UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

JOSÉ RENAN SILVA LUCIANO

REVIEWBIN:
UMA FERRAMENTA DE FÁCIL USO PARA REVISÃO DE CÓDIGO

CAMPINA GRANDE - PB

2021

JOSÉ RENAN SILVA LUCIANO

REVIEWBIN:
UMA FERRAMENTA DE FÁCIL USO PARA REVISÃO DE CÓDIGO

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Professor Dr. Matheus Gaudencio do Rêgo.

CAMPINA GRANDE - PB

2021

L937r Luciano, José Renan Silva.
Reviewbin: uma ferramenta de fácil uso para revisão de código. / José Renan Silva Luciano. – 2021.

10 f.

Orientador: Prof. Dr. Matheus Gaudencio do Rêgo.

Trabalho de Conclusão de Curso - Artigo (Curso de Bacharelado em Ciência da Computação) - Universidade Federal de Campina Grande; Centro de Engenharia Elétrica e Informática.

1. Engenharia de software. 2. Revisão de código. 3. Desenvolvimento de software. 4. Ferramenta didática para revisão de código. 5. Reviewbin – ferramenta para revisão de código. 6. Aplicação web – revisão de código. 7. Arquitetura Jamstack. 8. Teste de usabilidade. 9. Computer System Usability Questionnaire – CSUQ. 10. Lógica de back-end. I. Rêgo, Matheus Gaudencio do. II. Título.

CDU:004.41(045)

Elaboração da Ficha Catalográfica:

Johnny Rodrigues Barbosa
Bibliotecário-Documentalista
CRB-15/626

JOSÉ RENAN SILVA LUCIANO

REVIEWBIN:
UMA FERRAMENTA DE FÁCIL USO PARA REVISÃO DE CÓDIGO

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

Professor Dr. Matheus Gaudencio do Rêgo

Orientador – UASC/CEEI/UFCG

Professora Dr. Hyggo Oliveira de Almeida

Examinador – UASC/CEEI/UFCG

Professor Dr. Tiago Lima Massoni

Professor da Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 25 de Maio de 2021.

CAMPINA GRANDE - PB

ABSTRACT

Code review is one of the most important steps in the software development process. The practice allows for better quality and reliability of the artifacts, as they will be analyzed and discussed by other developers. Nowadays, some tools enable this process, such as GitHub and ReviewBoard, however, these tools are integrated into a Version Control System (VCS), for instance, Git and Subversion, making it impossible for people who do not know these technologies, such as students beginning in programming courses. There are also tools, like Codepost.io, which present themselves as a proprietary solution for code review in the classroom, but they are limited to correcting activities and add an extra complexity for setting up classes and creating activities on the platform. The objective of this work is to create a tool focused on code review, which allows students and teachers to enjoy the benefits of this practice in classroom activities, without the need of knowing any VCS technology and also be free for other use cases, like sending a code snippet so that someone can take a question, without the need for extra configurations.

Keywords: Education; Code review; Software engineering.

Reviewbin: Uma ferramenta de fácil uso para revisão de código

José Renan Silva Luciano
Universidade Federal de Campina Grande
Campina Grande, Paraíba, Brasil
jose.luciano@ccc.ufcg.edu.br

Matheus Gaudencio do Rêgo
Universidade Federal de Campina Grande
Campina Grande, Paraíba, Brasil
matheusgr@computacao.ufcg.edu.br

RESUMO

Revisão de código é uma das etapas mais importantes do processo de desenvolvimento de software. A prática permite uma melhor qualidade e confiabilidade dos artefatos, pois os mesmos serão analisados e discutidos por outros desenvolvedores. Existem hoje ferramentas que possibilitam esse processo, como o GitHub e ReviewBoard, porém, estas ferramentas são integradas a um Sistema de Controle de Versão (SCV), como Git e *Subversion*, impossibilitando seu uso por pessoas que não têm o conhecimento dessas tecnologias, como alunos iniciantes em cursos de programação. Há também ferramentas, como o Codepost.io, que se apresentam como uma solução proprietária para revisão em salas de aula, porém que se limitam à correção de atividades e adicionam uma burocracia extra para a configuração de turmas e criação de atividades na plataforma. O objetivo deste trabalho é criar uma ferramenta com foco em revisão de código, que permita que alunos e professores aproveitem os benefícios desta prática em atividades de sala de aula, sem a necessidade de ter conhecimento de nenhuma tecnologia de SCV e também que seja livre para outros casos de uso, como o envio de um trecho de código para que alguém possa tirar uma dúvida, sem a necessidade de configurações extras.

PALAVRAS-CHAVE

Educação, revisão de código, engenharia de software.

REPOSITÓRIO

https://github.com/joserenan/reviewbin

1. INTRODUÇÃO

A revisão de código é um processo bem estabelecido na indústria [8] por ser uma forma de garantir que o que foi implementado é de qualidade e que todos ou parte do time tem ciência do que está sendo feito.[1] No domínio de código aberto, ela é de extrema importância para validar se modificações feitas por contribuidores serão adotadas ou não. A prática consiste em outros membros do time de desenvolvimento analisarem alterações feitas no código em busca de apontar possíveis problemas e discutir sugestões de melhoria.

Durante o processo de revisão, o desenvolvedor que faz as alterações solicita a outros desenvolvedores, chamados de revisores, que analisem a solução. Nessa análise, os revisores buscam por problemas como potenciais *bugs*, problemas relacionados ao estilo e ao *design* do código, trechos de código inutilizados e formas mais simples de realizar a mesma atividade. Os revisores geralmente têm conhecimento do contexto daquela alteração e do código já existente, para que o processo seja melhor aproveitado. A fim de apontar os problemas e discuti-los, o revisor adiciona comentários nas linhas do código que deseja discutir e por fim decide se a alteração será aprovada ou se necessita de modificações. No caso da requisição de modificações, o desenvolvedor sob revisão pode tentar argumentar para defender sua solução ou tentar implementar de uma outra forma e solicitar revisão novamente.

Na sala de aula, em cursos de programação, é muito comum que professores proponham atividades e projetos de *software* como forma de avaliação. A correção dessas atividades pode ser feita de inúmeras formas, por exemplo, com a execução de testes automatizados que certificam que o código funciona como deveria para aqueles cenários propostos, ou com a revisão de código, o que é bastante benéfico para os alunos, pois eles terão um *feedback* mais específico sobre a forma que a solução foi implementada.

Existem hoje ferramentas para revisão, mas que são dependentes de algum Sistema de Controle de Versão (SCV) [9] como Git, *Subversion*, *Mercurial*, para submissão e gerenciamento dos artefatos. Assim, é necessário ter conhecimento de tais ferramentas para que se possa fazer uso dos mecanismos de revisão de código. Essa dependência é um problema em cenários nos quais a utilização desse tipo de tecnologia é um fator limitante, como o de alunos iniciantes em cursos de programação que podem ainda não possuir conhecimento dessas tecnologias. Algumas ferramentas, como o Codepost.io[2], criam um ambiente de revisão de código para sala de aula. Entretanto, essas trazem consigo um ferramental que limita seu uso para apenas correção de exercícios de alunos e turmas configuradas na plataforma, permitindo o envio de código apenas como submissão da solução dessas atividades.

[2] https://codepost.io

Buscando solucionar o problema acima descrito, foi desenvolvida a ferramenta **Reviewbin**[3], uma aplicação *web* que permite a submissão de artefatos de código de forma simplificada pela sua interface gráfica. Além disso, ela facilita o compartilhamento desses artefatos através do envio de uma URL (*Uniform Resource Locator*) para que possam ser revisados, e tem uma interface de revisão similar às alternativas existentes, a fim de trazer uma experiência de revisão próxima às ferramentas utilizadas pela indústria, mas sem a burocracia do processo de submissão e compartilhamento do código, focando no que importa, a revisão. Dessa forma, o Reviewbin se apresenta como uma alternativa para professores e alunos que desejam utilizar a revisão de código, não apenas para a submissão e avaliação de atividades, como também para a resolução de possíveis dúvidas através do compartilhamento e comentários no código, seja com professores, monitores ou outros alunos.

2. REVIEWBIN

O Reviewbin é uma aplicação *web* que pode ser acessada diretamente pelo navegador, através do *link* disponibilizado no repositório do projeto. Logo ao abrir a página inicial, como é mostrado na Figura 1, o usuário terá a possibilidade de utilizar as funcionalidades de submissão de artefatos de código e também a de autenticação, que serão explicadas nas próximas subseções juntamente às demais funcionalidades.

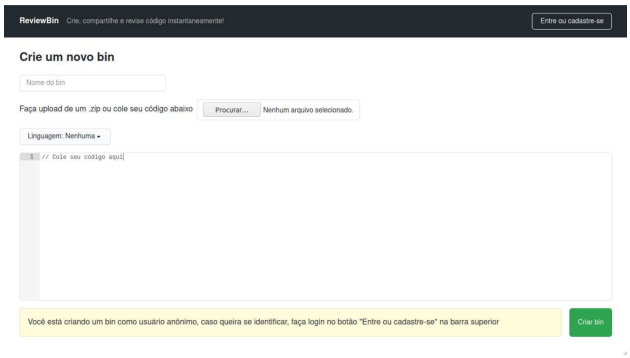

Figura 1 - Página inicial do Reviewbin

2.1 Submissão de Artefatos

A submissão de código a ser revisado no Reviewbin pode ser feita de duas maneiras: através do uso do editor de texto disponível na página inicial, ou realizando o *upload* de um arquivo *zip*. Após submetidos, os artefatos em processo de revisão são chamados de *bins*. Todos os *bins* possuem um nome ou descrição, um ou mais arquivos e um autor, que pode ser anônimo.

Para fazer a submissão de arquivos através do editor de texto, o usuário deve selecionar uma linguagem de programação e, em seguida, digitar o código no editor. Essa forma de submissão só suporta o *upload* de um arquivo por *bin*, sendo uma alternativa simples para pessoas que desejam compartilhar trechos de código, ou um algoritmo específico, que não precisa estar organizado em múltiplos arquivos.

Por sua vez, a submissão através do *upload* de arquivo *zip* é feita através do botão de envio de arquivos, que também consta na página inicial. Ao selecionar o arquivo a ser enviado, o editor de texto desaparece, indicando que não é possível realizar a submissão utilizando os dois métodos de envio ao mesmo tempo. O seletor de linguagens de programação também desaparece, pois será possível detectar a linguagem de programação utilizada através da extensão dos arquivos contidos no *zip*.

Sobre o arquivo *zip*, ele pode conter vários outros arquivos internamente, organizados em pastas na hierarquia que o usuário desejar, porém arquivos que não são textuais, como imagens, vídeos, arquivos binários ou arquivos que estão em diretórios ocultos são ignorados. Além disso, só é permitido o *upload* de um arquivo *zip* por *bin*. Essa é uma boa alternativa para a submissão de projetos que possuem uma estrutura de diretórios e múltiplos arquivos de código.

Ao submeter os artefatos, o usuário é redirecionado para a página do *bin*, na qual constará os arquivos submetidos e a possibilidade de iniciar a revisão. Além disso, será possível copiar a URL do *bin* para que possa ser compartilhada com outros usuários, que poderão acessar e realizar comentários, como será descrito nas próximas seções. A autoria do *bin* é dada ao usuário autenticado que fez a submissão do formulário de criação do mesmo e caso não haja usuário autenticado, é considerado que o autor é anônimo.

2.2 Autenticação

A autenticação é feita através do botão “Entre ou cadastre-se” disponível na barra superior da página inicial, que ao clicar, exibe uma janela de *pop-up* na qual o usuário deverá fazer *login* com sua conta Google. Após a realização do *login*, o usuário é retornado ao Reviewbin, e será possível notar que o botão “Entre ou cadastre-se” não estará mais visível, mas sim o nome do usuário e o botão “Sair”. O Reviewbin usará o nome, e-mail e foto do usuário, disponibilizada pela conta Google.

2.3 Página do Bin

Após a criação do *bin*, o usuário é redirecionado para a página do mesmo e recebe uma URL para acessá-lo. É possível, através do compartilhamento e consequentemente, o acesso dessa URL, que outras pessoas consigam ver o *bin*, revisá-lo e elaborar comentários.

Ao abrir a página do *bin* pela primeira vez, como mostrado na Figura 2, é exibida uma mensagem indicando que ainda não foi realizado nenhum comentário. Também são mostradas duas abas, uma chamada “Comentários” e outra “Arquivos”.

Figura 2 - Página inicial de um *bin* sem comentários

Através da aba “Comentários”, mostrada na Figura 3, é possível navegar pelos comentários realizados em ordem cronológica, do mais antigo para o mais recente, ou adicionar novos comentários,

[3] https://reviewbin.vercel.app/

funcionalidade que será descrita na próxima seção. Já na aba "Arquivos", é possível navegar pelos artefatos de código do *bin*, utilizando a barra de rolagem, ou a barra da lateral esquerda, que lista o nome de todos os arquivos do *bin* e permite a navegação direta para cada item através da ação de clique.

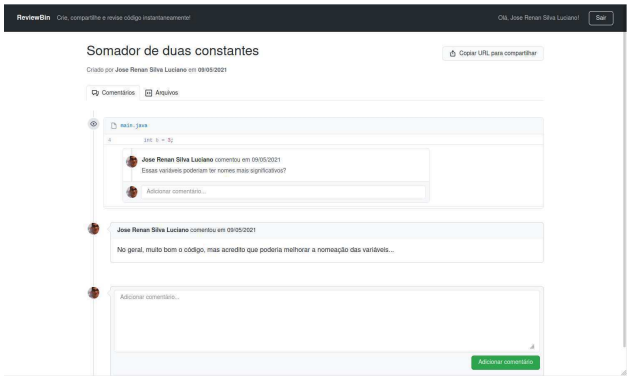

Figura 3 - Aba de comentários de um *bin*

Os artefatos mostrados na aba "Arquivos", mostrada na Figura 4, são exibidos com *syntax highlighting*, que é a formatação de cores e fontes do código de acordo com a sintaxe da linguagem de programação utilizada, para a melhor legibilidade do código durante o processo de revisão. Caso não seja possível detectar a linguagem de programação, ou se o código for escrito em uma linguagem não suportada, ele é exibido sem o *highlighting*.

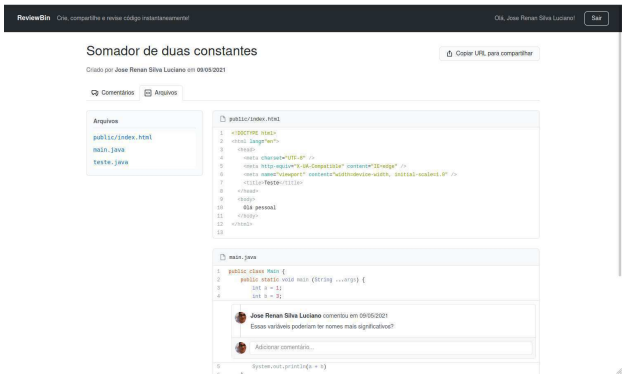

Figura 4 - Aba de arquivos de um *bin*

No caso do usuário não estar autenticado, ao acessar essa página, será mostrado um aviso indicando que todas as ações nela realizadas, como comentários e respostas, serão feitas anonimamente.

2.4 Revisão e Comentários

Os comentários podem ser realizados de duas formas, sendo elas: um comentário geral, sobre o *bin* como um todo; e os comentários no código, sobre algo especificamente relacionado a um trecho do código. Os comentários gerais são feitos no campo de texto localizado no fim da aba "Comentários". O usuário precisa digitar apenas o comentário que deseja fazer e submeter no botão de envio. Já os comentários no código são feitos na aba "Arquivos": ao passar o mouse sobre as linhas de código, um botão com um sinal de "+" aparecerá, indicando que pode ser adicionado um comentário naquela linha. Ao clicá-lo, um campo de texto ficará visível, e o usuário poderá inserir o comentário e submetê-lo através do botão de enviar.

Os comentários no código podem ser respondidos através dos campos de texto que surgirão abaixo do comentário após a submissão do mesmo, sendo assim iniciada uma discussão a respeito daquele trecho de código. Já os comentários gerais, por simplificação, podem ser respondidos com a adição de novos comentários na aba "Comentários".

Se o usuário estiver autenticado no momento da adição dos comentários, seu nome e foto serão exibidos ao lado do comentário, indicando que ele quem o fez. Caso contrário, será exibido que o comentário foi realizado anonimamente.

Todos os comentários, sejam gerais, no código ou respostas a comentários no código, são exibidos ordenados pelo momento da sua criação na aba "Comentários".

3. ARQUITETURA

O Reviewbin é implementado com auxílio do *framework* NextJS[4] utilizando a arquitetura *Jamstack* [3], que é a agregação das tecnologias Javascript, API e *Markup* em uma pilha para construção de aplicações *web* estáticas de alto desempenho, segurança e escalabilidade.

A lógica de *backend* foi feita utilizando a plataforma *Firebase*[5] e *serverless functions* [4], que servem de intermediário entre o cliente *web* e o *Firebase* para determinadas lógicas de negócio que serão melhores descritas nas próximas seções. O *Firebase* provê todos os recursos de *backend* necessários para a realização da autenticação de usuários, bem como bancos de dados para armazenamento dos metadados e alguns dados dos *bins*, como dados do autor, *link* para os arquivos, comentários, e também dispõe de um serviço de armazenamento de arquivos na nuvem. Na Figura 5 é possível ver como a arquitetura está organizada e como os componentes se comunicam.

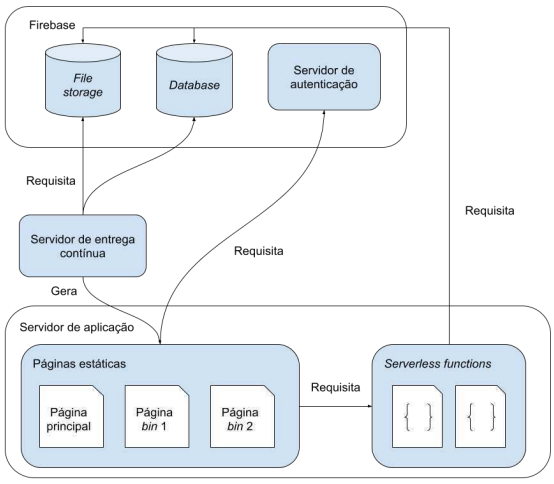

Figura 5 - Arquitetura do Reviewbin

[4] https://nextjs.org/
[5] https://firebase.google.com/

3.1 Aplicação Web

Como citado anteriormente, o *frontend* do Reviewbin foi implementado com o *framework* NextJS, que utiliza *React*[6] como biblioteca para a criação da interface de usuário, através de componentes implementados com uma linguagem que estende o *Javascript,* chamada de JSX [1].

Para as páginas, foi utilizada a funcionalidade de *Static Site Generation* (SSG) [2] do NextJS, que permite que todas as páginas de *bin* sejam pré-renderizadas no servidor de entrega contínua no momento da compilação para que seja feita a implantação. Em seguida, as páginas são guardadas em *cache* para que os próximos usuários a acessá-las não precisem solicitar que sejam renderizadas novamente.

3.1.1 Geração das Páginas

Durante a pré-renderização das páginas, é feita uma requisição de todos os *bins* salvos no *Firebase*. Todos os dados e arquivos dos *bins* são baixados, e as páginas HTML são geradas a partir do envio desses dados como parâmetro para os componentes *React*. Dessa forma, a responsabilidade da geração do HTML a partir da interpretação do código *Javascript* é tirada do cliente *web* e levada ao servidor de entrega contínua. Isso é algo benéfico para a página de *bins*, pois ela pode conter muitos arquivos a serem exibidos, o que é passível de lentidão. Sendo assim, em vez do usuário aguardar o *download* dos arquivos e comentários a serem exibidos e, além disso, a renderização dos mesmos pelo *React*, o usuário só terá de esperar o tempo de carregamento do HTML que já estará renderizado.

Para *bins* criados após a aplicação já ter sido implantada, ou seja, após a fase de pré-renderização, a geração acontecerá também no servidor, mas apenas no momento em que o primeiro usuário requisitar aquela página. Da mesma forma que as geradas no momento do *deploy*, essa nova página será guardada em um *cache* no servidor, a fim de que os próximos usuários não precisem aguardar que sejam geradas novamente.

Todas as requisições realizadas para a geração das páginas são feitas através de *Javascript* e diretamente da máquina que está compilando o código para que seja implantado. As requisições são enviadas diretamente ao *Firebase*, sem intermédio das *serverless functions*. No entanto, apesar das páginas serem pré-renderizadas estaticamente, existem também algumas requisições que são feitas no cliente, como a submissão de formulários para criação de *bins* e para adição de comentários, que modificam dinamicamente o estado da página. Essas requisições são chamadas via comunicação HTTP (*Hypertext Transfer Protocol*) pelo *Javascript* que está sendo executado no navegador para as *serverless functions* que, por sua vez, realizam a comunicação com o *Firebase*.

3.1.2 Interface de Usuário e Estilização

A interface de usuário do Reviewbin foi desenvolvida utilizando os componentes da biblioteca de código aberto *Primer*[7], que é desenvolvida e utilizada pelo time do GitHub. A escolha da biblioteca foi motivada por possuir componentes de interface desenhados para o contexto de repositórios de código e revisão, que se encaixam com o propósito do Reviewbin. Além disso, trazer semelhança visual com a plataforma do GitHub pode ser benéfico para fins de reaproveitamento do conhecimento tanto para usuários do GitHub que venham a usar o Reviewbin, bem como, para usuários que estarão conhecendo o processo de revisão através do Reviewbin e, em algum momento em sua carreira, usarão o GitHub para algum projeto.

A estilização dos componentes é feita através da biblioteca *styled-components*[8], que permite ao desenvolvedor a criação de componentes estilizados com sintaxe CSS (*Cascading Style Sheets)* dentro do código *Javascript*, ou seja, onde o componente é definido. Isso aproxima a estilização da lógica de negócio, permitindo a criação de componentes com *design* mais dinâmico e responsivo às interações do usuário.

Para o componente de editor de texto, foi utilizado o *Ace Editor*[9] com os temas CSS já presentes na biblioteca, sem nenhuma customização para realização do *syntax highlighting*. No entanto, para a exibição de arquivos na página de revisão foi criado um componente próprio, utilizando o *add-on runmode* do editor de texto *CodeMirror*[10]. A implementação deste componente será melhor discutida na seção 5.2.

3.2 Serverless Functions

A fim de abstrair a lógica de armazenamento de dados e arquivos do *frontend*, o Reviewbin usa *serverless functions*, que são disponibilizadas como *endpoints* HTTP seguindo o padrão REST (*Representational State Transfer*) [6]. Ao fazer uma requisição para criação de um novo *bin*, caso seja a partir do envio de um arquivo *zip*, será chamado um *endpoint* que ficará responsável por fazer os tratamentos desse arquivo, como a descompressão, remoção de arquivos que devem ser ignorados, e detecção da linguagem de programação do arquivo. Em seguida enviará para o *Firebase*, para que o *upload* seja feito e as demais informações do *bin* sejam salvas. A mesma *serverless function* é chamada em caso de criação de *bins* pelo editor de texto, mas nesse caso, a função fará a transformação do texto em arquivo, em vez de fazer o tratamento do *zip*.

Além da lógica de criação dos *bins*, também foi implementada uma função responsável pela recuperação e transformação dos dados de comentários em uma estrutura que simplifica a leitura pelo *frontend*.

A escolha por *serverless functions* se deu pela simplicidade para manutenção e o baixo custo de implantação, que são características desejáveis para a aplicação, dada a baixa complexidade e pontualidade das lógicas de *backend* que seriam executadas.

4. TESTES DE USABILIDADE

A fim de testar a qualidade do sistema quanto à sua usabilidade e à satisfação dos usuários, foi feito um levantamento utilizando o *Computer System Usability Questionnaire* (CSUQ) [7], no qual os usuários respondem a 19 afirmações de boa usabilidade do sistema com um valor de 1 a 7. Nesse caso, 1 significa completa discordância e 7 significa completa concordância com a afirmação. A média aritmética dos valores das respostas de cada pergunta indica o quão boa é a usabilidade do sistema quanto

[6] https://reactjs.org/
[7] https://primer.style/
[8] https://styled-components.com/
[9] https://ace.c9.io/
[10] https://codemirror.net/

àquele aspecto, sendo considerado um resultado melhor quanto mais o valor se aproxima de 7.

O teste foi executado com 14 estudantes da disciplina de Programação II do curso de Ciência da Computação da Universidade Federal de Campina Grande, dentre os quais quatro eram monitores. Foi solicitado que cada um dos 10 alunos não monitores fizessem, de forma não guiada, o *upload* de uma das atividades da disciplina, e que o *link* do *bin* fosse compartilhado com algum dos 4 monitores. Dessa forma, estes deveriam revisar o código, comentando sobre a sua qualidade, formas melhores de implementar e também dar um parecer geral sobre o que achou do código. Os alunos podiam argumentar pela plataforma respondendo aos comentários e, após a utilização, foi solicitado que respondessem ao questionário citado anteriormente.

Como é possível ver na Figura 6, os resultados apontaram bons indicadores, tendo a maioria dos itens avaliados uma média de usabilidade maior que 5, com exceção do item 9. Esse item é referente à exibição de mensagens de erro que indicam como corrigir ou solucionar o problema. Alguns usuários relataram, na seção de comentários da pergunta no questionário, não ter recebido mensagens de erro ao realizar *upload* de arquivos que não estavam no formato *zip* e, em vez disso, o sistema continuava carregando indefinidamente sem exibir nenhum retorno visual, sendo este o principal motivo do baixo indicador.

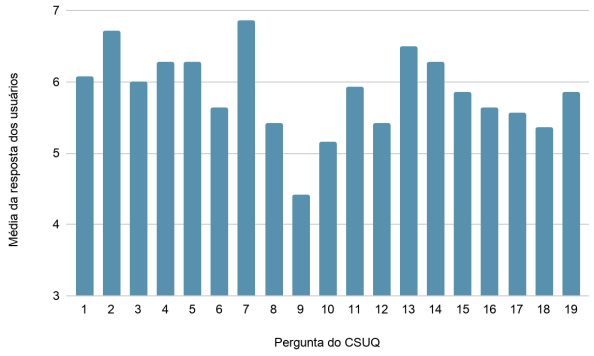

Figura 6 - Resultados do teste por pergunta do CSUQ

Outros problemas apontados no teste foram quanto à aba de comentários. Todos os monitores comentaram sobre a largura da visualização de código não ser suficiente para exibir o código como um todo, sendo necessário utilizar a barra de rolagem horizontal, que é de difícil acesso para arquivos com muitas linhas. Também foi comentado que a barra lateral que lista os arquivos do *bin* poderia estar sempre visível, de forma que o usuário não precise voltar ao topo sempre que desejar navegar diretamente para outro arquivo, aumentando a produtividade durante a correção.

Além dos problemas descritos, também foram apontadas algumas sugestões para próximas versões, como a adição de alguma área onde seja possível listar todos os *bins* criados pelo usuário, a adição da funcionalidade de edição e remoção de comentários e também a notificação de comentários por *e-mail*.

Os pontos positivos do sistema listados pelos usuários foram a simplicidade e objetividade do uso, a facilidade do compartilhamento de código e a facilidade para submissão de arquivos e projetos. Esses aspectos positivos justificam a média mais alta dentre as perguntas do questionário, a pergunta 7, que faz referência à facilidade em aprender a utilizar o sistema, mostrando que o Reviewbin cumpre com a simplicidade de uso inicialmente proposta neste trabalho.

5. EXPERIÊNCIA

Nesta seção, é discutido sobre como foi realizado o processo de desenvolvimento da plataforma, bem como os principais desafios encontrados ao longo do processo, detalhando sobre a implementação do componente de exibição e revisão de código, citado na seção 3.1.2.

5.1 Processo de Desenvolvimento

O desenvolvimento do Reviewbin se baseou no método ágil *Scrum* [5], consistindo na definição de uma lista de requisitos que devem ser divididos em listas menores, chamadas de *backlog*, a serem desenvolvidas em ciclos, chamados de *sprint*, e a realização de encontros para acompanhamento ao fim de cada *sprint*. A definição e priorização desses requisitos é feita pelo *Product Owner*, e os *backlogs* de cada *sprint* são definidos pelo *Scrum Master*. Foi decidido não utilizar outros rituais e papéis do *scrum* já que o time contava apenas com duas pessoas, o que tornaria o processo burocrático e inviável.

No Reviewbin, a definição de requisitos a serem implementados, bem como a priorização e definição dos *backlogs*, foi realizada em conjunto com o orientador deste trabalho, com quem eram realizadas reuniões de acompanhamento. A implementação do projeto se deu ao longo dos meses de janeiro a março de 2021 e foi dividida em três *sprints* de um mês cada.

As reuniões de acompanhamento foram de extrema importância para o processo de desenvolvimento, pois foi possível encontrar eventuais problemas e definir ajustes mais rapidamente, bem como avaliar se o que foi implementado estava de acordo com os critérios de aceitação da funcionalidade.

5.2 Principais Desafios

O foco do Reviewbin é a revisão, sendo assim, é esperado que a experiência do usuário ao realizar comentários no código seja a melhor possível. A fim de tornar isso possível, a plataforma usa como inspiração a interface de outras ferramentas consolidadas na indústria, como o GitHub e o GitLab, que exibem na sua interface o campo de comentários dentro do código, entre a linha de código a ser comentada e a linha seguinte.

Para replicar esta funcionalidade, seria necessário um componente *React* que permitisse a renderização de artefatos de código com *syntax highlighting,* mas também permitisse que o desenvolvedor customizasse o componente que envolve cada linha de código, a fim de que fosse possível, por exemplo, mostrar comentários entre elas.

Analisando as bibliotecas existentes de *syntax highlighting* para *React*, notou-se que as poucas que permitiam a customização do componente de linha de código não estavam mais sendo mantidas por seus times de desenvolvimento e possuíam documentações muito escassas.

A fim de solucionar o problema acima, foi implementado no Reviewbin um componente próprio para exibição e revisão de código com a ajuda do *CodeMirror*, um editor de texto feito em *Javascript* para navegadores *web*. Esse componente possui um

add-on chamado de *runmode*, que permite utilizar a funcionalidade de *syntax highlighting* fora do editor. O *runmode*, ao ser executado, quebra o código em uma lista de lexemas, percorre essa lista e executa, para cada iteração, uma função. Essa função é definida pelo desenvolvedor e recebe por parâmetro o lexema da iteração atual e seu respectivo estilo CSS. Através dela é possível construir um componente customizável para cada linha de código, que é composta pelo conjunto de lexemas com estilização CSS finalizado com um caractere de fim de linha. Possibilitando assim, a adição de comentários entre as linhas, indicadores de número da linha personalizados dentre outras customizações.

Outro problema encontrado ao longo do desenvolvimento foi a notória lentidão no carregamento da página do *bin*, dado o fato de que inicialmente o cliente fazia o *download* e executava a lógica para estilização de todos os arquivos diretamente no navegador. Esse problema foi resolvido, como citado anteriormente neste trabalho, com a pré-renderização dessas páginas no servidor através da utilização de SSG.

6. TRABALHOS FUTUROS

O Reviewbin busca ter como principal vantagem sobre as alternativas existentes a simplicidade para submissão e compartilhamento de código, a fim de que o foco seja na revisão. No entanto, existem algumas funcionalidades de revisão que estão presentes nessas alternativas, e que podem entregar uma melhor experiência de usuário em próximas versões do Reviewbin. Uma delas é o suporte a formatação de texto nos comentários, algo que é bem comum em outras ferramentas, seja com linguagens específicas para esta finalidade, como o *Markdown*, ou com um editor *rich text*. Outra funcionalidade relacionada é a de reações aos comentários com emojis, o que pode poupar tempo dos usuários ao responder ou endossar algumas revisões sem a necessidade de digitar.

É essencial, em trabalhos futuros, que sejam levados em consideração os comentários de problemas e sugestões apontados no teste de usabilidade, como os ajustes relacionados à melhor visualização de código e melhor navegação entre os arquivos, a fim de conseguir prover uma melhor experiência para os revisores.

Ainda sobre os testes de usabilidade, a realização de testes comparativos podem ser benéficos em trabalhos futuros, a fim de entender a performance do usuário na utilização da ferramenta em comparação com outras alternativas existentes, tendo em vista que o teste realizado apenas avalia a ferramenta por si só. Ademais, sugere-se o teste do uso da ferramenta em outros contextos além da sala de aula.

Um outro ponto relevante é a disponibilização de uma API para que outras plataformas possam criar integrações e, por exemplo, submeter *bins* para revisão no Reviewbin, o que pode abranger ainda mais os casos de uso da ferramenta e permitir que outros desenvolvedores criem uma experiência customizada de submissão e utilizem apenas as funcionalidades de compartilhamento e revisão.

7. AGRADECIMENTOS

Agradeço primeiramente a Deus por sempre me guiar em sua infinita majestade. Gratidão à minha família pelo apoio e por acreditarem que este trabalho fosse possível. Ao meu orientador Matheus Gaudencio, pela paciência e dedicação de seu tempo no acompanhamento deste trabalho. E aos demais professores do curso que sempre serviram de exemplo e inspiração para mim.

Gostaria de agradecer também aos meus amigos que estiveram sempre ao meu lado durante toda a graduação, com quem tive a oportunidade de compartilhar experiências de crescimento pessoal e profissional e também à comunidade OpenDevUFCG que me abriu muitas portas ao longo do curso.

8. REFERÊNCIAS

[1] [n. d.]. Introducing JSX. Retrieved May 13, 2021 from https://reactjs.org/docs/introducing-jsx.html

[2] [n. d.]. Static Site Generator. Retrieved May 13, 2021 from https://www.gatsbyjs.com/docs/glossary/static-site-generator/

[3] [n. d.]. What is Jamstack? Retrieved May 13, 2021 from https://jamstack.org/what-is-jamstack/

[4] [n. d.]. What is serverless? Retrieved May 13, 2021 from https://www.cloudflare.com/pt-br/learning/serverless/what-is-serverless/

[5] [n. d.]. What is Scrum? Retrieved May 13, 2021 from https://www.scrum.org/resources/what-is-scrum

[6] Feng, X., Shen, J., & Fan, Y. 2009. REST: An alternative to RPC for Web services architecture. In *2009 First International Conference on Future Information Networks* (pp. 7-10). IEEE.

[7] Lewis, J. R. 1995. IBM computer usability satisfaction questionnaires: psychometric evaluation and instructions for use. *International Journal of Human‐Computer Interaction*, *7*(1), 57-78.

[8] McIntosh, S., Kamei, Y., Adams, B., & Hassan, A. E. 2014. The impact of code review coverage and code review participation on software quality: A case study of the qt, vtk, and itk projects. In *Proceedings of the 11th Working Conference on Mining Software Repositories* (pp. 192-201).

[9] Spinellis, D. 2005. Version control systems. *IEEE Software*, *22*(5), 108-109.