição: (i) "Submetida", quando contribuição é enviada para outro usuário e é necessário aguardar revisão; (ii) "Aguardando revisão", quando alguém contribui para uma documentação de autoria do usuário, exigindo uma revisão; (iii) "Revisada", quando o autor aprova ou recusa a contribuição.

Quando a contribuição é aprovada, a alteração é automaticamente exibida. Caso a contribuição seja rejeitada, nenhuma ação é realizada.

Pesquisa

Como discutido no capítulo de fundamentação teórica (Diretriz #2), é extremamente importante fornecer mecanismos que permitam o usuário pesquisar sobre conteúdos específicos de uma documentação. Por conta disso, no menu de navegação da plataforma, há uma barra de pesquisa que permite buscar projetos, documentações e usuários através de um termo. O resultado retornado aparece abaixo da barra de pesquisa, contendo o nome do elemento e um ícone indicando seu tipo. (Figura 22)

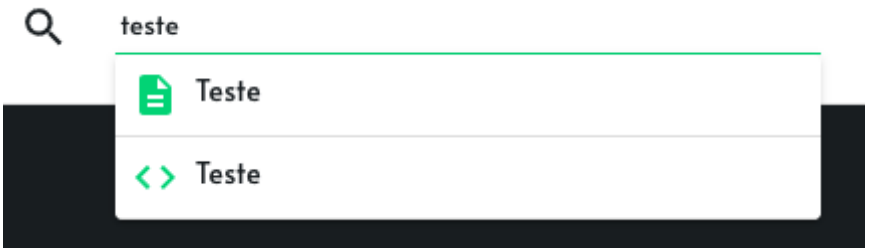

Figura 22: Resultados de uma busca na plataforma através da barra de pesquisa

Também é disponibilizada uma página exclusiva para pesquisa, possibilitando uma visão mais detalhada dos resultados. Ela inclui, por exemplo, o número de resultados retornados e a descrição do tipo dos elementos retornados (Figura 23).

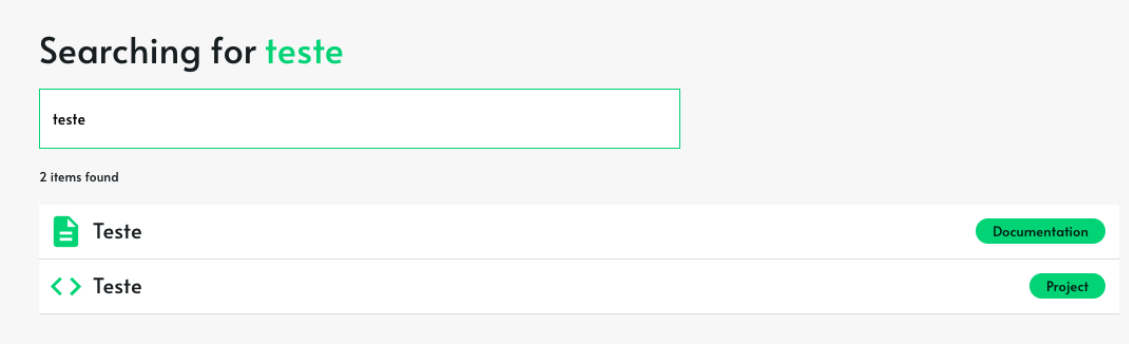

Figura 23: Resultados de uma busca na plataforma através da página de pesquisa detalhada

Explorar

A página Explorar (Figura 24) lista todas as documentações publicadas na ferramenta. Ela permite filtrar os resultados por linguagem de programação e ordená-los por número de visualizações ou por data de criação.

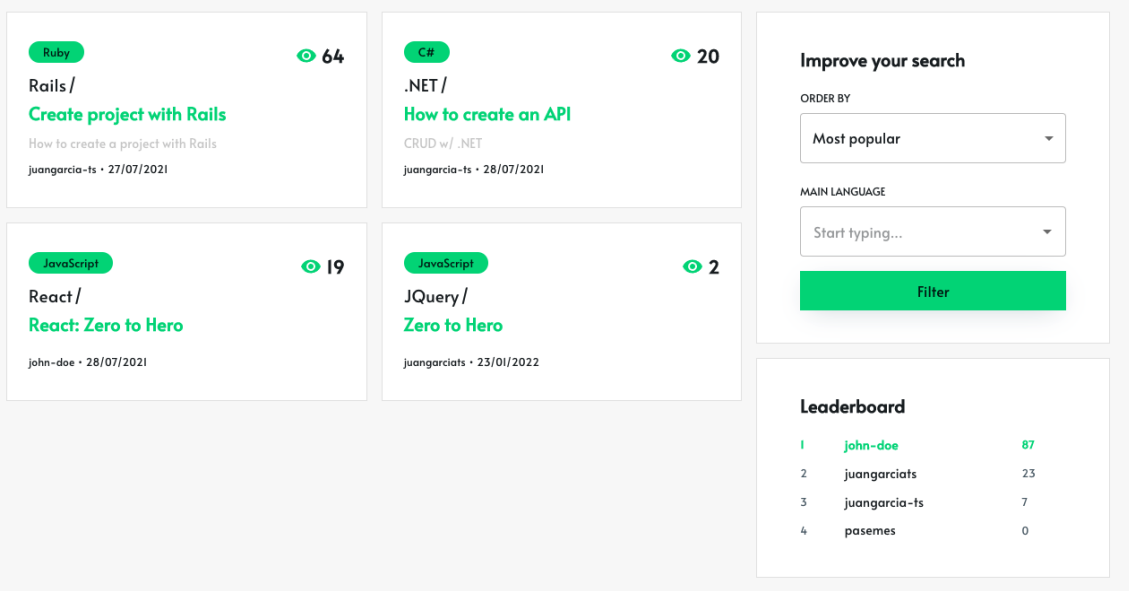

Figura 24: Página Explorar

Nessa página também é exibido o ranking de usuários. Com o objetivo de engajar o engajamento dos usuários na plataforma e a colaboração na construção das documentações (Premissa #2), algumas ações específicas fornecem uma pontuação, como visto na tabela abaixo.

Ação do usuário	Pontuação obtida
Comentário realizado	1
Documentação criada	5
Contribuição enviada	15
Contribuição aprovada	50

Tabela 5: Pontuação obtida por ação do usuário

Perfil do usuário

Dentro da página de perfil (Figura 25), é possível visualizar informações como nome, data de criação e pontos do usuário, além de todas as documentações publicadas e as atividades realizadas. Nesse primeiro momento, é possível visualizar somente dois tipos de atividades: (i) documentação criada e (ii) projeto criado.

Além disso, caso o perfil visualizado seja do usuário autenticado, é possível realizar o *logout* da plataforma.

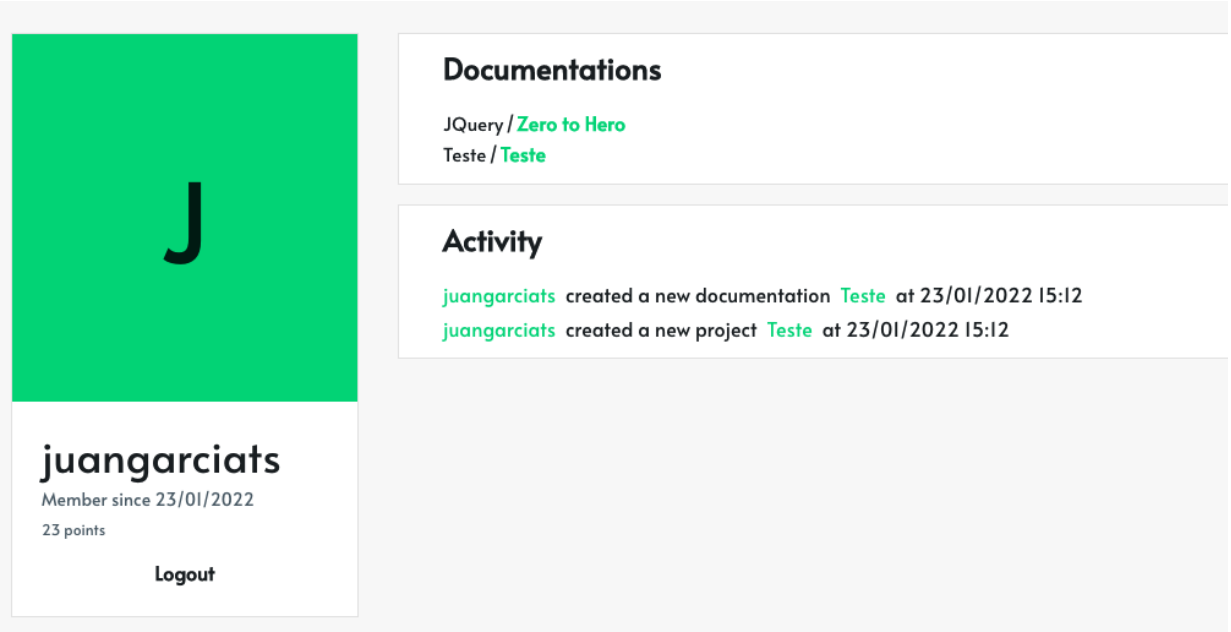

Figura 25: Perfil de um usuário na plataforma

5 Arquitetura e Design do Sistema

Embora o objetivo deste trabalho seja o rápido desenvolvimento de um protótipo, foram tomadas algumas decisões técnicas visando a escalabilidade da aplicação e uma eventual expansão da plataforma em trabalhos futuros.

Esse capítulo apresenta as decisões técnicas relacionadas a arquitetura e o design da aplicação, detalhando principalmente as tecnologias e bibliotecas utilizadas na concepção do *software*.

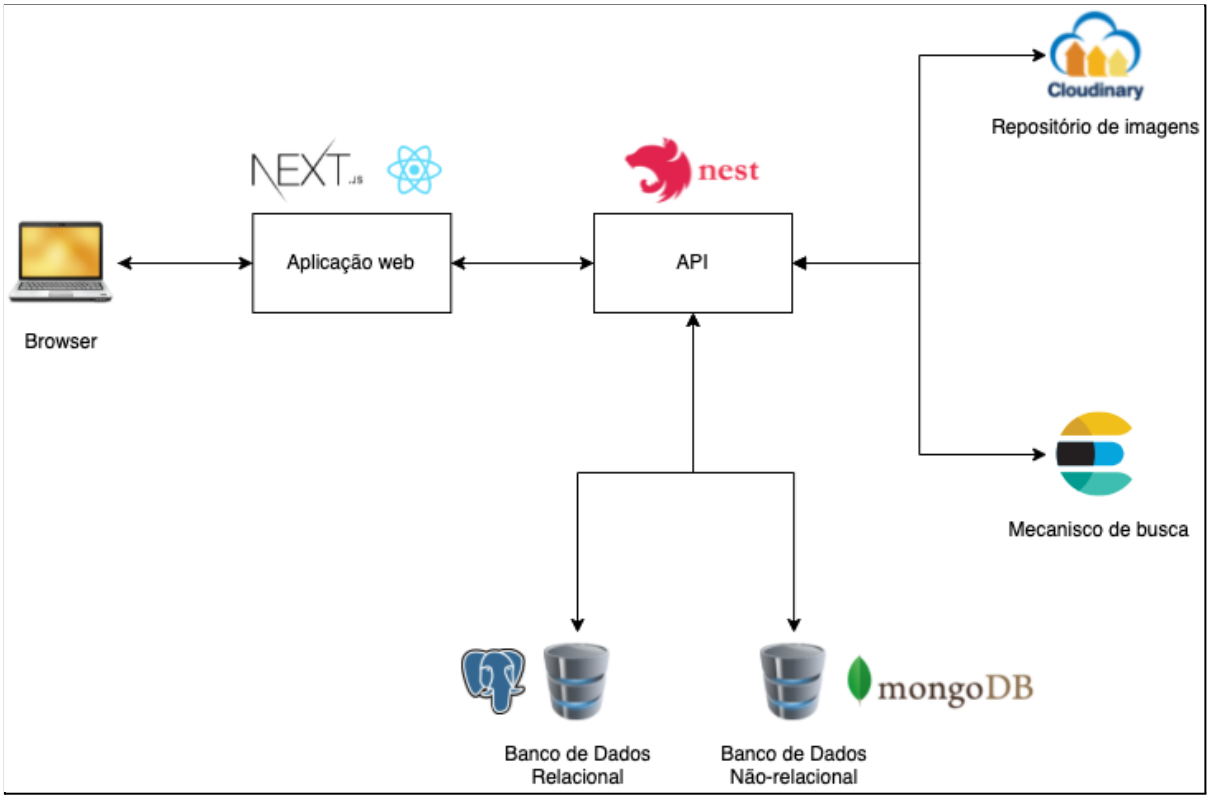

Figura 26: Diagrama simplificado com a arquitetura do sistema e as tecnologias utilizadas

5.1 Linguagem de programação

A linguagem de programação escolhida para o desenvolvimento da aplicação foi o Typescript[18], desenvolvida com base na linguagem Javascript[19], possibilitando a tipagem estática do código e também uma abordagem multi-paradigma. Apesar de ter sido desenvolvida por uma empresa de capital aberto, a Microsoft[20], o código-fonte da linguagem é aberto.

A motivação por trás da utilização da linguagem é, devido à compatibilidade com o Javascript e o número expressivo de bibliotecas e ferramentas disponíveis tanto para o desenvolvimento de aplicações *front-end* quanto *back-end*.

5.2 Aplicação web

Para o desenvolvimento da aplicação web, foi utilizado React, uma biblioteca Javascript de código aberto para construção de interfaces de usuário, mantida pela Meta. Ela pode ser utilizada tanto para construção de aplicações web quanto *mobile*. Ela segue uma abordagem declarativa e permite a construção de interfaces complexas a partir de um conjunto de elementos menores isolados chamados "componentes". A biblioteca, entretanto, não apresenta todos os recursos necessários para desenvolvimento de um site, carecendo de algumas funcionalidades importantes como ferramentas de roteamento (navegação), empacotamento de módulos, otimização de imagens, entre outros.

Para resolver os problemas citados no parágrafo anterior, também foi utilizado o framework NextJS[21], mantido pela empresa Vercel[22]. Além de fornecer as funcionalidades citadas anteriormente, ele apresenta outros recursos como possibilidade de pré-renderização das páginas no servidor - facilitando a indexação do site em motores de busca (Search engine optimization ou SEO) - e suporte a *Fast Refresh* no ambiente de produção, permitindo que alterações realizadas no código possam ser visualizadas sem a necessidade de recarregar a página.

Em relação à hospedagem da aplicação web, foi utilizada a plataforma da própria Vercel, mantenedora do NextJS. Ela é destinada para hospedagem de diferentes

[18] https://www.typescriptlang.org/
[19] https://www.javascript.com/
[20] https://www.microsoft.com/
[21] https://nextjs.org/
[22] https://vercel.com/

tipos de sites estáticos e frameworks front-ends. A principal motivação para sua utilização é o suporte nativo ao framework NextJS e integração *out-of-the-box* com o Github ao implantar a aplicação para o ambiente de produção.

5.3 API

Em relação ao desenvolvimento da API, o framework escolhido foi o NestJS[23]. Segundo o site oficial da ferramenta, o "NestJS é um framework para construção de aplicações Node.JS do lado do servidor eficientes e escaláveis" (tradução nossa).

O framework utiliza internamente o framework Express como padrão para criação de servidores HTTP, facilitando a construção de APIs. Além disso, ele também fornece suporte ao Typescript e disponibiliza um conjunto extensivo de funcionalidades nativas para serialização, utilização de bancos relacionais e não-relacionais, autenticação e diversas outras integrações.

5.4 Repositório de imagens

Para a hospedagem de imagens, há algumas soluções possíveis como o armazenamento no mesmo servidor da aplicação ou até mesmo a conversão das imagens para binário e armazenamento no banco de dados.

Essas opções, entretanto, acabam causando problemas futuros para os desenvolvedores quando surge a necessidade de escalar a aplicação. A primeira opção mencionada, por exemplo, dificulta o escalonamento horizontal do sistema, ou seja, a adição de novas réplicas da aplicação. A segunda opção, relacionado a conversão das imagens, também causa um grande impacto no banco de dados da aplicação a longo prazo, ocasionando um aumento expressivo da quantidade de *bytes* armazenados.

Uma alternativa é a utilização de provedores na nuvem, que armazenam essas imagens em diversos servidores espalhados pelo mundo. O serviço mais conhecido é o Amazon Simple Storage Service[24] (S3) que, segundo a própria Amazon [3], é "um serviço de armazenamento de objetos que oferece escalabilidade, disponibilidade de dados, segurança e performance líderes do setor. Isso significa que clientes de todos os tamanhos e setores podem usá-lo para armazenar e proteger qualquer volume de dados em vários casos de uso, como data lakes, sites, aplicações para dispositivos móveis,

[23] https://nestjs.com/
[24] https://aws.amazon.com/pt/s3/

backup e restauração, arquivamento, aplicações empresariais, dispositivos IoT e análises de big data.".

Para essa aplicação, entretanto, foi escolhida uma alternativa gratuita ao S3: o Cloudinary[25]. Assim como o serviço da Amazon, o Cloudinary fornece serviços de gerenciamento de arquivos na nuvem, porém fornece uma conta gratuita com capacidade de até 25 GB de armazenamento. Para integração e *upload* das imagens, é fornecida uma biblioteca em Javascript.

```
import {
  v2 as Cloudinary,
  UploadApiOptions,
  UploadApiResponse,
} from 'cloudinary';

You, 4 months ago | 1 author (You)
@Injectable()
export class CloudinaryClient {
  constructor() {
    Cloudinary.config({
      cloud_name: process.env.CLOUDINARY_CLOUD_NAME,
      api_key: process.env.CLOUDINARY_API_KEY,
      api_secret: process.env.CLOUDINARY_API_SECRET,
    });
  }

  async uploadImage(
    file: string,
    path: string,
    overwrite = true,
  ): Promise<UploadApiResponse> {
    return this.upload(file, {
      resource_type: 'image',
      public_id: `${process.env.CLOUDINARY_ROOT_FOLDER}/${path}`,
      overwrite,
    });
  }

  private async upload(
    file: string,
    config: UploadApiOptions,
  ): Promise<UploadApiResponse> {
    return new Promise((resolve, reject) => {
      Cloudinary.uploader.upload(file, config, (error, result) => {
        if (error) {
          reject(error);
          return;
        }

        resolve(result);
      });
    });
  }
```

Figura 27: Código para *upload* de imagens no Cloudinary

[25] https://cloudinary.com/

5.5 Mecanismo de busca

Com o intuito de fornecer um mecanismo de busca robusto, a tecnologia utilizada foi o Elasticsearch[26], um motor de busca baseado na biblioteca Apache Lucene[27]. Ele permite a busca e análise de dados textuais, numéricos, geoespaciais, estruturados e não estruturados de maneira fácil, fornecendo uma poderosa API Rest para manipulação desses dados. De acordo com o resultado fornecido pelo ranking DB-Engines[28], o Elasticsearch é o motor de busca empresarial mais utilizado no mundo.

Foi desenvolvido um serviço responsável pela inserção dos registros utilizando a biblioteca Javascript do Elasticsearch. Para manter os dados sincronizados com o banco de dados, foi utilizada uma abordagem baseada em eventos. A classe *ElasticsearchEventHandler* (Figura 28) consome os principais eventos disparados pela aplicação - como a criação de documentações e usuários (Figura 29) - e insere os registros manualmente através do serviço de sincronização (Figura 30).

```
export class ElasticsearchEventHandler {
  constructor(
    private readonly postgresManager: EntityManager,
    private readonly service: ElasticsearchSyncService,
  ) {}

  @OnEvent(Event.USER_CREATED)
  async handleUserCreatedEvent(payload: EventDTO): Promise<void> {
    const user = await this.postgresManager.findOneOrFail(User, payload.id);
    await this.service.insert('users', user);
  }

  @OnEvent(Event.PROJECT_CREATED)
  async handleProjectCreatedEvent(payload: ProjectCreatedEventDTO): Promise<void> {
    const project = await this.postgresManager.findOneOrFail(Project, payload.id);
    await this.service.insert('projects', project);
  }

  @OnEvent(Event.DOCUMENTATION_CREATED)
  async handleDocumentationCreatedEvent(payload: DocumentationCreatedEventDTO): Promise<void> {
    const doc = await this.postgresManager.findOneOrFail(Documentation, payload.id);
    await this.service.insert('docs', doc);
  }
}
```

Figura 28: Classe responsável por consumir os eventos e sincronizar com o Elasticsearch

[26] https://www.elastic.co/pt/what-is/elasticsearch
[27] https://lucene.apache.org/
[28] https://db-engines.com/en/

```
async create(newUser: CreateUserDTO): Promise<User> {
  const user = await this.repository.save(newUser);

  this.eventEmitter.emit(Event.USER_CREATED, new EventDTO({ id: user.id }));

  return user;
}
```

Figura 29: Exemplo de evento de criação de usuário

```
export class ElasticsearchSyncService {
  constructor(private readonly client: ESClient) {}

  async createIndexIfNotExists(indexName: string): Promise<void> {
    try {
      await this.client.indices.get({ index: indexName });
    } catch (e) {
      await this.client.indices.create({ index: indexName });
    }
  }

  async insert(indexName: string, body: Record<string, any>): Promise<void> {
    await this.createIndexIfNotExists(indexName);
    await this.client.index({ index: indexName, body });
    await this.refreshIndex(indexName);
  }
}
```

Figura 30: Serviço de inserção de dados no Elasticsearch

5.6 Banco de dados

A aplicação utiliza dois bancos de dados diferentes, PostgreSQL[29] e MongoDB[30], com casos de uso diferentes.

PostgreSQL

PostgreSQL é um banco de dados relacional, multiplataforma e de código-aberto, permitindo a utilização de transações ACID (Atomicidade, Consistência, Isolamento e Durabilidade). Esse banco, presente há mais de 20 anos no mercado, apresenta as principais funcionalidades de um banco relacional, como chaves estrangeiras, gatilhos e *views*.

A maior parte das entidades identificadas no projeto foram convertidas em tabelas no PostgreSQL, mapeando os relacionamentos entre elas. As tabelas da Figura

[29] https://www.postgresql.org/
[30] https://www.mongodb.com/

31, estão principalmente relacionadas aos usuários, projetos e documentações da ferramenta.

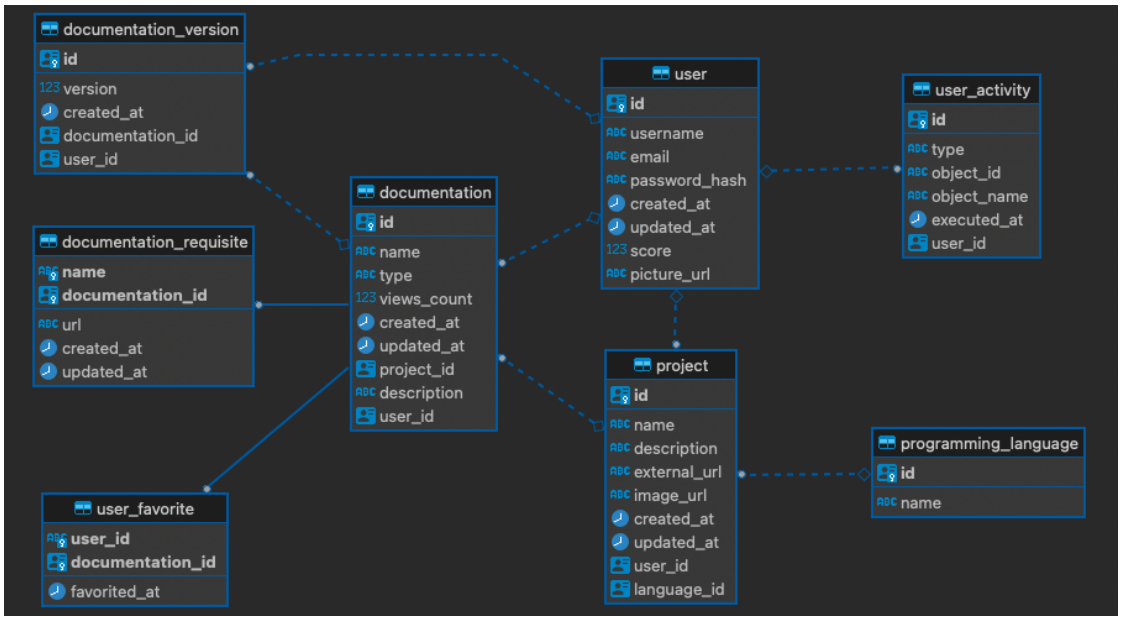

Figura 31: Modelagem do banco de dados relacional

A criação e gerenciamento do banco foi realizada através de um provedor na nuvem chamado ElephantSQL[31] (DBaaS ou DataBase as a Service), fornecendo um plano gratuito para aplicações com baixo volume de dados.

MongoDB

Para facilitar a inserção e visualização de algumas informações, um banco de dados não-relacional também foi utilizado, popularmente conhecido como um banco de dados NoSQL. A ferramenta escolhida foi o MongoDB, multiplataforma e orientada a documentos, que armazena os dados em uma estrutura similar ao formato JSON (Javascript Object Notation). A ferramenta foi desenvolvida e é atualmente mantida pela empresa MongoDB Inc.

No contexto da aplicação, a motivação por trás do seu uso é devido à necessidade de manipulação do conteúdo da documentação, além de respectivas subseções e o versionamento. Cada seção pode conter *n* subseções, resultando em uma estrutura similar à uma árvore. Um exemplo pode ser visto na Figura 32.

[31] https://www.elephantsql.com/

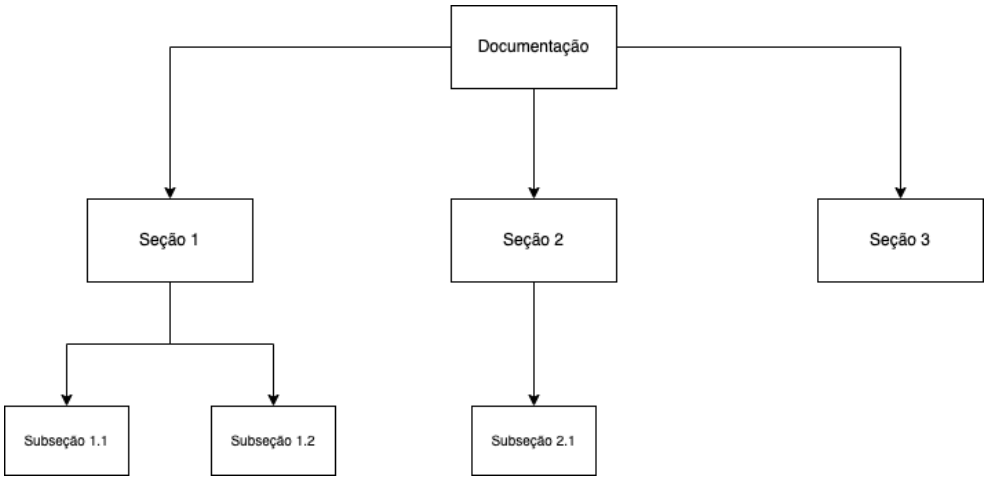

Figura 32: Possível estrutura de uma documentação

Ao ser editado, um novo documento é inserido no banco com um incremento no número da versão. Por conta disso, a estrutura de documento fornecida pelo banco de dados não-relacional diminui a complexidade da inserção e obtenção desse tipo de estrutura. Um exemplo de documento pode ser visto na Figura 33.


```
_id: ObjectId("611c291942cf4f001530119d")
sections: Array
  0: Object
     name: "Outro assunto"
     slug: "outro-assunto"
           "Testando

     content: ```typescript
           const x = 10;
           ```"
  1: Object
     name: "Getting Started"
     slug: "getting-started"
           "

     content:
           ## What's Rails?

           Rails is a web-application framework that include..."
  2: Object
     name: "Passo-a-passo"
           "# Welcome to Rails

           (A2) Criar botão
     content:
           # Passo 2: Criar botão

           Se , v..."
     slug: "passo-a-passo"
version: 4
documentationId: "8476780b-6c93-4808-ad70-ddd39cea2068"
createdBy: "a47e81a3-60af-419b-97e1-6feedf52579e"
createdAt: 2021-08-17T21:24:41.922+00:00
__v: 0
```


**Figura 33: Exemplo de documento no MongoDB**
```

Além do conteúdo da documentação, os comentários de cada documentação também são armazenados de forma similar, por conta da estrutura em árvore (comentários podem conter outros comentários como resposta).

Para permitir relacionar os dados entre os diferentes tipos de banco de dados, foi adicionado um novo campo (*documentationId)* no MongoDB, referenciando a chave primária de uma documentação no banco relacional PostgreSQL.

De forma similar ao PostgreSQL, foi utilizado um DBaaS para gerir a infraestrutura do banco. A plataforma escolhida foi o Mongo Atlas[32], desenvolvida pela própria criadora da tecnologia, utilizando o plano gratuito.

[32] https://www.mongodb.com/atlas/database

6 Prova de Conceito

Neste capítulo, a partir da documentação de um *software* existente, serão apontadas características contrárias às recomendações propostas no capítulo de fundamentação teórica (Seção 2.3).

A documentação escolhida é referente à linguagem de programação Go[33], desenvolvida pelo Google[34]. Para fins de simplificação, devido ao seu escopo reduzido, será utilizado na análise somente o tutorial "*Accessing a relational database*"[35] (Acessando um banco de dados relacional, tradução nossa).

6.1 Análise da documentação existente

A documentação relacionada ao "accessing a relational database" é dividida em 10 seções (Figura 34), sendo a primeira destinada aos pré-requisitos necessários e a última para o código completo.

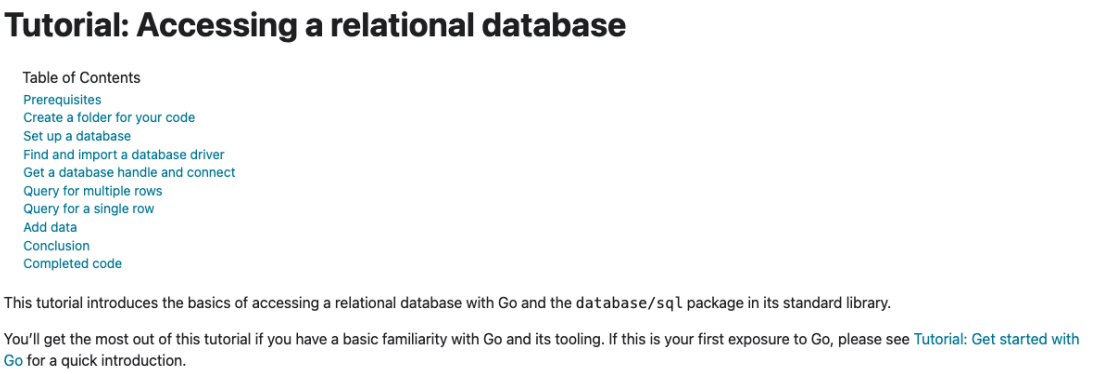

Figura 34: Lista de seções da documentação analisada

Em relação à estrutura do conteúdo, ele é disponibilizado em uma única coluna, indo em direção contrária à Diretriz #3, mencionada na seção 2.3. A plataforma LearningCurve, entretanto, segue a recomendação de três colunas (Figura 35), separando em um menu lateral para navegação, o conteúdo textual e eventuais pedaços de código.

[33] https://go.dev/
[34] https://about.google/
[35] https://go.dev/doc/tutorial/database-access

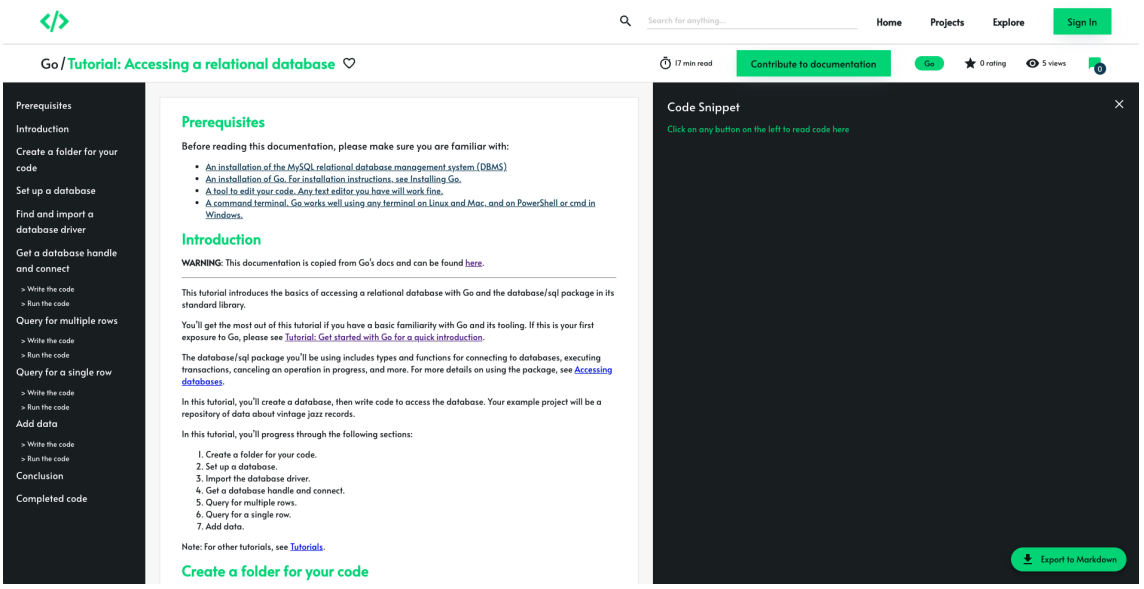

Figura 35: Menu lateral da documentação na plataforma

Além disso, diferentemente da plataforma desenvolvida neste trabalho (Figura 36), a documentação do Google não apresenta nenhuma barra de pesquisa para facilitar a navegação entre o conteúdo, dificultando a busca por um termo específico (Diretriz #2).

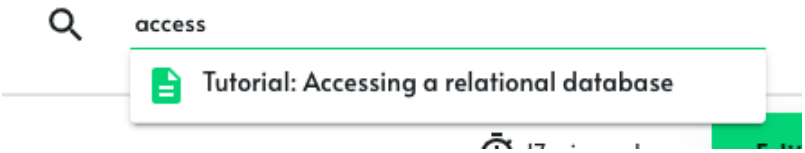

Figura 36: Visualização da nova documentação na barra de pesquisa

Referente aos códigos presentes na documentação (Figura 37), também não há nenhuma funcionalidade que permita que o código seja copiado de forma fácil. O código também não apresenta nenhuma formatação que facilite a leitura. Esses dois aspectos ferem a Diretriz #4. A plataforma LearningCurve, por outro lado, apresenta código-fonte formatado e permite a cópia utilizando somente um clique (Figura 38).

1. Into main.go, beneath the `import` code you just added, paste the following Go code to create a database handle.

```
var db *sql.DB

func main() {
    // Capture connection properties.
    cfg := mysql.Config{
        User:   os.Getenv("DBUSER"),
        Passwd: os.Getenv("DBPASS"),
        Net:    "tcp",
        Addr:   "127.0.0.1:3306",
        DBName: "recordings",
    }
    // Get a database handle.
    var err error
    db, err = sql.Open("mysql", cfg.FormatDSN())
    if err != nil {
        log.Fatal(err)
    }

    pingErr := db.Ping()
    if pingErr != nil {
        log.Fatal(pingErr)
    }
    fmt.Println("Connected!")
}
```

Figura 37: Exemplo de código na documentação da linguagem Go

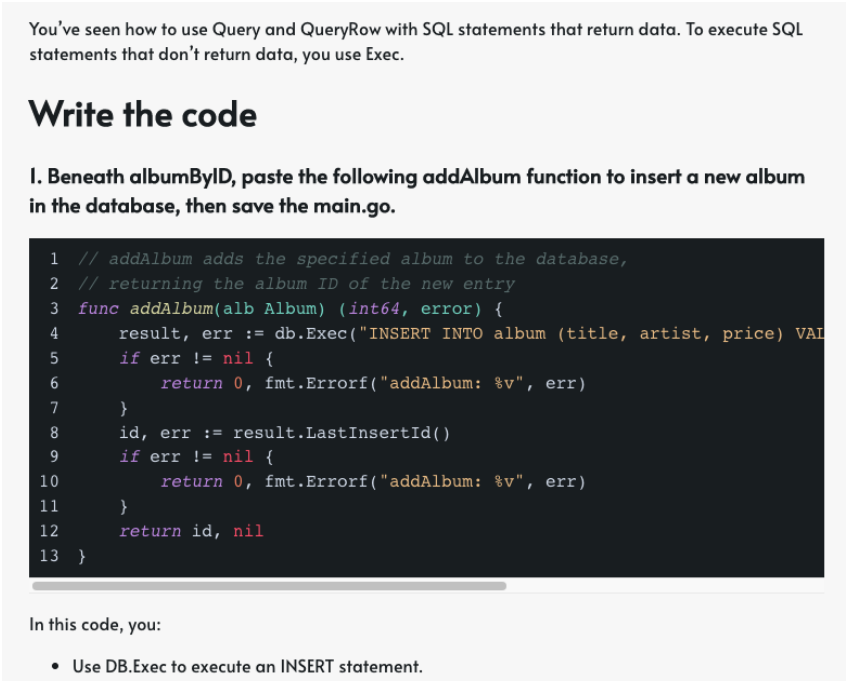

Figura 38: Exemplo de formatação gerada pelo Markdown

Por fim, também é importante destacar que não há nenhuma funcionalidade que permita que um usuário contribua com a documentação, também ignorando as premissas mencionadas.

7 Conclusão

7.1 Considerações finais

Como resultado final deste trabalho, uma plataforma para aprendizado de desenvolvimento de *software* foi desenvolvida. Embora o foco tenha sido o aspecto técnico, também foi possível reunir um conjunto de diretrizes e premissas para construção de documentações de qualidade. Essas informações não são restritas às documentações construídas através da plataforma, permitindo a todos os interessados seguirem as recomendações para aprimorar a escrita de suas documentações.

É importante ressaltar que até mesmo grandes empresas como a Google apresentam documentações problemáticas, como visto no capítulo anterior, reforçando a importância de um apoio ferramental para construção de documentações de alta qualidade, como proposto neste texto.

O foco da plataforma no longo prazo, entretanto, é ser mais que uma simples ferramenta, fornecendo um ambiente colaborativo que potencialize o aprendizado dos estudantes de desenvolvimento de *software*. Sugere-se, portanto, que o trabalho desenvolvido neste trabalho tenha continuidade, tornando a plataforma cada vez mais sofisticada.

7.2 Limitações e próximos passos

Devido à restrições de escopo, não foi possível avaliar de forma empírica o uso da ferramenta. Como indicação de próximos passos, é aconselhável a validação da plataforma e coleta de *feedback* com uma amostra de desenvolvedores.

Caso a avaliação seja positiva, há diversas funcionalidades que não foram desenvolvidas nesta primeira iteração da plataforma (MVP) e podem ser trabalhadas em futuros projetos. Alguns exemplos são: relacionamento e navegação entre documentações da plataforma, gráficos e métricas avançadas de uso, possibilidade de salvar trechos de uma documentação e avaliação qualitativa do conteúdo (clareza, complexidade do assunto, etc..).

Além disso, a gamificação da plataforma é outro aspecto que pode ser desenvolvido com mais profundidade em futuros trabalhos. Na primeira versão da plataforma, foram apresentadas algumas funcionalidades relacionadas, como a pontuação e ranking de usuários, porém ainda há espaço para aprimorá-las e conceber novas funcionalidades que possam potencializar a colaboração e engajamento dos usuários.

8 Referências

ASSOCIAÇÃO DAS EMPRESAS DE TECNOLOGIA DA INFORMAÇÃO E COMUNICAÇÃO. **Relatório Setorial Compacto**. Disponível em: https://brasscom.org.br/wp-content/uploads/2019/05/BRI2-2019-003a-Relat%C3%B3rio-Setorial-Compacto-v13.pdf. Acesso em: 21 nov. 2020.

BURKE, Q. et al. Understanding the Software Development Industry's Perspective on Coding Boot Camps versus Traditional 4-year Colleges. **Proceedings of the 49th ACM Technical Symposium on Computer Science Education - SIGCSE '18**, 2018.

CAMPOS, J. **Um relato de experiência sobre o uso do PDT (Platform Design Toolkit)**. Orientador: Paulo Sérgio Medeiros dos Santos. 2021. 42. Trabalho de iniciação científica – Bacharelado em Sistemas de Informação, Universidade Federal do Estado do Rio de Janeiro, Rio de Janeiro, 2021. Não publicado.

DAGENAIS, B.; ROBILLARD, M. P. Creating and evolving developer documentation. **Proceedings of the eighteenth ACM SIGSOFT international symposium on Foundations of software engineering - FSE '10**, 2010.

DE REUVER, M.; SØRENSEN, C.; BASOLE, R. C. The Digital Platform: A Research Agenda. **Journal of Information Technology**, v. 33, n. 2, p. 124–135, jun. 2018.

DORST, K. The core of "design thinking" and its application. **Design Studies**, v. 32, n. 6, p. 521–532, nov. 2011.

DRUMOND, K; ALVES, L. Uso de user story mapping para planejamento de produtos interativos. **Proceedings of the IX Symposium on Human Factors in Computing Systems (IHC '10)**. p. 243–248, outubro 2010.

GAGGERO, G. Program Documentation. **Programming for Software Sharing**, p. 175–188, 1983.

HEW, K. F.; CHEUNG, W. S. Students' and instructors' use of massive open online courses (MOOCs): Motivations and challenges. **Educational Research Review**, v. 12, p. 45–58, jun. 2014.

KNOWLES, M. S. **Self-directed learning : a guide for learners and teachers**. Englewood Cliffs, N.J.: Prentice Hall Regents, 1980.

LETHBRIDGE, T. C.; SINGER, J.; FORWARD, A. How software engineers use documentation: the state of the practice. **IEEE Software**, v. 20, n. 6, p. 35–39, nov. 2003.

LIEDTKA, J. Perspective: Linking Design Thinking with Innovation Outcomes through Cognitive Bias Reduction. **Journal of Product Innovation Management**, v. 32, n. 6, p. 925–938, 25 mar. 2014.

MENG, M.; STEINHARDT, S. M.; SCHUBERT, A. Optimizing API Documentation. **Proceedings of the 38th ACM International Conference on Design of Communication**, 3 out. 2020.

PATTON, J. et al. **User story mapping : discover the whole story, build the right product**. Beijing ; Sebastopol, Ca: O'reilly, 2014.

REICH, J.; RUIPÉREZ-VALIENTE, J. A. The MOOC pivot. **Science**, v. 363, n. 6423, p. 130–131, 10 jan. 2019.

SILLITO, J.; BEGEL, A. App-directed learning: An exploratory study. **2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)**, maio 2013.

STACK OVERFLOW. **Developer Survey Results 2019**. Disponível em: https://insights.stackoverflow.com/survey/2019#developer-profile-_-other-types-of-education. Acesso em: 22 nov. 2020.

TSCHIMMEL, K. Design Thinking as an effective Toolkit for Innovation. **XXIII ISPIM Conference: Action for Innovation: Innovating from Experience**, jan. 2012.

VAN DER MEIJ, H.; CARROLL, J. M. Principles and Heuristics for Designing Minimalist Instruction. **Technical Communication**, v. 42, n. 2, p. 243–261, 1995.

WAUTELET, Y. et al. Unifying and Extending User Story Models. **Advanced Information Systems Engineering**, v. 8484, p. 211-225, jun. 2014.

Apêndice A - Estórias de Usuário

Estória de usuário (Como usuário, eu quero…)	**Entrega prevista**	**Status**
Criar projeto	MVP	Implementado
Visualizar todos os projetos cadastrados	MVP	Implementado
Editar projeto	MVP	Adiado para Entrega 2
Excluir projeto	MVP	Adiado para Entrega 2
Alterar visibilidade do projeto (privado/público)	MVP	Adiado para Entrega 2
Adicionar pré-requisitos à uma documentação	MVP	Implementado
Escrever documentação	MVP	Implementado
Organizar as seções da documentação através de *drag'n'drop*	Entrega 2	Implementado
Estilizar e formatar documentação	Entrega 2	Implementado
Pesquisar conteúdo	MVP	Implementado
Visualizar sugestões de busca na barra de pesquisa	Entrega 2	Não implementado
Salvar parágrafos "favoritos"	MVP	Adiado para Entrega 2
Validar às modificações da documentação antes da publicação	MVP	Implementado
Escrever comentários sobre a documentação	MVP	Implementado
Responder comentários de uma documentação	Entrega 2	Implementado
Avaliar qualitativamente uma seção da documentação	MVP	Adiado para Entrega 2
Avaliar os pré-requisitos informados na	Entrega 2	Não implementado

documentação		
Avaliar qualitativamente a dificuldade da documentação	Entrega 2	Não implementado
Visualizar métricas da documentação (dados consolidados)	MVP	Implementado
Filtrar métricas da documentação por intervalo de tempo	Entrega 2	Não implementado
Visualizar gráficos sobre as métricas da documentação	Entrega 2	Não implementado
Visualizar pontuação	MVP	Implementado
Exportar documentação (Markdown)	MVP	Implementado