

INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS
DEPARTAMENTO DE COMPUTAÇÃO

LEANDRO LIBÉRIO MACHADO DA SILVA

Orientador: Prof. Dr. Reinaldo Silva Fortes

Coorientador: Prof. Dr. Saul Emanuel Delabrida Silva

**OPCODERS JUDGE:
APRIMORANDO O CORRETOR AUTOMÁTICO DE EXERCÍCIOS DE
PROGRAMAÇÃO COM BASE EM TESTES DE USABILIDADE**

Ouro Preto, MG
2023

INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS
DEPARTAMENTO DE COMPUTAÇÃO

LEANDRO LIBÉRIO MACHADO DA SILVA

**OPCODERS JUDGE:
APRIMORANDO O CORRETOR AUTOMÁTICO DE EXERCÍCIOS DE
PROGRAMAÇÃO COM BASE EM TESTES DE USABILIDADE**

Monografia apresentada ao Curso de Ciência da Computação da como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Reinaldo Silva Fortes

Coorientador: Prof. Dr. Saul Emanuel Delabrida Silva

Ouro Preto, MG
2023

Dedico esta monografia à minha família, que sempre me apoiou em minha vida acadêmica e, em especial, à minha mãe, principal pilar em toda a minha jornada.

Agradecimentos

Gostaria de expressar meus sinceros agradecimentos a todas as pessoas que estiveram ao meu lado durante esta jornada de construção e aprendizado. Agradeço, em especial, à minha querida família, cujo apoio incondicional foi o alicerce que me sustentou em todos os momentos. Em especial, à minha mãe Matilde, cuja dedicação, amor e incentivo foram sempre a luz que guiou meus passos, agradeço profundamente.

Não posso deixar de mencionar os professores que desempenharam um papel vital no desenvolvimento deste trabalho. Agradeço aos professores Reinaldo e Saul, cuja orientação e colaboração foram fundamentais para guiar meu entendimento nesta monografia. À Vivyann, que não apenas compartilhou seu conhecimento, mas também serviu de base para que este trabalho possa ter sido realizado.

Minha gratidão também se estende à minha querida Universidade Federal de Ouro Preto e aos outros professores do DECOM, que me proporcionaram um ambiente rico em conhecimento e oportunidades de crescimento.

Por fim, queria deixar um agradecimento também especial, à Luciana Rocha, minha psicóloga, que mesmo aparecendo em minha trajetória no final desta caminhada, teve uma papel fundamental para que ela pudesse ser finalizada.

Resumo

Nos dias atuais, o ensino de ciência da computação enfrenta desafios significativos devido à crescente complexidade tecnológica e às demandas em constante evolução da indústria. Nesse cenário, as ferramentas auxiliares, como corretores de código-fonte (normalmente denominados *juízes online*), desempenham um papel crucial ao oferecer suporte essencial para estudantes, educadores e profissionais da área. Na Universidade Federal de Ouro Preto (UFOP), é utilizado o *opCoders Judge* para auxiliar o ensino de programação introdutória. Essa plataforma web é eficiente em seu objetivo, contudo, como qualquer aplicativo, pode passar por melhorias. Pensando nisso, Cedraz (2023) realizou um estudo de usabilidade com alunos da Ciência da Computação, identificando problemas nas interfaces da plataforma e criou um protótipo como solução. O presente trabalho monográfico é uma continuação do trabalho de Cedraz (2023), na busca para realizar melhorias no *opCoders Judge*, visando deixá-lo mais próximo de seu estado ótimo. Para alcançar tal objetivo foi feito um novo estudo de usabilidade, com a participação voluntária de estudantes da disciplina Programação de Computadores I, público alvo atual da plataforma. Com os resultados deste estudo, foram propostas mudanças no protótipo feito por Cedraz (2023), buscando melhorar pontos que vão além da usabilidade. Por fim deu-se início à implementação da nova plataforma, utilizando ferramentas mais modernas do que as utilizadas no desenvolvimento atual do *opCoders Judge*. A implementação contou com o desenvolvimento de *frontend*, *backend* e remodelagem de banco de dados. Como resultado do estudo de usabilidade, foi percebido que os estudantes de Programação de Computadores I tem mais dificuldade do que os alunos do curso de Ciência da Computação. Acredita-se que protótipo tornou-se mais moderno e atrativo, futuramente recomenda-se realizar um estudo de usabilidade com a nova interface, para confirmar esta suspeita. Já a implementação foi feita iterativamente, começando com as funcionalidades fundamentais para compor a estrutura da plataforma. Por restrições de tempo e escopo, neste estágio do trabalho focamos no desenvolvimento das funcionalidades de *frontend* voltadas para a visão dos alunos e algumas funcionalidades de *backend* envolvidas, para funcionalidades administrativas pode-se utilizar recursos do Django.

Palavras-chave: Corretor de código-fonte. Usabilidade. Frontend. Backend. Banco de Dados.

Abstract

Currently, computer science education faces significant challenges due to the growing technological complexity and the ever-evolving demands of the industry. In this scenario, auxiliary tools, such as source code evaluators (commonly called online judges), play a crucial role by providing essential support for students, educators, and professionals in the field. At the *Universidade Federal de Ouro Preto* (UFOP), the *opCoders Judge* is utilized to assist in introductory programming education. While this web platform is efficient in its purpose, it can undergo improvements like any application. With this in mind, [Cedraz \(2023\)](#) conducted a usability study with students in the Computer Science course, identifying issues in the platform's interfaces and creating a prototype as a solution. This current work is a continuation of [Cedraz \(2023\)](#)'s research, aiming to improve *opCoders Judge*, bringing it closer to its optimal state. To achieve this goal, a new usability study was conducted with the voluntary participation of students in the Computer Programming I discipline, the current target audience of the platform. Based on the results of this study, changes were proposed to the prototype developed by [Cedraz \(2023\)](#), seeking to enhance aspects beyond usability. Finally, the implementation of the new platform began, using more modern tools than those employed in the current *opCoders Judge* development. The implementation included the development of frontend, backend, and a database redesign. As a result of the usability study, it was observed that students in the Computer Programming I discipline face more difficulties than students in the Computer Science course. The prototype is believed to have become more modern and attractive; future work would involve conducting a usability study with the new interface to confirm this hypothesis. The implementation was carried out iteratively, starting with fundamental features to compose the platform's structure. Due to time and scope constraints at this stage of the work, the focus was on developing frontend features aimed at the student's perspective and some related backend functionalities; for administrative features, resources from Django can be employed.

Keywords: Source code evaluators. Usability. Frontend. Backend. Database.

Lista de Ilustrações

Figura 1.1 – Diagrama da Metodologia.	4
Figura 3.1 – Resultado do teste SUS geral.	19
Figura 3.2 – Resultado do teste SUS para questões Positivas.	20
Figura 3.3 – Resultado do Teste SUS para questões Negativas.	21
Figura 3.4 – Resultado do SAM sobre Satisfação.	22
Figura 3.5 – Resultado do SAM sobre Motivação.	23
Figura 3.6 – Resultado do SAM sobre Sentimento de Controle.	24
Figura 3.7 – Resultado do teste de Perfil.	25
Figura 3.8 – Resultado geral do teste SUS - COM-T2.	27
Figura 3.9 – Resultado geral do teste SUS - MEC-T2.	28
Figura 3.10–Resultado geral do teste SUS - CAU-T2.	28
Figura 3.11–Resultado geral do teste SUS - AUR-T2.	29
Figura 3.12–Resultado questões positivas do teste SUS - COM-T2.	29
Figura 3.13–Resultado questões positivas do teste SUS - MEC-T2.	30
Figura 3.14–Resultado questões positivas do teste SUS - CAU-T2.	30
Figura 3.15–Resultado questões positivas do teste SUS - AUR-T2.	30
Figura 3.16–Resultado questões negativas do teste SUS - COM-T2.	31
Figura 3.17–Resultado questões negativas do teste SUS - MEC-T2.	31
Figura 3.18–Resultado questões negativas do teste SUS - CAU-T2.	31
Figura 3.19–Resultado questões negativas do teste SUS - AUR-T2.	32
Figura 3.20–Resultado de Satisfação no teste SAM - COM-T2.	32
Figura 3.21–Resultado de Satisfação no teste SAM - MEC-T2.	33
Figura 3.22–Resultado de Satisfação no teste SAM - CAU-T2.	33
Figura 3.23–Resultado de Satisfação no teste SAM - AUR-T2.	33
Figura 3.24–Resultado de Motivação no teste SAM - COM-T2.	34
Figura 3.25–Resultado de Motivação no teste SAM - MEC-T2.	34
Figura 3.26–Resultado de Motivação no teste SAM - CAU-T2.	35
Figura 3.27–Resultado de Motivação no teste SAM - AUR-T2.	35
Figura 3.28–Resultado de Sentimento de Controle no teste SAM - COM-T2.	35
Figura 3.29–Resultado de Sentimento de Controle no teste SAM - MEC-T2.	36
Figura 3.30–Resultado de Sentimento de Controle no teste SAM - CAU-T2.	36
Figura 3.31–Resultado de Sentimento de Controle no teste SAM - AUR-T2.	36
Figura 3.32–Resultado do teste de perfil - COM-T2.	37
Figura 3.33–Resultado do teste de perfil - MEC-T2.	38
Figura 3.34–Resultado do teste de perfil - CAU-T2.	38
Figura 3.35–Resultado do teste de perfil - AUR-T2.	39

Figura 3.36–Página Minhas Tarefas.	45
Figura 3.37–Header.	47
Figura 3.38–Página de Login.	48
Figura 3.39–Página de Perfil.	49
Figura 3.40–Histórico de Navegação.	50
Figura 3.41–Página Tarefa Aberta.	51
Figura 3.42–Componente de pop-up para correção.	52
Figura 3.43–Diagrama da arquitetura do <i>frontend</i>	53
Figura 3.44–Versão final do diagrama ER utilizada atualmente.	58
Figura 3.45–Diagrama ER de banco de dados dinâmico.	59
Figura 3.46–Versão final do diagrama ER proposta nesta monografia.	60
Figura 3.47–Fluxograma do Processo de Login.	69
Figura A.1 – Interface da página “Minhas Tarefas” atual do Opcoders.	76
Figura A.2 – Header atual do Opcoders.	77
Figura A.3 – Interface da página “Perfil” atual do Opcoders.	77
Figura A.4 – Interface da página de <i>Login</i> atual do Opcoders.	78
Figura A.1 – Interface da página Ajuda	85
Figura B.1 – Pop-up de alertas.	86
Figura B.2 – Pop-up de dicas e resumo.	86
Figura B.3 – Pop-up de correções.	87

Lista de Tabelas

Tabela 2.1 – Problemas de Usabilidade.	7
Tabela 3.1 – Resultados Teste SUS.	40
Tabela 3.2 – Resultados Teste SAM.	41
Tabela 3.3 – Resultados Teste Perfil.	42

Lista de Abreviaturas e Siglas

ABNT	Associação Brasileira de Normas Técnicas
DECOM	Departamento de Computação
CEP	Comitê de Ética e Pesquisa
SUS	System Usability Scale
SAM	Self-Assessment Manikin
MVC	Model-View-Controller
ORM	Object-Relational Mapping
SGBD	Sistema de Gerenciamento de Banco de Dados
CNS	Conselho Nacional de Saúde
CONEP	Comissão Nacional de Ética em Pesquisa
CAAE	Certificado de Apresentação de Apreciação Ética
CSRF	Cross-Site Request Forgery
CSS	Cross-Site Scripting
ER	Entidade Relacionamento
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
UFOP	Universidade Federal de Ouro Preto
COM-T1	Turma de Introdução a Computação - Primeiro Teste
PC-T1	Turma de Programação de Computadores I - Primeiro Teste
COM-T2	Turma de Introdução a Computação - Segundo Teste
MEC-T2	Turma de Engenharia Mecânica - Segundo Teste
CAU-T2	Turma de Engenharia de Controle e Automação - Segundo Teste
AUR-T2	Turma de Arquitetura e Urbanismo - Segundo Teste

Sumário

1	Introdução	1
1.1	Justificativa	2
1.2	Objetivos	3
1.2.1	Objetivos Específicos	3
1.3	Metodologia	3
1.4	Delineamento do Trabalho	4
2	Revisão Bibliográfica	5
2.1	Trabalhos Relacionados	5
2.2	Fundamentação Teórica	10
2.2.1	Corretor automático de código-fonte	10
2.2.2	PHP	10
2.2.3	HTML e CSS	11
2.2.4	Javascript e Typescript	11
2.2.5	React	11
2.2.6	TailwindCSS	12
2.2.7	React Router	12
2.2.8	Axios	12
2.2.9	Django	12
2.2.10	Django Rest Framework	13
2.2.11	Arquitetura MVC	13
2.2.12	PostgreSQL	14
3	Desenvolvimento	15
3.1	Testes de Usabilidade	15
3.1.1	Primeiro Teste de usabilidade	16
3.1.1.1	Métodos de Avaliação	16
3.1.1.2	Disposição do Teste	17
3.1.1.3	Resultados	18
3.1.1.3.1	SUS	18
3.1.1.3.2	SAM	21
3.1.1.3.3	Perfil dos Voluntários	23
3.1.2	Segundo Teste de usabilidade	26
3.1.2.1	Métodos de Avaliação	26
3.1.2.2	Disposição do Teste	26
3.1.2.3	Resultados	27
3.1.2.3.1	SUS	27
3.1.2.3.2	SAM	32

3.1.2.3.3	Perfil	37
3.1.3	Discussão	39
3.2	Alterações no Protótipo	44
3.2.1	Psicologia das Cores, Hierarquia Visual e Espaçamento	44
3.2.2	Página Minhas Tarefas	46
3.2.3	Header	47
3.2.4	Página Login	47
3.2.5	Página Perfil	48
3.2.6	Navegação Entre Páginas	49
3.2.7	Página Tarefa Aberta	50
3.3	Desenvolvimento do novo <i>opCoders Judge</i>	51
3.3.1	Frontend	51
3.3.1.1	Arquitetura utilizada	52
3.3.1.2	Funcionalidades Implementadas	53
3.3.1.2.1	Login	53
3.3.1.2.2	Página Minhas Tarefas	54
3.3.1.2.3	Página Perfil	54
3.3.1.2.4	Histórico de Navegação	55
3.3.1.2.5	Tarefa Aberta	55
3.3.1.2.6	Dicas	56
3.3.1.2.7	Página Ajuda	57
3.3.2	Remodelagem do Banco de Dados	57
3.3.2.1	Modelo da versão atual do banco de dados do <i>opCoders Judge</i>	57
3.3.2.2	Modelo da versão dinâmica do banco de dados do <i>opCoders Judge</i>	58
3.3.2.3	Modelo da versão proposto por esta monografia	59
3.3.3	Backend	62
3.3.3.1	Arquitetura utilizada	63
3.3.3.2	Segurança	63
3.3.3.3	Implementação de Funcionalidades	63
3.3.3.3.1	Operações com o Banco de Dados	64
3.3.3.4	CRUDs	64
3.3.3.5	Lógica de Negócio	65
3.3.3.6	Login	66
4	Considerações Finais	70
4.1	Conclusão	70
4.2	Trabalhos Futuros	72

Referências	73
-------------	----

Anexos	75
ANEXO A Interfaces da Versão Atual do <i>opCoders Judge</i>	76
ANEXO B Termo de Consentimento Livre e Esclarecido (TCLE)	79
ANEXO C Questionário de Perfil	82
Apêndices	84
APÊNDICE A Interface da página Ajuda do <i>opCoders Judge</i>	85
APÊNDICE B Design dos novos pop-ups do <i>opCoders Judge</i>	86

1 Introdução

O uso da tecnologia tem ajudado diversas áreas a se desenvolver e isso não seria diferente em um assunto tão importante quanto a educação. Há várias maneiras de aplicar diferentes tipos de tecnologias para o bem da educação. Elas desempenham um papel fundamental no ensino e servem tanto como via principal de aprendizado quanto como ferramenta auxiliar ao ensino presencial ou remoto. As plataformas digitais são ferramentas que permitem o acesso a diferentes tipos de conteúdo educacionais, possibilitando a criação de um ambiente de aprendizagem mais interativo e dinâmico.

O objeto de estudo e desenvolvimento deste trabalho é o *opCoders Judge*, uma plataforma digital criada em torno de um corretor automático voltado para o complemento do ensino de programação básica da UFOP, que permite aos estudantes resolverem exercícios práticos elaborados pelos professores. A ferramenta possui um corretor automatizado capaz de comparar a saída do código do aluno com a saída esperada para aquele problema, a partir dessa análise a nota do aluno na questão é atribuída. A respeito de sua composição, o site foi desenvolvido usando PHP tanto a parte *frontend* quanto *backend* de seu sistema, sua versão atual mantém essa mesma tecnologia, por fim o sistema de correção foi feito em Python. É válido ressaltar que o *opCoders Judge* foi desenvolvido apenas por alunos de Ciência da Computação da UFOP, em trabalhos de monografia de [Cedraz \(2023\)](#), [Patrocínio \(2023\)](#) e [Brito \(2019b\)](#), originando também o artigo científico de [Brito \(2019a\)](#).

É importante salientar que a área da computação historicamente possui um alto índice de evasão, há vários motivos para isso que foram bem observados por [Hoed \(2016\)](#), dificuldade no aprendizado no momento inicial do curso, falhas na didática para explicar algoritmos, mercado de trabalho extremamente competitivo e inflado para primeiras oportunidades de emprego, falta de amparo prático para solidificar os conceitos ensinados em sala de aula, etc. Visto esse contexto, toda forma de facilitar o aprendizado do aluno é válida e deve haver investimento para tal.

Conforme relatado nos trabalhos ([Brito, 2019b](#)) e ([Brito, 2019a](#)), fica nítido o valor do *opCoders Judge*. Contudo, atualmente ele é utilizado somente em Programação de Computadores I, uma disciplina composta por alunos de fora do curso de Ciência da Computação, que estudam esse campo de maneira mais discreta. O motivo disso é que ele apresenta problemas em sua interface, que podem tornar seu uso um tanto trabalhoso e desgastante, além disso, em seu banco de dados todas as questões são tratadas de forma igual, assim não há uma maneira de separar as questões por nível de dificuldade, tópico, palavras-chave e outras classificações, algo essencial caso a plataforma seja usada para diferentes disciplinas. Almeja-se que o *opCoders Judge* possa atender a disciplinas de um grau maior de complexidade, cursadas pelo curso de Ciência da Computação. Problemas de usabilidade do site são muito bem apresentados por [Cedraz \(2023\)](#),

onde foi realizado um método de inspeção e testes de usabilidade com alunos no curso de Ciência da Computação para identificar os problemas na interface e, por fim, foi elaborado um protótipo com propostas de correções dos problemas observados. Esses problemas em um *software* não só afetam a experiência do usuário, como também impedem que a plataforma possa expandir e, além disso, uma ferramenta que não possui uma interface intuitiva e satisfatória pode acabar se tornando um obstáculo e não um facilitador. É importante ressaltar que protótipo é focado em corrigir os problemas de usabilidade atuais, contudo as telas que o compõem apresentam um design básico e um visual não muito moderno.

Esta monografia é fundamentada em resolver os problemas de usabilidade do *opCoders Judge*, baseando-se no protótipo de Cedraz (2023), feito com base em testes de usabilidade aprovados pelo Comitê de Ética e Pesquisa (CEP) da UFOP, podendo ser identificado através do Certificado de Apresentação de Apreciação Ética (CAAE), que possui o número 63182722.9.0000.5150. Contudo, ele também passará de mudanças para torná-lo mais atrativo e moderno. Além disso, é necessário realizar alterações nas tecnologias do site, hoje em dia diversas tecnologias surgem ou evoluem para resolver problemas e melhorar a performance das aplicações, logo o *opCoders Judge* tem muito a ganhar ao substituírmos os conceitos em PHP utilizados para outras ferramentas com mais recursos.

Dado esse contexto, as próximas subseções tem por objetivo detalhar mais sobre o trabalho que está sendo desenvolvido por esta monografia. A Subseção de 1.1 é onde está apresentado os motivos que levaram à escolha do tema e a relevância do projeto. Em seguida, os objetivos gerais e específicos almejados nesse trabalho estão descritos na Subseção 1.2. A Subseção 1.3 descreve a abordagem adotada para o desenvolvimento do trabalho. Por fim, a divisão de seções desta monografia é relatada na Subseção 1.4.

1.1 Justificativa

Acredita-se que aprimorar o *opCoders Judge* nos aspectos mencionados anteriormente seria extremamente vantajoso para o ensino de programação na UFOP, viabilizando também que ele seja utilizado até mesmo por outras instituições. Essa melhoria permitiria que outros educadores utilizassem a plataforma de forma prática para reforçar os conceitos abordados em suas aulas, ampliando assim a eficácia do ensino.

Uma razão significativa que impulsiona esta monografia é a compreensão da importância do papel dos programadores habilitados em nossa sociedade. Na era moderna, a tecnologia está evoluindo rapidamente e tem um impacto positivo em várias áreas. Com isso em mente, é de extrema relevância investir na melhoria do ensino de programação, tanto nas escolas quanto nas universidades, a fim de promover o pensamento computacional em futuros profissionais em diversos nichos, pois cada vez fica mais nítido que aprender sobre programação é fundamental em muitas áreas. Ao fazer isso, contribuiremos para a preparação de indivíduos que poderão enfrentar

os desafios tecnológicos do futuro e promover o desenvolvimento contínuo da sociedade.

1.2 Objetivos

O objetivo geral desta monografia é discorrer sobre a implementação das melhorias da interface propostas no protótipo da monografia de Cedraz (2023), em paralelo a esse processo, fazer uma migração das tecnologias utilizadas na implementação do *opCoders Judge*, indo do *PHP* para tecnologias que colaborem para uma melhor performance da plataforma e por fim remodelar o banco de dados do sistema, para que este seja mais consistente e completo.

1.2.1 Objetivos Específicos

Para alcançar o objetivo principal de maneira coerente, será seguido os seguintes objetivos específicos:

- Repetir teste de usabilidade de Cedraz (2023), com voluntários sendo estudantes da disciplina de Programação de Computadores I.
- Solucionar os problema identificados, realizando alterações no protótipo atual.
- Implementar o *frontend* para solucionar os problemas de interface e melhorar a performance.
- Planejar e implementar um novo modelo de banco de dados que seja mais conciso e completo.
- Implementar o *backend* com alguma *framework* de *Python* buscando uma melhor compatibilidade com o sistema de correção em si.

1.3 Metodologia

Como é possível observar no diagrama da Figura 1.1, o primeiro passo para concretizar este projeto é compreender todos os problemas que afetam a experiência do usuário e corrigi-los durante o desenvolvimento. Embora o estudo de usabilidade realizado por Cedraz (2023) seja de extrema importância nessa etapa, é importante ressaltar ele foi conduzido apenas com alunos do curso de Ciência da Computação, que geralmente possuem maior familiaridade com atividades relacionadas à computação do que estudantes de outros cursos, essa decisão foi tomada pois os estudantes que teriam contato com a plataforma já haviam iniciado as aulas, portanto já haviam conhecido o *opCoders Judge*, portanto não eram elegíveis para o teste. Considerando que a plataforma não será utilizada exclusivamente por alunos de Ciência da Computação, será realizado um novo teste de usabilidade com estudantes de Programação de Computadores I, a fim de obter uma visão mais abrangente, pois esta é composta por alunos de vários cursos diferentes.

Com os resultados obtidos, o foco será realizar alterações no protótipo, conforme as novas informações, para que este possua uma interface com melhor usabilidade e mais intuitiva, atendendo a todos os tipos de estudantes que serão usuários da plataforma.

Após isso, o foco será nas questões mais técnicas de implementação. O banco de dados atual não atende o uso do *opCoders Judge* em diferentes disciplinas e possui algumas incongruências, portanto deverá ser feita uma reformulação do mesmo. Por fim, será feito um estudo para decidir as tecnologias da migração tanto do *frontend* quanto do *backend*, o *PHP* será substituído por tecnologias que melhorarão a performance da plataforma e terão mais compatibilidade com o sistema de correção. Como o sistema de correção do código foi feito com *Python*, o desenvolvimento do *backend* será usando uma *framework* também baseada nessa linguagem.

Através dessa abordagem de metodologia, almeja-se concluir os objetivos de forma satisfatório. É possível compreender melhor a proposta através do diagrama da Figura 1.1.

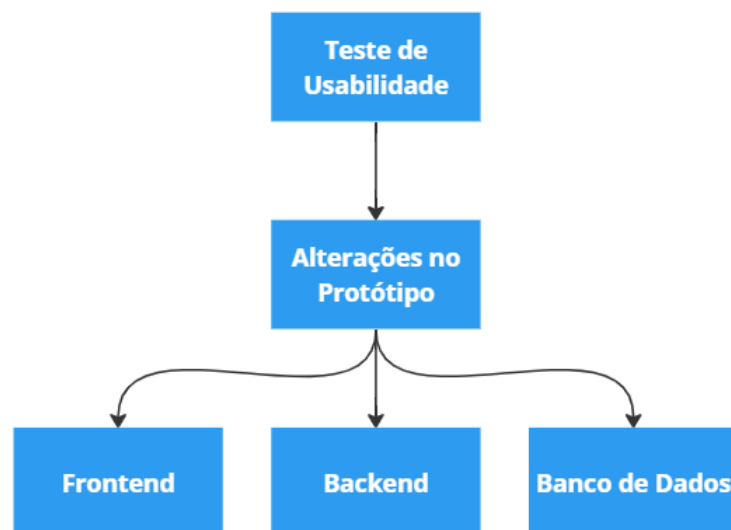


Figura 1.1 – Diagrama da Metodologia.

1.4 Delineamento do Trabalho

O restante deste trabalho está organizado da seguinte maneira:

Capítulo 2: Revisão Bibliográfica, onde será falado sobre os trabalhos relacionados e conceitos importantes para compreender o restante da monografia.

Capítulo 3: Desenvolvimento, onde será dissertado sobre as tarefas realizadas para atingir os objetivos propostos.

Capítulo 4: Conclusão, onde será relatado a análise em relação ao desfecho da monografia e discutido possíveis melhorias.

2 Revisão Bibliográfica

Esta seção é dedicada à apresentação dos trabalhos relacionados que foram fundamentais para o desenvolvimento desta monografia. Estes trabalhos são as monografias e o artigo citados na seção anterior, seus autores foram estudantes de ciência da computação na UFOP e cada um teve sua participação no desenvolvimento do *opCoders Judge* de alguma forma. Além disso, será discorrido sobre conceitos fundamentais para o entendimento do estudo realizado.

2.1 Trabalhos Relacionados

Os primeiros trabalhos a serem mencionados são a monografia (Brito, 2019b) e o artigo (Brito, 2019a), que foram fundamentais para o desenvolvimento do projeto *opCoders Judge*. O trabalho surgiu em resposta à preocupante taxa de desistência na área de programação. A ideia central era criar uma ferramenta prática que auxiliasse no aprendizado teórico, oferecendo uma forma de consolidar os conceitos aprendidos em sala de aula. A solução encontrada foi desenvolver um corretor de código, proporcionando aos professores a possibilidade de criar tarefas para que os alunos as resolvessem, com a correção sendo realizada por meio de uma ferramenta automatizada. No final do trabalho, também foi conduzido um estudo com 31 alunos que utilizaram a ferramenta, a fim de identificar possíveis problemas e áreas de melhoria, alguns questionários foram respondidos pelos mesmos ao final do semestre. Os resultados desse estudo foram positivos, mas também revelaram algumas limitações da ferramenta, como o funcionamento apenas *offline*. No entanto, é importante destacar que esta monografia desempenhou um papel crucial como ponto de partida para o projeto *opCoders Judge*. É válido ressaltar que, sem o embasamento fornecido por esta monografia, a realização do presente trabalho não seria possível.

Patrocínio (2023) realizou um importante trabalho no projeto do *opCoders Judge*, pois deu sequência aos trabalhos anteriores, corrigindo erros e implementando funcionalidades que ampliaram o potencial da ferramenta. O principal problema da primeira versão do corretor era que ele somente funcionava de forma *offline*, isso pode ser viável para uso pessoal, mas se tratando de uma ferramenta cujo objetivo é atender a várias turmas com grande número de alunos, é fundamental que ele esteja disponível sempre que seu uso seja solicitado. Com este pensamento, Patrocínio (2023) foi o responsável por criar a versão web do *opCoders Judge* em sua monografia, desenvolvendo tanto a interface que os usuários interagiriam, quanto o servidor responsável por manter o corretor sempre online. O objetivo do trabalho era a criação de um ambiente propício onde os professores poderiam registrar as tarefas e exercícios práticos que seus alunos deveriam resolver e estes poderiam submeter suas resoluções e obter as devidas respostas prontamente. O resultado foi bastante satisfatório, a ferramenta criada estava em um estado funcional e atendia aos requisitos iniciais, sua versão está sendo utilizada em sala de aula atualmente. Contudo, ela

possui algumas limitações que impedem que a ferramenta atenda a disciplinas com conteúdos mais avançados.

A respeito do estado atual do *opCoders Judge*, sob a perspectiva do usuário, é válido ressaltar que apenas estudantes matriculados em uma disciplina que utiliza o *opCoders Judge* terão acesso à plataforma. Uma vez que o usuário faça o *login*, ele terá acesso à página “minhas tarefas”, conforme ilustrado na Figura A.1 do Anexo A, onde há uma interface que disponibiliza as tarefas abertas para correção, aquelas que já foram entregues no prazo previsto e aquelas que foram perdidas. É válido ressaltar que há um menu lateral para navegação do usuário, porém na versão atual há somente dois módulos que o aluno pode acessar, “minhas tarefas” e “perfil”, como mostrado na Figura A.2 do Anexo A.

Observando a Figura A.3 do Anexo A, podemos ver que a página de perfil tem dois componentes, a finalidade do primeiro é meramente expositiva, nele estão dispostas informações sobre o aluno como: *nome*, *e-mail* e *CPF*. No segundo está a funcionalidade de troca de senha, caso essa funcionalidade seja utilizada, um *popup* aparecerá para relatar o resultado. Analisando esta página podemos concluir que ela aproveita muito mal o seu espaço e não possui muita utilidade, uma possível mudança seria acrescentar novas funcionalidades e disponibilizar informações mais interessantes, como a média das notas do usuário, por exemplo.

Uma vez que é aberta uma tarefa, o usuário é redirecionado para uma página que contém uma lista de seções vertical, sendo que a primeira seção é a descrição da tarefa e as demais questões que a compõem. Sobre a descrição da tarefa, primeiro temos os pré-requisitos para o estudante conseguir entender o problema e logo abaixo uma série de dicas para auxiliar o aluno. A estrutura das questões está dividida entre enunciado, exemplos, área para submissão de código e tabela com histórico de submissões. Ao abrir outra seção, a que estava aberta se fechará, de forma que não há como manter duas seções abertas ao mesmo tempo.

Alguns detalhes que compõem a interface do site em qualquer página, é um “Boas-vindas” no canto superior esquerdo, a hora e data no canto superior direito e um rodapé com a mensagem “feito com carinho pelo DECOM” e um *link* para o site da UFOP.

Um usuário pode enviar quantas soluções ele quiser, sendo que só será considerada a nota da mais alta caso esteja no prazo, pois é possível continuar enviando soluções mesmo após a expiração da tarefa, nesse caso o aluno poderá ver a avaliação do corretor, porém sua nota não será considerada na nota da tarefa.

Para finalizar a descrição do estado atual do *opCoders Judge*, podemos observar na Figura A.4 do Anexo A, que a página de *login* é bem simples, sendo dividida no centro em dois componentes. À esquerda há um componente meramente ilustrativo, que mostra a identidade da plataforma, à direita há os campos para efetuar o *login* e também uma funcionalidade de recuperação de senha.

Um outro trabalho que merece destaque no contexto do aprimoramento do *opCoders*

Judge é o realizado por [Mendonça \(2023\)](#). Nessa pesquisa, foi desenvolvido um inovador modelo de banco de dados, proporcionando a criação de múltiplas versões para as questões designadas aos alunos. Essa abordagem permite que uma pergunta seja apresentada de diferentes maneiras, preservando, no entanto, a essência do problema a ser avaliado. Tal inovação proporciona uma ampla gama de possibilidades, enriquecendo a personalização das questões e proporcionando métodos diversificados para testar o conhecimento dos alunos. Além disso, essa estratégia representa uma eficaz medida preventiva contra possíveis tentativas de cola entre colegas de turma, promovendo a integridade acadêmica.

O trabalho mais importante para a fundamentação desta monografia é o de [Cedraz \(2023\)](#). Ele foi totalmente focado na interface do projeto e na experiência de usuário, o objetivo era identificar todos os problemas relacionados a isso e elaborar um protótipo para ser implementado em trabalhos futuros. Com esse propósito, a autora realizou um teste de inspeção, seguindo as 10 Heurísticas de Nielsen, a fim de identificar possíveis falhas na interface. Além disso, foram conduzidos testes de usabilidade por meio da aplicação dos questionários *System Usability Scale (SUS)* e *Self-Assessment Manikin (SAM)* com estudantes de Ciência da Computação da UFOP. Essa abordagem de pesquisa permitiu identificar as principais dificuldades enfrentadas pelos usuários da plataforma. Com os resultados obtidos foi elaborado o protótipo que servirá de guia para a implementação da nova interface do *opCoders Judge* nesta monografia.

No desenvolvimento deste trabalho muito se falará sobre o protótipo e os problemas que ele soluciona, para deixar mais claro do que será falado, a Tabela 2.1 dispõe a descrição de cada problema, tal como seu nível de gravidade na escala de classificação de problemas usada na própria monografia ([Cedraz, 2023](#)).

Tabela 2.1 – Problemas de Usabilidade.

Problemas de Usabilidade Identificados na Versão Atual do <i>opCoders Judge</i>			
Nº	Descrição	Gravidade	Local
1	Na descrição da tarefa não informa aos usuários quais são os tipos de arquivos permitidos para envio. Como por exemplo arquivos com extensão.py ou .cpp.	Grave	Tela de envio de tarefas

Continua na próxima página.

Problemas de Usabilidade Identificados na Versão Atual do <i>opCoders Judge</i>			
Nº	Descrição	Gravidade	Local
2	Ao tentar submeter um código que não tenha a extensão .py a ferramenta não exibe uma mensagem de erro que informe ao usuário o que está acontecendo e ele não consegue submeter o exercício.	Catastrófico	Tela de envio das tarefas
3	A descrição de como utilizar a ferramenta se encontra apenas na tarefa "Testando o corretor automático", por serem informações importantes para a boa utilização da ferramenta deveriam estar de forma visível.	Catastrófico	Tela de envio das tarefas
4	A descrição da tarefa possui uma grande quantidade de informações e diminui a visibilidade das instruções mais relevantes.	Grave	Tela de envio das tarefas
5	O botão "Escolher arquivo" na cor cinza não dá o destaque necessário a ele.	Cosmético	Tela de envio das tarefas
6	Na página das tarefas, ao clicar em uma questão para ler o enunciado a descrição da mesma fica oculta, tirando do usuário o controle sobre o que ele deseja visualizar.	Médio	Tela de envio das tarefas
7	As frases com cores diferentes na descrição das questões passam a impressão de que são clicáveis.	Menor	Tela de envio das tarefas
8	Ao abrir a tela para visualizar a correção do código não tem um botão que permita o usuário retornar a tela anterior.	Médio	Tela de envio das tarefas

Continua na próxima página.

Problemas de Usabilidade Identificados na Versão Atual do <i>opCoders Judge</i>			
Nº	Descrição	Gravidade	Local
9	Ao enviar uma tarefa para correção o status dela fica "Pendente". O uso dessa expressão pode confundir o usuário, dando a entender que a tarefa não foi enviada.	Grave	Tela de envio das tarefas
10	Para que o aluno consiga visualizar a tarefa corrigida é necessário atualizar a página. Ao clicar para visualizar a correção outra página no navegador é aberta, saindo da que o usuário estava.	Catastrófico	Tela de envio das tarefas
11	Ao realizar a alteração da senha a ferramenta não oculta os caracteres digitados e não oferece o botão para exibir ou não os caracteres na tela. Dessa forma, ao realizar essa ação, se o usuário voltar a tela anterior ele consegue visualizar a senha digitada, tornando a ferramenta pouco segura.	Médio	Tela de perfil
12	Como o cadastro na ferramenta é feito de forma automática, pode acontecer do usuário não lembrar que as credenciais de login foram enviadas por email e não conseguir acessar a ferramenta.	Catastrófico	Tela de perfil
13	Na ferramenta não há nenhum menu que forneça ao usuário informações básicas para que ele consiga utilizar o sistema e concluir sua tarefa de forma satisfatória.	Catastrófico	Tela inicial

Fonte: (Cedraz, 2023).

Como pode ser observado, 11 dos 13 problemas encontrados está na tela de envio das tarefas. Este é, possivelmente, o componente mais importante do site, então é crucial que ele

sofra muitas alterações para alcançar um bom desempenho em questão de usabilidade, como foi elaborado no protótipo atual. As telas de perfil e inicial também sofreram mudanças importantes.

Todos os trabalhos que fazem parte desta revisão bibliográfica são de extrema importância para o andamento deste, pois a maioria deles constituem as versões anteriores do projeto. Suas descrições, problemas relatados, sugestões de trabalhos futuros e conclusões servirão para que o trabalho aqui desenvolvido evite possíveis problemas e caminhe para a direção certa, evoluindo o *opCoders Judge* para que ele alcance a excelência em sua função.

2.2 Fundamentação Teórica

Nesta seção serão apresentadas explicações a respeito de conceitos que abrangem esse projeto, visando facilitar a compreensão do que será mostrado no desenvolvimento. Também será relatado a definição, objetivo e motivo de escolha das tecnologias usadas no desenvolvimento do *opCoders Judge*.

2.2.1 Corretor automático de código-fonte

Um corretor automático de código-fonte é uma ferramenta computacional desenvolvida para auxiliar o processo de aprendizagem de programação, utilizando uma abordagem mais prática. De acordo com Brito (2019b), nos estudos de computação é bem comum algoritmos de difícil compreensão partindo de um ponto de vista mais teórico, isso é um dificultador para os professores que podem apenas passar os conceitos em sala de aula. Uma possível solução desse problema é a utilização de um corretor de código *online*, pois, neste ambiente, é possível registrar ou encontrar tarefas a serem submetidas para correção. A validação da solução dessas tarefas pode ser feita de várias maneiras, a mais comum é a comparação, caractere a caractere, da saída do código submetido com a saída esperada como solução daquela tarefa. Uma grande vantagem dessa ferramenta é que o próprio sistema realiza a validação de forma instantânea, eliminando assim a necessidade de um educador corrigir todas as tarefas submetidas de sua turma.

2.2.2 PHP

O *PHP* é uma linguagem de programação bastante amigável para iniciantes e muito utilizada para desenvolvimento web. Ela foi criada especificamente para criar páginas dinâmicas e interativas, permitindo que os desenvolvedores construam sites e aplicativos que sejam interativos e capazes de se comunicar com bancos de dados. Algo que diferencia o PHP de outras linguagens de programação é a capacidade de gerar *script* do lado do servidor, o que significa que o código é executado no servidor antes de enviar os dados resultantes para o navegador. Os demais conceitos são muito bem explicados no livro de Park (2002).

Como dito anteriormente, a primeira versão *online* do *opCoders Judge* foi implementada em *PHP*, tanto no *frontend* quanto no *backend*.

2.2.3 HTML e CSS

Como pode ser observado em [Parashar \(2022\)](#) São duas linguagens de marcação utilizadas *Hypertext Markup Language* (HTML) e *Cascading Style Sheets* (CSS), permitem a criação de páginas web estáticas, ou seja, não possuem funcionalidades (ou possuem muito pouco). Para lidar com essa parte é necessário o uso de uma linguagem de programação.

2.2.4 Javascript e Typescript

Uma linguagem de programação extremamente popular atualmente é o *Javascript*, ele possui muita eficiência em dar um dinamismo as páginas web, acrescentando funcionalidades e lógicas aos componentes estáticos criados com HTML e CSS da maneira que o desenvolvedor almejar. Contudo, o *Javascript* é uma linguagem fracamente tipada, isso facilita a ocorrência de erros no processo de implementação. Para resolver esse problema foi criado o *Typescript*, que de acordo com [Cherny \(2019\)](#), é uma linguagem de programação fortemente tipada, que funciona de maneira muito parecida com o *Javascript*, porém possui algumas restrições intencionais, que fazem com que o desenvolvimento seja mais seguro, por isso essa foi a linguagem de programação escolhida para o desenvolvimento do *opCoders Judge*.

2.2.5 React

React é uma biblioteca *JavaScript*, isto é, uma coleção de funcionalidades já prontas e feitas nessa linguagem. O *React* é muito popular e utilizada para criar interfaces de usuário interativas e responsivas. vale ressaltar que o *React* não é uma linguagem de programação como o *PHP*, mas sim uma biblioteca que permite construir componentes reutilizáveis para a criação de interfaces de usuário.

Segundo [Fedosejev \(2015\)](#), o *React* tem duas grandes vantagens que justificam sua alta popularidade. A primeira é sua estrutura voltada para a componentização, usando esta biblioteca você é incentivado a dividir a interface do usuário em componentes independentes e reutilizáveis, o que facilita bastante a manutenção e o desenvolvimento de aplicações mais complexas. A segunda vantagem é a utilização de uma técnica chamada *Virtual Dom*, que permite ao navegador atualizar apenas as partes necessárias da interface do usuário, essa técnica melhora bastante o desempenho das interfaces e proporciona uma experiência de usuário mais ágil.

Há diversas ferramentas possíveis de se utilizar para implementar o *frontend*, como: *Vue*, *Angular*, *Svelte*. Mas visando as vantagens citadas anteriormente, neste projeto as novas interfaces do usuário serão implementadas utilizando o *React*.

2.2.6 TailwindCSS

Uma *framework* utilizada neste projeto é a **TailwindCSS**, ele é feito para otimizar o uso do CSS. Essa *framework* possui diversos códigos CSS prontos e permite que eles sejam instanciados no HTML dos componentes, por meio de classes. Isso facilita e acelera o desenvolvimento *frontend*, contudo o lado negativo é que quando os componentes usam muitas dessas classes fornecidas pelo **TailwindCSS**, o código pode ser difícil de ler. Informações sobre essa ferramenta pode ser encontrada em sua documentação [Schoger \(2017\)](#).

2.2.7 React Router

No desenvolvimento do *frontend* é fundamental lidar com as rotas de navegação, que funcionam por meio da URL do *browser*. Para permitir a navegação e garantir a segurança das rotas, está sendo usado a biblioteca **React Router**, como o próprio nome já sugere, feita para funcionar em conjunto do *React*. Segundo [Ganatra \(2018\)](#), essa ferramenta é um conjunto de ferramentas, muito versáteis, para que um desenvolvedor consiga lida com qualquer sistema de paginação de uma plataforma online. É válido ressaltar que recentemente o **React Router** tem passado por muitas alterações em seu funcionamento, algumas alterações se devem pois antigas metodologias de utilização da ferramenta não funcionavam mais, portanto foi necessário no desenvolvimento desta monografia a utilização da documentação como orientação.

2.2.8 Axios

O sistema de *frontend* é fundamental para estabelecer a comunicação com o *backend*, e para facilitar essa interação, foi decidido por incorporar o **Axios** como ferramenta. Como é possível ver na seção sobre essa ferramenta em [Ratn \(2022\)](#), ele destaca-se por sua sintaxe concisa e intuitiva, simplificando significativamente a criação e manipulação de requisições HTTP. Em ambientes de desenvolvimento web dinâmicos, lidar com respostas assíncronas é muito importante. Nesse contexto, o **Axios** se destaca ao oferecer uma abordagem eficiente e eficaz para tratar operações simultâneas, permitindo que o sistema execute diversas ações de forma concorrente. Essa capacidade é crucial para a agilidade e responsividade de aplicações web, garantindo uma experiência de usuário mais fluida e eficiente.

2.2.9 Django

O *Django* é uma *framework Python*, que é uma linguagem de programação versátil e de fácil aprendizado, conhecida por sua legibilidade e sintaxe intuitiva. Segundo ([Forcier Paul Bissex, 2008](#)), *Django* é usado para desenvolvimento web escrito, ele fornece uma estrutura robusta e abrangente para criar aplicativos e plataformas de alta qualidade. O *Django* vem com uma série de recursos e funcionalidades prontas para uso, como autenticação de usuários, administração do

site, *Object-Relational Mapping (ORM)* para acesso a banco de dados, tratamento de formulários, entre outros. Isso traz uma otimização muito grande no desenvolvimento.

Django possui uma série de vantagens que justificam o seu uso nesta monografia, elas incluem a facilidade de criação de aplicativos web, a segurança embutida, o suporte a banco de dados relacionais e uma arquitetura bem estruturada. Sua arquitetura é fechada com base no padrão de projeto *Model-View-Controller (MVC)*.

Por padrão, este *framework* vem implementado o banco de dados *SQLite* por a ser uma ferramenta bem leve. Contudo, o *Django* tem compatibilidade com outros bancos de dados e permite a alteração nas configurações. Para este projeto será utilizado o *PostgreSQL*.

Mesmo tendo excelentes opções para o desenvolvimento de *backend* como *Laravel*; *Ruby on Rails*; *ExpressJS*, para esse projeto foi escolhido o *Django*, não só pelas inúmeras vantagens descritas anteriormente, mas pelo fato de que ser baseado em *Python* resulta em uma maior compatibilidade com o sistema de correção de código fonte.

2.2.10 Django Rest Framework

O *Django Rest Framework* desempenha um papel essencial no projeto, estendendo as capacidades do Django para simplificar a criação de APIs *RESTful*. Conforme mostrado em [Vainikka \(2018\)](#), essa extensão adiciona funcionalidades cruciais, como manipulação de dados, autenticação e serialização, alinhando-se aos princípios e convenções da arquitetura REST. Sua integração proporciona eficiência no desenvolvimento, seguindo boas práticas e padrões estabelecidos para a construção de APIs sólidas e escaláveis. Essa escolha estratégica reforça a base do sistema, garantindo uma implementação coesa e alinhada às demandas contemporâneas de desenvolvimento web.

2.2.11 Arquitetura MVC

A estrutura do servidor da plataforma é uma parte muito importante do projeto, o direcionamento do desenvolvimento varia de acordo com o padrão de arquitetura que se usa. Conforme podemos ver em ([Spath, 2020](#)), arquitetura *MVC* é um padrão de arquitetura de software amplamente utilizado para desenvolvimento de aplicativos, sua abordagem é separar e organizar as partes do código em três componentes principais, *model*, *view* e *controller*.

O *model* é responsável por representar e gerenciar determinada entidade do aplicativo. Com ele podemos manipular e validar os dados, lidar com a lógica de negócio, interagir com o banco de dados realizando as operações básicas, como: inserir, buscar, deletar e atualizar os dados.

A *view* é responsável por apresentar os dados ao usuário. Ela pode atuar retornando os dados pedidos em uma requisição *http* e apresentar uma interface gráfica ou representação visual

dos dados. Contudo, ela não contém a lógica de negócio, ela apenas recebe as informações do *model* e as repassa para o usuário em algum formato.

O *controller* é o intermediador, ele recebe as interações do usuário na *view* e regula as ações correspondentes no *model*. Ele também é responsável por receber requisições *http* e determinar qual ação executar em resposta.

Essa separação dos componentes no padrão *MVC*, permite uma melhor organização do código, uma vez que a localização e responsabilidades do código estão previamente estabelecidas, isso ajuda na disposição dos diretórios e manutenção do projeto. É válido ressaltar que os componentes são independentes entre si, logo, realizar alterações em um deles não impacta em outro, conseqüentemente, isso facilita a reutilização dos mesmos. É interessante ressaltar que essa é a arquitetura atualmente utilizada pelo *opCoders Judge*, logo alguns detalhes de implementação poderão ser aproveitados.

2.2.12 PostgreSQL

A definição do *PostgreSQL* é muito bem explicada em (Obe, 2012), segundo os autores é um Sistema de Gerenciamento de Banco de Dados (**SGBD**), contudo é popularmente conhecido como um banco de dados *SQL*. É uma ferramenta feita para organizar e armazenar dados de forma estruturada e aplicando regras. Com ele é possível criar tabelas, armazenar e buscar informações de forma prática e confiável, isso se deve ao fato do *PostgreSQL* oferecer recursos avançados de integridade de dados, garantindo que as informações armazenadas sejam consistentes e estejam livres de corrupção.

MySQL; *MongoDB* e *SQLite* foram outras opções de **SGBD** estudadas, porém para o *opCoders Judge* foi escolhido o *PostgreSQL* pensando no objetivo de atender a mais disciplinas e turmas do que atualmente, dessa forma com o passar dos tempos cada vez mais informação precisa ser armazenada, para que essa escalabilidade ocorra consistentemente é necessário um banco de dados mais robusto do que o *SQLite*, que vem por padrão no *Django*.

3 Desenvolvimento

Neste capítulo serão apresentados os métodos utilizados tanto na pesquisa quanto no desenvolvimento. A Seção 3.1 descreverá o embasamento dos testes de usabilidade feito para este projeto. A Seção 3.2 explicará as mudanças que devem ser realizadas no protótipo feito por Cedraz (2023). Por fim, a Seção 3.3 discorrerá sobre a implementação que foi realizada referente às funcionalidades do *opCoders Judge*, sendo elas divididas em: *frontend*, *backend* e banco de dados.

3.1 Testes de Usabilidade

No trabalho de Bastien (2010), podemos ver que teste de usabilidade é uma técnica de extrema importância para o desenvolvimento de algum produto ou sistema interativo. Através dele podemos avaliar aspectos como eficácia, eficiência, satisfação e facilidade de uso do produto. Muitas vezes os desenvolvedores por estarem muito acostumados com a interface do produto, ou até mesmo, por terem um conhecimento técnico mais elevado sobre aquele assunto do que usuários comuns, não conseguem identificar alguns aspectos como problemas. E é nesse tipo de situação que podemos enxergar claramente o valor que testes de usabilidade possuem.

De acordo com a Resoluções 466/2012 e 510/2016 do Conselho Nacional de Saúde (CNS), todas as pesquisas envolvendo seres humanos no Brasil devem seguir procedimentos éticos. Para garantir tais procedimentos, a criação da Comissão Nacional de Ética em Pesquisa (CONEP) em 1996 estabeleceu a coordenação dos Comitês de Ética em Pesquisa, responsáveis por avaliar projetos de pesquisa, que tem o objetivo de garantir o respeito à dignidade humana, consentimento informado e minimização de riscos. O CEP aprova eticamente os projetos, protegendo os participantes e assegurando a integridade das pesquisas, ou seja, é um requisito fundamental para a realização de testes de usabilidade.

Por se tratar de um trabalho que envolve testes com seres humanos, no trabalho de Cedraz (2023), foi encaminhado ao CEP-UFOP os documentos referentes a pesquisa para avaliação. O trabalho foi aprovado pelo Comitê de Ética da UFOP e pode ser identificado através do Certificado de Apresentação de Apreciação Ética (CAAE) de número 63182722.9.0000.5150. Por estas razões, para garantir todos os procedimentos éticos, os testes de usabilidade realizado nesta monografia seguem o mesmo padrão de aplicação adotado pelo antecessor, complementando os testes com usuário em acordo com o projeto aprovado no Comitê de Ética.

Como falado anteriormente, Cedraz (2023) conduziu um teste de usabilidade com voluntários para identificar os problemas de usabilidade das interfaces atuais do *opCoders Judge*. Embora a utilização da plataforma, até o momento, seja restrita aos alunos da disciplina de

Programação de Computadores I, a qual é uma disciplina eletiva aberta a estudantes de diversos cursos além de Ciência da Computação, o teste foi realizado com alunos específicos do curso de Ciência da Computação. Essa diferença de “público-alvo” ocorreu devido ao momento em que o teste estava para acontecer, os alunos da disciplina já estavam frequentando aula, logo já conheciam a plataforma. Para realizar um teste de usabilidade é importante que os voluntários nunca tenham tido contato com o produto, qualquer experiência prévia pode influenciar sua interação com o sistema e prejudicar os resultados.

Com esse teste foi possível tirar resultados valiosos, contudo é intuitivo pensar que, estudantes de Ciência da Computação tenham mais facilidade para interagir com uma interface de um corretor de código-fonte online, por estarem mais familiarizados com esse tipo de plataforma, ou até mesmo por ser provável de já terem utilizado de plataformas similares anteriormente. Por este motivo, neste trabalho foi realizado novos testes de usabilidade, dessa vez com estudantes da disciplina Programação de Computadores I. Em outras palavras, as conclusões tiradas analisando os resultados do teste realizado somente com estudantes de Ciência da Computação, podem não ter identificado algumas problemáticas da plataforma como problemas a serem tratados, portanto, para realizar uma análise mais profunda no estudo dessas questões foram realizados dois testes de usabilidade nesta monografia.

Desta forma esta seção está dividida em 3 subseções, sendo que a Subseção 3.1.1 discorrerá sobre o processo e resultados do primeiro teste, a Subseção 3.1.2 seguirá o mesmo padrão e descreverá as informações do segundo teste realizado e por fim a Subseção 3.1.3 realizará uma discussão acerca dos resultados dos dois testes.

3.1.1 Primeiro Teste de usabilidade

Para explicar todos os aspectos que dizem respeito ao primeiro teste realizado, as próximas Subseções consistem em 3.1.1.1 para explicar quais foram os métodos de avaliação adotados, 3.1.1.2 detalha apresenta mais detalhes e por fim 3.1.1.3 descreverá os resultados encontrados.

3.1.1.1 Métodos de Avaliação

Visando comparar os resultados dos testes de usabilidade e garantir os requisitos éticos, o novo teste seguiu o mesmo padrão do antigo, incluindo o uso do mesmo protótipo. O objetivo dos voluntários era apenas enviar uma tarefa, previamente preparada, para avaliação. É importante ressaltar que esse teste foi realizado em uma plataforma chamada **Figma**, esta é uma plataforma de prototipação, ela permite a criação de interfaces e navegação entre elas, dessa forma foi possível criar uma simulação do que seria uma interação com a plataforma *opCodere Judge*, logo todos os elementos eram componentes *UI* interativos, até mesmo o arquivo que seria submetido para resolução. Isso ocorreu, pois o protótipo feito por Cedraz (2023) não havia sido implementado.

Um dos métodos que compõem essa avaliação é o formulário **SUS**, ele é uma ferramenta

muito eficiente para analisar a facilidade de uso, aprendizado e usabilidade de um sistema, além de ser bem simples de ser aplicado (Sauro, 2011). Além disso, outro aspecto importante de ser avaliado, que pode ser verificado pelo SUS, é a satisfação dos usuários pela interface, mesmo se as páginas não tivessem erros que impeçam o funcionamento adequado do sistema, é de fundamental importância que elas sejam atraentes e interessantes de se interagir.

Além de analisar a satisfação dos usuários, é interessante aprofundar ainda mais o entendimento deles e verificar como a utilização da plataforma afeta suas emoções e sentidos, já que a utilização da plataforma acontece de forma bastante frequente pelos estudantes (Barbosa, 2010). Assim, também foi usado o **SAM**, por ser “uma escala capaz de identificar a satisfação, motivação e sentimento de domínio do usuário ao utilizar o sistema”, segundo Cedraz (2023).

Cada usuário tem uma “bagagem intelectual” consigo, alguns podem ter mais dificuldades do que outros por estarem mais familiarizados com esse tipo de plataforma, para podermos ter um entendimento melhor das características de cada um, também foi fornecido um formulário que diz respeito ao **perfil** do usuário, assim podemos ter uma análise mais justa ao comparar os resultados de cada voluntário.

Por fim, para detalhar ainda mais a pesquisa durante a execução do teste foi observado e anotado todas as interações consideradas relevantes, é fundamental entender a linha de raciocínio do usuário conforme os elementos das interfaces vão sendo introduzidos. Além disso, é claro, foi observado o tempo que cada voluntário levou para concluir o teste, isso é importante para entender o quão intuitiva a interface está e se, mesmo sem conhecer a plataforma, esta tenha elementos o suficiente para “ensinar” ao usuário como concluir o objetivo desejado.

3.1.1.2 Disposição do Teste

Nielsen (2006) acredita que em estudos qualitativos os testes de usabilidade devem ter pelo menos 20 voluntários, contudo não há concordância com esse número no meio científico deste ramo. Portanto, foi utilizado, no teste de Cedraz (2023), um total de 30 usuários para esse teste, pois, segundo Macefield (2016), é possível encontrar em média 99% de problemas de interface utilizando essa quantidade. Neste trabalho, a convocação foi feita via e-mail pelo professor da disciplina, contudo por razões de disponibilidade de tempo dos estudantes, no tempo determinado para os testes (três dias), foi alcançado somente 20 voluntários para o teste.

Antes da realização do teste, foi passado para cada participante o Termo de Consentimento Livre e Esclarecido, conforme o Anexo B, após este assinado o teste descrito anteriormente foi iniciado. Após o término da tarefa, é válido ressaltar que cabia ao usuário declarar que havia alcançado o objetivo, os voluntários responderam o questionário **SAM** em manuscrito e os questionários de **perfil**, mostrado no Anexo C, e **SUS** foram enviados por e-mail, em formato de *Google Forms*.

Tanto no trabalho de Cedraz (2023) quanto no primeiro teste desta monografia, foram

realizados com um voluntário por vez, com o aplicador do teste cronometrando o tempo. Com esta abordagem a análise é mais profunda, pois é possível observar todo o “caminho” que o voluntário percorre na navegação da plataforma virtual, para cumprir a atividade.

É importante ressaltar que nesse teste os participantes consistiram em alunos que cursavam a disciplina **Programação de Computadores I**, sem haver separação por turma ou curso, contudo essa informação foi requerida no formulário de perfil, de forma que se for interessantes, essa separação em grupos era possível de se fazer.

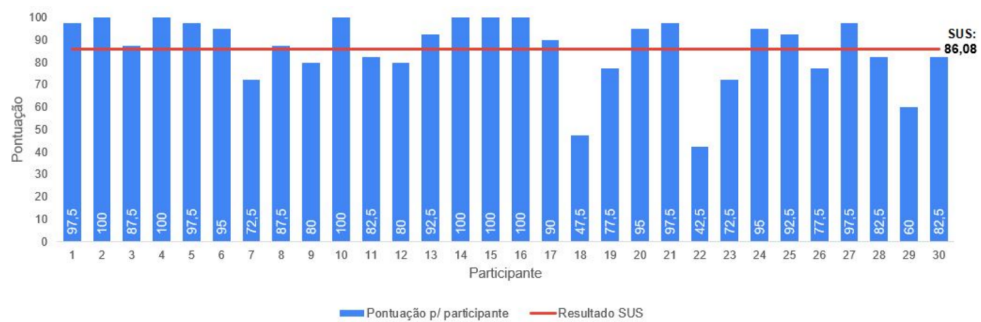
3.1.1.3 Resultados

Nesta seção, serão abordados detalhes sobre a execução dos testes **SUS**, **SAM** e de **Perfil**, e também será apresentado e comparado os resultados obtidos com o teste **COM-T1** (feito com a turma da computação de **Introdução a Programação** por [Cedraz \(2023\)](#)) e **PC-T1** (feito nesta monografia com alunos de diversos cursos da disciplina de **Programação de Computadores I**).

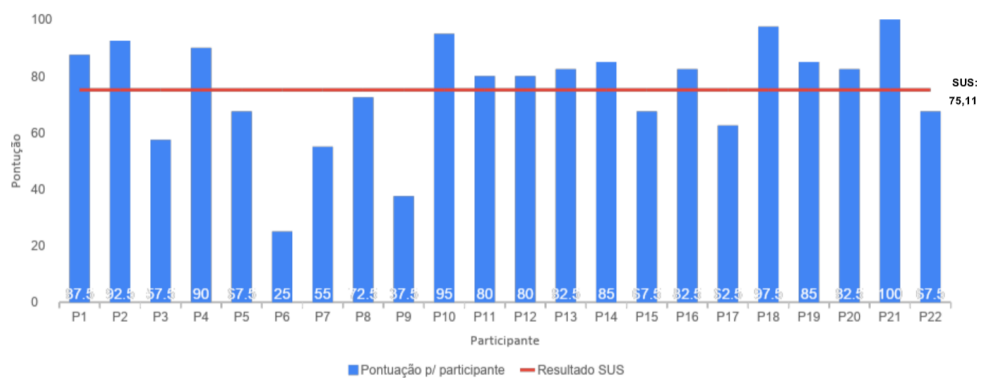
3.1.1.3.1 SUS

Com a aplicação do questionário SUS seguindo o mesmo padrão utilizado em [Cedraz \(2023\)](#), foi possível avaliar a efetividade, eficiência e satisfação dos participantes, enquanto é traçado um paralelo com os diferentes voluntários que colaboraram nos dois testes. O questionário é composto pelas seguintes perguntas, sendo elas para indicar pontos positivos e negativos do objeto avaliado:

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



(a) COM-T1.



(b) PC-T1.

Figura 3.1 – Resultado do teste SUS geral.

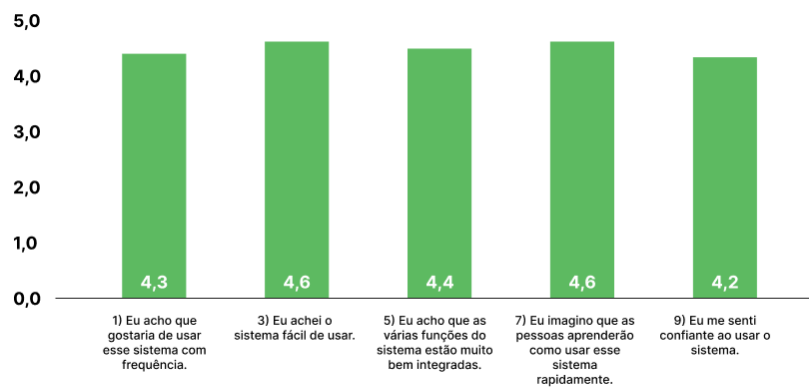
A Figura 3.1 representa o resultado dos testes **COM-T1** e **PC-T1**. É possível notar que o resultado geral da aplicação para o primeiro grupo, seguindo as métricas do questionários SUS, foi de **86,06**, este valor é considerado um resultado muito bom.

Já sobre o segundo grupo, o resultado desse teste foi de **75,11**. Ainda que o resultado do teste tenha sido positivo, houve uma queda significativa, de **10,95**, na avaliação em relação ao primeiro grupo.

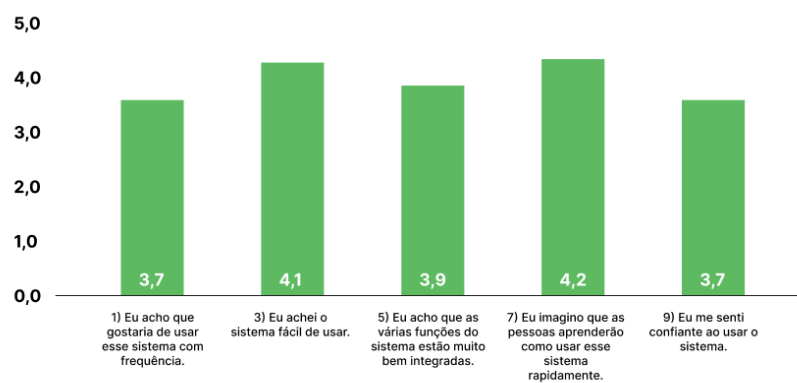
Para fazer uma análise mais profunda do formulário **SUS**, foi analisado os resultados das questões positivas e negativas. É válido ressaltar que em gráficos das questões positivas, quanto maior a nota melhor, já nos gráficos das questões negativas, ilustrado posteriormente, é o contrário, quanto maior os valores, pior o resultado.

A respeito das questões positivas, na Figura 3.2, apresentam-se os resultados das questões obtidos no teste **COM-T1**. É evidente que todas as questões receberam pontuações superiores a **4** e que a média dos resultados é de **4,42**, o que é um bom indicativo de qualidade. A pontuação mínima foi de **4,2**, enquanto a máxima atingiu **4,6**, refletindo uma diferença de apenas **0,4**. Nota-se que os valores atribuídos às questões neste teste estão bastante próximos entre si.

A Figura 3.2 também dispõe o resultado das questões positivas referentes a **PC-T1**. É possível observar que, diferente do teste de **COM-T1**, 3 das 5 questões tiveram um resultado abaixo de **4**. A menor nota atingiu um valor de **3,7**, enquanto a maior **4,2**, também não houve uma grande disparidade entre elas. Com uma média de **3,92**, há uma queda de **0,5** para o teste



(a) COM-T1.



(b) PC-T1.

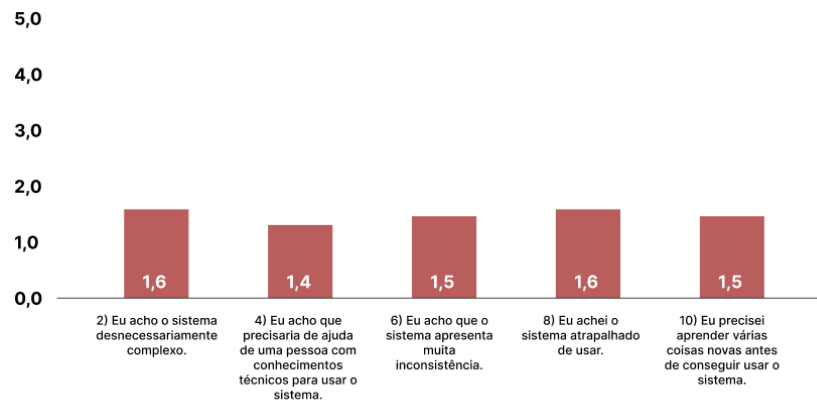
Figura 3.2 – Resultado do teste SUS para questões Positivas.

anterior. Além disso, analisando cada pergunta separadamente, é possível notar que em nenhuma questão o resultado, do teste com alunos de **Programação de Computadores I**, foi igual ou superior ao da turma de **Ciência da Computação**, evidenciando uma diferença significativa.

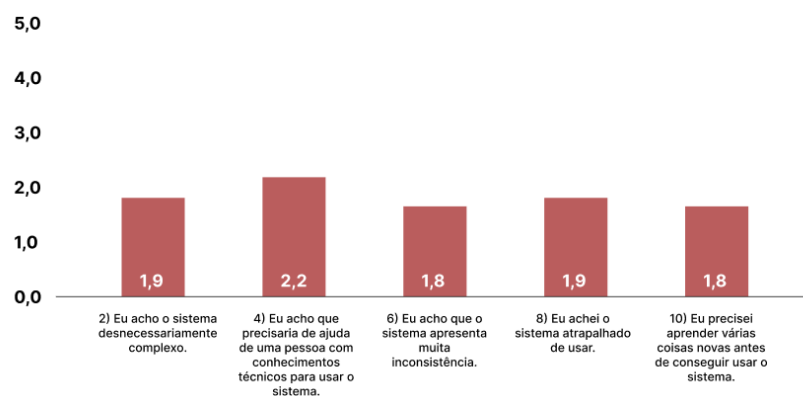
Fazendo uma análise das questões negativas do teste **SUS**, referentes a **COM-T1**, é possível notar que todos os valores estão baixos, o que é um indicativo positivo. A média das questões atingiu um valor de **1,52**, sendo que o pior resultado alcançou um valor de **1,6** e o melhor **1,4**, ou seja, variou muito pouco. Conforme os dados disponibilizados na Figura 3.3.

A Figura 3.3 também evidencia que todas as questões da turma do teste **PC-T1** obtiveram resultados inferiores em comparação com a turma de **COM-T1**. Os resultados para todas as questões situaram-se em torno de **2**, com uma delas alcançando até uma pontuação superior. A média das questões negativas dessa turma totalizou **1,92**, representando uma diferença de **0,4** em relação à turma de **Ciência da Computação**.

A partir desses dados é possível perceber que em nenhuma análise **PC-T1** conseguiu resultados melhores do que **COM-T1**, isso é um indício positivo para a validação da tese que motiva esses novos testes de usabilidade. Porém, ainda é necessário mais dados para aprovar essa teoria.



(a) COM-T1.



(b) PC-T1.

Figura 3.3 – Resultado do Teste SUS para questões Negativas.

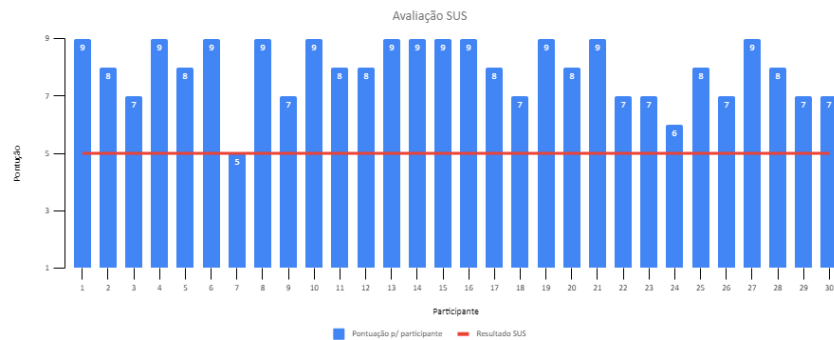
3.1.1.3.2 SAM

Utilizando o questionário **SAM**, foi viável avaliar as três dimensões emocionais (satisfação, motivação e sentimento de domínio) dos participantes após a interação com a interface. Para cada dimensão foi apresentada uma imagem e uma escala de 1 a 9, foi solicitado ao voluntário que respondesse o valor que ele mais achava adequado para cada dimensão, de acordo com seu sentimento a respeito do *opCoders Judge*. Para a análise das respostas foram divididas em três grupos:

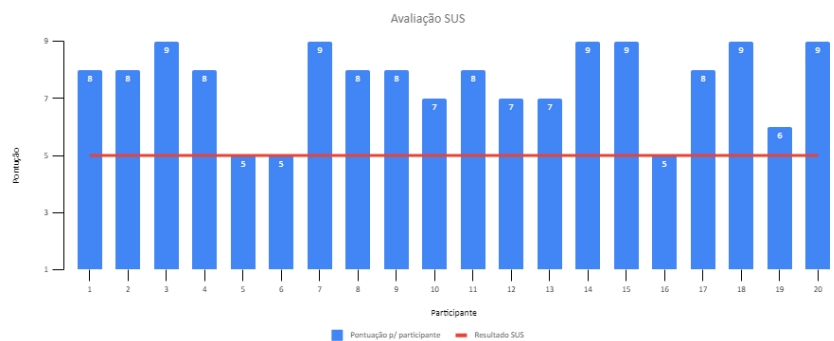
- **Respostas Negativas:** respostas com valor de 1 a 4.
- **Respostas Neutras:** respostas com valor igual a 5.
- **Respostas Positivas:** respostas com valor de 6 a 9.

Conforme ilustrado na Figura 3.4, a respeito da dimensão de **Satisfação**, a média do teste **COM-T1** foi de **8,0** pontos. Além disso, somente um aluno teve uma resposta neutra, nenhum uma resposta negativa e o restante respostas positivas. Isso demonstra um resultado bastante positivo em relação a esse quesito. Já os alunos de **PC-T1**, obtiveram uma média de **7,6** e três

alunos obtiveram respostas neutra, nenhum resposta negativa e o restante respostas positivas. Um resultado bastante positivo, contudo ainda ligeiramente menor do que **COM-T1**.



(a) COM-T1.



(b) PC-T1.

Figura 3.4 – Resultado do SAM sobre Satisfação.

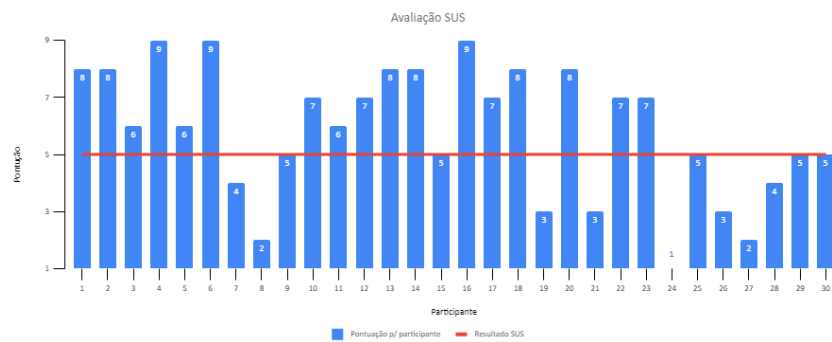
Em relação à **Motivação**, conforme os dados da Figura 3.5, **COM-T1** obteve uma média de **5,8** e 8 voluntários obtiveram resposta negativa, sendo uma delas uma avaliação de **1**, a pior nota possível no teste. Um valor bastante inferior em relação à **Satisfação**.

É interessante observar que **PC-T1** obteve a mesma média em relação à **Motivação**, **5,8**. Desta turma, 6 voluntários avaliaram com nota negativa, sendo duas delas a avaliação mínima possível, como evidenciado na mesma Figura 3.5.

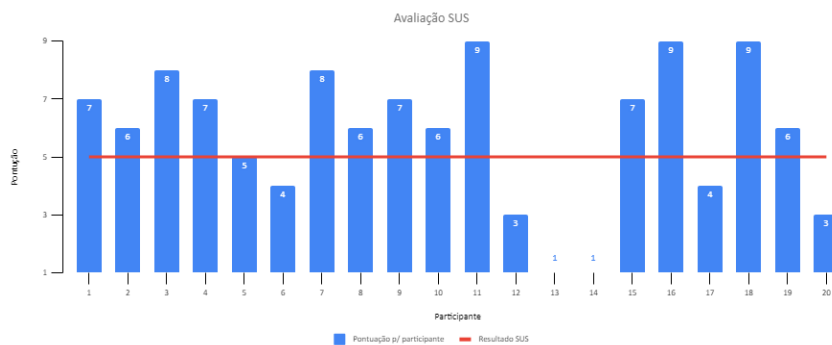
Em outros quesitos do teste **SAM** e até avaliações do teste **SUS** houve discrepância nos resultados dessas duas turmas, diferentemente do quesito de **Motivação**. Contudo, mesmo as turmas tendo a mesma média, é um número insatisfatório, que indica que mudanças precisam ser feitas para que os usuários se sintam mais motivados a utilizar a interface do *opCoders Judge*.

A respeito do **Sentimento de Controle**, último requisito avaliado pelo teste **SAM**, os alunos de **COM-T1** obtiveram uma média de **7,8**, tendo apenas uma resposta neutra e nenhuma negativa, assim como mostra os dados na Figura 3.6.

Na Figura 3.6 também é possível ver as diferenças em relação à turma **PC-T1**. A turma obteve uma média de **7,0**, além das respostas positivas, houve uma negativa e quatro neutras. Um resultado decente, contudo, há uma diferença significativa de **0,8**.



(a) COM-T1.



(b) PC-T1.

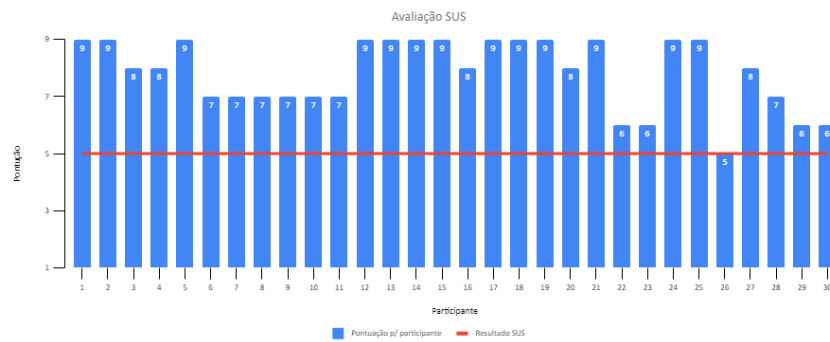
Figura 3.5 – Resultado do SAM sobre Motivação.

3.1.1.3.3 Perfil dos Voluntários

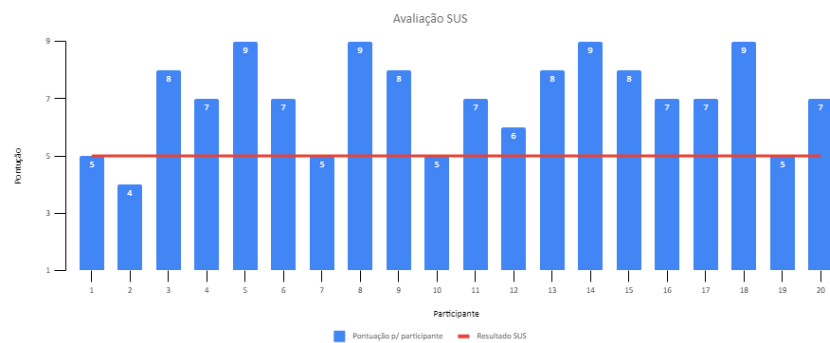
É importante entender melhor sobre o perfil dos voluntários que estão realizando o teste, pois isso ajuda a traçar padrões nos dados analisados, assim como auxilia na interpretação de algumas informações.

O teste de perfil foi elaborado com diversas questões que ajudavam a entender mais sobre os conhecimentos prévios do voluntário, tal como o contexto que está inserido e também sobre a performance do teste. As questões que compunham o teste de perfil consistem em:

- Tempo de duração da execução do teste;
- Gênero;
- Curso;
- Semestre da faculdade;
- Se já usou algum corretor antes do teste;
- Se gosta de programar;
- Se já programava antes da faculdade;
- Se é a primeira vez que está cursando a disciplina;



(a) COM-T1.



(b) PC-T1.

Figura 3.6 – Resultado do SAM sobre Sentimento de Controle.

- Considerações sobre o uso da plataforma;

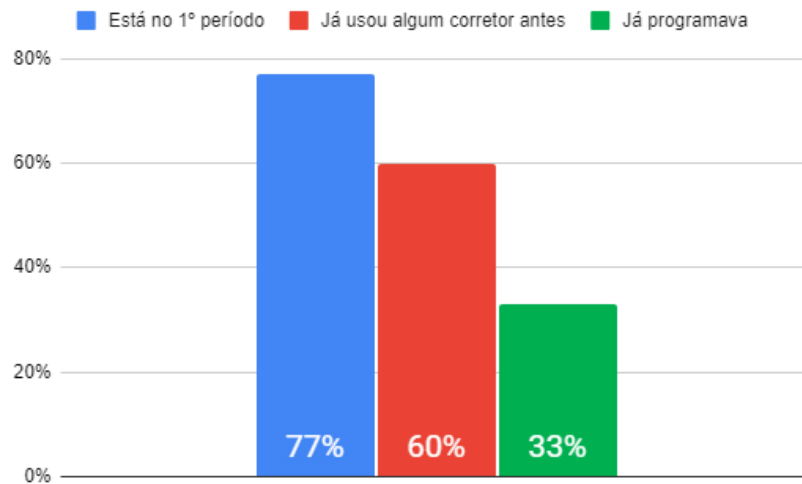
As questões que foram mais interessantes, para o objetivo desta investigação, na análise do teste de perfil foram:

- Parcela dos voluntários que estão no primeiro período;
- Parcela dos voluntários que já usaram algum corretor online de código-fonte antes;
- Parcela dos voluntários que gostam de programar;
- Parcela dos voluntários que já programavam antes de entrar na faculdade.

Esses foram os critérios considerados mais impactantes na avaliação dos usuários dos testes **SAM** e **SUS**. Pois um aluno no primeiro período indica que ainda pode ser leigo nas ferramentas utilizadas pelo curso. É intuitivo pensar que se o voluntário já usou algum corretor antes, ele terá mais familiaridade com a ferramenta, pois embora as interfaces sejam diferentes, ainda seguem a mesma proposta, por isso essa é considerada a questão mais impactante nos desempenhos dos testes. Se um voluntário gostar de programar e já tiver uma experiência nisso, é possível que ele tenha tido contato com outras ferramentas desse meio e é válido de se pensar que o padrão adotado para as interfaces dos corretores de código-fonte online, também esteja presente em outras ferramentas do contexto da programação.

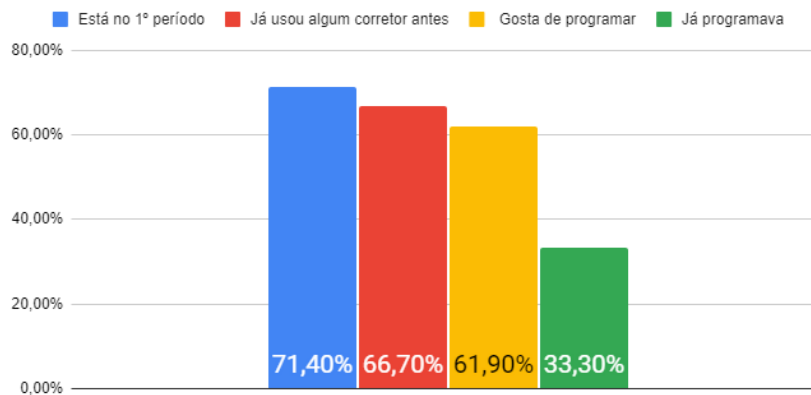
A análise do tempo que cada voluntário demorou para executar é interessante, contudo não foi considerado tão viável na análise, pois no segundo teste executado, os próprios voluntários que cronometraram o tempo. Dessa forma, havia a possibilidade de margem de erro na marcação do tempo.

Perfil - Introdução a Programação



(a) COM-T1.

Perfil - Programação de Computadores I



(b) PC-T1.

Figura 3.7 – Resultado do teste de Perfil.

A Figura 3.7 dispõe as informações de destaque no teste de perfil de **COM-T1** e **PC-T2**. A respeito do primeiro grupo, primeiramente é importante ressaltar que, a questão se o voluntário gosta de programar foi inserida nos testes desta monografia, dessa forma não há esse dado disponível nesse em **COM-T1**. Mas é possível perceber que na turma de voluntários **77%** deles estavam no primeiro período e apenas **33%** já programavam antes de entrar na faculdade. É interessante observar que mesmo com uma turma, majoritariamente composta por iniciantes em programação, **60%** já havia utilizado algum corretor antes do teste, isso indica que alunos de **Introdução a Programação** tem boas chances de já estarem familiarizados com uma plataforma similar ao *opCoders Judge*.

A respeito do teste **PC-T1**, os dados são relativamente similares. **71,40%** dos alunos que participaram estavam no primeiro período da faculdade, **66,70%** já haviam utilizado um corretor antes, **61,90%** afirmaram que gostam de programar e **33,30%** já programavam antes de ingressar na faculdade, como é possível observar na Figura 3.7. É complicado de se analisar esses dados, pois se trata de alunos de diferentes cursos condensados, contudo é possível afirmar que os dados são similares com os do teste anterior e que mais da metade dos voluntários gosta de programar.

3.1.2 Segundo Teste de usabilidade

Foi constatado que uma nova bateria de testes pudesse, dessa vez com um volume bem maior de voluntários, trazer mais resultados para o experimento. Dessa forma, um segundo teste foi realizado, para descrevê-lo, será utilizado a mesma organização de seção. A Subseção 3.1.2.1 explicará os critérios de avaliação, a 3.1.2.2 relatará os moldes em que o teste foi realizado e finalmente, a Subseção 3.1.2.3 discorrerá sobre os resultados obtidos.

3.1.2.1 Métodos de Avaliação

A respeito dos métodos e critérios de avaliação, não houve mudanças do primeiro teste para o segundo. Os voluntários foram submetidos à mesma tarefa, utilizando também o **Figma** para aplicação do teste. Após a realização da tarefa, os mesmos foram submetidos aos 3 questionários aplicados anteriormente, **SUS**, **SAM** e de **Perfil**. A fim de simplificar o processo, os questionários foram fornecidos aos voluntários do teste de maneira impressa. Para facilitar a coleta de informações e evitar possíveis complicações, optou-se por entregar os formulários impressos aos participantes.

É importante ressaltar, que o segundo teste foi conduzido no semestre subsequente ao primeiro, já após a coleta e análise dos dados iniciais. Com base nos resultados obtidos no primeiro teste, foram realizadas algumas modificações no protótipo proposto por Cedraz (2023). As mudanças consistiam basicamente em alterações estéticas e serão melhor explicadas na Seção de 3.2. Dessa forma, ao executar o segundo teste, optou-se por utilizar o protótipo já modificado, o que funcionou como uma validação das alterações implementadas.

3.1.2.2 Disposição do Teste

Foi observado que a abordagem de análise individual de cada voluntário foi o principal fator determinando do baixo número de participantes. Isso se deve pois, essa análise individual demanda muito tempo, e isso prejudica bastante a realização dos testes, uma vez que a aplicação deles é realizada no começo do período letivo, nas primeiras aulas, antes das turmas começarem a utilizar da plataforma oficialmente.

Para a condução desta nova série de testes, os moldes da abordagem proposta por Cedraz (2023) foram levemente ajustados. Optou-se por selecionar três turmas da disciplina de

Programação de Computadores I, cada uma composta por alunos de cursos distintos, porém com algum curso predominante, sendo eles **Engenharia Mecânica** na primeira, **Engenharia de Controle de Automação** na segunda e **Arquitetura e Urbanismo** na terceira. A principal modificação consistiu na aplicação dos testes para uma turma completa de uma vez, visando ampliar o alcance do estudo e resolver questões temporais mencionadas anteriormente. Como essa abordagem possui algumas diferenças, a comparação com o teste com os alunos de **Introdução à Programação**, do curso de **Ciência da Computação**, feito por Cedraz (2023), podia ser complicada. Por esse motivo, foi acrescentado uma quarta turma para essa nova séries de testes, a turma da mesma disciplina testada por Cedraz (2023), **Introdução à Programação**.

Desta forma, foi realizado o teste com 118 voluntários no total, sendo eles 31 da turma de **Introdução a Programação**; 27 da turma majoritariamente composta por alunos da **Engenharia Mecânica**; 26 da turma de **Engenharia de Controle de Automação** e por fim, 34 da turma, em sua maioria, de alunos de **Arquitetura e Urbanismo**.

Entretanto, alguns pontos são importantes de ressaltar, com essa abordagem, a análise individual do percurso de cada voluntário durante o teste não foi possível, por conta disso, o tempo foi cronometrado pelos próprios voluntários que estavam participando do teste, seguindo orientações dadas antes da execução.

3.1.2.3 Resultados

Aqui serão abordados detalhes sobre a execução e resultado do segundo teste. Este que como mencionado acima, utiliza dos mesmos 3 questionários **SUS**, **SAM** e de **Perfil**. Os resultados discorridos aqui dizem respeito ao teste com **COM-T2** (segundo teste da disciplina de **Introdução a Programação**), **MEC-T2** (**Engenharia Mecânica**), **CAU-T2** (**Engenharia de Controle de Automação**) e **AUR-T2** (**Arquitetura e Urbanismo**).

3.1.2.3.1 SUS

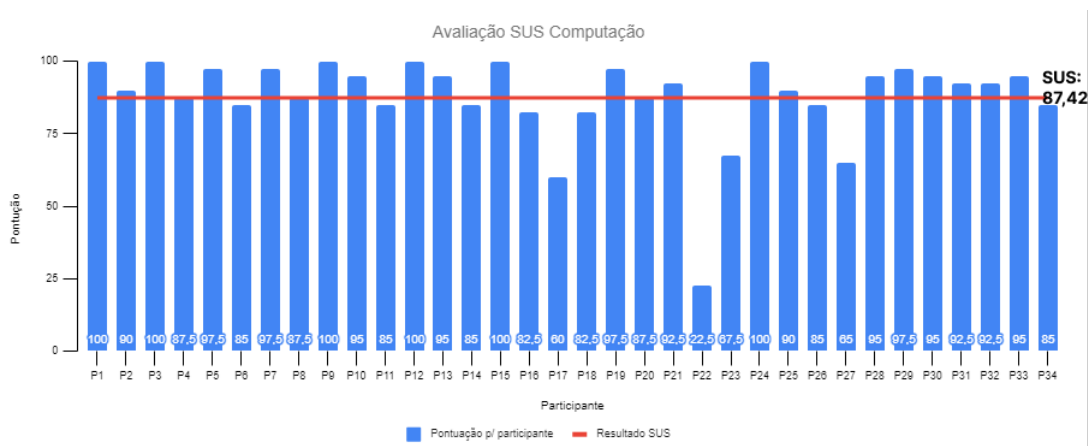


Figura 3.8 – Resultado geral do teste SUS - COM-T2.

É possível afirmar que os resultados do teste geral do **SUS** ficaram entre os valores dos resultados dos testes **COM-T1** e **PC-T1**. Como é possível observar na Figura 3.8, **COM-T2** teve o maior resultado, atingindo um valor de **87,42**. Houveram poucas avaliações baixas, em destaque para a menor, com um valor de **22,5**.

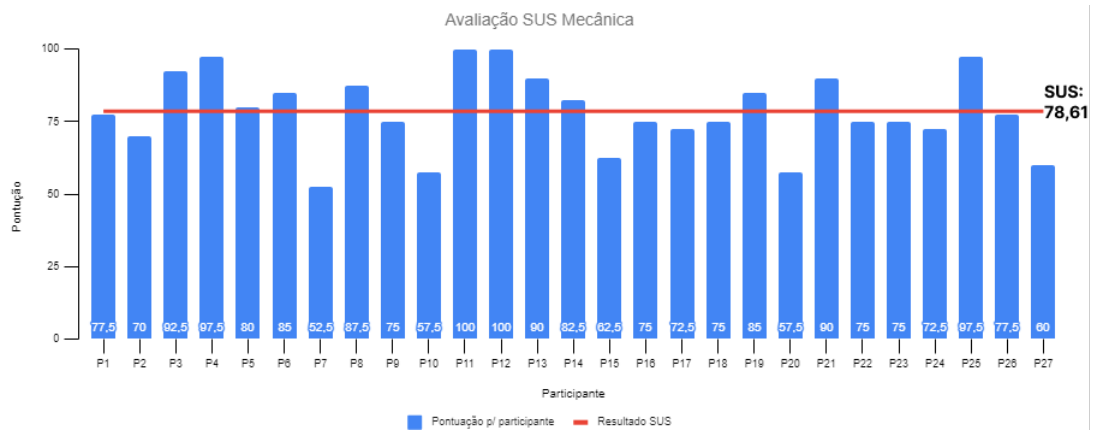


Figura 3.9 – Resultado geral do teste SUS - MEC-T2.

Em seguida, a respeito de **MEC-T2**, seu resultado geral foi de **78,61**, como mostrado na Figura 3.9. Ainda que melhor do que o resultado de **PC-T1**, foi o pior resultado neste segundo teste. Além disso, também é válido ressaltar que foi bastante inferior em relação ao resultado de **COM-T1**.

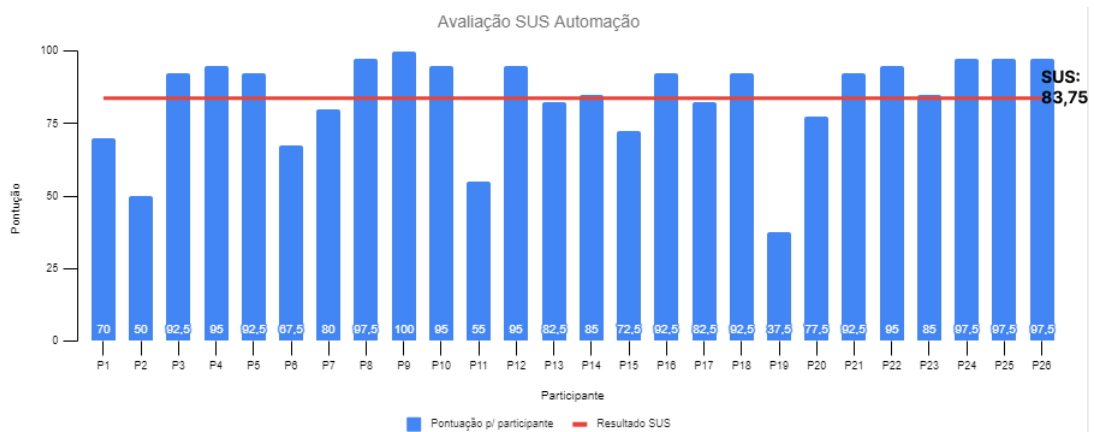


Figura 3.10 – Resultado geral do teste SUS - CAU-T2.

A turma que alcançou o segundo melhor resultado, nesta série de testes, foi a turma composta, majoritariamente, por alunos de **Engenharia de Controle de Automação**, pois o resultado de **CAU-T2** alcançou **83,75**, devidamente ilustrado na Figura 3.10.

Por fim, conforme ilustrado na Figura 3.11, **AUR-T2** alcançou um resultado de **80,15**, sendo a terceira maior avaliação deste segundo teste aplicado.

É possível observar que o teste com alunos da computação, no segundo teste, tiveram um resultado excelente, superando até mesmo o resultado do teste de Cedraz (2023). Colaborando com a suspeita de que os alunos de **Programação de Computadores I** tem um pouco mais

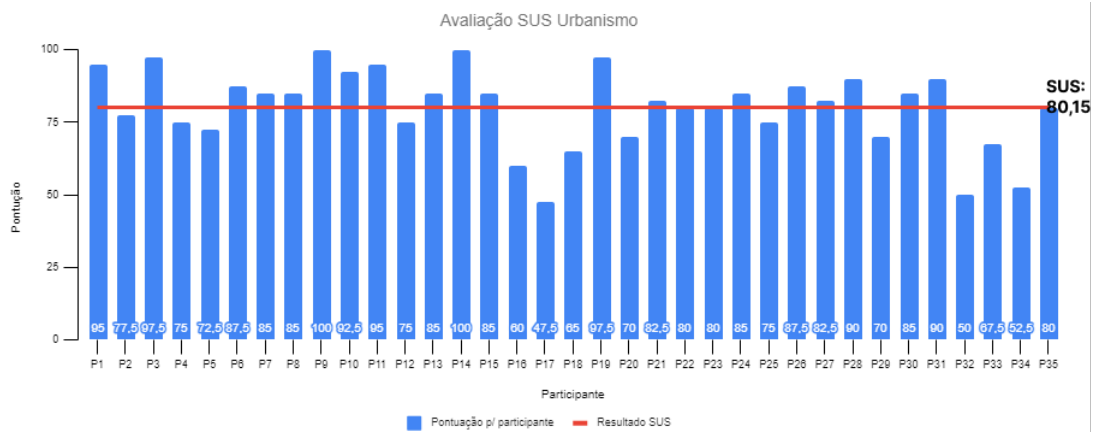


Figura 3.11 – Resultado geral do teste SUS - AUR-T2.

de dificuldade com o *opCoders Judge* do que os alunos da computação, todas as turmas dessa disciplina tiveram resultados inferiores.

Fazendo a análise agora das questões positivas separadamente e observando a Figura 3.12, é possível notar que **COM-T2** teve uma média **4,42**, exatamente a mesma média que **COM-T1**. O menor resultado foi **4,1** e o maior **4,6**, também varia pouco de uma para a outra.

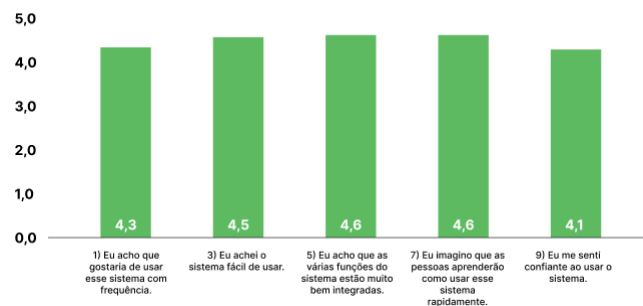


Figura 3.12 – Resultado questões positivas do teste SUS - COM-T2.

De acordo com a Figura 3.13, a média dos resultados de **MEC-T2** foi de **4,06**. O menor resultado sendo **4** e o maior **4,2**, foi a turma que variou menos entre essas notas, apenas **0,2** de diferença. É interessante ressaltar que, essa turma teve o pior resultado geral de avaliação do **SUS**, contudo não obteve nenhuma nota abaixo de **4**, o que indica que, nas questões positivas, não houve nenhum resultado que se destacasse.

A turma do teste **CAU-T2** alcançou uma média de **4,26**, o segundo melhor resultado até aqui. A questão com menor avaliação correspondeu a **3,9** e a maior com **4,7**, uma diferença de **0,8**, a mais significativa até o momento. Conforme ilustrado na Figura 3.14.

Por fim, observando a Figura 3.15, a turma de **AUR-T2** alcançou uma média de **4,2**. Avaliando cada questão separadamente é possível notar que os resultados dessa turma foram muito semelhantes com **CAU-T2**, sendo que não houve uma diferença maior que **0,1** em nenhuma questão. Isso também pode ser observado pelo valor semelhante de médias.

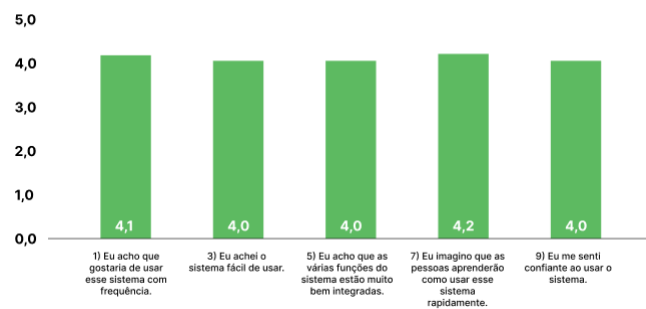


Figura 3.13 – Resultado questões positivas do teste SUS - MEC-T2.

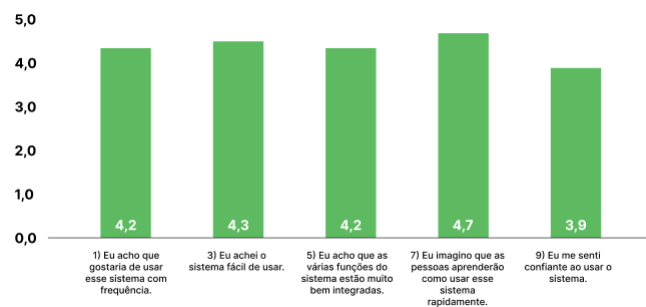


Figura 3.14 – Resultado questões positivas do teste SUS - CAU-T2.

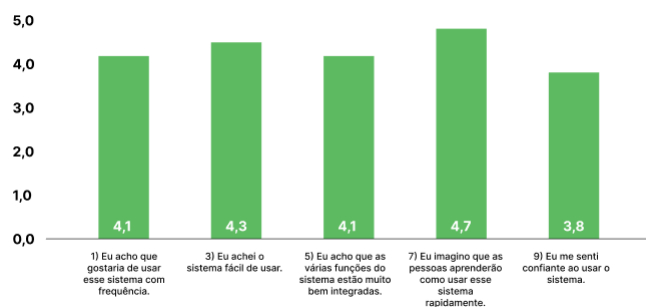


Figura 3.15 – Resultado questões positivas do teste SUS - AUR-T2.

É importante ressaltar que, considerando todos os testes realizados nesta monografia e na de Cedraz (2023), a questão que obteve o menor resultado foi a número 9 – “**Eu me senti confiante ao usar o sistema**”, o que pode indicar que em aspectos gerais a plataforma precisa de melhorias para ser mais intuitiva. Já a questão que obteve a maior nota em todos os resultados foi a 7 – “**Eu imagino que as pessoas aprenderão como usar esse sistema rapidamente**”, o que é um bom indicativo que o sistema está organizado de maneira simples e a curva de aprendizagem é curta.

Já a respeito dos resultados das questões negativas referentes ao segundo teste. Conforme a Figura 3.16, COM-T2 obteve uma média de 1,43, um resultado melhor do que a mesma turma em COM-T1, por uma diferença de 0,9. A questão com o melhor resultado foi de 1,3, enquanto a pior 1,6.

Como é ilustrado na Figura 3.17, MEC-T2 teve uma média de 1,88, em suas questões

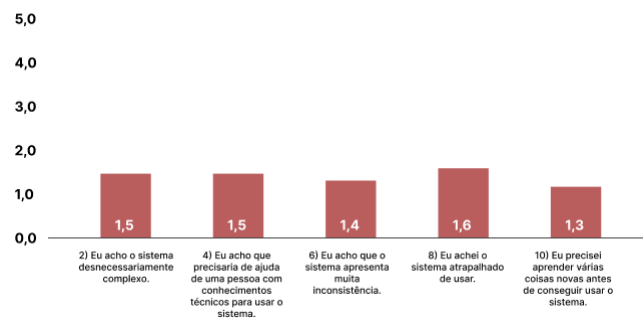


Figura 3.16 – Resultado questões negativas do teste SUS - COM-T2.

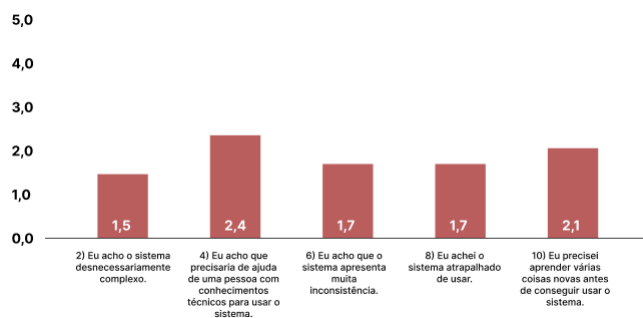


Figura 3.17 – Resultado questões negativas do teste SUS - MEC-T2.

negativas, um resultado ligeiramente melhor que **PC-T1**, contudo o pior de todas as turmas do segundo teste. O questão que obteve o maior valor alcançou um resultado de **2,4** e a menor **1,5**, uma diferença significativa de **0,9** entre elas.

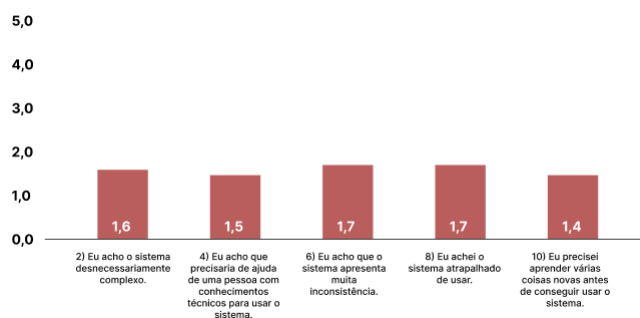


Figura 3.18 – Resultado questões negativas do teste SUS - CAU-T2.

Já **CAU-T2** alcançou uma média de **1,58**, bem próxima de **COM-T1**, contudo ligeiramente pior. A questão que obteve o melhor resultado foi de **1,4** e a pior **1,7**, como é possível de se observar pela Figura 3.18.

Por fim, a turma do teste **AUR-T2** atingiu uma média de **1,78**. Sendo que, nos resultados, o maior valor alcançado por uma questão foi de **2,4** e o menor **1,5**, conforme ilustrado pela Figura 3.19.

Fazendo uma análise em todos os gráficos, é possível notar que, de maneira geral, a questão que teve os piores desempenhos foi a 4 – “**Eu acho que precisaria de ajuda de uma**

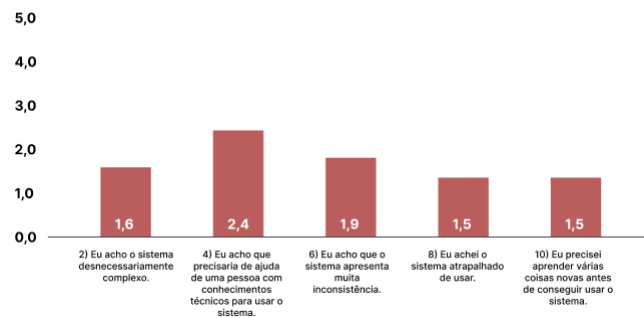


Figura 3.19 – Resultado questões negativas do teste SUS - AUR-T2.

pessoa com conhecimentos técnicos para usar o sistema”. Isso é um problema sério, pois a ideia do *opCoders Judge* é que a plataforma tenha uma interface simples e intuitiva, em que qualquer tipo de usuário consiga navegar e a parte dos conhecimentos técnicos fosse reservado apenas para o momento da criação da resposta de uma questão, o que não fez parte dos testes de usabilidades, o arquivo de código virtual foi fornecido aos voluntários no momento do teste.

3.1.2.3.2 SAM

Agora será relatado sobre os resultados de **Satisfação** para as turmas do segundo teste desta monografia. **COM-T2** a média dos resultados foi de **8**, sendo que apenas 1 voluntário avaliou com nota neutra e um total de 13 avaliaram com nota máxima. Tais dados são comprovados pela Figura 3.20 e é válido ressaltar esta turma alcançou a mesma nota que **COM-T1**.

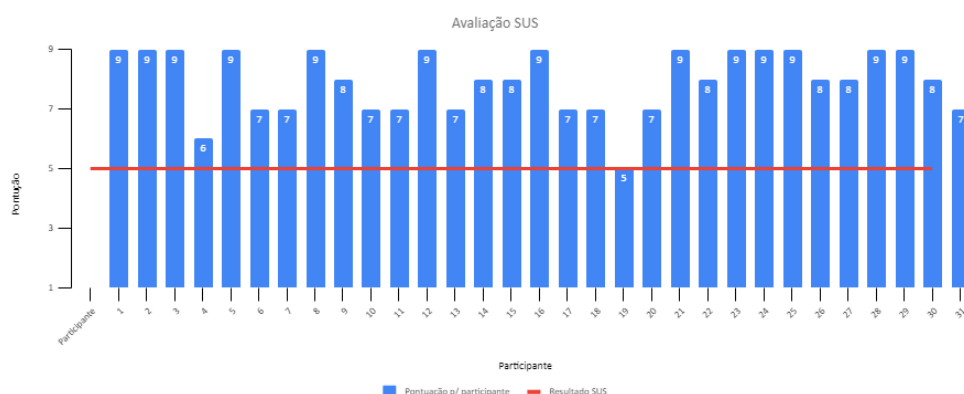


Figura 3.20 – Resultado de Satisfação no teste SAM - COM-T2.

A turma de **MEC-T2**, como é possível observar na Figura 3.21, obteve 3 avaliações com nota neutra e alcançou uma média de **7,7**. Ainda que positivo, um resultado levemente inferior a **COM-T1** e **COM-T2** e ligeiramente superior à turma de a **PC-T1**.

A turma de **CAU-T2** teve um resultado inesperado. Até o momento, em todos os testes estava tendo resultados melhores do que as outras turmas de **Programação de Computadores I**, do primeiro e segundo testes. Contudo, a média alcançada, em relação à **Satisfação**, foi de **7,4**, ainda que um resultado bom, foi o pior resultado de todas as turmas testadas. Como pode ser

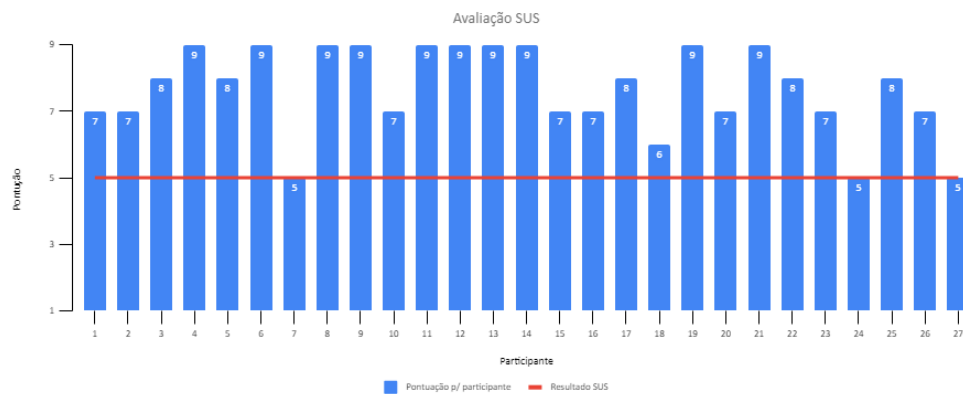


Figura 3.21 – Resultado de Satisfação no teste SAM - MEC-T2.

observado na Figura 3.22, obteve 5 avaliações com nota neutra e somente 7 voluntários avaliaram com nota máxima.

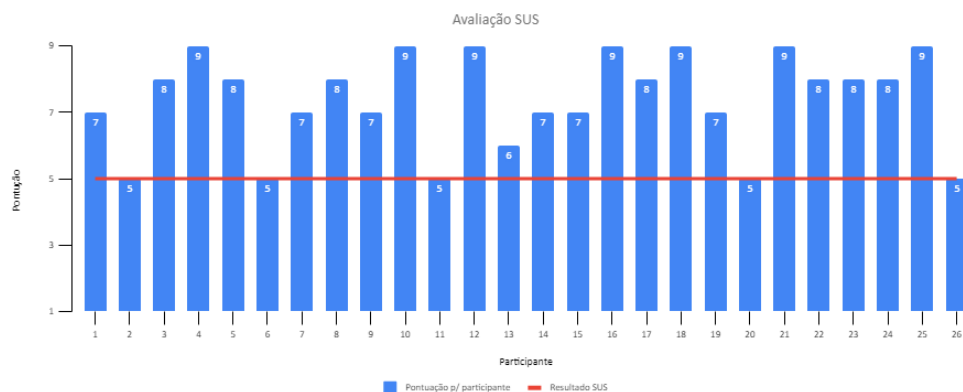


Figura 3.22 – Resultado de Satisfação no teste SAM - CAU-T2.

Já **AUR-T2**, mesmo tendo uma média superior à turma de **CAU-T2**, alcançando **7,6**, foi a única turma que obteve uma nota negativa, como mostrado na Figura 3.23. Contudo, obteve um número de 15 voluntários que avaliaram com nota máxima e 5 com nota neutra. É possível observar, a partir disso, que houve uma discrepância considerável nas avaliações. É válido ressaltar que a média foi a mesma da turma de **PC-T1**.

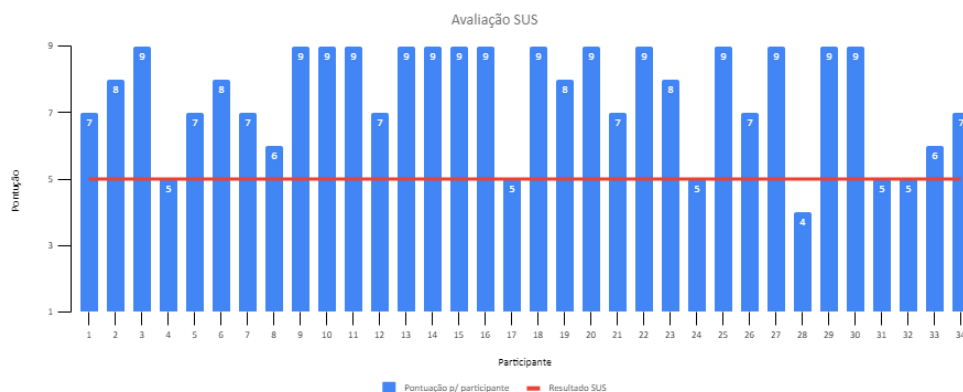


Figura 3.23 – Resultado de Satisfação no teste SAM - AUR-T2.

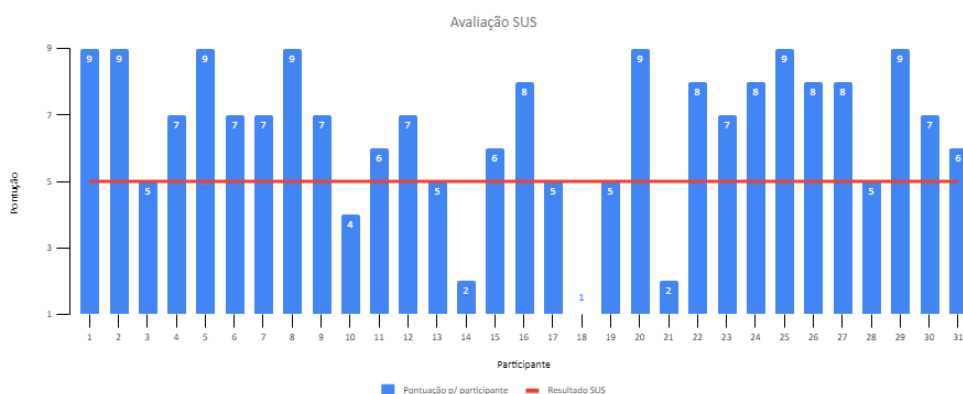


Figura 3.24 – Resultado de Motivação no teste SAM - COM-T2.

Sobre os resultados de **Motivação**, a turma **COM-T2** alcançou uma média bem superior as outras duas, atingindo um valor de **6,6**, ainda não é o ideal, porém esse número representa uma melhora significativa. Como mostrado na Figura 3.24, houveram 4 voluntários que avaliaram com uma nota negativa, 5 com nota neutra e 7 com nota máxima.

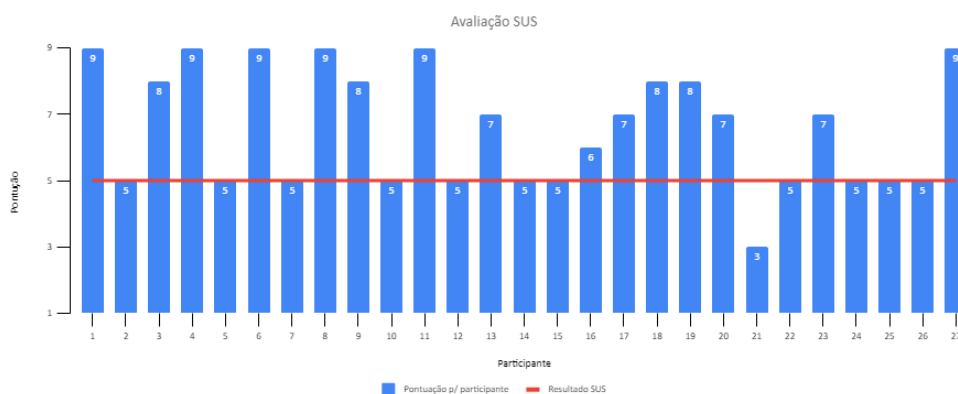


Figura 3.25 – Resultado de Motivação no teste SAM - MEC-T2.

O resultado da turma **MEC-T2** foi uma surpresa positiva, esta turma não estava tendo resultados tão bons quanto as ambas as turmas **COM-T1** e **COM-T2**, contudo em relação à **Motivação** os resultados deste teste, alcançou uma nota de **6,6**, também representando uma melhora significativa em relação aos testes de **COM-T1** e **PC-T1** e a mesma nota da turma de **COM-T2**. Utilizando os dados da Figura 3.25, é válido destacar que obtiveram apenas 1 avaliação negativa, 6 avaliações com nota máxima e impressionantes 11 avaliações com nota neutra, o maior número até agora.

As outras duas turmas restantes não obtiveram melhoras expressivas em relação a **COM-T1** e **PC-T1**. **CAU-T2** alcançou uma média de **5,8**, sendo que 6 avaliações foram negativas, sendo 1 delas com nota mínima, 5 com nota neutra e apenas 3 com nota máxima, conforme ilustrado na Figura 3.26.

Por fim, a turma de **AUR-T2** obteve uma média de **6**, sendo que 6 dessas avaliações foram negativas, com 3 delas tendo nota mínima, 10 avaliações neutras e 9 notas máximas. Houve uma

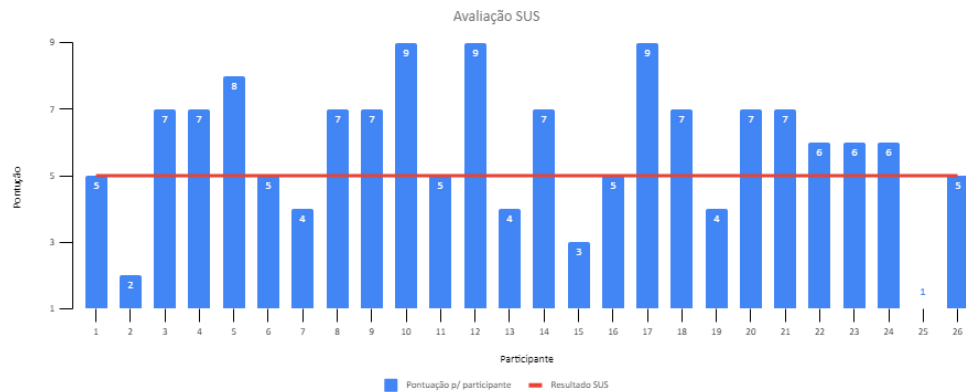


Figura 3.26 – Resultado de Motivação no teste SAM - CAU-T2.

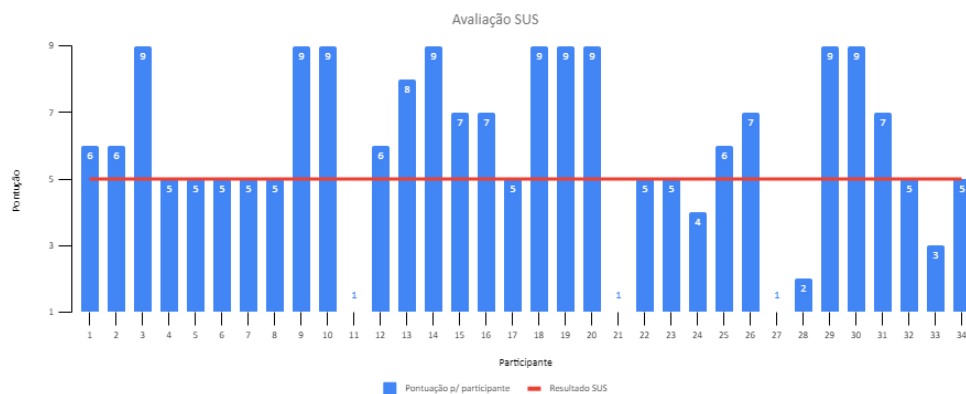


Figura 3.27 – Resultado de Motivação no teste SAM - AUR-T2.

grande discrepância em relação às avaliações dos voluntários, como mostrado na Figura 3.27.

Embora todas as avaliações do segundo teste, que já contava com alterações, tenham sido superiores, era esperado um resultado melhor dos testes. A maior valor alcançado foi de **6,6**, um número que não representa um resultado ruim, contudo, para o *opCoders Judge* é almejado obter resultados melhores.

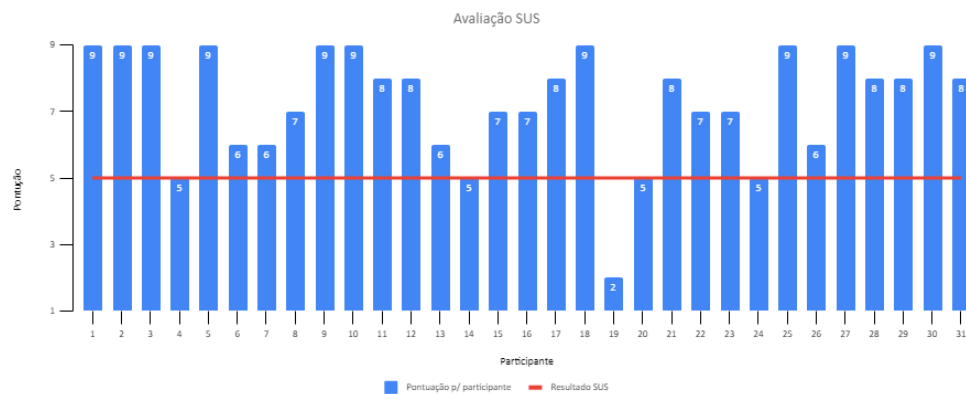


Figura 3.28 – Resultado de Sentimento de Controle no teste SAM - COM-T2.

Analisando os resultados da perspectiva de **Sentimento de Controle**. A respeito da turma de **COM-T2**, a média alcançada foi de **7,3**, um número que está entre as duas turmas

anteriores, sendo **0,5** menor do que o teste **COM-T1**. Como mostrado na Figura 3.28, houve apenas 1 avaliação negativa, 4 avaliações neutras e 10 com nota máxima entre os voluntários.

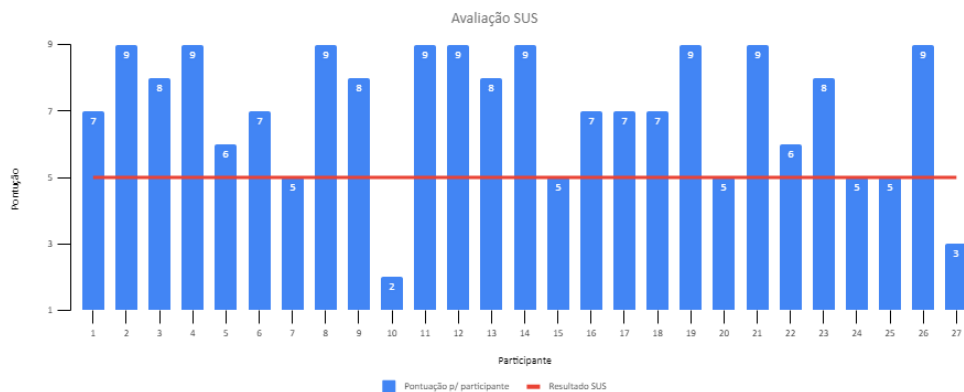


Figura 3.29 – Resultado de Sentimento de Controle no teste SAM - MEC-T2.

A turma de **MEC-T2** obteve um resultado ligeiramente menor, alcançando uma média de **7**, mesmo valor da turma de **PC-T1**, no primeiro teste. Dentre as avaliações dos voluntários, 2 foram negativas, 5 foram neutras e 9 foram com nota máxima, como mostrado na Figura 3.29.

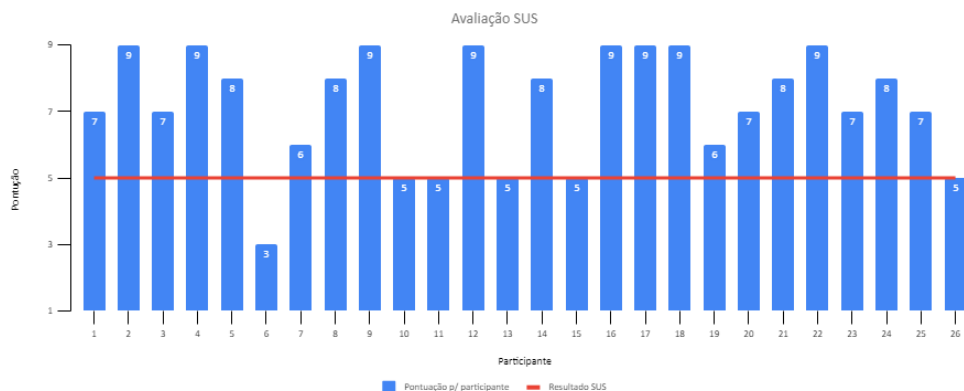


Figura 3.30 – Resultado de Sentimento de Controle no teste SAM - CAU-T2.

Com resultados bem próximo das duas turmas anteriores, **CAU-T2** alcançou uma média de **7,2**. A Figura 3.30 evidencia que houveram apenas 1 nota negativa, 4 neutras e 8 máximas.

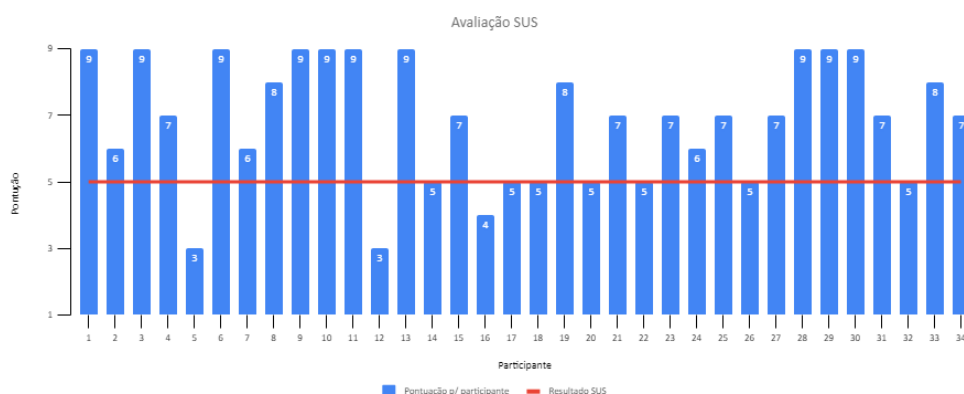


Figura 3.31 – Resultado de Sentimento de Controle no teste SAM - AUR-T2.

Por fim, a turma de **AUR-T2** atingiu uma média de **6,9**, o pior resultado de todos os testes, ainda que não haja tanta discrepância nos resultados. A respeito das avaliações dos resultados vale a pena destacar que houveram 3 notas negativas, 7 neutras e 10 com nota máxima, conforme ilustrado na Figura 3.31.

Ainda que as turmas da disciplina de **Programação de Computadores I**, do segundo teste, tenham realizado os mesmos com alterações no protótipo, que visavam melhorar a perspectiva do usuário, todas as turmas obtiveram médias inferiores em relação ao teste de **COM-T1** e, com exceção de **AUR-T2**, iguais ou superiores aos alunos de **PC-T1**.

3.1.2.3.3 Perfil

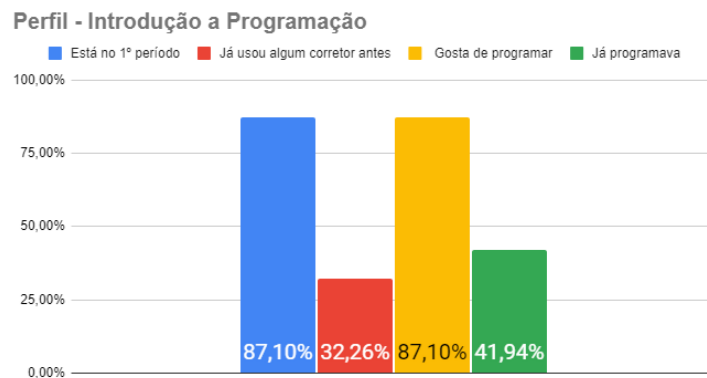


Figura 3.32 – Resultado do teste de perfil - COM-T2.

A respeito das informações do segundo teste, **COM-T2**, observando a Figura 3.32, é perceptível que **87,10%** dos alunos estavam no primeiro período, um número consideravelmente maior do das outras turmas anteriores. Além disso, somente **32,26%** dos voluntários já haviam utilizado um corretor de código-fonte online, aproximadamente metade das turmas anteriores. **87,10%** dos alunos gostam de programar e **41,94%** já programa antes da faculdade.

É intuitivo de se pensar que por ser uma turma menos familiarizada com a utilização de corretor de código-fonte online, os resultados seriam piores, mas esta turma obteve os melhores resultados nos testes **SUS** e **SAM**. Uma justificativa para isso são o grande valor nas questões “Gosta de programar”, pois isso sugere que o voluntário está familiarizado com outras ferramentas no meio da programação e estas ferramentas podem ter uma interface similar com a do *opCoders Judge*.

A análise do segundo teste das turmas da disciplina de **Programação de Computadores I**, é mais interessante por conter as informações separadas em turmas, logo é mais fácil de se traçar um perfil. Nos resultados da turma de **MEC-T2**, ilustrados na Figura 3.33, foi constatado que **74,07%** dos alunos estavam no primeiro período da faculdade e apenas **7,41%** já haviam usado algum corretor antes, um número bastante baixo. Além disso, **81,48%** afirmam que gostam de programar e somente **11,11%** já programavam antes da faculdade.

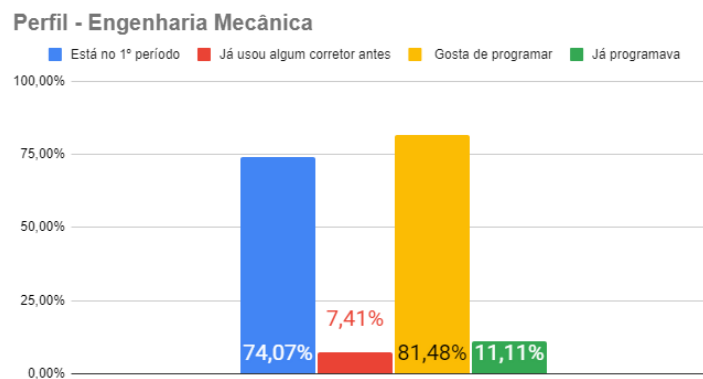


Figura 3.33 – Resultado do teste de perfil - MEC-T2.

É interessante observar que os alunos desta disciplina são os que menos utilizaram um ambiente de código-fonte online, isto pode explicar o motivo deles terem obtido os piores resultados nos testes **SUS** e **SAM**.

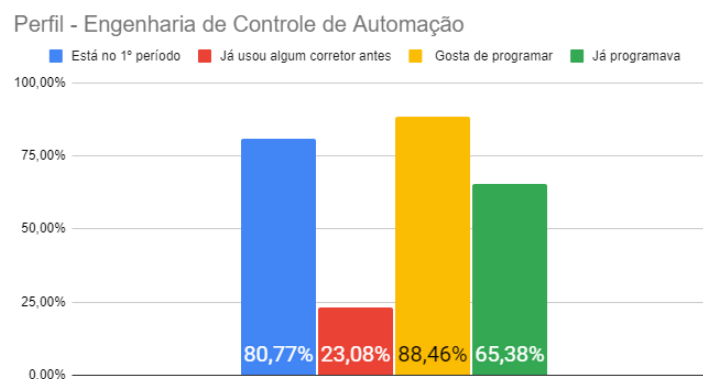


Figura 3.34 – Resultado do teste de perfil - CAU-T2.

Na Figura 3.34, é possível observar os resultados do teste de perfil da turma de **CAU-T2**. **80,77%** dos alunos estavam no primeiro semestre da faculdade, **23,08%** deles já haviam utilizado um corretor de código-fonte online, **88,46%** gostam de programar e **65,38%** já programava antes.

Uma informação importante desses resultados, é que nas duas últimas questões esta turma teve os maiores valores de todas as turmas analisadas, isso pode colaborar para ela ter tido os melhores resultados nos testes **SUS** e **SAM**, entre as turmas da disciplina de **Programação de Computadores I**.

Por fim, na análise dos resultados da turma de **AUR-T2**, temos que **85,29%** dos alunos estavam no primeiro período, **26,47%** deles já haviam utilizado algum corretor de código-fonte online, **67,65%** gosta de programar e somente **8,82%** já programava antes de ingressar à faculdade.

Nos testes **SUS** e **SAM**, os resultados de **AUR-T2** com **CAU-T2** foram similares, porém um pouco piores, uma perspectiva que pode justificar isso é que eles tem resultados parecidos na

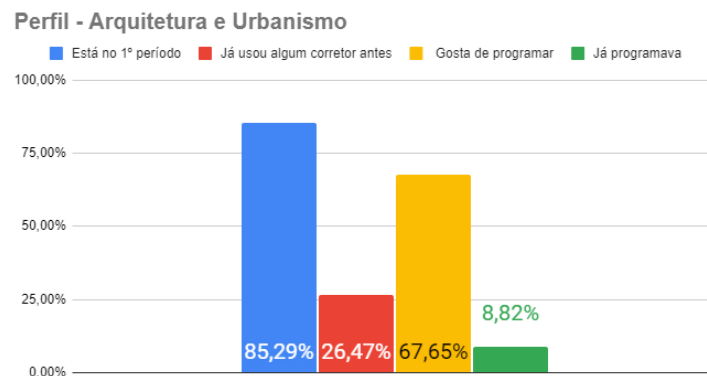


Figura 3.35 – Resultado do teste de perfil - AUR-T2.

questão de “Já usou algum corretor antes”, contudo nas questões de “Gosta de programar” e “Já programava” os alunos tiveram valores mais baixos.

3.1.3 Discussão

É interessante comparar os testes **COM-T1** e **PC-T1**, pois eles utilizaram o exato mesmo protótipo. As avaliações demonstraram uma clara diferença entre as perspectivas dos usuários, sendo que os do primeiro grupo obtiveram resultados melhores em todos os aspectos de ambos os testes. Esse fato colabora fortemente para a premissa de que os alunos das turmas da disciplina de **Programação de Computadores I** tem mais dificuldade de interagir de **Introdução a Programação**, disciplina do curso de **Ciência da Computação**.

A respeito do segundo teste, foram constatadas diferenças entre as 4 turmas. A turma de **COM-T2**, obteve resultados significativamente melhores em quase todos os requisitos, em nenhuma análise outra turma teve um resultado melhor, contudo **MEC-T2** teve o mesmo resultado em relação à **Motivação** no teste **SAM**. Portanto, é mais um ponto que apoia a premissa dos testes.

É interessante constatar também, nos resultados do segundo teste aplicado, que é possível separar as 4 turmas em 3 grupos de dificuldade. A turma de **COM-T2** obteve os melhores resultados, tendo um resultado insatisfatório somente em relação à **Motivação** no teste **SAM**, com isso podemos definir que este grupo teve uma interação muito boa com a plataforma. O segundo grupo é composto pelas disciplinas **CAU-T2** e **AUR-T2**, pois ambas as disciplinas obtiveram resultados semelhantes, sendo que a primeira teve avaliações um pouco melhores, ainda assim, ambas as turmas tiveram uma boa interação com a plataforma, tendo poucos pontos preocupantes. Por fim, o último grupo, é composto somente pela turma de **MEC-T2**, com exceção dos resultados de **Motivação**, todas as avaliações foram menores do que as outras 3 turmas, com isso podemos afirmar que este tipo de usuário é o que tem mais dificuldade de interação com a plataforma e as alterações a se fazer tem que se basear fortemente neste grupo.

Fazendo uma análise mais profunda de cada teste, algumas constatações ficam evidentes.

Primeiro será percorrido sobre o teste **SUS**, todos os resultados deste teste podem ser visualizados na Tabela 3.1. É importante lembrar que as notas das questões positivas e negativas vão de 1 a 5, sendo que elas têm sentidos opostos, quanto maior o valor do resultado para as questões positivas, melhor ele é, para as negativas, quanto menor, mais ideal.

Turma	Geral	Positivas	Negativas
COM-T1	86,06	4,42	1,52
PC-T1 I	75,11	3,92	1,92
COM-T2	87,42	4,42	1,43
MEC-T2	78,61	4,06	1,88
CAU-T2	83,75	4,26	1,58
AUR-T2	80,15	4,20	1,78

Tabela 3.1 – Resultados Teste SUS.

O protótipo proposto por Cedraz (2023), foi utilizado para os testes das duas primeiras turmas da tabela 3.1, sendo que o usuário que utiliza a plataforma, atualmente, consiste somente nos alunos de **Programação de Computadores I**, contudo o protótipo foi criado utilizando como base os resultados do teste com a turma de **Introdução a Computação**. A partir disso é possível evidenciar um problema, a diferença nos resultados gerais dessas duas turmas é de **10,95**, um valor difícil de se ignorar. Devido a isso, é intuitivo afirmar que alguns pensamentos e estratégias usados na elaboração do protótipo podem não ter sido adequados ao usuário atual do *opCoders Judge*.

Tendo em mente o problema levantado anteriormente, algumas mudanças de design no protótipo foram realizadas. É possível ver influência das mudanças nos resultados do segundo teste, ilustrado nas 4 últimas turmas da Tabela 3.1. Em relação às turmas de **COM-T1** e **COM-T2**, podemos notar que o resultado utilizando o segundo protótipo teve um ligeiro aumento de **0,82**, influenciado somente pela diferença nos resultados das questões negativas.

Já comparando os resultados da turma geral de **PC-T1**, com as turmas específicas dessa disciplina no segundo teste, podemos ver bastante variação. A turma que melhor se destacou nesse teste foi a de **CAU-T2**, superando o resultado da turma geral em **8,64**. Entretanto, a turma que menos se destacou foi a **MEC-T2**, apresentando uma diferença no resultado de **3,5**, esse valor ainda assim representa uma melhoria destacável.

Para a elaboração de um protótipo é importante considerar os usuários que apresentam mais dificuldade, visando atender a todo público-alvo, com isso em mente, embora o resultado de **MEC-T2** seja positivo, é interessante pensar em possíveis melhorias para aproximar ainda mais esse valor das turmas da computação.

A respeito do teste **SAM**, as seguintes análises podem ser feitas a partir dos dados da Tabela 3.2. Comparando os resultados das mesmas turmas por categoria, a respeito de **Satisfação**, não houve diferença entre as turmas **COM-T1** e **COM-T2**. As turmas da disciplina **Programação de Computadores I** também seguiu essa linha, todas elas tiveram resultados muito próximos.

Turma	Satisfação	Motivação	Controle
COM-T1	8,0	5,8	7,8
PC-T1 I	7,6	5,8	7,0
COM-T2	8,0	6,6	7,3
MEC-T2	7,7	6,6	7,0
CAU-T2	7,4	5,8	7,2
AUR-T2	7,6	6,0	6,9

Tabela 3.2 – Resultados Teste SAM.

Já em relação à **Motivação** houve algumas diferenças positivas. A turma de **COM-T2**, que utilizou o segundo protótipo, teve um resultado superior à turma **COM-T1**, que testou o primeiro protótipo, com uma diferença de **0,6**. A respeito das turmas da disciplina de **Programação de Computadores I**, **CAU-T2** obteve o mesmo resultado de **PC-T1**, contudo tanto **MEC-T2**, quanto **AUR-T2** tiveram resultados superiores, sendo que o primeiro grupo alcançou uma diferença positiva de **0,6**. É válido ressaltar que embora houve progresso, os valores do resultado ainda não são satisfatórios, sendo preciso melhorar.

Por fim, a respeito de **Sentimento de Controle**, é possível observar que de maneira geral os resultados não foram satisfatórios. Sobre as turmas de computação houve uma queda de **0,5** de **COM-T1** para **COM-T2** nos resultados. Já comparando **PC-T1** com as demais turmas de **Programação de Computadores I**, somente a turma de **CAU-T2** conseguiu um resultado levemente superior, **MEC-T2** e **AUR-T2** obtiveram um resultado igual ou pior.

É possível que a abordagem utilizada no segundo teste para atingir um volume maior de voluntários tenha influenciado negativamente esses resultados. Isso se dá, pois, uma vez que aplicado o teste de maneira individual, é possível garantir mais clareza na explicação, assim como o voluntário pode estar mais a vontade para tirar dúvidas. No segundo teste, as explicações e orientações foram feitas para a turma inteira de uma vez, assim talvez a compreensão tenha sido um pouco prejudicada, o que prejudicaria a avaliação a respeito de **Sentimento de Controle**.

Contudo, para o prosseguimento do desenvolvimento da interface do *opCoders Judge* é necessário interpretar que as alterações não foram efetivas, no quesito **Sentimento de Controle**, sendo preciso pensar em mais melhorias.

Dados os testes feitos e considerando que houve protótipos diferentes usados para avaliação, é possível traçar as seguintes ordenações decrescentes de resultados:

Primeiro Protótipo

1. COM-T1;
2. PC-T1;

Segundo Protótipo

1. COM-T2;
2. CAU-T2;
3. AUR-T2;
4. MEC-T2.

Na Tabela 3.3 estão disponibilizados os resultados de todas as turmas em relação a todas as questões, sendo elas:

- Questão 1: Está no 1º período;
- Questão 2: Já usou algum corretor antes;
- Questão 3: Gosta de programar;
- Questão 4: Já programava.

Turma	Questão 1	Questão 2	Questão 3	Questão 4
COM-T1	77%	60%	-	33%
PC-T1	71,40%	66,70%	61,90%	33,30%
COM-T2	87,10%	32,26%	87,10%	41,94%
MEC-T2	74,07%	7,41%	81,48%	11,11%
CAU-T2	80,77%	23,08%	88,46%	65,38%
AUR-T2	85,29%	26,47%	67,65%	8,82%

Tabela 3.3 – Resultados Teste Perfil.

É interessante de se observar que na Tabela 3.3, que a variação da **questão 1** foi a menor de todas, um indicativo de que todas as turmas testadas eram compostas, em sua maioria, por iniciantes em suas respectivas áreas.

Talvez os resultados mais interessantes do teste de perfil sejam em relação à **questão 2**, por haver uma grande discrepância entre as turmas do primeiro e segundo teste. Seria intuitivo de se pensar que as turmas que possuem poucos alunos que utilizaram um corretor de código-fonte, tivesse os piores resultados nos testes **SUS** e **SAM**, contudo isso não acontece para a turma de **COM-T2**, que obteve o melhor desempenho. Anteriormente foi levantada a hipótese de que os valores altos nas **questões 3 e 4** apoiam a boa performance nos testes. Contudo, essa justificativa possui um contraponto, os valores das **questões 3 e 4** da turma de **CAU-T2** são ainda maiores do que a dessa turma, porém os resultados no teste **SUS** e **SAM** não foram tão bons.

Como mencionado anteriormente, as **questões 3 e 4** têm o propósito de aprofundar a compreensão sobre a experiência de programação de um voluntário, partindo da premissa de

que alguém que tenha utilizado outras ferramentas no campo da computação pode ter maior familiaridade com a interface do *opCoders Judge*. Embora essa hipótese não seja totalmente confirmada por valores substancialmente altos nessas questões, os resultados ainda constituem indicativos significativos para apoiar essa premissa. Assim, é possível afirmar que, mesmo com pontuações mais elevadas nas **questões 3 e 4**, não há garantia de que a turma de **CAU-T2** seja mais familiarizada com uma interface de código-fonte online do que a turma de **COM-T2**. Além disso, na **Questão 2**, a segunda turma apresenta **9,18%** a mais em comparação com a primeira, o que pode justificar, em parte, os resultados superiores nos testes dessa turma.

No demais, os baixos valores encontrados a respeito das **questões 2 e 4**, podem justificar o motivo das turmas de **MEC-T2** e **AUR-T2** terem tido os piores resultados nos testes **SUS** e **SAM**.

Após a execução do primeiro teste foi perguntado a opinião dos voluntários sobre a plataforma. A grande maioria dos estudantes demonstraram descontentes com a interface do primeiro protótipo, do ponto de vista estético. Com isso, foi percebido que havia a necessidade de realizar algumas mudanças no protótipo, para deixar as páginas do *opCoders Judge* mais atraentes e satisfatórias de se interagir.

Além disso, o formulário de **perfil** do segundo teste havia um campo para saber a opinião dos voluntários a respeito da experiência com o *opCoders Judge*. A maioria dos comentários envolveu mensagens positivas a respeito da interface ser atraente, simples, prática e com potencial. Isso indica que as mudanças realizadas do primeiro protótipo para o segundo foram bem recebidas, é possível ver isso nos resultados dos testes também. Contudo, observando os resultados do teste **SAM**, a respeito de **Motivação**, é válido de se pensar que ainda é necessário melhorar diversos aspectos.

Além disso, durante a execução do teste foi percebido uma grande dificuldade dos voluntários, principalmente ao entender que havia conseguido cumprir o objetivo, como explicado anteriormente, o usuário entender que havia cumprido a tarefa fazia parte do desafio, tendo sido percebido que muitos usuários ficavam confusos ao terminar a tarefa e começavam a vagar pela plataforma buscando orientação. Alguns voluntários justificaram a dificuldade afirmando que não esperavam que a tarefa deles no teste fosse tão fácil. Com isso, mesmo que ligeiramente, foi percebido a necessidade de demonstrar com mais clareza para os usuários que suas tarefas haviam sido submetidas.

Com isso, após a análise os resultados de ambos os testes, foi confirmada a hipótese que os motivava, os alunos de **Programação de Computadores I** obtiveram resultados considerados inferiores aos das turmas da computação, dados suas respectivas métricas, em praticamente todas as avaliações feitas. A partir disso, novas alterações na interface do *opCoders Judge* tem de ser feito, pois, atualmente os alunos dessa turma são os usuários ativos da plataforma e o protótipo proposto por [Cedraz \(2023\)](#) baseou-se na perspectiva de alunos que tinham mais compreensão desse tipo de ambiente.

3.2 Alterações no Protótipo

Em seu trabalho, [Cedraz \(2023\)](#) criou um protótipo para o *opCoders Judge*, mais especificamente para as interfaces referentes às áreas nas quais o usuário possui acesso. Esse protótipo foi criado visando solucionar os problemas identificados via inspeção por investigação realizada pela autora e de testes de usabilidade, conduzidos pela mesma. Contudo, os testes de usabilidade, realizados nesse projeto, tem por objetivo complementar a pesquisa anterior, assim, com os novos resultados alterações foram feitas a fim de melhorar a usabilidade da plataforma. Como mencionado na seção anterior, com os resultados do primeiro teste foi criado um protótipo com mudanças e este foi usado na aplicação do segundo teste, que por sua vez possibilitou a criação do protótipo final a partir de seus resultados. É importante ressaltar que, por motivos de simplificação e para não estender muito o texto, somente o protótipo final será relatado com detalhes nesta seção, porém será pontuado as alterações que foram realizadas no protótipo criado antes do segundo teste.

3.2.1 Psicologia das Cores, Hierarquia Visual e Espaçamento

Além das correções adicionais pensadas para aperfeiçoar a usabilidade, faz parte do escopo deste trabalho realizar alterações no protótipo para tornar as interfaces mais atraentes e modernas. O protótipo anterior foi feito visando solucionar os problemas de usabilidade encontrados, mas ainda carecia de detalhes para aprimorar a satisfação do usuário ao utilizar a plataforma. Tendo em vista que os estudantes matriculados em uma disciplina auxiliada pelo *opCoders Judge*, utilizariam a plataforma com frequência e por um longo período, é de fundamental importância que esta tenha aspectos que influenciem positivamente as emoções e o agrado dos usuários.

De acordo com [Imtiaz \(2016\)](#), há diversos aspectos que tem de ser pensados para se criar o design de um website. O cérebro humano é afetado por estímulos enquanto ele visualiza elementos e interage com o ambiente, portanto é importante considerar conceitos da **psicologia cognitiva e comportamental** no processo de criação de interfaces digitais. É necessário tentar entender como os usuários percebem, interpretam e interagem com sites, como suas emoções e comportamentos são influenciados pelas escolhas de design e como tornar a experiência do usuário mais agradável, intuitiva e eficiente.

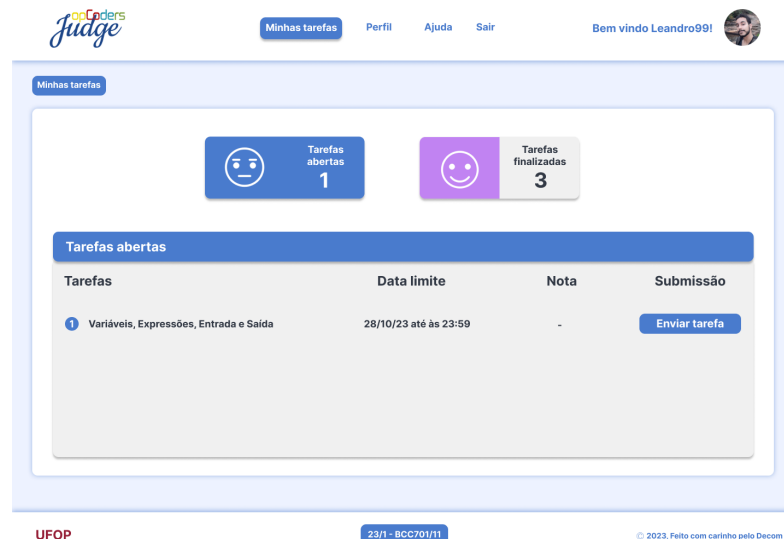
Na construção do novo protótipo do *opCoders Judge*, feito utilizando a ferramenta *Figma*, o primeiro aspecto considerado foi a **usabilidade**, talvez esse seja o principal parâmetro a se considerar ao se tratar de interfaces e o protótipo anterior corrigiu diversos problemas de usabilidade, logo é de extrema importância manter o mesmo nível de qualidade de usabilidade ou elevá-lo.

Outro aspecto a se considerar foi a psicologia das cores, esse foi o ponto que mais precisava de melhoria. O antigo protótipo foi baseado em cores mais escuras e não possuía muita variedade, para dar um ar mais moderno e atraente. Para resolver esse problema foi acrescentado

cores e tons a paleta de cores do *opCoders Judge*, além disso, alguns tons mais escuros foram substituídos por cores mais claras e vivas, como mostrado, esse tipo de estilo de cor transmite uma sensação de leveza aos usuários, segundo [Intiaz \(2016\)](#). A Figura 3.36, oriunda do protótipo no *Figma*, mostra a página **Minhas Tarefas**, do antigo protótipo e do novo, respectivamente, para fins comparativos.



(a) Antigo Protótipo.



(b) Novo Protótipo.

Figura 3.36 – Página Minhas Tarefas.

Como podemos ver na Figura 3.36, mudanças bem positivas podem ser alcançadas ao se utilizar da **hierarquia visual**, isto é, ao organizar o conteúdo para direcionar a atenção do usuário para as informações mais relevantes. Para entender esse conceito devemos compreender que uma interface possui várias camadas, no protótipo anterior as interfaces, no geral, consistiam em um fundo branco (primeira camada), com o conteúdo da página disposto em um “*container*”,

que possuía uma cor de fundo diferente (segunda camada). Essa simplicidade acarreta alguns pontos negativos, como pouco destaque em um componente importante como o *header*, que não possuía cor de fundo próprio; transmite a sensação de “pobreza” no design da interface e não incentiva o usuário a focar em todos os pontos importantes. Para solucionar essa questão é ideal acrescentar algumas cores de fundo em alguns componentes, essa alteração criará novas camadas e dará a sensação de uma plataforma mais diversificada e completa.

Os aspectos levantados até aqui, foram o embasamento utilizado para realizar as alterações do protótipo utilizado para o segundo teste de usabilidade relatado na Seção 3.1. Dessa forma, as mudanças realizadas consistiram na alteração de cores mais modernas e atraentes, com o intuito de melhorar a experiência do usuário. Além disso, também foi realizada uma adequação no tamanho dos componentes, para dispor melhor as informações. As alterações no protótipo relatadas daqui para frente, consistem em mudanças feitas após a realização do segundo teste, utilizando seus resultados como embasamento.

No novo protótipo o tamanho dos elementos foi reduzido, isso aconteceu, pois antes havia um certo exagero nos espaços e tamanho de fonte, talvez pela ausência de muitos elementos em algumas páginas. Além dessa redução de tamanho, foi retirado o componente referente à data, seu conteúdo podia ser duas coisas, a data e hora atual, isso acontecia na maioria das páginas, mas na página de alguma tarefa em aberto, seu conteúdo era substituído por quanto tempo o estudante ainda tinha para entregar a tarefa. Há valor na segunda funcionalidade, pois em qualquer aparelho que o usuário esteja usando para acessar a plataforma, seja computador ou celular, também é informado a data e hora atual.

3.2.2 Página Minhas Tarefas

É possível observar ainda na Figura 3.36 que foi reduzido o número de “cards” que compõem o filtro de tipos de tarefa. Foi observado que a divisão de “tarefas concluídas” e “tarefas perdidas” não foi bem aceita durante os testes, gerando um pouco de confusão. Para resolver esse problema elas foram unificadas e é possível fazer a distinção delas pela presença ou ausência de nota na tarefa. Por exemplo, se um usuário não realiza nenhuma submissão para uma tarefa antes do prazo de entrega terminar, ela permanecerá sem nota para sempre, mesmo que este usuário realize submissões posteriormente.

Além do filtro por estado da tarefa, também foi acrescentado um filtro que separa as tarefas por turmas. Este acréscimo foi pensado, pois caso um usuário estivesse vinculado a muitas turmas, o ambiente poderia mostrar um grande volume de tarefas e a quantidade de informações na tela seria caótica. Isso afetaria principalmente a professores administradores que estão vinculados a uma grande quantidade de turmas.

Quando usuários entram pela primeira vez no site, é claro que não irão saber todas as funcionalidades e com o que podem interagir a princípio. Contudo, uma interface de alta qualidade

possui elementos para comunicar informações ao usuário, sem a utilização de texto, isso pode ser feito pelo próprio design dos elementos e também por técnicas dinâmicas de animações. Além de deixar a interface mais intuitiva, as interações se tornam mais prazerosas de serem feitas, portanto, foram acrescentados diversos efeitos de animação e elementos que podem ser alguma funcionalidade tiveram sua cor intensificada para inferir isso.

Um acréscimo interessante foi o do componente de *Footer*. Ele não consiste em um componente exclusivo da página **Minhas Tarefas**, pois ele é um componente global. Contudo, ele está diretamente relacionado com esta página, pois é nele que é feito o filtro de tarefas por turma.

3.2.3 Header



(a) Antigo Protótipo.

(b) Novo Protótipo.

Figura 3.37 – Header.

Conforme a Figura 3.37, o *header*, isto é, o componente que está sempre no topo da interface, responsável pela navegação das páginas, passou por algumas mudanças significativas. No antigo protótipo, ele consistia no menu de navegação, um componente de data e hora, uma mensagem de boas-vindas para o usuário e um componente para retroceder a navegação, isso tudo ocupava muito espaço. No novo protótipo a disposição dos componentes está mais enxuta, o fundo branco foi mantido, porém, agora ele contrasta com o fundo azulado do restante da página. Para contrastar ainda mais, foi adicionada uma sombra em tom azul-escuro. Além disso, neste projeto o usuário terá a opção de personalizar um apelido e selecionar uma foto de perfil para sua conta, ambas essas funcionalidades são mostradas no *header*. E, por fim, a opção de retroceder a navegação foi substituído por um histórico de navegação interativo que será melhor explicado adiante.

3.2.4 Página Login

A página **Login** passou por grandes mudanças, no protótipo anterior havia formulário para *login* e para recuperação de senha, funcionalidades essenciais para o tipo de login da plataforma atual. Contudo, uma ferramenta que cresceu muito nos últimos anos é o *login* social, isto é, método de autenticação onde os usuários podem fazer *login* em um site ou aplicativo utilizando suas

contas de redes sociais existentes. Houve algumas problemáticas com as credenciais de alunos no passado, o sistema de *login* atual conta com a utilização de e-mail e senha para autenticar o acesso, a senha é gerada automaticamente para o primeiro acesso do aluno, utilizando de caracteres aleatórios. Posteriormente, na plataforma, é possível alterar a senha de maneira arbitrária, contudo esta abordagem não é a ideal. Por este motivo foi decidido que o *login* para a plataforma será realizado apenas por *login* social utilizando o **Google**. Toda a lógica de *login*, criação de conta e alteração de turmas são efetuadas pelo sistema do site quando o usuário realizou o login social **Google**, o funcionamento do sistema para alcançar tal funcionalidade será explicado nos próximos capítulos. O e-mail institucional da UFOP é uma conta *Google*, logo todos os usuários terão acesso garantido, além disso, com essa mudança o login passa a ser mais fácil e rápido. Como não havia mais necessidade dos formulários de *login* e recuperação de senha, foi feita uma reorganização dos elementos na página **Login**.



Figura 3.38 – Página de Login.

3.2.5 Página Perfil

A página **Perfil** de usuário passou por mudanças, no protótipo antigo ela era composta por dois “*containers*”, um para mostrar os dados pessoais do usuário e o outro para tanto permitir troca de senha, quanto também para mostrar algumas estatísticas do desempenho do usuário, como ilustrado na Figura 3.39. Com a nova proposta, na autenticação de login, não há mais a necessidade da funcionalidade de troca de senha, por esse motivo essa página se tornou pouco útil. Contudo, página **Perfil** de usuário é importante em qualquer plataforma, então como podemos observar também na Figura 3.39, a funcionalidade de troca de senha foi substituída por um “*container*” disponibilizando informações sobre o desempenho do estudante na plataforma e outro para o usuário personalizar o apelido de sua conta e sua imagem de perfil, ambas são novas funcionalidades criadas nesse trabalho. É válido ressaltar que esta página é um espaço com

muitas possibilidades de uso, podendo colocar diversos tipos de funcionalidades ou informações que dizem respeito ao usuário nela, por esse motivo a versão atual do novo protótipo não é considerada a versão final, estando sujeito a mudanças em trabalhos futuros.

Antigo Protótipo: A interface apresenta uma barra superior com o logo 'Judge' e links para 'Minhas tarefas', 'Perfil', 'Dicas', 'Ajuda' e 'Sair'. Abaixo, uma mensagem de boas-vindas 'Seja bem-vindo, Nome Aluno!' e um relógio digital mostrando 'Data e hora: 01/01/2023 23:59'. O conteúdo principal é dividido em duas colunas. A esquerda, sob o título 'Dados pessoais:', há campos para 'Nome' (com o placeholder 'Nome e sobrenome'), 'Email' (com o placeholder 'nome.sobrenome@aluno.ufop.edu.br') e 'CPF' (com o placeholder '123.456.789-10'). A direita, sob o título 'Alterar senha:', há campos para 'Senha atual', 'Nova senha' e 'Repetir nova senha', seguidos por um botão 'Alterar senha'.

(a) Antigo Protótipo.

Novo Protótipo: A interface atualizada apresenta uma barra superior com o logo 'Judge' e links para 'Minhas tarefas', 'Perfil', 'Ajuda' e 'Sair'. Abaixo, uma mensagem de boas-vindas 'Bem vindo Leandro99!' com uma imagem de perfil. O conteúdo principal é dividido em três seções. A esquerda, sob o título 'Dados pessoais:', há campos para 'Nome' (com o placeholder 'Visitante'), 'Email' (com o placeholder 'Visitante@gmail.com'), 'Curso' (com o placeholder 'Ciências da Computação') e 'Disciplina Padrão' (com o placeholder 'BCC 202 - Estrutura de Dados'). Abaixo disso, há uma seção 'Estatísticas' com 'Tarefas finalizadas' (5), 'Tarefas abertas' (3), 'Questões Resolvidas' (77) e 'Média' (1.83). A direita, sob o título 'Perfil de Usuário', há um campo 'Apelido' (com o placeholder 'Leandro99') e um botão 'Escolha uma imagem' com uma seta para cima, seguido por um botão 'Salvar'.

(b) Novo Protótipo.

Figura 3.39 – Página de Perfil.

3.2.6 Navegação Entre Páginas

Um dos acréscimos feito na monografia de Cedraz (2023) foi uma opção de retroceder a navegação do usuário, sem a necessidade de usar a *browser*. No novo protótipo tem muito mais opções de navegação, com isso é possível que em determinado momento seja necessário inúmeras chamadas dessa função de retroceder, cada vez que ela é chamada a página web é atualizada, isso faz com que essa funcionalidade não seja muito performática. Portanto, uma solução pensada para resolver esse problema foi implementar um histórico de navegação, ele conterá todo o caminho que o usuário percorreu conforme a “url” atual, dessa forma com apenas um clique o

usuário poderá retroceder para qualquer página que ele já tenha acessado de forma mais fácil. Essa solução aparece em alguns websites nos últimos anos, um exemplo bem conhecido é a plataforma do *Google Drive*.

A imagem mostra uma barra de navegação com um fundo azul claro e um botão azul com o texto "Minhas tarefas > Variáveis, Expressões, Entrada e Saída" em branco.

Figura 3.40 – Histórico de Navegação.

3.2.7 Página Tarefa Aberta

A página **Tarefa Aberta** passou por uma mudança importante. O conteúdo dela pode ser muito grande, pois ela contém um menu de questões, estas ficavam dispostas na vertical, logo a página cresciam bastante verticalmente conforme o conteúdo das questões era aberto. Uma solução para esse problema foi deixar o menu de questões na horizontal e fixo no topo do componente, assim o usuário pode navegar entre as questões com mais praticidade e o conteúdo total da página fica mais polido, conforme é possível ver na Figura 3.41.

Na Figura 3.41, é evidente uma significativa alteração no protótipo de Cedraz (2023), que diz respeito à forma como a plataforma incorporará as dicas em seu sistema. Anteriormente, as dicas eram visualizadas na página **Dica**, acessível por meio do cabeçalho, e também podiam ser encontradas no botão localizado no contêiner do título da tarefa, vinculado à respectiva tarefa. No modelo atual, as dicas podem estar associadas tanto a uma tarefa quanto a uma questão. No entanto, dado que é improvável que o usuário busque uma dica de maneira isolada, optou-se por excluir a página **Dica** do modelo. De forma que, o usuário pode acessar as dicas relacionadas a uma tarefa ou questão na página **Tarefa Aberta**, visualizando o conteúdo por meio de *pop-ups*. Além disso, foi adicionado um botão de resumo para consolidar as informações demandadas ao usuário nesta mesma página.

Outra mudança significativa na plataforma envolve a exclusão da página dedicada à visualização da correção de uma questão. Com o intuito de otimizar o conteúdo do *opCoders Judge*, optou-se pela remoção dessa página, uma vez que seu conteúdo pode ser facilmente acessado por meio de um *pop-up*, conforme ilustrado na Figura 3.42.

Alguns componentes não passaram por grandes modificações e sim apenas algumas alterações nas cores, espaçamento e tamanho, sendo estes a página **Ajuda**, conforme A, e os *pop-ups*, encontrados na Apêndice B. É válido ressaltar que isso se aplica somente ao conteúdo dessas páginas em si, já que elas são compostas também pelos componentes *header*, que foi bastante modificado, e *footer*, que não existia no protótipo anterior.

opCoders Judge
Corretor Automático de Código-Fonte

Minhas tarefas Perfil Dicas Ajuda Sair

Tempo restante: 2 dias e 2 horas

Variáveis, Expressões, Entrada e Saída

Para realizar a tarefa é necessário ler os capítulos 2 e 3 do livro da disciplina. Além de assistir às aulas teóricas correspondentes.

Questão 01

Progressão geométrica é uma sequência numérica que possui uma razão fixa denominada q onde, a partir da definição do primeiro termo a_1 , os termos subsequentes são calculados individualmente pela razão q multiplicada pelo seu antecessor.

Implemente um programa que leia, como entradas dos usuários, os valores reais representando o primeiro termo (a_1) e a razão (q), o valor inteiro representando o número n . O programa calcula o valor do termo a_n e imprime o resultado no terminal com uma precisão de 2 casas decimais.

Selecione a linguagem a ser utilizada:

C Python

Enviar nova solução de código-fonte:

Enviar arquivo

Tentativas

Data de envio	Nota	Código fonte	Correção
17/01/23 às 20:46	2	Baixar código	Ver correção

(a) Antigo Protótipo.

opCoders Judge

Minhas tarefas Perfil Ajuda Sair Bem vindo Leandro99!

Variáveis, Expressões, Entrada e Saída

Para realizar a tarefa é necessário ler os capítulos 2 e 3 do livro da disciplina. Além de assistir às aulas teóricas correspondentes.

Questão 01 **Questão 02**

Progressão geométrica é uma sequência numérica que possui uma razão fixa denominada q onde, a partir da definição do primeiro termo a_1 , os termos subsequentes são calculados individualmente pela razão q multiplicada pelo seu antecessor.

Implemente um programa que leia, como entradas dos usuários, os valores reais representando o primeiro termo (a_1) e a razão (q), o valor inteiro representando o número n . O programa calcula o valor do termo a_n e imprime o resultado no terminal com uma precisão de 2 casas decimais.

Selecione a linguagem a ser utilizada:

C

Enviar nova solução de código-fonte:

Enviar arquivo

Tentativas

Data de envio	Código fonte	Correção	Nota
17/01/23 às 20:40	Ver código	Ver correção	9.2

UFOP 23/1 - BCC701/11 © 2023, Feito com carinho pelo Decon

(b) Novo Protótipo.

Figura 3.41 – Página Tarefa Aberta.

3.3 Desenvolvimento do novo *opCoders Judge*

Nesta seção, será detalhado o desenvolvimento do novo *opCoders Judge*, focando nos aspectos essenciais do *frontend*, *backend* e banco de dados. Exploraremos a criação da interface do usuário, a lógica de negócios e a estrutura eficiente de armazenamento de dados.

3.3.1 Frontend

Impulsionado pela crescente demanda por interfaces de usuário interativas, responsivas e atraentes, o ambiente do *frontend* tem evoluído muito nos últimos anos. Nesse contexto, ele é uma das peças-chave no desenvolvimento de websites modernos, desempenhando um

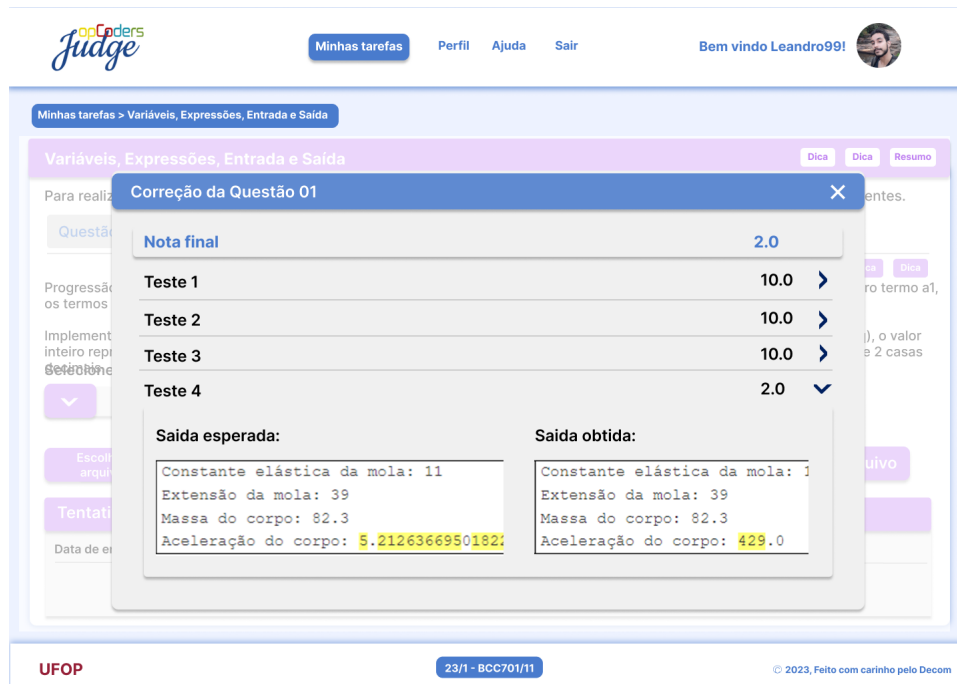


Figura 3.42 – Componente de pop-up para correção.

papel crucial na experiência do usuário e na efetividade das aplicações online. É na etapa do desenvolvimento *frontend* que o protótipo desenvolvido neste trabalho será implementado com algumas ferramentas.

Para relatar todos os pontos a respeito do desenvolvimento da parte *frontend*, essa seção foi dividida em subseções. A Subseção 3.3.1.1 discorrerá sobre a arquitetura adotada para o código. Por último, a Subseção 3.3.1.2 contará quais foram as implementações feitas nessa monografia.

3.3.1.1 Arquitetura utilizada

É bem comum que o padrão de arquitetura seja decidido tendo em mente dois aspectos, o objetivo desejado e as tecnologias utilizadas. O *React* utiliza um padrão chamado *Component-Based*, essa abordagem baseia-se, como o próprio nome sugere, em componentes reutilizáveis e independentes, que podem ser compostos para criar a interface do usuário de uma aplicação. Cada componente representa uma parte específica da interface e pode ser composto por outros componentes menores.

Essa arquitetura não é tão rigorosa a respeito da disposição das partes do código, portanto a organização das pastas varia de desenvolvedor para desenvolvedor. Contudo, a parte mais importante do projeto são os componentes reutilizáveis, por isso eles são mantidos isolados de outras partes do código, para serem acessados somente quando for necessário.

A organização da arquitetura do *frontend* é composta por um arquivo *main*, responsável por fazer a renderização de todas as interfaces, ele invoca o arquivo *app*, cujo objetivo é lidar com

todas as rotas configuradas utilizando o *React Router Dom*, cada rota aponta para uma página tem sua própria interface e pode estar invocando componentes reutilizáveis se for cabível. Por exemplo, no contexto do *opCoders Judge*, todas as páginas possuem o componente *header*, *footer* e o *navigator* (este corresponde ao histórico de navegação). Logo, todas as páginas invocam instância desses componentes, porém também possuem seus próprios componentes únicos. Esse funcionamento pode ser melhor ilustrado observando a Figura 3.43.

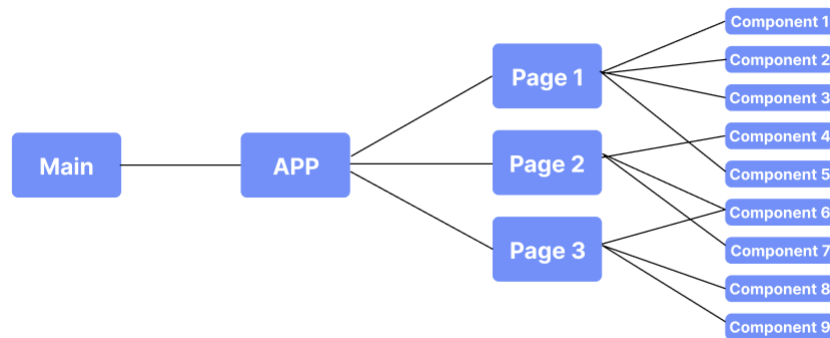


Figura 3.43 – Diagrama da arquitetura do *frontend*.

3.3.1.2 Funcionalidades Implementadas

Neste trabalho, na parte do *frontend*, foram implementados diversas interfaces: a página **Login**, página **Minhas Tarefas**, página **Tarefa Aberta**, página **Ajuda**, página **Perfil**, também foi implementada o sistema de autenticação de login utilizando *Local Storage*. Estas páginas constituem todas as interfaces para o funcionamento do *opCoders Judge*, na perspectiva do usuário. As páginas referentes à área administrativa da plataforma, onde os diferentes tipos de administradores poderiam fazer operações como criar uma nova tarefa, não foram consideradas no protótipo e implementação, pois a ferramenta **Django Rest Framework** oferece um ambiente, já com interface, onde é possível realizar todas essas operações de maneira satisfatória.

3.3.1.2.1 Login

A respeito da página **Login**, a interface foi implementada conforme o protótipo criado neste trabalho. A integração com a *API Google* foi realizada utilizando a biblioteca *react-oauth*, que oferece uma implementação simples e rápida de um botão de *login Google*. Com isso, a verificação das credenciais é feita pela própria *API Google*.

Para o controle de autenticação de *login*, utilizou-se um recurso do *React* chamado **UseContext**. Esse recurso permite a criação de um ambiente isolado acessível por qualquer componente da aplicação. A variável de controle de autenticação encontra-se nesse ambiente.

Na página **Login**, o usuário realiza o *login Google*. Em seguida, o *frontend* envia uma requisição HTTP para a rota do backend responsável por verificar se aquele usuário pode acessar

a plataforma. Caso seja autorizado, o *backend* retorna uma resposta positiva, juntamente com um *token* válido de acesso. O *frontend* armazena esse *token* no *local storage* do navegador do usuário. Dessa forma, o sistema pode verificar se o usuário está logado ao checar o *local storage*.

Além disso, é criado um *token* relacionado ao tempo que o usuário pode ficar inativo sem ser desconectado do *opCoders Judge*. Se o usuário não realizar nenhuma ação na plataforma dentro desse intervalo de tempo, o *logout* é feito, os *tokens* são apagados, e o usuário é redirecionado para a página **Login**. É válido destacar que todos os *tokens* criados no *local storage* estão criptografados para nenhuma informação ser vazada.

3.3.1.2.2 Página Minhas Tarefas

Sobre a interface da página de **Minhas Tarefas**, assim como as outras páginas, ela acessa o ambiente global do *UseContext* e verifica se o *token* de acesso está no *local storage* para checar se o usuário está autenticado, caso a resposta seja negativa o usuário é encaminhado para a página **Login**, se positiva o conteúdo da página é mostrado. Na primeira vez que o usuário *logar* na plataforma, ele será recebido por uma caixa com instruções para utilizar o *opCoders Judge*, o sistema saberá se é a primeira vez utilizando uma variável no banco de dados. Quando essa caixa de instruções for fechada, ou nas próximas vezes que o usuário visitar a plataforma, é renderizado as tarefas das turmas que o usuário está matriculado.

É válido destacar a principal mudança da página **Minhas Tarefas** na versão atual do *opCoders Judge* para esta que está sendo desenvolvida, neste projeto não é possível visualizar todas as tarefas da turma, agora é feito um filtro utilizando um botão com opções para controlar se será visualizado somente tarefas “abertas”, “finalizadas”. Além disso, é feito também uma filtragem pela turma, caso um usuário tenha feito parte de mais de uma e para os professores que lecionam em várias turmas e precisam separar as tarefas para não haver problemas, como mencionado na seção anterior.

Foi utilizado a classe “*hover*” para introduzir animações com troca de cor de fundo para transmitir uma sensação de movimento e informar ao usuário com o que ele pode interagir.

3.3.1.2.3 Página Perfil

A página **Perfil** foi implementada conforme o novo protótipo, ela possui três componentes: “Dados Pessoais”, “Estatísticas” e “Perfil de Usuário”. O componente de “Dados Pessoais” tem por objetivo mostrar algumas informações sobre a conta do usuário. Quando ele realiza o *login*, suas informações de conta são carregadas no *useContext*, e é dessa maneira que a página **Perfil** obtém os dados renderizados no componente “Dados Pessoais”.

O componente “Estatísticas” é construído principalmente por meio de consultas SQL para recuperar dados relacionados às tarefas e questões realizadas pelo usuário, com algumas operações adicionais sobre esses dados.

Por fim, o componente “Perfil do Usuário” destina-se à personalização do apelido e da imagem de perfil. É importante destacar que foram adotadas medidas para reduzir o espaço de armazenamento das imagens. Uma delas envolve o fato de que, por padrão, o **Django** armazena uma nova imagem sempre, sem verificar se uma imagem idêntica já existe no banco de dados. Para contornar esse problema, foi implementada a seguinte lógica: ao alterar a imagem de perfil, a imagem antiga é removida do banco de dados, garantindo que cada usuário tenha apenas uma imagem armazenada. Antes de salvar a imagem no banco de dados, ocorre um processamento em duas etapas. Primeiramente, a imagem é redimensionada para as dimensões de 160 x 160, visando reduzir o espaço ocupado. Em seguida, é gerado um nome padrão para a imagem, composto por “profile_image_user_(ID do usuário)_(5 caracteres aleatórios).(extensão da imagem)”. Os caracteres aleatórios são utilizados para evitar problemas de cache do navegador. Então um exemplo prático desse cenário seria: caso o usuário de ID 3 altera sua imagem de perfil, o nome da imagem no banco de dados pode ser algo como “profile_image_user_3_jSAu4.png”.

3.3.1.2.4 Histórico de Navegação

Quando o usuário seleciona uma tarefa, é redirecionado para uma nova página, cuja URL consiste na URL da página anterior, com o acréscimo do nome da tarefa, sem caracteres especiais, acentuação e espaços, o nome da tarefa nesse formato é denominado de “slug”. O componente de **Histórico de Navegação** analisa a URL da página atual e constrói uma lista, onde cada posição representa o nome de uma página navegável. Por exemplo, ao acessar a tarefa denominada “Operações Fundamentais”, a URL da página será algo como: “https://www.opcoders.decom.ufop.br/minhas-tarefas/operacoes-fundamentais”. Dessa forma, a lista navegável gerada pelo **Histórico de Navegação** incluirá as posições “minhas-tarefas” e “operacoes-fundamentais”.

Antes de renderizar o “*container*” interativo com essas posições, é realizada uma busca no banco de dados para identificar a tarefa pelo campo “slug”. Isso possibilita recuperar o título contendo seu nome apropriado, incluindo caracteres especiais, espaços e acentuação correta. Em relação às demais posições da lista navegável, elas passam por uma formatação padrão para adequar o nome. Não é necessária uma lógica mais complexa, uma vez que as páginas **Minhas Tarefas**, **Perfil** e **Ajuda** não possuem caracteres especiais ou acentuação.

3.3.1.2.5 Tarefa Aberta

A página **Tarefa Aberta** é o componente mais complexo de toda a plataforma *opCoders Judge*, devido ao grande volume de requisições feitas ao *backend*. Quando a página é carregada, uma consulta SQL é realizada usando o “slug” da URL para identificar a tarefa selecionada. Após essa operação, são efetuadas novas consultas SQL para buscar os dados das questões vinculadas a ela, os dados das dicas vinculadas à tarefa e às questões, as entregas referentes a cada questão e, por fim, quais linguagens são aceitas nas entregas.

É válido ressaltar que a decisão sobre quais linguagens são aceitas em uma questão é tomada com base em algum referencial, que pode ser a própria questão, tarefa, turma ou disciplina, hierarquicamente decidido nessa ordem. Isso pode ser compreendido melhor ao observar o seguinte cenário fictício: existe uma turma “BCC701”, correspondente à disciplina “Programação de Computadores I”, com a tarefa “Operações Fundamentais” vinculada a ela. Essa tarefa possui apenas a “Questão 01” para ser resolvida. Quando o usuário seleciona essa tarefa para visualização, na consulta SQL para buscar quais linguagens são aceitas para a “Questão 01”, primeiro verificamos se, na criação da questão, foram especificadas as opções de linguagens aceitas. Se sim, elas são renderizadas; se não, verificamos se isso foi especificado na tarefa “Operações Fundamentais”. Se ainda não, esse procedimento se repete para a turma “BCC701” e depois para “Programação de Computadores I”.

Ainda sobre a página **Tarefa Aberta**, ao usuário submeter uma entrega, é verificado se todos os campos da entrega foram preenchidos e se a opção de linguagem é condizente com a extensão do arquivo. Em todos os casos, é renderizada uma mensagem por meio de *pop-up* para explicar se foi possível realizar a entrega; em caso negativo, é explicado o problema. Quando a entrega é feita, uma requisição SQL é realizada para criar um novo registro de entrega. Esse registro não possui nota e tem um campo e é marcado como pendente, indicando que ainda não passou pelo sistema de correção. Após a criação desse registro, é criada uma instância dessa entrega na pasta **Entregas**.

O sistema de correção proposto por Brito (2019b) opera *offline*, mas seu funcionamento não é manual. Para otimizar o processo, foi desenvolvida uma funcionalidade que o coloca para rodar periodicamente, com intervalos curtos de tempo. Sua lógica consiste em acessar a pasta **Entregas** para recuperar tanto a entrega submetida por um usuário quanto o código de saída definido como a resposta ideal para a questão, ambos localizados dentro dessa pasta. Tendo acessado esses dois arquivos, o serviço recupera as informações das entregas pendentes e realiza as correções, atualizando os dados das entregas com os resultados obtidos.

3.3.1.2.6 Dicas

Por fim, é importante destacar que no “*container*” do título da tarefa, na página **Tarefa Aberta**, estão os botões correspondentes ao resumo e dicas da tarefa, se houver. Logo acima do enunciado da questão, encontram-se as dicas referentes à questão, também se houver. O conteúdo das dicas e do resumo é renderizado por meio de *pop-up*.

É válido ressaltar que um usuário pode não ter mais permissão para realizar submissões para uma tarefa, isso acontece pois o usuário só pode fazer submissões para turmas em que ele, atualmente, está matriculado no sistema da UFOP, caso isso mude, o usuário perde a permissão de realizar submissões. Também é possível que tanto um usuário quanto uma turma inteira específicos podem ter tido seus direitos de submissão revogados, isso pode acontecer caso um administrador constate que um usuário não é mais aluno da universidade, dessa forma é dispensável o controle

de submissão daquele usuário para turmas específicas. Quando um semestre termina, as turmas dele se tornam inativas, de forma que não é mais necessário realizar submissões para as mesmas.

3.3.1.2.7 Página Ajuda

A página **Ajuda** possui um design simples, conceitualmente sua função é disponibilizar mensagens que possam sanar dúvidas dos usuários referentes à utilização da plataforma. As mensagens são criadas por administradores. As mensagens de ajuda são listadas mostrando o título em um “*container*”, uma vez que o usuário clique em algum deles será renderizado um *pop-up* mostrando o conteúdo da mensagem.

3.3.2 Remodelagem do Banco de Dados

A utilização de banco de dados é um fator crucial para qualquer plataforma digital no mundo contemporâneo. Dados são gerados em volumes exponenciais e a informação é considerada um dos bens mais valiosos, a utilização de bancos de dados tornou-se indispensável em diversos setores da sociedade. De acordo com [Matsumoto \(2008\)](#), as empresas que buscam melhorar sua eficiência operacional e tomar decisões estratégicas embasadas em dados, até instituições de pesquisa e órgãos governamentais que necessitam organizar e analisar abundância de informações, os bancos de dados desempenham um papel fundamental.

3.3.2.1 Modelo da versão atual do banco de dados do *opCoders Judge*

No contexto desse trabalho, o *opCoders Judge* armazena dados que compõem toda a sua organização, estudantes, professores, tarefas e questões. O armazenamento desses dados é especialmente importante, pois, por exemplo, o *opCoders Judge* não é uma plataforma aberta que qualquer usuário que tiver interesse pode realizar um cadastro, isso até o momento atual, os estudantes matriculados na disciplina são cadastrados pelos administradores, que lhes fornece o acesso e quando terminam a disciplina o acesso lhes é revogado. Além disso, para fins organizacionais, diversos outros aspectos têm de estar armazenado no banco de dados, como as questões que podem compor as tarefas, tarefas já preparadas, disciplinas que fazem uso do *opCoders Judge*, turmas e semestres para que os resultados fiquem registrados, todo esse fluxo de dados torna essencial que o banco de dados seja eficiente, garanta a integridade dos dados e escale bem. O atual banco de dados, proposto por [Patrocínio \(2023\)](#), pode ser observado no modelo de entidade relacionamento da Figura 3.44.

Este modelo é bem completo e eficiente em abranger os componentes necessários para um bom funcionamento da plataforma, contudo possui alguns pontos de melhoria que fará com que o *opCoders Judge* possa ser uma ferramenta mais completa, versátil e intuitiva.

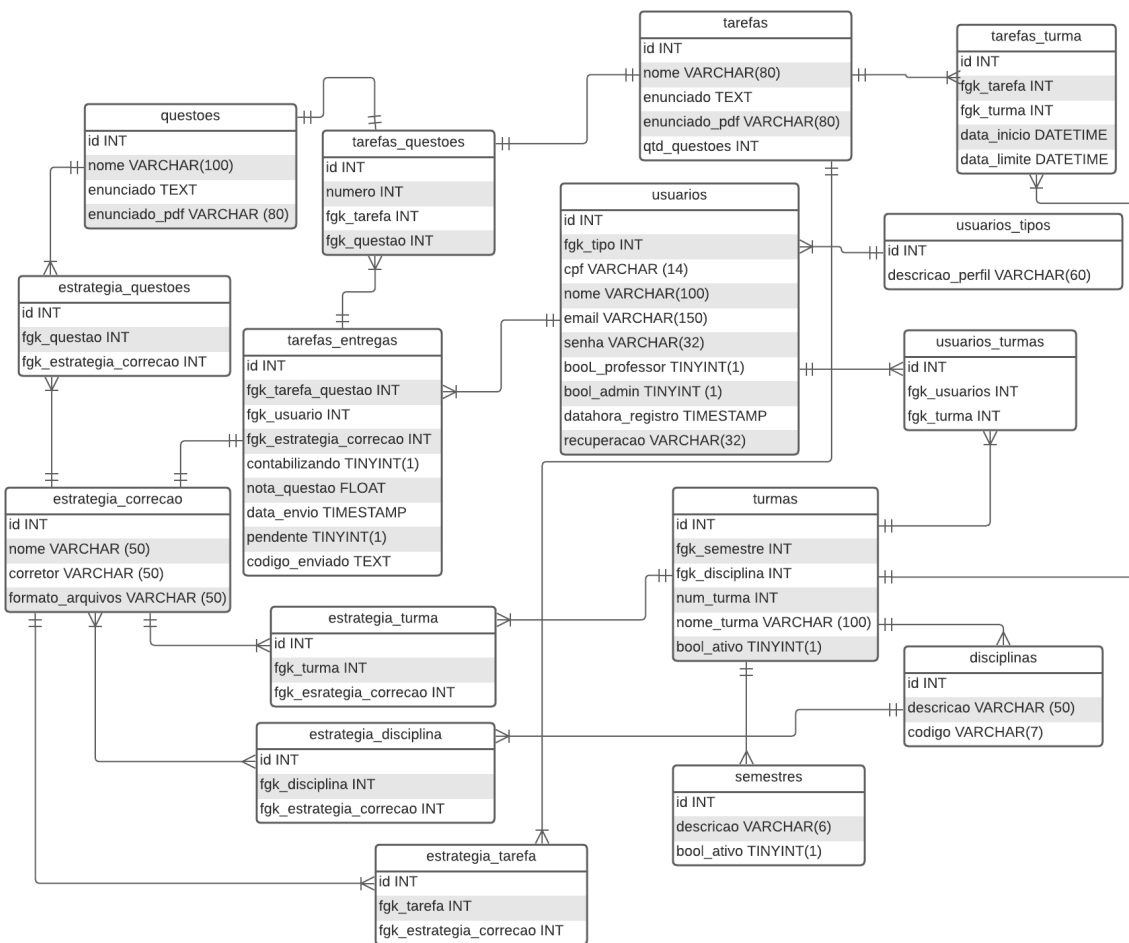


Figura 3.44 – Versão final do diagrama ER utilizada atualmente.

3.3.2.2 Modelo da versão dinâmica do banco de dados do *opCoders Judge*

A capacidade de um corretor de código-fonte suportar a geração de questões dinâmicas, ou seja, questões que podem apresentar diferentes versões do enunciado, é crucial para proporcionar uma experiência de aprendizado mais enriquecedora e desafiadora. A diversificação das questões promove a adaptação aos diversos níveis de habilidade dos alunos, estimula o pensamento crítico e impulsiona o desenvolvimento contínuo. Além disso, possibilita a criação de avaliações mais abrangentes, avaliando não apenas a memorização, mas também a compreensão e aplicação prática dos conceitos. Com isso, pode-se entender melhor a importância do trabalho desenvolvido por [Mendonça \(2023\)](#) e na Figura 3.45 é possível entender como foi elaborado o diagrama entidade relacionamento, proposto em seu trabalho, para que o *opCoders Judge* tenha banco de dados dinâmico.

Este modelo é bem eficiente em sua proposta, ele é focado em ter as ferramentas necessárias para a utilização de um sistema dinâmico, contudo ainda lhe faltam muitos elementos para dar o suporte necessário para o funcionamento completo do sistema do *opCoders Judge*, pois ele não possui suporte para o controle de turmas e para os diferentes tipos de correção (*opCoders Judge* trabalha com diferentes linguagens de programação). A falta desses elementos se deve a

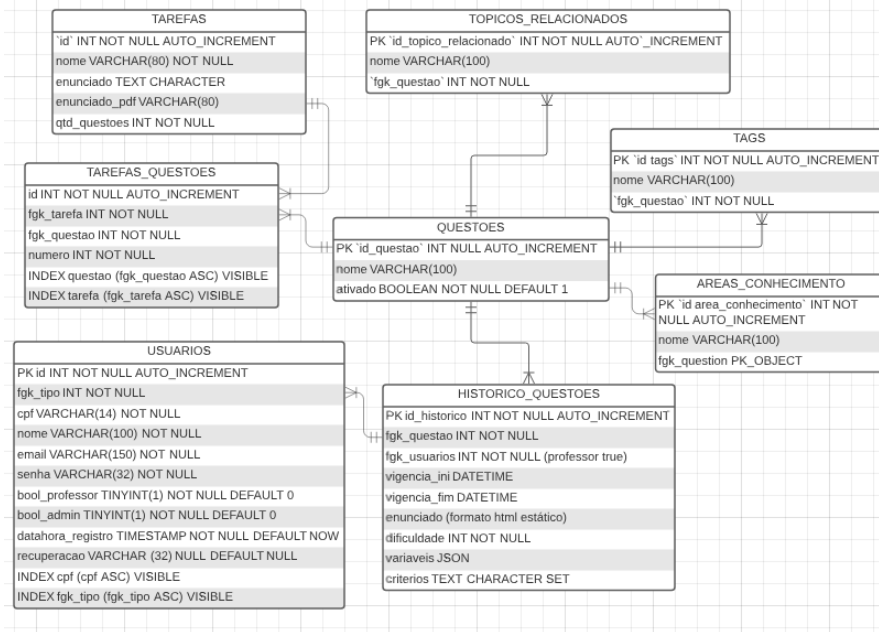


Figura 3.45 – Diagrama ER de banco de dados dinâmico.

proposta específica na qual o modelo foi elaborado.

3.3.2.3 Modelo da versão proposto por esta monografia

Em relação ao banco de dados desenvolvido nesta monografia, o objetivo visado foi de criar um modelo que consiste na combinação dos modelos propostos por [Patrocínio \(2023\)](#) e [Mendonça \(2023\)](#) em um só, com o acréscimo de algumas melhorias pontuais observadas durante o desenvolvimento do protótipo e algumas alterações realizadas para o melhor uso de algumas funcionalidades do **Django**. A Figura 3.46 ilustra a versão final proposta neste trabalho.

A mudança mais perceptível é a alteração da nomenclatura em português para o inglês, motivada pela adesão ao padrão amplamente adotado na comunidade de desenvolvimento de sistemas. Essa modificação também é justificada pela universalidade do inglês, facilitando a compreensão e colaboração em um contexto global e isso pode trazer vantagens e oportunidades no futuro.

Antes de avançar na explicação das alterações realizadas, é importante compreender uma das vantagens do uso do **Django**. No contexto deste projeto, onde *frontend* e *backend* são sistemas distintos, a comunicação entre eles não é direta, sendo necessário utilizar requisições web, conforme detalhado anteriormente. No entanto, essas requisições podem apresentar desafios, especialmente quando o *frontend* solicita dados que demandam processamento e tempo de resposta do *backend*. Isso pode resultar em períodos nos quais o *frontend* fica impedido de renderizar as informações solicitadas. Em uma plataforma complexa como o *opCoders Judge*, com uma considerável quantidade de requisições, variáveis como estabilidade do servidor e tráfego de rede podem impactar a experiência do usuário.

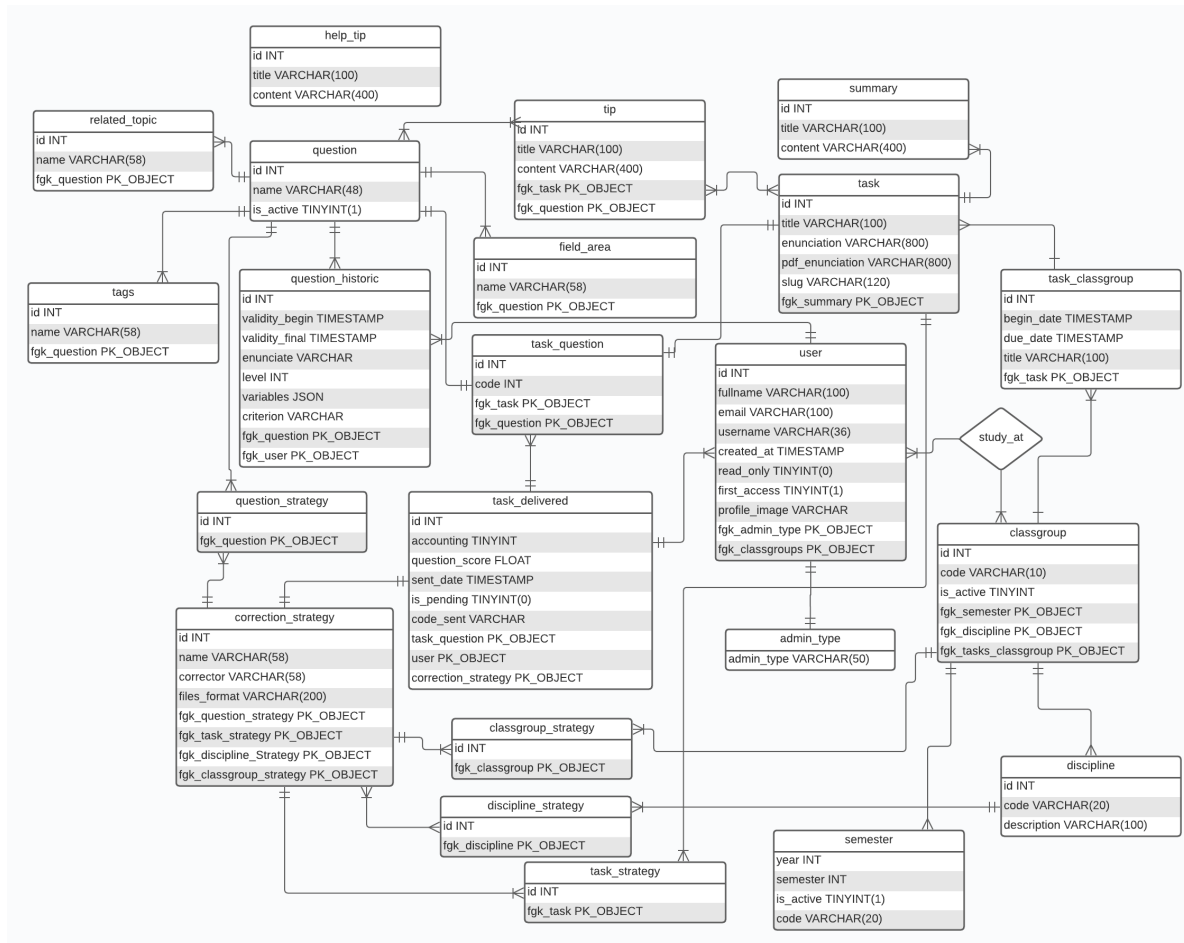


Figura 3.46 – Versão final do diagrama ER proposta nesta monografia.

No banco de dados, existem as chamadas chaves estrangeiras, identificadas pelos campos com a nomenclatura “fgk” no início do nome. Essas chaves estrangeiras representam relacionamentos entre tabelas. Por exemplo, a tabela **user** possui um campo chamado *fgk_classgroups*, indicando um relacionamento com a tabela **classgroup**. Nesse contexto, isso significa que um usuário pode estar matriculado em várias turmas. O conteúdo armazenado nesse campo são os IDs dos registros da tabela **classgroup** vinculados ao registro específico da tabela **user**. Em um sistema tradicional, para renderizar as tarefas de uma turma na qual o usuário está matriculado, seriam necessárias duas requisições encadeadas, uma para acessar as turmas do usuário e outra, através do campo *fgk_task_classgroup*, para acessar as tarefas.

No entanto, no **Django**, é possível reduzir a quantidade de requisições com um planejamento adequado de chaves estrangeiras. Por meio de uma configuração no código, ao invés de os campos “fgk” armazenarem IDs, eles podem armazenar objetos JSON correspondentes aos registros em si. No exemplo mencionado anteriormente, é possível acessar os dados da turma e das tarefas através do registro do usuário sem a necessidade das duas requisições mencionadas anteriormente.

Embora essa estratégia seja eficiente para diminuir a quantidade de requisições, é impor-

tante considerar que uma requisição única com muitos objetos JSON encadeados pode gerar um volume de dados significativo, podendo levar mais tempo do que uma abordagem de segmentação de requisições. Portanto, foi avaliado quais chaves estrangeiras poderiam ser alteradas para carregar objetos JSON, visando melhorar a eficiência do sistema sem comprometer a consistência do banco de dados.

A respeito do modelo dinâmico, ele foi bastante preservado, tendo sido feito poucas alterações. A tabela **user** deste modelo é a mesma do modelo atual do *opCoders Judge* e esta sofreu algumas alterações nos seus campos. Foram removidos os campos *cpf*, *bool_professor*, *bool_admin* e *recuperacao*, além disso, foi inserido dois campos para deixar a plataforma mais estética, sendo eles *nickname* e *image_path*, o primeiro é um apelido que o usuário pode dar a sua conta para assim ser chamado, durante seu acesso, e o segundo refere-se ao caminho para sua imagem de perfil, que até o atual momento está sendo salva em um diretório no *backend*. Além disso, foi acrescentado duas chaves estrangeiras, uma para representar sua relação com a tabela **user_classgroup** e a outra para representar sua relação com **admin_type**. Por fim, foi acrescentado o campo *first_access* para fazer o controle de quando exibir o componente de tutorial no frontend e *read_only* para caso o usuário não tenha mais permissão para fazer submissões.

Já o modelo da versão atual, por ser mais abrangente, teve mais alterações. Foi criada uma nova tabela chamada *admin_type*, a função desta tabela é identificar quando um usuário é um administrador do *opCoders Judge*. No modelo anterior essa função era realizada utilizando os campos *bool_admin* e *bool_professor*, mas como mencionado anteriormente eles foram deletados. No *opCoders Judge* haverá vários tipos de administradores, diferenciando pela sua atuação acadêmica, por exemplo, podem utilizar a plataforma um professor e um pós-graduando, ambos atuando como administradores. Portanto, a solução para criar essa distinção foi a remoção do campo *bool_admin* e a criação da tabela *admin*, com o campo *type* para especificar sua atuação.

Em algumas tabelas foram removidos alguns campos considerados redundantes ou sem importância, como, por exemplo, na tabela **task**, nela foi removido o campo *qtd_questoes*, por ser interpretado que essa informação não é relevante ao ponto de ser armazenada no banco de dados.

A tabela **classgroup** foi a maior mudança do banco de dados. O que motivou essa alteração foi a necessidade de criação de uma segunda filtragem na página **Minhas Tarefas**, além da filtragem pelo tipo de tarefa, “aberta” e “finalizada”, foi constatado a necessidade de separar as tarefas considerando o semestre e a disciplina, pois especialmente para professores que utilizavam o *opCoders Judge* em várias disciplinas, estava se tornando muito caótico visualizar todas as tarefas de todas as suas turmas. Portanto, a solução encontrada para esse problema foi atribuir a tabela **classgroup** a responsabilidade de fazer esse controle de visualização de tarefas, por um relacionamento ternário entre **user**, **semester** e **discipline**. É importante ressaltar que o relacionamento entre as tabelas de usuário e turma possui um campo booleano especificado pelo nome de *read_only*, essa adição foi pensada para fazer o controle do caso de um usuário que não possa fazer submissões para uma turma em específico.

A tabela **task_classgroup** existe para atribuir diferentes datas de entrega para as tarefas. Dessa forma, nos registros da tabela **task** tem-se a especificação do enunciado, um registro dessa tabela faz relação com um da tabela **task_classgroup** para, através do campo *due_date* ser especificado a data de entrega. E assim, o registro da tabela **task_classgroup** é quem faz relação com o registro da tabela **classgroup**, que é a especificação da turma em si. Com isso, diferentes turmas podem ter datas de entrega para diferentes tarefas.

Uma grande mudança realizada da versão atual para a proposta nesta monografia, foi o acréscimo de dicas e mensagens de ajuda na plataforma. Foi criado a tabela **help_tip**, que diz respeito as mensagens de ajuda em relação ao uso do *opCoders Judge*, essa tabela não faz relacionamento com nenhuma outra, pois seu funcionamento é isolado das demais. Ela é utilizada apenas para armazenar os dados das mensagens.

A tabela **tip**, diz respeito as dicas que podem auxiliar o usuário na compreensão e realização de uma tarefa ou questão. Portanto, ela possui um relacionamento com a tabela **question** e **task**, vale ressaltar que uma mesma dica pode pertencer a uma questão e a uma tarefa que abriga essa mesma questão. Além disso, para melhorar ainda mais a compreensão do conteúdo difundido na plataforma, foi criada uma tabela **summary**, ela tem a função de resumir o conteúdo cobrado em uma tarefa, portanto ela também tem um relacionamento com a tabela **task**. Como pode ser observado na Figura 3.46, as tabelas **tip**, **help_tip** e **summary** tem os mesmos campos, isso se dá pelo seu funcionamento similar.

Por fim, um aspecto bem importante do banco de dados é sobre os campos *read_only*. Como explicado anteriormente, é possível que os direitos de submissão de código sejam revogados. É possível que uma turma inteira seja sinalizada para não receber mais submissões, assim como um usuário pode ter os direitos de submeter código revogados para todas as suas turmas, por fim também é possível que um usuário seja impedido de fazer submissões para uma turma em específico. Para atender essas aos dois primeiros casos, há campo um campo *read_only* na tabela **user** e um campo *is_active* na tabela **classgroup**. Além disso, essas duas tabelas possuem uma tabela para representar o relacionamento entre elas, chamada de **study_at**, essa tabela também possui um campo *read_only* para atender ao último caso.

3.3.3 Backend

A principal parte de um website mais complexo é o *backend*, ele é responsável por toda a lógica que o servidor da aplicação precisa para sustentar suas funcionalidades. Em outras palavras, é a fundação para o funcionamento coeso e a entrega de serviços de qualidade nas aplicações Web modernas. O *opCoders Judge* é uma plataforma institucional que precisa armazenar muitas informações para o seu funcionamento, nesse contexto é necessário um *backend* para realizar operações em um banco de dados e levá-las até o *frontend*, além de desenvolver toda a lógica de negócio para atender a demanda do produto.

A respeito das subseções que compõem a seção de *backend*, foi adotado um estilo semelhante à Seção 3.3.1. A Subseção 3.3.3.1 discorrerá sobre a arquitetura utilizada para organizar a estrutura do código. A Subseção 3.3.3.2 explicará os aspectos de segurança da aplicação. Por fim, a Subseção 3.3.3.3 falará sobre as funcionalidades implementadas.

3.3.3.1 Arquitetura utilizada

Por padrão, o Django utiliza a arquitetura MVC como explicado na Subseção 2.2.11. Diferentemente do que acontece com o *React*, ao inicializar um projeto Django é gerada uma estrutura de pastas mais constricta, onde cada pasta tem um objetivo definido mais especificamente.

No contexto do *opCoders Judge*, os *models* criados são as entidades ou tabelas do banco de dados, as *views* recebem a requisição *HTTP* do *frontend*, em seguida o *controller* realiza as devidas ações no model que contem as informações desejadas pelo *frontend*, em seguida uma resposta é retornada para este.

3.3.3.2 Segurança

A segurança é uma preocupação crítica em qualquer aplicação e essa questão deve levada ainda mais a sério em uma plataforma que armazena informações e possui um controle de usuário. Felizmente, o *Django Rest Framework* oferece várias camadas de segurança para proteger as APIs construídas com eles.

Como já dito na Subseção 3.3.1.2, no *opCoders Judge* utilizaremos a autenticação por meio de *token*, essa abordagem é uma escolha muito popular e prática. Nesse projeto o *backend* há diversos passos de segurança realizados durante o *login* e ainda são apoiados pela verificação de credenciais da *api Google* que é bastante confiável. Para garantir que nenhuma informação é vazada, as decodificações dos *tokens* mais importantes são realizadas pelo *backend*.

Além disso, o *Django Rest Framework* inclui um middleware que possui proteções integradas contra-ataques *Cross-Site Request Forgery* (CSRF) e *Cross-Site Scripting* (CSS), tornando a aplicação bem mais segura.

Também é válido ressaltar que o *Django Rest Framework* é bem eficiente em no tratamento de erros, ele evita que a divulgação de informações sensíveis ao usuário final, o que é importante para a segurança do sistema. Além disso, O *Django*, em conjunto com o *Django Rest Framework*, ajuda a proteger contra vulnerabilidades de injeção de SQL, garantindo que as consultas ao banco de dados sejam feitas de maneira segura.

3.3.3.3 Implementação de Funcionalidades

As funcionalidades que foram desenvolvidas neste trabalho consistem na implementação necessária para realizar **operações com o banco de dados**; algumas **lógicas de negócio** para atender ao sistema idealizado do *opCoders Judge*; o sistema de **login**, que também é uma lógica de

negócio, contudo merece um destaque especial devido a sua complexidade; e por fim a disposição e descrição dos **CRUDs** criados.

3.3.3.3.1 Operações com o Banco de Dados

O principal objetivo do *backend* nesse projeto é permitir que as operações com o banco de dados sejam feitas de maneira adequada. Para isso é necessário o desenvolvimento de um CRUD para cada uma das tabelas do modelo de entidade relacionamento. O desenvolvimento de um CRUD, utilizando o *Django Rest Framework*, é realizado em quatro passos.

Primeiramente, uma **classe de modelo** é definida para a tabela. Nessa classe, são especificados todos os campos da entidade, juntamente com as propriedades às quais estão sujeitos, como o número máximo de caracteres, se o campo é obrigatório, se pode ter um valor vazio, entre outros. Adicionalmente, é possível estabelecer lógicas mais complexas. Por exemplo, na tabela **task**, o usuário deve fornecer um valor para o campo *title*, enquanto o campo *slug* não terá seu valor definido manualmente. O conteúdo deste campo é gerado automaticamente pela cópia do valor do campo *title* e a aplicação de uma formatação que remove caracteres especiais, acentuações e substitui espaços por “-”, garantindo assim uma padronização adequada.

A seguir vem a serialização dos dados, é uma parte fundamental que desempenha um papel crucial na conversão de dados complexos, como objetos do modelo Django, em formatos que podem ser facilmente renderizados em JSON, XML ou outros tipos de conteúdo. Para isso, é criada uma **classe de serializer**. Nela é definido quais dos campos informados no modelo vão ser visíveis quando o CRUD for realizar alguma operação. Por exemplo, se um modelo tem os campos x, y e z, e no *serializer* é informado somente x e y, o resultado de uma operação de READ será apenas x e y, mesmo tendo o valor de z armazenado na tabela. É nessa etapa que é realizado a configuração que permite as chaves estrangeiras retornarem objetos JSON e não os valores de IDs.

A seguir é criada uma **classe de view**, nessa etapa é onde são utilizados o *serializer* e o modelo definidos anteriormente para criar funções que vão realizar as operações SQL e também definir a lógica de negócio. As operações SQL são basicamente o que é utilizado para realizar as operações de CRUD e a lógica de negócio consiste em desenvolver um funcionamento que atenda aos comportamentos esperados do produto.

3.3.3.4 CRUDs

É importante ressaltar que os CRUDs de todas as tabelas têm pelo menos funções para realizar as seguintes operações:

- Operação de leitura de um registro, definido pelo id;
- Operação de leitura de todos os registros;

- Operação de escrita de um novo registro, definido pelo id;
- Operação de alteração de dados de um registro, definido pelo id;
- Operação de delete de um registro, definido pelo id.

Contudo, algumas tabelas possuem algumas funções a mais para atender a necessidades específicas que o *frontend* possa requisitar. Essas funções basicamente consistem em realizar operações utilizando como parâmetro outros campos sem ser o id. Como, por exemplo, foi explicado anteriormente que a página **Tarefa Aberta** renderiza as informações da tarefa utilizando o *slug* presente na url, contudo o CRUD padrão, criado para todas as entidades, só consegue encontrar um único registro utilizando o ID, portanto foi criada uma nova função capaz de localizar um registro no banco de dados utilizando o campo *slug*. As demais funções extras criadas para outras tabelas tem um comportamento similar, também utilizando outro campo, no lugar do ID, para realizar alguma operação.

3.3.3.5 Lógica de Negócio

A tabela de **question_historic** tem como objetivo abranger o estado de uma questão, de forma que uma questão estará ligada a vários registros de históricos de questão. Nela constará quem fez a alteração e será utilizado o somente o histórico de questão criado por último, por ser o mais atual. Logo, Para atender essa lógica de negócio foi elaborado a seguinte funcionalidade:

- O registro de histórico de questão mais atual terá sempre seu campo *validity_final* com valor nulo, pois este campo demarca quando a validade daquele estado expirou, ou seja, quando um estado mais atual foi gerado;
- Quando um administrador criar um histórico de questão de uma questão pela primeira vez, esse campo estará definido como nulo;
- Uma vez que o histórico de questão de uma questão for criado, históricos subsequentes daquela mesma questão serão gerados toda vez que um administrador alterar as informações da questão;
- Quando um administrador realizar uma operação de PATCH para alterar os dados de uma questão, o registro mais atual terá somente o campo *validity_final* alterado, preenchido com a data e hora atual e será gerado um novo registro de histórico com esse campo vazio e com as alterações propostas pelo administrador feitas.

A tabela **user** também possui uma lógica de negócio. O objetivo aqui é criar uma melhor administração das imagens no banco de dados, pois o tratamento padrão do *Django* não é muito eficiente. Para tal é feito os seguintes procedimentos:

- Em uma operação de PATCH, é checado se o usuário está fornecendo uma nova imagem para atualizar seu perfil;
- Caso essa verificação tenha uma resposta positiva, é verificado se já há uma imagem vinculada aquele usuário no banco de dados;
- Caso haja, aquela imagem é deletada e a nova é salva no banco de dados. Caso não haja, a imagem simplesmente é salva no banco de dados.

Já para a tabela **task** e **task_classgroup**, é necessária uma lógica de negócio para o tratamento de alguns campos. Pois tanto o campo *slug* da primeira, quanto o *title* da segunda são definidos com base no conteúdo do campo *title* também da primeira. Logo é realizado a seguinte lógica de negócio:

- Em uma operação de PATCH, é checado se o administrador está fornecendo um novo title para a tarefa;
- Em caso positivo, tanto o campo *slug* quanto o campo *title*, da tabela **task_classgroup**, tem seus valores atualizados com base no novo valor informado;
- Por fim é atualizado o valor do campo *title* da tabela **task**.

3.3.3.6 Login

A lógica de negócio mais complexa, incontestavelmente, é a do processo de *login*. Trata-se de um sistema automatizado avançado que oferece uma interface prática, permitindo que o usuário faça *login* usando sua conta do *Google*. Nesse cenário, o sistema deve conseguir identificar o contexto do usuário e realizar o tratamento de dados necessário.

Os contextos possíveis pensados que um usuário que está tentando acessar o *opCoders Judge* pode se encaixar são:

- O usuário é um administrador;
- O usuário não é um administrador e está acessando a plataforma pela primeira vez.
- O usuário não é um administrador e não está acessando a plataforma pela primeira vez.
- O usuário não tem acesso à plataforma.

É de extrema importância que o sistema de *login* possua uma lógica eficiente para compreender o contexto de cada usuário, uma vez que diferentes ações são adotadas em cada caso. Para auxiliar no reconhecimento desse contexto, a equipe de Tecnologia da Informação (NTI) da UFOP foi solicitada a criar um microserviço capaz de retornar as disciplinas nas quais

um aluno foi matriculado em algum período, fornecendo o e-mail do aluno e o semestre como parâmetros. Esse microserviço desempenha um papel fundamental na decisão de conceder ou não permissão de acesso ao sistema. Ao verificar as disciplinas registradas no banco de dados da plataforma, é possível determinar, por meio do retorno do microserviço, se o aluno está matriculado em disciplinas que utilizam o *opCoders Judge*.

Entretanto, considerando que os códigos desenvolvidos pelos usuários podem continuar sendo relevantes mesmo após o término da matrícula em uma disciplina que utiliza a plataforma, foi implementada a possibilidade de acesso para usuários que não estão mais matriculados, mas que utilizaram a plataforma em algum momento durante sua matrícula. Essa flexibilidade visa permitir que os usuários possam continuar utilizando a plataforma, mesmo após encerrarem suas atividades acadêmicas relacionadas à disciplina.

Para a implementação dessa lógica de negócio, foi desenvolvido o seguinte funcionamento:

1. A API do *Google* verifica as credenciais e retornar um *token*, contendo as informações do perfil *Google* do usuário, porém criptografadas;
2. Utilizando uma funcionalidade oferecida pela *API Google*, o *token* é validado e decodificado, obtendo assim as informações do usuário;
3. É realizado uma operação **READ** nos registros da tabela **user**, utilizando como base o e-mail *Google*;
4. É verificado se o usuário é administrador, se sim o *login* é efetuado;
5. Utiliza-se o microserviço para definir o conjunto denominado **Matriculadas**, este conjunto corresponde as disciplinas que o usuário está matriculado e são utilizadas pelo *opCoders Judge*;
6. É realizado uma busca no banco de dados para definir o conjunto de **Vinculadas**, isto é, as turmas vinculadas ao usuário;
7. Caso os dois conjuntos estejam vazios, o usuário não possui acesso à plataforma e o *login* não é efetuado;
8. A partir desses dois conjuntos são gerados outros três conjuntos:
 - Conjunto **Novas Turmas**, que corresponde as disciplinas que estão em **Matriculadas**, mas não constam em **Vinculadas**. Caso haja disciplinas nesse conjunto, significa que o aluno está matriculado em algumas turmas, no sistema da UFOP, que o *opCoders Judge* não está ciente;

- Conjunto **Antigas Turmas**, que corresponde as disciplinas que estão em **Vinculadas**, mas não constam em **Matriculadas**. Caso haja disciplinas neste conjunto, significa que o aluno não está mais matriculado, no sistema da UFOP, em disciplinas que o *opCoders Judge* entende que está;
 - Conjunto **Mesmas Turmas**, que corresponde as disciplinas que estão em ambos **Matriculadas** e **Vinculadas**. Caso haja disciplinas neste conjunto, significa que o aluno está matriculado, no sistema da UFOP, em disciplinas que o *opCoders Judge* entende que está, mas como é possível haver várias turmas em uma mesma disciplina, pode haver diferença na enumeração da turma.
9. **Novas Turmas**: Se houver alguma turma nesse conjunto, o sistema irá vincular ela ao usuário;
 10. **Antigas Turmas**: Se houver alguma turma nesse conjunto, o sistema irá atribuir o valor do campo *read_only* da relação **user** com **classgroup** daquele usuário como verdadeiro, sinalizando que ele não pode mais fazer submissões;
 11. **Mesmas Turmas**: Se houver alguma turma nesse conjunto, é checado a enumeração do código das turmas, se houver alguma diferença, significa que o usuário trocou de turma, logo assim como anteriormente, o campo *read_only* da relação tem seu valor atribuído como verdadeiro e a turma com o código vindo do microserviço é vinculada ao usuário;
 12. *Login* efetuado com sucesso.

A partir dessa implementação, o contexto do usuário pode ser identificado com sucesso e é possível realizar todas as devidas operações de dados, antes de o *login* ser autorizado. O funcionamento pode ser melhor compreendido na Figura 3.47.



Figura 3.47 – Fluxograma do Processo de Login.

4 Considerações Finais

Neste capítulo serão apresentadas as considerações finais a respeito do trabalho realizado. A Seção 4.1 abordará as conclusões obtidas tendo em vista um panorama do que foi feito. Já a Seção 4.2 apresenta os possíveis trabalhos, melhorias e correções a serem realizados para que a versão do *opCoders Judge* proposta aqui, se torne uma plataforma completa, sólida e eficiente.

4.1 Conclusão

A crescente integração da tecnologia na educação melhora a maneira como os alunos aprendem e interagem com os conceitos da ciência da computação, a prática desses conceitos é de extrema importância para a consolidação do aprendizado. Dentro desse contexto, as ferramentas digitais desempenham um papel crucial ao proporcionar ambientes dinâmicos, que complementam o ensino de sala de aula. Entre essas ferramentas, os corretores de código-fonte emergem como uma peça fundamental, promovendo o desenvolvimento de habilidades e competências essenciais.

No âmbito da criação e operação de uma plataforma online de correção de código-fonte, a qualidade é um pilar fundamental para garantir o seu funcionamento eficiente e eficaz. A qualidade, nesse contexto, abrange diversos aspectos que dizem respeito a todos os estágios de desenvolvimento e implementação da plataforma. Por esse motivo esse trabalho tratou de todos os aspectos julgados essenciais para o desenvolvimento adequado do *opCoders Judge*, aperfeiçoamento de interface buscando melhorar a usabilidade e a estética, implementação do *frontend* e *backend* e por fim remodelagem do banco de dados.

Dando sequência ao trabalho de [Cedraz \(2023\)](#), foi realizado dois novos testes de usabilidade, fundamentais para o melhor entendimento das necessidades de interface, pois estes contaram com a participação de seu público alvo, alunos da disciplina Programação de Computadores I. O primeiro teste foi realizado com alunos de diferentes cursos desta disciplina, de maneira individual e sem separação. Já o segundo foi feito com turmas específicas de uma vez. Como os testes constituem-se da continuação de outro trabalho, é de grande importância que o mesmo padrão fosse mantido ao máximo. Os resultados gerais contribuíram para confirmar a hipótese que motivava esta tarefa, os estudantes de Programação de Computadores I tiveram mais dificuldade ao interagir com a interface do *opCoders Judge*, do que os alunos da computação que participaram do teste de usabilidade anterior. É importante relatar que o segundo teste utilizou de um protótipo alterado com base no primeiro teste, utilizando seus resultados como base. Dito isso, os resultados do segundo teste foram melhores do que o primeiro, dessa forma podemos concluir que as alterações tiveram um efeito positivo, contudo foi necessário realizar outras alterações, com a finalidade aproximar ainda mais a qualidade de interação dos alunos de Programação de Computadores I dos alunos do curso de Ciência da Computação.

Um dos aspectos centrais neste trabalho diz respeito às modificações realizadas no protótipo inicial proposto por [Cedraz \(2023\)](#). Essas adaptações surgiram após a condução dos testes de usabilidade, já comentado anteriormente. O principal objetivo desta etapa era a criação de um ambiente mais atraente e moderno para a interação dos usuários com o *opCoders Judge*. A motivação dessas alterações foi promover uma experiência de usuário enriquecedora, onde os usuários se sintam mais confortáveis e engajados ao utilizar a plataforma. É relevante ressaltar que o protótipo inicial revelou-se eficaz ao tratar as questões de usabilidade identificadas, colaborando para que poucas modificações, a respeito de usabilidade, fossem necessárias nesta monografia.

A respeito do planejamento e implementação do *frontend* e *backend*, foi realizado um trabalho com o foco priorizando as funcionalidades julgadas fundamentais para o funcionamento do *opCoders Judge*. Esta é a fase que utiliza os resultados das duas etapas anteriores para criar o *opCoders Judge* de fato. Em ambos os casos, todas as funcionalidades, da perspectiva do usuário, foram implementadas conforme o planejamento do protótipo. Uma plataforma como esta requer um espaço para ocorrer a administração, nele são realizadas operações como: criação de questões, tarefas, dicas, mensagens de ajuda, etc. O *backend* possui implementação para atender os requisitos destas operações. Entretanto, no que diz respeito ao *frontend*, são necessárias interfaces que as permitam, contudo, a criação destas interfaces não fez parte do escopo desse trabalho, pois a ferramenta usada no *backend*, *Django Rest Framework*, gera automaticamente uma interface que permite a realização das operações mencionadas. Contudo, essa interface não possui o mesmo padrão de design das demais interfaces do *opCoders Judge*. A partir disso, é possível concluir que a plataforma está apta para ser utilizada.

Outro tópico trabalhado nesta monografia foi a remodelagem do banco de dados, feita para atender às novas necessidades e realizar alguns polimentos no antigo sistema. Também foram acrescentados componentes no banco de dados, para que ele tenha suporte para questões dinâmicas. Sendo assim, o modelo de banco de dados desenvolvido nesta monografia constitui na junção dos trabalhos feitos por [Patrocínio \(2023\)](#) e [Mendonça \(2023\)](#), com algumas adaptações e acréscimos julgados necessários. Ao final desta etapa, conclui-se que o banco de dados atende à função que lhe é atribuída.

Ao final deste estudo, é possível concluir que as diferentes partes trabalhadas se conectam bem. Os testes de usabilidade realizados com os alunos da disciplina Programação de Computadores I e Introdução a Programação renderam resultados úteis para as alterações no protótipo, que por sua vez serviu de base para o desenvolvimento do *frontend* e este é apoiado pela interação do *backend* com o banco de dados. Tudo isso se soma para criar um projeto onde todos os esforços se alinham, buscando tornar o *opCoders Judge* uma ferramenta eficaz para o ensino e prática da programação. Contudo, devido ao grande volume de trabalho realizado e às restrições de tempo, os testes de funcionamento foram feitos de maneira manual, não foi possível a implementação de testes automatizados, mecanismos muito importantes nesse tipo de projeto, para validar totalmente o funcionamento e eficiência dos componentes implementados.

4.2 Trabalhos Futuros

É importante reconhecer que este projeto, embora tenha alcançado o estado final, ainda pode contar com diversas melhorias e correções, colaborações que tornarão o *opCoders Judge* uma ferramenta ainda mais completa e eficiente.

É muito importante conduzir um novo teste de usabilidade com os estudantes da disciplina Programação de Computadores I e Introdução a Programação seguindo os modelos do segundo teste aplicado. Conforme exposto nesta monografia, muitas alterações foram feitas, tornando vital a obtenção de informações que validem as mudanças. É muito importante não realizar alterações demais na abordagem utilizada, para permitir uma melhor comparação com os testes anteriores, entretanto uma mudança bem interessante é utilizar a própria plataforma como ambiente de teste. Tal abordagem possibilitará uma avaliação mais precisa dos efeitos das alterações e a detecção de eventuais problemas de usabilidade resultantes das mudanças implementadas, que possam não ter sido percebidos. E claro, se novos problemas forem identificados, deve-se pensar em soluções no protótipo, antes de avançar para a implementação.

Um aspecto muito importante atualmente, no que diz respeito ao *frontend*, é a **responsividade**. Isto é, capacidade das interfaces da plataforma de se adaptarem para as diferentes resoluções dos aparelhos eletrônicos capazes de acessar o *opCoders Judge*. Embora as interfaces da plataforma funcionem bem caso um usuário acesse a plataforma com uma resolução de *desktop* ou *notebook*, elas não possuem essa capacidade de adaptação e um usuário que acessar por meio de tablet ou celular terá dificuldades em navegar por elas.

Ainda sobre o *frontend*, é fundamental a criação de interfaces para a área de administração da plataforma. É importante que todas as interfaces que compõem o *opCoders Judge* sigam o mesmo padrão, para dar uma sensação de uniformidade e fluidez. Visando validar o design das interfaces, é interessante pensar na elaboração de testes de usabilidade direcionado ao público desta área da plataforma. A área de administração não será acessada por usuários normais, como alunos de alguma turma que utiliza do *opCoders Judge*, ela está destinada a professores, monitores e outras pessoas que possuam uma posição de tutor. Logo, em caso de aplicação de teste de usabilidade nas interfaces nessa área, os voluntários realizadores do teste, tem de pertencer ao perfil mencionado.

Por fim, para que os componentes implementados tenham seu funcionamento totalmente validado, é necessário implementar testes automatizados, para garantir que as funcionalidades estão funcionando conforme o desejado. Recomenda-se realizar uma pesquisa para decidir qual biblioteca de testes utilizar. Contudo, em relação ao *backend*, o *Django* já vem com a *unittest* implementada. Para o *frontend*, uma biblioteca muito popular para elaboração de testes de uma aplicação feita com *React*, é o *Jest*.

Referências

- BARBOSA, B. S. S. **Interação Humano-Computador**. [S.l.]: Elsevier Brasil, 2010.
- BASTIEN, J. M. C. Usability testing: a review of some methodological and technical aspects of the method. **Elsevier**, v. 1, p. 6, 2010.
- BRITO, P. S. S. O uso de corretores automáticos para o ensino de programação de computadores para alunos de engenharia. **UFOP**, v. 1, p. 10, 2019.
- BRITO, P. S. S. **O Uso de Ferramentas Computacionais Para o Ensino de Programação Para Alunos de Engenharia**. Monografia de Bacharelado — Universidade Federal de Ouro Preto, 2019.
- CARVALHO, M. de. **Construindo o saber: técnicas de metodologia científica**. [S.l.]: Papirus Editora, 1989. ISBN 9788530800710.
- CEDRAZ, V. F. **Uma Avaliação de Usabilidade do Corretor de Exercícios de Introdução à Programação OpCoders Judge**. Monografia de Bacharelado — Universidade Federal de Ouro Preto, 2023.
- CHERNY, B. **Programming Typescript Making Your Javascript Applications Scale**. [S.l.]: O'Reilly, 2019.
- FEDOSEJEV, A. **React.js Essentials**. 1. ed.. ed. [S.l.]: Packt Publishing, 2015.
- FORCIER PAUL BISSEX, W. C. J. **Python Web Development With Django**. 1. ed.. ed. [S.l.]: Addison Wesley, 2008.
- GANATRA, S. React router quick starter guide. **Packt**, v. 1, p. 135, 2018.
- HOED, R. M. **Análise da evasão em cursos superiores: o caso da evasão em cursos superiores da área de Computação**. Dissertação de Mestrado — Universidade de Brasília, 2016.
- IMTIAZ, S. The psychology behind web design. **McMaster University**, v. 1, p. 29, 2016.
- MACEFIELD, J. M. S. R. **How to Determine the Right Number of Participants for Usability Studies**. 2016. Acesso em: 31 julho de 2023. Disponível em: <https://www.uxmatters.com/mt/archives/2016/01/how-to-determine-the-right-number-of-participants-for-usability-studies.php>.
- MATSUMOTO, C. Y. A importância do banco de dados em uma organização. **Maringá Management**, v. 1, p. 11, 2008.
- MENDONÇA, T. S. **Projeto e Desenvolvimento de uma Plataforma de Gestão de Questões Dinâmicas para o Ensino de Programação de Computadores**. Monografia de Bacharelado — Universidade Federal de Ouro Preto, 2023.
- NIELSEN, J. **Quantitative Studies: How Many Users to Test?** 2006. Acesso em: 31 julho de 2023. Disponível em: <https://www.nngroup.com/articles/quantitative-studies-how-many-users/>.

- OBE, L. H. R. **PostgreSQL – Up and Running**. 1. ed.. ed. [S.l.]: OReilly, 2012.
- PARASHAR, U. K. G. S. T. Responsive webpage using html css. **Institute of Electrical and Electronics Engineers**, v. 1, p. 4, 2022.
- PARK, T. C. J. **PHP Bible**. 2. ed.. ed. [S.l.]: Wiley, 2002.
- PATROCÍNIO, J. A. do. **OpCoders Judge: Uma Versão Online Para o Corretor Automático de Exercícios de Programação do Projeto OpCoders**. Monografia de Bacharelado — Universidade Federal de Ouro Preto, 2023.
- RAMPAZZO, L. **Metodologia científica**. [S.l.]: Edições Loyola, 2005. ISBN 9788515024988.
- RATN, S. R. S. R. Web development for different interesting systems. **DSpace/Manakin Repository**, v. 1, p. 22, 2022.
- SAURO, J. **Measuring Usability with the System Usability Scale**. 2011. Acesso em: 31 julho de 2023. Disponível em: <<https://measuringu.com/sus/>>.
- SCHOGER, A. W. J. R. D. H. S. **How to Determine the Right Number of Participants for Usability Studies**. 2017. Disponível em: <<https://tailwindcss.com/docs/installation>>.
- SPATH, P. **Beginning Java MVC 1.0: Model View Controller Development to Build Web, Cloud, and Microservices Applications**. 1. ed.. ed. [S.l.]: Apress, 2020.
- VAINIKKA, J. Full-stack web development using django rest framework and react. **Metropolia**, v. 1, p. 34, 2018.

Anexos

ANEXO A – Interfaces da Versão Atual do *opCoders Judge*

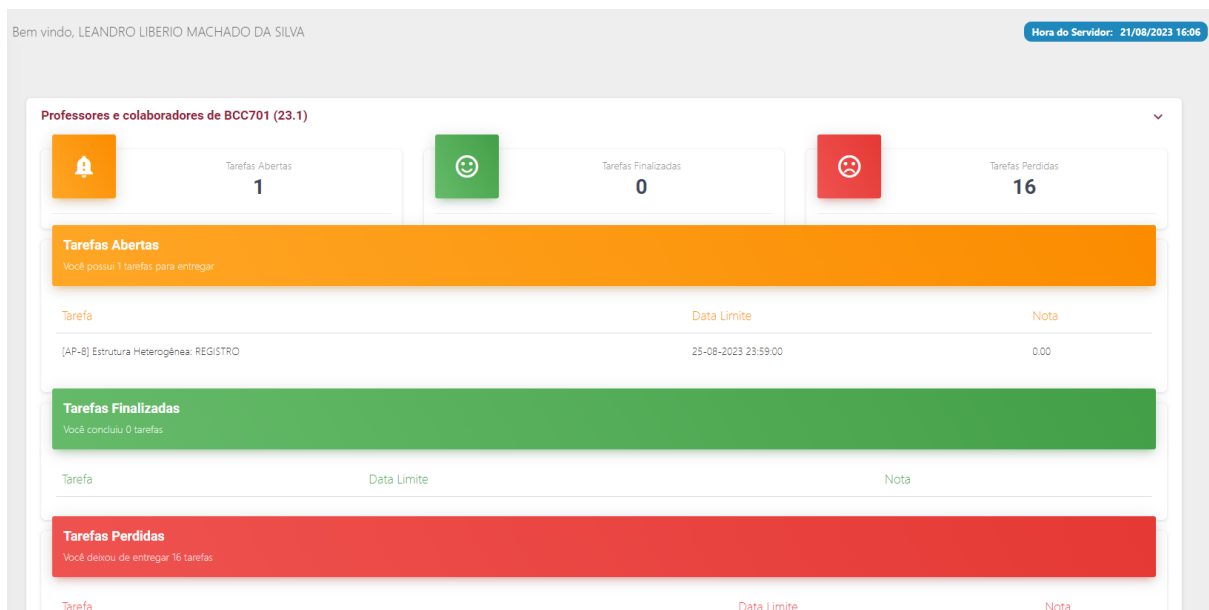


Figura A.1 – Interface da página “Minhas Tarefas” atual do Opcoders.

A Figura A.1 mostra a versão atual da página **Minhas Tarefas**, é possível observar que o escalamento de todas as tarefas é na vertical, tornando quase que obrigatório o uso de “scroll”.

Para a navegação da plataforma, como observado na Figura A.2, o componente de *header* permite que o usuário navegue pelas páginas **Minhas Tarefas** e **Perfil**, que compõem as páginas referentes a perspectiva do usuário comum. Os administradores possuem outras páginas para poderem cuidar das questões administrativas, tais como criar questões e tarefas, por exemplo.

A página **Perfil** permite que o usuário visualize alguns dos seus dados pessoais, cadastrados utilizando o sistema da UFOP e também alterar a sua senha. Esta é uma funcionalidade importante, pois a senha do usuário é gerada automaticamente por meio de caracteres aleatórios, logo é uma senha muito difícil de ser lembrada pelos usuários. A interface da página pode ser observada na Figura A.3.

O sistema de *login*, possui uma interface simples, utilizando de cores claras em sua interface e e-mail e senha como credenciais de acesso. Também possui a funcionalidade de recuperar a senha caso o usuário a tenha perdido. Como ilustrado na Figura A.4.



Figura A.2 – Header atual do Opcoders.

Bem vindo, LEANDRO LIBERIO MACHADO DA SILVA Hora do Servidor: 21/08/2023 16:07

Dados pessoais:

Nome

LEANDRO LIBERIO MACHADO DA SILVA

Email

leandro.liborio@aluno.ufop.edu.br

CPF

***928.016-**

Alterar senha:

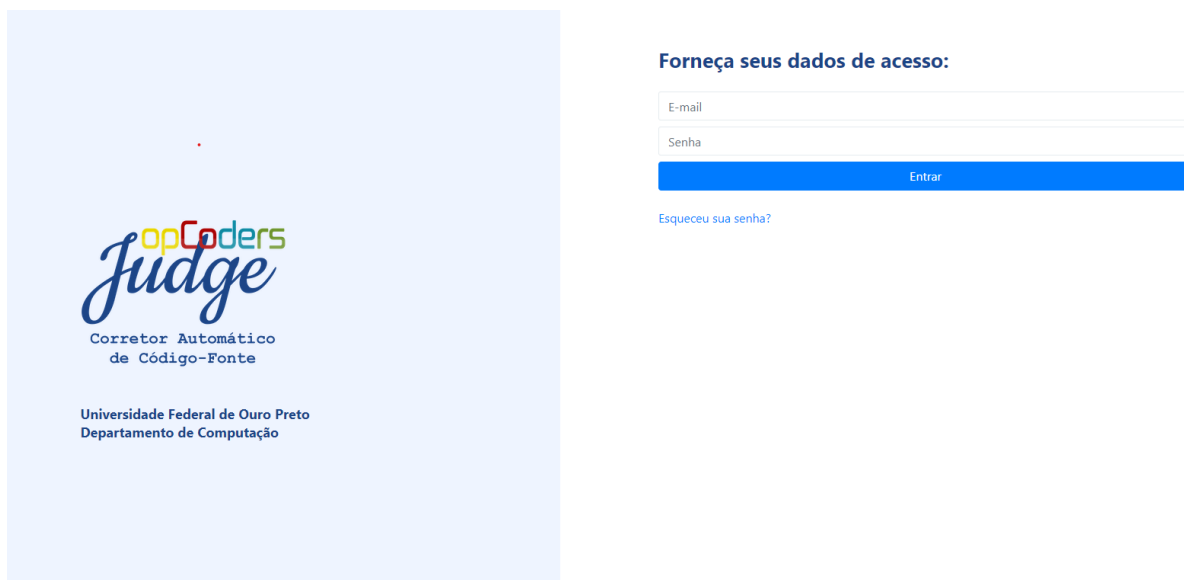
Senha atual:

Nova senha:

Repetir nova senha:

ALTERAR SENHA

Figura A.3 – Interface da página “Perfil” atual do Opcoders.



opCoders
Judge

Corretor Automático
de Código-Fonte

Universidade Federal de Ouro Preto
Departamento de Computação

Forneça seus dados de acesso:

E-mail

Senha

Entrar

[Esqueceu sua senha?](#)

Figura A.4 – Interface da página de *Login* atual do Opcoders.

ANEXO B – Termo de Consentimento Livre e Esclarecido (TCLE)

Título da pesquisa: “Avaliação de usabilidade de ferramentas online no ensino de programação de computadores”.

Desenvolvido por: Prof. Dr. Saul Emanuel Delabrida Silva, Prof. Dr. Reinaldo Silva Fortes e Vivyann Fernandes Cedraz.

Prezado (a) participante,

Convidamos você a participar voluntariamente do estudo **“Avaliação de usabilidade de ferramentas online no ensino de programação de computadores”**, elaborado pelos professores Saul Emanuel Delabrida Silva e Reinaldo Silva Fortes, e pela aluna Vivyann Fernandes Cedraz.

O presente trabalho tem como objetivo geral avaliar a usabilidade da interface configurada do Moodle e da ferramenta de correção automática de código-fonte, OP Coders Judge.

O convite a sua participação se deve ao fato de que a avaliação das ferramentas devem ser feitas por alunos do ensino superior. Os resultados dos testes serão usados pelos pesquisadores para avaliar a usabilidade da interface dos sistemas.

Sua participação é voluntária, ou seja, não é obrigatória, e você tem plena autonomia para decidir se quer ou não participar, bem como suspender sua participação a qualquer momento, mesmo tendo iniciado as atividades. Você não será penalizado de nenhuma maneira caso decida não consentir sua participação, ou desistir da mesma. Contudo, ela é muito importante para conclusão da pesquisa.

Para participar você deverá fazer acesso à plataforma Moodle e ao OP Coders Judge por meio de computador com acesso à internet e realizar o conjunto de tarefas proposto. As tarefas são:

- Consultar material da aula 01;
- Abrir o Moodle da disciplina de Programação, responder a Atividade 01 e submeter;
- Consultar o resultado da correção da Atividade 01;
- Enviar uma dúvida ao professor da disciplina através do Moodle, de modo que todos os colegas de classe consigam acompanhar a discussão;
- Acessar o diário de classe e conferir notas;
- Acessar a bibliografia da disciplina;

- Identificar qual a data da primeira prova.

Ao final de realização das tarefas, devem ser preenchidos 2 formulários sobre a avaliação de usabilidade da interface configurada do Moodle e 2 formulários sobre a avaliação de usabilidade do sistema OP Coders Judge. Estes questionários fazem parte do objeto de pesquisa sobre a percepção dos participantes sobre as interfaces.

A pesquisa a ser desenvolvida tem aprovação do Comitê de Ética, segundo a Resolução CNS 510/2016, obedecendo às diretrizes e normas reguladoras de pesquisas que envolvem seres humanos. Essa Resolução abrange os cinco pilares básicos da bioética que são: autonomia, não maleficência, beneficência, justiça e equidade, visando assegurar os direitos e deveres sobre a comunidade científica, aos participantes da pesquisa e ao Estado.

Esta pesquisa apresenta riscos mínimos aos participantes já que se trata de avaliação de usabilidade. Para mitigar possíveis quebras de sigilo garante-se que apenas os pesquisadores terão acesso ao banco de dados. Se o(a) Sr(a) sentir constrangimento ao responder alguma pergunta, terá liberdade para não responder ou para interromper o preenchimento do questionário a qualquer momento. Em caso de desistência, os dados serão apagados do banco de dados. Manteremos apenas o registro da informação de desistência anonimizado para fins estatísticos como métrica da pesquisa.

Sua participação no estudo não deve acarretar despesas ou custos. Haverá garantia de ressarcimento se fizer necessário. Se houver algum dano, comprovadamente decorrente da presente pesquisa, você terá direito à indenização, por meio das vias judiciais, como dispõem o Código Civil, o Código de Processo Civil, e na Resolução nº 510/2016, do Conselho Nacional de Saúde.

Os benefícios e vantagens em participar é que esta pesquisa poderá implicar em melhorias na interface das ferramentas e possibilitar a boa experiência dos usuários.

Todos os documentos relativos à pesquisa serão guardados em local restrito pelo prazo de 5 (cinco) anos em custódia dos pesquisadores. Com o fim deste prazo todos os documentos da pesquisa serão descartados.

Como a identificação do participante da pesquisa é indispensável pela característica analítica do projeto, informamos que os dados coletados serão analisados de forma anônima para fins de avaliação e divulgação científica. Os dados poderão ser apagados se requisitado pelo participante da pesquisa, mesmo após o término das atividades.

Ressaltamos que a pesquisa está em conformidade com a Resolução 510/2012 do CNS e Ofício Circular nº 2/2021/CONEP/SECNS/MS.

Os resultados gerais poderão ser divulgados em palestras dirigidas ao público participante, artigos científicos e monografias/dissertações/teses. Os resultados de forma individual serão repassados aos participantes, a equipe de pesquisadores responsáveis fica à disposição para

eventuais esclarecimentos.

Caso tenha alguma dúvida sobre a pesquisa, durante a sua participação ou posteriormente, o(a) Sr(a) poderá entrar em contato conosco por meio dos contatos que estão explicitados neste Termo.

Em caso de dúvidas sobre o estudo, você poderá entrar em contato com os pesquisadores: Reinaldo Silva Fortes (reifortes@ufop.edu.br), Saul Emanuel Delabrida Silva (e-mail: saul.delabrida@ufop.edu.br) ou Vivyann Fernandes Cedraz (vivyann.cedraz@aluno.ufop.edu.br).

Em caso de denúncias ou reclamações sobre sua participação e sobre questões éticas do estudo, você pode entrar em contato com a secretaria do **Comitê de Ética em Pesquisa (CEP) da Universidade Federal Ouro Preto**. Endereço: Centro de Convergência, Campos Universitário, UFOP. Telefone: (31) 3559-1368. Email: cep.propp@ufop.edu.br.

O Comitê de Ética em Pesquisa envolvendo seres humanos (CEP) tem como finalidade defender os interesses dos participantes da pesquisa em sua integridade e dignidade e contribuir no desenvolvimento da pesquisa dentro de padrões éticos, sendo responsável pela avaliação e acompanhamento dos aspectos éticos de todas as pesquisas envolvendo seres humanos.

Este Termo de Consentimento Livre e Esclarecido deverá ser assinado pelo pesquisador responsável e pelo participante do estudo e rubricado em todas as folhas da via. É muito importante que você guarde em seus arquivos uma cópia deste documento

Consentimento Livre e Esclarecido

Após receber os esclarecimentos sobre os objetivos, importância e o modo como os dados serão coletados nessa pesquisa, além de conhecer os riscos, desconfortos e benefícios que ela trará para mim e ter ficado ciente de todos os meus direitos, concordo em participar da pesquisa **Avaliação de usabilidade de ferramentas online no ensio de programação de computadores.**

CONSENTIMENTO

Eu _____, após ter sido suficientemente esclarecido (a) pelo pesquisador sobre a realização desta pesquisa, como está escrito neste Termo, declaro que consinto em participar da pesquisa por livre e espontânea vontade.

Data: ____/____/____

Assinatura: _____

Pesquisador Responsável - Assinatura: _____

Prof. Dr. Reinaldo Silva Fortes

ANEXO C – Questionário de Perfil

O presente formulário tem o objetivo de conhecer o perfil dos participantes da pesquisa "Avaliação de Usabilidade de Ferramentas Online no Ensino de Programação de Computadores".

1) Qual o seu gênero?

- a. Feminino
- b. Masculino
- c. Prefiro não dizer
- d. Outros

2) Qual o seu curso?

3) Em qual período da graduação você está? (Informe apenas o número. Ex. 1; 2; 3 etc.)

4) Já teve contato com o Moodle antes?

- a. Sim
- b. Não

5) Já usou algum corretor automático de código-fonte antes?

- a. Sim
- b. Não

6) Gosta de programação?

- a. Sim
- b. Não

7) Já programava antes de iniciar a disciplina de introdução a programação?

a. Sim

b. Não

8) Está cursando a disciplina de introdução a programação pela primeira vez?

a. Sim

b. Não

9) O que você achou da interface da ferramenta?

Apêndices

APÊNDICE A – Interface da página *Ajuda do opCoders Judge*

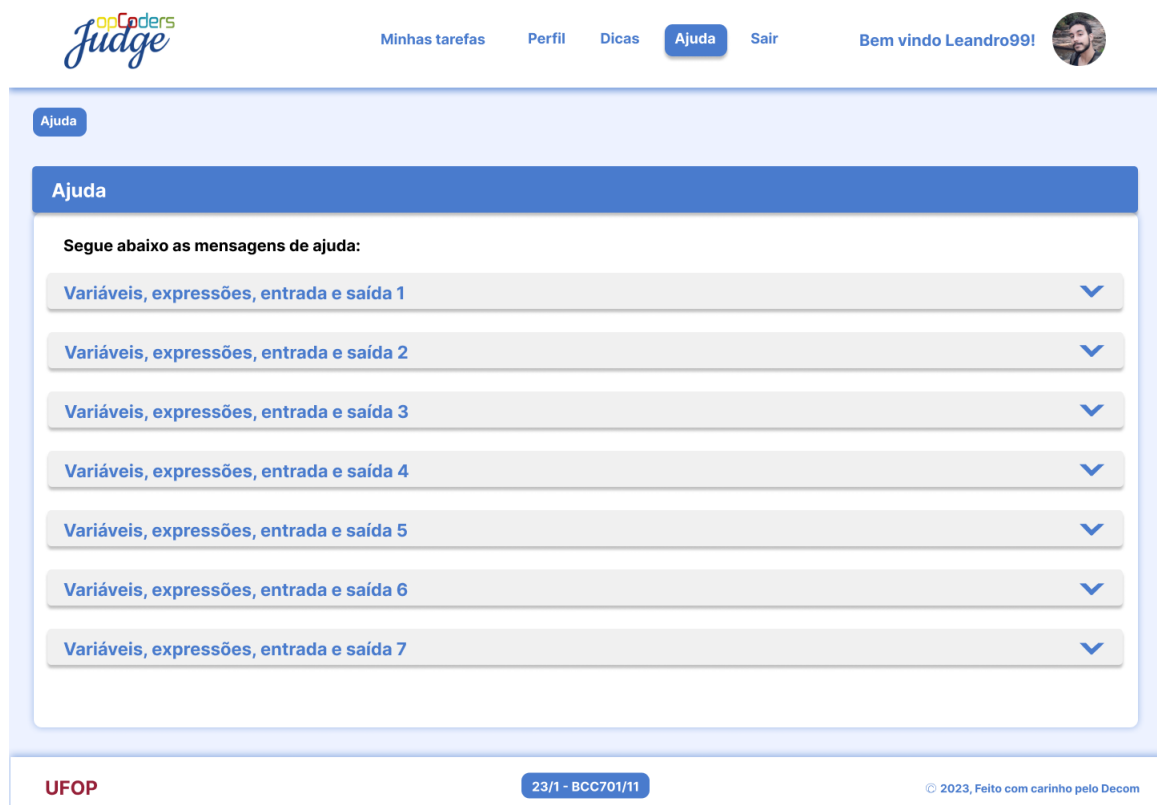


Figura A.1 – Interface da página **Ajuda**.

A Figura A.1 mostra a página **Ajuda**, proposta por essa monografia. Ela consta com componentes retangulares dispostos em um “container”, o contraste das cores é utilizado para mostrar a hierarquia dos componentes. Caso haja um “overflow” de componentes de ajuda, um scroll interno no “container” aparecerá.

APÊNDICE B – Design dos novos pop-ups do *opCoders Judge*

Neste apêndice será mostrado os diferentes tipos de *pop-ups* utilizados pelo *opCoders Judge*.

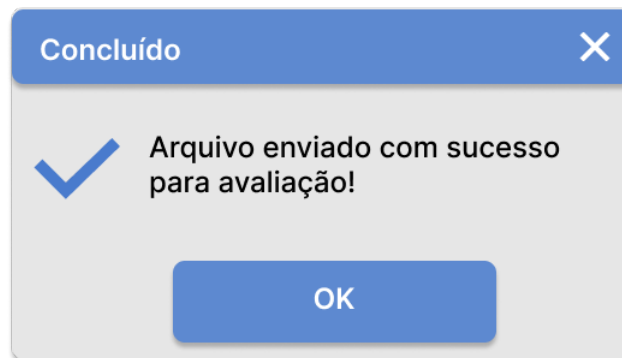


Figura B.1 – Pop-up de alertas.

A Figura B.1 ilustra o *pop-up* mais simples possível. Ele é utilizado para mostrar o resultado de uma submissão do usuário. Quando o usuário vai entregar o código-fonte, algumas ações devem ter sido feitas: ele tem de ter escolhido uma linguagem de programação, um arquivo de código-fonte e a extensão do arquivo tem que ser condizente com a linguagem de programação. Caso alguma dessas condições não tenha sido seguida, um *pop-up* aparecerá na tela relatando o erro em questão. Caso não haja nenhum erro, aparecerá um *pop-up* confirmando a submissão. É válido ressaltar que este componente é possível variar o título, o texto e o ícone.

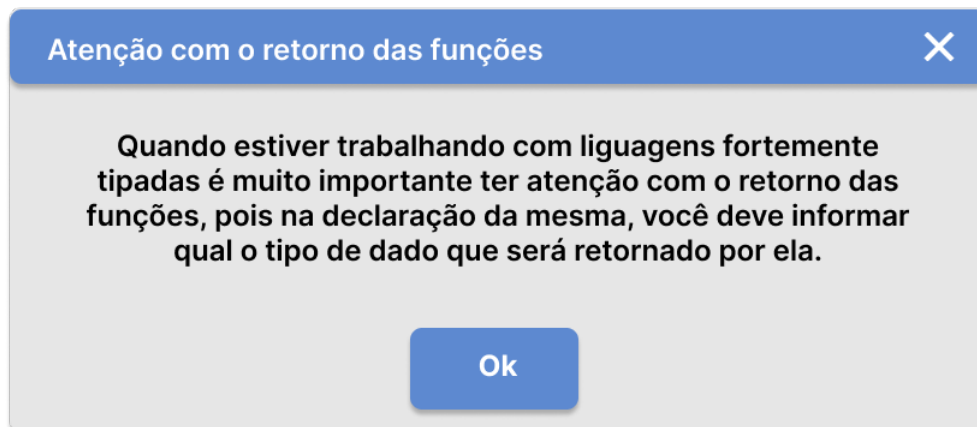


Figura B.2 – Pop-up de dicas e resumo.

Já a Figura B.2, mostra que o modelo de *pop-up* referente as dicas e resumo que um administrador pode vincular a questões ou tarefas. A diferença deste componente para o *pop-up* anterior, é que este não possui ícone e ele foi modelado para conter textos maiores.

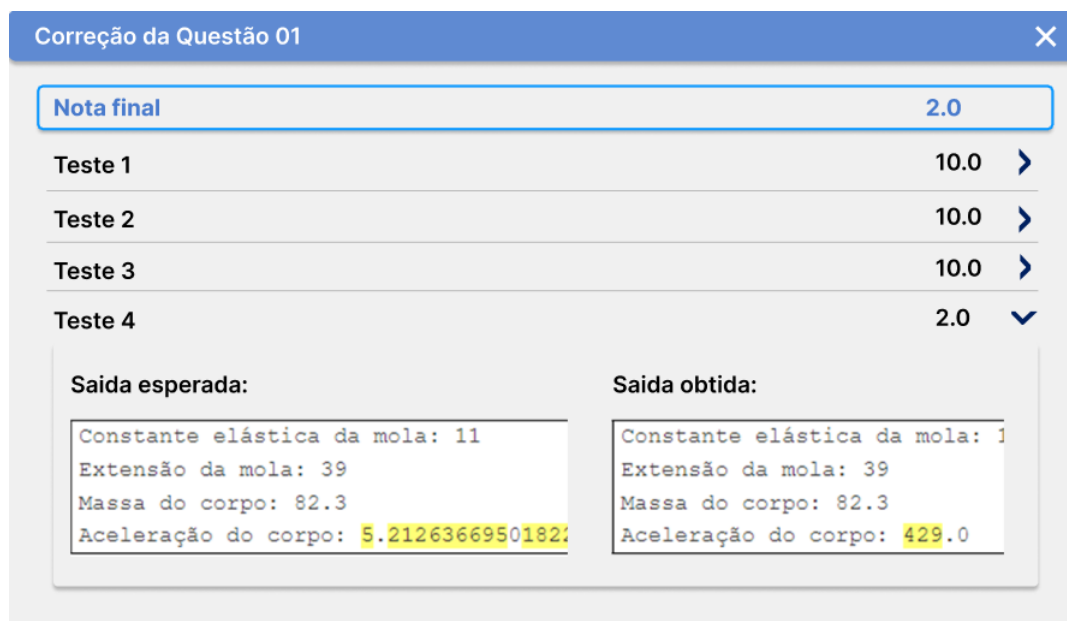


Figura B.3 – Pop-up de correções.

Por fim, a Figura B.3 mostra como é o componente de *pop-up* destinado a mostrar a correção do código da submissão. Ele é composto por uma lista que mostra cada teste que avalia a questão e sua respectiva nota. Quando o usuário clicar no teste será mostrado a diferença da resposta esperada pelo corretor, para a resposta do programa enviado pelo usuário.