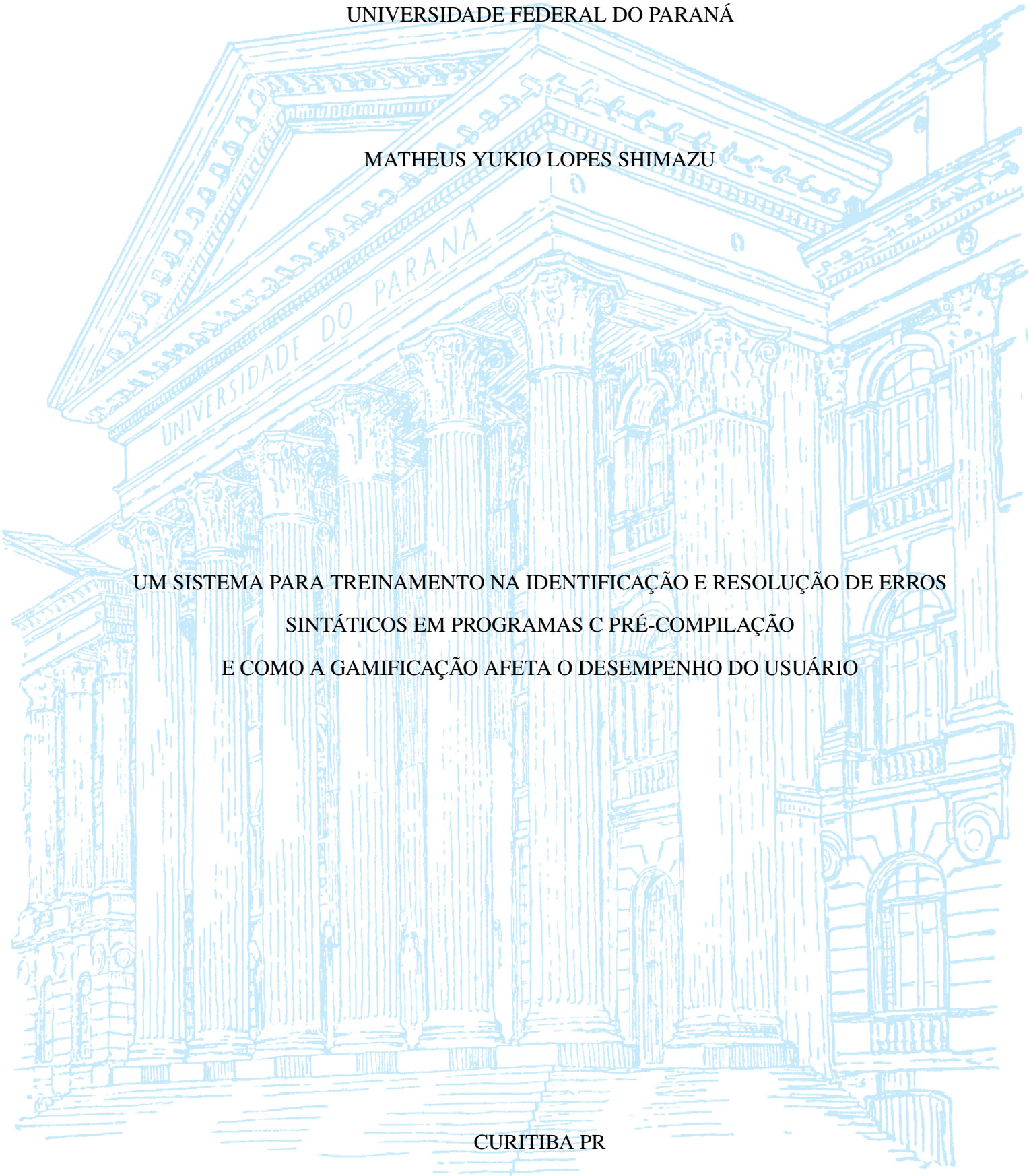
UNIVERSIDADE FEDERAL DO PARANÁ

MATHEUS YUKIO LOPES SHIMAZU



UM SISTEMA PARA TREINAMENTO NA IDENTIFICAÇÃO E RESOLUÇÃO DE ERROS SINTÁTICOS EM PROGRAMAS C PRÉ-COMPILAÇÃO
E COMO A GAMIFICAÇÃO AFETA O DESEMPENHO DO USUÁRIO

CURITIBA PR

2023

MATHEUS YUKIO LOPES SHIMAZU

UM SISTEMA PARA TREINAMENTO NA IDENTIFICAÇÃO E RESOLUÇÃO DE ERROS SINTÁTICOS EM PROGRAMAS C PRÉ-COMPILAÇÃO E COMO A GAMIFICAÇÃO AFETA O DESEMPENHO DO USUÁRIO

Trabalho apresentado como requisito parcial à conclusão do Curso de Bacharelado em Ciência da Computação, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Prof. Andrey Pimentel.

CURITIBA PR

2023

Aos meus pais que me criaram e forneceram condições para eu estudar, conseguir entrar numa faculdade e concluir minha graduação.

AGRADECIMENTOS

Agradeço os meus professores que me ensinaram o que utilizei para chegar até aqui, meus colegas que me ajudaram e deram sugestões sobre este trabalho e finalmente agradeço os voluntários que aceitaram meu pedido de testagem e avaliação do sistema proposto.

RESUMO

A tarefa de *debugging* é algo recorrente e frustrante para alunos de programação. No aprendizado de programação, dentre os diferentes tipos de erros, os de sintáxe são negligenciados por serem mais simples e rápidos de resolver, com a justificativa de que só é necessário as mensagens de erro do compilador. Este artigo propõe um objeto educacional para auxiliar o aprendizado de programação na linguagem C e treinar programadores a identificar erros sintáticos mais rapidamente, além de avaliar o engajamento gerado por gamificação. Com os dados obtidos por uma avaliação de sentimentos em relação ao sistema, este trabalho chega a resultados positivos em relação a utilidade do objeto de aprendizagem proposto e a influência da gamificação no desempenho. Com isso, este trabalho conclui que a ferramenta proposta seria útil no aprendizado de programação, além de ser fácil e engajante de usar.

Palavras-chave: Gamificação. Programação em C. Erros de Sintaxe. Debugging. Ensino de Programação.

ABSTRACT

The task of debugging is something recurrent and frustrating for programming students. In programming education, between the different types of errors, the syntax ones are neglected because they are simpler and faster to solve, with the justification that only compiler error messages are necessary. This article proposes an educational object to help learning programming in the C language and train programmers to identify syntactic errors more quickly, in addition to evaluating the engagement generated by gamification. With the data obtained by an evaluation of feelings towards the system, this work reaches positive results regarding the usefulness of the proposed learning object and the influence of gamification on performance. With this, this work concludes that the proposed tool would be useful in programming learning, in addition to being easy and engaging to use.

Keywords: Gamification. Programming in C. Syntax Errors. Debugging. Programming Education.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Programar é algo difícil de se aprender(Robins et al., 2003). Os erros possíveis são inúmeros e podem ser tão simples como a falta de um caractere e tão complexos e exóticos que programadores experientes gastam horas para resolver.

A linguagem C é relativamente de "baixo nível", permitindo operações questionáveis assumindo que o programador sabe o que está fazendo, o que propicia erros mais que linguagens de "alto nível". Uma análise superficial de uma apostila de C(Hara e Zola, 2008) mostra dezenas de possíveis erros.

A ideia deste projeto surgiu quando o autor viu e analisou o jogo *Papers, Please*, onde o jogador é um inspetor de fronteira e deve inspecionar passaportes por irregularidades e permitir ou não a entrada ao país. A cada dia regras novas são adicionadas dificultando o seu trabalho e no fim de cada fase o jogador recebe dinheiro usado para despesas e melhorias no jogo.

Esta avaliação de Silvador (2014) resume bem o espírito do jogo:

> "It's hard at first because you're not accustomed to it. There is SO MUCH information to go through and compare, and it takes some time to find your own niche and get used to what you have to check, what papers to check first, and all that. As you keep going you'll find that you slowly get faster and more efficient, and you'll remember certain details like city names and stamps, which in turn will speed things up as well. Practice and perseverence is key.
>
> You may also start to notice minor details that help, such as the fact you don't have to wait for the stamp to rise up before returning the passport. As soon as it's down you can slide it away. Another tip is that when you interrogate someone about something, like an expired date, there's nothing they can say or do to change it so while they are responding you can be stamping denied.".

Percebeu-se então as similaridades entre achar irregularidades em passaportes e achar erros sintáticos de programas, ambos são difíceis no começo, mas com experiência a tarefa fica mais fácil e rápida, com familiaridade vem truques e atalhos que facilitam o processo.

Para comparação, veja o que Lobo (2023) diz sobre programação:

> "Muitos principiantes em programação ficam frustrados quando não conseguem detectar os bugs. Com a experiência, vão ver que começam a detectá-los com rapidez. O importante é não entrarem em pânico".

Com isso dá para ver que é possível usar um jogo para treinar pessoas com pouca familiaridade em uma tarefa de forma repetitiva mas engajante, rápida e prática.

1.1 CARACTERIZAÇÃO DO PROBLEMA

É comum a presença de erros em códigos de programas, dentre eles os mais simples de resolver mas também os mais recorrentes entre programadores iniciantes são erros sintáticos.

O tempo gasto em *debugging* é custoso e erros sintáticos, embora simples e rápidos de resolver, são tão comuns que somados levam um tempo equivalente a um erro mais complexo e demorado. Por serem mais fáceis de resolver, erros sintáticos tendem a ser negligenciados por métodos de ensino em favor de estruturas, lógicas e erros considerados mais críticos e importantes.

Os métodos comuns utilizados na resolução de erros sintáticos são mensagens de erros de compilador e experiência, ambos os quais um programador experiente compreende e usa, mas que para iniciantes causa frustração por não entender as mensagens e faltar a dita experiência.

1.2 OBJETIVOS GERAIS

O objetivo deste trabalho é criar um sistema que auxilie programadores, iniciantes ou veteranos, da linguagem C a melhorar a eficácia e a eficiência na identificação e correção de erros sintáticos de código com o uso de elementos de gamificação para gerar engajamento, assim como verificar a influência destes elementos nos resultados obtidos.

1.3 OBJETIVOS ESPECÍFICOS

- Investigar erros sintáticos recorrentes na programação;
- Investigar trabalhos relacionados na identificação e correção de erros sintáticos;
- Escolher tipos de erros sintáticos comuns para serem implementados no sistema;
- Elaboração de programas simples assim como as versões com erros sintáticos dos tipos escolhidos;
- Definição de elementos de gamificação que possam aumentar engajamento no sistema;
- Criação do sistema de treinamento que utiliza os programas elaborados e mecanismos definidos;
- Implementar um servidor que permita armazenar dados de desempenho dos usuários para análise futura;
- Criação de um questionário para avaliação do sistema;
- Testar do sistema com alunos de computação;
- Validar o sistema de treinamento proposto;
- Validar os elementos de gamificação utilizados no sistema;
- Validar escolhas visuais e espaciais implentadas no sistema.

1.4 JUSTIFICATIVA

Trabalhos relacionados desta área sugerem três abordagens comuns: uma em que o usuário recebe um código errado e o erro de compilação correspondente e deve corrigir tal erro; outra em que o usuário deve receber um código errado e fazê-lo funcionar; e outra em que deve-se focar em melhorar as mensagens de erro mostradas pelo compilador e isso melhorará o desempenho do programador.

Entretanto essas abordagens têm como prerrogativa que o código foi ou será compilado antes do final do exercício. Para diferenciar destas abordagens, este trabalho busca treinar depurar códigos **antes** da compilação, não dando mensagens de compilação, misturando códigos sem erros e nem mesmo indicando se estão errados ou não.

Espera-se que o uso de exercícios simples e tipos de erros limitados simplifique o processo, aumentando o foco e rapidez de resolução, de modo que o usuário aprenda por si mesmo ao longo dos exercícios a ser eficiente para encontrar os erros. A inclusão de gamificação deve aumentar engajamento e incentivar o usuário a acertar o máximo possível em um menor tempo.

1.5 ORGANIZAÇÃO DO TRABALHO

Este trabalho possui a seguinte organização: este capítulo contém uma introdução geral ao conteúdo do documento, com uma caracterização do problema, objetivos gerais e específicos e justificativa. O próximo capítulo(2) faz uma breve descrição de conceitos básicos necessários para o entendimento deste assunto.

O capítulo 3 apresenta trabalhos relacionados a este como outros artigos que apresentam jogos educacionais e métodos de ensino a programação. O quarto capítulo descreve o sistema proposto. O capítulo 5 traz a validação do sistema implementado. Por último está o capítulo 6 onde a conclusão e trabalhos futuros se encontram.

2 CONCEITOS FUNDAMENTAIS

Neste capítulo serão apresentados conceitos fundamentais que serão utilizados no decorrer deste trabalho.

2.1 INFORMÁTICA NA EDUCAÇÃO

Desde 1979 a informática vem sendo usada na área de educação no Brasil(Tajra, 2011). O termo Informática na Educação se refere ao uso de recursos computacionais não só no ambiente escolar, mas integrado com atividades educacionais e sociais.

Esta área abrange o uso de software genéricos como aplicativos de comunicação e armazenamento de dados para acessar recursos educacionais como livros em PDF, até software específicos que têm como principal propósito ensinar algo, por exemplo o aplicativo Duolingo, que ensina idiomas. Também nesta área estão inclusos software que auxiliam indiretamente a educação, como sistemas organizacionais que ajudam na burocracia, portais de alunos, etc.

2.2 SISTEMAS TUTORES INTELIGENTES(STI)

STIs são sistemas que dão suporte a aprendizagem, fazendo o papel de um tutor para o aluno e adaptando o conteúdo e estratégias utilizadas de acordo com o desempenho do usuário(Gavidia e ANDRADE, 2003).

Um STI tradicional possui uma arquitetura com 4 módulos:

- Modelo do Aluno: onde as características individuais do aluno são modeladas e guardadas, isso inclui o conhecimento já conhecido, exercícios resolvidos, erros feitos anteriormente, entre outros.
- Modelo do Domínio: também conchecido como Modelo do Conhecimento do Especialista, ele armazena o conhecimento e suas relações, assim como os exercícios e respostas.
- Modelo do Tutor: é responsável por mediar a verificação de respostas do Módulo do Domínio com as informações do Modelo do Aluno para decidir qual deve e se deve ocorrer intervenções tutoriais, sendo isto dar dicas, orientações ou escolher qual será o próximo exercício.
- Modelo da Interface: permite a interação entre o aluno e o sistema.

2.3 COMPILADOR DE C E MENSAGENS DE ERROS

O compilador de C mais popular é o GCC(GNU Compiler Collection), sendo um pacote de Software Livre ligado ao projeto GNU. O GCC foi criado como um compilador para a linguagem C em 1985 por Richard Stallman, mas atualmente oferece suporte para outras linguagens como C++, Objective-C, Fortran e Java (de Oliveira et al., 2007).

Um compilador recebe um código em texto e o traduz para um programa binário executável, dando avisos ou alertas de erros caso tenha encontrado algum erro no código. O GCC é bem catalogado e apresenta mensagens para praticamente todos tipos de erros de compilação.

Entretanto, uma série de estudos (Gomes et al. (2015), Gomes e Amaral (2016), Gomes e do Amaral (2016)) mostra que tais mensagens de erros são muitas vezes complexas, não significativas e causam confusão para programadores iniciantes, além de ser em Inglês, o que dificulta ainda mais a identificação e correção do erro.

Há uma linha de pesquisa que busca melhorar as mensagens de erro geradas por compiladores a fim de melhorar e facilitar o aprendizado de programação.

2.4 DEBUGGING

Todo programador já fez algum programa com erros, portanto *debugging*(depuração), que é encontrar e corrigir *bugs* ou erros de código, é algo que todos programadores já fizeram e é essencial para construção de programas e também para o próprio aprendizado de programação (Lowe, 2019; Bofferding et al., 2022; Whalley et al., 2023; McCauley et al., 2008; Li et al., 2022).

Existem 3 categorias principais de erros:

- Erros de compilação, que são identificados pelo compilador e incluem erros sintáticos e semânticos.
- Erros em tempo de execução, são erros que ocorrem por algo crítico que interrompe a execução do programa já compilado, como uma divisão por zero ou acesso proibido de memória.
- Erros lógicos, que ocorrem quando o código está correto e funcionando, mas está fazendo algo diferente do que esperado. Um exemplo é utilizar milisegundos pensando que está em segundos, o código irá funcionar, mas o resultado será mil vezes maior.

O processo de Debugging pode ser resumido na figura 2.1 e consiste em ciclos de compilações, execuções e análises dos resultados, corrigindo quaisquer erros encontrados em cada parte do processo e reiniciando o ciclo para verificar se a correção funcionou ou não.

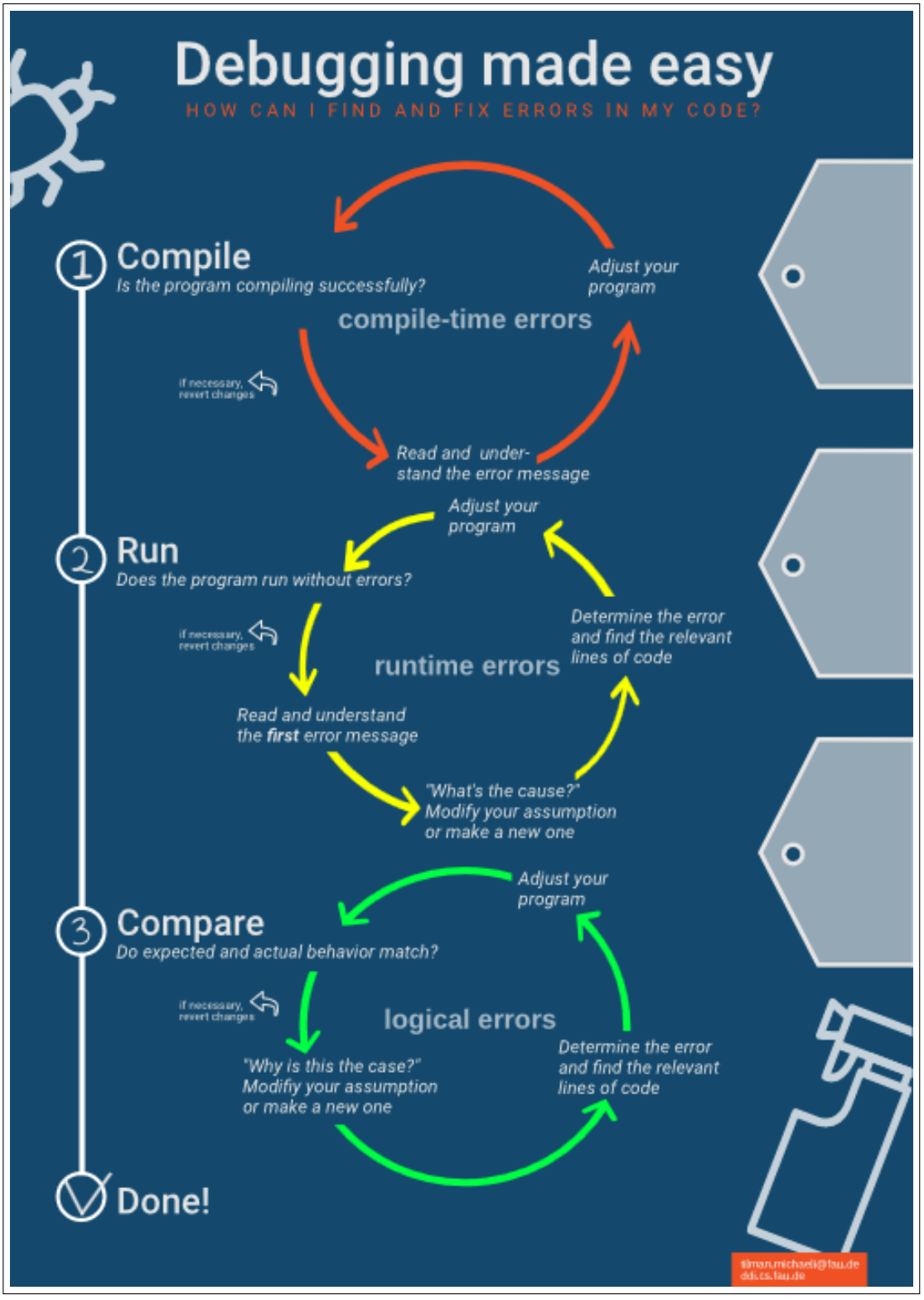


Figura 2.1: Ciclo de Debugging(Michaeli e Romeike, 2019)

2.5 ERROS SINTÁTICOS DO C

Dentre os três tipos de erros apresentados na seção anterior, os erros sintáticos são os mais comuns e também os mais fáceis de resolver. A presença deles é apontada pelo compilador e a correção é simplesmente entender a mensagem de erro resultante e fazer a modificação correspondente.

Erros sintáticos podem ocorrer por descuido, usar pedaços de códigos de terceiros sem o devido cuidado, ou falta de familiaridade com a linguagem. Lowe (2019) defende que "*Experience yields automation, which provides programmers with both speed and accuracy, even in simple programming tasks*". Isto é, com experiência um programador consegue escrever códigos rapidamente e com menos erros.

Uma pesquisa de Gomes et al. (2015) mostra os erros e *warnings* mais comuns entre iniciantes nas figuras 2.2 e 2.3.

Figura 2.2: Erros mais comuns(Gomes et al., 2015)

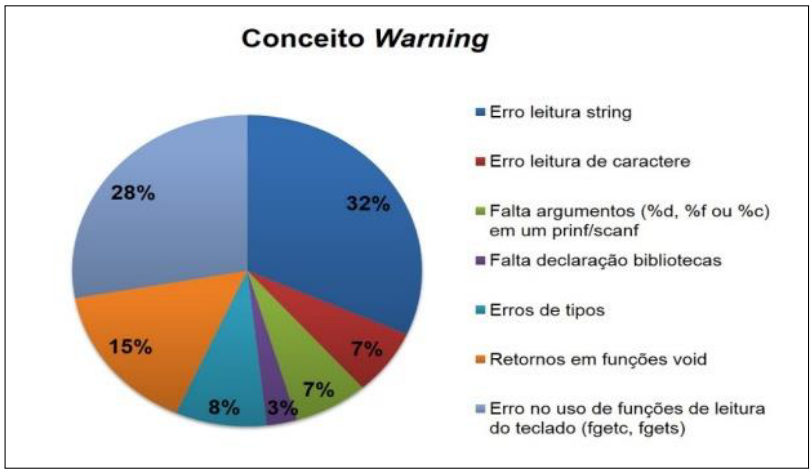


Figura 2.3: Warnings mais comuns(Gomes et al., 2015)

2.6 GAMIFICAÇÃO

A gamificação consiste em introduzir elementos de jogos em contextos fora de entretenimento a fim de gerar engajamento. Tais contextos podem ser bem variados, com os mais comuns sendo em Negócios, Saúde, Esportes e Educação(Costa e Marchiori, 2015).

Os elementos de gamificação podem ser separados em três tipos: dinâmicas de jogo, mecânicas de jogo e componentes. Esses tipos estão em ordem decrescente de abstração, com dinâmicas sendo temas e aspectos gerais do jogo como na tabela 2.1, mecânicas sendo fatores que orientam e delimitam as ações do jogador, formando o funcionamento do jogo como descrito na tabela 2.2, enquanto componentes são elementos específicos mais concretos e visíveis na interface do jogo como resumido na tabela 2.3 (Costa e Marchiori, 2015).

Os diferentes tipos de elementos se relacionam de modo que cada componente tem um papel em uma ou mais mecânicas e cada mecânica influencia uma ou mais dinâmicas. Por exemplo, o componente Pontos pode ser usado como condição das mecânicas Desafios, Recompensas e Vitória, que por sua vez trazem dinâmicas como Emoções e Progressão.

Tabela 2.1: Dinâmicas de Jogo – Conceituações(Costa e Marchiori, 2015)

Dinâmicas	Descrição
`Emoções`	Jogos podem criar diferentes tipos de emoções, especialmente a da diversão (reforço emocional que mantém as pessoas jogando)
`Narrativa`	Estrutura que torna o jogo coerente. A narrativa não tem que ser explícita, como uma história em um jogo. Também pode ser implícita, na qual toda a experiência tem um propósito em si
`Progressão`	Ideia de dar aos jogadores a sensação de avançar dentro do jogo
`Relacionamentos`	Refere-se à interação entre os jogadores, seja entre amigos, companheiros ou adversários
`Restrições`	Refere-se à limitação da liberdade dos jogadores dentro do jogo

2.7 CONCLUSÃO

Neste capítulo foram apresentados e brevemente explicados conceitos considerados relevantes e fundamentais para o entendimento deste trabalho como informática na educação, STIs, debugging, compiladores e mensagens de erro, erros sintáticos de C e gamificação.

Com isso espera-se que o leitor entenda os princípios deste projeto e não precise de informações exteriores para compreender os trabalhos relacionados e a proposta que vem a seguir.

Tabela 2.2: Mecânicas de Jogo – Conceituações(Costa e Marchiori, 2015)

Mecânicas	Descrição
Aquisição de recursos	O jogador pode coletar itens que o ajudam a atingir os objetivos
Avaliação (*Feedback*)	A avaliação permite que os jogadores vejam como estão progredindo no jogo
Chance	Os resultados de ação do jogador são aleatórios para criar uma sensação de surpresa e incerteza
Cooperação e competição	Cria-se um sentimento de vitória e derrota
Desafios	Os objetivos que o jogo define para o jogador
Recompensas	O benefício que o jogador pode ganhar a partir de uma conquista no jogo
Transações	Significa compra, venda ou troca de algo com outros jogadores no jogo
Turnos	Cada jogador no jogo tem seu próprio tempo e oportunidade para jogar. Jogos tradicionais, como jogos de cartas e jogos de tabuleiro muitas vezes dependem de turnos para manter o equilíbrio no jogo, enquanto muitos jogos de computador modernos trabalham em tempo real
Vitória	O “estado” que define ganhar o jogo

Tabela 2.3: Componentes de Jogo – Conceituações(Costa e Marchiori, 2015)

Componentes	Descrição
Avatar	Representação visual do personagem do jogador
Bens virtuais	Itens dentro do jogo que os jogadores podem coletar e usar de forma virtual e não real, mas que ainda tem valor para o jogador. Os jogadores podem pagar pelos itens ou moeda do jogo ou com dinheiro real
Boss	Um desafio geralmente difícil no final de um nível que tem deve ser derrotado, a fim de avançar no jogo
Coleções	Formadas por itens acumulados dentro do jogo. Emblemas e Medalhas são frequentemente parte de coleções
Combate	Disputa que ocorre para que o jogador derrote oponentes em uma luta
Conquistas	Recompensa que o jogador recebe por fazer um conjunto de atividades específicas
Conteúdos desbloqueáveis	A possibilidade de desbloquear e acessar certos conteúdos no jogo se os pré-requisitos forem preenchidos. O jogador precisa fazer algo específico para ser capaz de desbloquear o conteúdo
Emblemas/ medalhas	Representação visual de realizações dentro do jogo
Gráfico Social	Capacidade de ver amigos que também estão no jogo e ser capaz de interagir com eles. Um gráfico social torna o jogo uma extensão de sua experiência de rede social.
Missão	Similar a “conquistas”. É uma noção de jogo de que o jogador deve fazer executar algumas atividades que são especificamente definidas dentro da estrutura do jogo
Níveis	Representação numérica da evolução do jogador. O nível do jogador aumenta à medida que o jogador se torna melhor no jogo.
Pontos	Ações no jogo que atribuem pontos. São muitas vezes ligadas a níveis
Presentes	A possibilidade distribuir ao jogador coisas como itens ou moeda virtual para outros jogadores
Ranking	Lista jogadores que apresentam as maiores pontuações/conquistas/itens em um jogo
Times	Possibilidade de jogar com outras pessoas com mesmo objetivo

3 TRABALHOS RELACIONADOS

Neste capítulo, serão analisados alguns trabalhos considerados relacionados a área de pesquisa deste projeto. Dentre estes trabalhos estão jogos educacionais para ensino de programação, métodos de ensino a programação e artigos sobre *debugging*.

3.1 ANALISADOR CFACIL

É apresentado e incrementado em vários artigos (Gomes et al., 2015; Gomes e Amaral, 2016; Gomes e do Amaral, 2016). O software contém analisadores sintático, léxico e semântico que são utilizados para gerar mensagens de erro do mesmo modo que um compilador como o GCC faria, só que busca mostrar mensagens em Português mais significativas e claras (Figura 3.1).

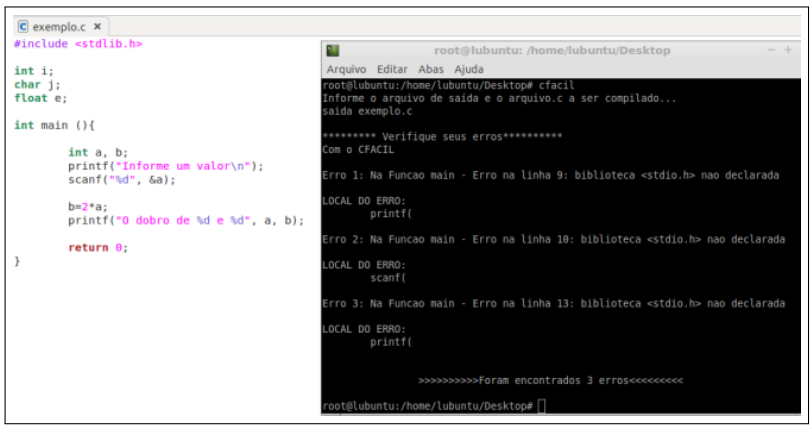


Figura 3.1: Exemplo CFacil(Gomes e Amaral, 2016)

Vale a pena mencionar as pesquisas preliminares(Gomes et al., 2015), que contém os erros sintáticos mais comuns em C, assim como outros dados interessantes. Este trabalho teve resultados positivos em relação a melhoria de desempenho dos alunos.

3.2 GIDGET: AN ONLINE DEBUGGING-FIRST GAME

Este software também é foco de diversos artigos (Lee e Ko, 2011; Lee, 2012, 2014; Lee e Ko, 2014; Lee et al., 2014; Lee e Ko, 2015; Liu et al., 2017), sendo este último por um autor diferente do criador.

É um jogo que ensina programação para iniciantes com um método *debugging-first*, isto é, ele encoraja o aprendizado por meio de debugação de programas já existentes antes de criar novos programas (Figura 3.2). Os resultados foram positivos, sendo o jogo capaz de ensinar novatos programação com conceitos básicos como condicionais e loops em 5 horas.

3.3 DEBUGGING WITH STACK OVERFLOW

O artigo de Li et al. (2022) analisa o comportamento de iniciantes e experts quando depuram usando o site Stack Overflow. Dentre diversos dados e conclusões destaca-se que tanto

Figura 3.2: Exemplo Gidget(Lee et al., 2014)

a experiência em programação geral como a específica da linguagem afetam significativamente o desempenho na busca de links úteis e no sucesso da identificação e correção de erros.

3.4 ENHANCING SYNTAX ERROR MESSAGES APPEARS INEFFECTUAL

Este artigo(Denny et al., 2014) é curioso pois, comparado a maioria dos artigos que visam melhorar as mensagens de erro dos compiladores e conseguem poucos resultados(Kummerfeld e Kay, 2003; Becker et al., 2018), ele não mostra melhoria significativa no desempenho dos alunos que utilizaram o sistema.

3.5 CONCLUSÃO

Com isso espera-se que o leitor tenha noção de trabalhos similares ou relacionados e obtenha contexto para entender e compará-los com a proposta de projeto a seguir.

4 PROPOSTA

O sistema proposto é uma ferramenta web de treinamento para a identificação e resolução de erros sintáticos em C **antes** da compilação. O software usa elementos de gamificação como pontuação limites de tempo para gerar engajamento enquanto mantém exercícios simples para obter uma interação rápida e frenética. No final de cada sessão é possível ver seu desempenho em estatísticas como quantidade de acertos e tempo médio por exercício.

4.1 MÓDULOS

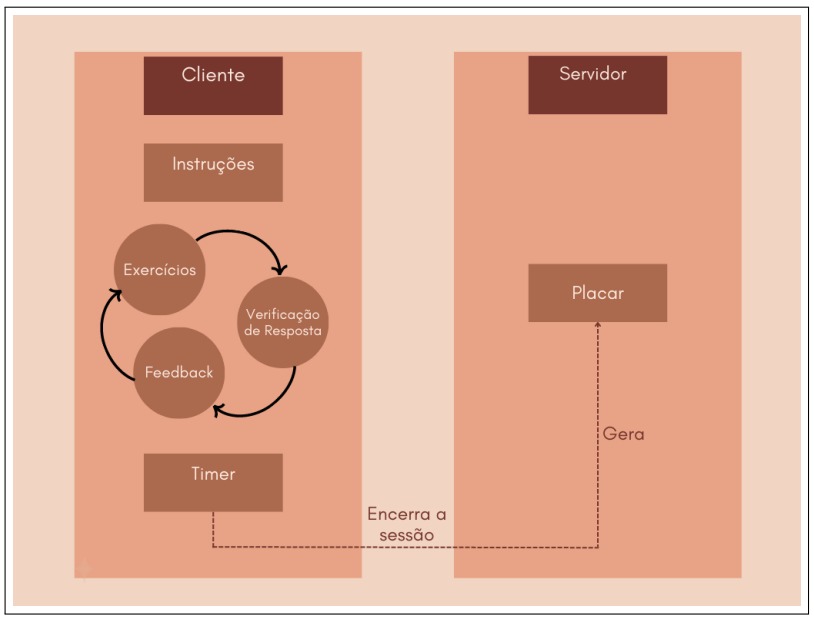


Figura 4.1: Arquitetura do sistema

O sistema pode ser dividido em 6 módulos: o de instruções, timer, feedback, exercícios, verificação de resposta e placar. Estes módulos podem ser separados nos lados Cliente e Servidor, como indica a figura 4.1. Cada módulo será explicado nas subseções a seguir.

4.1.1 Instruções

Este módulo explica as regras de jogo e dá exemplos de cada erro possível dentre os exercícios. Ele pode ser visto no anexo A.

4.1.2 Timer

Na parte superior da tela há um timer de três minutos que mostra tanto uma barra de progresso(que enche de verde da esquerda para a direita conforme passa o tempo) quanto o tempo restante em números no formato [minutos:segundos].

Quando este contador chega em zero todos dados são processados, resumidos e então os dados relevantes que restam são enviados para o servidor com um redirecionamento de página usando GET. O servidor então irá processar e armazenar os dados e devolver o Placar.

4.1.3 Módulo de feedback

Este módulo fica abaixo da barra de progresso(timer) e apresenta a pontuação, a contagem de acertos/erros consecutivos e mensagens significativas em relação às respostas do usuário. Dentro dele está a contagem de acertos e erros consecutivos no formato “NX feedback”, sendo N o número de erros ou acertos consecutivos e o feedback sendo informações relevantes como se o usuário acertou ou errou o exercício e o lugar e tipo do exercício.

Para destacar a situação e melhorar o entendimento do usuário, os avisos são coloridos em verde(para acertos) e vermelho(para erros), além de apresentarem os últimos três avisos em um tamanho cada vez menor e mais desbotado, dando uma ideia de fluidez, sequência e progresso. Também presente neste módulo está a pontuação, que para cada acerto aumenta em (10 multiplicado por N número de acertos consecutivos). De forma similar um erro diminui a pontuação, com o detalhe de que a pontuação nunca ficará menor que 0.

4.1.4 Exercícios

Na parte direita da tela fica o lugar dos exercícios, cada erro ou acerto muda o exercício aleatoriamente, alterando a imagem mostrada. Internamente cada exercício tem um código e cada código tem a resposta correspondente guardada.

Durante a construção do sistema notou-se que a mudança de exercícios era um tanto abrupta e por ser aleatória e existir exercícios diferentes mas parecidos, acontecia de o usuário não perceber que mudou o exercício. Para resolver isso foi implementado um efeito de *fade-out* para suavizar as transições.

4.1.5 Verificação de resposta

Este módulo fica na parte inferior esquerda da tela e consiste em duas perguntas:

A primeira questiona o usuário se o programa apresentado está certo ou não e verifica se o usuário acertou esta resposta. Caso tenha marcado que o programa estava certo independente de ter respondido corretamente, ou marcado errado apesar do programa estar certo, o exercício é trocado e o feedback e a pontuação são atualizados. Caso tenha respondido que o programa estava errado e acertado, será apresentado a segunda pergunta.

A segunda pergunta questiona em qual linha do código está o erro e qual o tipo do erro, qualquer resposta será verificada e resultará no feedback apropriado e na troca de exercício.

4.1.6 Placar

O servidor prepara a tela de placar baseado nos dados enviados no término do exercício, mostrando as seguintes estatísticas: número de acertos, erros e total de programas, maior sequência de acertos e de erros, menor e maior tempo levado em um exercício e a média dos tempos levados por exercício. Também mostra a pontuação do jogador, um link para jogar novamente e outro para responder um questionário de avaliação.

4.2 REGRAS DO JOGO

O jogador tem 3 minutos para acertar o máximo de exercícios que conseguir. Para cada exercício apresentado o jogador deve analisar o código a procura de erros. Caso o código esteja correto, o jogador deve marcar “Sim” para a pergunta “O programa está certo?”, o que pulará para um novo exercício. Caso esteja incorreto, o jogador deve marcar “Não”, e então deverá

responder a linha e qual o tipo do erro encontrado e clicar em "Verificar"para ir para o próximo exercício.

Existem 8 erros implementados nos exercícios:

1. Faltou ‘;’ no final da linha: Há alguma linha de comando sem ‘;’ no final. A linha correspondente é a mesma que falta o ‘;’ ;

2. Faltou fechar parênteses ou chaves: Foi aberto parênteses ou chaves - ‘(‘ ou ‘{‘- que não foram fechadas-’)’ ou ‘}’. A linha correspondente deve ser indicada como a mesma linha em que foi aberto os parênteses ou chaves;

3. Variável não foi declarada: Foi usado uma variável que não foi declarada antes. A linha correspondente é a primeira linha que tal variável for usada;

4. Tipo incompatível da variável: A variável foi atribuída um valor ou comparada com um tipo diferente do declarado inicialmente. A linha do erro é a primeira linha onde foi feita a atribuição ou comparação;

5. Variável com valor nulo: Foi usado uma variável que não foi atribuída valor antes. A linha correspondente é a primeira linha que o valor da variável for utilizado;

6. Atribuição de vetores: Foi utilizado equivocadamente o comando de atribuição ‘=’ para atribuir um vetor inteiro a outro vetor ao invés de usar a função ‘strcpy()’ ou elemento por elemento. A linha do erro é onde for usado tal comando;

7. Faltou biblioteca: Foi utilizado uma função que pertence a uma biblioteca mas tal biblioteca não foi incluída. A linha correspondente deve ser a linha em que a função foi usada pela primeira vez;

8. Passou dos limites do vetor: Foi acessado um endereço do vetor além dos limites dele(de 0 a TAM-1, com TAM sendo o tamanho do vetor). A linha do erro deve ser onde ocorreu o acesso proibido.

Acertar um exercício indicará tal corretude e incrementará a pontuação em 10 multiplicado pelo número de acertos consecutivos. De forma similar errar indicará o erro e qual deveria ser a resposta correta e subtrai a pontuação em 10 vezes o número de erros consecutivos, mas a pontuação nunca ficará abaixo de 0. Independente de acerto ou erro o exercício será mudado de forma aleatória. Ao fim do tempo o jogo acaba imediatamente e o placar é apresentado.

4.3 CARACTERÍSTICAS EDUCACIONAIS

Bugs são comuns em qualquer código, de profissionais até iniciantes, mas para novatos a tarefa de encontrá-los e corrigi-los pode ser frustrante e demorada.

O projeto visa treinar o usuário nesta tarefa por meio de repetições ao fornecer diversos programas simples com erros comuns em um período de tempo limitado e dando feedback em relação à resposta dada pelo usuário e a resposta correta sobre a presença de bugs no código e a categoria do erro.

Com a repetição e feedback vem a experiência e rapidez em achar bugs em programas. A limitação do número de tipos de bugs reduz o número de fatores que o usuário precisa levar em conta, facilitando a tarefa e focando o aprendizado e treinamento. Elementos de gamificação servem como incentivo para o melhorar o desempenho:

- O timer e barra de progresso dão uma sensação de urgência, o que incita o usuário a ser mais rápido em cada exercício para fazer mais exercícios dentro do tempo, aumentando sua *eficiência*.
- A contagem de pontuação e acertos/erros consecutivos dão uma mensuração do quão bem o usuário está jogando e incentiva um foco maior para acertar mais e errar menos, melhorando a *eficácia*.

4.4 IMPLEMENTAÇÃO

4.4.1 Linguagens usadas

HTML e JavaScript(com JQuery e math.js) para o site, Python3 para o servidor e .txt para guardar informações genéricas sobre desempenho de jogadores.

4.4.2 Estruturas de dados por módulo

4.4.2.1 Instruções

A parte de texto é armazenada em DIVs de HTML e as imagens em PNG's.

4.4.2.2 Timer

O tempo é definido em uma variável de JavaScript que é usado numa função *setInterval*, que a cada intervalo de tempo aumenta a barra de progresso e atualiza o contador de tempo restante na tela. Quando a barra de progresso estiver cheia(width=100%) a função *setInterval* para e dispara a função que finaliza o jogo.

4.4.2.3 Módulo de Feedback

A pontuação é armazenada em um . Os consecutivos são guardados em uma variável JavaScript que, quando atualizada, é copiada para um , o feedback é criado conforme necessário e colocado num .

4.4.2.4 Verificação de Resposta

Cada rádio de resposta "Sim"ou "Não"sobre a corretude do código tem valor 0 ou 1, respectivamente, a linha é lida de um campo de texto e os tipos de erros estão numa lista *drop-down* com cada tipo tendo um valor de 1 a 8 que será usado internamente, 0 significa que não há erros, de forma que é possível utilizar o mesmo valor para conferir se está errado e para verificar qual erro.

4.4.2.5 Placar

O exercício atual, número de acertos e erros totais, maior acertos seguidos e erros seguidos, e o menor e maior tempo levado em um exercício são guardados em variáveis JavaScript, além disso existem dois vetores em Javascript para armazenar dinamicamente cada exercício feito, um que guarda apenas os tempos em milissegundos e outro que guarda apenas 1 ou 0 para saber se o usuário acertou ou errou, respectivamente, o exercício.

Quando encerrado o tempo, os vetores são processados no lado do cliente para mandar para o servidor apenas estatísticas como médias e desvios padrões de todos tempos juntos, tempos

dos acertos, tempos dos erros. Além disso, para comparar o desenvolvimento do usuário na mesma sessão também é coletado essas mesmas métricas para primeira metade e segunda metade dos vetores, tornando possível comparar as médias no começo da sessão com as do fim. O servidor então armazena esses dados em um arquivo .txt e mostra um placar com algumas dessas informações resumidas.

O endereço do servidor em si é salvo em uma string do arquivo JavaScript, que deve ser modificado dependendo do endereço IP do computador que o servidor estiver sendo executado. Isso é importante pois o computador usado não é fixo mas sim o que estiver disponível no laboratório de informática.

Outro detalhe importante é que por o servidor ser executado no laboratório de informática do Departamento de Informática da UFPR, onde há diversas proteções e restrições sobre a rede, não foi possível utilizar POST nem PHP no servidor, além de impedir a funcionalidade do Placar caso o usuário tente utilizar o sistema por um lugar fora da mesma rede e deixar o servidor vulnerável a desligamento ou interferência por alunos do laboratório.

4.4.3 Exercícios

Os exercícios foram pré-preparados manualmente e salvos em imagens PNG com nomes X.PNG, X sendo o código do exercício. As respostas(tipo de erro e linha do erro) são armazenadas em HTML dentro de cujo id contém o código do respectivo exercício(ex: "Xerrado"e "Xlinha", com X sendo o mesmo código usado nos PNGs) para facilitar a busca e relação entre a imagem e a resposta.

Apenas o X é salvo para saber qual exercício é o atual e quando alterado apenas troca-se a fonte(src) da imagem no HTML com o X.PNG atualizado. Foram implementados 3 algoritmos simples como o famoso "Hello World", o cálculo de fatorial e a soma de dígitos de um número, assim como variações desses programas com erros, totalizando 14 exercícios diferentes.

4.5 CENÁRIO DE USO

A tela inicial e as instruções podem ser vistas no anexo A. A seguir será apresentado um cenário de uso com telas.

A figura 4.2 mostra a tela inicial ao iniciar o jogo. Note que o tempo é 3:00 e a barra de progresso está completamente branca. O código mostrado aparenta estar certo, então vamos marcar que "Sim", o programa está certo.

A figura 4.3 mostra a tela após marcar "Sim". Note que o tempo passou e a barra e contador mudaram de acordo, o exercício também mudou, os pontos foram atualizados e temos o feedback em verde que acertamos 1 exercício seguido e que de fato o código estava correto.

Desta vez, após analisar o código foi detectado um erro na linha 9, onde se utiliza a variável 'm', que nunca foi declarada. Tendo descoberto isso, marcamos "Não", não está correto o código. A figura 4.4 mostra a tela após marcar "Não". Note que só o tempo mudou, o exercício e o feedback continuam os mesmos. Agora no lugar da pergunta de Sim ou Não aparece a pergunta "Qual a linha e tipo do erro?"com um lugar para digitar o número da linha e um menu *drop-down* com diferentes tipos de erros. Na figura 4.5 aparece a resposta preenchida.

Após clicar em Verificar ou apertar "Enter", iremos para a figura 4.6, onde vemos que acertamos o erro e o lugar, ganhamos pontos, aumentamos nosso número de acertos seguidos e temos um novo exercício para fazer. Neste exercício há um erro na linha 7, onde é usado o 'i', que não foi declarado. Mas desta vez, por descuido marcamos "Sim", o que resulta na figura 4.7.

Como erramos, perdemos pontos e nossa sequência de acertos. Podemos ver o feedback em vermelho que mostra qual deveria ser a resposta correta, também podemos ver as mensagens

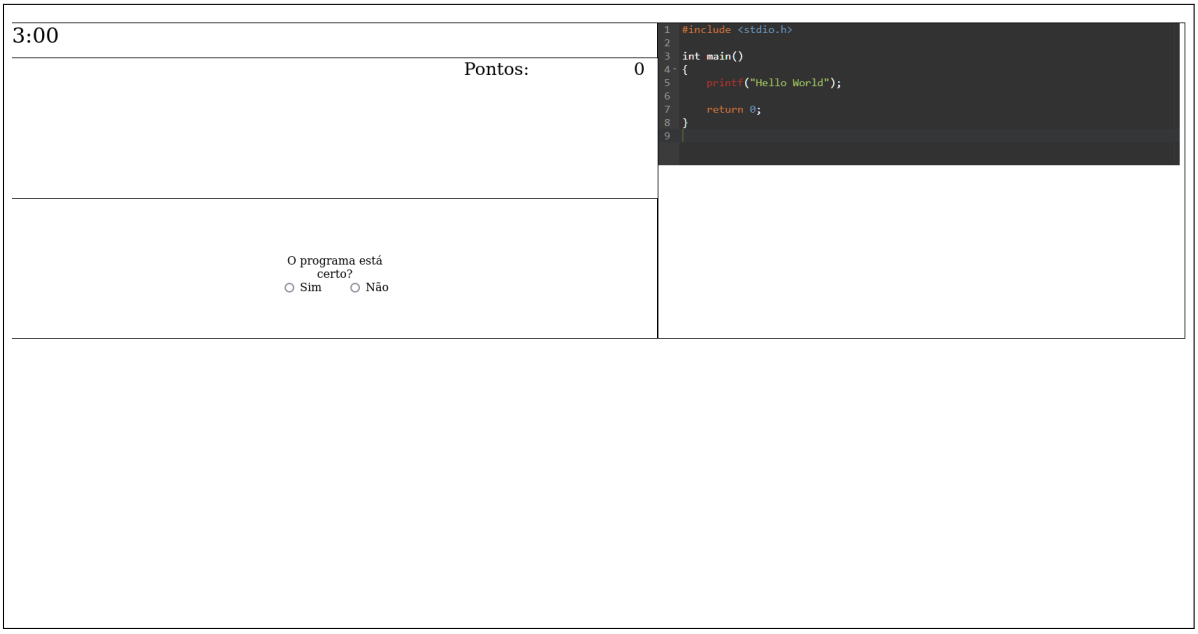


Figura 4.2: Tela 1

Figura 4.3: Tela 2

anteriores em tamanho decrescente com cor mais transparente, dando uma ideia de progresso, que estamos na mensagem maior e mais recente. De qualquer modo o exercício mudou.

Avançando alguns segundos adiante e alguns exercícios depois nos deparamos com a figura 4.8. Desta vez, talvez por costume e sem ter certeza da resposta, marcamos que está errado, mas infelizmente o código estava correto. Com isso temos uma mensagem de erro diferente como vemos na figura 4.9.

Avançando mais um pouco temos a figura 4.10, onde falta apenas 1 segundo para acabar esta sessão do jogo. Quando tal segundo passa, o usuário é encaminhado para uma página criada pelo servidor que contém o Placar como mostra a figura 4.11, encerrando este cenário de uso.

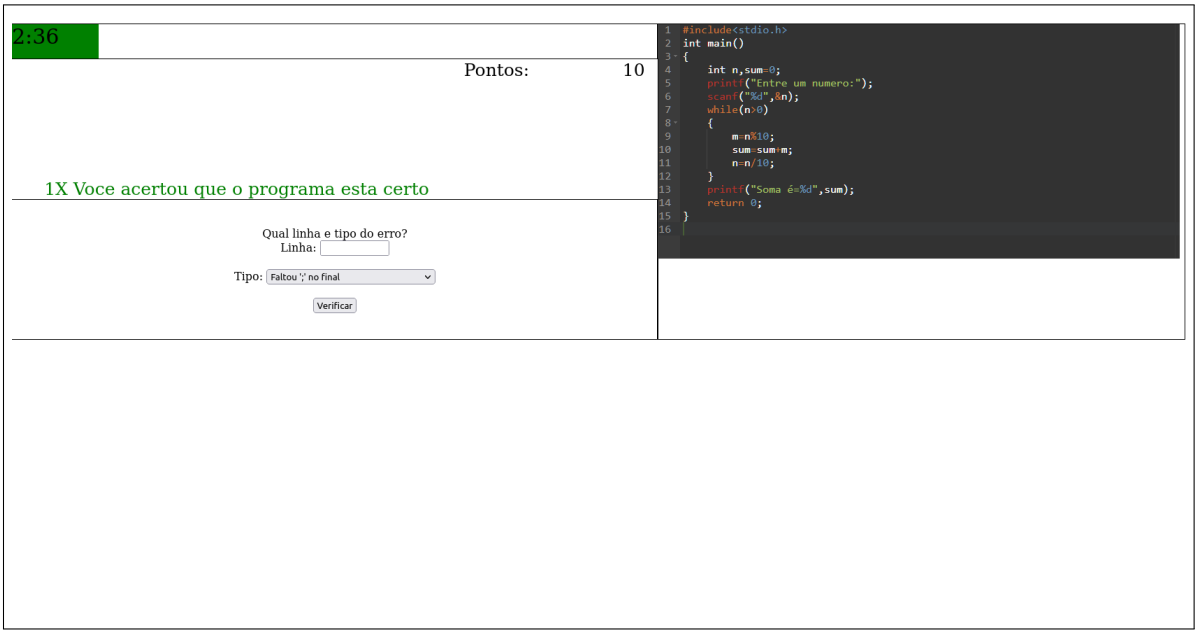


Figura 4.4: Tela 3

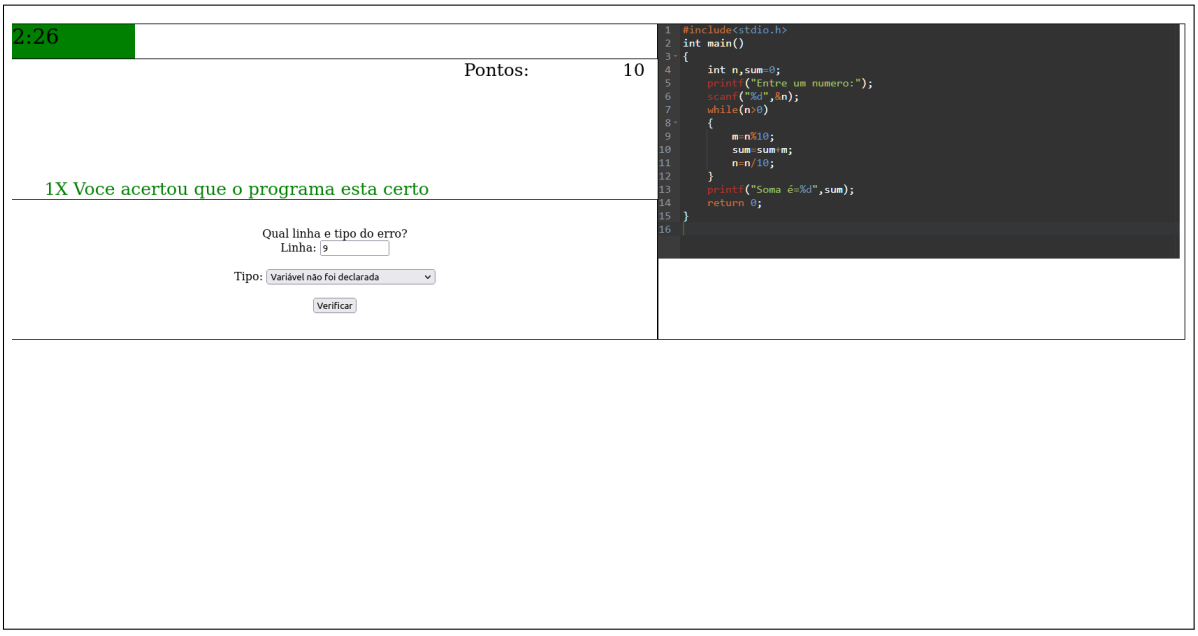


Figura 4.5: Tela 4

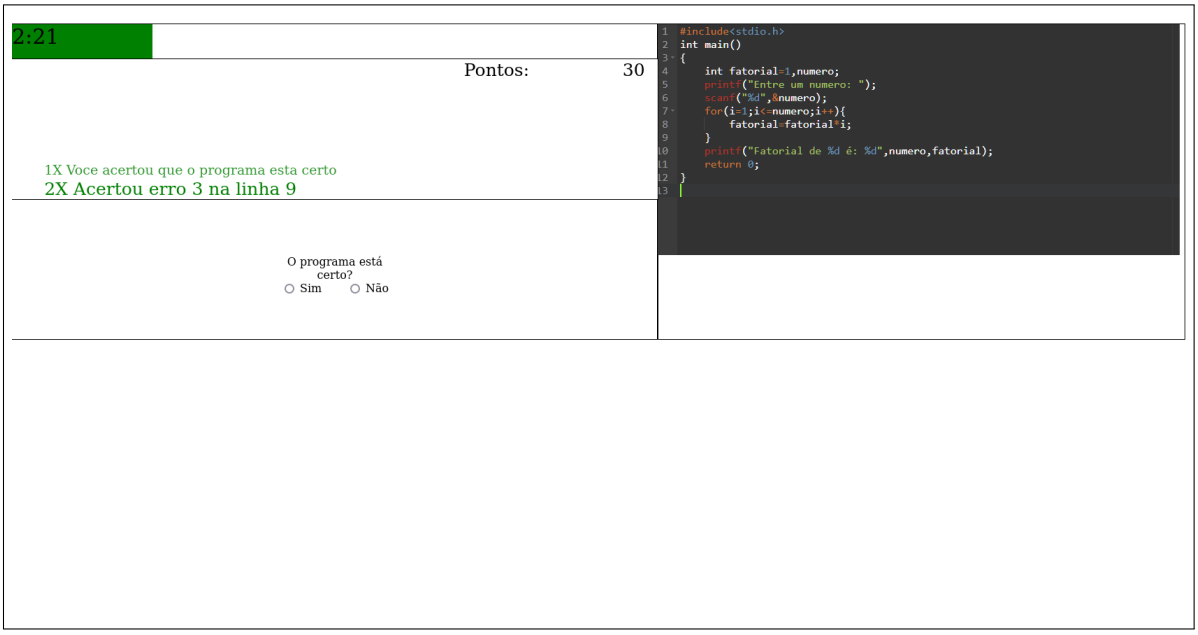


Figura 4.6: Tela 5

Figura 4.7: Tela 6

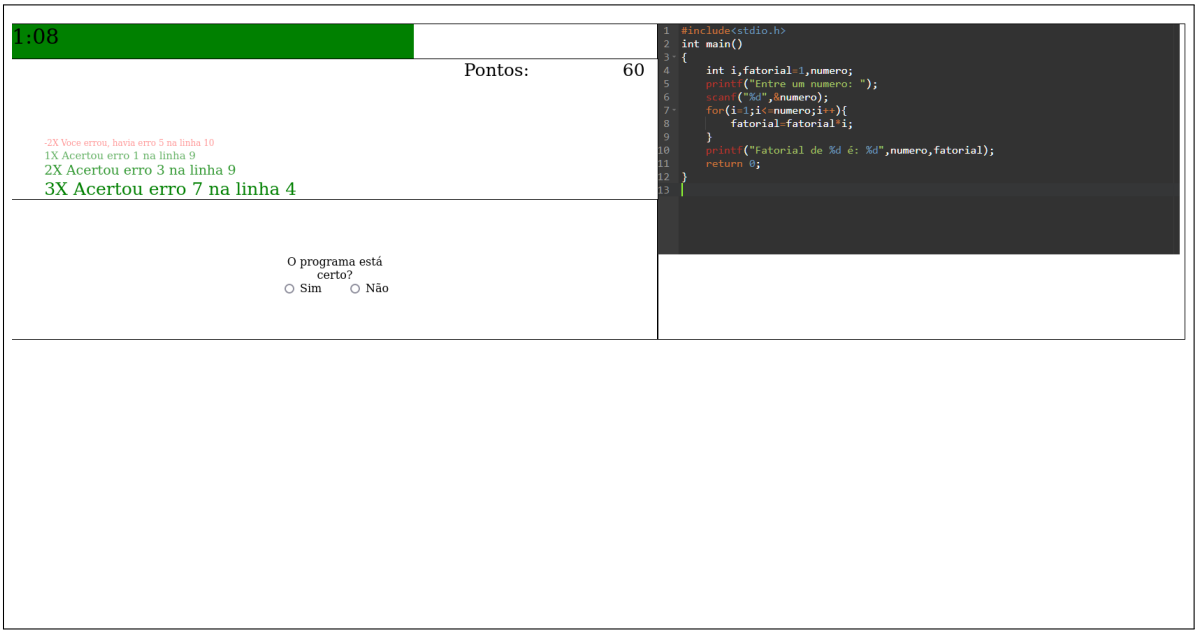


Figura 4.8: Tela 7

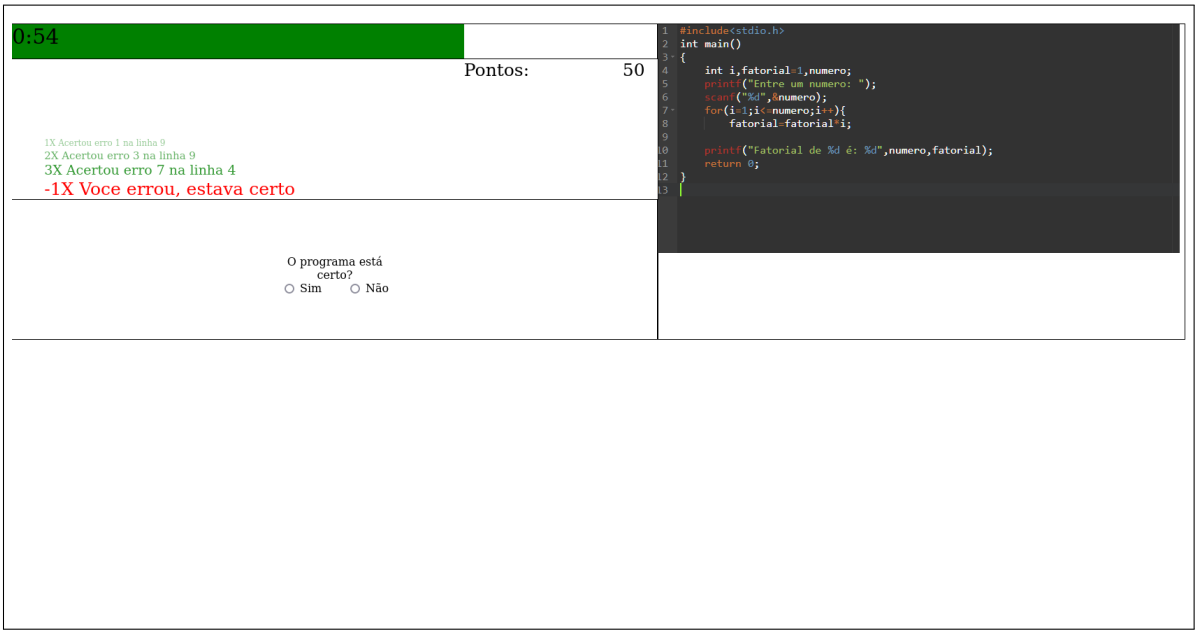


Figura 4.9: Tela 8

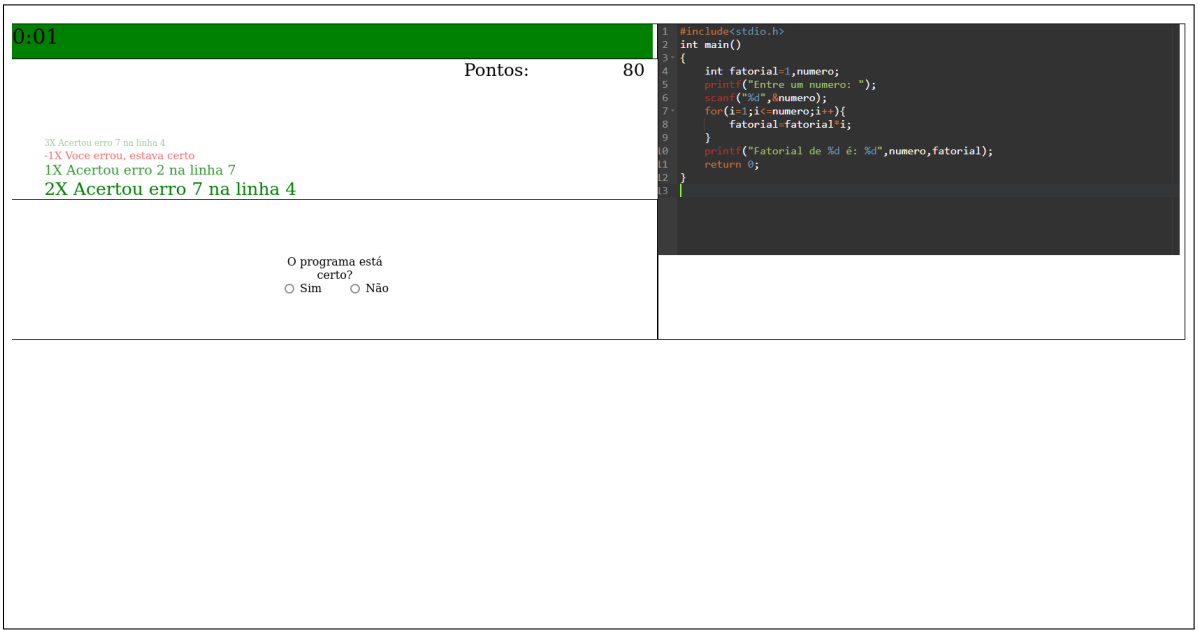


Figura 4.10: Tela 9

Figura 4.11: Placar

4.6 CONCLUSÃO

A proposta foi apresentada com as descrições do sistema, dos módulos, das regras do jogo, das características educacionais, da implementação e de um cenário de uso. O sistema foi criado para treinar iniciantes de programação na busca e identificação de erros no código sem o auxílio de mensagens de erro do compilador. A ideia é fazer com que o programador consiga identificar erros por conta própria e não dependa da compilação, assim saindo da "programação movida a compilação". É uma ferramenta que também ajuda programadores inexperientes a ganhar familiaridade com diversos erros comuns de forma rápida e prática, ensinando com os erros de outros para que o usuário não precise aprender errando.

Com este capítulo concluído será apresentado os experimentos e resultados para validação do sistema.

5 VALIDAÇÃO

Este capítulo apresentará o questionário utilizado para avaliação da ferramenta. Os dados obtidos serão apresentados e uma discussão sobre estes resultados serão feitos.

5.1 QUESTIONÁRIO DE AVALIAÇÃO

Para a coleta de dados de avaliadores foi criado um questionário baseado no modelo TAM(Technology Acceptance Model) no *Google Forms* que pode ser visto no apêndice B. Cada questão de múltipla escolha é uma escala de 1 a 5 com os seguintes significados:

1. Discordo Muito
2. Discordo Pouco
3. Neutro
4. Concordo Pouco
5. Concordo Muito

As perguntas são divididas nas seguintes categorias:

- Percepção de Utilidade: Onde as perguntas visam identificar o quão **Útil** a ferramenta é baseado na percepção dos avaliadores.
- Percepção de Facilidade de Uso: Que visam identificar o quão **Fácil de usar** a ferramenta é.
- Gamificação: Que buscam identificar a influência dos elementos de gamificação no desempenho dos avaliadores.
- Elementos Visuais: Que avaliam o efeito e adequação das escolhas visuais no projeto.
- Comentários e Sugestões: Onde avaliadores podem dar comentários e sugestões do quê acharam bom, ruim, problemas e como pode ser melhorado.

Os avaliadores são voluntários do Curso de Bacharelado de Ciência da Computação, incluindo tanto alunos quanto professores.

5.2 RESULTADOS

Tendo explicado o formato do questionário, agora serão mostrados os resultados por categoria.

5.2.1 Utilidade

Os resultados das perguntas sobre percepção de utilidade são mostrados nas figuras 5.1, 5.2, 5.3, 5.4, 5.5 e 5.6.

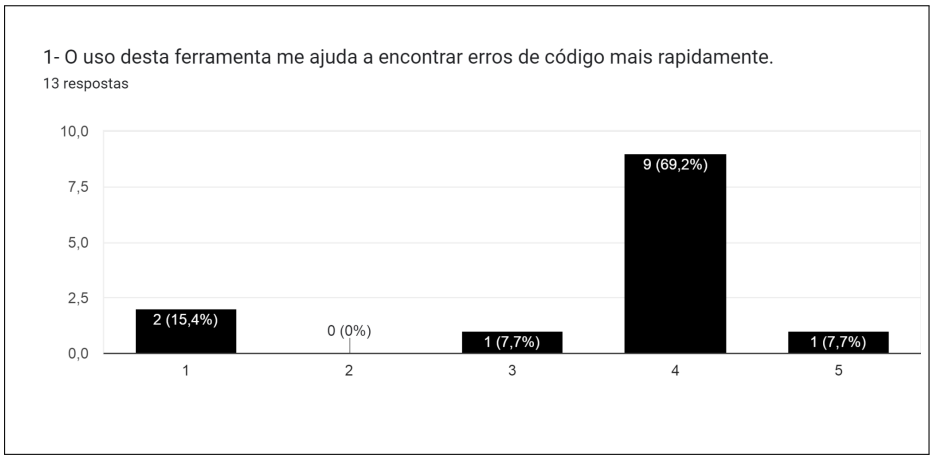


Figura 5.1: Resultado da pergunta 1

Figura 5.2: Resultado da pergunta 2

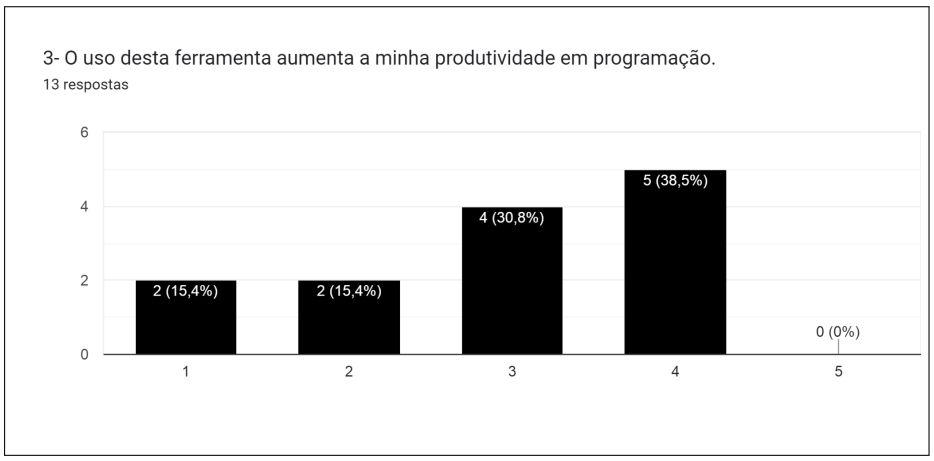


Figura 5.3: Resultado da pergunta 3

Figura 5.4: Resultado da pergunta 4

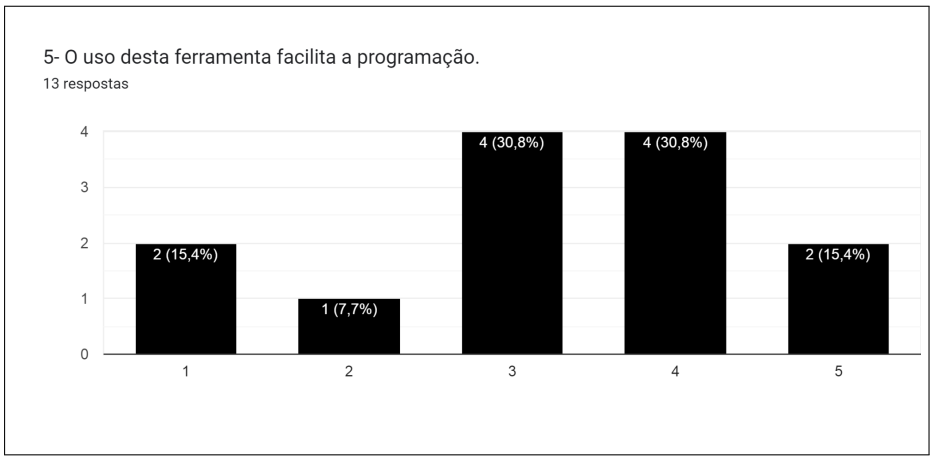


Figura 5.5: Resultado da pergunta 5

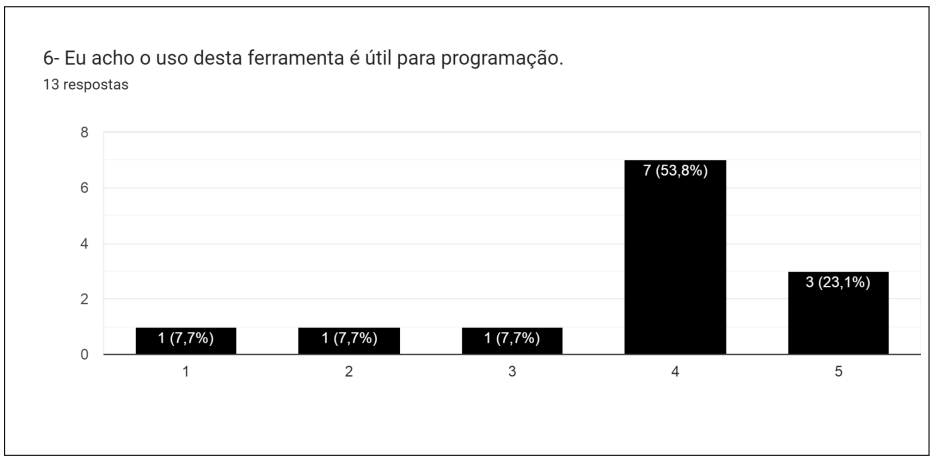


Figura 5.6: Resultado da pergunta 6

5.2.2 Facilidade de Uso

Os resultados das perguntas sobre percepção de facilidade são mostrados nas figuras 5.7, 5.8, 5.9, 5.10, 5.11 e 5.12.

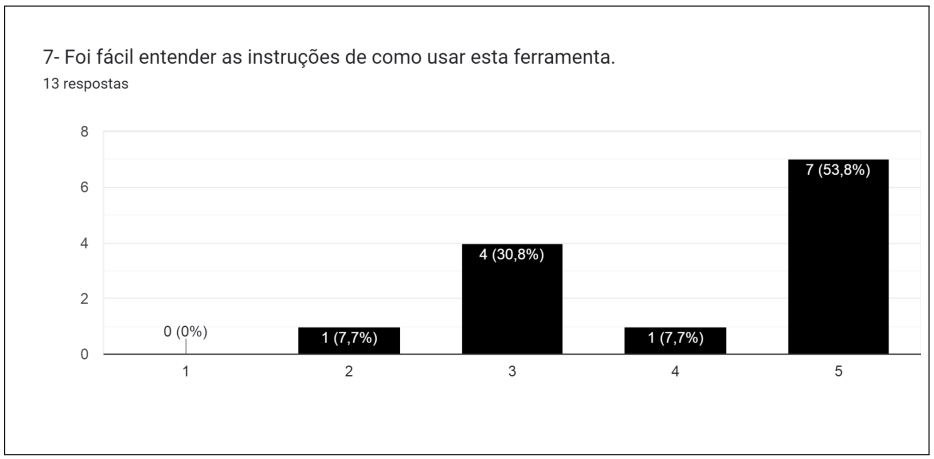


Figura 5.7: Resultado da pergunta 7

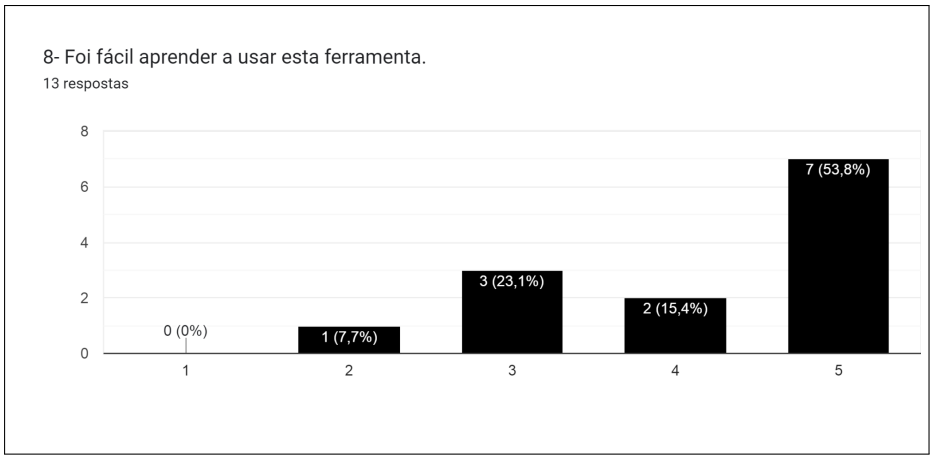


Figura 5.8: Resultado da pergunta 8

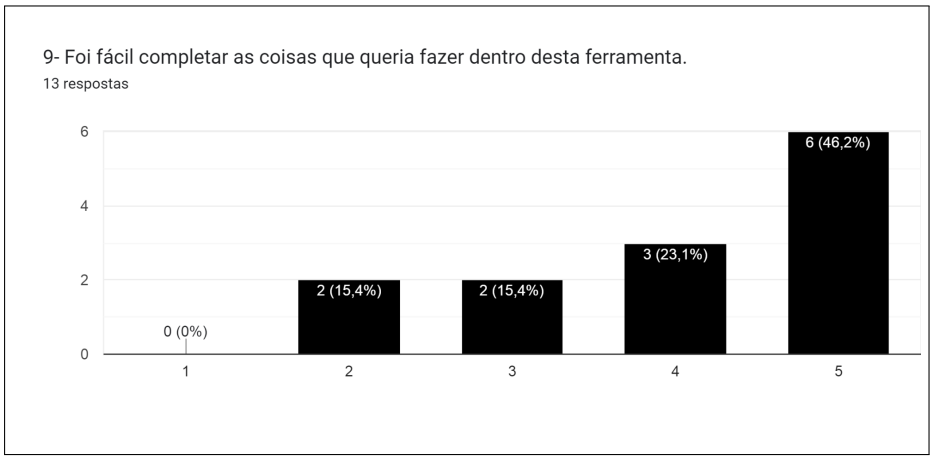


Figura 5.9: Resultado da pergunta 9

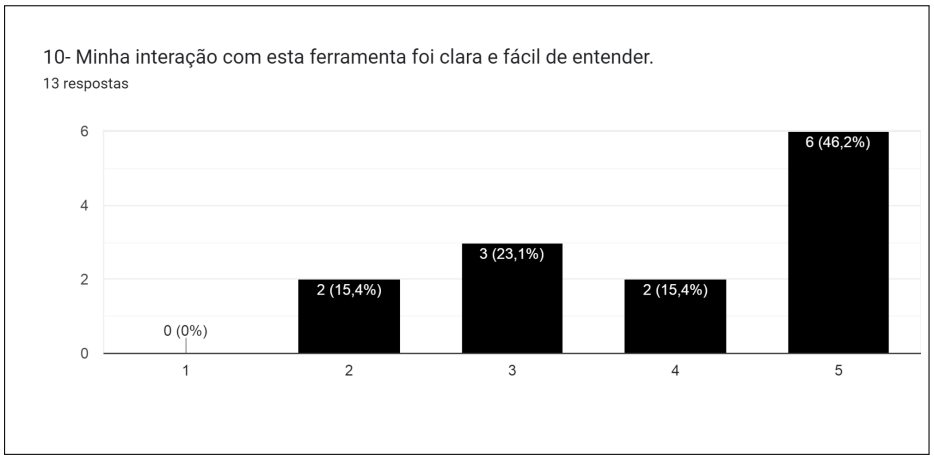


Figura 5.10: Resultado da pergunta 10

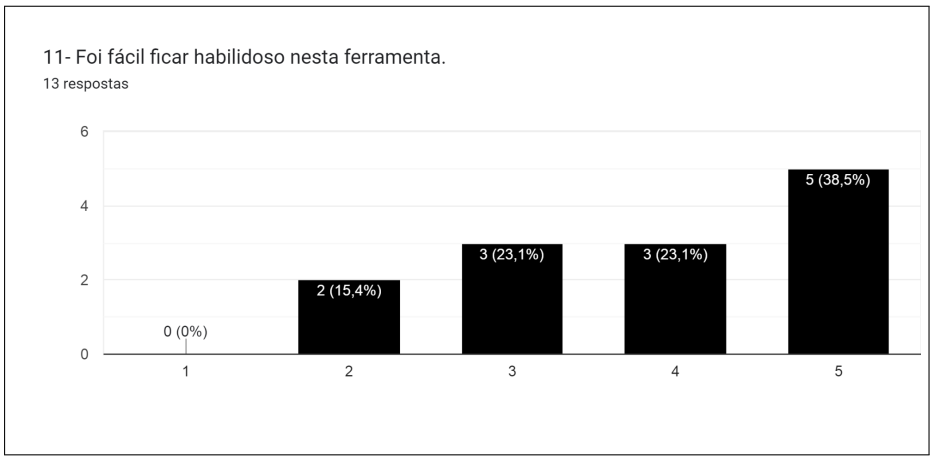


Figura 5.11: Resultado da pergunta 11

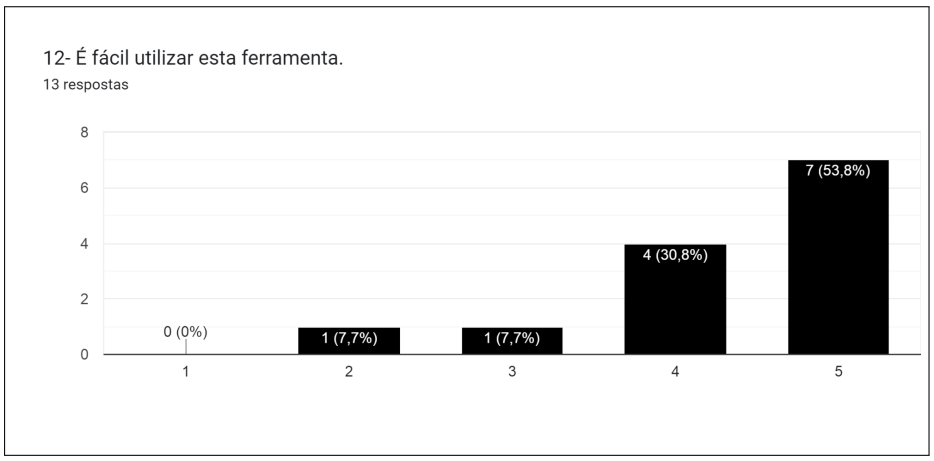


Figura 5.12: Resultado da pergunta 12

5.2.3 Gamificação

Os resultados das perguntas sobre gamificação são mostrados nas figuras 5.13, 5.14 e 5.15.

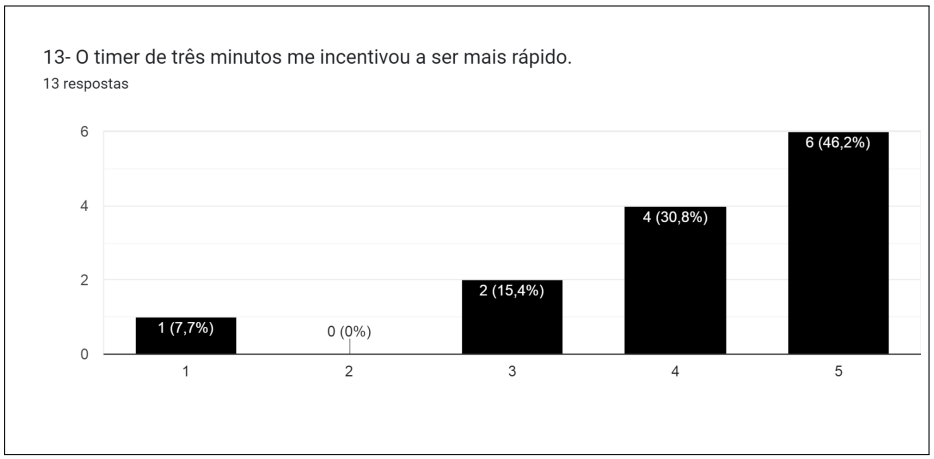


Figura 5.13: Resultado da pergunta 13

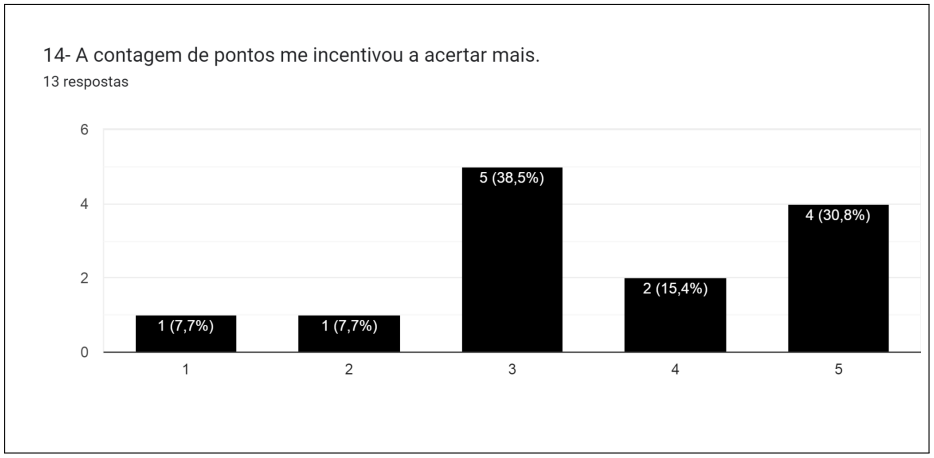


Figura 5.14: Resultado da pergunta 14

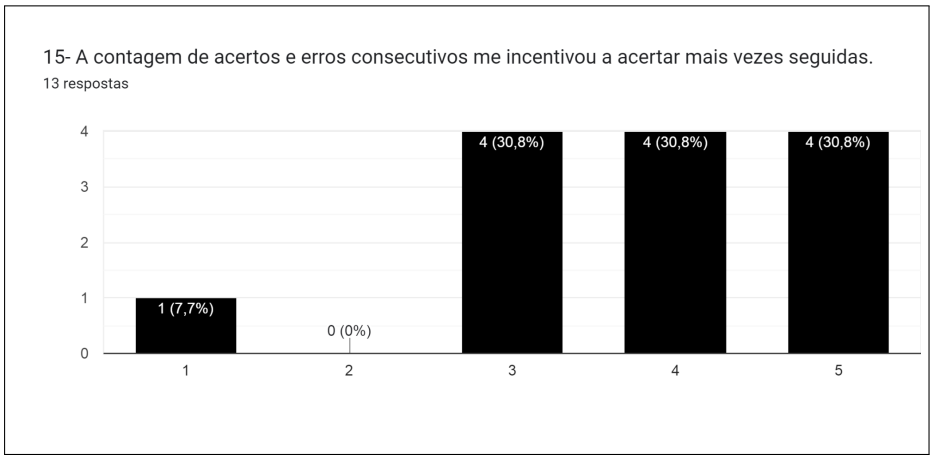


Figura 5.15: Resultado da pergunta 15

5.2.4 Elementos Visuais

Os resultados das perguntas sobre elementos visuais são mostrados nas figuras 5.16, 5.17, 5.18 e 5.19.

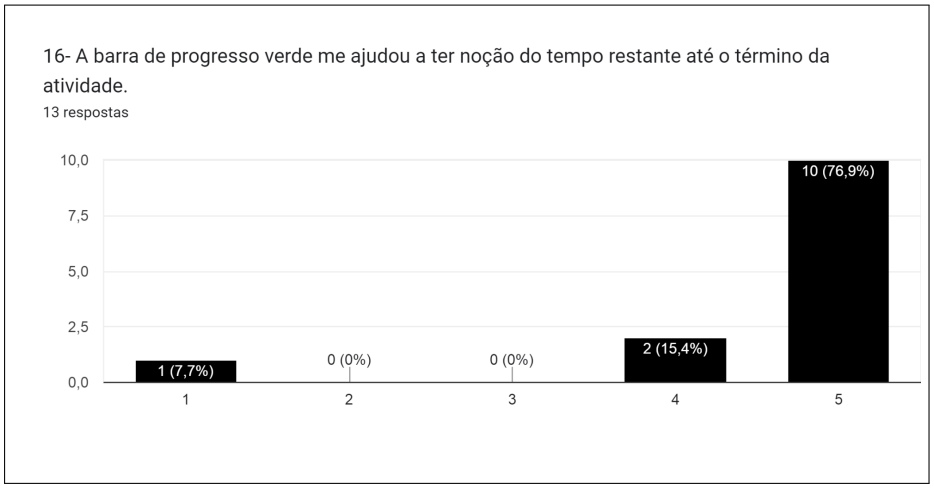


Figura 5.16: Resultado da pergunta 16

Figura 5.17: Resultado da pergunta 17

5.2.5 Comentários e Sugestões

Na questão aberta de comentários e sugestões houve vários comentários positivos e negativos sobre o sistema, além de problemas e sugestões de melhorias. Para ser breve os comentários serão categorizados e não serão tratados individualmente mas sim por assunto.

- Comentários positivos: Se resumem a elogios sobre a proposta e funcionamento do programa.
- Sobre o não funcionamento do Placar: Este *bug* era esperado visto que o placar depende do servidor, que não pode ser alcançado por conexões fora da rede interna do laboratório além de correr risco de ser desligado por outros alunos. Fora os acessos remotos, uma boa parte das avaliações aconteceram num período após o computador que rodava o servidor ter sido desligado.

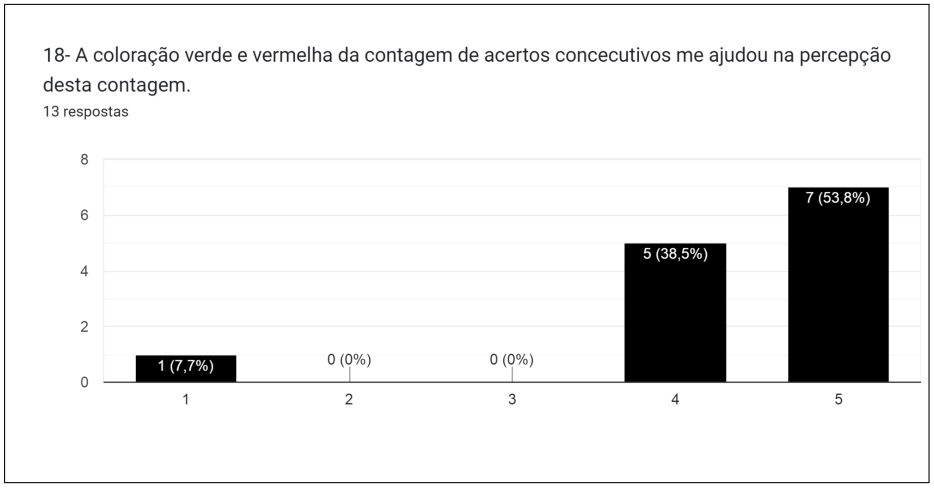


Figura 5.18: Resultado da pergunta 18

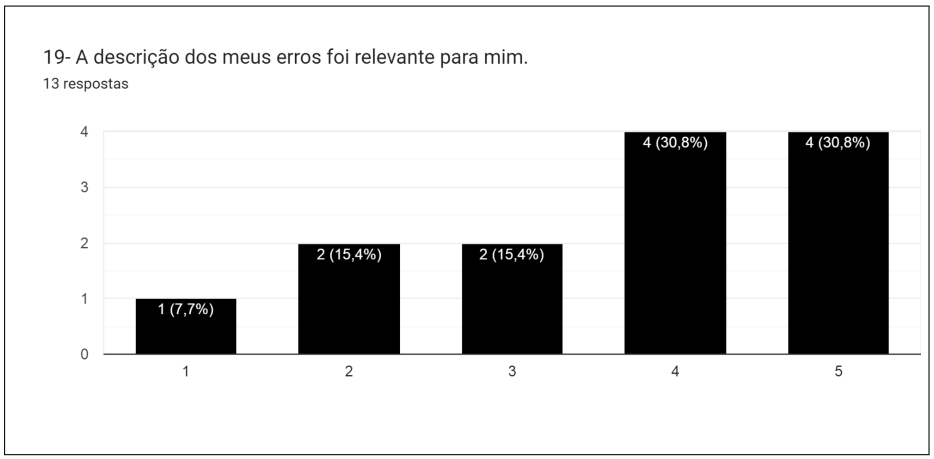


Figura 5.19: Resultado da pergunta 19

- Sobre desejo de estatísticas: Está relacionado à categoria anterior, tais estatísticas estão no Placar, que não funcionou.
- Sobre a repetição de exercícios: Por ser aleatório é possível repetir exercícios, a similaridade de exercícios(mesmo algoritmo com erros diferentes) também pode ter influenciado esta opinião. Este problema pode ser solucionado por modificações no sistema de escolha de exercícios para que não repita e na ampliação do catálogo de exercícios, diluindo os similares.
- Sobre a relevância de um sistema para erros sintáticos: Não só é incoerente com a proposta quanto justifica a criação de tal sistema.
- Não percepção de efeitos fade-out: Tais efeitos foram inclusos justamente por causa da transição abrupta, entretanto podem ser acentuados mais ainda.
- Não percepção de outros elementos: Provavelmente por falta de atenção ou irrelevância a resolução dos exercícios(é melhor focar em acertar e consequentemente ganhar muitos pontos do que prestar atenção em quantos pontos está fazendo).

- Sobre o conteúdo do feedback do erro: A escolha de mostrar o código do erro ao invés da descrição foi feita para reduzir a mensagem e não ocupar muito espaço da interface, facilitando a legibilidade. É possível ser alterado.
- Pausas entre exercícios para analisar o erro e qual deveria ser a resposta e códigos certos: Foge um pouco da proposta, mas é possível. Sugestão lida e descartada para este trabalho, talvez em um trabalho futuro.
- Irrelevância dos pontos por falta de comparação com outros jogadores: Sugestão lida e considerada, um ranking resolve isto.
- Uso de *markdown* no lugar de imagens para mostrar código: Sugestão lida e considerada, mas descartada por facilitar o uso de copia-e-cola no texto para um compilador a fim de obter vantagens indevidas.
- Comentários negativos(não específicos): Fora sentimentos negativos não há como analisar.

5.3 DISCUSSÃO

Analisando as respostas é possível dizer que os sentimentos em relação a ferramenta foram em maioria positivas embora não unânimes. Os comentários podem ser separados em algumas categorias:

- Comentários positivos sobre a ferramenta
- Comentários negativos válidos(concretos) sobre a ferramenta
- Comentários negativos/neutros sobre a **experiência** na ferramenta(sem justificativa concreta)
- Comentários negativos inválidos/incoerentes com a proposta
- Sugestões de melhoria válidas
- Sugestões de melhoria inválidas/incoerentes com a proposta

Os sentimentos dos comentários podem ser resumidos com comentários em geral positivos, com problemas válidos como a falta do Placar e irrelevância dos Pontos, sugestões incoerentes com a proposta e sugestões válidas, interessantes e promissoras.

Um dos participantes respondeu o mínimo em todas as questões e não apontou nenhum defeito, apenas insatisfação e será desconsiderado da análise.

Analisando as respostas sobre a utilidade da ferramenta, conseguimos concluir que: A ferramenta é um pouco útil na melhoria de desempenho na programação.

Analisando as respostas sobre a facilidade de uso da ferramenta, conseguimos concluir que: A ferramenta é fácil de aprender a usar.

Analisando as respostas sobre a gamificação na ferramenta, conseguimos concluir que: O elemento de timer de fato trouxe urgência e incentivou a velocidade de resolução. A contagem de pontos foi pouco efetiva sem um ranking ou histórico para comparar. A contagem de acertos e erros consecutivos gerou resultados inconclusivos, mas não negativos.

Analisando as respostas sobre os elementos visuais na ferramenta, conseguimos concluir que: A barra de progresso teve um efeito significativamente positivo na percepção da passagem

de tempo. Os efeitos de transição suaves não foram muito perceptíveis e a opinião foi ambivalente. A coloração verde e vermelha foram bem úteis na percepção da contagem de erros e acertos. A descrição dos erros foi relevante, mas não muito.

5.4 CONCLUSÃO

A avaliação mostra resultados positivos mas não sem contestação para a utilidade e facilidade da ferramenta e a efetividade da gamificação. Percebe-se pela junção das respostas com os comentários que os resultados poderiam ser consideravelmente melhorados com mudanças e incrementos na implementação do sistema, mas que a ideia proposta em si funciona e tem resultados que validam este trabalho.

Os resultados foram apresentados e brevemente discutidos, no próximo capítulo serão feitas inferências sobre os resultados e conclusões sobre o projeto, além de melhorias em trabalhos futuros.

6 CONCLUSÃO

Neste trabalho de graduação, foi apresentado um sistema web, que tem como objetivo o auxílio no treinamento de iniciantes em programação na identificação e resolução de erros sintáticos. Utilizando-se de elementos de gamificação e a ausência de compilações e modificações no código, é esperado que estudantes ganhem experiência com a sintaxe da linguagem C e consigam escrever códigos mais corretos com menor necessidade de depender de mensagens de erro do compilador.

Após a testagem e avaliação do objeto educacional obtivemos respostas positivas. O sentimento geral dos testadores foi de positividade em relação ao uso do sistema para treinamento de *debugging*. Embora os resultados não sejam absolutos, a ideia proposta é coerente e válida, com críticas negativas sendo principalmente fundamentadas na implementação e não na teoria, o que permite melhorias. A avaliação serve apenas para medir o sentimento dos testadores em relação a ferramenta, para medir a eficácia real seria necessário os dados de desempenho dos usuários, que não foram recebidos devido a problemas no servidor por razão de desligamento do dispositivo. Outro dado importante é que os avaliadores sentiram que os elementos de gamificação usados tiveram um impacto positivo em seu desempenho.

Neste trabalho, foi visto os trabalhos relacionados que se propõem a melhorar o *debugging* e o ensino de programação, conscientizando o leitor sobre o contexto onde este trabalho está inserido. Foi exposto os erros sintáticos mais recorrentes. Foi proposto o sistema para treinamento de *debugging*. E por fim, esta ferramenta foi avaliada, validando a ferramenta proposta. Com isso, os objetivos deste trabalho de graduação foram atingidos.

6.1 TRABALHOS FUTUROS

Nesta seção, será discutido os trabalhos futuros desta ferramenta, melhorias e correções de *bugs*, novas funcionalidades e mudanças.

6.1.1 Melhorias e Correções

Primeiramente vem as correções de *bugs* e melhorias de coisas que não estão boas:

- O servidor responsável pelo Placar precisa ser melhorado, como por exemplo apresentar uma versão offline do Placar e verificar a conexão com o servidor antes de redirecionar a página;
- O tempo de Fade-out pode ser aumentado ou o efeito em si pode ser mudado para Slide;
- O número e variedade de exercícios precisa ser aumentado;
- Feedback de erros melhor;
- O *layout* da página precisa ser reconfigurada para melhor experiência;
- A redução de cliques necessários para resolução de cada exercício, atualmente 5, poderia ser 3 ou até menos.

6.1.2 Novas Funcionalidades

Ainda é possível adicionar novas funcionalidades neste projeto como:

- Inclusão de um Ranking;
- Inclusão de Perfil com histórico e recorde pessoal;
- Novos elementos de gamificação podem ser introduzidos como personagens, narrativa, níveis diferentes, etc.
- Análise de tipos de erros com o passar do tempo. Atualmente só são analisados os acertos ou erros ao longo do tempo, desconsiderando o tipo específico.

6.1.3 Mudanças

Alguns fatores opcionais que podem ser mudados, não necessariamente por estarem ruins, mas para experimentar configurações e propostas diferentes a fim de obter experiências melhores e pesquisas alternativas:

- Modificar a escolha de exercícios para que não seja aleatória, mas de forma que não se repita exercícios ou que repita os exercícios mais errados;
- Pausas entre exercícios para verificação de erro e comparação do código errado e qual deveria ser certo.
- Automatizar a geração de erros a partir de uma base de exemplos corretos.

REFERÊNCIAS

Becker, B. A., Goslin, K. e Glanville, G. (2018). The effects of enhanced compiler error messages on a syntax error debugging test. Em *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, páginas 640–645.

Bofferding, L., Kocabas, S., Aqazade, M., Haiduc, A.-M. e Chen, L. (2022). The effect of play and worked examples on first and third graders' creating and debugging of programming algorithms. Em *Computational Thinking in PreK-5: Empirical Evidence for Integration and Future Directions*, páginas 19–29. Association for Computing Machinery.

Costa, A. C. S. e Marchiori, P. Z. (2015). Gamificação, elementos de jogos e estratégia: uma matriz de referência. *InCID: Revista de Ciência da Informação e Documentação*, 6(2):44–65.

de Oliveira, R. L., de Albuquerque Reis, T., Viera, M. A. e Charao, A. S. (2007). Avaliaçao do suportea programaçao multithread com openmp no compilador gcc. Em *VIII Workshop de Software Livre*.

Denny, P., Luxton-Reilly, A. e Carpenter, D. (2014). Enhancing syntax error messages appears ineffectual. Em *Proceedings of the 2014 conference on Innovation & technology in computer science education*, páginas 273–278.

Gavidia, J. J. Z. e ANDRADE, L. C. V. d. (2003). Sistemas tutores inteligentes. *Trabalho de Conclusão da Disciplina de IA, Universidade Federal do Rio de Janeiro. Rio de Janeiro–RJ: UFRJ*.

Gomes, M. e Amaral, E. (2016). Uma proposta de ferramenta para simplificar a depuração de códigos em c, por alunos iniciantes. Em *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, volume 5, página 1029.

Gomes, M., Becker, L., Gestaro, L., Amaral, É. e Tarouco, L. (2015). Um estudo sobre erros em programação: reconhecendo as dificuldades de programadores iniciantes. Em *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, volume 4, página 1398. sn.

Gomes, M. S. e do Amaral, É. M. (2016). Simplificando a depuração de códigos na linguagem c-uma solução para alunos iniciantes. *RENOTE*, 14(1).

Hara, C. e Zola, W. (2008). *Linguagem C - Notas de Aula , Parte I Programação Básica em C*. `https://www.inf.ufpr.br/cursos/ci067/Docs/NotasAula.pdf`.

Kummerfeld, S. K. e Kay, J. (2003). The neglected battle fields of syntax errors. Em *Proceedings of the fifth Australasian conference on Computing education-Volume 20*, páginas 105–111.

Lee, M. J. (2012). Social debugging game for learning & engagement. Em *2012 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, páginas 227–228. IEEE.

Lee, M. J. (2014). Gidget: An online debugging game for learning and engagement in computing education. Em *2014 ieee symposium on visual languages and human-centric computing (vl/hcc)*, páginas 193–194. IEEE.

Lee, M. J., Bahmani, F., Kwan, I., LaFerte, J., Charters, P., Horvath, A., Luor, F., Cao, J., Law, C., Beswetherick, M. et al. (2014). Principles of a debugging-first puzzle game for computing education. Em *2014 IEEE symposium on visual languages and human-centric computing (VL/HCC)*, páginas 57–64. IEEE.

Lee, M. J. e Ko, A. J. (2011). Personifying programming tool feedback improves novice programmers' learning. Em *Proceedings of the seventh international workshop on Computing education research*, páginas 109–116.

Lee, M. J. e Ko, A. J. (2014). A demonstration of gidget, a debugging game for computing education. Em *2014 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, páginas 211–212. IEEE.

Lee, M. J. e Ko, A. J. (2015). Comparing the effectiveness of online learning approaches on cs1 learning outcomes. Em *Proceedings of the eleventh annual international conference on international computing education research*, páginas 237–246.

Li, A., Endres, M. e Weimer, W. (2022). Debugging with stack overflow: Web search behavior in novice and expert programmers. Em *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Software Engineering Education and Training*, páginas 69–81.

Liu, Z., Zhi, R., Hicks, A. e Barnes, T. (2017). Understanding problem solving behavior of 6–8 graders in a debugging game. *Computer Science Education*, 27(1):1–29.

Lobo, F. (2023). Erros frequentes em c. `https://www.fernandolobo.info/pi/erros-frequentes.html`. Acessado em 20/05/2023.

Lowe, T. (2019). Debugging: The key to unlocking the mind of a novice programmer? Em *2019 IEEE Frontiers in Education Conference (FIE)*, páginas 1–9. IEEE.

McCauley, R., Fitzgerald, S., Lewandowski, G., Murphy, L., Simon, B., Thomas, L. e Zander, C. (2008). Debugging: a review of the literature from an educational perspective. *Computer Science Education*, 18(2):67–92.

Michaeli, T. e Romeike, R. (2019). Improving debugging skills in the classroom: The effects of teaching a systematic debugging process. Em *Proceedings of the 14th workshop in primary and secondary computing education*, páginas 1–7.

Robins, A., Rountree, J. e Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education*, 13(2):137–172.

Silvador (2014). Papers, please discussions. `https://steamcommunity.com/app/239030/discussions/0/522730700069334131/#c522730700083151772`. Acessado em 21/08/2022.

Tajra, S. F. (2011). *Informática na Educação: novas ferramentas pedagógicas para o professor na atualidade*. Saraiva Educação SA.

Whalley, J., Settle, A. e Luxton-Reilly, A. (2023). A think-aloud study of novice debugging. *ACM Transactions on Computing Education*.

APÊNDICE A – INTRUÇÕES

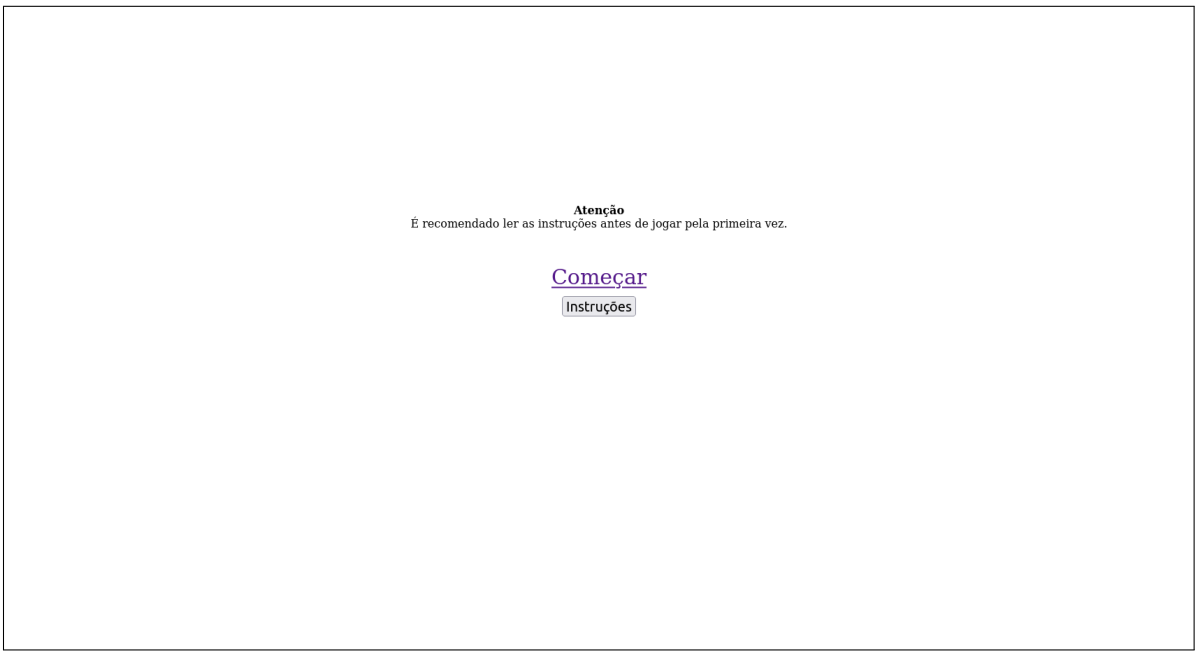


Figura A.1: Página inicial do sistema

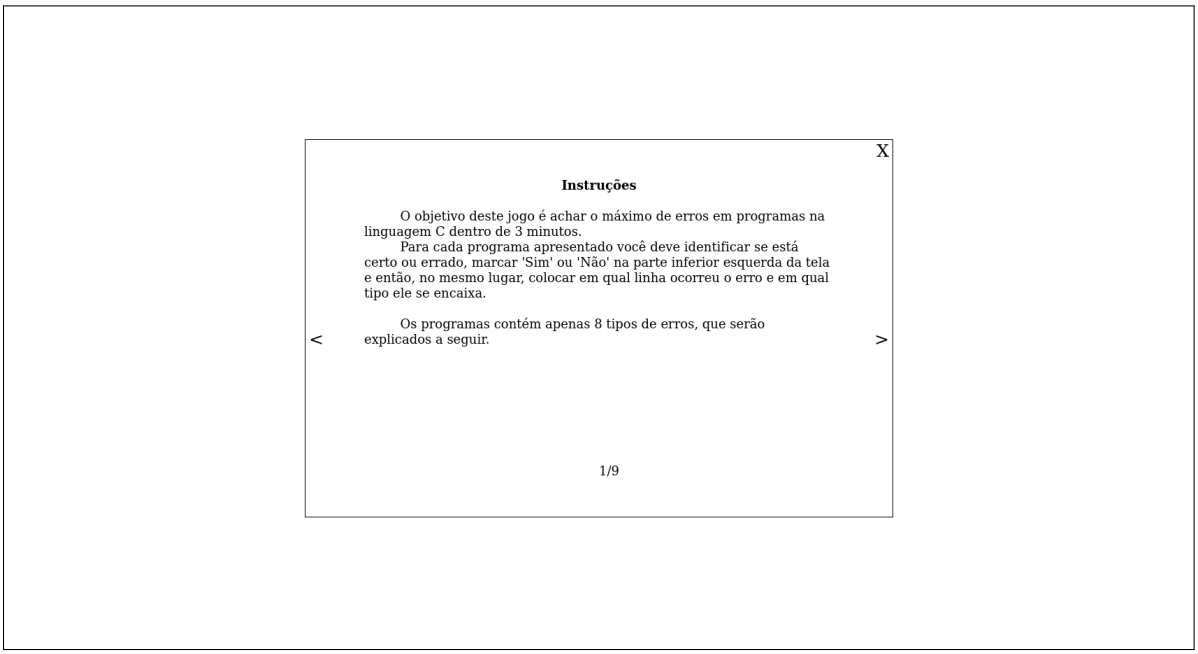


Figura A.2: Parte 1 de 9 do tutorial do sistema

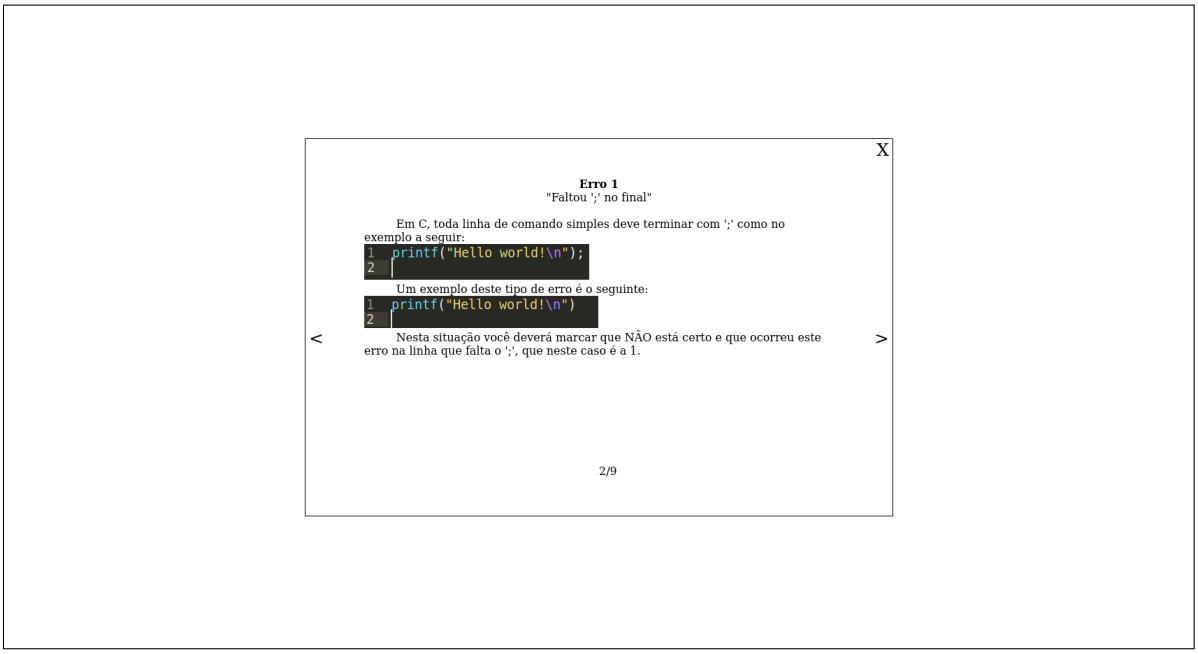


Figura A.3: Parte 2 de 9 do tutorial do sistema

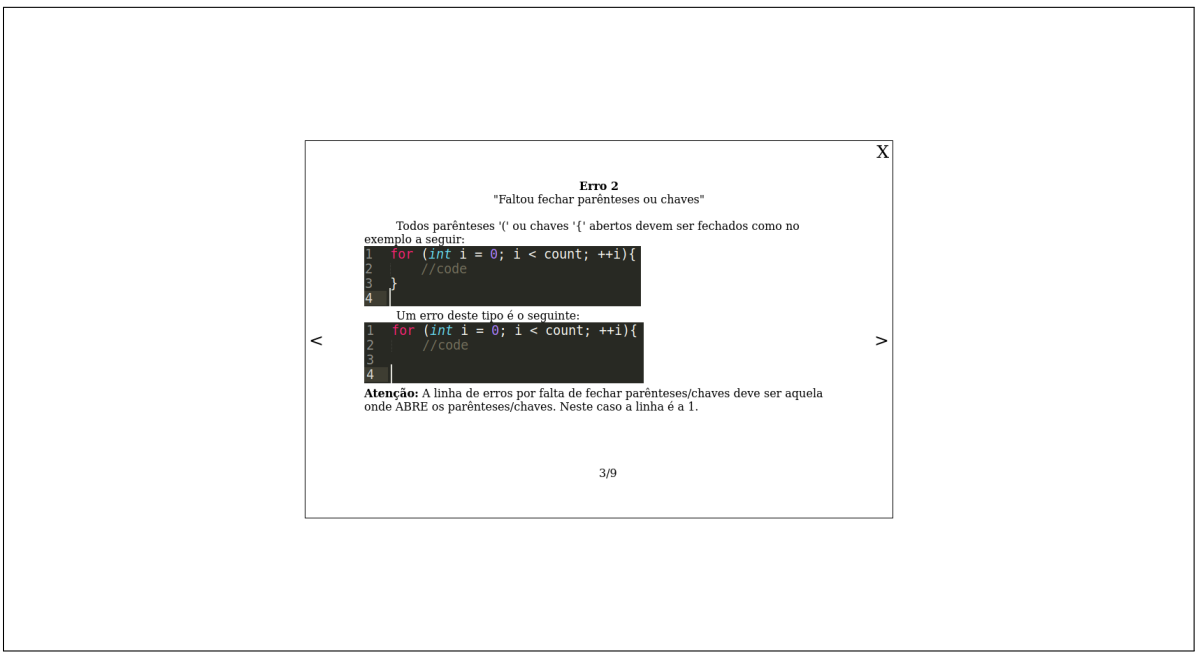


Figura A.4: Parte 3 de 9 do tutorial do sistema

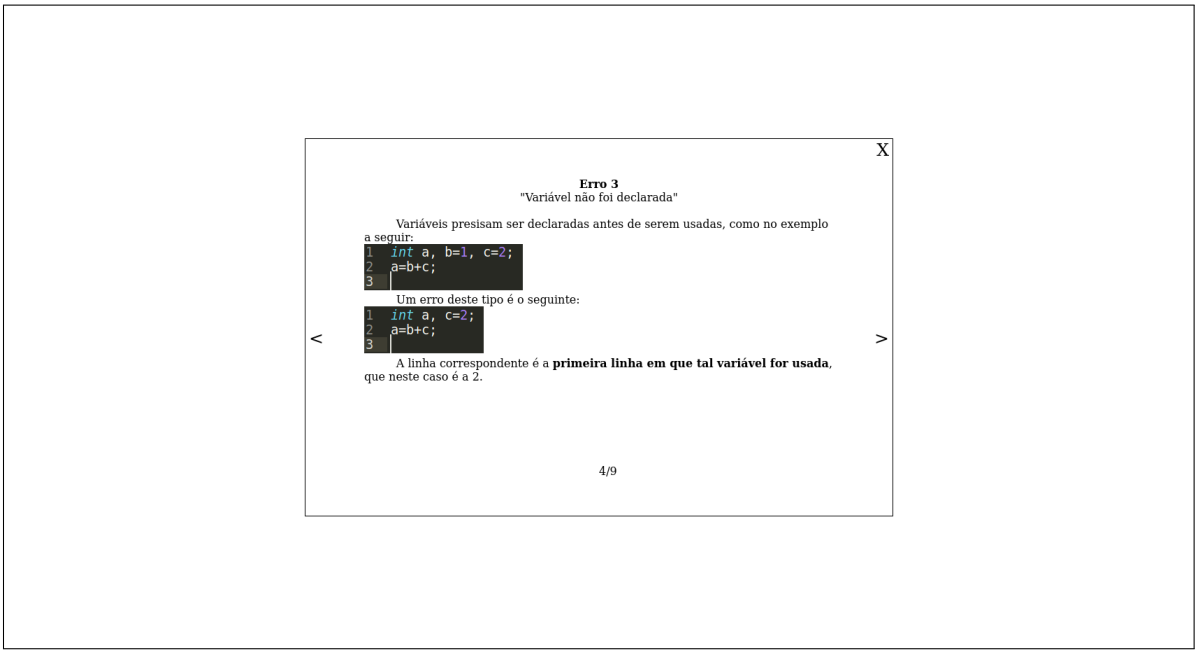


Figura A.5: Parte 4 de 9 do tutorial do sistema

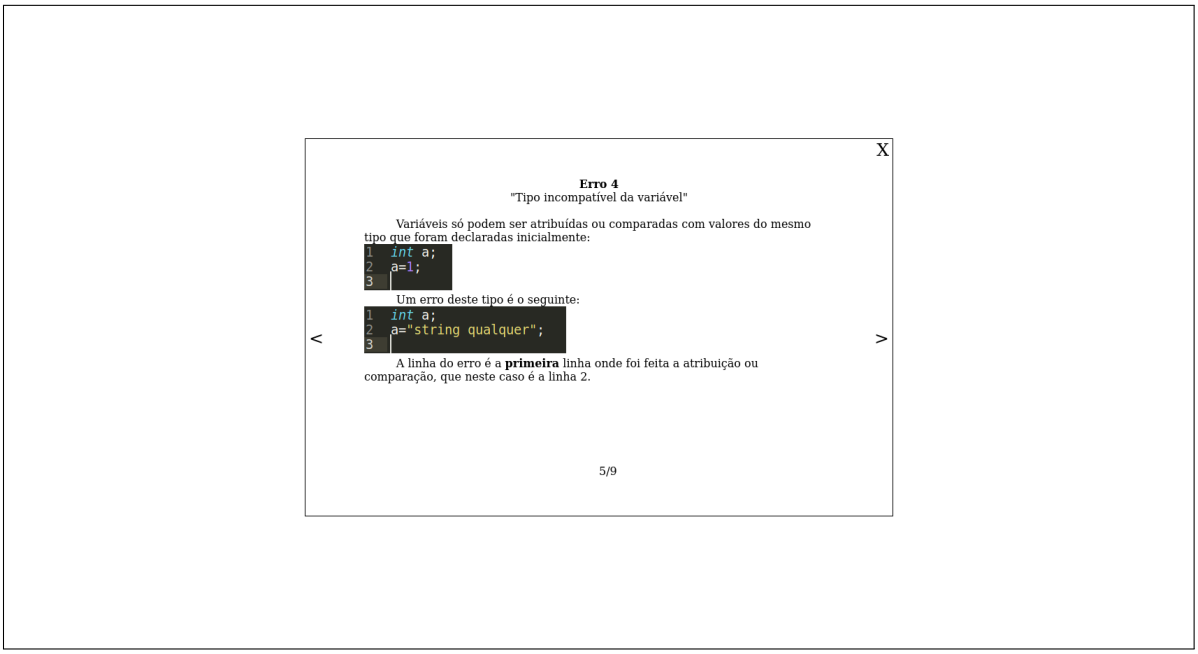


Figura A.6: Parte 5 de 9 do tutorial do sistema

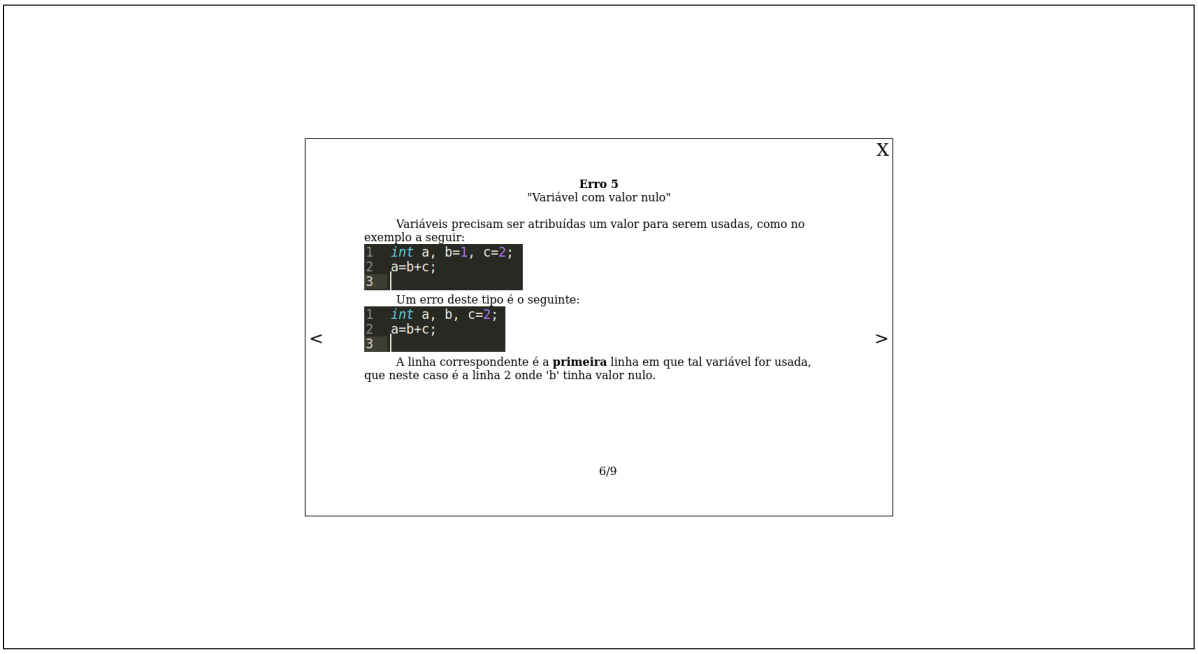


Figura A.7: Parte 6 de 9 do tutorial do sistema

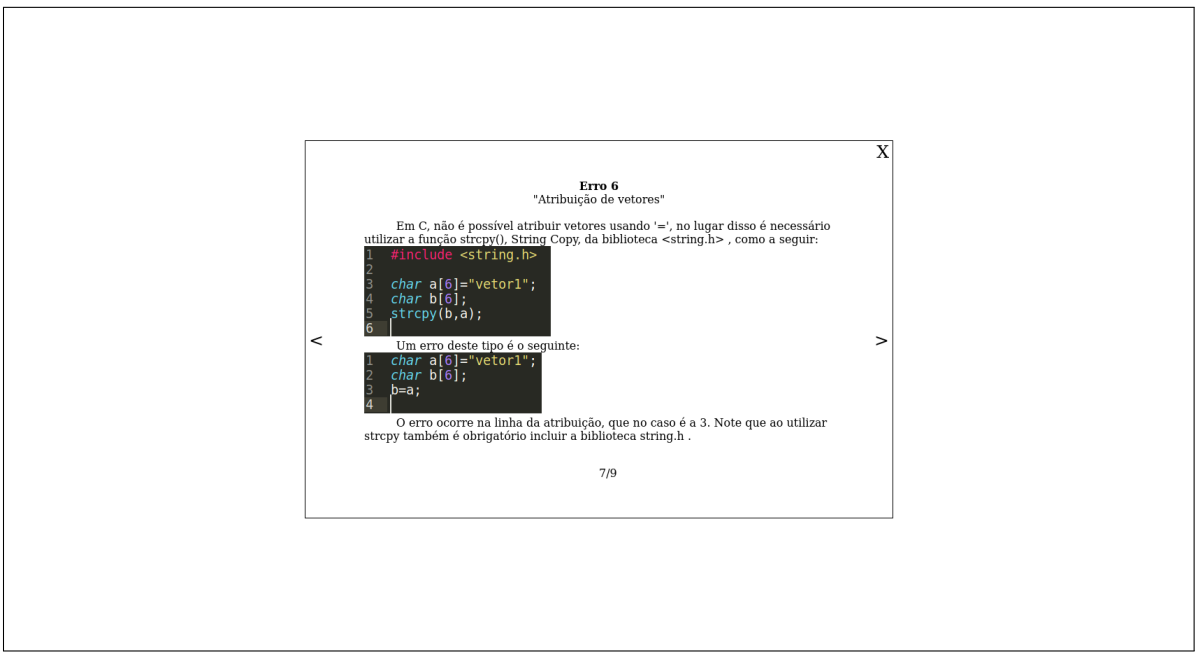


Figura A.8: Parte 7 de 9 do tutorial do sistema

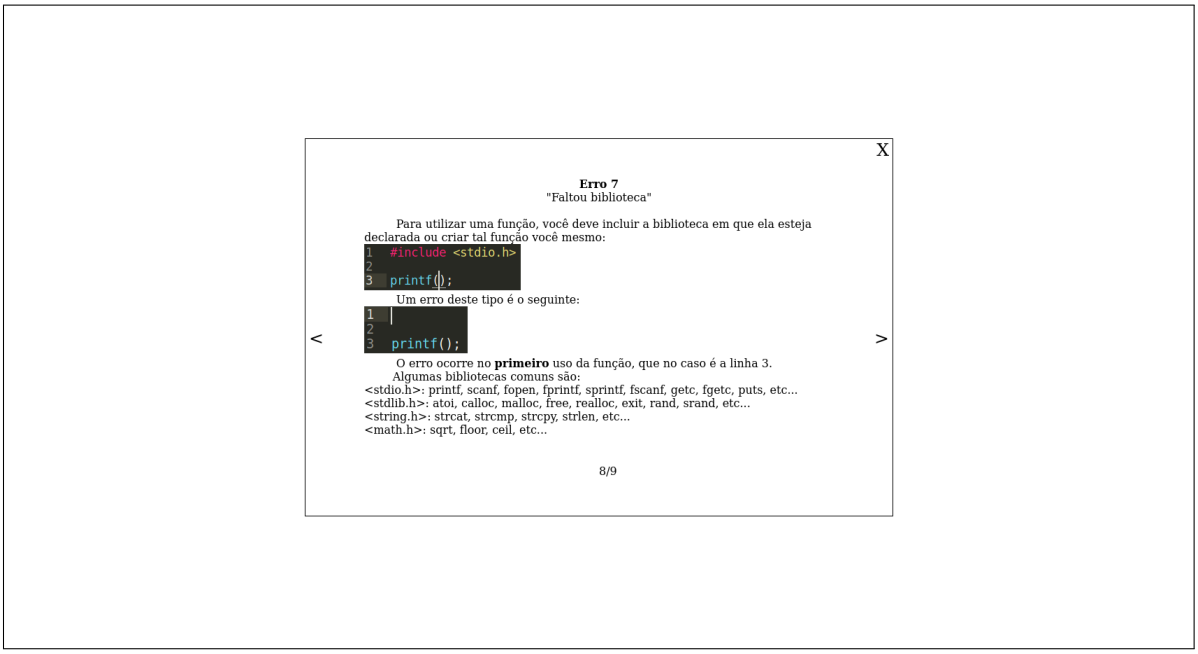


Figura A.9: Parte 8 de 9 do tutorial do sistema

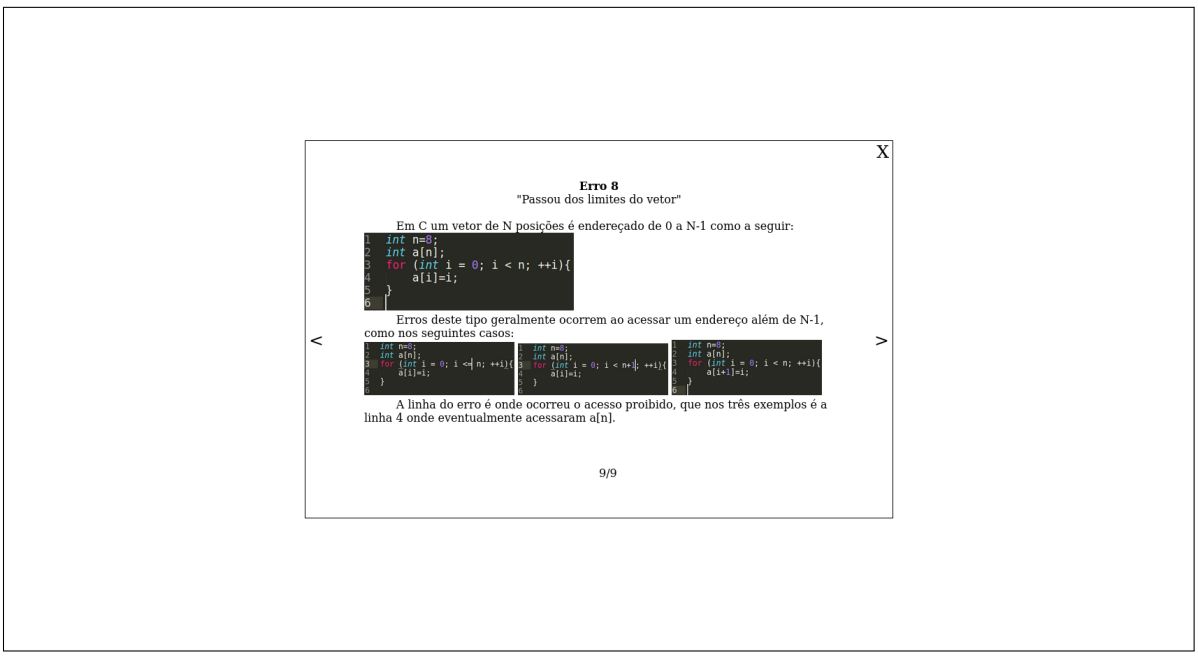
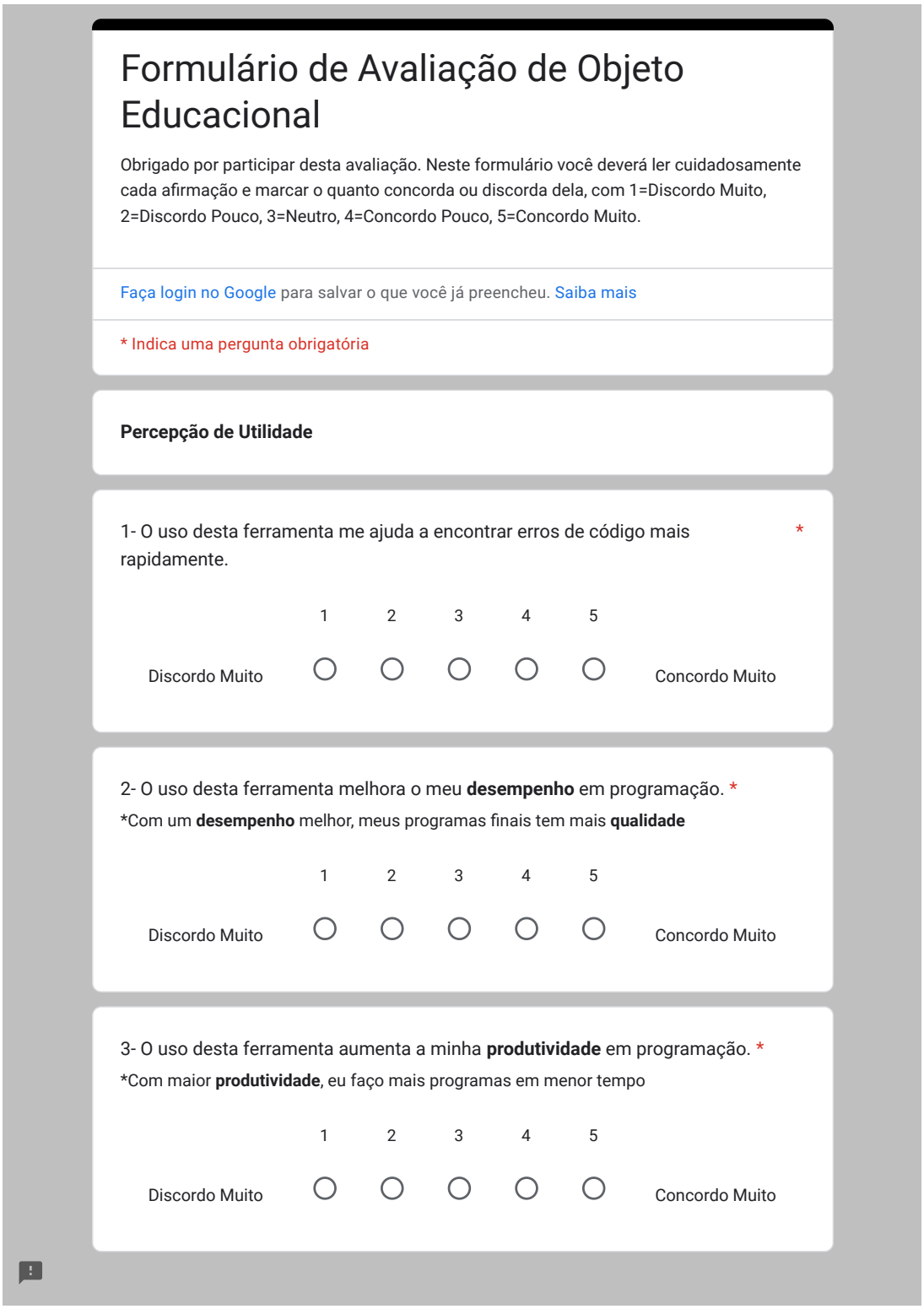


Figura A.10: Parte 9 de 9 do tutorial do sistema

APÊNDICE B – FORMULÁRIO DE AVALIAÇÃO DE OBJETO EDUCACIONAL



Formulário de Avaliação de Objeto Educacional

Obrigado por participar desta avaliação. Neste formulário você deverá ler cuidadosamente cada afirmação e marcar o quanto concorda ou discorda dela, com 1=Discordo Muito, 2=Discordo Pouco, 3=Neutro, 4=Concordo Pouco, 5=Concordo Muito.

Faça login no Google para salvar o que você já preencheu. Saiba mais

* Indica uma pergunta obrigatória

Percepção de Utilidade

1- O uso desta ferramenta me ajuda a encontrar erros de código mais rapidamente. *

	1	2	3	4	5	
Discordo Muito	○	○	○	○	○	Concordo Muito

2- O uso desta ferramenta melhora o meu **desempenho** em programação. *

*Com um **desempenho** melhor, meus programas finais tem mais **qualidade**

	1	2	3	4	5	
Discordo Muito	○	○	○	○	○	Concordo Muito

3- O uso desta ferramenta aumenta a minha **produtividade** em programação. *

*Com maior **produtividade**, eu faço mais programas em menor tempo

	1	2	3	4	5	
Discordo Muito	○	○	○	○	○	Concordo Muito

Figura B.1: Parte 1 de 5 do formulário usado para validação.

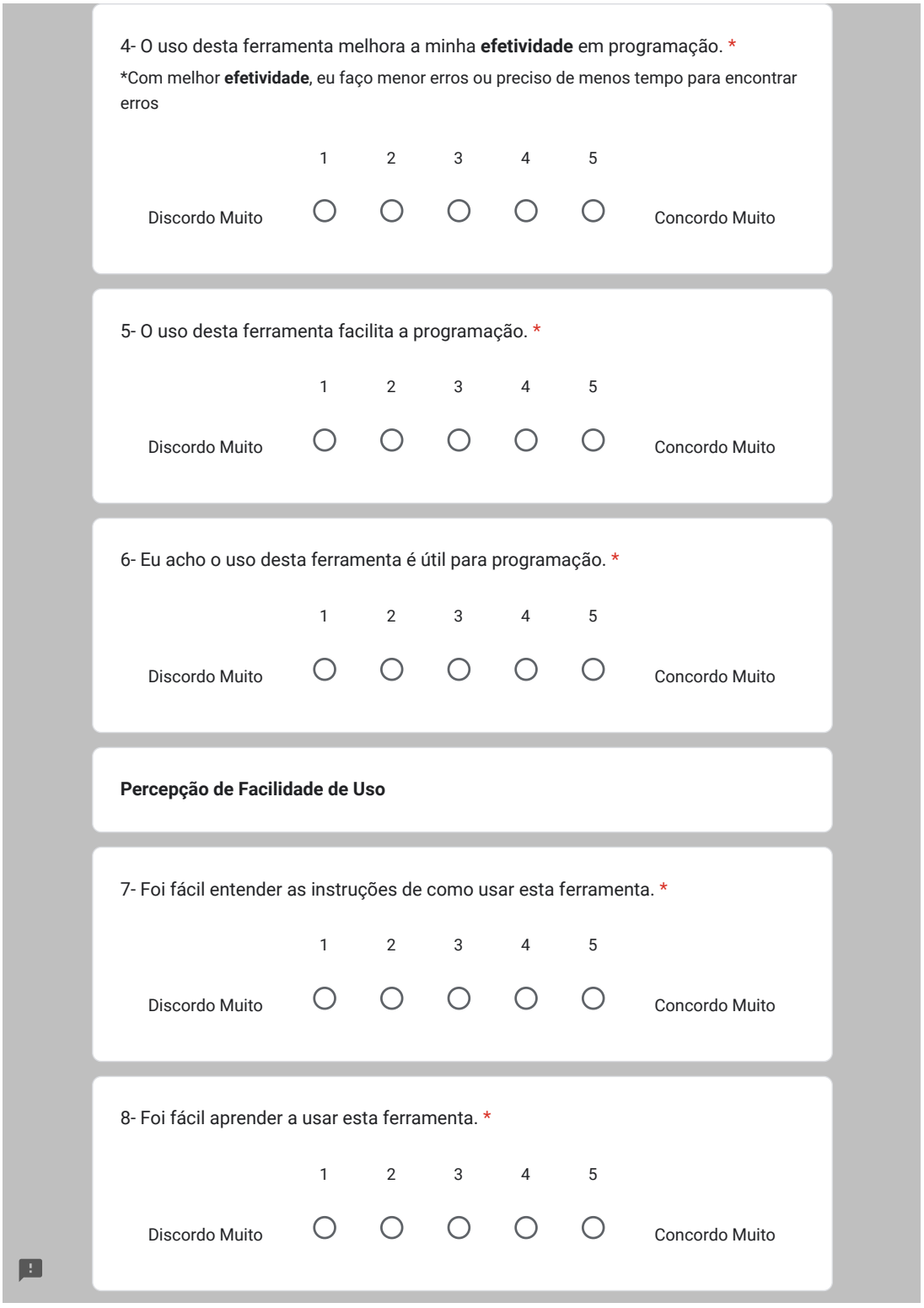


Figura B.2: Parte 2 de 5 do formulário usado para validação.

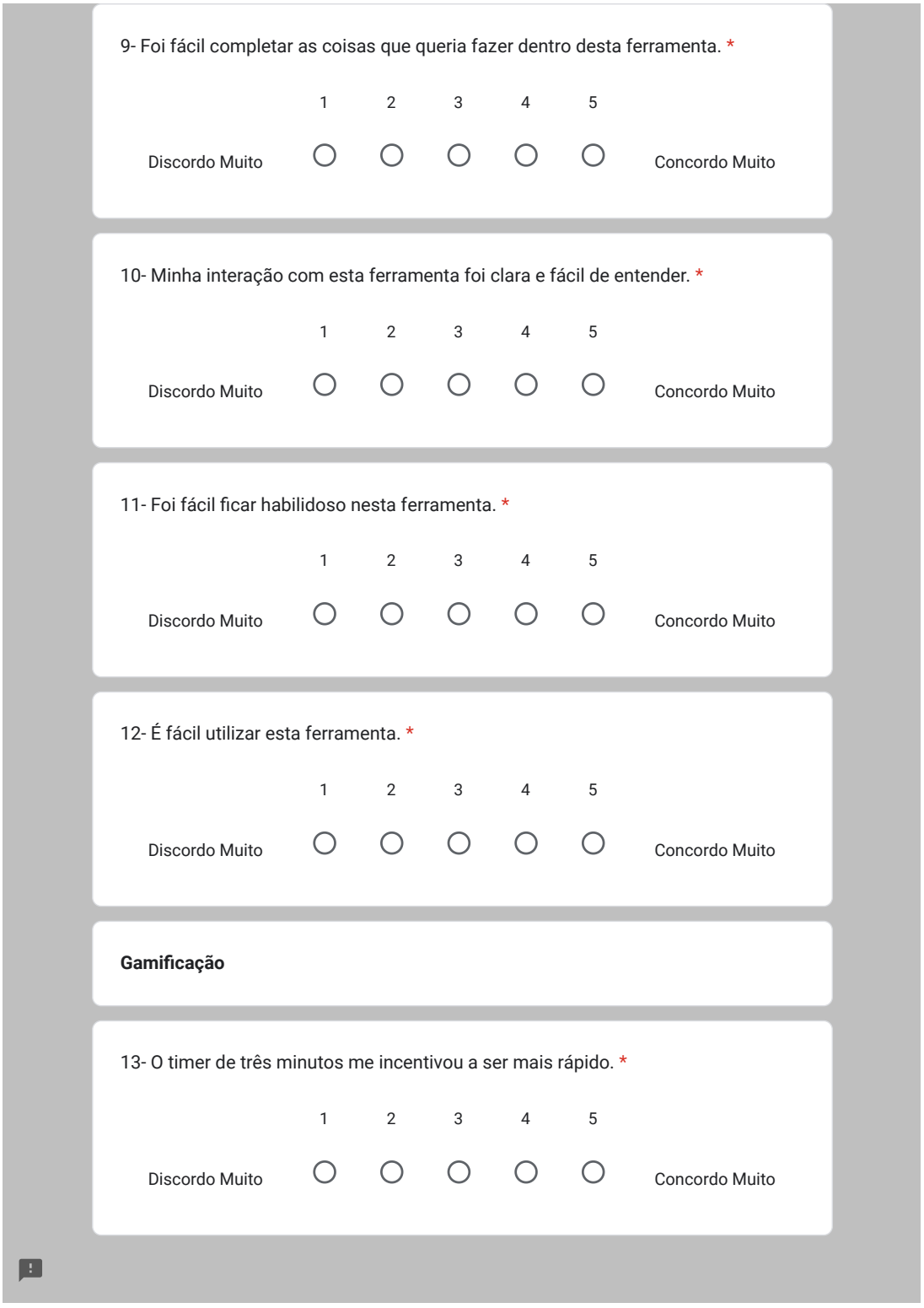


Figura B.3: Parte 3 de 5 do formulário usado para validação.

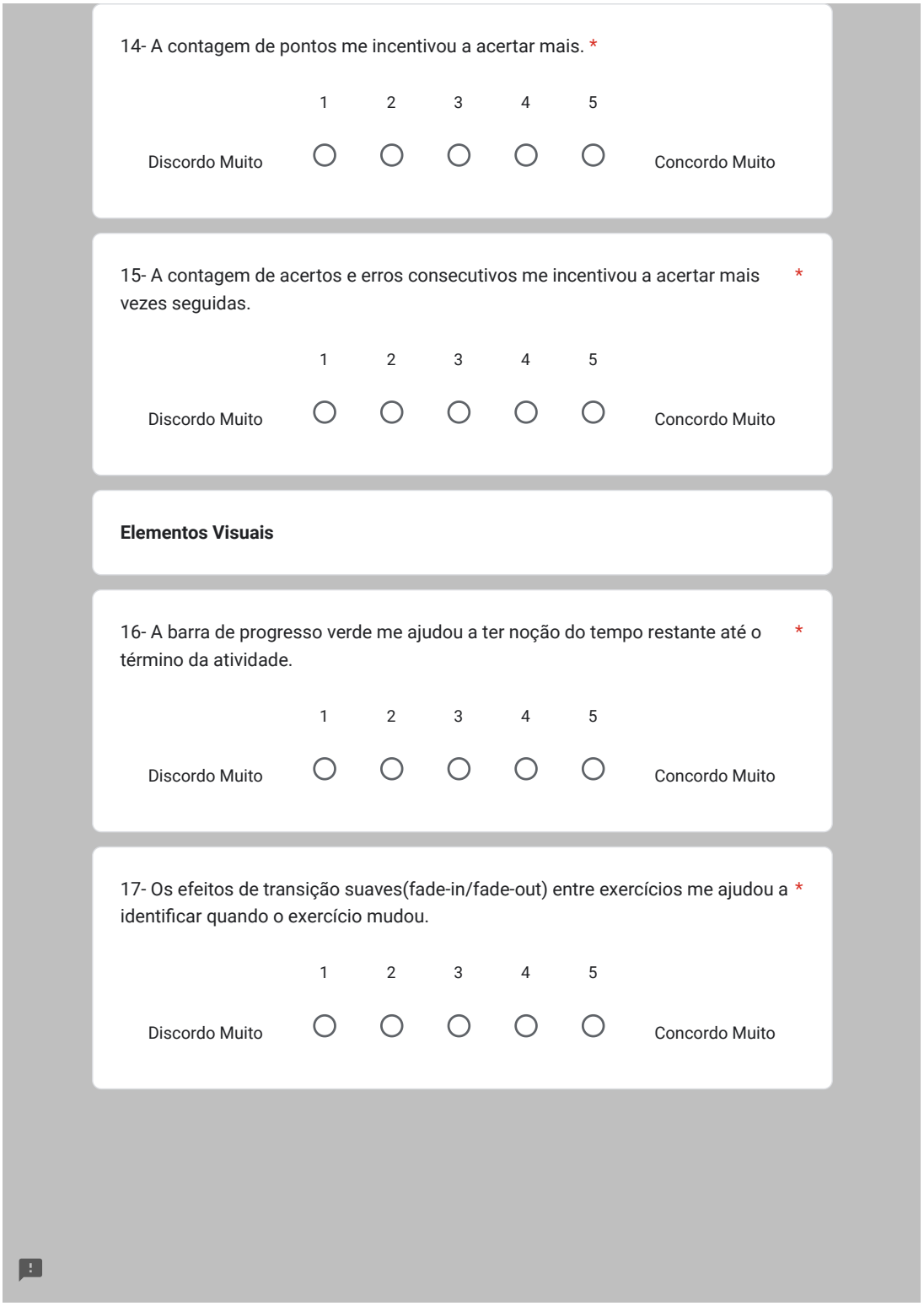


Figura B.4: Parte 4 de 5 do formulário usado para validação.

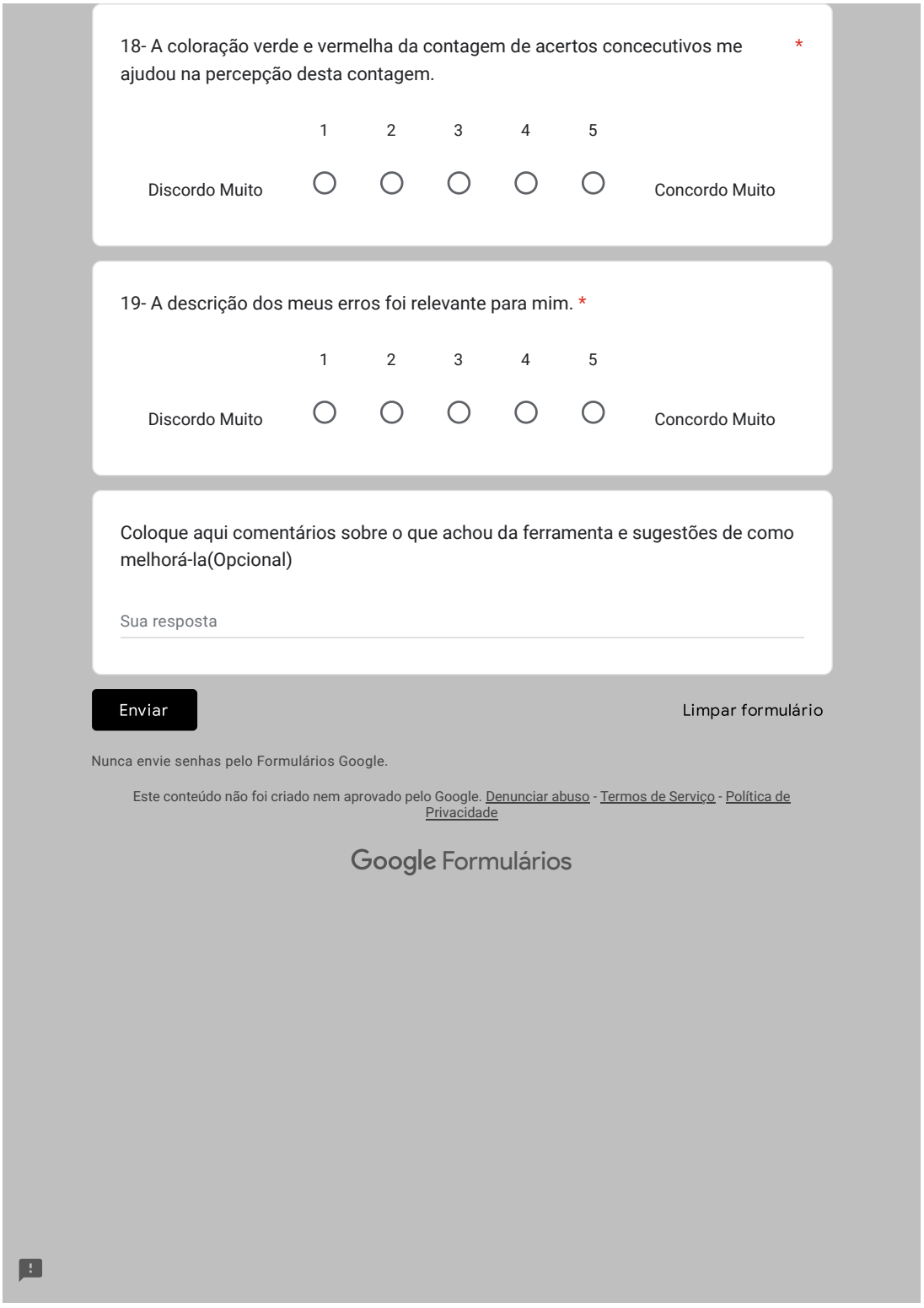


Figura B.5: Parte 5 de 5 do formulário usado para validação.