INSTITUTO FEDERAL DO PIAUÍ – IFPI
PRÓ-REITORIA DE ENSINO
DIRETORIA DE ENSINO
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

RELATÓRIO TÉCNICO DE SOFTWARE

Code Class. Uma Ferramenta para Avaliação de Código Fonte em Ambiente Educacional

LUCAS ALVES DE LIMA

Rafael Torres Anchiêta
ORIENTADOR

PICOS, PI
2024

LUCAS ALVES DE LIMA

CODE CLASS. UMA FERRAMENTA PARA AVALIAÇÃO DE CÓDIGO FONTE EM AMBIENTE EDUCACIONAL

Relatório Técnico de Software apresentado como exigência parcial para obtenção do diploma do Curso de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia do Piauí, Campus Picos.

Orientador: Prof. Dr. Rafael Torres Anchiêta

PICOS
2024

RESUMO

Este trabalho apresenta o desenvolvimento de um sistema chamado Code Class cujo objetivo é avaliar códigos fontes. Avaliar códigos fonte é uma tarefa desafiadora e laboriosa, principalmente em ambiente educacional, onde o professor deve avaliar de maneira minuciosa diversos códigos fontes de alunos. O Code Class foi desenvolvido com o propósito de auxiliar o professor nessa atividade. O sistema ainda encontra-se em modo de desenvolvimento, mas está publicamente disponível no GitHub https://github.com/Lukas00-gif/Hi-Lo.

Palavras-chave: Educação; Programação; Código Fonte.

SUMÁRIO

1. INTRODUÇÃO

Analisar código-fonte é uma atividade intrinsecamente desafiadora, exigindo do professor uma minuciosa avaliação de cada linha, e uma cuidadosa revisão e identificação de possíveis problemas.

Nesse contexto, surge a necessidade de ferramentas eficientes que facilitem e aprimorem esse processo. Este trabalho apresenta uma aplicação dedicada à avaliação de códigos-fonte usando a linguagem de programação Python, desenvolvido na framework Vue 3, Bootstrap, o banco de dados firebase e flask a micro web framework em Python. Projetada para otimizar o tempo e esforço do professor, a ferramenta oferece funcionalidades que incluem: Criar sala de aula, editar a sala de aula, desativar a sala de aula, Criar as atividades e fazer a comparação de código-fonte.

No cenário específico da programação competitiva, em que a prática intensiva de resolução de problemas é crucial para o desenvolvimento de habilidades algorítmicas e de codificação, destaca-se a importância de plataformas especializadas. O BeeCrowd[1], por exemplo, é reconhecido por sua abordagem centrada na programação competitiva. Ele oferece um ambiente desafiador com uma ampla gama de problemas, direcionando seus usuários para aprimorar suas habilidades em competições e desafios de algoritmos.

Em contraste, o Code Class assume uma abordagem mais voltada para a avaliação de códigos-fonte no contexto educacional tradicional facilitando a comunicação entre aluno e professor. Enquanto o BeeCrowd se concentra em desafios intensos e competitivos. Ambas as plataformas desempenham papeis distintos no ecossistema educacional.

1.1. OBJETIVOS

1.1.1. Geral

O objetivo geral deste trabalho é desenvolver uma ferramenta para auxiliar os professores na avaliação de código fonte.

1.1.2. Específicos

Para alcançar o objetivo geral do trabalho, definiu-se os seguintes objetivos específicos:

[1] https://www.beecrowd.com.br/judge/en/login

- Uma ferramenta Web que auxilie o professor na avaliação de códigos fonte;
- Um método para verificação da saída esperada do código do professor com o resultado da saída do código do aluno.

2. TECNOLOGIAS ENVOLVIDAS

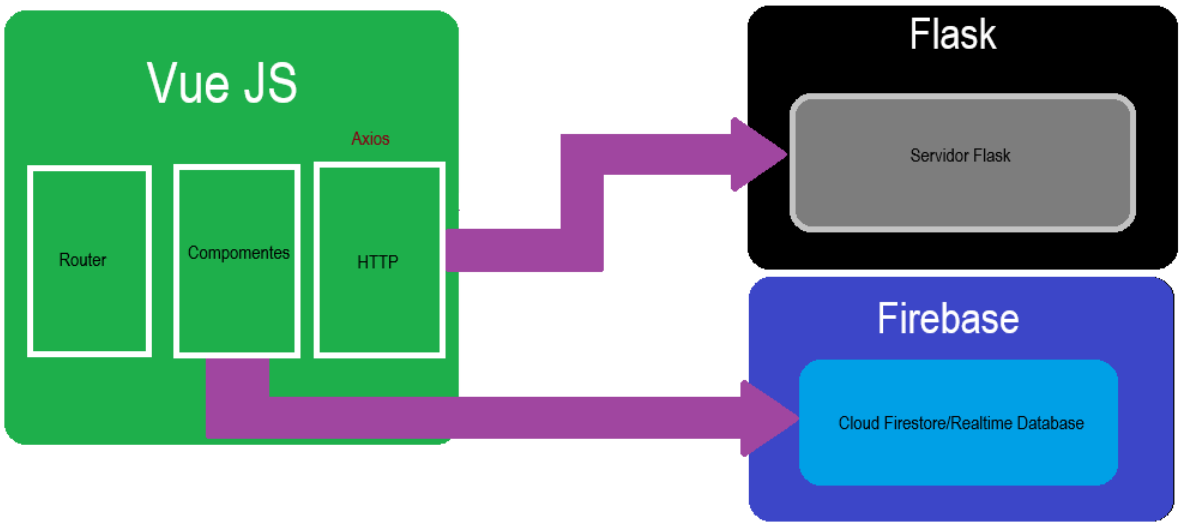
Todo o desenvolvimento foi realizado empregando HTML, CSS e JavaScript, fazendo uso da framework Vue JS versão 3 e Bootstrap como *front-end* e integrando o Firebase como banco de dados para *back-end*. O ambiente de codificação escolhido foi o Visual Studio Code[2], reconhecido por sua leveza, potência e capacidade de viabilizar uma codificação eficiente e robusta.

O servidor responsável por receber as requisições da aplicação principal foi feito em Flask, usando Python. Para o controle de versionamento do código, optou-se pelo Git e pelo GitHub. O Git, uma ferramenta de código aberto, é utilizada gratuitamente para gerenciar versões de projetos, enquanto o GitHub é uma plataforma de hospedagem de código-fonte e arquivos, proporcionando um controle de versão eficaz através do Git.

A aplicação em Vue JS usa o banco de dados firebase, o Cloud Firestore e o Realtime Database para o armazenamento de dados em geral da aplicação, conforme apresentado na Figura 1.

[2] https://code.visualstudio.com/

Figura 1: Fluxo da Aplicação.



Fonte: Elaborado Pelo Autor.

3. MODELAGEM DO PROJETO

3.1. LEVANTAMENTO DE REQUISITOS

A fase de Levantamento de requisitos é o estágio de construção do software encarregado de identificar as exigências a serem atendidas pelo sistema. Conforme mencionado por Paula Filho (2001), a engenharia de requisitos compreende um conjunto de métodos utilizados para identificar, elaborar, documentar e validar os requisitos de um software.

Os requisitos são definidos em:

- Requisitos funcionais:

Os Requisitos Funcionais descrevem as funcionalidades ou os serviços que se espera que o sistema realize em benefício dos usuários (PAULA FILHO 2000). Para o software desenvolvido foram definidos os seguintes requisitos funcionais de acordo com a Tabela 1.

Tabela 1: Requisitos Funcionais do Code Class.

Código	Requisito	Descrição
RF1	Cadastro de Professor/Aluno	Como Professor/Aluno gostaria de me cadastrar na plataforma e acessá-lá

RF2	Login de Professor/Aluno	Como Professor/Aluno gostaria de efetuar o login na plataforma para acessar a mesma
RF3	Ver as atividades	Como Professor/Aluno gostaria de ver as atividades disponíveis no mural de atividades
RF4	Ver as Pessoas na Sala	Como Professor/Aluno gostaria de ver as pessoas que estão na sala de aula
RF5	Ver o Perfil	Como Professor/Aluno gostaria de ver o meu perfil.
RF6	Editar Perfil	Como Professor/Aluno gostaria de editar o meu perfil.
RF7	Criar Sala de Aula	Como Professor gostaria de criar a sala de aula
RF8	Editar as Informações da Sala	Como Professor gostaria de editar as informações da sala de aula
RF9	Desativar Sala de Aula	Como professor gostaria de desativar a sala de aula
RF10	Criar a Atividade	Como professor gostaria de criar a atividade para os alunos responderem.
RF11	Sair da Sala de Aula	Como Aluno gostaria de sair da sala de aula.
RF12	Responder às atividades	Como Aluno gostaria de responder às atividades que estão no mural.

RF13	Comparação de Código	Como professor gostaria de comparar a saída do meu código com a saída do código do aluno.

Fonte: Elaborado pelo Autor.

- Requisitos não-funcionais:

  Os Requisitos Não-Funcionais são aqueles que não dizem respeito diretamente às funcionalidades providas pelo sistema. Podem está relacionados a propriedades de sistemas emergentes como, disponibilidade, requisições, desempenho e outros atributos de qualidade do produto (PAULA FILHO, 2000). Para o software desenvolvido foram definidos os seguintes requisitos não-funcionais de acordo com a Tabela 2.

Tabela 2: Requisitos Não-Funcionais do Code Class.

Código	**Requisito**	**Descrição**
RNF1	Disponibilidade	A aplicação deve permanecer 24h por dia 7 dias por semana
RNF2	Tecnologias	A aplicação será feita em Vue 3 no front end o banco de dados é o firebase e o servidor é feito em python
RNF 3	Requisição	O servidor terá que dar a resposta para as requisições no tempo máximo de 5 segundos

RNF4	Autenticação	A aplicação só será acessada para aqueles que estejam autenticados pelo próprio sistema
RNF5	Usabilidade	O usuário deverá realizar as atividades no sistema
RNF6	Confiabilidade	Os dados não podem ser perdidos no momento do cadastro e durante a aplicação

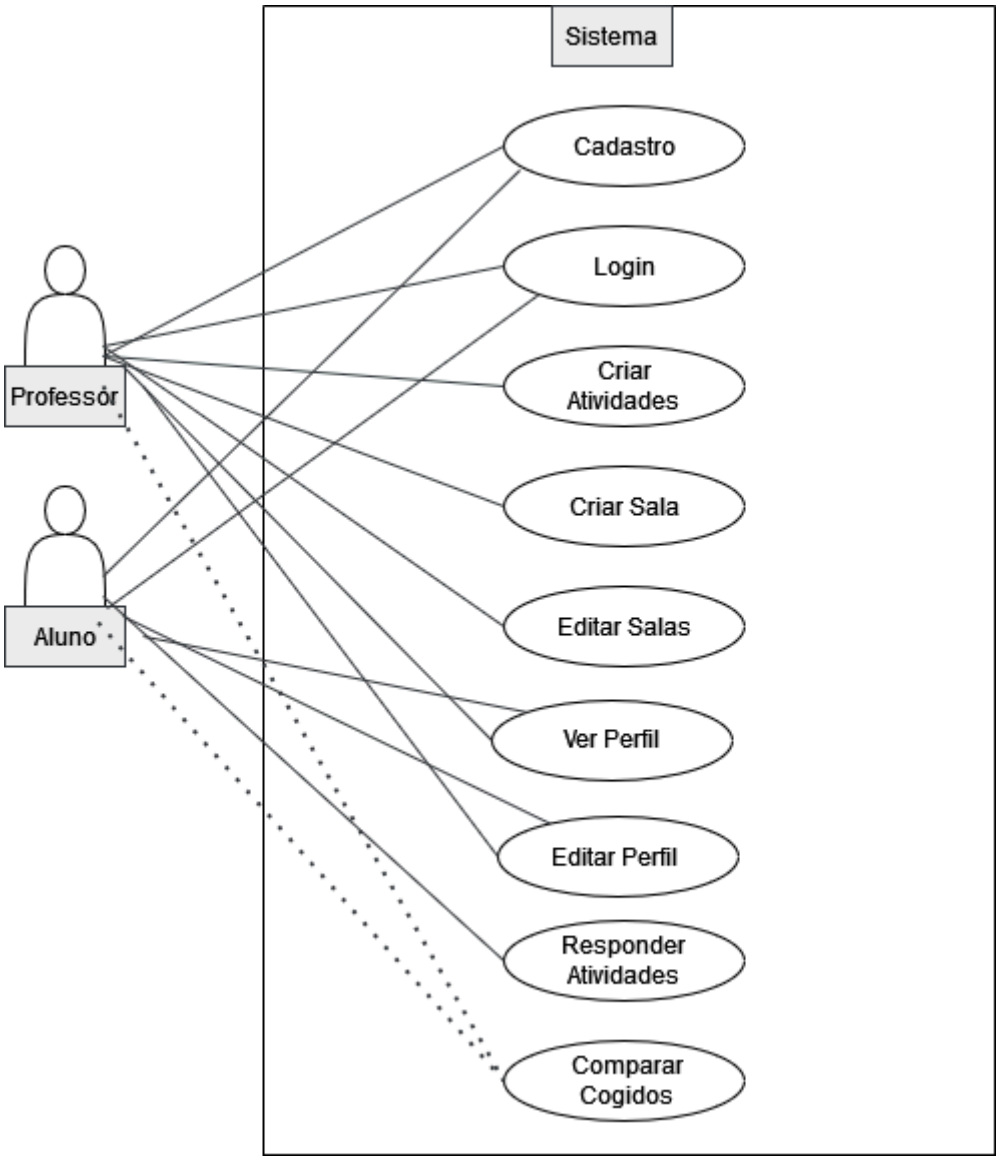
Fonte: Elaborado pelo Autor.

3.2. DIAGRAMAS DE CASOS DE USO

O diagrama de caso de uso é um modelo gráfico de mais fácil entendimento na documentação de um projeto, descreve uma função que o sistema desempenha para alcançar a meta do usuário, deve produzir um resultado observável que seja valioso para o usuário do sistema. Nele, são observados as execuções que cada ator vai desempenhar. A Figura 2 demonstra o diagrama de caso de uso da aplicação.

A Figura 2 é o diagrama de caso de uso do sistema atual. O Professor pode se cadastrar, fazer o login, criar as salas de aulas, criar as atividades, editar as salas de aulas. O Aluno pode se cadastrar, fazer o login no sistema e responder as atividades. Tanto o professor quanto o aluno podem enviar os códigos-fontes para o sistema e o mesmo irá comparar os códigos.

Figura 2: Diagrama de Caso de Uso.



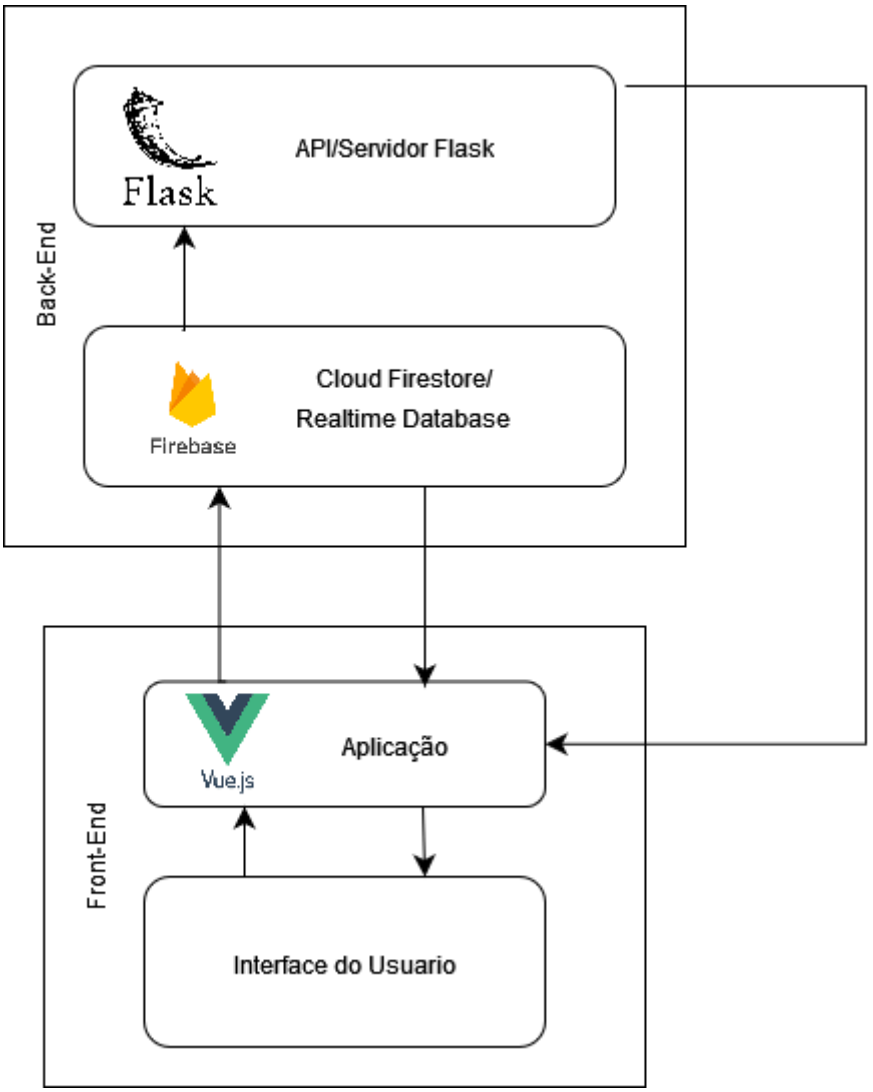
Fonte: Elaborado Pelo Autor.

3.3. ARQUITETURA DO SISTEMA

A arquitetura consiste de um modelo de alto nível que possibilita um entendimento e uma análise mais fácil do software a ser desenvolvido. Uma boa arquitetura é essencial para a organização da composição do sistema.

O uso de arquitetura para representar soluções de software foi incentivada e principalmente por (GARLAN e PERRY, 1995; KAZMAN, 2001).

Figura 3: Arquitetura do Sistema.



Fonte: Elaborado Pelo Autor.

O front-end é elaborado em Vue, tudo armazenado no firebase, usando o Cloud Firestore e o Realtime Database. O servidor vai receber a saída do código professor que é colocado pelo o mesmo e o código do aluno que estão armazenados no firebase e assim com o resultado ele é retornado na aplicação onde o usuário pode ver o resultado, conforme detalhado na Figura 3.

3.4. INTERFACE

A interface gráfica do Code Class é baseada em um design mais minimalista onde tem mínimos elementos na tela.

Quando o usuário professor faz o login, na tela de *home* (Figura 4), tem botões de desativar as salas de aulas, editar as salas, criar as salas que são exclusivos do professor, onde o aluno não tem acesso a elas. A notificação aparece sempre que o professor desativa uma sala.

Dentro da sala de aula, a tela para o professor tem um botão de criar atividades que é responsável por criar as atividades para os alunos (Figura 5). Na Figura 5, existem 3 campos. O primeiro campo solicita que o professor insira um título para a atividade. No segundo campo, o professor descreve a atividade a ser realizada. No terceiro campo, o professor informa a saída esperada para a atividade proposta. Essa saída será comparada com a saída do código do aluno.

Quando o aluno faz o login, ele tem uma visualização de acordo com a Figura 6. Na tela, existem as salas disponíveis que o aluno está matriculado, ele pode entrar na sala ou pode sair da sala quando quiser. O aluno pode entrar em uma nova sala a partir de um código de acesso disponibilizado pelo professor através do botão “Adicionar Código”, e também o mesmo pode acessar livremente o perfil. O perfil é onde o usuário pode mudar o seu email, o seu nome e o seu sobrenome e atualizar o mesmo clicando no botão “Atualizar Perfil”. Quando a sala está desativada ainda na figura 6 é mostrada a mensagem em vermelho e os botões da sala estão desativados, não podendo clicar.

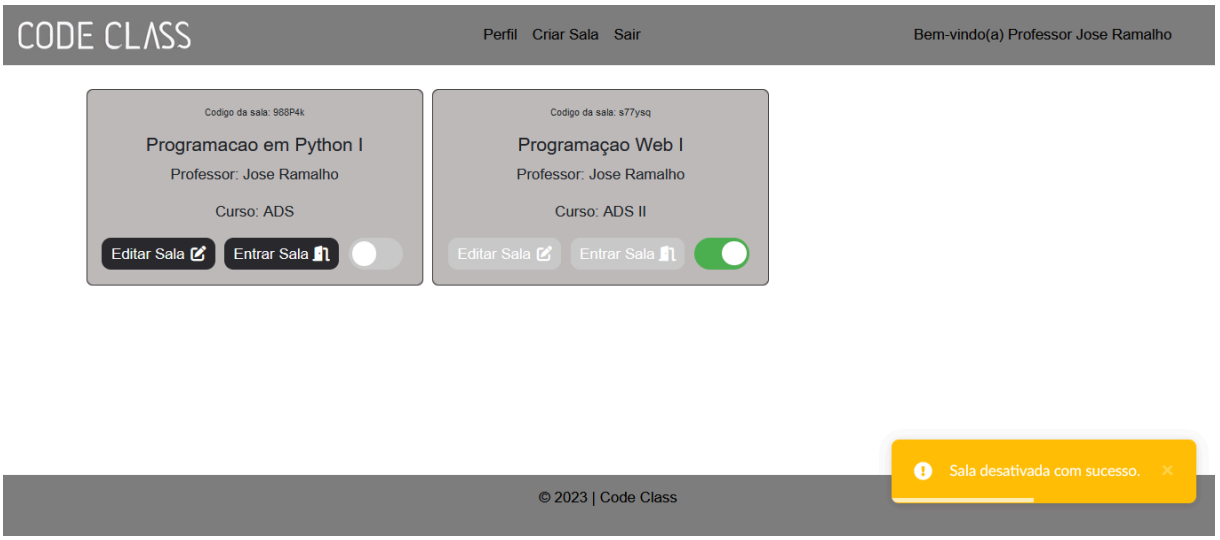
Quando o aluno entra na sala de aula, é exibido uma tela de acordo com a Figura 7. Ele pode ver as pessoas que estão na sala através do botão “pessoas na sala” e pode ver as atividades que estão disponíveis para ele responder, através do botão “Abrir Atividade” o aluno pode abri-la e responder a atividade.

Quando o aluno abre a atividade para responder a tela fica conforme a Figura 8, existem dois campos onde o primeiro campo o aluno deve colocar um título para a resposta da atividade e o segundo campo deve ser o código em Python para a atividade e somente, depois desses dois campos estarem devidamente preenchidos o botão de “Enviar Resposta” estará disponível para o aluno clicar. Quando o aluno clicar no botão “Enviar Resposta” a ferramenta vai mandar o codigo do aluno para o banco de dados e através de uma função do código ela vai ser resgatada e vai ser mandada para o servidor onde o código do aluno vai ser rodado em um arquivo

separado que vai ser criado temporariamente e a saída do código do aluno vai ser comparada com a saída do codigo do professor.

A Figura 9 é quando o aluno respondeu a atividade e o sistema retornou se ele acertou ou ele errou, caso a saída do código do aluno seja igual a saída do professor a mensagem que vai aparecer é “Você Acertou” ou se o aluno errou “Você ERROU” irá aparecer. A notificação aparece quando o aluno tenta responder novamente a mesma atividade, onde isso não é permitido.

Figura 4: Interface Gráfica: Tela de Home do Professor.

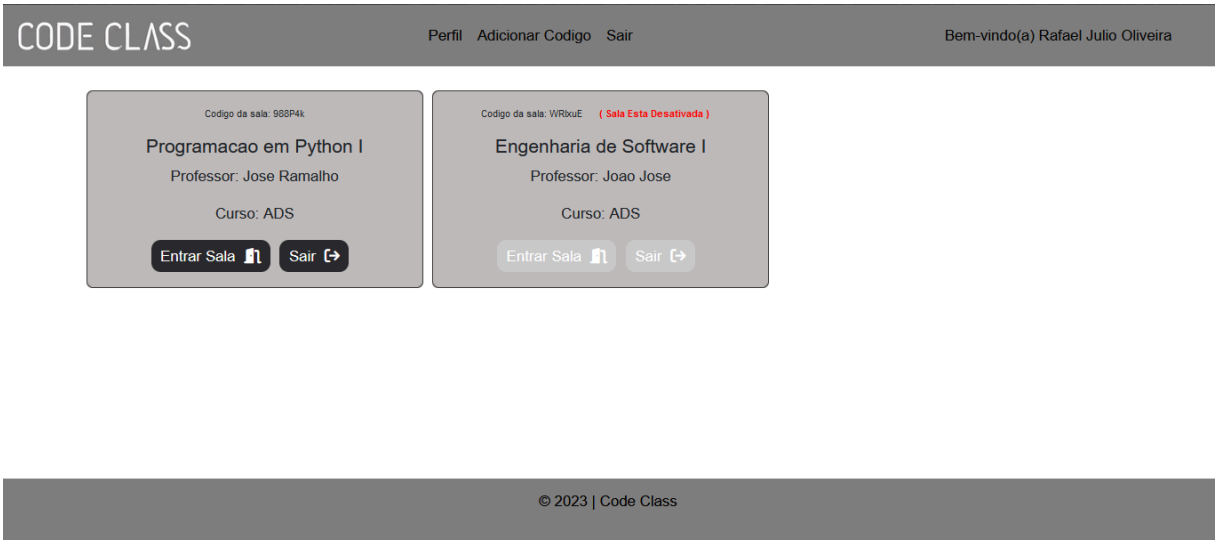


Elaborado Pelo Autor

Figura 5: Interface Gráfica 2: Tela de Criar Atividade.

Elaborado Pelo Autor.

Figura 6: Interface gráfica 3: Tela Home Aluno.



Elaborado Pelo Autor

Figura 7: interface gráfica 4: Dentro da Sala com a Atividade.

Elaborado Pelo Autor

Figura 8: Interface gráfica 5: Aluno Respondendo a Atividade.

Elaborado Pelo Autor

Figuras 9: Interface gráfica 6: Aluno Respondendo Novamente a Mesma Atividade.

Elaborado Pelo Autor

4. SOFTWARE

4.1. IMPLANTAÇÃO

É a fase do ciclo de vida de um software, que corresponde à passagem do software para a produção. O Code Class ainda está em modo desenvolvimento. Ele está publicamente disponível no link https://github.com/Lukas00-gif/Hi-Lo.

4.2. TESTES

O teste de software é a fase do desenvolvimento do software a fim de fornecer informações sobre sua qualidade em relação ao contexto em que ele deve operar, se relaciona com o conceito de verificação e validação. Isso inclui o processo de utilizar o produto para encontrar seus defeitos.

1 - Teste de Unidade: São feitos no nível baixo perto do código fonte da aplicação. Consiste em testar métodos e funções individuais de classe, componente ou módulos usado pelo software.

Testado na Aplicação: O Servidor

Objetivo do Teste: o objetivo do teste foi verificar se a saída do console do professor e o resultado do código do aluno estão funcionando corretamente.

Resultado: Exibindo o resultado CORRETAMENTE

2 - Teste Funcional: Estes testes verificam a saída de uma ação e não verificam os estados intermediários do sistema ao executar essa ação.

Testado na Aplicação: Criar Sala

Objetivo do Teste: O objetivo do teste neste componente é ver se as mensagens estão sendo exibidas corretamente para o usuário professor.

Resultado: Exibindo as mensagens CORRETAMENTE. Quando os campos são preenchidos a mensagem de sala é criada e exibida para o usuário professor. E quando os campos não estão preenchidos ou estão faltando algum para ser preenchido a mensagem de erro é exibida para o usuário como esperado.

Testado na Aplicação: Editar Sala

Objetivo do Teste: O objetivo do teste neste componente é ver se a mensagem “Alterações Feitas com Sucesso” está sendo exibida corretamente para o usuário professor, os campos estão devidamente preenchidos quando acessar o componente e o botão do componente “Salvar Alterações” deve estar desabilitado até o usuário fazer uma alteração em alguns dos campos.

Resultado: Exibindo a mensagem CORRETAMENTE. Todos os campos estão devidamente preenchidos e o botão “Salvar Alterações” está fazendo a sua função CORRETAMENTE.

3 - Teste de Ponta-a-Ponta: Replica o Comportamento do usuário com o software em um ambiente de aplicativo completo.

Testando na Aplicação: Cadastro

Objetivo do Teste: O objetivo desse teste é ver se o componente está fazendo a sua função corretamente. Cadastrando o usuário e mostrando a mensagem de sucesso, ou mostrando a mensagem de erro para o usuário caso esteja algo errado no cadastro.

Resultado: Exibindo a Mensagem de sucesso para o usuário CORRETAMENTE, mais durante o teste foi detectado um BUG onde mesmo com os campos incompletos a mensagem de erro não é mostrada para o usuário. Esse BUG foi CORRIGIDO.

Testando na Aplicação: Login

Objetivo do Teste: O objetivo desse teste é ver se o componente está fazendo a sua função corretamente. Fazendo o login para entrar na aplicação com as credenciais corretas sendo do professor ou aluno e aparecer a mensagem de sucesso, caso as credenciais estejam incorretas deve exibir a mensagem de credenciais incorretas.

Resultado: Está fazendo toda a função CORRETAMENTE. Exibindo a mensagem quando o usuário é logado seja ele aluno ou professor e caso as credenciais estejam INCORRETAS a mensagem de erro está sendo exibido CORRETAMENTE.

4 - Teste de Fumaça: São testes básicos que verificam a funcionalidades básicas da aplicação. São feitas para terem execução rápida e sua meta é garantir que os principais recursos do sistema estejam funcionando conforme o esperado.

Testando na Aplicação: Testar o *Authguard*

Objetivo do Teste: O objetivo do teste é ver se o *Authguard* que é uma função que impede usuários não autenticados de acessar o sistema pela url, esteja funcionando corretamente exibindo uma mensagem de “Faça Login Primeiro”, caso o usuário não autenticado tente acessar páginas que não tem acesso. E retornar para a página de login caso o usuário coloque alguma url inválida.

Resultados: O *Authguard* está fazendo a função CORRETAMENTE. Tanto como impedir de usuários não autenticados de entrar no sistema pela url como colocar uma url inválida.

5. CONSIDERAÇÕES FINAIS

O protótipo da aplicação que está em modo Desenvolvimento, Code Class, é uma alternativa para avaliação de códigos-fonte mais voltado para um ambiente escolar, onde tem-se uma maior interação entre professor e aluno.

O grande obstáculo do software é a segurança, onde atualmente o mesmo se encontra com pouca segurança, já que é um protótipo, para evitar problemas como SQL Injection[3], por exemplo. No entanto, foram realizados diversos testes de software a fim de garantir uma certa qualidade para uso da ferramenta.

Por fim, espera-se que a ferramenta possa auxiliar professores e alunos, principalmente para as disciplinas que envolvam programação, tornando o processo de avaliação do código-fonte mais rápido e confiável.

6. TRABALHOS FUTUROS

- **Sistema de Alertas de Atividade:**
    - Implementação de um sistema de alertas que informa os alunos sobre novas atividades disponíveis para serem realizadas, incentivando a participação ativa e o cumprimento de prazos.

[3] É um tipo de vulnerabilidade em que um invasor usa uma parte do código SQL (Structured Query Language) para manipular um banco de dados e obter acesso a informações potencialmente valiosas. (https://www.kaspersky.com.br/resource-center/definitions/sql-injection)

- **Compatibilidade Móvel:**
    - Com um design adaptado para dispositivos móveis, a plataforma deve atender a crescente prevalência de alunos que utilizam smartphones, garantindo acessibilidade e conveniência.
- **Diversificação de Linguagens de Programação:**
    - Expansão do leque de linguagens de programação disponíveis para exercícios, abrangendo além do Python, incluindo JavaScript, C, Java e outras. Isso amplia a diversidade de experiências de aprendizado.
- **Correções Detalhadas:**
    - O sistema deve permitir que os alunos acessem as correções de suas atividades, proporcionando uma visão clara dos erros cometidos.

REFERÊNCIAS

GARLAN, David; PERRY, Dewayne E. Introduction to the special issue on software architecture. IEEE Trans. Software Eng., v. 21, n. 4, p. 269-274, 1995.

KAZMAN, Rick; ASUNDI, Jai; KLEIN, Mark. Quantifying the costs and benefits of architectural decisions. In: Proceedings of the 23rd International Conference on Software Engineering. ICSE 2001. IEEE, 2001. p. 297-306.

PAULA FILHO, W. P. Engenharia de software: fundamentos, métodos e padrões. Rio de Janeiro: Livros Técnicos e Científicos, 2001.

PAULA FILHO, W. P. Engenharia de software: fundamentos, métodos e padrões. São Paulo: Livros Técnicos e Científicos, 2000.