Plataforma *Web* de Auxílio de Discentes e Docentes Baseada nas Questões do POSCOMP e Enade

Felipe Davi do Nascimento Lopes[1], **Lucas Oliveira Bleyer**[1], **Edinilson da Silva Vida**[1]

[1]Instituto Federal de Santa Catarina (IFSC)
Rua Heitor Villa Lobos, 225, 88506-400 – Lages/SC – Brasil

{felipe.dn,lucas.b2003}@aluno.ifsc.edu.br

edinilson.vida@ifsc.edu.br

Abstract. *This paper presents a customizable web platform for searching questions, simulated exams, and performance reports, developed for the POSCOMP and Enade exams. Users can search questions by year, subject, difficulty, and exam type, and create personalized simulated exams. The platform generates feedback on answers and detailed performance reports. Built with React and Firebase, it supports real-time data synchronization. A mixed-method approach was used for data collection, focusing on user interaction and performance. Results show increased user engagement and improved exam preparation. The platform provides an adaptive, data-driven learning experience.*

Resumo. *Este artigo apresenta uma plataforma web customizável para pesquisa de questões, simulados e relatórios de desempenho, desenvolvida para as provas do POSCOMP e Enade. A plataforma permite a pesquisa de questões por ano, disciplina, dificuldade e tipo de prova, além da criação de simulados personalizados. São gerados feedbacks e relatórios detalhados de desempenho. Desenvolvida com React e Firebase, a plataforma oferece sincronização de dados em tempo real. A metodologia combinou coleta de dados quantitativos e qualitativos, com foco na interação do usuário e desempenho. Os resultados mostraram maior engajamento e melhoria na preparação para os exames. A plataforma oferece uma experiência de aprendizado adaptativa e baseada em dados.*

1. Introdução

O Exame Nacional para Ingresso na Pós-Graduação em Computação (POSCOMP) testa conhecimentos na área de Computação e tem como objetivo específico avaliar os conhecimentos de candidatos a Programas de Pós-Graduação em Computação oferecidos no Brasil (SBC, 2024). O Exame Nacional de Desempenho dos Estudantes (Enade) avalia o rendimento dos concluintes dos cursos de graduação em relação aos conteúdos programáticos previstos nas diretrizes curriculares dos cursos, o desenvolvimento de competências e habilidades necessárias ao aprofundamento da formação geral e profissional (INEP, 2024). A necessidade de ferramentas que aprimorem o estudo e a preparação para exames é clara.

Este projeto atende à demanda por uma ferramenta centralizada que facilite o estudo e a preparação para os exames POSCOMP e Enade. Atualmente, a falta de recursos

integrados e organizados prejudica tanto os estudantes quanto os professores, que enfrentam dificuldades para acessar materiais e preparar alunos de forma eficaz. A ausência de uma base de dados abrangente e ferramentas de simulação eficazes limita a otimização do processo de aprendizagem e avaliação.

Diante desse problema, o objetivo deste trabalho é auxiliar professores e estudantes na preparação para os exames Enade e POSCOMP por meio do desenvolvimento de uma plataforma *web* chamada Pesquise!, que tem como foco centralizar e organizar questões dos exames Enade e POSCOMP, facilitando a busca e o estudo direcionado. Para atingir esse objetivo, foram estabelecidas as metas específicas:

- Pesquisar, catalogar e cadastrar questões de provas anteriores do Enade e POSCOMP em um banco de dados.
- Implementar um serviço *web*, com módulos de plataforma *web*.
- Implementar um gerador de simulados com base nessa plataforma *web*.
- Desenvolver interfaces *web* para acessar as funcionalidades disponibilizadas pela plataforma.
- Avaliar a usabilidade e as funcionalidades do sistema.

A pesquisa adota uma abordagem quantitativa e qualitativa, com foco em dados práticos e percepções dos usuários. A coleta de dados foi realizada por meio de questionários quantitativos, utilizando a escala SUS *(System Usability Scale)* para avaliar a usabilidade da plataforma. A análise quantitativa foi feita com base nas pontuações da escala SUS, enquanto os dados qualitativos foram analisados por análise de conteúdo. Além disso, foi realizada uma pesquisa bibliográfica para embasar o estudo e comparar os resultados com pesquisas anteriores.

Este artigo é dividido em 6 seções gerais. Na seção 1, é apresentado o tema principal, discutindo sua relevância e contextualizando a necessidade e importância do projeto. Na seção 2 é o referencial teórico que fundamenta e embasa a pesquisa. A seção 3 detalha os requisitos do sistema e apresenta a modelagem da plataforma. Na seção 4, são descritas as funcionalidades desenvolvidas para a plataforma. Na seção 5, é realizada a análise crítica e discussão dos resultados obtidos. Por fim, na seção 6, são feitas as considerações finais, resumindo as principais descobertas, discutindo limitações do estudo e apontando possíveis direções para futuras pesquisas.

2. Referencial Teórico

Esta seção apresenta o referencial teórico, dividido em três partes distintas. Nas subseções 2.1 e 2.2, são discutidos, respectivamente, o Enade e o POSCOMP, descrevendo o histórico de cada prova, suas características de aplicação, objetivos e importância. Por fim, na subseção 2.3, são analisados projetos semelhantes ao deste estudo, detalhando suas principais características, vantagens e desvantagens.

2.1. Enade

O Enade é uma das ferramentas fundamentais do Sistema Nacional de Avaliação da Educação Superior (SINAES), criado pela Lei 10.861/2004 no Brasil. O principal objetivo do SINAES é garantir um processo nacional de avaliação das instituições de ensino superior, dos cursos de graduação e do desempenho acadêmico dos estudantes (BRASIL,

2004). O Enade é aplicado a cada três anos, conforme o ciclo do SINAES, e tem por finalidade mensurar e acompanhar o aprendizado e o desempenho dos discentes nos cursos de ensino superior.

A avaliação do Enade considera diversos aspectos, incluíndo os conteúdos programáticos estabelecidos nos currículos das graduações, as necessidades do mercado de trabalho e o nível mínimo de qualidade de um curso exigido pelo Ministério da Educação (MEC).

De acordo com a Nota Técnica 12/2017 da Diretoria de Avaliação da Educação Superior, conforme DAES (2017), a estrutura do Enade é composta por 40 questões divididas em duas partes. A primeira parte aborda a Formação Geral (FG), avaliando aspectos éticos, competentes e comprometidos com a sociedade. A segunda parte, denominada Componente Específico (CE), avalia o conhecimento específico de cada área de formação. A nota final do discente é obtida pela média ponderada das questões das partes FG e CE.

Conforme apresentado no Quadro 1, desde sua criação em 2004, a área de Tecnologia da Informação (TI) foi avaliada nos anos de 2005, 2008, 2011, 2014 e 2017. Os cursos são classificados em cinco categorias de conceito, de 1 a 5, sendo 1 o conceito mais baixo e 5 o melhor conceito. Cursos que não obtiveram conceito são classificados como Sem Conceito (SC).

Quadro 1. Panorama nacional dos cursos de Ciência da Computação

Ano	Cursos	Conceito 1	Conceito 2	Conceito 3	Conceito 4	Conceito 5
2005	270	1,5%	9,3%	48,9%	15,6%	4,4%
2008	295	2,0%	28,1%	32,5%	12,5%	5,4%
2011	803	7,7%	39,8%	33,6%	13,1%	3,7%
2014	291	4,8%	43,0%	27,1%	16,2%	6,9%
2017	311	5,5%	30,5%	31,2%	22,5%	9,3%

Fonte: MEC (2019)

O cálculo do conceito do Enade envolve a obtenção do desempenho médio dos alunos concluíntes nas partes de Formação Geral e Componente Específico. Os resultados são convertidos em uma variável discreta, classificando os cursos de acordo com parâmetros pré-definidos.

A análise das questões do Enade leva em consideração seu índice de facilidade, classificando-as de acordo com o percentual de acertos, conforme pode ser observado no Quadro 2. Questões muito fáceis têm percentuais de acertos iguais ou superiores a 86%, enquanto as muito difíceis têm percentuais de acertos iguais ou inferiores a 15%.

Essas características do Enade permitem a construção de sistemas de apoio e preparação para o exame, categorizando as questões e apresentando-as de acordo com a evolução dos alunos.

2.2. POSCOMP

O Exame Nacional para Ingresso na Pós-Graduação em Computação é organizado desde 2002 pela Sociedade Brasileira de Computação (SBC) e pela Fundação Universidade Empresa de Tecnologia e Ciências (FUNDATEC), que é a responsável pelas inscrições e

Quadro 2. Classificação de questões segundo o índice de facilidade

Classificação	Índice de Facilidade
Muito fácil	$\geq 0,86$
Fácil	$0,61$ a $0,85$
Médio	$0,41$ a $0,60$
Difícil	$0,16$ a $0,40$
Muito difícil	$\leq 0,15$

Fonte: (DAES, 2017)

aplicação da prova. Em parceria com a Sociedade Peruana de Computação, desde 2006, o exame passou a ser realizado no Peru (SBC, 2024). A partir de 2023, o exame teve o formato de aplicação das provas alterado para exclusivamente *on-line*, considerando a possibilidade de alcançar mais participantes, evitar a necessidade de deslocamento e proporcionar um melhor resultado dos conhecimentos na área de computação (FUNDATEC, 2023).

O POSCOMP é uma prova aplicada anualmente, composta por 70 questões teórico-objetivas. A prova tem duração de 4 horas e é dividida em três áreas de conhecimento: Matemática (20 questões), Fundamentos da Computação (30 questões) e Tecnologia de Computação (20 questões). Cada uma dessas áreas é subdividida em subáreas específicas, somando um total de 25 subáreas. O Quadro 3 especifica as áreas e subáreas do exame:

Quadro 3. Áreas e subáreas do POSCOMP

POSCOMP		
Matemática	Fundamentos da Computação	Tecnologia de Computação
Álgebra Linear	Análise de Algoritmos	Banco de Dados
Análise Combinatória	Algoritmos e Estrutura de Dados	Compiladores
Cálculo Diferencial e Integral	Arq. e Organização de Computadores	Computação Gráfica
Geometria Analítica	Circuitos Digitais	Engenharia de *Software*
Lógica Matemática	Linguagens de Programação	Inteligência Artificial
Matemática Discreta	Ling. Formais, Autômatos e Computab.	Processamento de Imagens
Probabilidade e Estatística	Organização de Arquivos e Dados	Redes de Computadores
	Sistemas Operacionais	Sistemas Distribuídos
	Técnicas de Programação	
	Teoria dos Grafos	

Fonte: FUNDATEC (2023)

De acordo com os números disponibilizados pela SBC (2023), a média de acertos nas provas, apresentada no Quadro 4 para o período de 2015 a 2023, reflete a alta exigência e dificuldade do exame, mesmo considerando que o máximo de acertos possível é 70. Os candidatos que realizam o exame têm acesso ao seu resultado, individualmente, bem como a indicação das questões certas e erradas, a média e o desvio padrão (SBC, 2024).

Quadro 4. Médias do POSCOMP

Ano	Nº Inscritos Homologados	Média Total
2023	-	31,66
2022	-	28,94
2019	-	28,67
2018	3447	28,8
2017	4017	32,2
2016	4219	29,62
2015	3559	29,1

Fonte: SBC (2023)

2.3. Trabalhos Similares

Esta seção apresenta projetos semelhantes ao desenvolvido, focados em facilitar o acesso a questões do Enade e POSCOMP, destacando suas características, vantagens e limitações, e evidenciando as lacunas que justificam a proposta do presente trabalho.

O primeiro trabalho, de Callegari et al. (2020), é o aplicativo POSCOMP *Math*. Este aplicativo oferece simulados de Matemática com questões de edições anteriores, resoluções passo a passo e análise de desempenho. Embora o sistema atenda à necessidade de simulados, ele se limita a uma única disciplina (Matemática) e não oferece uma plataforma integrada para outras disciplinas, como a proposta neste projeto. Além disso, a atualização de questões foi sugerida para futuras edições, indicando uma limitação quanto à manutenção contínua do banco de dados.

Tiago et al. (2020) desenvolveu uma plataforma composta por um sistema *web* e um aplicativo móvel para a preparação do Enade e POSCOMP. Com um banco de mais de 1500 questões, a plataforma gera simulados e oferece gráficos de desempenho. Contudo, a plataforma não aborda a personalização da experiência de estudo baseada no desempenho do usuário, o que limita sua eficácia na adaptação do conteúdo para as necessidades específicas de cada aluno.

Zotti et al. (2016) descreve um sistema de simulados que utiliza técnicas de recomendação baseadas no desempenho do usuário. Embora a recomendação personalize a experiência, o sistema não se integra com exames como o Enade ou POSCOMP, o que torna sua aplicabilidade mais restrita em comparação com a solução proposta neste projeto, que visa atender especificamente a essas duas avaliações.

Finalmente, Queiroz e Antonello (2019) desenvolveram uma plataforma *web* para o cadastro e manutenção de questões, utilizando React e Firebase. Embora o sistema atenda aos objetivos de gerenciamento de questões, ele não oferece funcionalidades

avançadas de pesquisa e simulação, nem a análise detalhada de desempenho dos alunos, recursos essenciais no projeto em questão.

No Quadro 5, a coluna Autores identifica os criadores de cada sistema. *Framework* lista a tecnologia usada, incluindo React, Angular e RapidMiner. *Database* indica a base de dados escolhida, sendo Firebase a mais comum, exceto em um caso que usa Oracle. Portabilidade descreve a plataforma de acesso, como *Web*, *Mobile* ou *Web Service*. A coluna Pesquisa destaca se o sistema permite busca personalizada de questões, recurso disponível apenas no sistema proposto. Relatório indica se há geração de relatórios de desempenho, presente na maioria dos sistemas. Por fim, Simulado sinaliza a presença de módulos de simulado, outro recurso comum, exceto em um dos sistemas comparados.

Quadro 5. Análise comparativa dos trabalhos similares

Autores	*Framework*	*Database*	Portabilidade	Pesquisa	Relatório	Simulado
Pesquise! (Sistema proposto)	React	Firebase	*Web*	Sim	Sim	Sim
Callegari et al. (2020)	React Native	Firebase	*Mobile*	Não	Sim	Sim
Tiago et al. (2020)	Angular	Firebase	*Web Service*	Não	Não	Sim
Zotti et al. (2016)	RapidMiner	Oracle	Não consta	Não	Sim	Sim
Queiroz e Antonello (2019)	React	Firebase	*Web*	Não	Não	Não

3. Especificações de Requisitos e Modelagem

Esta seção descreve as tecnologias e metodologias usadas no desenvolvimento do projeto, incluindo exemplos práticos da prototipação do sistema com telas representativas e suas explicações. Um diagrama de caso de uso também é apresentado, ilustrando as principais interações entre os usuários e o sistema. O modelo de dados, que organiza e estrutura as informações dentro do sistema, também é abordado, assegurando a eficiência e integridade na manipulação dos dados.

Cada subseção explora detalhadamente os componentes que sustentam o desenvolvimento do sistema, destacando sua importância e aplicação no contexto do projeto. Essa abordagem proporciona uma visão abrangente dos requisitos necessários para garantir a funcionalidade e o sucesso do sistema proposto.

3.1. Tecnologias e Metodologias

Nesta seção, serão detalhadas as principais tecnologias e metodologias adotadas no desenvolvimento deste sistema *web*. As tecnologias escolhidas incluem uma combinação de linguagens de programação, *frameworks*, bancos de dados e outras ferramentas que formam a base técnica do sistema. As metodologias abordam os princípios e práticas que guiaram a gestão do projeto, desde a concepção inicial até a implementação e os testes.

3.1.1. React

React é um *framework* destinado a criação de interfaces de usuário interativas com uma abordagem declarativa, tornando o código mais previsível e facilita a depuração. Componentes encapsulados gerenciam seu próprio estado, quando combinados, formam interfaces complexas (REACT, 2024). No projeto, foi utilizado para criar a versão *web* da plataforma, garantindo desempenho e eficiência no desenvolvimento.

3.1.2. Firebase

Firebase é uma plataforma do Google que oferece armazenamento em nuvem, autenticação de usuários e banco de dados em tempo real para o desenvolvimento de aplicativos móveis e *web* (FIREBASE, 2024). No projeto, o Firebase foi utilizado para armazenar questões e dados de *login*, garantindo acesso instantâneo às informações e segurança na autenticação.

3.1.3. Trello

O Trello é uma ferramenta de gerenciamento de projetos que organiza tarefas de forma visual e colaborativa por meio de quadros, listas e cartões. Com funcionalidades como prazos e integração com Google Drive e Slack, facilita a colaboração e produtividade (TRELLO, 2024). No projeto, o Trello foi usado para organizar e monitorar o progresso das tarefas, com quadros representando etapas, listas detalhando tarefas e cartões indicando responsabilidades e prazos, mantendo a equipe alinhada e centralizando a gestão do projeto.

3.1.4. Scrum

O Scrum é uma métodologia ágil para gerenciamento de projetos que organiza o trabalho em ciclos curtos e iterativos chamados *Sprints*, de duas a quatro semanas, permitindo ajustes contínuos com base no *feedback* (CRUZ, 2013). No desenvolvimento, as tarefas foram divididas em *Sprints* semanais, com o orientador atuando como *Project Manager*. Essa estrutura garantiu uma gestão ágil, com entregas incrementais e rápida adaptação a mudanças e *feedbacks*.

3.1.5. Figma

O Figma é uma ferramenta de *design* colaborativo em nuvem que facilita a criação de interfaces de usuário (UI - *User Interface*) e experiências de usuário (UX - *User Experience*) com colaboração em tempo real e protótipos interativos (FIGMA, 2024). No projeto, foi utilizado para desenvolver *wireframes* e protótipos das telas, como pesquisa, registro e *login*, permitindo visualização e planejamento das interfaces antes da implementação.

3.2. Teste SUS de Usabilidade

O *System Usability Scale* (SUS) é uma ferramenta de avaliação de usabilidade desenvolvida por John Brooke (BROOKE, 1986) para fornecer uma medição rápida e eficiente da percepção dos usuários sobre a usabilidade de sistemas. Com um questionário de dez itens em escala *Likert* de cinco pontos, o SUS permite avaliar a facilidade de uso percebida pelos usuários.

Em uma análise retrospectiva, Brooke (BROOKE, 2013) destacou a eficácia e a ampla adoção do SUS em estudos de usabilidade, o que reforça sua utilidade para avaliar sistemas variados. A aplicação do SUS na plataforma proposta visa identificar melhorias na experiência do usuário de forma prática e orientada por dados.

3.3. Requisitos Funcionais, Não Funcionais e Protótipo

Nesta subseção, são definidos os requisitos funcionais e não funcionais do sistema Pesquise!, juntamente com a apresentação de um protótipo baseado nesses requisitos. O protótipo serve como uma representação visual do fluxo de interação do usuário com o sistema, permitindo identificar e corrigir possíveis inconsistências ou realizar melhorias na interface e na experiência do usuário.

3.3.1. Requisitos Funcionais e Não Funcionais

Os requisitos essenciais para o pleno funcionamento da plataforma estão organizados em duas categorias principais: requisitos funcionais, que detalham as funcionalidades e comportamentos necessários do sistema; e requisitos não funcionais, que estabelecem as qualidades e restrições, como segurança, desempenho e usabilidade.

Os Requisitos Funcionais (RF) definem as operações fundamentais que a plataforma deve realizar para atender às necessidades dos usuários de maneira eficiente e segura.

- **RF01** - O sistema deve permitir a autenticação e recuperação de contas.
- **RF02** - O sistema deve oferecer níveis de acesso distintos para professores e alunos.
- **RF03** - O sistema deve manter um banco de dados com questões do Enade e POSCOMP.
- **RF04** - O sistema deve permitir pesquisa detalhada e criação de simulados personalizados.
- **RF05** - O sistema deve gerar relatórios detalhados de desempenho por área e disciplina.

Os Requisitos Não Funcionais (RNF) especificam os critérios de qualidade que o sistema deve obedecer, garantindo sua confiabilidade, escalabilidade e conformidade com as normas vigentes.

- **RNF01** - O sistema deve estar em conformidade com a Lei Geral de Proteção de Dados Pessoais (LGPD).
- **RNF02** - O sistema deve permitir autenticação segura e armazenamento de dados de forma escalável.
- **RNF03** - O sistema deve ter uma interface intuitiva e responsiva.
- **RNF04** - O sistema deve ser otimizado para eficiência no uso de recursos e redução de custos de hospedagem.

3.3.2. Protótipo da Interface da Plataforma

O protótipo permitiu simular cenários reais, coletar *feedback* e ajustar o *design* antes do desenvolvimento final. A Figura 1 ilustra a tela de pesquisa e seleção de questões. As questões podem ser adicionadas a simulados personalizados, e a interface mostra em tempo real a quantidade de questões e tópicos selecionados, além de permitir a consulta do gabarito sem necessidade de iniciar um simulado.

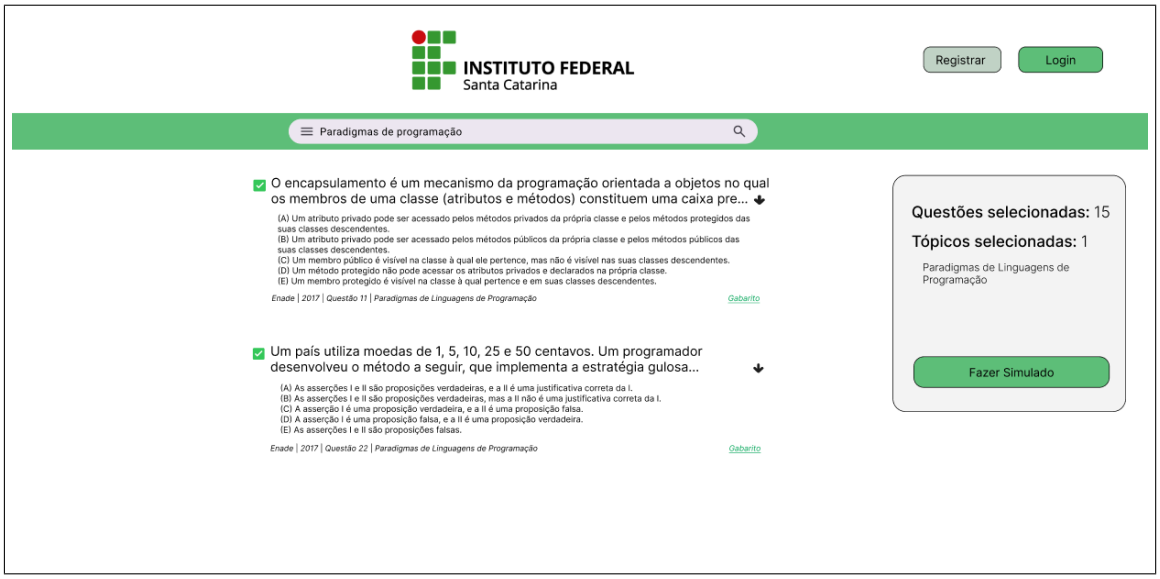


Figura 1. Tela de pesquisa de questões

A Figura 2 mostra a tela do simulado, com uma barra de progresso que atualiza em tempo real, indicando o número de questões respondidas e restantes. O usuário pode selecionar uma alternativa por questão e navegar entre elas livremente, podendo finalizar o simulado a qualquer momento. Um cronômetro registra o tempo total da realização do simulado.

Figura 2. Tela do simulado

A Figura 3 mostra a tela de resultados do simulado, com a barra de progresso indicando questões corretas e incorretas. Alternativas erradas são destacadas em vermelho e as corretas em verde. A tela exibe estatísticas detalhadas, como porcentagem de acertos e erros, matérias abordadas, e aproveitamento geral, além de um gráfico de radar que ilustra áreas de facilidade e dificuldade. O usuário pode revisar suas respostas ou refazer o simulado.

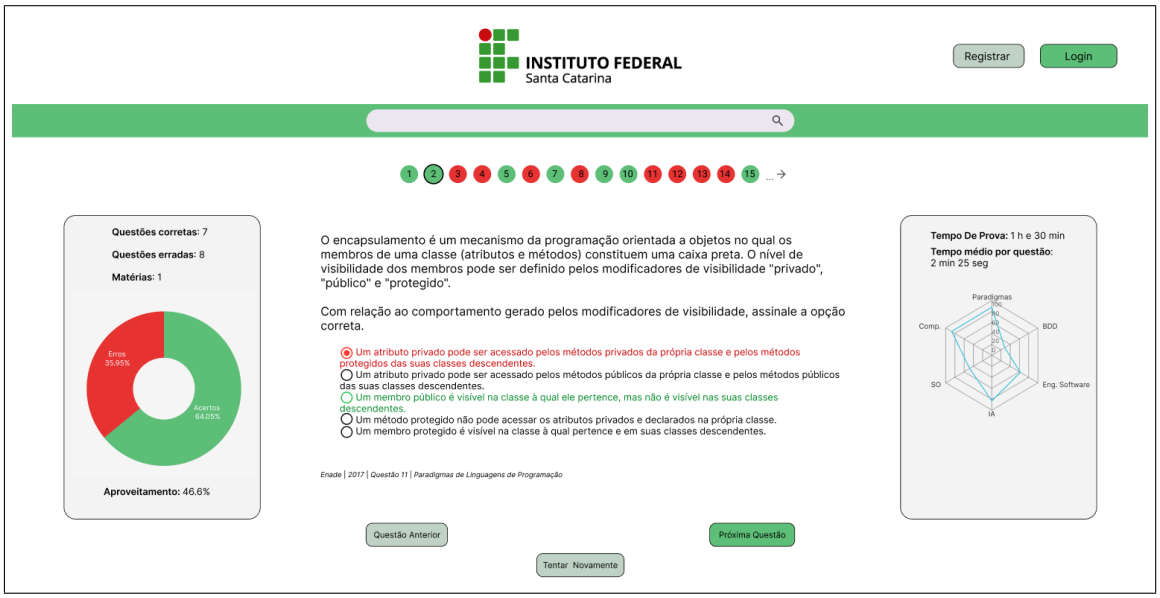


Figura 3. Tela de resultado do simulado

A Figura 4 mostra a tela de registro de questões. O usuário deve escolher o tipo de prova (Enade ou POSCOMP), a disciplina correspondente e o ano da prova, garantindo a correta categorização. Também é necessário informar o número da questão, o nível de dificuldade e o texto base. As alternativas devem ser inseridas individualmente, com a opção de incluir imagens, se necessário.

Figura 4. Tela para o registro de questões

A Figura 5 exibe a tela de estatísticas do usuário, dividida em duas partes: uma apresenta estatísticas gerais, como o total de questões resolvidas e acertos e erros, com gráficos de setores e radar; a outra parte oferece uma visão detalhada do desempenho, permitindo a personalização da busca no histórico e visualização de acertos ao longo dos dias com um gráfico de linhas, que permite acesso direto aos simulados de cada dia.

Figura 5. Tela de estatísticas

3.4. Diagrama de Caso de Uso

Esta subseção apresenta o diagrama de caso de uso da plataforma *web*, que destaca as principais interações entre usuários e funcionalidades. A Figura 6 exemplifica essas interações, facilitando a visualização dos requisitos funcionais e servindo como referência no desenvolvimento e teste das funcionalidades.

Figura 6. Digrama de caso de uso do sistema

3.5. Modelo de Banco de Dados

A estrutura do banco de dados foi configurada no Firebase, garantindo a integridade e a sincronização em tempo real dos dados. A Figura 7 ilustra o diagrama do banco de dados, destacando as coleções para armazenar dados do usuário, questões do Enade e do POSCOMP e simulados.

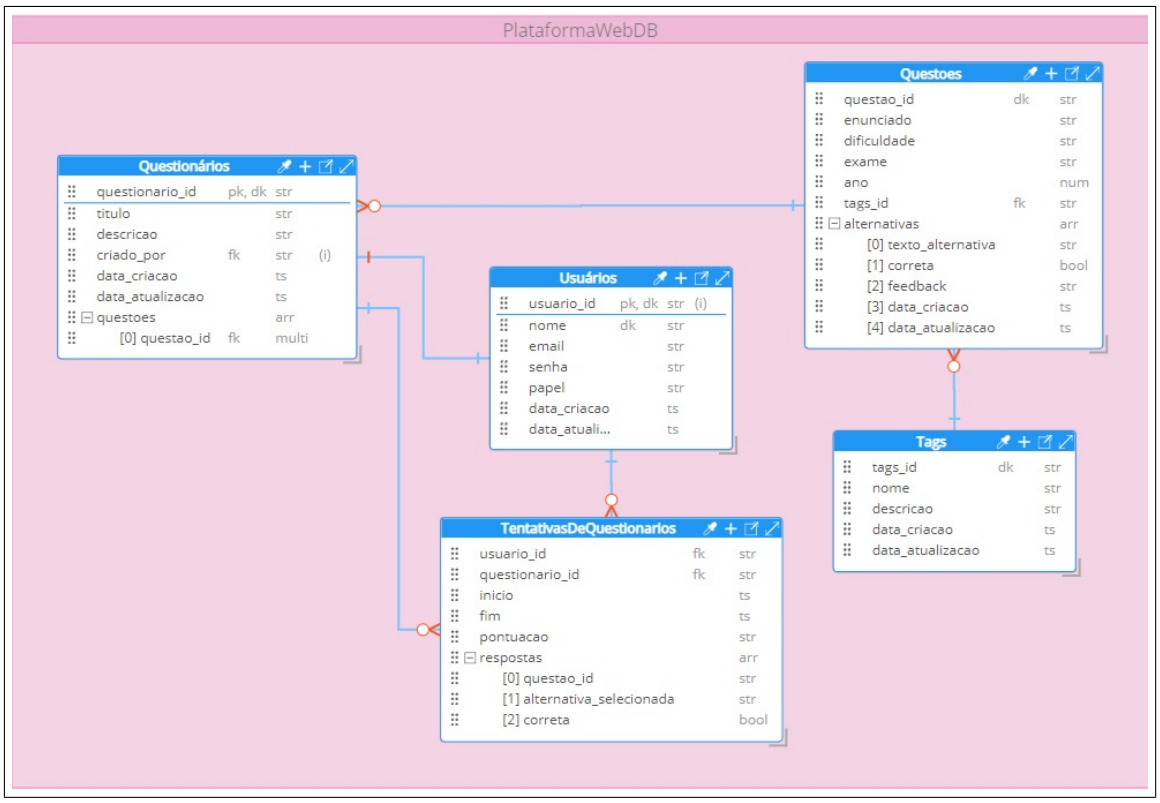


Figura 7. Modelo conceitual do banco de dados no Firebase

4. Desenvolvimento

A seção de desenvolvimento da plataforma foi organizada em módulos específicos: o Módulo de Autenticação do Usuário gerencia o *login*, registro e recuperação de senha, integrando validações e *Firebase Auth*. O Módulo de Questões permite o cadastro e edição de questões por administradores. O Módulo de Pesquisa e Simulados facilita a busca de questões e criação de simulados personalizados. O Módulo de Relatório exibe resultados detalhados dos simulados, com gráficos e estatísticas do desempenho do usuário.

O projeto foi hospedado no *Firebase* e gerenciado através do GitHub[1], garantindo controle das alterações no código e facilitando a colaboração. O uso do GitHub permitiu o armazenamento centralizado do código e o acompanhamento das versões. As tarefas foram organizadas no Trello, com quadros separados para cada *Sprint* e os objetivos de cada módulo. A Figura 8 mostra o quadro do Trello, destacando a estrutura das *sprints* definidas.

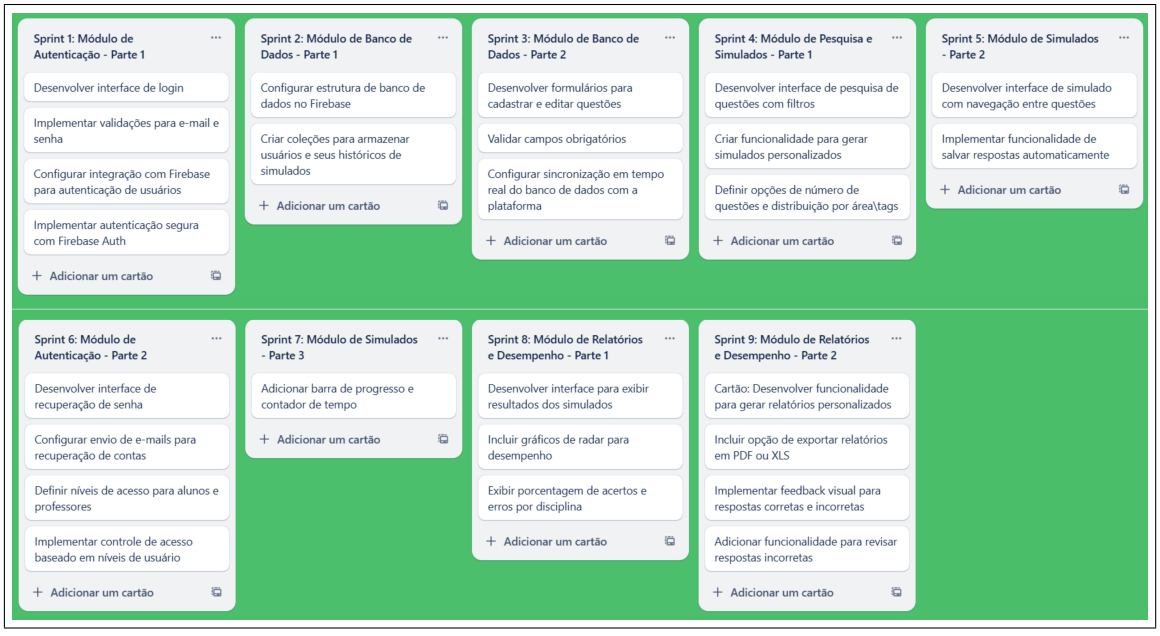


Figura 8. Quadro do Trello com a estrutura das sprints

[1]Repositório do projeto: `https://github.com/FelipeDNL/tcc-site-pesquisas-enade-poscomp`

4.1. Módulo de Autenticação do Usuário

Na tela de *login* do sistema, ilustrada na Figura 9, os usuários podem inserir suas credenciais (*e-mail* e senha) para acessar a plataforma de maneira segura. Em caso de esquecimento da senha, a interface também oferece a opção de recuperação, orientando o usuário para redefinir suas credenciais ou registrar uma nova conta.

Figura 9. Interface do *login*

As validações de *login* foram implementadas para assegurar que os dados inseridos atendam aos padrões de segurança. A integração com o *Firebase Auth* foi configurada, fornecendo uma camada extra de proteção durante o acesso. A Figura 10 ilustra o trecho do código que realiza a validação e autenticação dos usuários, garantindo que apenas aqueles com credenciais válidas consigam acessar a plataforma.

```
function Login() {

    const [email, setEmail] = useState("");
    const [senha, setSenha] = useState("");
    const { login } = useUser(); // Pega a função login do contexto
    const navigate = useNavigate();

    const handleLogin = (e) => {

        e.preventDefault();

        signInWithEmailAndPassword(auth, email, senha)
        .then((userCredential) => {
            const user = userCredential.user;

            // Chame a função de login com os dados do usuário
            login({
                name: user.displayName || "Usuário",
                email: user.email,
                photo: user.photoURL || '/default-user.jpg'
            });

            Swal.fire({
                icon: "success",
                title: "Login feito!",
                text: "Bem-vindo de volta!"
            });

            navigate('/')
        })
        .catch((error) => {
            console.error("Erro ao fazer login:", error.message);

            Swal.fire({
                icon: "error",
                title: "Erro ao fazer login",
                text: error.message
            });
        });
    };
```

Figura 10. Código de *login* do usuário com Firebase

A Figura 11 apresenta a tela de registro, que faz parte do módulo de registro de conta, onde os usuários inserem nome, sobrenome, *e-mail* e senha. Esse processo de registro conta com validações de segurança que verificam a força da senha e a formatação correta do *e-mail* e o tipo de conta.

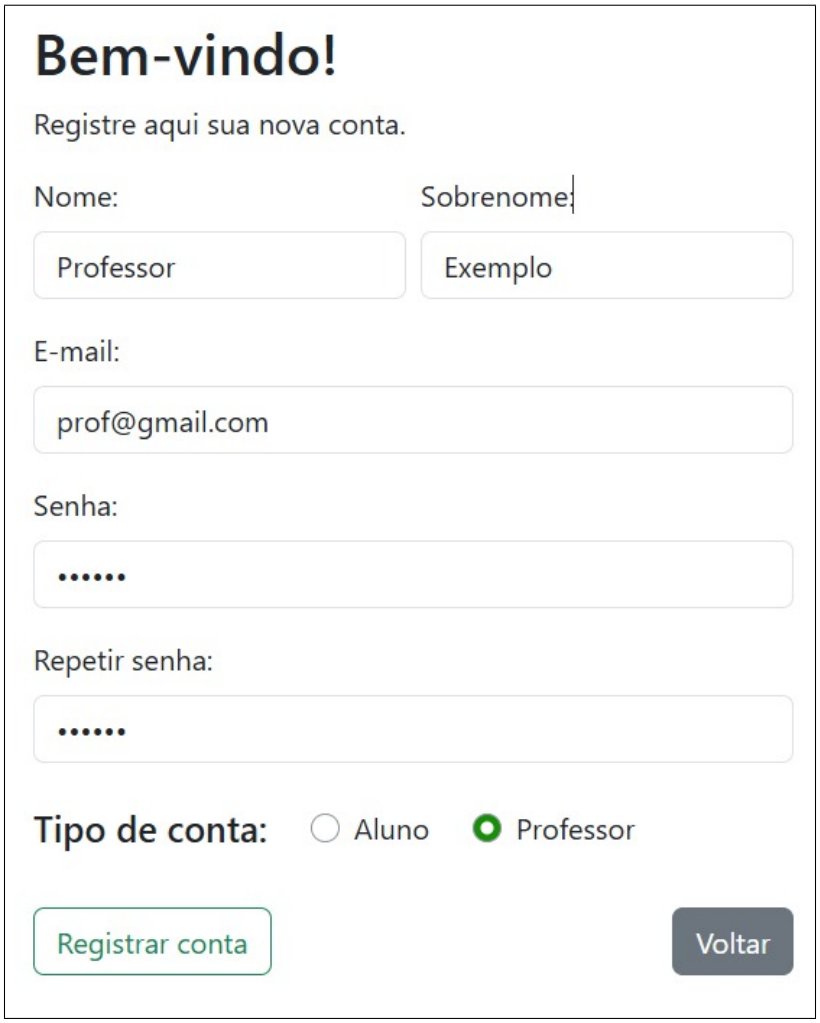
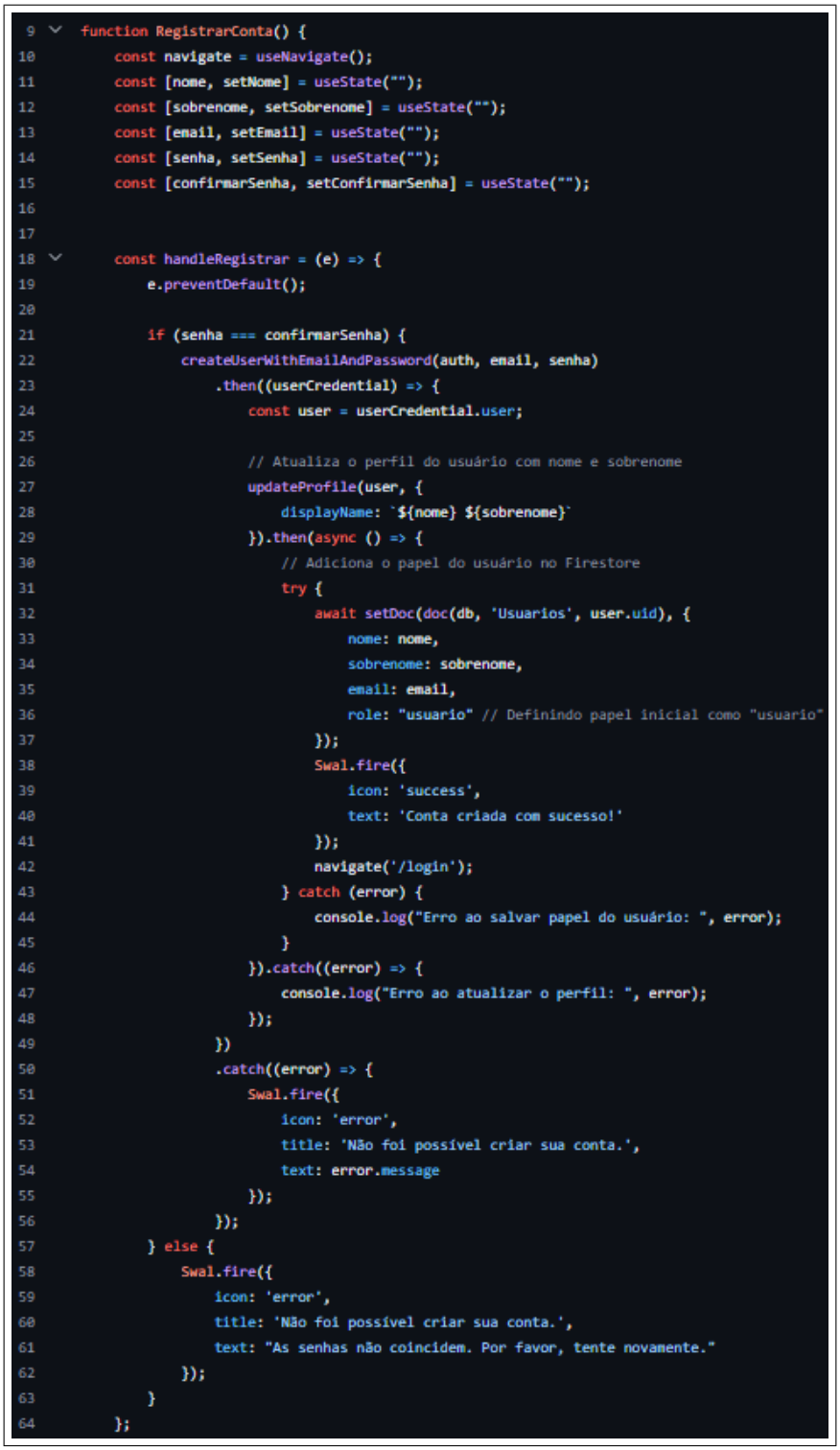


Figura 11. Interface de registro do usuário

A integração com o *Firebase Auth* adiciona uma camada de segurança extra ao gerenciamento das credenciais, garantindo que apenas dados válidos processados. A Figura 12 exibe o trecho de código responsável pelo registro e autenticação segura no *Firebase Auth*.



```
function RegistrarConta() {
    const navigate = useNavigate();
    const [nome, setNome] = useState("");
    const [sobrenome, setSobrenome] = useState("");
    const [email, setEmail] = useState("");
    const [senha, setSenha] = useState("");
    const [confirmarSenha, setConfirmarSenha] = useState("");


    const handleRegistrar = (e) => {
        e.preventDefault();

        if (senha === confirmarSenha) {
            createUserWithEmailAndPassword(auth, email, senha)
                .then((userCredential) => {
                    const user = userCredential.user;

                    // Atualiza o perfil do usuário com nome e sobrenome
                    updateProfile(user, {
                        displayName: `${nome} ${sobrenome}`
                    }).then(async () => {
                        // Adiciona o papel do usuário no Firestore
                        try {
                            await setDoc(doc(db, 'Usuarios', user.uid), {
                                nome: nome,
                                sobrenome: sobrenome,
                                email: email,
                                role: "usuario" // Definindo papel inicial como "usuario"
                            });
                            Swal.fire({
                                icon: 'success',
                                text: 'Conta criada com sucesso!'
                            });
                            navigate('/login');
                        } catch (error) {
                            console.log("Erro ao salvar papel do usuário: ", error);
                        }
                    }).catch((error) => {
                        console.log("Erro ao atualizar o perfil: ", error);
                    });
                })
                .catch((error) => {
                    Swal.fire({
                        icon: 'error',
                        title: 'Não foi possível criar sua conta.',
                        text: error.message
                    });
                });
        } else {
            Swal.fire({
                icon: 'error',
                title: 'Não foi possível criar sua conta.',
                text: "As senhas não coincidem. Por favor, tente novamente."
            });
        }
    };
```

Figura 12. Código de registro do usuário com Firebase

4.2. Módulo de Questões

Somente usuários com perfil de professor têm permissão para registrar novas questões na plataforma. A Figura 13 exibe o formulário de cadastro de questões, disponível para esses usuários.

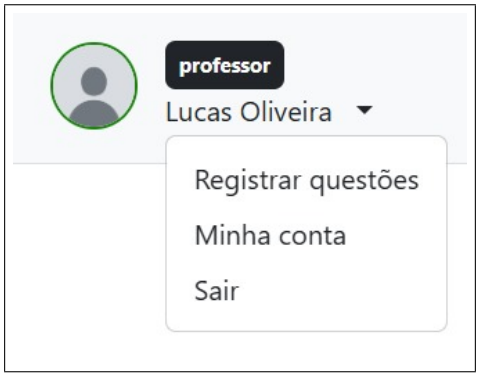


Figura 13. Opção de registro de questões para o perfil de professor

O cadastro e a edição de questões são feitos por meio de um formulário destinado a criação de questões do Enade e POSCOMP. Nas Figuras 14 e 15, são apresentadas as partes do formulário de cadastro de questões, que assegura que todos os dados necessários sejam preenchidos corretamente.

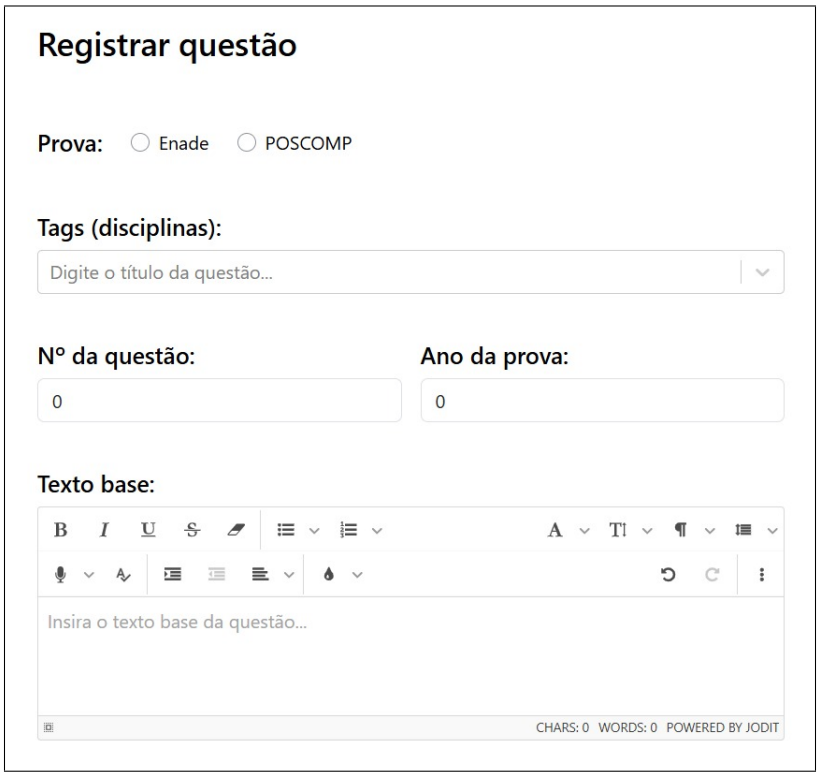


Figura 14. Interface do formulário de cadastro de questões

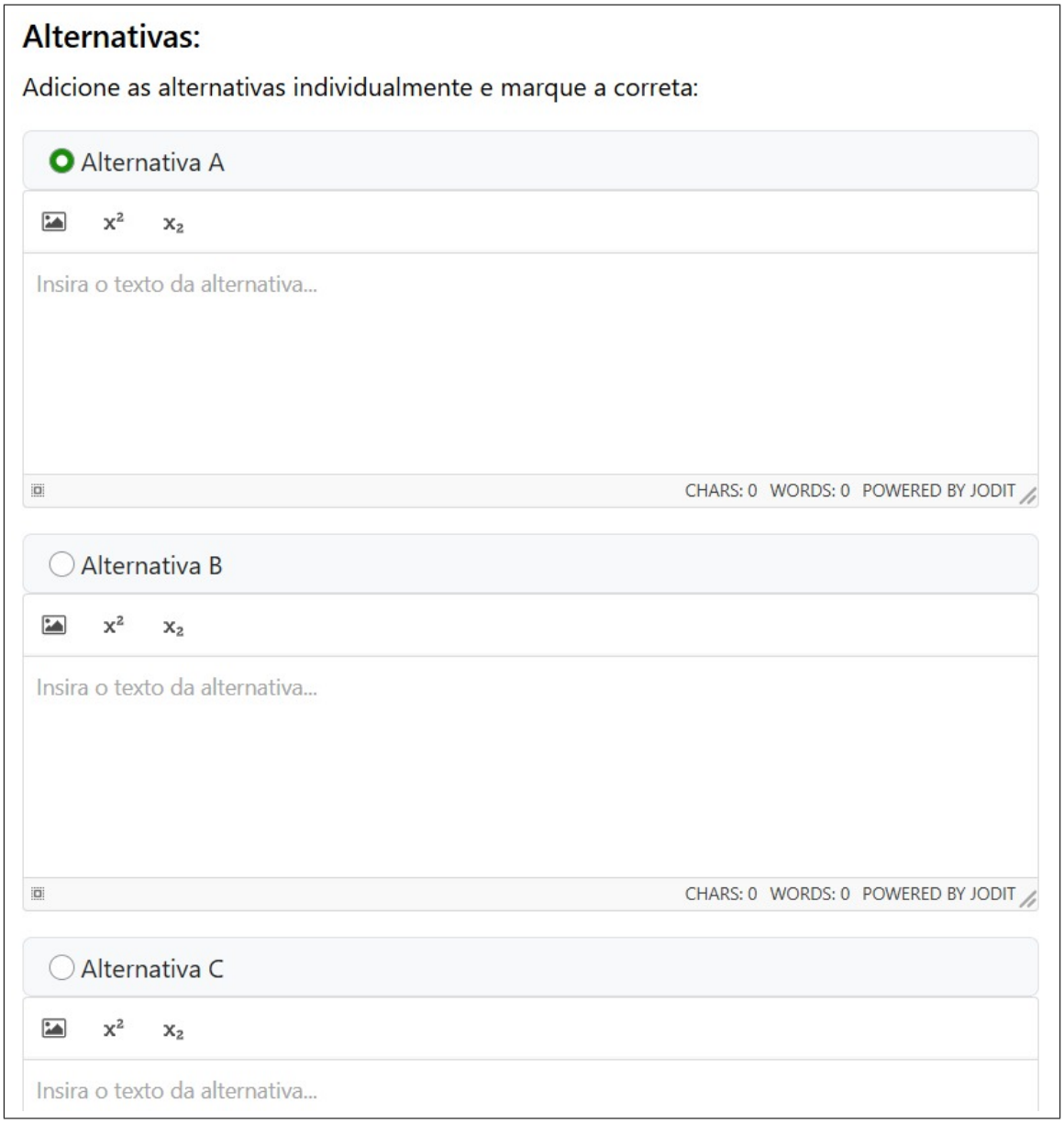


Figura 15. Interface do formulário de cadastro das alternativas das questões

A sincronização em tempo real permite que as questões do banco de dados sejam atualizadas instantaneamente na interface do usuário. O *Firestore* é um banco de dados NoSQL do Firebase que permite a sincronização em tempo real entre o servidor e os dispositivos dos usuários. Já o *Typesense* é uma ferramenta de busca de código aberto, projetada para oferecer consultas rápidas e eficientes. A Figura 16 apresenta o trecho de código responsável pela sincronização entre o *Firestore* e o *Typesense*.

Na função `syncFirestoreTo\textit{Typesense}`, as questões são recuperadas do *Firestore* e verificadas para determinar se a coleção "Questoes" já existe no *Typesense*. Caso não exista, a coleção é criada. Em seguida, os documentos são enviados ao *Typesense* com a opção `upsert`, que adiciona ou atualiza os registros conforme necessário.

```
// Função para sincronizar Firestore com Typesense
async function syncFirestoreToTypesense() {
  try {
    // Pega as questões do Firestore
    const documents = await getQuestionsFromFirestore();

    // Verifica se a coleção já existe
    const collections = await client.collections().retrieve();
    const collectionExists = collections.some((col) => col.name === 'Questoes');

    if (!collectionExists) {
      // Cria a coleção 'questoes' se ela não existir
      await client.collections().create(schema);
      console.log('Coleção questoes criada com sucesso.');
    }

    // Envia as questões ao Typesense com upsert
    const result = await client.collections('Questoes').documents().import(documents, { action: 'upsert' });
    console.log('Sincronização completa!', result);
  } catch (error) {
    console.error('Erro ao sincronizar com Typesense:', error.importResults);
  }
}
```

Figura 16. Código de sincronização das questões com Firebase

O ChatGPT foi utilizado como uma ferramenta auxiliar para a catalogação das questões do POSCOMP. O processo envolveu a mescla das questões com os respectivos gabaritos, culminando na geração de um arquivo no formato JSON. Esse arquivo, contendo um total de 1540 questões, abrange os exames realizados entre os anos de 2002 e 2024, incluindo as questões anuladas.

Um detalhe peculiar foi observado no ano de 2017, que contou com duas versões distintas de provas, ambas devidamente incluídas no catálogo. Contudo, devido à limitação de tempo, apenas as questões dos anos mais recentes, 2023 e 2024, foram revisadas com maior atenção. Para trabalhos futuros, há a necessidade de aprimorar o processo de inclusão de imagens e fórmulas matemáticas para complementar o conteúdo das demais questões.

Além disso, foi desenvolvido um *script* em Node.js que se conecta ao banco de dados Firebase para realizar a carga automática de todas as questões catalogadas. A Figura 17 apresenta o *script* que facilita a integração com a plataforma, permitindo o armazenamento organizado e a recuperação eficiente das questões por meio de consultas. A solução garante escalabilidade e flexibilidade para futuras atualizações, como a adição de imagens e fórmulas.

script_import.js > ...

```
const admin = require("firebase-admin");
const fs = require("fs");
const path = require("path");

// Caminho para o arquivo de credenciais
const serviceAccount = require("./token.json");

admin.initializeApp({
  credential: admin.credential.cert(serviceAccount),
});

const firestore = admin.firestore();

// Pasta onde os arquivos JSON estão localizados
const jsonDir = "./jsonFiles"; // Altere para o caminho correto

// Função para carregar os dados no Firestore
const importData = async () => {
  const collectionRef = firestore.collection("Questoes");

  // Lê todos os arquivos JSON na pasta
  const files = fs.readdirSync(jsonDir).filter((file) => file.endsWith(".json"));

  for (const file of files) {
    console.log(`Processando o arquivo: ${file}`);
    const filePath = path.join(jsonDir, file);
    const data = JSON.parse(fs.readFileSync(filePath, "utf8"));

    // Itera pelas questões do arquivo
    for (const key in data) {
      const questionData = data[key];

      // Adiciona o documento à coleção com uma ID gerada automaticamente
      await collectionRef.add(questionData);
      console.log(`Questão ${questionData.numeroQuestao} do ano ${questionData.anoProva} adicionada com sucesso.`);
    }
  }
  console.log("Processo de importação concluído!");
};

importData().catch((error) => {
  console.error("Erro ao importar dados:", error);
});
```

Figura 17. *Script* responsável pela carga das questões

4.3. Módulo de Pesquisa e Simulados

O módulo de Pesquisa e Simulados permite o usuário localizar questões específicas usando palavras-chave, como “poscomp linguagens”, filtrando diretamente por exame e tema. Na Figura 18, é possível ver o resultado dessa busca, com as questões relacionadas exibidas na tela. O usuário pode selecionar as questões de interesse e, ao lado direito, visualizar o total de questões escolhidas para o simulado. Com a opção “Iniciar simulado” já disponível, o usuário consegue iniciar um simulado personalizado com as questões selecionadas e tempo de simulado .

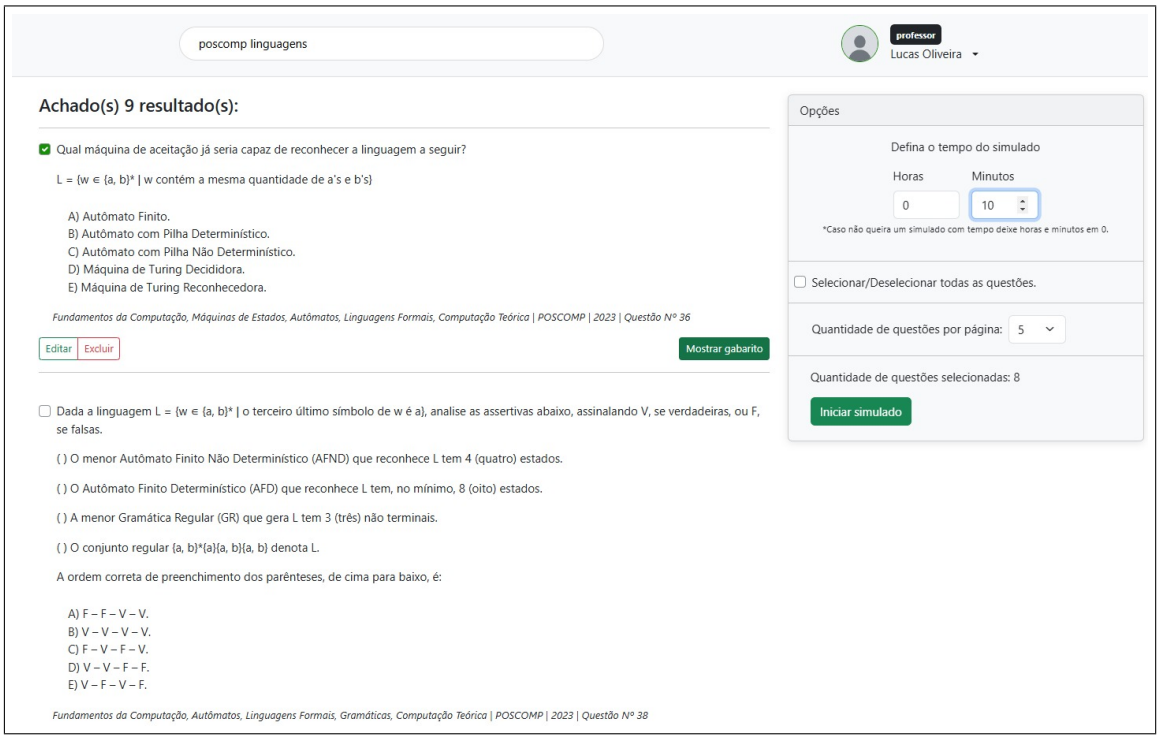


Figura 18. Interface de pesquisa de questões e criação de simulado

A Figura 19 apresenta o trecho do código de configuração do cliente *Typesense*. Esse cliente é configurado para se conectar ao servidor *Typesense* especificado, utilizando o *host*, porta e protocolo corretos. A chave de API permite realizar operações de busca com segurança, e o `connectionTimeoutSeconds` define o tempo de espera para conexão antes de uma possível falha.

```
// Configuração do Typesense
const client = new Typesense.Client({
  nodes: [
    {
      host: '9th5gyxsw6bv8kc1p-1.a1.typesense.net',  // Seu host do Typesense (ou da nuvem)
      port: '443',       // Porta onde o Typesense está rodando
      protocol: 'https',
    }
  ],
  apiKey: 'MdjjgNcf8mCaTzdm3lqCNcIytZZJML9q',          // Chave da API do Typesense - Chave Search Only
  connectionTimeoutSeconds: 2,
});
```

Figura 19. Código da configuração do *Typesense*

A Figura 20 apresenta o trecho do código responsável pela barra de busca. Esse componente permite aos usuários pesquisar questões usando termos específicos, com base nos campos `alternativas`, `textoBase`, `tipoProva` e `tags`. Ao pressionar "Enter", o termo digitado é utilizado para realizar a pesquisa e os resultados encontrados são exibidos na página de pesquisa.

```
const barraProcura = () => {
  const navigate = useNavigate();
  const [searchTerm, setSearchTerm] = useState('');

  const handleSearch = async (e) => {
    if (e.key === 'Enter' && searchTerm) {
      syncFirestoreToTypesense()
      try {
        const searchResults = await client.collections('Questoes').documents().search({
          q: searchTerm,
          query_by: 'alternativas,textoBase,tipoProva,tags',
        });

        // Navega para a página de pesquisa, passando os resultados no estado
        navigate('/pesquisa', { state: { results: searchResults.hits.map(hit => hit.document) } });
      } catch (error) {
        console.error('Erro na busca:', error);
      }
    }
  };
```

Figura 20. Código responsável pelas buscas da barra de pesquisas

A interface de navegação do simulado, representada pela Figura 21, permite o usuário alternar entre as questões na parte inferior com os botões “Questão anterior”, “Próxima questão” e contador de tempo de prova. Na parte superior, uma barra de progresso exibe o número total de questões, destacando em verde a questão atual. Esse indicador visual ajuda o usuário a acompanhar seu avanço e a identificar quais perguntas ainda precisam ser respondidas.

O botão “Finalizar simulado” também está disponível, possibilitando o encerramento do simulado a qualquer momento. Essa estrutura de navegação torna o fluxo do simulado mais organizado e acessível, permitindo ao usuário focar nas questões sem preocupações com o controle de progresso.

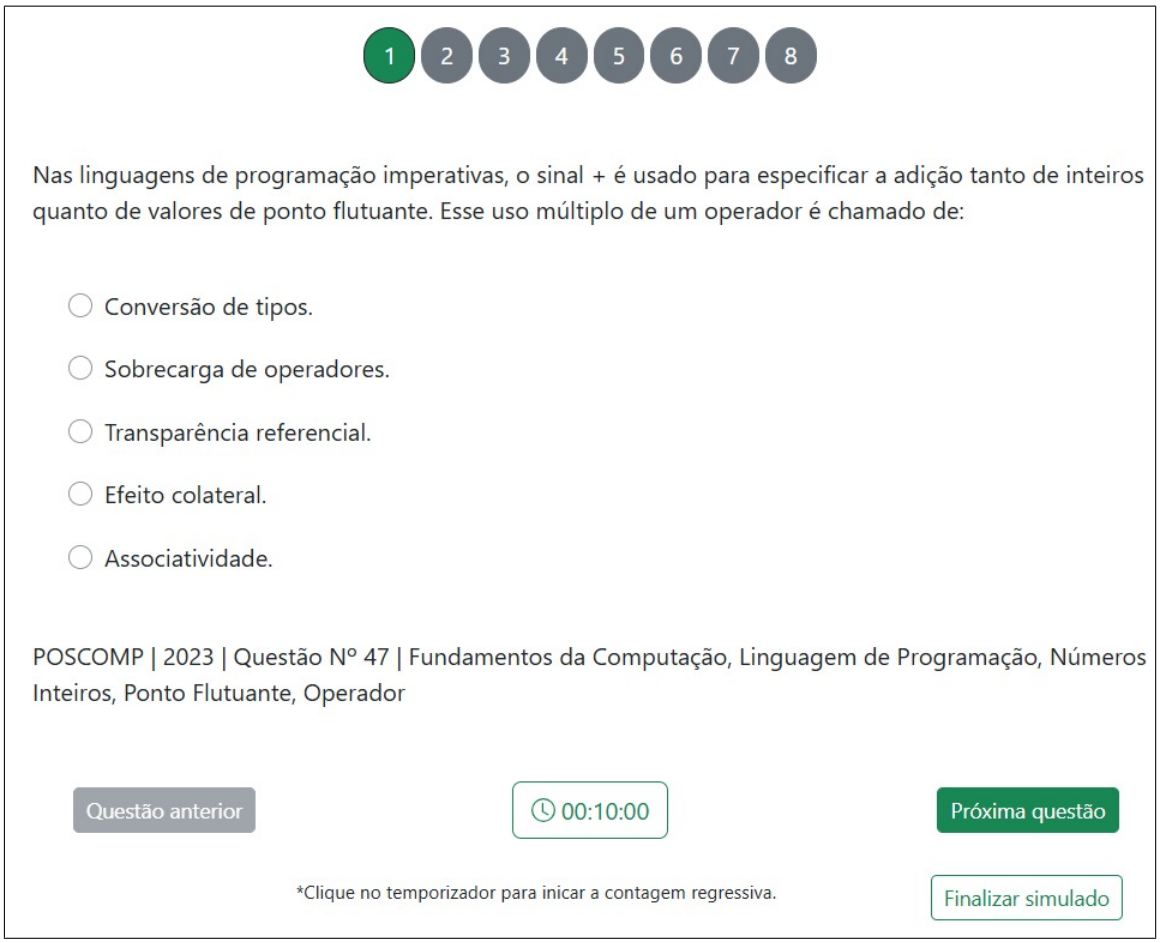
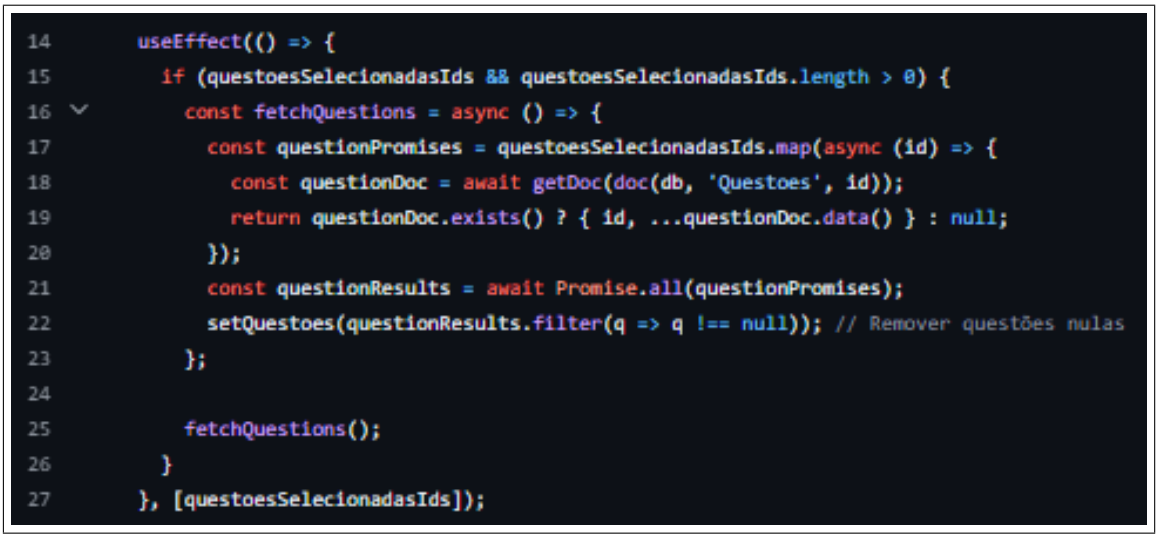


Figura 21. Interface do simulado iniciado

A função `fetchQuestions` é responsável por buscar as questões armazenadas no *Firestore* com base nos identificadores fornecidos em `questoesSelecionadasIds`. Esse processo envolve a execução de promessas assíncronas, nas quais cada questão é verificada para garantir sua existência antes de ser adicionada ao estado `questoes`. Esse filtro elimina automaticamente questões nulas, assegurando que somente dados válidos sejam processados pela aplicação, como mostrado na Figura 22, que apresenta o trecho de código para essa configuração.

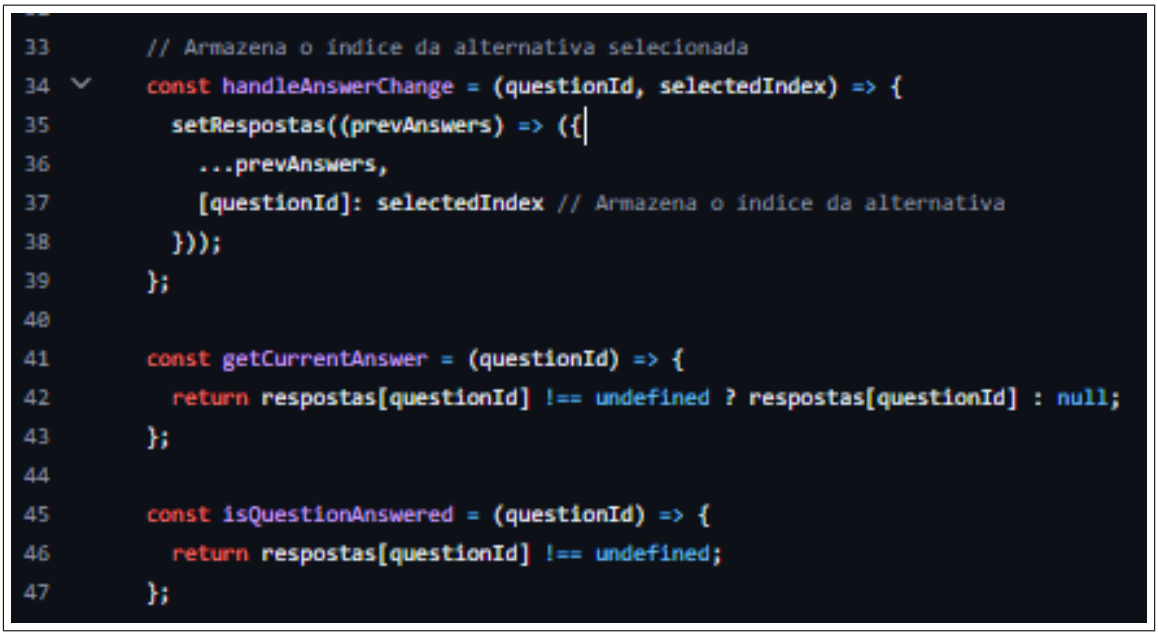


```
useEffect(() => {
  if (questoesSelecionadasIds && questoesSelecionadasIds.length > 0) {
    const fetchQuestions = async () => {
      const questionPromises = questoesSelecionadasIds.map(async (id) => {
        const questionDoc = await getDoc(doc(db, 'Questoes', id));
        return questionDoc.exists() ? { id, ...questionDoc.data() } : null;
      });
      const questionResults = await Promise.all(questionPromises);
      setQuestoes(questionResults.filter(q => q !== null)); // Remover questões nulas
    };

    fetchQuestions();
  }
}, [questoesSelecionadasIds]);
```

Figura 22. Código de busca as questões no *Firestore*

A função `handleAnswerChange`, ilustrada na Figura 23, permite registrar as respostas do usuário associando o índice da alternativa escolhida a cada questão específica. Ao atualizar dinamicamente o estado `respostas`, a aplicação mantém um registro organizado e acessível das respostas, facilitando o acesso e a análise posterior das escolhas do usuário.



```
// Armazena o índice da alternativa selecionada
const handleAnswerChange = (questionId, selectedIndex) => {
  setRespostas((prevAnswers) => ({
    ...prevAnswers,
    [questionId]: selectedIndex // Armazena o índice da alternativa
  }));
};

const getCurrentAnswer = (questionId) => {
  return respostas[questionId] !== undefined ? respostas[questionId] : null;
};

const isQuestionAnswered = (questionId) => {
  return respostas[questionId] !== undefined;
};
```

Figura 23. Código de armazenamento de respostas

A função `handleFinalizeSimulado`, demonstrada na Figura 24, realiza a consolidação dos resultados do simulado. Esse método calcula o total de questões respondidas e não respondidas, salvando as informações na coleção “Simulados” do *Firestore*. Caso o usuário esteja autenticado, o ID do simulado recém-criado é anexado ao seu perfil, preservando o histórico de tentativas. Essa funcionalidade assegura que cada simulado seja documentado de forma estruturada e associada ao usuário correspondente.

```
49  const handleFinalizeSimulado = async () => {
50    const questoesRespondidas = Object.keys(respostas).length;
51    const questoesNulas = questoes.length - questoesRespondidas;
52
53    const simuladoData = {
54      questoesSelecionadasIds: questoesSelecionadasIds,
55      questoesNulas,
56      questoesRespondidas,
57      respostas: respostas,
58      date: Timestamp.now()
59    };
60
61    // Adiciona o simulado ao banco de dados e obtém o ID do novo documento
62    const docRef = await addDoc(collection(db, 'Simulados'), simuladoData);
63
64    // Obtém o UID do usuário logado
65    const user = auth.currentUser;
66    if (user) {
67      const userUid = user.uid;
68
69      // Referência ao documento do usuário e ao campo 'simuladosId' na subcoleção
70      const userDocRef = doc(db, 'Usuarios', userUid);
71
72      try {
73        // Atualiza o documento do usuário, adicionando o ID do simulado recém-criado ao campo 'simuladosId'
74        await updateDoc(userDocRef, {
75          'simuladosId': arrayUnion(docRef.id),
76        });
77
78        // Navega para a página de resultados do simulado
79        navigate(`/resultado/${docRef.id}`);
80      } catch (error) {
81        console.error("Erro ao atualizar a subcoleção de simulados do usuário: ", error);
82      }
83    } else {
84      console.error("Nenhum usuário logado!");
85    }
86  };
```

Figura 24. Código de finalização do simulado

A Figura 25 exibe a interface de resultados do simulado e a visão geral do desempenho do usuário. A seção “Desempenho do usuário” exibe o número total de questões, a quantidade de questões respondidas, o número de acertos, erros e questões não respondidas (nulas). A interface também inclui um gráfico do tipo *doughnut* para representar visualmente a proporção de acertos, erros e questões nulas.

Além disso, a interface apresenta um gráfico radar que detalha a média de acertos por disciplina, permitindo ao usuário avaliar o desempenho em cada área do simulado. Abaixo dos gráficos, os botões numerados permitem navegar entre as questões, destacando em verde as questões corretas e em vermelho as incorretas. Ao selecionar uma questão, a resposta dada pelo usuário é exibida, juntamente com as alternativas e o indicador da resposta correta.

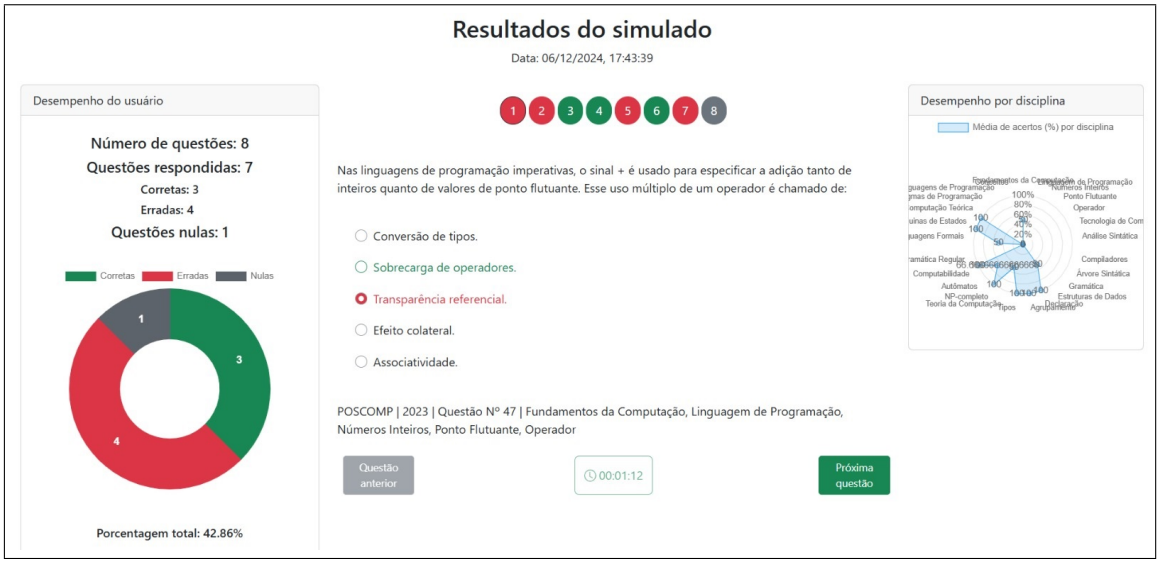


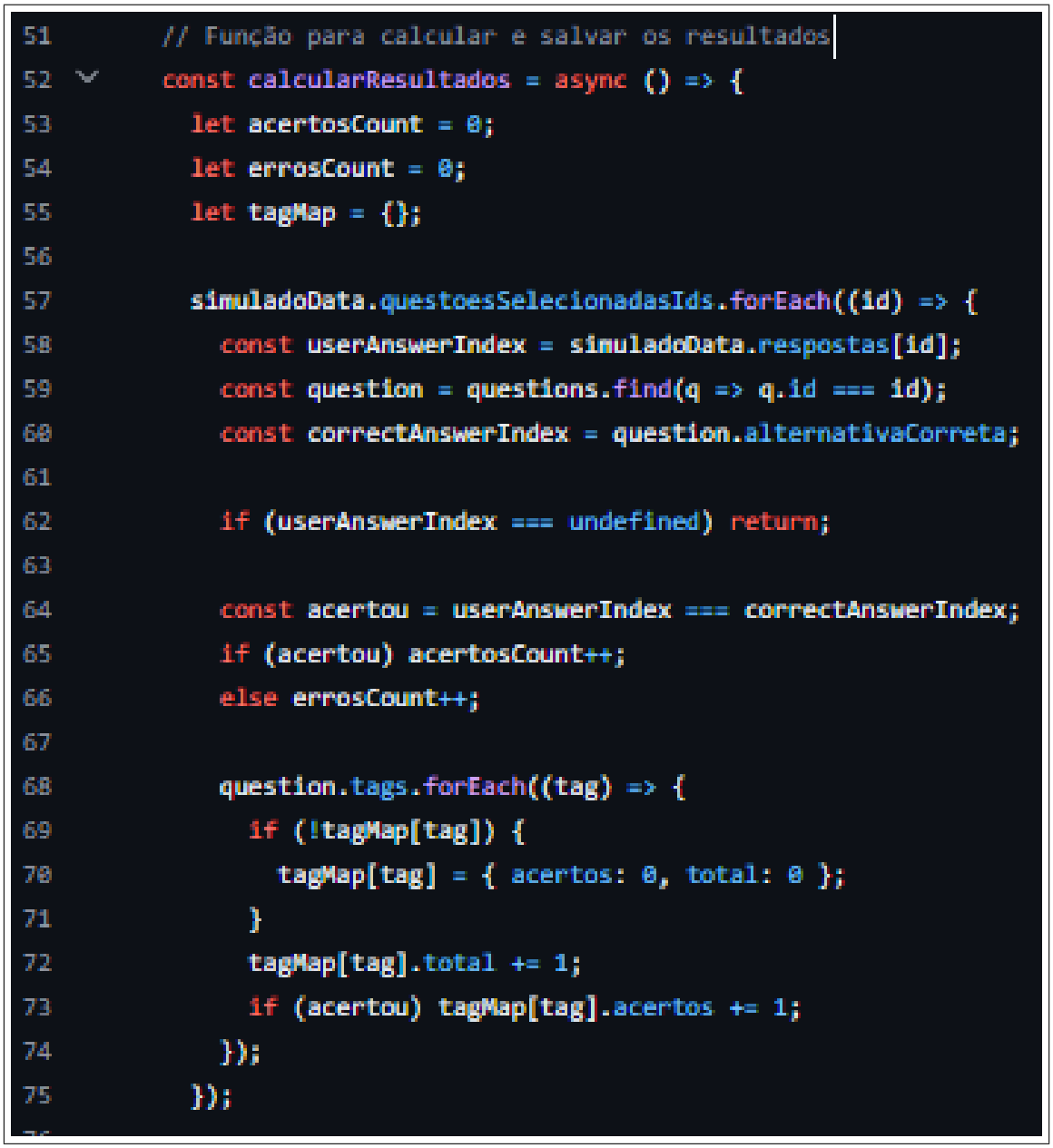
Figura 25. Interface dos resultados do simulado

A Figura 26 apresenta o código responsável pelo carregamento e organização dos dados dos simulados a partir do banco de dados *Firestore*. Esse trecho ilustra a função `fetchData`, que busca as informações dos simulados e organiza as questões com base nos seus identificadores. Essa organização permite que o sistema recupere e exiba as questões de forma eficiente, facilitando a posterior análise dos resultados do usuário.

```
22      useEffect(() => {
23        const fetchData = async () => {
24          const docRef = doc(db, 'Simulados', id);
25          const docSnap = await getDoc(docRef);
26          if (docSnap.exists()) {
27            setSimuladoData(docSnap.data());
28          } else {
29            console.log('Nenhum documento encontrado!');
30          }
31        };
32        fetchData();
33      }, [id]);
34
35      useEffect(() => {
36        if (simuladoData) {
37          const fetchQuestions = async () => {
38            const questionsArray = await Promise.all(
39              simuladoData.questoesSelecionadasIds.map(async (questionId) => {
40                const questionRef = doc(db, 'Questoes', questionId);
41                const questionSnap = await getDoc(questionRef);
42                return { id: questionId, ...questionSnap.data() };
43              })
44            );
45            setQuestions(questionsArray);
46          };
47          fetchQuestions();
48        }
49      }, [simuladoData]);
```

Figura 26. Código de carregamento e organização dos dados

Na Figura 27, é exibido o código que realiza o cálculo dos resultados do simulado e o mapeamento das *tags* associadas a cada questão. A função `calcularResultados` compara as respostas do usuário com as respostas corretas e contabiliza o número de acertos, erros e médias de desempenho por *tag*. O mapeamento das *tags* permite uma análise detalhada do desempenho do usuário em diferentes áreas de conhecimento, auxiliando na identificação de pontos fortes e oportunidades de melhoria.



```
51      // Função para calcular e salvar os resultados
52      const calcularResultados = async () => {
53        let acertosCount = 0;
54        let errosCount = 0;
55        let tagMap = {};
56
57        simuladoData.questoesSelecionadasIds.forEach((id) => {
58          const userAnswerIndex = simuladoData.respostas[id];
59          const question = questions.find(q => q.id === id);
60          const correctAnswerIndex = question.alternativaCorreta;
61
62          if (userAnswerIndex === undefined) return;
63
64          const acertou = userAnswerIndex === correctAnswerIndex;
65          if (acertou) acertosCount++;
66          else errosCount++;
67
68          question.tags.forEach((tag) => {
69            if (!tagMap[tag]) {
70              tagMap[tag] = { acertos: 0, total: 0 };
71            }
72            tagMap[tag].total += 1;
73            if (acertou) tagMap[tag].acertos += 1;
74          });
75        });
```

Figura 27. Código do cálculo dos resultados e mapeamento de *tags*

A Figura 28 mostra o código de configuração e exibição dos gráficos usados para visualizar o desempenho do usuário no simulado. São configurados dois tipos de gráficos: o gráfico *doughnut*, que apresenta a proporção de acertos e erros, e o gráfico radar, que exibe o desempenho por *tag*. Essas visualizações gráficas proporcionam um *feedback* visual objetivo, permitindo ao usuário compreender facilmente seu desempenho geral e por área de conhecimento.

```
155       const finalData = {
156         labels: data.map((item) => item.label),
157         datasets: [
158           {
159             data: data.map((item) => Math.round(item.value)),
160             backgroundColor: data.map((item) => item.color),
161             borderColor: data.map((item) => item.color),
162             borderWidth: 1,
163             dataVisibility: new Array(data.length).fill(true),
164           },
165         ],
166       };
167
168       // Configuração do radar chart
169       const radarData = {
170         labels: Object.keys(mediaTags),
171         datasets: [
172           {
173             label: 'Média de acertos (%) por disciplina',
174             data: Object.values(mediaTags),
175             backgroundColor: 'rgba(54, 162, 235, 0.2)',
176             borderColor: 'rgba(54, 162, 235, 1)',
177             borderWidth: 1,
178           },
179         ],
180       };
```

Figura 28. Código de configuração e exibição dos gráficos

Em síntese, o módulo de pesquisa e simulado oferece uma experiência prática e orientada ao usuário, permitindo a realização de simulados personalizados, o registro de respostas e a análise de resultados de forma clara e visual. A funcionalidade dos gráficos proporciona *feedback* imediato, destacando o desempenho em cada área e guiando o usuário no aprimoramento dos conhecimentos necessários para alcançar melhores resultados.

4.4. Módulo de Relatórios e Desempenho

A Figura 29 apresenta a página referente à conta do usuário, que atua como um *dashboard*. Esta interface é projetada para fornecer um panorama detalhado das informações e estatísticas do usuário. O painel inclui seções para atualização de dados pessoais como *e-mail*, senha e foto de perfil. Além disso, apresenta estatísticas totais das questões que o usuário já realizou, incluindo o total de questões respondidas, corretas e erradas. Também é exibida uma média percentual de acertos por disciplina, permitindo ao usuário avaliar seu desempenho geral.

Além dessas informações, o *dashboard* disponibiliza um histórico completo dos simulados realizados. O usuário pode acessar antigos simulados e visualizar um gráfico de linha que mostra o percentual de acertos em relação à data do simulado. Essa funcionalidade é útil para monitorar o progresso do usuário ao longo do tempo, identificar áreas de melhoria e acompanhar a evolução do desempenho. Com essas ferramentas, o usuário tem uma visão clara de seu desempenho e pode tomar decisões informadas para otimizar seu estudo e preparação.

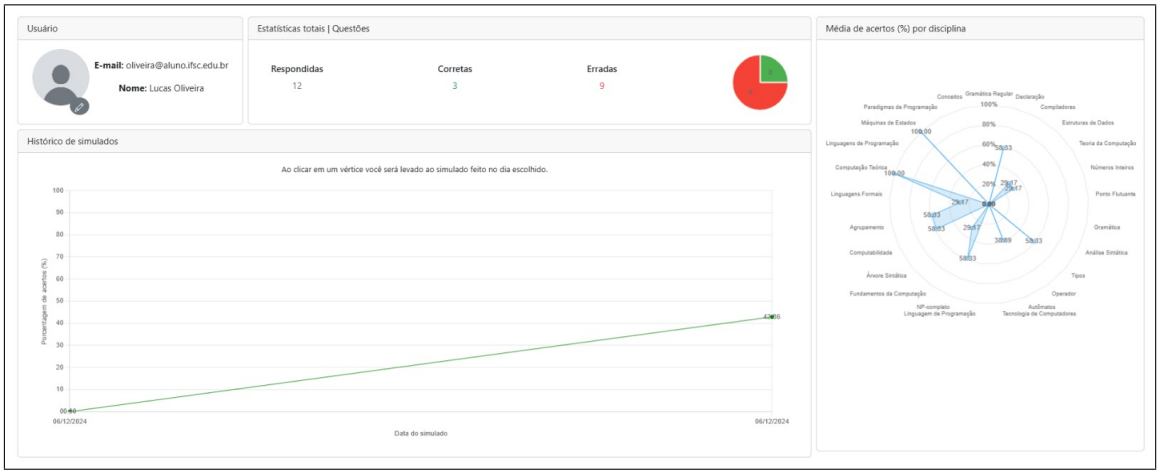
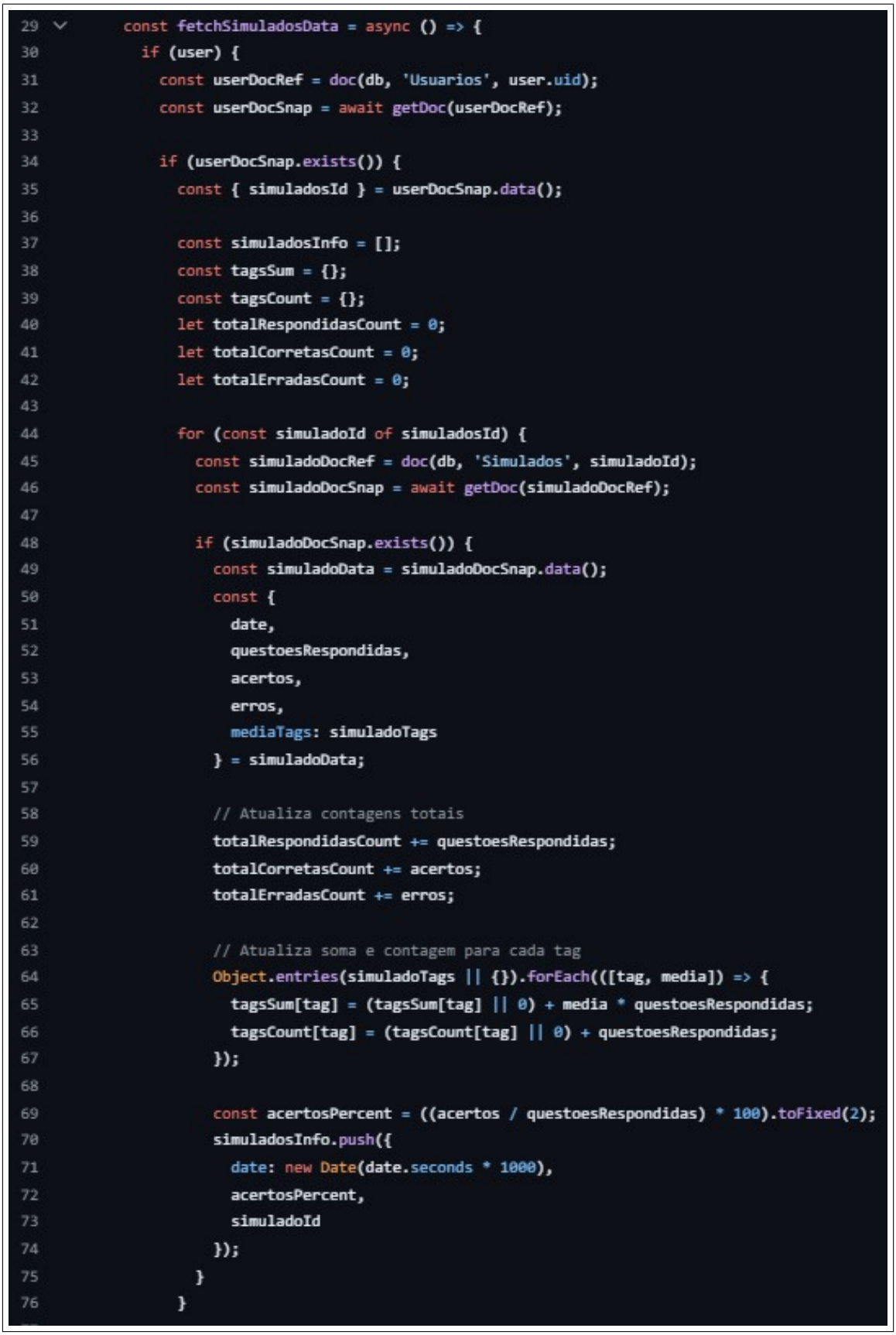


Figura 29. Interface das estatísticas gerais do usuário

A Figura 30 apresenta o código responsável pelo carregamento e organização dos dados dos simulados no banco de dados *Firestore*. Este trecho ilustra a função `fetchSimuladosData`, que busca as informações detalhadas dos simulados associados ao usuário e organiza os resultados com base nos identificadores dos simulados. A função processa as informações dos simulados, incluindo a contagem de questões respondidas, acertos e erros, além da média das *tags* associadas a cada simulado. Isso permite que o sistema recupere e exiba os dados dos simulados de maneira eficiente, facilitando a análise dos resultados pelos usuários.



```
const fetchSimuladosData = async () => {
  if (user) {
    const userDocRef = doc(db, 'Usuarios', user.uid);
    const userDocSnap = await getDoc(userDocRef);

    if (userDocSnap.exists()) {
      const { simuladosId } = userDocSnap.data();

      const simuladosInfo = [];
      const tagsSum = {};
      const tagsCount = {};
      let totalRespondidasCount = 0;
      let totalCorretasCount = 0;
      let totalErradasCount = 0;

      for (const simuladoId of simuladosId) {
        const simuladoDocRef = doc(db, 'Simulados', simuladoId);
        const simuladoDocSnap = await getDoc(simuladoDocRef);

        if (simuladoDocSnap.exists()) {
          const simuladoData = simuladoDocSnap.data();
          const {
            date,
            questoesRespondidas,
            acertos,
            erros,
            mediaTags: simuladoTags
          } = simuladoData;

          // Atualiza contagens totais
          totalRespondidasCount += questoesRespondidas;
          totalCorretasCount += acertos;
          totalErradasCount += erros;

          // Atualiza soma e contagem para cada tag
          Object.entries(simuladoTags || {}).forEach(([tag, media]) => {
            tagsSum[tag] = (tagsSum[tag] || 0) + media * questoesRespondidas;
            tagsCount[tag] = (tagsCount[tag] || 0) + questoesRespondidas;
          });

          const acertosPercent = ((acertos / questoesRespondidas) * 100).toFixed(2);
          simuladosInfo.push({
            date: new Date(date.seconds * 1000),
            acertosPercent,
            simuladoId
          });
        }
      }
```

Figura 30. Código de carregamento e organização dos dados do usuário

```
149       const radarData = {
150         labels: Object.keys(mediaTags),
151         datasets: [
152           {
153             label: 'Média de Acertos (%) por Disciplina',
154             data: Object.values(mediaTags),
155             backgroundColor: 'rgba(54, 162, 235, 0.2)',
156             borderColor: 'rgba(54, 162, 235, 1)',
157             borderWidth: 1,
158           },
159         ],
160       };
161
162       const radarOptions = {
163         scales: {
164           r: {
165             angleLines: { display: false },
166             suggestedMin: 0,
167             suggestedMax: 100,
168             ticks: {
169               stepSize: 20,
170               callback: (value) => `${value}%`,
171             },
172           },
173         },
174         plugins: { legend: { display: false } },
175         responsive: true,
176       };
177
178       // Configuração Pie Cahrt
179       const pieData = {
180         labels: ['Acertos', 'Erros'],
181         datasets: [
182           {
183             data: [totalCorretas, totalErradas],
184             backgroundColor: ['#4CAF50', '#F44336'],
185             hoverBackgroundColor: ['#66BB6A', '#EF5350'],
186           },
187         ],
188       };
189
190       const pieOptions = {
191         responsive: true, // Gráfico responsivo
192         maintainAspectRatio: false, // Permite que o gráfico preencha o contêiner
193         plugins:
194         {
195           legend:
196             { display: false }
197         }
```

Figura 31. Código de configuração dos gráficos do *dashboard*

A Figura 31 mostra o código de configuração e a exibição dos gráficos usados para analisar o desempenho do usuário nos simulados. Três tipos de gráficos são utilizados para oferecer uma visão abrangente: o gráfico de linha, que visualiza a evolução do desempenho ao longo do tempo usando os dados armazenados no *array* `simuladosData`. Cada ponto na linha verde representa a média percentual de acertos de um simulado específico, com os rótulos no eixo X correspondendo às datas dos simulados. Ao clicar em um ponto, o usuário é redirecionado para a página do simulado correspondente.

O gráfico radar exibe o desempenho do usuário por disciplina, com cada disciplina representada como um ponto no radar. As médias de acertos percentual são mapeadas usando o objeto `mediaTags`, com áreas de conhecimento visualmente distintas através de fundos coloridos, facilitando a análise. Cada rótulo no gráfico radar corresponde a uma disciplina específica. O gráfico de pizza mostra a distribuição dos acertos e erros totais, utilizando verde para acertos e vermelho para erros, com as proporções exibidas como porcentagens para uma análise visual clara do desempenho geral do usuário.

5. Análise Crítica

Para análise crítica, o teste de usabilidade SUS foi aplicado com um questionário de 10 perguntas padrão SUS, abordando aspectos como facilidade de aprendizado, simplicidade da interface e integração das funcionalidades.O questionário é composto por 10 perguntas, cada uma com respostas avaliadas em uma escala *Likert* de 1 a 5. Para calcular a pontuação do SUS, cada resposta recebe um valor específico ajustado de acordo com a fórmula estabelecida, conforme apresentado na Figura 32.

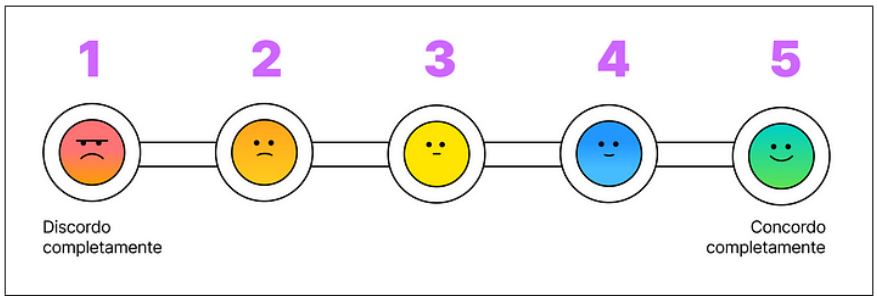


Figura 32. Escala *Likert* do teste SUS Fonte: Brasil (2024)

As 10 perguntas aplicadas foram:

- Eu acho que gostaria de usar esse sistema com frequência.
- Eu acho o sistema desnecessariamente complexo.
- Eu achei o sistema fácil de usar.
- Eu acho que precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o sistema.
- Eu acho que as várias funções do sistema estão muito bem integradas.
- Eu acho que o sistema apresenta muita inconsistência.
- Eu imagino que as pessoas aprenderão como usar esse sistema rapidamente.
- Eu achei o sistema atrapalhado de usar.
- Eu me senti confiante ao usar o sistema.
- Eu precisei aprender várias coisas novas antes de conseguir usar o sistema.

5.1. Cálculo do Teste SUS

A Figura 33 ilustra a pontuação de aceitabilidade do SUS, que varia de 0 a 100. Essa pontuação reflete a percepção dos usuários em relação à facilidade de uso e à eficácia do sistema avaliado.

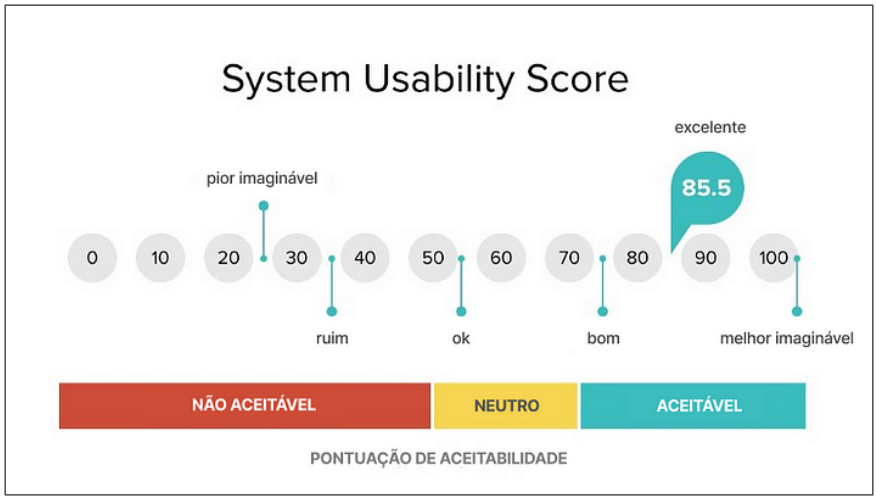


Figura 33. Pontuação de aceitabilidade do SUS Fonte: Brasil (2024)

O cálculo do *System Usability Scale* envolve as seguintes etapas (Brasil, 2024):

1. **Identificar as perguntas ímpares e pares:**
    - Perguntas ímpares (1, 3, 5, 7, 9): subtraia 1 da resposta.
    - Perguntas pares (2, 4, 6, 8, 10): subtraia o valor da resposta de 5.
2. **Somar os valores ajustados:**
    - Após o ajuste das respostas (ímpares e pares), some os valores.
3. **Multiplicar o total por 2,5:**
    - O valor total ajustado deve ser multiplicado por 2,5 para obter a pontuação final do SUS, que varia de 0 a 100.

O questionário de teste SUS contou com a participação de 18 respondentes, divididos entre professores e alunos. Entre os participantes, 4 eram professores, representando 22,2% do total, enquanto 14 eram alunos, correspondendo a 77,8%. Essa distribuição demonstra um maior engajamento por parte dos estudantes na avaliação da usabilidade, mas garante também uma perspectiva significativa dos docentes, promovendo uma visão mais ampla e diversificada sobre a experiência de uso.

A Figura 34 mostra que a média do SUS foi de 86,9, o que indica um alto nível de usabilidade do sistema avaliado, posicionando-o na faixa considerada “excelente” de acordo com os critérios de interpretação da escala. Esse valor sugere que os usuários consideraram o sistema intuitivo, eficiente e agradável de usar, com poucas dificuldades significativas.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Perfil	Pergunta 1	Pergunta 2	Pergunta 3	Pergunta 4	Pergunta 5	Pergunta 6	Pergunta 7	Pergunta 8	Pergunta 9	Pergunta 10	Pontuação Bruta do SUS	Pontuação Final do SUS
2	Aluno(a)	3	2	4	1	3	1	4	2	4	1	31	77,5
3	Aluno(a)	3	1	5	1	4	3	5	1	4	1	34	85
4	Aluno(a)	4	1	5	1	5	1	5	1	5	1	39	97,5
5	Aluno(a)	5	4	4	3	5	1	5	1	5	1	34	85
6	Aluno(a)	4	2	4	1	4	3	4	1	4	1	32	80
7	Aluno(a)	5	1	4	1	5	3	5	2	1	1	32	80
8	Aluno(a)	3	4	4	1	4	2	4	3	3	1	27	67,5
9	Aluno(a)	5	1	5	1	1	1	5	1	5	1	36	90
10	Aluno(a)	5	1	5	1	5	1	4	1	5	1	39	97,5
11	Aluno(a)	4	1	5	1	4	1	5	1	4	1	37	92,5
12	Aluno(a)	4	4	4	4	4	4	4	4	4	4	20	50
13	Aluno(a)	4	1	5	1	4	1	5	1	5	1	38	95
14	Aluno(a)	4	1	5	1	5	2	5	1	5	1	38	95
15	Aluno(a)	5	1	5	1	5	1	5	1	5	1	40	100
16	Professor(a)	4	1	5	1	1	1	5	1	5	1	35	87,5
17	Professor(a)	5	1	5	1	4	1	5	1	5	1	39	97,5
18	Professor(a)	5	5	5	1	4	1	5	1	5	1	35	87,5
19	Professor(a)	5	1	5	1	5	1	5	1	5	1	40	100
20												Média:	86,9

Figura 34. Cálculo do teste SUS

A pontuação reflete que a maioria das funcionalidades do sistema atende às expectativas dos usuários, proporcionando uma experiência positiva e satisfatória. Vale ressaltar que uma pontuação superior a 80 é geralmente associada a produtos com alta aceitação e facilidade de uso, fatores que podem influenciar diretamente na adoção e fidelidade dos usuários ao sistema.

O teste de usabilidade SUS destacou a principal contribuição do sistema: uma interface intuitiva e bem integrada que facilita o uso e atende às necessidades dos usuários, com uma pontuação elevada que reflete alta satisfação. No entanto, limitações foram observadas, como a complexidade inicial no cadastro de questões e a ausência de recursos avançados para personalização de simulados, indicando áreas que podem ser aprimoradas para atender melhor a casos específicos e ampliar a usabilidade do sistema.

6. Considerações Finais

Este trabalho desenvolveu uma plataforma centralizada para pesquisa e prática de questões do Enade e POSCOMP, atendendo à necessidade de estudantes e profissionais por uma ferramenta eficiente e integrada. O sistema modular facilita o acesso às questões, a criação de simulados personalizados e o acompanhamento de desempenho, promovendo um estudo mais direcionado e dinâmico.

Um dos principais desafios foi o cadastro das questões, exigindo padronização e organização para garantir a qualidade e funcionalidade do sistema. Apesar disso, a plataforma alcançou seu objetivo, proporcionando uma solução prática e relevante para os usuários.

Como trabalhos futuros, sugere-se a implementação de técnicas de *machine learning* para análise preditiva do desempenho dos usuários, permitindo a criação de simulados personalizados com base em áreas de dificuldade identificadas. Além disso, a integração de *Application Programming Interface*(API) para automação do cadastro de questões e a utilização de *Optical Character Recognition* (OCR) para a extração automática de questões de documentos PDF podem otimizar o processo de catalogação. A adoção de microsserviços para modularizar funcionalidades como autenticação, geração de relatórios e busca de questões também pode melhorar a escalabilidade e manutenção da plataforma. Por fim, a incorporação de *blockchain* para garantir a integridade e autenticidade das questões pode ser uma solução inovadora para aumentar a confiabilidade do sistema.

Referências

BRASIL (2004). Presidência da república casa civil subchefia para assuntos jurídicos. Disponível em: https://www.sbc.org.br/sobre-asbc/. Acesso em: 25 abr. 2024.

Brasil, U. D. (2024). Guia atualizado de como utilizar a escala SUS (system usability scale) no seu produto. https://encr.pw/guia-atualizado-de-como-utilizar-a-escala-sus. Acesso em: 05 dez. 2024.

BROOKE, J. (1986). SUS: A "quick and dirty" usability scale. In Jordan, P. W., Thomas, B., Weerdmeester, B. A., e McClelland, A. L., editors, *Usability Evaluation in Industry*. Taylor and Francis, London. Disponível em: https://www.researchgate.net/publication/319394819_SUS_--_a_quick_and_dirty_usability_scale. Acesso em: 28 out. 2024.

BROOKE, J. B. (2013). SUS - a retrospective. *Journal of Usability Studies*, 8(2):29–40. Disponível em: https://www.researchgate.net/publication/285811057_SUS_a_retrospective. Acesso em: 10 nov. 2024.

Callegari, B. V., Oliveira, K. A. D., Karsburg, V. G., e Albiero, F. W. (2020). Desenvolvimento de um aplicativo para a análise de questões do POSCOMP na área de matemática e suas recorrências. https://repositorio.ifsc.edu.br/handle/123456789/2454. Acesso em: 04 mar. 2024.

CRUZ, F. (2013). Scrum e PMBOK unidos no gerenciamento de projetos. Disponível em: https://www.google.com.br/books/edition/Scrum_e_PMBOK_unidos_no_Gerenciamento_de/SJA37S2QGR0C?hl=pt-BR&gbpv=0. Acesso em: 07 abr. 2024.

DAES (2017). Nota técnica CGCQES n. 12/2017 - cálculo da nota final do Enade. Disponível em: https://download.inep.gov.br/educacao_superior/enade/notas_tecnicas/2017/Nota_Tecnica_CGCQES_n12_2017_Calculo_da_nota_final_do_Enade.pdf. Acesso em: 25 abr. 2024.

FIGMA (2024). Figma. Disponível em: https://www.figma.com/pt-br/. Acesso em: 10 nov. 2024.

FIREBASE (2024). Google firebase team. Disponível em: https://firebase.google.com/?hl=pt-br. Acesso em: 07 maio 2024.

FUNDATEC (2023). Exame POSCOMP 2023 tem novidades: inscrições foram prorrogadas e provas serão on-line. Disponível em: https://www2.fundatec.org.br/2023/08/02/exame-poscomp-2023/. Acesso em: 14 abr. 2024.

INEP (2024). Exame nacional de desempenho dos estudantes. Disponível em: https://www.gov.br/inep/pt-br/areas-de-atuacao/avaliacao-e-exames-educacionais/enade. Acesso em: 14 abr. 2024.

MEC (2019). O que é o conceito preliminar de curso? Disponível em: https://www.gov.br/mec/pt-br. Acesso em: 26 abr. 2024.

Queiroz, J. P. A. e Antonello, R. (2019). Plataforma web para manutenção de um banco de questões on-line. https://publicacoes.ifc.edu.br/index.php/micti/article/view/925. Acesso em: 07 out. 2024.

REACT (2024). React documentation. Disponível em: https://pt-br.react.dev/blog/2023/03/16/introducing-react-dev. Acesso em: 01 nov. 2024.

SBC (2023). Média do desvio padrão. Disponível em: https://www.sbc.org.br/documentos-da-sbc?task=download.send&id=1296&catid=185&m=0. Acesso em: 02 maio 2024.

SBC (2024). Exame nacional para ingresso na pós-graduação em computação. Disponível em: https://www.sbc.org.br/educacao/poscomp. Acesso em: 14 abr. 2024.

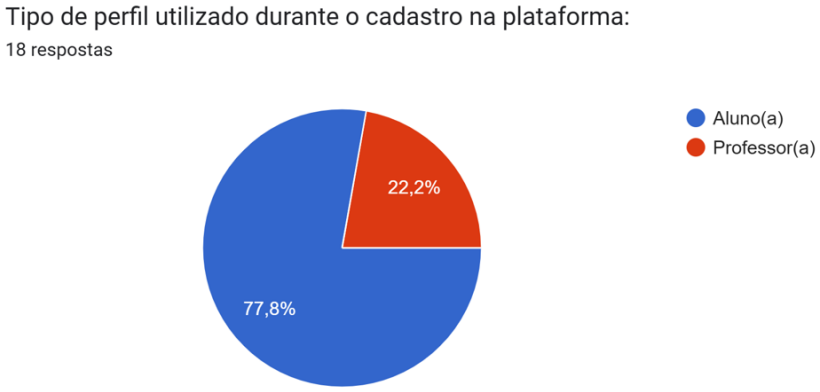
Tiago, C. R., Junior, W. F. C., e Neto, W. C. B. (2020). Plataforma para auxílio na preparação de estudantes para as avaliações do ENADE e POSCOMP. Disponível em: https://repositorio.ifsc.edu.br/handle/123456789/2458. Acesso em: 04 mar. 2024.

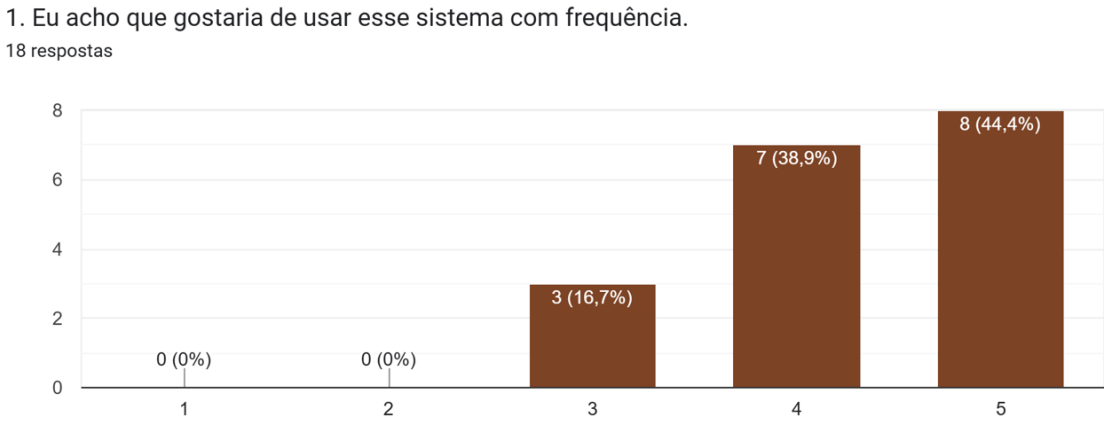
TRELLO (2024). Trello. Disponível em: https://trello.com/. Acesso em: 07 maio 2024.

Zotti, M. H., De Sá, M. H. I., e Franco (2016). Sistema de simulados com recomendações baseado no desempenho do usuário. https://encr.pw/documentos-poa-biblioteca-clovis-vergara-marques. Acesso em: 07 jun. 2024.

A. Respostas do Teste SUS

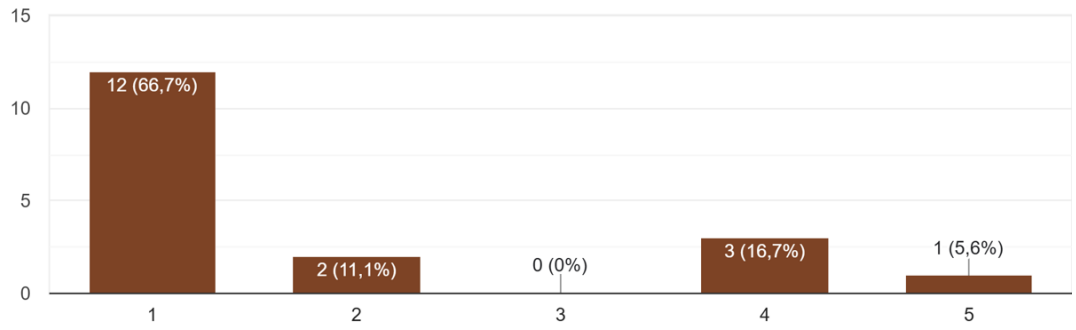
O questionário do SUS contou com as respostas dos participantes para as 10 perguntas relacionadas à usabilidade do sistema. A seguir, apresentamos as imagens contendo os gráficos e as informações detalhadas das respostas obtidas.





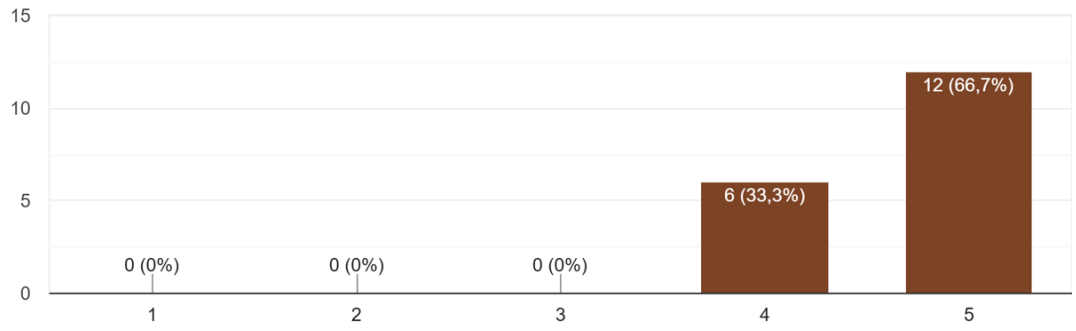
2. Eu acho o sistema desnecessariamente complexo.

18 respostas



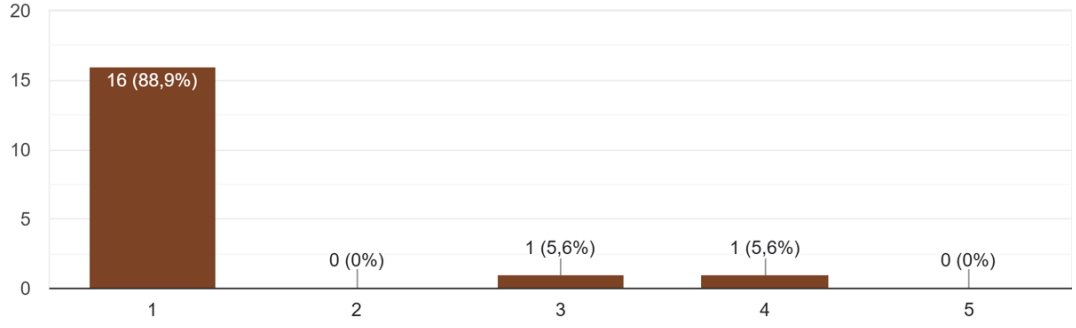
3. Eu achei o sistema fácil de usar.

18 respostas



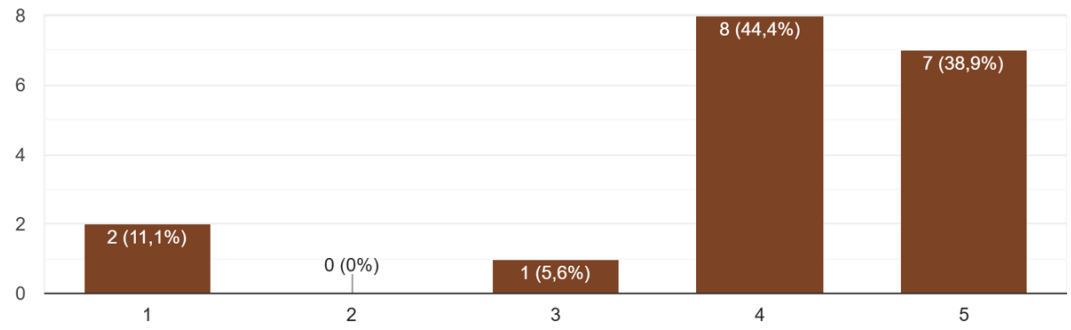
4. Eu acho que precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o sistema.

18 respostas



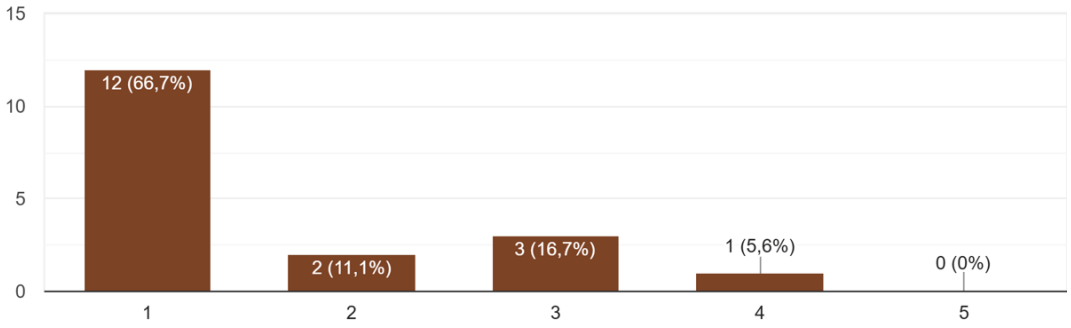
5. Eu acho que as várias funções do sistema estão muito bem integradas.

18 respostas



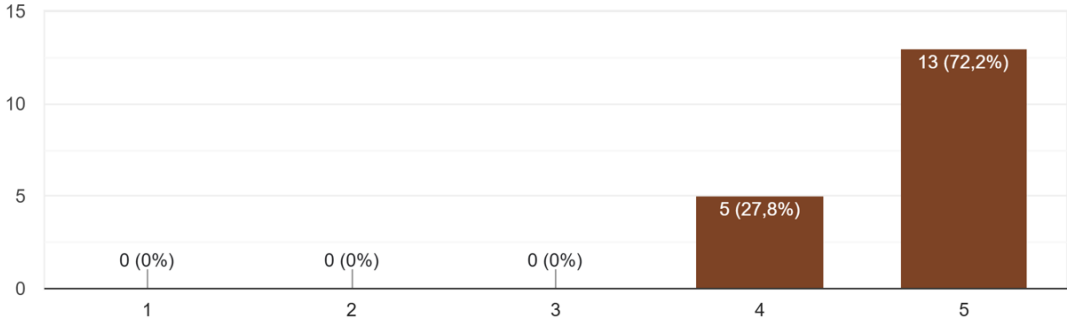
6. Eu acho que o sistema apresenta muita inconsistência.

18 respostas



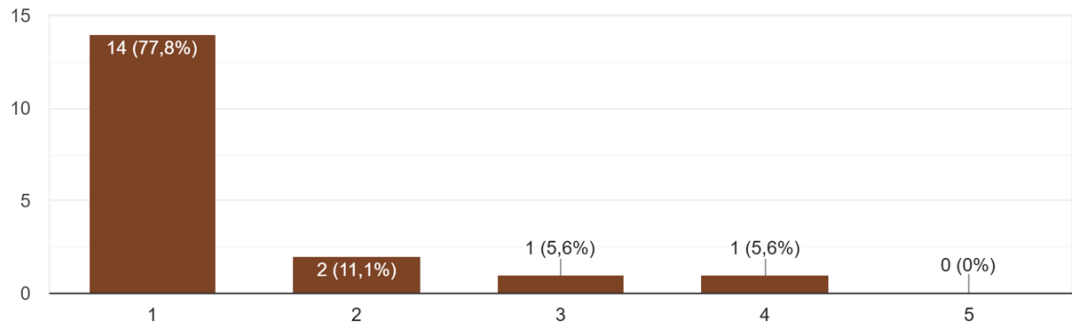
7. Eu imagino que as pessoas aprenderão como usar esse sistema rapidamente.

18 respostas



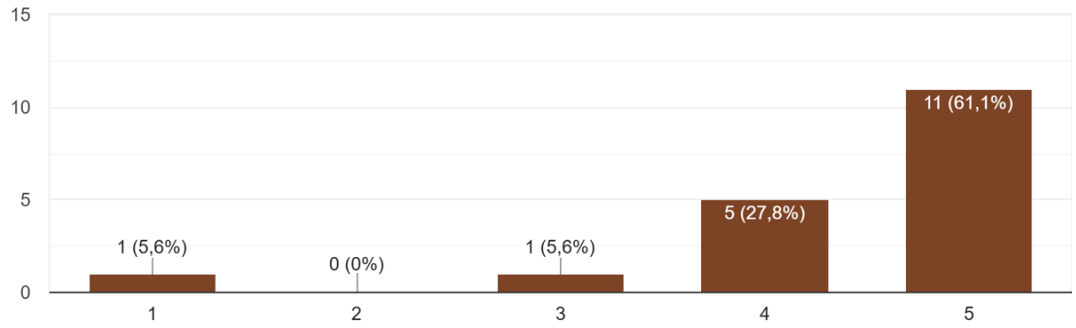
8. Eu achei o sistema atrapalhado de usar.

18 respostas



9. Eu me senti confiante ao usar o sistema.

18 respostas



10. Eu precisei aprender várias coisas novas antes de conseguir usar o sistema.

18 respostas