

Seleção Adaptativa de Questões para Avaliação Rápida de Código em um Ambiente Gamificado de Ensino de Programação

William Martins¹, Kleydson Beckman Barbosa^{1,2}, Allan K. B. S. da Cruz¹,
Sávio Câmara³, Carlos de Salles S. Neto^{1,2}

¹ TeleMídia@MA Lab - Universidade Federal do Maranhão (UFMA)
São Luís, Maranhão, Brasil

² Programa de Pós-Graduação em Ciência da Computação -
Universidade Federal do Maranhão (UFMA) São Luís, Maranhão, Brasil

³ Pontifícia Universidade Católica (PUC)
Rio de Janeiro – Rio de Janeiro – Brasil

{william.valther, kleydson.beckman}@discente.ufma.br,

{saviopatrckc, allankassio}@gmail.com,

carlos.salles@ufma.br

Abstract. *Introductory programming courses often emphasize code writing and automatic assessment, offering fewer opportunities for students to develop critical code evaluation skills. This paper presents and analyzes a gamified mode for rapid code assessment, in which students compare two code snippets under time constraints and immediate feedback. The mode uses an adaptive question selection strategy that balances content diversity and difficulty progression. Based on real usage telemetry, we analyzed 3,774 valid answers from 83 sessions and 29 students. Results indicate an overall accuracy of 89.98% in the rapid code assessment questions. However, this performance is not uniformly distributed: it varies according to programming concept and error type. Lower accuracy occurs in categories that require more conceptual judgment, such as best practices, scope, and comparison errors, suggesting that telemetry can support difficulty diagnosis and adaptive question recalibration.*

Resumo. *O ensino introdutório de programação costuma enfatizar a escrita de código e a avaliação automática, oferecendo menos oportunidades para desenvolver a avaliação crítica de código existente. Este artigo apresenta e analisa um modo gamificado para avaliação rápida de código, no qual estudantes comparam dois trechos de código sob restrição de tempo e com feedback imediato. A proposta utiliza seleção adaptativa de questões, equilibrando diversidade de conteúdo e progressão de dificuldade. Com base em telemetria real de uso, foram analisadas 3.774 respostas válidas, associadas a 83 sessões e 29 estudantes. Os resultados indicam acurácia geral de 89,98% nas respostas às questões de avaliação rápida de código. Entretanto, esse desempenho não se distribui de forma uniforme: há variações por conceito de programação e por tipo de erro avaliado. As menores acurácias ocorrem em categorias que exigem julgamento mais conceitual, como boas práticas, escopo e comparação, sugerindo*

que a telemetria pode apoiar o diagnóstico de dificuldades e a recalibração das questões adaptativas.

1. Introdução

Aprender programação exige não apenas a escrita de código, mas também sua leitura, interpretação e avaliação — competências fundamentais para depuração, refatoração e identificação de *code smells* [Lister et al. 2004, Fowler et al. 1999, Lozano et al. 2024]. Apesar disso, a maioria dos ambientes de ensino foca na submissão de soluções a juízes automáticos, oferecendo poucas oportunidades para o julgamento crítico de códigos de terceiros. Muitas dessas ferramentas priorizam a correção de erros em detrimento de uma orientação formativa e adaptativa [Keuning et al. 2018]. Sob a ótica da Taxonomia de Bloom revisada, essa lacuna negligencia o nível “avaliar”, no qual o estudante realiza julgamentos com base em critérios estruturados [Anderson and Krathwohl 2001].

Para mitigar esse cenário, jogos educacionais e gamificação adaptativa têm sido usados para promover engajamento e persistência [Zhan et al. 2022], buscando o equilíbrio entre desafio e habilidade preconizado pela teoria do fluxo [Chiotaki et al. 2023, Hong et al. 2024, Csikszentmihalyi 1990]. Contudo, o estado de fluxo depende do alinhamento estrito entre mecânicas de jogo, objetivos instrucionais e apoio pedagógico [Stidham 2022, Scaico et al. 2024, Zhong and Zhou 2024, Chou et al. 2023].

Este artigo apresenta e analisa um modo gamificado para treinamento e avaliação rápida de código. A atividade estrutura-se como um mini-game estilo arcade, no qual o estudante responde a questões de escolha binária entre dois trechos de código, sob restrição de tempo e com *feedback* imediato. O diferencial da proposta é a seleção adaptativa de questões, que equilibra a diversidade de conteúdo e a progressão da dificuldade, alinhando-se a sistemas tutores inteligentes [Chrysafiadi and Virvou 2024] e a exercícios de depuração focados em identificação de erros [Pădurean et al. 2025].

A avaliação, de caráter quantitativo e exploratório, baseia-se na telemetria de uso de 29 estudantes, totalizando 3.774 respostas válidas em 83 sessões. Na ausência de grupo controle ou pré/pós-teste, o estudo não estabelece causalidade sobre o ganho de aprendizagem, mas investiga evidências de desempenho, progressão, percepção de dificuldade e padrões de interação dos usuários.

Diante disso, investigam-se três perguntas de pesquisa: **QP1:** quais evidências de desempenho e progressão são observadas durante o uso do modo arcade com seleção adaptativa? **QP2:** a seleção adaptativa favorece uma experiência de desafio equilibrado, considerando dificuldade, conceitos e tipos de erro? **QP3:** quais padrões de telemetria podem indicar progressão ou dificuldade durante o uso? As contribuições do artigo são: (i) uma estratégia adaptativa para seleção de questões de avaliação rápida de código; (ii) sua integração a um modo gamificado com tempo limitado, pontuação, combos e feedback imediato; e (iii) uma análise empírica baseada em telemetria educacional.

2. Fundamentação Teórica e Trabalhos Relacionados

Esta seção apresenta os fundamentos teóricos que orientam o estudo e situa a proposta em relação a trabalhos relacionados. Inicialmente, discute-se a avaliação de código como

competência relevante no ensino de programação. Em seguida, são abordados o papel do feedback automatizado e da adaptação em ambientes educacionais de programação. Por fim, apresentam-se relações entre gamificação, aprendizagem baseada em jogos, fluxo e o posicionamento específico deste trabalho.

2.1. Avaliação de código no ensino de programação

O ensino introdutório de programação é frequentemente organizado em torno da resolução de problemas, da escrita de algoritmos e da submissão de soluções a mecanismos de correção automática. Essas práticas são importantes para desenvolver a aplicação de conceitos, mas não contemplam integralmente outras competências necessárias à formação do programador, como ler, rastrear, comparar e avaliar trechos de código. A leitura e o rastreamento de programas já foram apontados como habilidades essenciais para iniciantes, pois permitem compreender o comportamento de um programa antes da produção de uma nova solução [Lister et al. 2004].

A avaliação de código também se relaciona a práticas como depuração, revisão por pares, manutenção e refatoração. A refatoração pressupõe identificar oportunidades de melhoria em código existente, sem alterar seu comportamento externo [Fowler et al. 1999]. De forma complementar, o reconhecimento de *code smells* pode aproximar o ensino de programação de critérios de qualidade de software, apoiando a análise de más práticas e padrões estruturais problemáticos [Lozano et al. 2024]. Essa competência pode ser compreendida à luz da Taxonomia de Bloom revisada, pois a avaliação de código se aproxima do nível avaliar, no qual o estudante realiza julgamentos com base em critérios [Anderson and Krathwohl 2001]. Trabalhos recentes, como o BugSpotter, reforçam essa direção ao explorar exercícios de depuração nos quais estudantes precisam localizar e compreender erros em trechos de código [Pădurean et al. 2025].

2.2. Feedback automatizado e adaptação em programação

Ambientes educacionais de programação frequentemente utilizam feedback automatizado para apoiar estudantes durante a resolução de exercícios. Uma revisão sistemática sobre geração automática de feedback em programação mostra que muitas ferramentas se concentram em identificar erros nas soluções submetidas, mas ainda enfrentam desafios quanto à orientação formativa, à adaptação ao contexto e ao apoio ao processo de aprendizagem [Keuning et al. 2018]. Nesse cenário, mecanismos adaptativos podem ampliar o papel dos ambientes educacionais ao ajustar atividades e estratégias de apoio com base em evidências do comportamento do estudante.

Em sistemas tutores inteligentes para programação, variáveis como desempenho, tipos de erro, tempo de resolução e necessidade de ajuda podem ser utilizadas para personalizar estratégias instrucionais [Chrysafiadi and Virvou 2024]. Essa perspectiva é relevante para o presente trabalho, pois a seleção de questões de avaliação rápida de código pode considerar não apenas a dificuldade estimada de cada item, mas também o histórico recente do estudante, sua taxa de acerto, o tempo de resposta e os padrões de erro observados.

2.3. Gamificação, aprendizagem baseada em jogos e fluxo

Jogos educacionais e estratégias de gamificação têm sido investigados como formas de favorecer engajamento, motivação e persistência no ensino de programação. Uma meta-análise sobre gamificação em educação em programação indica efeitos positivos, mas

ressalta que os resultados dependem do desenho pedagógico da atividade e da integração adequada dos elementos de jogo aos objetivos de aprendizagem [Zhan et al. 2022]. Revisões recentes sobre aprendizagem adaptativa baseada em jogos também destacam que ambientes gamificados podem se beneficiar de mecanismos capazes de ajustar desafios, conteúdos e trajetórias às necessidades dos estudantes [Chiotaki et al. 2023]. De modo semelhante, estudos sobre gamificação personalizada indicam que diferentes elementos de jogo podem ser adaptados considerando características, desempenho e preferências dos estudantes [Hong et al. 2024].

Um conceito central nessa discussão é o estado de fluxo, entendido como uma experiência de alto envolvimento associada ao equilíbrio entre desafio e habilidade [Csikszentmihalyi 1990]. Revisões sobre fluxo em aprendizagem baseada em jogos mostram que esse construto tem sido relacionado a concentração, clareza de objetivos, feedback imediato, percepção de controle e equilíbrio entre dificuldade e competência [Stidham 2022]. No entanto, estudos recentes indicam que a gamificação não produz fluxo automaticamente: elementos gamificados podem favorecer microexperiências de fluxo, mas a tarefa, o contexto e as características individuais também influenciam a experiência [Scaico et al. 2024]. Pesquisas sobre jogos personalizados em educação em computação também indicam que a personalização pode aumentar o engajamento, mas não necessariamente gerar fluxo sem recursos instrucionais adequados [Zhong and Zhou 2024].

2.4. Posicionamento deste trabalho

Este trabalho investiga um modo gamificado de avaliação rápida de código baseado em questões curtas de escolha binária. Diferentemente de ambientes centrados na escrita completa de programas, a proposta enfatiza a comparação entre trechos de código, o reconhecimento de padrões de erro e a tomada de decisão sob restrição de tempo. Esse desenho segue a recomendação de que mecânicas de jogos educacionais estejam vinculadas aos processos cognitivos esperados, como aplicar, analisar e avaliar [Chou et al. 2023].

O diferencial da proposta está na seleção adaptativa das questões, que busca equilibrar diversidade de conteúdo e progressão de dificuldade. Neste artigo, contudo, a contribuição empírica não consiste em demonstrar causalmente que a seleção adaptativa produz ganho de aprendizagem. A proposta é analisada a partir de evidências observacionais geradas por telemetria real de uso. Assim, o trabalho se posiciona na interseção entre ensino de programação, avaliação de código, gamificação, aprendizagem adaptativa e análise de dados educacionais, buscando compreender quais padrões de desempenho, progressão e dificuldade emergem durante o uso do modo arcade.

3. Modo Gamificado e Seleção Adaptativa de Questões

Esta seção apresenta o modo gamificado de avaliação rápida de código e a estratégia adaptativa utilizada para selecionar questões durante a atividade. O artefato foi desenvolvido como um mini-game educacional em formato arcade, integrado a uma plataforma de ensino de programação. A proposta busca complementar atividades tradicionais de escrita de código, enfatizando leitura crítica, comparação entre alternativas, identificação de inconsistências e tomada de decisão sob restrição de tempo. Esse desenho segue a recomendação de que jogos educacionais articulem mecânicas de jogo a objetivos cog-

nitivos específicos, em vez de apenas adicionarem elementos motivacionais à atividade [Chou et al. 2023, Zhan et al. 2022].

3.1. Visão geral do modo arcade

No modo arcade, o estudante participa de uma sequência de rodadas curtas. Em cada rodada, são apresentados dois trechos de código relacionados a um mesmo conceito de programação, e o estudante deve selecionar qual alternativa está correta ou mais adequada. A atividade utiliza mecânicas como energia, distância percorrida, pontuação, combos, progressão de dificuldade e feedback imediato. A Figura 1 ilustra os elementos visuais do modo arcade.

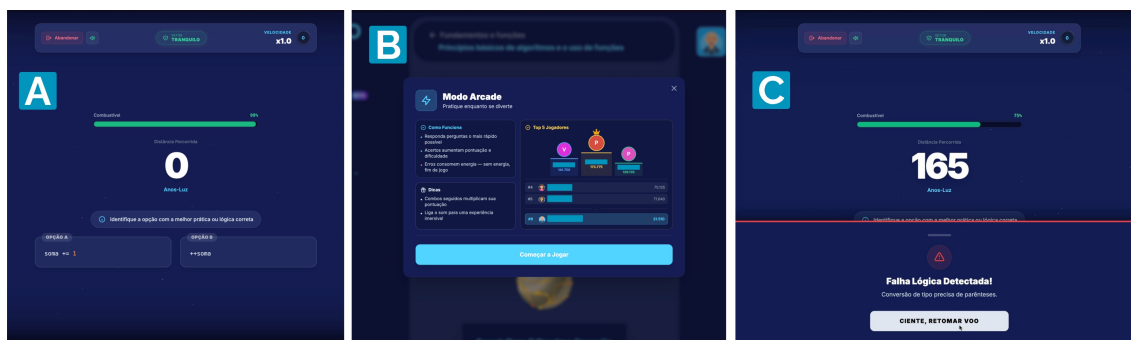


Figura 1. Elementos visuais do Modo Arcade: (a) Início da sessão; (b) Jogabilidade ativa com barra de combustível e comparação de código; (c) Feedback imediato após uma escolha incorreta.

A energia funciona como mecanismo de pressão temporal: à medida que a sessão avança, o estudante precisa responder de forma rápida para permanecer ativo no jogo. Respostas corretas contribuem para a progressão e para a manutenção de combos, enquanto respostas incorretas interrompem a sequência de acertos e afetam o desempenho da sessão.

Do ponto de vista pedagógico, o modo arcade enfatiza a habilidade de avaliar código. Na Taxonomia de Bloom revisada, avaliar envolve realizar julgamentos com base em critérios [Anderson and Krathwohl 2001]. Assim, ao comparar dois trechos de código e decidir qual alternativa está correta ou mais adequada, o estudante mobiliza leitura crítica, análise comparativa e julgamento. Essa proposta complementa exercícios tradicionais de programação, que tendem a privilegiar a produção de soluções completas.

3.2. Estrutura das questões

O banco de questões é composto por templates parametrizados. Cada template possui um conceito de programação, um nível de dificuldade, um tipo de mecânica ou erro e um conteúdo estruturado. Na base analisada, o sistema contém 809 templates ativos, distribuídos em seis conceitos de programação e dez tipos de mecânicas de erro ou avaliação. A Tabela 1 apresenta essas mecânicas.

Essa organização permite registrar não apenas se o estudante acertou ou errou, mas também que tipo de julgamento de código esteve envolvido em cada resposta. Erros sintáticos e de indentação tendem a exigir reconhecimento visual mais direto, enquanto

Tabela 1. Tipos de mecânicas utilizadas nas questões do modo arcade

Mecânica	Descrição
<i>SYNTAX_ERROR</i>	Identificação de erro sintático no trecho de código.
<i>ASSIGNMENT_ERROR</i>	Uso incorreto de atribuição ou atualização de valores.
<i>COMPARISON_ERROR</i>	Uso inadequado de operadores ou expressões de comparação.
<i>LOOP_ERROR</i>	Problemas em estruturas de repetição, limites ou controle do laço.
<i>INDEX_ERROR</i>	Acesso incorreto a posições de vetores ou listas.
<i>LOGIC_ERROR</i>	Falhas de lógica que produzem comportamento incorreto.
<i>TYPE_ERROR</i>	Incompatibilidades ou usos inadequados de tipos de dados.
<i>INDENTATION_ERROR</i>	Erros relacionados à indentação e estrutura visual do código.
<i>BEST_PRACTICE</i>	Comparação entre soluções quanto à clareza ou boa prática.
<i>SCOPE_ERROR</i>	Uso incorreto de escopo de variáveis ou funções.

questões de boas práticas, escopo ou comparação podem demandar avaliação mais conceitual. Essa distinção é importante porque a avaliação de código não se limita à detecção de erros formais, mas envolve também julgamento sobre adequação, legibilidade e qualidade da solução [Fowler et al. 1999, Lozano et al. 2024].

3.3. Telemetria coletada

A cada resposta, o sistema registra informações sobre a interação do estudante com a questão, incluindo identificador da sessão, identificador do template, acerto ou erro, tempo de resposta, alternativa selecionada, energia, combo, distância acumulada e pontuações atribuídas pelos agentes de seleção. Esses dados permitem reconstruir a trajetória do estudante durante a sessão e analisar desempenho, velocidade de resposta, progressão e comportamento adaptativo.

Além das respostas individuais, o sistema mantém um estado agregado do jogador, com indicadores como conhecimento estimado, frustração, engajamento, repetição, dificuldade recente e histórico de questões. O uso de dados de desempenho, tempo e padrões de erro como base para adaptação é coerente com abordagens de sistemas tutores inteligentes em programação [Chrysafiadi and Virvou 2024].

3.4. Estratégia adaptativa de seleção

A seleção das questões busca equilibrar dois objetivos pedagógicos: diversidade de conteúdo e progressão de dificuldade. O primeiro objetivo evita repetição excessiva de conceitos, tipos de erro ou formatos de questão. O segundo busca manter o estudante em uma zona adequada de desafio, evitando tanto a permanência em questões muito fáceis quanto a exposição abrupta a questões excessivamente difíceis.

Esse equilíbrio se relaciona à teoria do fluxo, segundo a qual experiências de alto envolvimento tendem a ocorrer quando há adequação entre desafio e habilidade [Csikszentmihalyi 1990]. Em ambientes educacionais gamificados, esse equilíbrio também depende de clareza de objetivos, feedback imediato e alinhamento entre mecânicas e objetivos instrucionais [Stidham 2022, Scaico et al. 2024, Zhong and Zhou 2024].

Para operacionalizar a seleção adaptativa, o modelo considera dois agentes internos. O Agente A é orientado à diversidade e atribui maior valor a questões que ampliam a variedade da experiência. O Agente B é orientado à progressão de dificuldade e atribui maior valor a questões que promovem avanço pedagógico compatível com o estado

atual do estudante. A escolha final da questão considera as pontuações desses agentes e uma pontuação agregada. Essa formulação se aproxima de abordagens de aprendizagem adaptativa baseada em jogos, nas quais o sistema ajusta conteúdo ou dificuldade a partir de dados de interação do estudante [Chiotaki et al. 2023, Hong et al. 2024].

Formalmente, o estado do estudante no instante t pode ser representado por:

$$s_t = (k_t, f_t, e_t, r_t, d_t, h_t) \quad (1)$$

em que k_t representa a estimativa de conhecimento, f_t o índice de frustração, e_t o índice de engajamento, r_t o índice de repetição, d_t a dificuldade recente e h_t o histórico recente de questões. A partir desse estado, o sistema avalia um conjunto de questões candidatas e calcula uma pontuação para cada uma delas:

$$Score(q) = \lambda U_A(q, s_t) + (1 - \lambda) U_B(q, s_t) \quad (2)$$

em que $U_A(q, s_t)$ representa a utilidade da questão q segundo o critério de diversidade, $U_B(q, s_t)$ representa a utilidade segundo o critério de progressão de dificuldade e λ controla o peso relativo entre os dois objetivos. Na implementação analisada, essas dimensões são registradas como pontuações dos agentes e pontuação final associadas a cada resposta, permitindo investigar como a adaptação influenciou a sequência de questões apresentada.

3.5. Calibração dos templates

Além da seleção em tempo real, o sistema mantém registros de calibração dos templates, como total de tentativas, total de acertos, tempo médio de resposta e taxa de sucesso. Esses dados permitem comparar a dificuldade estimada originalmente com a dificuldade observada empiricamente durante o uso.

Essa calibração é relevante porque a dificuldade pedagógica de uma questão nem sempre corresponde à dificuldade percebida pelos estudantes. Assim, a telemetria gerada pelo modo arcade apoia tanto a análise educacional quanto o refinamento futuro do mecanismo adaptativo.

4. Método

Este estudo adota uma abordagem quantitativa, observacional e exploratória, baseada na análise de telemetria gerada durante o uso do modo arcade em uma plataforma educacional de ensino de programação. A análise não busca estabelecer causalidade entre a seleção adaptativa e a aprendizagem dos estudantes, pois não houve grupo controle nem aplicação de pré e pós-teste. O objetivo é investigar evidências de desempenho, progressão e dificuldade a partir dos registros produzidos durante as sessões de jogo.

4.1. Base de dados

Os dados foram extraídos de um banco PostgreSQL com registros do modo arcade entre 20/04/2026 e 27/05/2026. Foram consideradas apenas as tabelas relacionadas à atividade gamificada, incluindo sessões, respostas, templates de questões, conceitos, estados agregados dos jogadores e calibração dos templates. Informações pessoais, como nome,

e-mail e credenciais, foram excluídas da análise, e os estudantes foram tratados por identificadores internos anonimizados.

A base contém 101 sessões registradas, das quais 83 possuem ao menos uma resposta associada. No total, foram analisadas 3.774 respostas produzidas por 29 estudantes. O banco de questões contém 809 templates ativos, organizados em seis conceitos de programação e dez tipos de mecânicas de erro ou avaliação.

As sessões possuem três estados: *ACTIVE*, *COMPLETED* e *ABORTED*. Como o modo arcade possui dinâmica de sobrevivência, o status *ABORTED* não foi interpretado automaticamente como abandono pedagógico. Para as análises no nível de resposta, foram consideradas todas as respostas válidas associadas a sessões existentes; sessões *ACTIVE* foram desconsideradas apenas em análises dependentes do encerramento da sessão.

4.2. Variáveis e métricas

A unidade principal de análise foi a resposta individual, associada a um estudante, uma sessão e um template de questão. A partir dessa unidade, foram calculadas métricas agregadas por estudante, sessão, dificuldade, conceito e tipo de mecânica. As principais variáveis utilizadas foram: *studentId*, *sessionId*, *templateId*, *isCorrect*, *timeTakenMs*, *difficultyLevel*, *mechanicType*, *conceptId*, *comboAtMoment*, *distanceAtMoment*, *fuelAtMoment*, *agentAScore*, *agentBScore* e *finalScore*.

A principal métrica de desempenho foi a acurácia, definida neste estudo como a proporção de respostas corretas nas questões de avaliação rápida de código. Ela foi calculada pela razão entre o número de respostas corretas e o total de respostas em cada agrupamento:

$$Acurácia = \frac{respostasCorretas}{totalDeRespostas} \quad (3)$$

A acurácia foi calculada de forma geral e também por dificuldade, conceito, tipo de mecânica, sessão e estudante. O tempo de resposta foi analisado em segundos, considerando média, mediana e percentis. Também foram observados indicadores de progressão, como número de questões por sessão, distância alcançada, combo e evolução do desempenho ao longo das sessões. Para investigar a atuação da seleção adaptativa, foram analisadas as pontuações dos agentes de diversidade e progressão, bem como a pontuação final associada às questões selecionadas.

4.3. Procedimento de análise

O procedimento foi organizado em quatro etapas. Primeiro, foram selecionadas as tabelas relacionadas ao modo arcade e removidos os campos pessoais não necessários ao estudo. Segundo, as respostas foram vinculadas aos respectivos templates, conceitos, tipos de mecânica e sessões. Terceiro, foram calculadas estatísticas descritivas da base, incluindo quantidade de estudantes, sessões, respostas, templates e distribuição temporal dos registros. Por fim, foram analisados desempenho e tempo de resposta por dificuldade, conceito e mecânica, além de indicadores de progressão e comportamento adaptativo.

As análises foram conduzidas de forma descritiva e exploratória. Essa escolha é adequada ao estágio atual da pesquisa, pois a base representa dados naturais de uso

do sistema, sem delineamento experimental controlado. Assim, os resultados devem ser interpretados como evidências observacionais sobre o comportamento dos estudantes no modo arcade, e não como prova causal de ganho de aprendizagem.

5. Resultados e Discussão

Esta seção apresenta os resultados da análise da telemetria do modo arcade. Como o estudo é observacional, sem grupo controle ou pré/pós-teste, os achados são interpretados como evidências de desempenho, progressão e dificuldade durante o uso do sistema, e não como prova causal de aprendizagem.

5.1. Desempenho geral

A base analisada contém 3.774 respostas válidas, associadas a 83 sessões com interação e 29 estudantes com respostas registradas. A acurácia geral foi de 89,98%, com tempo médio de resposta de 3,27 segundos e mediana de 2,16 segundos. O percentil 95 do tempo de resposta foi de 9,13 segundos, indicando que a maior parte das decisões ocorreu em poucos segundos, o que é coerente com a proposta de avaliação rápida de código.

Cada sessão com respostas teve, em média, 45,47 questões respondidas, com mediana de 34 e máximo de 153 questões em uma única sessão. Considerando os estudantes, a média foi de 130,14 respostas por estudante, com mediana de 73. A acurácia média por estudante foi de 86,29%, com mediana de 89,19%. Esses dados mostram que o modo arcade gerou um volume expressivo de microinterações avaliativas, permitindo observar padrões de desempenho em diferentes níveis de agregação.

Tabela 2. Indicadores gerais da base analisada

Indicador	Valor
Respostas válidas	3.774
Sessões com respostas	83
Estudantes com respostas	29
Acurácia geral	89,98%
Tempo médio de resposta	3,27 s
Tempo mediano de resposta	2,16 s
Percentil 95 do tempo de resposta	9,13 s
Média de respostas por sessão	45,47
Mediana de respostas por sessão	34
Máximo de respostas em uma sessão	153

5.2. Desempenho por dificuldade

A Tabela 3 apresenta o desempenho por nível de dificuldade. Todos os níveis tiveram acurácia acima de 87%. No entanto, a acurácia não cresceu ou diminuiu de forma linear: o nível 2 apresentou o menor desempenho, enquanto o nível 4 apresentou o maior.

A maior acurácia no nível 4 sugere duas interpretações possíveis. A primeira é que a seleção adaptativa pode ter direcionado questões mais difíceis a estudantes com melhor desempenho. A segunda é que parte dos templates classificados como difíceis pode estar superestimada, exigindo recalibração. Esse resultado reforça a importância de revisar empiricamente a dificuldade dos itens a partir dos dados de uso.

Tabela 3. Desempenho por nível de dificuldade

Dificuldade	Respostas	Acurácia	Tempo médio
1	372	90,05%	2,25 s
2	1.054	87,57%	4,18 s
3	1.226	88,99%	3,41 s
4	1.122	93,32%	2,60 s

5.3. Desempenho por tipo de mecânica

A Tabela 4 mostra a acurácia por tipo de mecânica ou erro. As maiores acurácias ocorreram em *INDENTATION_ERROR* e *SYNTAX_ERROR*, enquanto os menores desempenhos ocorreram em *BEST_PRACTICE*, *COMPARISON_ERROR* e *SCOPE_ERROR*.

Tabela 4. Desempenho por tipo de mecânica

Mecânica	Respostas	Acurácia	Tempo médio
<i>BEST_PRACTICE</i>	428	71,26%	3,65 s
<i>COMPARISON_ERROR</i>	166	81,93%	4,51 s
<i>SCOPE_ERROR</i>	172	81,98%	4,71 s
<i>INDEX_ERROR</i>	253	88,54%	3,11 s
<i>LOGIC_ERROR</i>	320	89,69%	4,26 s
<i>TYPE_ERROR</i>	431	91,65%	3,39 s
<i>ASSIGNMENT_ERROR</i>	237	91,98%	2,54 s
<i>LOOP_ERROR</i>	427	93,21%	2,99 s
<i>SYNTAX_ERROR</i>	974	96,10%	2,69 s
<i>INDENTATION_ERROR</i>	366	97,27%	3,01 s

Esses resultados indicam que erros mais visuais ou formais, como sintaxe e indentação, foram identificados com maior facilidade. Por outro lado, categorias como boas práticas, escopo e comparação exigem julgamento mais conceitual sobre clareza, contexto e comportamento lógico. O menor desempenho em *BEST_PRACTICE* é especialmente relevante, pois esse tipo de questão exige avaliar qual alternativa é mais adequada, e não apenas identificar se o código executa corretamente. Isso aproxima a atividade do nível *avaliar* da Taxonomia de Bloom revisada [Anderson and Krathwohl 2001].

5.4. Desempenho por conceito

A análise por conceito também revelou diferenças importantes. Conforme a Tabela 5, os menores desempenhos ocorreram em *Vetores II* e *Comando IF*, enquanto os maiores ocorreram em *Vetores I* e *Laços de repetição*.

Tabela 5. Desempenho por conceito de programação

Conceito	Respostas	Acurácia	Tempo médio
<i>Vetores II</i>	225	76,00%	3,79 s
<i>Comando IF</i>	353	84,14%	3,71 s
Fundamentos e funções	1.301	88,39%	3,39 s
Expressões relacionais	419	89,74%	4,32 s
Laços de repetição	980	94,39%	2,83 s
<i>Vetores I</i>	496	96,17%	2,39 s

A menor acurácia em *Vetores II* sugere maior dificuldade em questões que envolvem manipulação mais complexa de listas, índices ou padrões avançados de iteração. Já o

desempenho em *Comando IF* indica que condicionais ainda podem representar uma fonte relevante de erro, especialmente quando combinadas a comparações ou lógica booleana. Esses achados mostram que a telemetria pode apoiar diagnósticos mais específicos do que uma nota agregada, indicando conceitos e padrões de erro que exigem reforço. Essa possibilidade dialoga com sistemas tutores inteligentes, nos quais dados de desempenho, tempo e tipos de erro podem orientar estratégias instrucionais [Chrysafiadi and Virvou 2024].

5.5. Evidências de progressão

Para investigar indícios de progressão, as sessões foram analisadas em ordem cronológica. Ao dividir as sessões com respostas em quatro quartis temporais, observou-se aumento da acurácia ponderada de 84,59% no primeiro quartil para 92,61% no terceiro, com leve queda para 91,00% no quarto. O tempo médio ponderado caiu de 3,78 segundos no primeiro quartil para 2,83 segundos no terceiro, voltando a 3,56 segundos no quarto.

Tabela 6. Evolução agregada ao longo dos quartis temporais

Quartil	Respostas	Acurácia ponderada	Tempo médio
Q1	649	84,59%	3,78 s
Q2	920	89,46%	3,08 s
Q3	1.083	92,61%	2,83 s
Q4	1.122	91,00%	3,56 s

A análise por estudante reforça esse padrão. Entre os 18 estudantes com duas ou mais sessões, 15 apresentaram aumento de acurácia entre a primeira e a última sessão registrada. A diferença média foi de 17,68 pontos percentuais, com mediana de 14,57 pontos percentuais. Além disso, 17 dos 18 estudantes reduziram o tempo médio de resposta, com redução média de 3,76 segundos. Considerando apenas estudantes cujas primeira e última sessões tiveram pelo menos cinco respostas, 12 de 14 apresentaram aumento de acurácia, e todos reduziram o tempo médio de resposta.

Esses resultados sugerem familiarização progressiva com a dinâmica do modo arcade e possível ganho de fluência na avaliação rápida de código. No entanto, como não há pré/pós-teste ou grupo controle, não é possível afirmar que a melhora decorre exclusivamente da seleção adaptativa.

5.6. Comportamento da seleção adaptativa

A análise das pontuações dos agentes indica que o agente de progressão teve maior influência sobre a pontuação final. A pontuação média do Agente A, relacionado à diversidade, foi de -0,28, enquanto a do Agente B, relacionado à progressão, foi de 12,63. A pontuação final média foi de 9,18. A correlação entre a pontuação do Agente B e a pontuação final foi praticamente perfeita ($r \approx 0,99$), enquanto a correlação entre a pontuação do Agente A e a pontuação final foi negativa e fraca ($r \approx -0,26$).

Esse resultado indica que o mecanismo favoreceu, em maior grau, a progressão de dificuldade. Isso é coerente com a proposta de avanço pedagógico, mas também aponta a necessidade de reavaliar os pesos atribuídos à diversidade e à dificuldade. Caso o objetivo seja equilibrar variedade e desafio, a arquitetura deve evitar que um dos agentes controle de forma excessiva a seleção final. Esse cuidado é relevante porque, em ambientes gamificados, o engajamento depende não apenas da dificuldade, mas também de

Tabela 7. Indicadores do comportamento adaptativo

Indicador	Valor
Pontuação média do Agente A	-0,28
Pontuação média do Agente B	12,63
Pontuação final média	9,18
Correlação entre Agente B e pontuação final	$r \approx 0,99$
Correlação entre Agente A e pontuação final	$r \approx -0,26$
Correlação entre pontuação final e dificuldade	$r \approx 0,24$
Correlação entre pontuação final e acerto	próxima de zero

variedade, feedback e alinhamento pedagógico [Csikszentmihalyi 1990, Stidham 2022, Scaico et al. 2024].

5.7. Síntese das perguntas de pesquisa

Em relação à **QP1**, os dados indicam alto desempenho geral nas tarefas de avaliação rápida de código, com acurácia de 89,98%, e indícios de progressão entre estudantes com múltiplas sessões. Quanto à **QP2**, o sistema apresentou questões em diferentes níveis de dificuldade, conceitos e mecânicas, embora a maior influência do Agente B indique que o equilíbrio entre progressão e diversidade ainda pode ser refinado. Em relação à **QP3**, a telemetria permitiu identificar padrões de dificuldade, especialmente em *BEST_PRACTICE*, *SCOPE_ERROR*, *COMPARISON_ERROR*, *Vetores II* e *Comando IF*.

De modo geral, os resultados sugerem que a avaliação rápida de código pode complementar atividades tradicionais de programação, favorecendo leitura crítica, comparação entre alternativas e identificação de erros. Entretanto, os achados devem ser interpretados como evidências observacionais, pois a ausência de grupo controle e de pré/pós-teste impede afirmar causalmente que a seleção adaptativa produziu aprendizagem. Além disso, métricas como acurácia e tempo de resposta são indicadores indiretos, os dados vêm de uma única plataforma e a maior acurácia em questões de nível 4 indica necessidade de recalibração empírica dos templates.

6. Conclusão

Este artigo apresenta e analisa um modo gamificado para avaliação rápida de código em um ambiente educacional de ensino de programação. A proposta busca complementar práticas tradicionais de escrita de programas ao enfatizar leitura crítica, comparação entre trechos de código, identificação de erros e tomada de decisão sob restrição de tempo.

A partir da telemetria real de uso, são analisadas 3.774 respostas válidas, associadas a 83 sessões e 29 estudantes. Os resultados indicam acurácia geral de 89,98% nas respostas às questões de avaliação rápida de código e tempo médio de resposta de 3,27 segundos. Entretanto, esse desempenho não se distribui de forma uniforme. As análises por mecânica e conceito mostram maior facilidade em erros formais ou visuais, como *SYNTAX_ERROR* e *INDENTATION_ERROR*, e maior dificuldade em categorias que exigem julgamento mais conceitual, como *BEST_PRACTICE*, *SCOPE_ERROR* e *COMPARISON_ERROR*, além dos conceitos *Vetores II* e *Comando IF*.

Também são observados indícios de progressão entre estudantes com múltiplas sessões, com aumento de acurácia e redução do tempo médio de resposta. No entanto,

como o estudo não utiliza grupo controle nem pré/pós-teste, os resultados devem ser interpretados como evidências observacionais, e não como comprovação causal de aprendizagem. A análise dos agentes adaptativos indica predominância do agente de progressão, apontando a necessidade de refinamento no equilíbrio entre diversidade e dificuldade.

De modo geral, os achados indicam que a telemetria do modo arcade oferece subsídios para compreender como estudantes interagem com questões de avaliação de código e quais tipos de erro ou conceito apresentam maior dificuldade.

Como trabalhos futuros, pretende-se realizar estudos controlados, comparar versões adaptativas e não adaptativas do modo arcade, recalibrar empiricamente os templates de questões e investigar a relação entre desempenho no jogo e desempenho em atividades tradicionais de programação.

Referências

- Anderson, L. W. and Krathwohl, D. R., editors (2001). *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives*. Longman, New York, NY.
- Chiotaki, D., Pouloupoulos, V., and Karpouzis, K. (2023). Adaptive game-based learning in education: A systematic review. *Frontiers in Computer Science*, 5:1062350.
- Chou, Y.-S., Hou, H.-T., Chang, K.-E., and Su, C.-L. (2023). Designing cognitive-based game mechanisms for mobile educational games to promote cognitive thinking: an analysis of flow state and game-based learning behavioral patterns. *Interactive Learning Environments*, 31(5):3285–3302.
- Chrysafiadi, K. and Virvou, M. (2024). Perfusit: Personalized fuzzy logic strategies for intelligent tutoring of programming. *Electronics*, 13(23):4827.
- Csikszentmihalyi, M. (1990). *Flow: The Psychology of Optimal Experience*. Harper & Row, New York, NY.
- Fowler, M., Beck, K., Brant, J., Opdyke, W., and Roberts, D. (1999). *Refactoring: Improving the Design of Existing Code*. Addison-Wesley Professional, Reading, MA.
- Hong, Y., Saab, N., and Admiraal, W. (2024). Approaches and game elements used to tailor digital gamification for learning: A systematic literature review. *Computers & Education*, 212:105000.
- Keuning, H., Jeurig, J., and Heeren, B. (2018). A systematic literature review of automated feedback generation for programming exercises. *ACM Trans. Comput. Educ.*, 19(1).
- Lister, R., Adams, E. S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., McCartney, R., Moström, J. E., Sanders, K., Seppälä, O., Simon, B., and Thomas, L. (2004). A multi-national study of reading and tracing skills in novice programmers. ITiCSE-WGR '04, page 119–150, New York, NY, USA. Association for Computing Machinery.
- Lozano, R., Prabhu, R. S., and Cheon, Y. (2024). Code smells for assessing and improving students' coding skills and practices. In *Innovations in Computing Research*, volume 1058 of *Lecture Notes in Networks and Systems*, pages 617–631. Springer, Cham.

- Pădurean, V.-A., Denny, P., and Singla, A. (2025). BugSpotter: Automated generation of code debugging exercises. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1*, SIGCSE TS 2025, pages 896–902, New York, NY, USA. Association for Computing Machinery.
- Scaico, P. D., Oliveira, W., Hamari, J., MilHosseini, S., and Brambilla, A. (2024). Understanding undergraduate students' flow state in gamified and non-gamified educational systems: A qualitative case study. In *2024 IEEE Frontiers in Education Conference (FIE)*, pages 1–9. IEEE.
- Stidham, S. F. (2022). An integrative review of the conceptualization and assessment of the learner flow experience in the digital game-based learning environment between 2011 and 2021.
- Zhan, Z., He, L., Tong, Y., Liang, X., Guo, S., and Lan, X. (2022). The effectiveness of gamification in programming education: Evidence from a meta-analysis. *Computers and Education: Artificial Intelligence*, 3:100096.
- Zhong, L. and Zhou, J. (2024). Personalized games for computer science education in higher education: the effect of personalization feature on students' engagement and flow state. *Journal of Computing in Higher Education*, pages 1–28.