UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

CESAR BATISTA

FELIPE AVELINO PANTOJA RÊGO

IMPLEMENTAÇÃO E AVALIAÇÃO DE UMA VERSÃO INICIAL DO QA+: UM JOGO PARA O ENSINO DE TESTE E QUALIDADE DE SOFTWARE

CURITIBA

2024

CESAR BATISTA
FELIPE AVELINO PANTOJA RÊGO

IMPLEMENTAÇÃO E AVALIAÇÃO DE UMA VERSÃO INICIAL DO QA+: UM JOGO PARA O ENSINO DE TESTE E QUALIDADE DE SOFTWARE

Implementation and Evaluation of an Initial Version of QA+: A Game for Teaching Software Testing and Quality

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia da Computação do Curso de Bacharelado em Engenharia da Computação da Universidade Tecnológica Federal do Paraná.
Orientadora: Profª. Drª. Maria Claudia Figueiredo Pereira Emer
Coorientador: Prof. Me. Vladimir Belinski

CURITIBA
2024

CESAR BATISTA
FELIPE AVELINO PANTOJA RÊGO

IMPLEMENTAÇÃO E AVALIAÇÃO DE UMA VERSÃO INICIAL DO QA+: UM JOGO PARA O ENSINO DE TESTE E QUALIDADE DE SOFTWARE

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia da Computação do Curso de Bacharelado em Engenharia da Computação da Universidade Tecnológica Federal do Paraná.

Data de aprovação: 25/junho/2024

Maria Claudia Figueiredo Pereira Emer
Doutora
Universidade Tecnológica Federal do Paraná

Adolfo Gustavo Serra Seca Neto
Doutor
Universidade Tecnológica Federal do Paraná

Juliano Mourão Vieira
Mestre
Universidade Tecnológica Federal do Paraná

CURITIBA
2024

AGRADECIMENTOS

Gostaria de agradecer aos meus pais, principalmente aos meus irmãos, Fernando e Gabriel, que me ajudaram neste projeto e que estão nos meus pensamentos em todos os momentos.

RESUMO

Contexto: A atividade de teste é crucial para a garantia da qualidade de produtos de software. No entanto, alunos de nível superior frequentemente apresentam lacunas de conhecimento e falta de motivação para aprender sobre teste de software, resultando em escassez de mão de obra especializada sobre esse assunto na indústria. Nesse cenário, se mostra necessário apoiar o ensino de teste e qualidade. A gamificação, que se baseia na utilização de elementos de jogos em contextos que não são de jogos, pode consistir em uma solução para os desafios mencionados. Ao tornar o processo de aprendizado mais interativo e envolvente, a gamificação pode aumentar a motivação dos alunos e facilitar a compreensão de tópicos complexos. **Objetivo:** Tendo conhecimento desse contexto, neste trabalho objetivamos desenvolver e avaliar uma versão inicial do QA+, um jogo que emprega elementos de gamificação para apoiar o ensino de teste e qualidade de software no nível superior acadêmico. **Método:** O desenvolvimento do jogo foi realizado por meio de um processo que englobou a delimitação de seu escopo, sua modelagem e codificação utilizando tecnologias web como React.js, Typescript, Java 21 e Spring Boot. Para a avaliação do QA+ conduzimos um estudo de caso com alunos de graduação, por meio do qual foram coletadas com o apoio de um questionário online as percepções dos estudantes acerca de aspectos como usabilidade do jogo e experiência do jogador. Sobre os dados coletados conduzimos análises quantitativas. **Resultados:** A partir da análise dos dados coletados de nove estudantes evidenciamos que o QA+ foi considerado esteticamente atraente e de fácil compreensão e uso, tendo igualmente causado nos discentes sentimentos como confiança, satisfação e diversão, aspectos essenciais para a manutenção de seu engajamento e interesse em relação ao aprendizado. Ainda, os dados coletados mostraram que o jogo foi apontado como relevante pelos alunos, lhes tendo auxiliado a relembrar, compreender e aplicar conceitos de teste de software. Ao mesmo tempo, foram evidenciadas oportunidades de melhoria para o jogo, tais como a inclusão de funcionalidades de personalização e atividades mais desafiadoras, bem como recursos para capturar ainda mais a atenção dos discentes e lhes auxiliar a não cometer erros durante a utilização do jogo. **Conclusão:** Este trabalho contribuiu com o rol de estudos que buscam aprimorar a educação em teste e qualidade de software. Por meio dele foram evidenciados indícios positivos em relação à aplicabilidade do QA+ como recurso educacional para apoiar o ensino de teste e qualidade de software no nível superior acadêmico. Futuras melhorias no jogo, incluindo a implementação de funcionalidades adicionais, podem aumentar ainda mais sua eficácia e atratividade.

Palavras-chave: teste de software; qualidade de software; gamificação; ferramenta educacional; desenvolvimento de software.

ABSTRACT

Context: Testing activity is crucial for ensuring the quality of software products. However, higher education students often have knowledge gaps and lack motivation to learn about software testing, resulting in a shortage of specialized labor in this field in the industry. In this context, it is necessary to support the teaching of testing and quality. Gamification, which involves using game elements in non-game contexts, can be a solution to the mentioned challenges. By making the learning process more interactive and engaging, gamification can increase students' motivation and facilitate the understanding of complex topics. **Objective:** Given this context, in this work we aimed to develop and evaluate an initial version of QA+, a game that employs gamification elements to support the teaching of software testing and quality at the higher education. **Method:** The development of the game was carried out through a process that included defining its scope, modeling the game, and coding it using web technologies such as React.js, Typescript, Java 21, and Spring Boot. To evaluate QA+, we conducted a case study with undergraduate students, collecting their perceptions of the game's usability and player experience through an online questionnaire. We performed quantitative analyses on the collected data. **Results:** From the analysis of data collected from nine students, we found that QA+ was considered aesthetically appealing and easy to understand and use, causing feelings of confidence, satisfaction, and fun among the students, essential aspects for maintaining their engagement and interest in learning. Furthermore, the collected data showed that the game was deemed relevant by the students, helping them recall, understand, and apply software testing concepts. However, opportunities for improvement were also identified, such as the inclusion of customization features and more challenging activities, as well as resources to better capture students' attention and help them to avoid making mistakes while using the game. **Conclusion:** This work contributes to the body of studies that aim to improve education on software testing and quality. It provided positive indications regarding the applicability of QA+ as an educational resource to support the teaching of software testing and quality at the higher education level. Future improvements to the game, including the implementation of additional features, may further increase its effectiveness and attractiveness.

Keywords: software testing; software quality; gamification; educational tool; software development.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTAGEM DE CÓDIGOS FONTE

LISTA DE ABREVIATURAS E SIGLAS

Siglas

API	*Application Programming Interface*
DER	Diagrama Entidade-Relacionamento
GGCC	*Gamifying Graph Coverage Criteria*
J2EE	Java 2 *Enterprise Edition*
JPA	*Java Persistence API*
JSP	*Jakarta Server Pages*
MEEGA	*Model for the Evaluation of Educational GAmes*
MVC	*Model-View-Controller*
MVP	*Minimum Viable Product*
RESTful	*Representational State Transfer*
SBTM	*Session-Based Test Management*
TCLE	Termo de Consentimento Livre e Esclarecido
TE	Teste Exploratório
UTFPR	Universidade Tecnológica Federal do Paraná
V&V	Verificação e Validação
VV&T	Validação, Verificação e Teste

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta uma visão geral do trabalho, contemplando seu contexto, motivação e justificativa, sua questão de pesquisa, objetivo, contribuições e estrutura.

1.1 Contexto, motivação e justificativa

O desenvolvimento de sistemas é um processo complexo e, assim, suscetível a erros. Nesse contexto, a atividade de teste de software é muito importante, uma vez que objetiva a identificação de erros, possibilitando, então, a correção desses e o aprimoramento da qualidade dos produtos de software (Delamaro; Maldonado; Jino, 2016).

Apesar da importância do teste de software, alunos de nível superior costumam apresentar lacunas de conhecimento em relação a essa atividade e falta de motivação para aprendê-la (Garousi *et al.*, 2020; Melo *et al.*, 2020). Ainda, como consequência, na indústria há uma falta de mão de obra especializada em relação a teste de software (Valle; Barbosa; Maldonado, 2015).

Diante disso, nos últimos anos estudos têm sido conduzidos visando encontrar meios para aprimorar a educação em teste. Um dos meios sugeridos é a utilização de jogos sérios e gamificação, alternativas já empregadas em outras áreas de conhecimento e/ou níveis de ensino, a exemplo da educação básica (Fadel *et al.*, 2014).

Segundo Zichermann e Cunningham (2011), a gamificação explora os níveis de engajamento do indivíduo para a resolução de problemas. Ela se baseia na utilização de elementos de jogos — isto é, desafios, metas, recompensas, etc (Fadel *et al.*, 2014) — em atividades que não são de jogos. Um dos objetivos da gamificação no ensino é aumentar a motivação dos alunos, contribuindo para o engajamento desses em relação ao aprendizado (Fadel *et al.*, 2014). Ainda, ao se considerar jogos digitais, tem se percebido que a gamificação pode ser utilizada para apoiar o ensino de tópicos que, em modelos tradicionais[1] de ensino, os alunos teriam mais dificuldade para se sentir motivados a estudar.

Tendo conhecimento desse contexto, neste trabalho buscamos apoiar o ensino de teste de software no nível superior acadêmico por meio da implementação e avaliação de uma versão inicial do jogo QA+, um jogo sério e educacional proposto por Belinski (2021) e que utiliza elementos de gamificação para o ensino de teste e qualidade de software.

1.2 Questão de pesquisa

Definimos para este trabalho a seguinte questão de pesquisa:

[1] O ensino tradicional é comumente definido como aquele no qual o aluno passivamente recebe informações de um instrutor (Prince, 2004).

QP1: Considerando a percepção de alunos de nível superior acadêmico, qual é a aplicabilidade[2] de uma versão inicial do jogo QA+ como recurso educacional para apoiar o ensino de teste e qualidade de software?

1.3 Objetivo

Desenvolver e avaliar uma versão inicial do QA+, um jogo para apoiar o ensino de teste e qualidade de software no nível superior acadêmico.

1.4 Contribuições

Este trabalho contribuiu para a resolução ou, ao menos, redução dos problemas apresentados na Seção 1.1 por meio de:

- Desenvolvimento e disponibilização de uma versão inicial do QA+, um jogo para apoiar o ensino de teste e qualidade de software no nível superior acadêmico.
- Evidências científicas acerca da aplicabilidade do jogo desenvolvido (considerando percepções de alunos de nível superior acadêmico).

1.5 Estrutura do trabalho

Este capítulo apresentou o contexto, motivação e justificativa do trabalho, sua questão de pesquisa, objetivo, contribuições e estrutura. Por sua vez, o Capítulo 2 expõe o referencial teórico da pesquisa e o Capítulo 3 os trabalhos relacionados. Em sequência, o Capítulo 4 descreve o método de pesquisa empregado e o Capítulo 5 expõe e discute os resultados do trabalho. Por fim, o Capítulo 6 é destinado à conclusão da pesquisa, sendo seguido das referências e apêndices.

[2] Neste estudo o termo “aplicabilidade” diz respeito à viabilidade e impacto da utilização do jogo como recurso para apoiar o ensino de teste e qualidade de software. A aplicabilidade do QA+ foi aferida através da avaliação dos fatores de qualidade de jogos educacionais conhecidos como usabilidade (estética, aprendizibilidade, operabilidade, acessibilidade e proteção contra erros do usuário) e experiência do jogador (confiança, desafio, satisfação, diversão, atenção focada, relevância e aprendizagem percebida), conforme definições de Petri, Wangenheim e Borgatto (2019). Maiores detalhes serão apresentadas nas próximas seções deste trabalho.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta o referencial teórico do trabalho por meio da definição dos termos e conceitos necessários ao entendimento da pesquisa.

2.1 Qualidade e teste de software

A qualidade é um aspecto muito importante no contexto de desenvolvimento de software. Segundo Delamaro, Maldonado e Jino (2016), a garantia da qualidade costuma englobar uma série de atividades conhecidas como Validação, Verificação e Teste (VV&T), as quais objetivam garantir que tanto o processo de construção do software seja adequado quanto o produto em si esteja em conformidade com suas especificações. As atividades de VV&T podem ser classificadas como estáticas ou dinâmicas. As estáticas são aquelas que não requerem a execução ou existência de um programa ou modelo para serem realizadas. Já as dinâmicas exigem a execução do programa ou modelo, sendo o teste um exemplo desse tipo de atividade.

Conforme Delamaro, Maldonado e Jino (2016), o teste envolve a execução de um programa ou modelo utilizando dados de entrada específicos e a conferência de seu comportamento (se é igual ao esperado). Quando o comportamento evidenciado é distinto do esperado diz-se que um defeito foi revelado. Complementarmente, Myers, Badgett e Sandler (2012) mencionam que o objetivo do teste não é mostrar que um programa está correto, mas sim revelar a presença de defeitos caso esses existam. Logo, a atividade de teste é muito importante, pois a identificação de defeitos em um programa/modelo permite com que esses sejam analisados e corrigidos, evitando ou, ao menos, reduzindo prejuízos aos seus criadores e utilizadores.

2.1.1 Terminologia

De acordo com Delamaro, Maldonado e Jino (2016), um defeito (*fault*) corresponde a um passo, processo ou definição de dados incorreta. Por sua vez, engano (*mistake*) é a ação humana que produz o defeito (um erro do programador, por exemplo). Já o erro (*error*) é um estado inconsistente ou inesperado do programa, enquanto uma falha (*failure*) é um desvio visível de sua especificação. Logo, pode-se dizer que um engano produz um defeito, que pode ocasionar um erro durante uma execução do programa. Se esse erro for propagado até a saída do programa ele ocasionará uma falha (resultado distinto do esperado).

Também segundo Delamaro, Maldonado e Jino (2016), um cenário de teste é comumente composto por: (a) modelo/programa P; (b) domínio de entrada $D(P)$ – conjunto dos possíveis valores que podem ser utilizados para executar P; (c) domínio de saída $D(S)$ – conjunto dos possíveis resultados produzidos por P; (d) especificação $S(P)$ do programa P; (e) dado de teste d – um elemento de $D(P)$; (f) caso de teste $(d, S(d))$ – par formado por um dado

de teste d e pelo resultado esperado para a execução de P com d, denotado por $S(d)$; (g) conjunto de teste ou conjunto de casos de teste T – conjunto dos casos de teste utilizados durante uma determinada atividade de teste; e (h) oráculo – instrumento (um testador ou ferramenta, por exemplo) que decide se a saída obtida a partir de uma determinada execução de P é igual à saída esperada.

2.1.2 Cenário típico da atividade de teste

Para Delamaro, Maldonado e Jino (2016), em um cenário típico da atividade de teste, uma vez definido um conjunto de casos de teste T a partir de $D(P)$, executa-se o programa em teste P com T e conferem-se os resultados obtidos. A conferência de cada execução é realizada por um oráculo (O), que pode utilizar para tal a especificação do programa $S(P)$ ou outro artefato que defina o comportamento de P. Caso o resultado da execução de P seja igual ao esperado, então nenhum erro foi identificado. Todavia, se para algum caso de teste o resultado obtido for diferente do esperado, então um defeito foi revelado.

2.1.3 Fases da atividade de teste

A atividade de teste costuma ser dividida em fases, as quais representam momentos distintos no ciclo de vida do desenvolvimento de um software e, portanto, possuem objetivos específicos. De acordo com Delamaro, Maldonado e Jino (2016), são fases típicas da atividade de teste: (1) teste de unidade; (2) teste de integração; e (3) teste de sistema.

O teste de unidade visa testar as subpartes de um software, conhecidas como unidades, que são as menores partes constituintes do sistema (Febbraro *et al.*, 2013). Essas unidades podem ser funções, procedimentos, métodos ou classes. Geralmente, os erros encontrados nessa fase incluem cálculos feitos com fórmulas incorretas, algoritmos inadequados, troca de passos, estruturas faltantes, erros de digitação, entre outros. Os testes dessa fase são normalmente realizados pelo próprio desenvolvedor, conforme ele escreve as funções do sistema.

Por sua vez, o teste de integração foca nas estruturas de comunicação do sistema, objetivando integrar as unidades que foram testadas individualmente e que, agora, devem funcionar em conjunto (Valle, 2017). Ao integrar as unidades é necessário verificar se o software se comportará de maneira adequada, especialmente porque sistemas complexos geralmente são desenvolvidos por mais de uma pessoa e cada uma pode interpretar a especificação de maneira diferente, o que pode ocasionar conflitos em integrações e lacunas diversas (uma função pode não estar preparada para receber um determinado tipo de dado, por exemplo). Para realizar testes de integração é fundamental compreender os componentes internos do software e, por isso, os testes dessa fase são geralmente conduzidos pela equipe de desenvolvimento.

Por fim, depois que todas as unidades foram integradas, inicia-se a fase de teste de sistema, cujo objetivo é verificar se o software se comporta conforme as especificações do sistema, abrangendo requisitos funcionais e não funcionais. Esse tipo de teste costuma ser realizado por testadores.

2.1.4 Técnicas e critérios de teste

Segundo Delamaro, Maldonado e Jino (2016), para se mostrar por meio de teste que um programa P está correto seria necessário executá-lo com todo o seu domínio de entrada $D(P)$. Contudo, devido à cardinalidade de $D(P)$, tal abordagem é em geral inviável, uma vez que se o domínio de entrada for muito grande ou infinito pode ser muito difícil ou impossível testar P com todos os elementos de $D(P)$ em um tempo factível (Wazlawick, 2013).

Tendo conhecimento do exposto acima, se faz necessário criar conjuntos de testes reduzidos, mas que, ao mesmo tempo, encontrem o máximo de defeitos possíveis. Uma heurística para isso consiste em dividir o domínio de entrada em subdomínios/partições nas quais seus elementos se comportem de maneira semelhante. Com isso, duas abordagens são possíveis para teste: o teste de partição e o teste aleatório. No teste de partição ou teste de subdomínio, cada subdomínio é testado individualmente, sendo utilizado como dado de teste apenas um elemento de cada subdomínio. Já no teste aleatório deve-se selecionar aleatoriamente um número suficientemente grande de dados de teste, de modo que estatisticamente se espera todos os subdomínios possíveis sejam contemplados no teste.

Por sua vez, para a identificação dos subdomínios de teste são estabelecidas “regras” conhecidas como “critérios de teste”. Assim, a partir de critérios distintos podem ser obtidos subdomínios diferentes e conjuntos variados de teste. De acordo com Delamaro, Maldonado e Jino (2016) os critérios de teste costumam ser agrupados em técnicas complementares e diferenciadas pela fonte/tipo de informação utilizada para definir os subdomínios. Uma dessas técnicas é a funcional, que será apresentada na próxima seção.

2.1.5 Técnica Funcional

Segundo Myers, Badgett e Sandler (2012), na técnica funcional o sistema é considerado como uma caixa preta, uma vez que os casos de teste são estabelecidos unicamente a partir das especificações do programa, sem que sejam considerados detalhes de sua implementação interna para tal. Uma vez definidos os casos de teste, são então fornecidas entradas ao sistema e avaliadas as suas saídas, de modo a verificar se essas estão em conformidade com os objetivos especificados.

Conforme Delamaro, Maldonado e Jino (2016), por se basearem somente em especificações, os critérios de teste dessa técnica podem ser aplicados em todas as fases de teste e em

sistemas implementados com qualquer paradigma de programação. Ainda, pelo mesmo motivo, a qualidade dos testes funcionais está fortemente atrelada às especificações do sistema, uma vez que especificações incompletas ou incorretas refletirão nos casos de teste criados.

São exemplos de critérios dessa técnica (Delamaro; Maldonado; Jino, 2016):

- Particionamento de equivalência: esse critério recomenda a divisão do domínio de entrada em subdomínios (tal como mencionado na Subseção 2.1.4), os quais são nomeados de classes de equivalência. Feito isso, o conjunto de teste deve ser formado por um caso de teste para cada classe de equivalência definida.
- Análise do valor limite: a experiência mostra que casos de teste que exploram condições limite (valores exatamente sobre ou imediatamente acima ou abaixo dos limitantes de classes de equivalência) têm uma maior probabilidade de encontrar defeitos. Sabendo disso, a análise do valor limite utiliza as mesmas partições/classes que seriam definidas pelo particionamento de equivalência, mas recomenda que a seleção dos dados ocorra de modo a explorar os limites das classes de equivalência (e não de forma aleatória). Ainda, esse critério orienta que tanto o domínio de entrada quanto o de saída sejam considerados para a criação dos casos de teste.
- Teste funcional sistemático: esse critério combina o particionamento de equivalência e a análise do valor limite. Tal como na análise do valor limite, ele requer testes com valores exatamente sobre e imediatamente acima e abaixo dos limitantes de cada partição. Todavia, diferentemente dos critérios anteriores, o teste funcional sistemático exige a definição de dois casos de teste (e não somente um) para cada classe de equivalência.

2.2 Gamificação

De acordo com Fadel *et al.* (2014), a gamificação consiste na utilização de elementos de jogos — isto é, desafios, metas, recompensas, etc — em atividades que não são de jogos. Na educação ela integra elementos de jogos para aprimorar os resultados de aprendizagem e aspectos sociais, promovendo um aprendizado interativo e eficaz, equilibrando o uso da tecnologia para prevenir o isolamento social. Ela atua como uma ferramenta motivacional, melhorando a experiência do usuário, a motivação e a conquista de objetivos por meio de elementos como desafios e recompensas. Ao alinhar-se com os objetivos educacionais e as necessidades dos alunos, a gamificação aumenta a motivação e a participação dos estudantes, promovendo engajamento e melhores resultados de aprendizado em várias áreas. A gamificação se cruza com narrativas de quadrinhos hipermídia, enfatizando elementos como cenários de fantasia, objetivos claros e feedback para experiências de aprendizado envolventes. Os jogos envolvem os jogadores em papéis, desafios e regras, conectando mundos de jogos com a realidade através de referências narrativas, focando no fechamento de tarefas, regras e feedback para engajar emo-

cionalmente os jogadores, enquanto aumenta a motivação com elementos de fantasia, ciclos rápidos de resposta e reconhecimento. Sistemas gamificados rastreiam métricas de desempenho para impulsionar o engajamento, promover a interação e monitorar o progresso através de elementos simbólicos para reforço, destacando a importância de desafios e missões para uma experiência de jogo coesa.

Iniciativas acadêmicas usam gamificação para fomentar a colaboração e experiências de aprendizagem diversificadas. A gamificação na educação aumenta a participação dos alunos e os resultados de aprendizagem através de pontos, desafios e recompensas. Integrar elementos de jogos nas práticas de ensino melhora o engajamento e a interação na educação moderna, defendendo o uso da tecnologia para experiências de aprendizado aprimoradas (Fadel *et al.*, 2014).

2.3 Considerações finais do capítulo

Este capítulo apresentou o referencial teórico do trabalho, sendo nele definidos os termos e conceitos necessários ao entendimento da pesquisa. O capítulo seguinte apresenta os trabalhos relacionados.

3 TRABALHOS RELACIONADOS

Nesta seção será apresentada uma lista não exaustiva de trabalhos relacionados à presente pesquisa, os quais foram identificados por meio de uma revisão da literatura. Nas seções 3.1 a 3.4 serão detalhados cada um dos trabalhos correlatos. Por fim, na Seção 3.5 serão descritos os diferenciais do QA+ em relação às propostas dos demais estudos.

3.1 Estratégia para ensino de teste exploratório

O Teste Exploratório (TE) é uma abordagem de teste manual que permite ao testador explorar o sistema e adquirir conhecimento do programa enquanto realiza os testes. Essa abordagem não segue casos de teste pré-estabelecidos, proporcionando flexibilidade e *feedback* rápido, embora sofra com a falta de documentação estruturada. Para melhorar essa questão, técnicas como o *Session-Based Test Management* (SBTM) podem ser integradas ao TE.

Sabendo disso, Costa e Oliveira (2021) propõem uma estratégia sistemática de ensino de TE (apoiada em SBTM) utilizando gamificação, a qual visa proporcionar aos estudantes um melhor entendimento dessa abordagem, capacitando-os para o mercado de trabalho e contribuindo para futuras pesquisas no campo.

Para avaliar a estratégia criada, Costa e Oliveira (2021) conduziram um estudo não-experimental em um laboratório de informática com uma equipe de quatro estudantes. O estudo foi dividido em sete dias, incluiu aulas teóricas e práticas, uma prova avaliativa e premiação final.

Os resultados quantitativos do estudo de Costa e Oliveira (2021) mostraram que os alunos tiveram um bom desempenho geral, embora alguns conceitos tenham sido confundidos com etapas do processo de Verificação e Validação (V&V). Em termos qualitativos, os alunos avaliaram positivamente os exercícios e as aulas, destacando a importância do *feedback* e do tempo para praticar. Assim, segundo os autores, a estratégia gamificada para o ensino de TE foi bem-sucedida em engajar os estudantes e prepará-los para aplicar o conhecimento sobre esse tópico de teste no mercado.

3.2 *Bug Hunt*

O *Bug Hunt* (Elbaum *et al.*, 2007) é uma aplicação web que visa engajar os estudantes na aprendizagem de estratégias de teste de software de maneira prática e acessível. A aplicação foi construída com tecnologias Java 2 *Enterprise Edition* (J2EE), utilizando uma arquitetura *Model-View-Controller* (MVC). Nela, os exercícios e o desempenho dos estudantes são armazenados em um banco de dados relacional. Por sua vez, sua interface consiste em páginas *Jakarta Server Pages* (JSP) com acesso restrito baseado no tipo de conta do usuário (estudante ou instrutor).

Cada lição do *Bug Hunt* é composta por uma descrição dos conceitos de teste que o estudante deve compreender e, ao final da lição, alguns exercícios e testes que os discentes devem responder, bem como instruções, artefatos e *feedback*. Ao final da lição é apresentado um resumo do desempenho do estudante em termos de detecção de falhas e outras informações pertinentes.

O *Bug Hunt* é composto por quatro lições, as quais abordam os seguintes tópicos: (1) conceitos e terminologia de teste (uma introdução a conceitos básicos de suíte de testes e organização das atividades de teste); (2) teste de caixa preta (focando na relação entre entradas e saídas do programa para descobrir diferentes categorias de entrada); (3) teste de caixa branca (focando no uso do código fonte para encontrar falhas, com ênfase na cobertura de instruções); e (4) automação e eficiência dos testes (integração dos conceitos das lições anteriores com introdução à automação de testes).

A partir da avaliação da aplicação/ferramenta, Elbaum *et al.* (2007) observaram que o *Bug Hunt* ajudou a melhorar o interesse e o engajamento dos estudantes em atividades de teste de software. Ainda, a ferramenta também se mostrou eficaz em integrar conceitos de teste, contribuindo para a formação de melhores práticas de desenvolvimento entre os estudantes.

3.3 As Aventuras de Jack Test

"As Aventuras de Jack Test" (Macêdo, 2014) é um jogo educacional desenvolvido para ensinar conceitos de teste de software. Seu objetivo principal é reforçar tópicos apresentados em sala de aula, capacitando o aluno a diferenciar as fases de teste e compreender conceitos como depuração, verificação e validação.

O jogo é online e foi desenvolvido utilizando as linguagens HTML5 e Javascript. Nele, o jogador assume o papel de Jack Test, um engenheiro de teste que sofre um acidente de avião e precisa testar partes de um barco construído para escapar de uma ilha cheia de desafios.

O jogo é dividido em oito etapas: quatro fases e quatro desafios, cada um relacionado a um conteúdo específico de teste de software. Abaixo são descritos os conteúdos de cada fase/desafio, bem como as atividades que o jogador deve realizar nelas.

- Fases: (1) teste unitário (escolher as alternativas corretas referentes ao teste unitário); (2) teste de integração (escolher as alternativas corretas referentes ao teste de integração); (3) teste de validação (identificar o teste a ser realizado em uma situação descrita); e (4) teste de sistema (identificar o teste a ser realizado em uma situação descrita).

- Desafios: (1) garantia de qualidade de software (completar frases com conceitos de garantia de qualidade de software); (2) métodos de teste de software (classificar características entre testes de caixa preta e caixa branca); (3) processo de teste de software

(ordenar as fases do processo de teste de software); e (4) depuração (desvendar a palavra secreta relacionada à depuração).

Para avaliar a efetividade do jogo um questionário foi aplicado a alunos de Engenharia de Software da Faculdade de Tecnologia e Ciência de Vitória da Conquista - BA, os quais tiveram acesso ao jogo. De acordo com Macêdo (2014), os resultados da avaliação de “As Aventuras de Jack Test” demonstraram que a gamificação pode ser uma abordagem eficaz para o ensino de conceitos de teste de software, uma vez que o jogo não apenas aumentou a motivação e o engajamento dos alunos, mas também facilitou uma compreensão mais profunda dos conceitos técnicos. Os resultados sugerem que o jogo tem um potencial significativo como ferramenta educacional complementar.

3.4 GGCC

O *Gamifying Graph Coverage Criteria* (GGCC) (Souza; Borges; Durelli, 2022) é uma plataforma online que suporta atividades pós-treinamento direcionadas ao ensino de critérios de cobertura de grafos. A plataforma emprega elementos lúdicos para engajar os alunos e tornar o aprendizado mais envolvente e motivador. A ferramenta inclui material teórico, quizzes, reforços positivos (como mensagens de parabéns e efeitos sonoros/visuais) e a concessão de *badges* ao completar quizzes.

O conceito de gamificação utilizado no GGCC visa motivar os discentes a partir do uso de elementos básicos de jogos, tais como sistemas de pontuação, troféus, medalhas e moedas virtuais, o que pode ser classificado como “*shallow gamification*” (em português, gamificação rasa). Segundo Souza, Borges e Durelli (2022), isso é adequado dado o curto período necessário para estudar o conteúdo.

Para avaliar o GGCC Souza, Borges e Durelli (2022) conduziram um experimento com alunos de graduação e mestrado da Universidade Federal de São João del-Rei, comparando a abordagem gamificada GGCC com uma abordagem tradicional de ensino. No estudo os participantes foram divididos em dois grupos: um grupo utilizou slides e um questionário (abordagem tradicional), enquanto o outro grupo utilizou a plataforma GGCC para atividades pós-treino.

A partir da análise dos resultados do experimento, Souza, Borges e Durelli (2022) evidenciaram que o uso da gamificação, mesmo que de forma rasa, pode melhorar o engajamento e a motivação dos alunos em atividades de aprendizado de curto prazo, sendo que a plataforma GGCC mostrou-se eficaz em aumentar a compreensão dos critérios de teste de software através de uma abordagem lúdica e interativa.

3.5 Diferenciais do QA+

O QA+ possui alguns diferenciais em relação aos jogos/propostas apresentadas nas seções anteriores.

De acordo com Belinski (2021), a principal diferença do QA+ em relação a outros jogos existentes para o ensino de teste e qualidade de software é que ele compreende um ambiente com dinâmicas próprias que possibilitam que o aluno não somente aprenda, mas também ensine. Para isso, o QA+ disponibiliza materiais pré-definidos, mas igualmente permite a criação e compartilhamento de conteúdo pelo próprio jogador, com a aplicação de dinâmicas do jogo sobre o conteúdo personalizado. Ainda, o QA+ apresenta um conjunto de recursos de gestão e acompanhamento voltados ao apoio das atividades desempenhadas por quem ensina.

Maiores detalhes sobre o QA+ e as funcionalidades previstas para o jogo serão descritos na Subseção 5.1.1.

3.6 Considerações finais do capítulo

Este capítulo apresentou trabalhos relacionados a presente pesquisa, ou seja, que utilizam gamificação para o ensino de teste e qualidade de software. Ainda, descreveu os diferenciais do QA+ em relação às propostas desses estudos.

O capítulo seguinte apresenta o método de pesquisa empregado neste trabalho.

4 MÉTODO

Este capítulo apresenta o método de pesquisa empregado no trabalho, o qual foi composto por duas fases. A Seção 4.1 discorre acerca das atividades da Fase 1, a qual compreendeu a implementação de uma versão inicial[1] do QA+. Por sua vez, a Seção 4.2 descreve as atividades da Fase 2, relacionada à avaliação do jogo desenvolvido na Fase 1.

4.1 Fase 1 — Implementação do QA+

A Fase 1 correspondeu à implementação de uma versão inicial do QA+. Primeiramente, devido a restrições de tempo para a execução deste trabalho, foi delimitado o escopo de implementação em relação à proposta original de Belinski (2021). O resultado da atividade de delimitação de escopo será exposto na Subseção 5.1.1.

Em sequência, foi realizada a modelagem e codificação do QA+, cujos artefatos serão detalhados nas Subseções 5.1.2 e 5.1.3, respectivamente. Para a execução destas atividades foram utilizadas as seguintes ferramentas e/ou tecnologias:

- Linguagens de programação e/ou marcação:
  - Para o *front-end*: React.js com Typescript.
  - Para o *back-end*: Java 21 com framework Spring Boot.
- Ambientes/plataformas de desenvolvimento:
  - Para o *front-end*: Visual Studio Code.
  - Para o *back-end*: IntelliJ IDEA.
- Banco de dados e seus gerenciadores: MySQL v8.1.0.
- Publicização/disponibilização do código: GitHub[2].

As ferramentas e/ou tecnologias listadas acima foram escolhidas em decorrência de sua ampla utilização na indústria de software. Ainda, cabe ressaltar que:

- Para o *front-end*, o React.js foi escolhido devido ao seu amplo uso industrial, vasta quantidade de conteúdos disponíveis na internet para resolução de dúvidas e quantidade considerável de componentes prontos disponíveis na internet (a exemplo de

[1] Versão com escopo limitado em relação à proposta original de Belinski (2021). Todavia, destaca-se que neste trabalho esta foi a única versão implementada. O termo "inicial" é empregado, pois: (1) este é o primeiro estudo que implementa parte do QA+; e (2) é esperado que estudos futuros deem continuidade à implementação, adicionando outras funcionalidades originalmente previstas para o jogo.

[2] Em https://github.com/fewiip/qa/ consta o código-fonte do *front-end* enquanto em https://github.com/Shundaa/qa/ consta o código-fonte do *back-end*.

editores de *markdown*). Já a escolha do Typescript decorreu do fato dessa linguagem possuir mais funcionalidades do que o Javascript (permite trabalhar com orientação a objetos, possui mais segurança, permite o uso do ESLint para análise do código, etc).

- Para o *back-end*, escolhemos a linguagem Java por essa ser amplamente utilizada na indústria de software, apresentar um desempenho robusto e capacidade de escalabilidade. Por sua vez, o Spring Boot tira proveito dessas características, facilitando a criação de aplicações que podem ser facilmente escaladas horizontalmente (em vários servidores) e verticalmente (adicionando mais recursos ao servidor).

4.2 Fase 2 — Avaliação do QA+

A Fase 2 correspondeu à avaliação do QA+, a qual foi realizada através da condução de um estudo de caso com alunos de graduação.

Devido a limitação de tempo para a execução da fase de avaliação, a técnica utilizada para a seleção dos participantes foi a amostragem por conveniência. Assim, foram convidados a participar do estudo alunos de cursos da área de tecnologia da informação (*e.g.:* Sistemas de Informação, Engenharia de Controle e Automação e Engenharia da Computação) da Universidade Tecnológica Federal do Paraná (UTFPR), mesma instituição dos autores deste trabalho.

Por sua vez, o *design* da pesquisa foi não-experimental com pós-teste (*one-shot post-test only design*) — conforme Petri, Wangenheim e Borgatto (2019) — e único grupo. Assim, no estudo de caso os participantes foram inicialmente submetidos ao tratamento (jogo QA+). Em seguida, lhes foi solicitado para responder um questionário de caracterização do participante e um instrumento de medição (questionário descrito na Subseção 4.2.1) que visou coletar as suas percepções sobre o jogo/avaliar a aplicabilidade do QA+. Todo esse processo foi mediado por um formulário do *Google Forms*, o qual pode ser visualizado no Apêndice A.

Por fim, objetivando responder a questão de pesquisa deste estudo, foram conduzidas análises quantitativas sobre os dados coletados. Os resultados das análises podem ser encontrados na Seção 5.2.

4.2.1 Questionário MEEGA+

O instrumento de medição utilizado para coletar as percepções dos participantes sobre o jogo, avaliando a aplicabilidade do QA+ (conforme definição do Capítulo 1), foi o MEEGA+. Esse instrumento pode ser definido como um modelo sistemático para a avaliação de jogos usados para o ensino de computação no contexto de cursos superiores dessa área, sendo uma evolução de sua versão anterior, o *Model for the Evaluation of Educational GAmes* (MEEGA) (Petri; Wangenheim; Borgatto, 2019).

O MEEGA+ consiste em um questionário composto por itens/afirmativas a serem valoradas pelos participantes utilizando uma escala Likert de 5 pontos (com respostas que variam de discordo totalmente a concordo totalmente). Essas afirmativas estão agrupadas em duas dimensões (usabilidade e experiência do jogador), as quais se organizam em subdimensões. As subdimensões de usabilidade avaliam aspectos relacionados a estética, aprendizibilidade, operabilidade, acessibilidade e proteção contra erros do usuário. Por sua vez, as subdimensões de experiência do jogador avaliam aspectos de confiança, desafio, satisfação, diversão, interação social, atenção focada, relevância e aprendizagem percebida (Petri; Wangenheim; Borgatto, 2019).

A versão do MEEGA+ utilizada neste trabalho (dimensões, subdimensões e itens) pode ser visualizada no Quadro 1.

Quadro 1 – Itens do questionário de avaliação do QA+

Dimensão	Subdimensão	Item	Descrição do Item
Usabilidade	Estética	1	O design do jogo é atraente (interface, gráficos, etc.)
		2	Os textos, cores e fontes combinam e são consistentes.
	Aprendizibilidade	3	Eu precisei aprender poucas coisas para poder começar a jogar o jogo.
		4	Aprender a jogar este jogo foi fácil para mim.
		5	Eu acho que a maioria das pessoas aprenderiam a jogar este jogo rapidamente.
	Operabilidade	6	Eu considero que o jogo é fácil de jogar.
		7	As regras do jogo são claras e compreensíveis.
	Acessibilidade	8	As fontes (tamanho e estilo) utilizadas no jogo são legíveis.
		9	As cores utilizadas no jogo são compreensíveis.
		10	O jogo permite personalizar a aparência (fonte e/ou cor) conforme a minha necessidade.
	Proteção contra erros do usuário	11	O jogo me protege de cometer erros.
		12	Quando eu cometo um erro é fácil de me recuperar rapidamente.
Experiência do jogador	Confiança	13	Quando olhei pela primeira vez o jogo, eu tive a impressão de que seria fácil para mim.
		14	A organização do conteúdo me ajudou a estar confiante de que eu iria aprender com este jogo.
	Desafio	15	Este jogo é adequadamente desafiador para mim.
		16	O jogo oferece novos desafios (oferece novos obstáculos, situações ou variações) com um ritmo adequado.
		17	O jogo não se torna monótono nas suas tarefas (repetitivo ou com tarefas chatas).
	Satisfação	18	Completar as tarefas do jogo me deu um sentimento de realização.

(continua)

Quadro 1 – Itens do questionário de avaliação do QA+

(continuação)

Dimensão	Subdimensão	Item	Descrição do Item
		19	É devido ao meu esforço pessoal que eu consigo avançar no jogo.
		20	Me sinto satisfeito com as coisas que aprendi no jogo.
		21	Eu recomendaria este jogo para meus colegas.
	Diversão	22	Eu me diverti com o jogo.
		23	Aconteceu alguma situação durante o jogo (elementos do jogo, competição, etc.) que me fez sorrir.
	Atenção focada	24	Houve algo interessante no início do jogo que capturou minha atenção.
		25	Eu estava tão envolvido no jogo que eu perdi a noção do tempo.
		26	Eu esqueci sobre o ambiente ao meu redor enquanto jogava este jogo.
	Relevância	27	O conteúdo do jogo é relevante para os meus interesses.
		28	É claro para mim como o conteúdo do jogo se relaciona com uma disciplina acadêmica que aborda teste de software.
		29	O jogo é um método de ensino adequado para uma disciplina acadêmica que aborda teste de software.
	Aprendizagem percebida	30	O jogo contribuiu para relembrar os conceitos de teste de software.
		31	O jogo contribuiu para compreender os conceitos de teste de software.
		32	O jogo contribuiu para utilizar os conceitos de teste de software.

Adaptado de Petri, Wangenheim e Borgatto (2019).

Por fim, acerca dos itens do Quadro 1, cabe destacar que as seguintes adaptações foram realizadas em relação ao questionário original apresentado no estudo de Petri, Wangenheim e Borgatto (2019):

- No item 1 foram removidos os exemplos que mencionavam tabuleiros e cartas, uma vez que esses não são elementos utilizados no QA+.
- A subdimensão nomeada “Interação social” não foi utilizada, uma vez que os recursos de interação social previstos por Belinski (2021) não fizeram parte do escopo de implementação deste trabalho.
- O texto dos itens 28 e 29 eram originalmente “É claro para mim como o conteúdo do jogo está relacionado com a disciplina” e “O jogo é um método de ensino adequado

para esta disciplina". Todavia, como a avaliação não ocorreu no contexto de uma disciplina, eles foram adaptados respectivamente para "É claro para mim como o conteúdo do jogo se relaciona com uma disciplina acadêmica que aborda teste de software" e "O jogo é um método de ensino adequado para uma disciplina acadêmica que aborda teste de software".

- Como o jogo não foi avaliado no contexto de uma disciplina, na subdimensão "Aprendizagem percebida" foram mantidas apenas as afirmativas adicionais sugeridas por Petri, Wangenheim e Borgatto (2019), ou seja, aquelas relacionadas aos níveis cognitivos de lembrança, compreensão e aplicação (seguindo a taxonomia de Bloom), uma vez que representam o que pode ser aprendido durante um curso de graduação. Essas afirmativas seguem o formato "O jogo contribuiu para <verbo conforme nível do objetivo de aprendizagem (cognitivo, psicomotor, afetivo)> <objetivo/conceito>".

4.3 Considerações finais do capítulo

Este capítulo apresentou o método de pesquisa do trabalho, tendo exposto suas fases e uma visão geral das atividades e instrumentos que as compuseram. O capítulo seguinte apresenta e discute os resultados do estudo.

5 RESULTADOS E DISCUSSÃO

Este capítulo expõe e discute os resultados obtidos para cada fase da pesquisa. Na Seção 5.1 serão apresentados os resultados da Fase 1, relacionada à implementação de uma versão inicial do QA+. Por sua vez, na Seção 5.2 serão detalhados e discutidos os resultados da Fase 2, correspondente à avaliação do QA+.

5.1 Fase 1 — Implementação do QA+

Esta seção detalha a implementação do QA+ realizada neste trabalho.

5.1.1 Delimitação de escopo

O QA+ é um jogo sério, digital e online proposto por Belinski (2021)[1] para apoiar o ensino de teste e qualidade de software. No jogo é prevista a utilização de dinâmicas, mecânicas e componentes diversos de jogos para que os estudantes possam se engajar em relação ao aprendizado do conteúdo. Esses elementos[2] estão organizados em funcionalidades que possibilitam aos estudantes criar e se inscrever em múltiplas turmas, acessar o conteúdo de lições, resolver quizzes e desafios (individualmente ou interagindo com outros usuários), personalizar seu perfil, etc. Por sua vez, para instrutores/docentes o QA+ oferece ferramentas para a manutenção e acompanhamento de turmas.

Em decorrência de limitações de tempo, neste trabalho o escopo de implementação do QA+ precisou ser limitado. Assim, na Subseção 5.1.1.1 serão apresentados os requisitos contemplados nesta pesquisa. Por sua vez, na Subseção 5.1.1.2 serão apresentados os requisitos não implementados.

5.1.1.1 Requisitos implementados (total ou parcialmente)

Esta seção apresenta os requisitos implementados neste trabalho, bem como eventuais alterações nesses e/ou implementações parciais. Na Subseção 5.1.1.1.1 constam os requisitos funcionais (RF), enquanto na Subseção 5.1.1.1.2 são listados os requisitos não funcionais (RNF).

[1] Belinski (2021) apresenta a ideia do QA+, disponibilizando um protótipo executável e artefatos (como diagramas de fluxo de negócio, requisitos funcionais e não funcionais) para auxiliar na compreensão da lógica do jogo. Contudo, em seu trabalho ainda não havia sido realizada qualquer implementação/codificação do QA+, sendo isso sugerido pelo autor como uma oportunidade de trabalho futuro.

[2] Belinski (2021) utiliza elementos de jogos como pontos/estatísticas, conquistas, *badges*, quadro de líderes/classificação, desafios, chance/sorte, competição, combate, *feedback*, *avatar*, desbloqueio de conteúdo e níveis.

Para fins de mapeamento, nas seções a seguintes a numeração dos requisitos é a mesma utilizada por Belinski (2021), sendo também desta obra os textos entre aspas. Os requisitos a seguir foram escolhidos para implementação (em detrimento dos demais existentes), pois neste trabalho buscou-se desenvolver uma versão correspondente a um *Minimum Viable Product* (MVP) (em português, Produto Mínimo Viável) do jogo, tendo sido focado, então, nas funcionalidades básicas do QA+.

5.1.1.1.1 Requisitos Funcionais

RF1: "O sistema deve permitir que um usuário crie uma conta no jogo por meio do informe de nome, usuário, e-mail e senha". Este requisito foi parcialmente atendido, uma vez que não foi implementado o campo de e-mail.

RF2: "O sistema deve apresentar uma página de login que permita o acesso ao jogo por meio do informe de: (1) e-mail ou nome de usuário e (2) senha". Como citado no RF1, o campo de e-mail não foi implementado. Assim, o acesso ocorre unicamente por meio do informe de nome de usuário e senha.

RF4: “O sistema deve apresentar um esquema de recompensas aos usuários baseado em pontos (de tipos nomeados *bugs*, moedas e vitórias/derrotas) e *badges* (nomeados de conquistas)”. Este requisito foi parcialmente atendido, pois as conquistas/*badges* não foram implementadas e, apesar do totalizador de vitórias/derrotas ter sido criado, esse ainda não é atualizado, uma vez que os requisitos RF27 e RF28 (que manipulam os valores desse campo) não foram implementados.

RF5: "O sistema deve apresentar uma tela principal de lições contendo um resumo do quadro de líderes (RF6, RF7, RF8), a indicação da próxima conquista a ser alcançada pelo usuário (RF9, RF10), um resumo das estatísticas do usuário (RF11, RF12), as unidades de ensino-aprendizagem do jogo (RF14, RF15, RF16) e um quadro relacionado à funcionalidade de desafio diário (RF19)". Este requisito foi parcialmente atendido, uma vez que não foram implementadas as funcionalidades de desafio diário e indicação da próxima conquista a ser alcançada pelo usuário.

RF6: "O quadro de líderes deve apresentar um ranking (em ordem decrescente) dos usuários com a maior quantidade de pontos de bugs no momento, devendo ser apresentadas nele as informações de ordem de classificação, avatar, nome e total de pontos de bugs do usuário".

RF11: "O resumo de estatísticas do usuário deve apresentar para cada estatística (pontos de bugs, moedas, vitórias e refis) seu nome, imagem ilustrativa e saldo atual do usuário".

RF12: "Abaixo da estatística de refis deve ser apresentado um botão para reposição. Quando este for acionado deverão ser atribuídos 5 refis e consumidas 5 moedas do usuário". Este requisito foi implementado de outra forma. Quando o botão de reposição de refis é acionado é atribuído um refil e consumida uma moeda do usuário. Ainda sobre refis e moedas cabe

mencionar que são automaticamente atribuídas 5 moedas e 5 refis ao usuário quando esse se inscreve em uma turma. Ademais, houve uma alteração no que foi originalmente proposto para os refis: para Belinski (2021) os refis funcionariam com um conceito análogo ao de "vidas", ou seja, ao errar a resposta de um quiz o usuário gastaria um refil e não ganharia a pontuação de bugs da questão. Por sua vez, neste trabalho o funcionamento dos refis foi implementado da seguinte forma: ao errar a resposta de um quiz o usuário gasta um refil, mas ainda assim ganha os pontos de bugs da questão.

RF14: "As unidades de ensino-aprendizagem do jogo devem ser organizadas em seções, sendo que cada seção deve ser identificada por um título". Neste trabalho as seções foram renomeadas para capítulos.

RF15: "Cada unidade de ensino-aprendizagem deve ser identificada por um título e um símbolo de lupa em uma das seguintes combinações de cores: (a) cinza: se a unidade estiver bloqueada; (b) amarelo e branco: se a unidade estiver desbloqueada (a borda deve ser colorida em dourado conforme o usuário progride no material disponibilizado; deve ser apresentado no centro da lupa uma “rachadura” na cor branca caso a conclusão da unidade tenha ocorrido a mais de 30 dias); ou (c) amarelo e dourado (somente): se a unidade tiver sido concluída pelo usuário em período menor ou igual a 30 dias". Este requisito foi implementado com algumas modificações. Seu comportamento atual é: (a) uma lição com lupa cinza está bloqueada; (b) uma lição com a lupa branca está desbloqueada, mas ainda não foi acessada; (c) uma lição com a lupa parcialmente colorida foi acessada, mas ainda não teve seu quiz resolvido; e (d) uma lição com a lupa totalmente colorida foi acessada e teve seu quiz concluído. O comportamento de “rachadura” não foi implementado.

RF16: "Cada unidade de ensino-aprendizagem deve ser composta por uma lição (RF17) e um conjunto de atividades (nomeados desafios; RF18). Ao se clicar em uma unidade desbloqueada deverão ser apresentadas opções de acesso à sua lição e atividades relacionadas". Neste trabalho os desafios foram renomeados para quiz.

RF17: "Cada lição deve ser composta por um título, conteúdos em formato multimídia, uma lista de referências e uma opção para acesso aos desafios da unidade". Este requisito foi implementado com algumas alterações: a lista de referências não foi disponibilizada na lateral da tela, mas sim textualmente ao final da lição. Em relação ao conteúdo das lições, esse pode ser adicionado por meio de um editor de *markdown*, sendo possível incluir imagens se elas possuírem um link externo.

RF18: "Cada conjunto de desafios deve apresentar uma série de atividades ao usuário (com correção automática) relacionadas ao conteúdo da unidade. Por meio da resolução das atividades o usuário poderá acumular pontos e moedas (proporcionais à sua quantidade de acertos)". O fluxo de desafios foi implementado como está apresentado na Figura 1.

RF20: "O sistema deve disponibilizar uma página nomeada “Turmas”, a qual deve permitir com que o usuário crie, busque e participe de turmas adicionais (além da turma padrão QA+). Por meio desta página deve ser possível acessar funcionalidades de: (a) criação de turma

(RF21); (b) gerenciamento de turma criada pelo próprio usuário (RF22); (c) acesso à turma adicional em que o usuário já participa (RF25); (d) busca de turmas por termo; e (e) solicitação de acesso/ingresso em turma criada por outro usuário (RF26)". Não foi implementada a funcionalidade do RF26 como originalmente previsto (o usuário consegue se inscrever na turma, mas a inscrição não passa por um processo de avaliação).

RF21: "A funcionalidade de criação de turma deve disponibilizar recursos para informe de nome, imagem e situação da turma, bem como para importação dos dados padrão da turma QA+, exportação dos dados da turma, adição de nova seção, nova unidade, nova lição, novo desafio, edição e remoção de dados". Requisito parcialmente atendido, uma vez que não foram implementadas as funcionalidades de definição de imagem e situação da turma, tampouco de importação e exportação de dados da turma.

RF23: "O painel de controle de turma deve apresentar opções para acesso a funcionalidades de: (a) manutenção dos dados da turma (edição dos mesmos campos do RF21); (b) exclusão da turma; (c) manutenção das solicitações pendentes de ingresso na turma (visualização, aceite e negativa); (d) manutenção dos inscritos (busca, visualização e remoção da turma); (e) visualização da turma como inscrito (visão descrita no RF25)". Requisito parcialmente atendido, dado que não foram implementadas as funcionalidades de manutenção das solicitações pendentes de ingresso na turma e de manutenção dos inscritos.

RF29: "O sistema deve apresentar uma página de perfil que possibilite ao usuário atualizar seus dados de cadastro, escolher um avatar, visualizar suas estatísticas de desempenho e conquistas obtidas no jogo (RF9), excluir sua conta e realizar logout". Requisito parcialmente atendido, uma vez que não foram implementadas as funcionalidades de escolha de avatar e conquistas.

5.1.1.1.2 Requisitos não funcionais

RNF2: "O sistema deve estar disponível na internet, sendo acessível minimamente a partir dos navegadores Google Chrome, Mozilla Firefox, Microsoft Edge e Safari".

RNF3: "O sistema deve ser disponibilizado minimamente na língua portuguesa".

RNF4: "O sistema deve possuir uma interface intuitiva e de fácil utilização, permitindo novos usuários navegarem com o mínimo de esforço ou treinamento".

RNF5: "O acesso ao jogo deve ser restrito a usuários autenticados no sistema".

RNF6: "O sistema deve estar disponível para acesso 24 horas por dia e 7 dias por semana".

RNF7: "O tempo de resposta para a apresentação do quadro de estatísticas dos inscritos (em uma turma) não pode ser superior a 15 segundos".

5.1.1.2 Requisitos não implementados

Esta seção lista os requisitos previstos por Belinski (2021) que não foram implementados neste trabalho em decorrência de limitações de tempo. É importante mencionar que como o código-fonte deste estudo se encontra disponível publicamente (vide Subseção 5.1.3) tais requisitos podem ser implementados em trabalhos futuros, incrementando a versão do software criado neste trabalho.

5.1.1.2.1 Requisitos Funcionais

RF3: "O sistema deve disponibilizar na página de login um recurso para recuperação de senha".

RF7: "A posição em que o usuário logado no sistema se encontra no quadro de líderes deve ser destacada para facilitar sua localização".

RF8: "No quadro de líderes apresentado na tela principal devem ser expostas minimamente as informações: do usuário classificado na 1ª posição, do usuário logado, do usuário que o antecede no ranking (se houver) e do usuário que o sucede no ranking (se houver)".

RF9: "Para cada conquista passível de obtenção pelo usuário deverão ser apresentados: título e imagem ilustrativa da conquista, descrição do que é necessário para obtê-la, indicadores numéricos e ilustrativos do progresso do usuário em relação à obtenção da conquista".

RF10: "A próxima conquista geral a ser alcançada pelo usuário deve ser apresentada em um quadro no topo da tela de lições, sendo igualmente disponibilizado um botão de acesso à visualização de todas as conquistas (ao clicar neste, o usuário deve ser redirecionado à sua página de perfil (RF29))".

RF13: "Uma vez ao dia, caso o saldo de refis do usuário seja inferior a 5, 1 refil será acrescentado automaticamente ao seu saldo de refis".

RF19: "O quadro relacionado à funcionalidade de desafio diário deve apresentar um título, explicação, ícone e opção para começar o desafio (habilitada somente caso o usuário ainda não tenha clicado nele no dia em questão). Nesta funcionalidade deve ser possibilitado que o usuário responda a um desafio extra (intitulado de “desafio diário”) por dia. Caso o usuário acerte a resposta do desafio, além de sua pontuação normal, ele poderá ganhar um adicional de pontos (nomeado “pontos da sorte”) de valor até três vezes superior à pontuação normal do desafio (pontuação gerada aleatoriamente)".

RF22: "A funcionalidade de gerenciamento de turma deve apresentar um painel de controle (RF23) e um quadro de estatísticas dos inscritos (RF24)".

RF24: "O quadro de estatísticas de inscritos em uma turma deve apresentar campos para busca de inscrito, ordenação dos resultados por tipo de estatística (total de bugs, total de vitórias ou total de conquistas) e sentido (crescente ou decrescente), bem como um grid na qual devem ser listados os nomes dos inscritos, seus usuários, links para seus perfis nas

turmas QA+ e minha turma, estatísticas nas turmas QA+ e minha turma e totalizadores gerais das estatísticas".

RF25: "Ao acessar uma turma adicional será disponibilizado ao usuário seus recursos, sendo que esta difere da turma padrão QA+ ao passo que: (a) apresenta conteúdo personalizado; (b) apresenta uma barra com o nome da turma e seu criador, bem como opção para deslogar; e (c) não apresenta acesso à funcionalidade de turmas".

RF26: "Ao realizar uma solicitação de acesso em turma o usuário será listado na funcionalidade “Solicitações Pendentes de Ingresso na Turma” na página de gestão da turma em questão".

RF27: "O sistema deve apresentar uma página nomeada “Arena”, a qual deverá conter explicações sobre esta funcionalidade, opção para iniciar duelo (RF28) e quadros de estatísticas e próxima conquista do usuário na funcionalidade".

RF28: "O sistema deve apresentar uma funcionalidade de duelos, por meio da qual os jogadores podem competir entre si em disputas de cinco minutos que envolvam a resolução de desafios e acarretem na acumulação de pontos e desbloqueio de conquistas".

5.1.1.2.2 Requisitos não funcionais

RNF1: "O sistema deve atender o que se encontra disposto na Lei nº 13.709/2018 (Lei Geral de Proteção de Dados - LGPD)”.

5.1.2 Modelagem

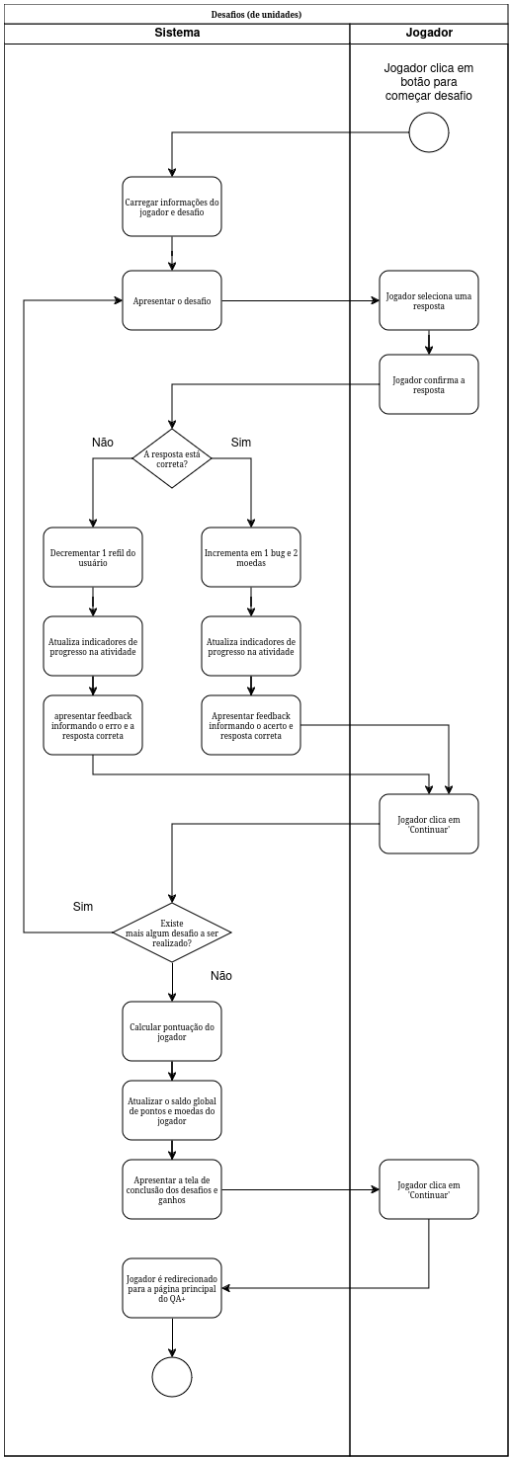
Nesta seção serão apresentados detalhes da modelagem do QA+. Na Subseção 5.1.2.1 serão exibidos os artefatos relacionados a fluxo de negócio, enquanto na Subseção 5.1.2.2 serão descritas informações referentes a modelagem de banco de dados.

5.1.2.1 Diagramas de fluxo de negócio

Na Figura 1 é apresentado o diagrama de fluxo de negócio relacionado à resolução dos quizzes associados a uma lição. Por meio da figura é possível perceber que, após o jogador/usuário clicar no botão para iniciar o quiz/desafio, inicialmente o sistema carrega as informações do quiz e do jogador (sua quantidade de refis). Em sequência, o jogador seleciona e confirma a escolha de uma alternativa de resposta. Feito isso, o sistema verifica se a alternativa escolhida é a correta e: (1) atualiza os saldos de refis, bugs e moedas do jogador (a depender da corretude da resposta); (2) atualiza os indicadores de progresso do jogador no quiz; e (3) apresenta ao jogador se ele acertou ou errou a questão (indicando a resposta correta). Em sequência, o jogador clica em "Continuar". Se ainda houver algum quiz a ser resolvido para a lição os passos

anteriores são repetidos. Caso contrário, é calculada a pontuação do jogador, atualizados seus saldos globais de pontos e moedas e apresentada a tela de conclusão do quiz. Por fim, ao clicar em “Continuar” o jogador é redirecionado para a página principal da turma no QA+.

Figura 1 – Fluxo da resolução dos quizzes associados a uma lição

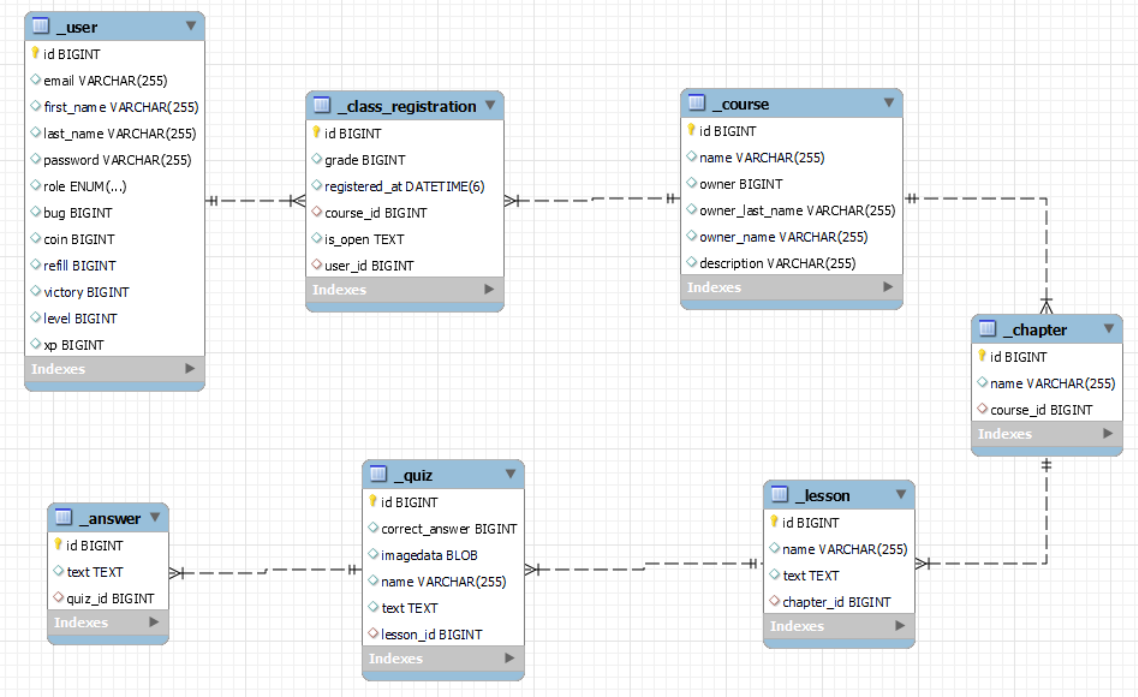


Fonte: Adaptado de Belinski (2021).

5.1.2.2 Banco de dados

Esta seção apresenta detalhes acerca do banco de dados do QA+. Inicialmente, na Figura 2 é exibido o Diagrama Entidade-Relacionamento (DER) do jogo. Esse diagrama foi gerado com a ferramenta MySQL Workbench.

Figura 2 – Diagrama Entidade-Relacionamento



Fonte: Autoria própria (2024).

Por meio da Figura 2 é possível evidenciar que um usuário (cujos dados são armazenados na tabela _user) pode estar registrado em várias turmas (tabela _class_registration). Por sua vez, uma turma (tabela _course) pode ter seu conteúdo organizado em vários capítulos (tabela _chapter), sendo que cada capítulo pode ter várias lições (tabela _lesson). Por fim, a cada lição podem ser associados vários quizzes (tabela _quiz) e a cada quiz várias alternativas de resposta (tabela _answer).

Por fim, de modo a facilitar a compreensão da estrutura do banco de dados, são detalhados os campos de cada tabela e sua função a seguir.

Tabela _user

- id: identificador único do usuário
- email: endereço de e-mail do usuário
- first_name: nome do usuário
- last_name: sobrenome do usuário

- password: senha do usuário
- role: função do usuário (por exemplo, administrador, estudante, professor)
- bug, coin, refill, victory, level, xp: campos relacionados ao sistema de gamificação
- Relacionada com a tabela _class_registration (um usuário pode se registrar em várias turmas)

Tabela _class_registration

- id: identificador único do registro
- grade: nota do usuário na turma
- registered_at: data e hora do registro na turma
- course_id: identificador da turma
- is_open: indica se a turma está aberta ou fechada
- user_id: identificador do usuário
- Relacionada com a tabela _user através do campo user_id
- Relacionada com a tabela _course através do campo course_id

Tabela _course

- id: identificador único da turma
- name: nome da turma
- owner: identificador do dono/criador da turma
- owner_name: nome do dono/criador da turma
- owner_last_name: sobrenome do dono/criador da turma
- description: descrição da turma
- Relacionada com a tabela _class_registration através do campo id
- Relacionada com a tabela _chapter (uma turma pode ter vários capítulos)

Tabela _chapter

- id: identificador único do capítulo
- name: nome do capítulo

- course_id: identificador da turma ao qual o capítulo pertence
- Relacionada com a tabela _course através do campo course_id
- Relacionada com a tabela _lesson (um capítulo pode ter várias lições)

Tabela _lesson

- id: identificador único da lição
- name: nome da lição
- text: texto/conteúdo da lição
- chapter_id: identificador do capítulo ao qual a lição pertence
- Relacionada com a tabela _chapter através do campo chapter_id
- Relacionada com a tabela _quiz (uma lição pode ter vários quizzes)

Tabela _quiz

- id: identificador único do quiz
- correct_answer: identificador da alternativa de resposta correta
- imagedata: dados de imagem associados ao quiz
- name: nome do quiz
- text: texto/conteúdo do quiz
- lesson_id: identificador da lição à qual o quiz pertence
- Relacionada com a tabela _lesson através do campo lesson_id
- Relacionada com a tabela _answer (um quiz pode ter várias respostas)

Tabela _answer

- id: identificador único da alternativa de resposta
- text: texto da alternativa de resposta
- quiz_id: identificador do quiz ao qual a alternativa de resposta pertence
- Relacionada com a tabela _quiz através do campo quiz_id

5.1.3 Codificação

Inicialmente, cabe mencionar que o código-fonte deste trabalho se encontra disponível em dois repositórios públicos: em https://github.com/fewiip/qa/ consta o código-fonte do *front-end* enquanto em https://github.com/Shundaa/qa/ consta o código-fonte do *back-end*.

Acerca da codificação do *front-end* — cuja maioria das telas foram baseadas nos protótipos de Belinski (2021) — cabe ressaltar o uso do React.js. Essa biblioteca possui diversos componentes prontos que foram utilizados neste projeto para, por exemplo, inserir um editor *markdown* no QA+, realizar requisições via HTTP e salvar dados localmente.

Por sua vez, para a codificação do servidor *back-end* em Java foi utilizado o *framework* Spring Boot e alguns de seus módulos:

- **Spring Boot Starter Web** que é uma dependência do Spring Boot que facilita a criação de aplicações web, incluindo *Representational State Transfer* (RESTful) *services*, com o *framework* Spring. Esse *starter* reúne todas as bibliotecas necessárias para configurar uma aplicação web, como o servidor embutido Tomcat, Spring MVC e outros componentes essenciais, permitindo desenvolver a aplicação sem a necessidade de configurar manualmente todas as dependências e componentes. Essa dependência simplifica a configuração inicial, fornecendo todas as dependências e configurações necessárias para criar uma aplicação web.
- **Spring MVC** é um módulo para construir aplicações web e RESTful *Application Programming Interface* (API) usando o padrão MVC.
- **Spring Boot Starter Tomcat** é um módulo que inclui o Apache Tomcat como o servidor de aplicação embutido padrão para aplicações web. Com esse módulo podemos executar a aplicação diretamente como uma aplicação Java padrão (usando o java -jar). Isso simplifica tanto o desenvolvimento local quanto a implantação em produção, eliminando a necessidade de empacotar a aplicação como um WAR e implantá-la em um servidor de aplicação separado.
- **Spring Data *Java Persistence API* (JPA)** é um módulo do projeto Spring Data que simplifica o uso da API de persistência JPA em aplicações Spring. Ele fornece abstrações e implementações prontas para realizar operações de banco de dados, eliminando a necessidade de escrever grande parte do código normalmente associado à utilização do JPA.
- **Spring Boot Starter Security** é um módulo que facilita a integração e configuração do *Spring Security*. O *Spring Security* é um *framework* que oferece autenticação, autorização e várias outras funcionalidades de segurança. Fornece mecanismos robustos para autenticação de usuários e controle de acesso baseado em permissões e papéis.

Protege aplicações web contra ataques comuns, como *Cross-Site Scripting*, *Cross-Site Request Forgery* e ataques de sessão.

Ainda, visando a utilização de boas práticas de código e uma abordagem estruturada e modularizada no projeto, foi utilizado o *design patterns Models, Repositories, Services* e *Controllers* (Collings, 2021) (explicado a seguir com exemplos de código do QA+).

Models (em português, modelos) são representações em código das entidades do banco de dados, utilizando o *framework* JPA do Spring para facilitar a persistência dos dados. No código da Listagem 1 é apresentado um exemplo de modelo do QA+, onde a classe Answer representa a tabela _answer.

Listagem 1 – Classe Answer

```
1  @Entity
2  @Table(name = "_answer")
3  public class Answer {
4      @Id
5      @GeneratedValue(strategy = GenerationType.IDENTITY)
6      private Long id;
7
8      @Column(columnDefinition = "TEXT")
9      private String text;
10
11     @ManyToOne
12     @JoinColumn(name = "quiz_id")
13     @JsonIgnore
14     private Quiz quiz;
15     // constructor, getters and setters
16 }
```

Fonte: Autoria própria (2024).

Por sua vez, ***repositories*** (em português, repositórios) são responsáveis pela interação direta com o banco de dados, abstraindo a lógica de acesso aos dados. O código da Listagem 2 é um exemplo de repositório do QA+ que estende a JpaRepository do Spring Data JPA para a entidade ClassRegistration, a qual fornece métodos como save, findById, findAll, deleteById, entre outros, sem a necessidade de implementação explícita, pois o Spring Data JPA já oferece implementações padrão para essas operações.

Listagem 2 – Interface ClassRegistrationRepository

```
1  @Repository
2  public interface ClassRegistrationRepository extends
3                   JpaRepository<ClassRegistration, Long> {
4      List<ClassRegistration> findClassRegistrationsByCourseId(Long courseId);
5  }
```

Fonte: Autoria própria (2024).

Já os ***services*** (em português, serviços) encapsulam a lógica de negócio da aplicação e são responsáveis por processar e manipular dados, aplicando regras de negócio específicas. No exemplo do código da Listagem 3 (um trecho de código do QA+) é utilizado um repositório para buscar um usuário por id e mapear para a classe UserDao. Caso o usuário não seja encontrado, uma exceção *ResourceNotFoundException* é lançada, a qual é tratada globalmente na aplicação para retornar uma resposta adequada ao cliente.

Listagem 3 – Class UserServiceImpl

```
1  @Service
2  public class UserServiceImpl implements UserService {
3
4      private final UserRepository userRepository;
5
6      @Override
7      public UserDao getUser(Long id) {
8          User user = userRepository.findById(id).orElseThrow(
9              () -> new ResourceNotFoundException("User with id:
10             " + id + " not found"));
11         return ObjectMapperUtils.map(user, UserDao.class);
12     }
13 }
```

Fonte: Autoria própria (2024).

Por fim, ***controllers*** (em português, controladores) são responsáveis por receber requisições HTTP, delegar o processamento para os serviços apropriados e retornar as respostas adequadas para o cliente. No exemplo do código da Listagem 4 (um trecho de código do QA+) o método getUser recebe uma requisição GET na URL /api/v1/user/id e utiliza o serviço UserService para buscar um usuário pelo ID fornecido. A anotação @PreAuthorize("hasRole('USER')") garante que apenas usuários autenticados que contenham no mínimo a role USER tenham acesso a esse endpoint, garantindo segurança na aplicação.

Listagem 4 – Class UserController

```
1  @RestController
2  @RequestMapping("/api/v1/user")
3  @RequiredArgsConstructor
4  public class UserController {
5
6      private final UserService userService;
7
8      @GetMapping("/{id}")
9      @PreAuthorize("hasRole('USER')")
10     public ResponseEntity<UserDao> getUser(@PathVariable Long id) {
11         return ResponseEntity.ok(userService.getUser(id));
12     }
13 }
```

Fonte: Autoria própria (2024).

5.1.4 Apresentação do QA+

Nesta seção apresentaremos a versão do QA+ implementada neste trabalho. Um possível fluxo de utilização do jogo é descrito abaixo.

Primeiramente, o usuário deve acessar[3] a tela de login do QA+ (Figura 3). Caso o usuário ainda não tenha uma conta, ele deverá então clicar em “Criar Conta” e proceder com o seu cadastro e acesso por meio da tela de criação de conta (Figura 4). Por sua vez, caso já tenha uma conta deverá realizar login a partir do informe de nome de usuário e senha.

Figura 3 – Tela de login

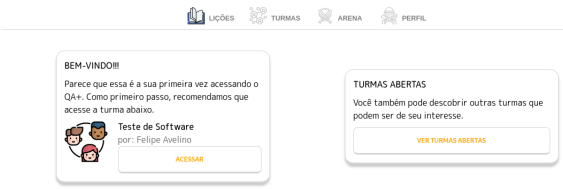
Fonte: Autoria própria (2024).

Figura 4 – Tela de criação de conta

Fonte: Autoria própria (2024).

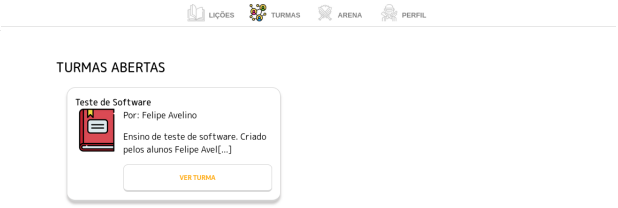
Após realizar login, o jogador é introduzido à tela inicial do QA+ (Figura 5), lhe sendo apresentadas duas opções: uma para acessar a turma “Teste de Software” (turma padrão do jogo, na qual já são disponibilizados capítulos, lições e quizzes sobre os tópicos da Seção 2.1) e outra para visualizar todas as turmas abertas (Figura 6).

Figura 5 – Tela de boas-vindas



Fonte: Autoria própria (2024).

Figura 6 – Tela de turmas abertas

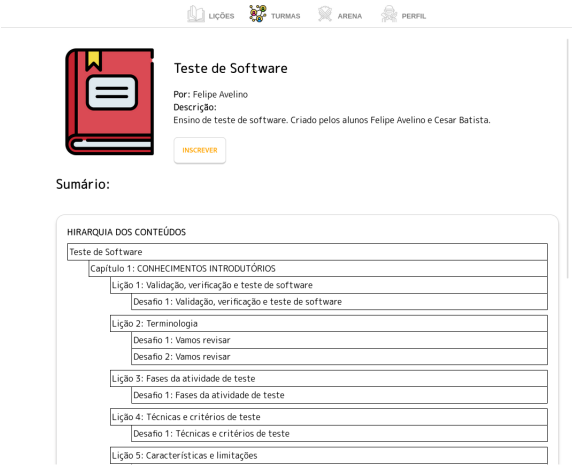


Fonte: Autoria própria (2024).

Ao acessar a turma “Teste de Software” o jogador visualizará o sumário da turma e uma opção para se inscrever nela (Figura 7). Após clicar em “Inscrever” será então redirecionado para a tela da turma (Figura 8), na qual constam o quadro de líderes, as estatísticas do jogador e o conteúdo da turma (capítulos, lições e quizzes).

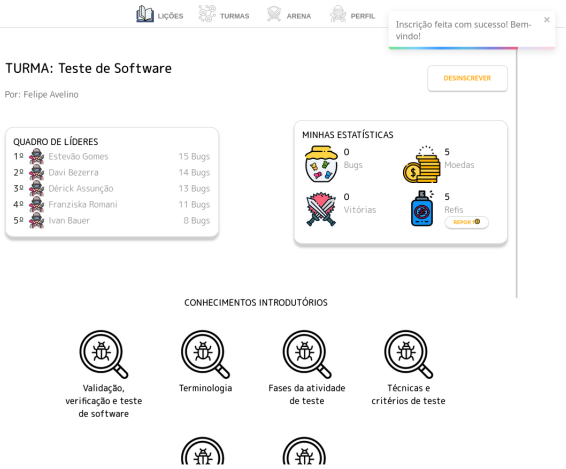
[3] O jogo QA+ ficou disponível/online apenas durante o período de avaliação deste trabalho. Para novas utilizações devem ser seguidas as instruções contidas nos repositórios indicados na Subseção 5.1.3.

Figura 7 – Tela de sumário da turma



Fonte: Autoria própria (2024).

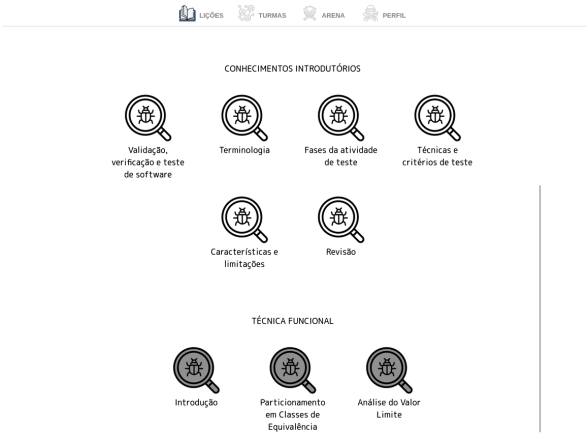
Figura 8 – Tela da turma



Fonte: Autoria própria (2024).

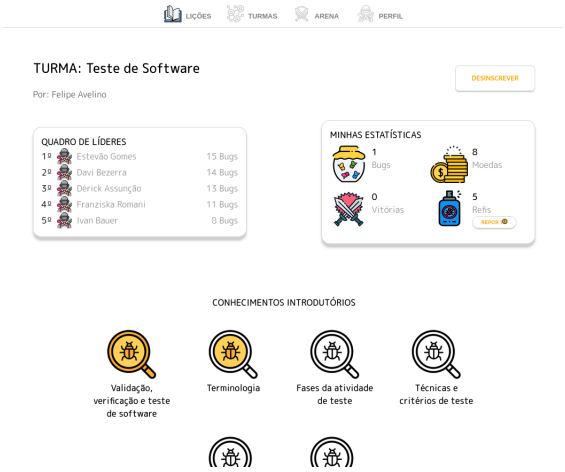
Quando o jogador acaba de se inscrever na turma os ícones das lições do primeiro capítulo estarão na cor branca, enquanto os das demais lições estarão na cor cinza (Figura 9). O ícone da lição na cor branca indica que ela está desbloqueada, mas ainda não foi acessada. Já a cor cinza indica que a lição está bloqueada (o desbloqueio ocorrerá após o acesso de todas as lições do capítulo anterior). Por sua vez, quando o jogador acessa uma lição o seu ícone é parcialmente colorido, sendo totalmente colorido quando os quizzes associados à lição são completados (vide exemplos na Figura 10).

Figura 9 – Lições bloqueadas e desbloqueadas



Fonte: Autoria própria (2024).

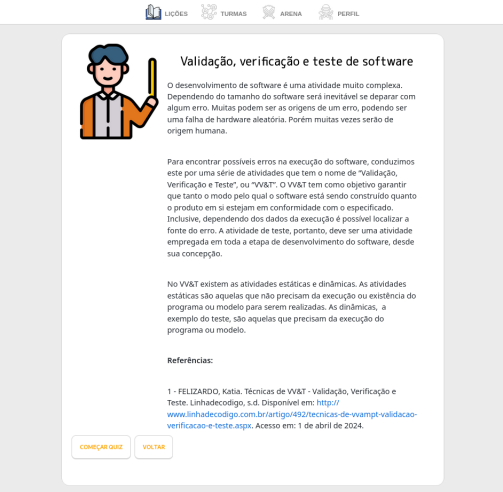
Figura 10 – Lições em andamento e finalizadas



Fonte: Autoria própria (2024).

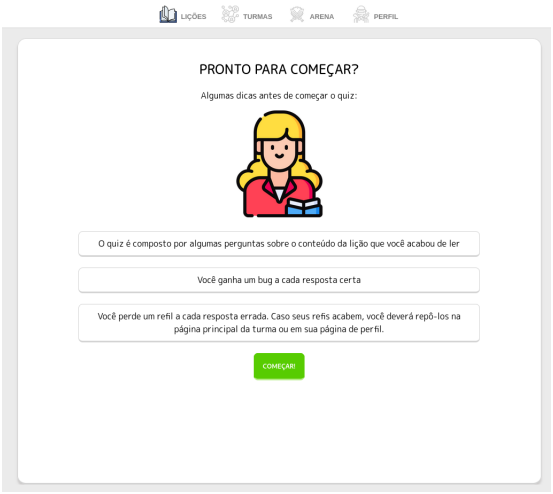
Quando o jogador clica em uma das lições ele é levado à tela de lição (Figura 11). Nela ele pode visualizar o conteúdo da lição e optar por retornar para a tela de turma ou começar os quizzes associados à lição. Ao clicar em “Começar Quiz” o usuário é redirecionado para uma tela de explicação das regras da funcionalidade de quiz (Figura 12).

Figura 11 – Tela de lição



Fonte: Autoria própria (2024).

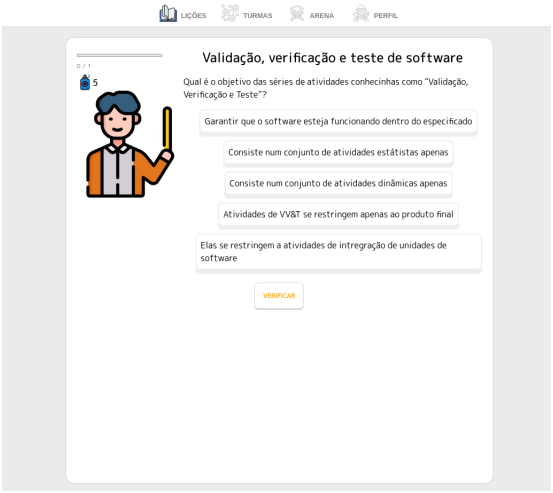
Figura 12 – Tela de quiz: regras



Fonte: Autoria própria (2024).

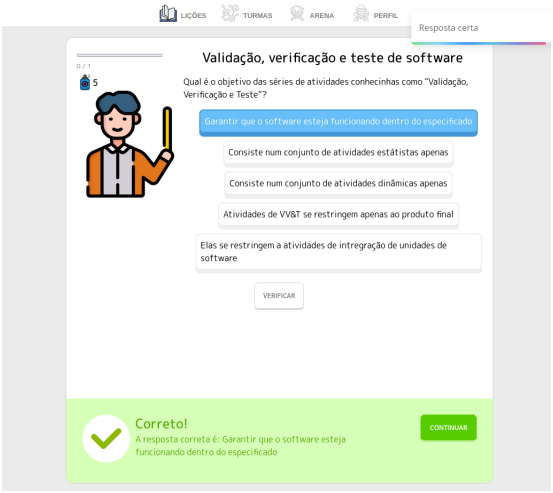
Estando na tela de explicação, ao clicar em “Continuar” é apresentada a tela do quiz (Figura 13), na qual é mostrada a pergunta do quiz, as alternativas de resposta e o botão “Verificar”. Após a seleção de uma alternativa de resposta deve-se clicar em “Verificar” para confirmar a escolha. Se a alternativa escolhida for a correta será indicado o acerto (Figura 14).

Figura 13 – Tela de quiz



Fonte: Autoria própria (2024).

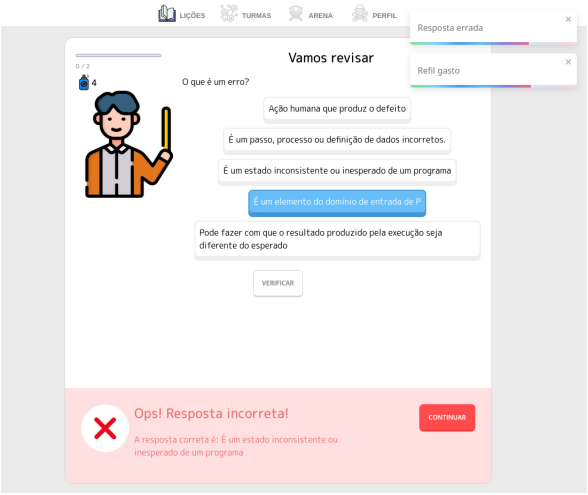
Figura 14 – Tela de quiz: resposta certa



Fonte: Autoria própria (2024).

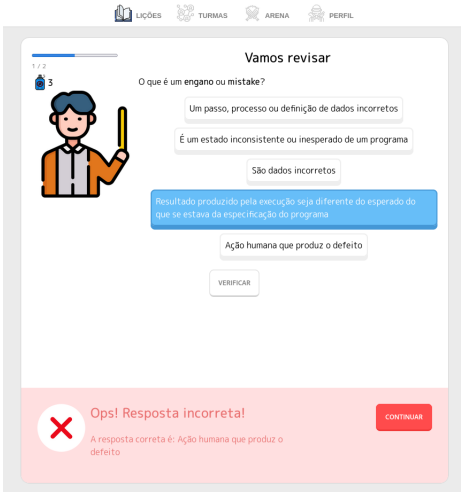
Por sua vez, se o jogador tiver escolhido a alternativa errada será automaticamente gasto um refil (se ainda houver algum disponível), sendo o jogador notificado sobre isso (Figura 15). Ainda, será indicado que a resposta escolhida estava incorreta e qual era a alternativa certa (Figura 16).

Figura 15 – Notificação de refil gasto



Fonte: Autoria própria (2024).

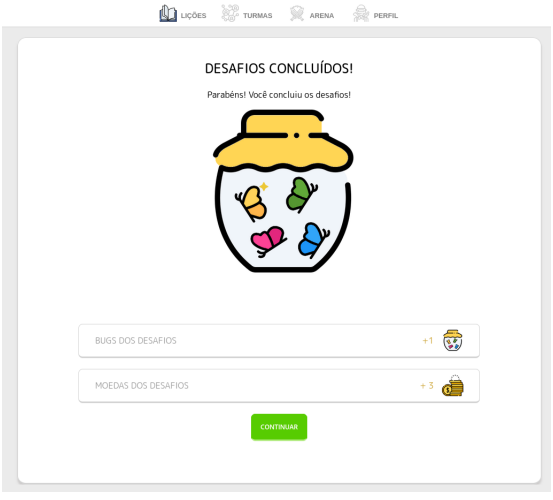
Figura 16 – Tela de quiz: resposta errada



Fonte: Autoria própria (2024).

Em sequência, se não houverem outros quizzes a serem resolvidos o jogador será redirecionado para a tela de conclusão do quiz (Figura 17). Cada resposta correta do jogador no quiz lhe dará um ponto de *bug* e duas moedas. Se o jogador errar uma pergunta e tiver gasto um refil ele ainda ganhará o ponto de bug, mas não ganhará as moedas. Se ele tiver errado a pergunta e não tiver gasto um refil (por não possuir mais nenhum disponível) ele não ganhará ponto de bug e tampouco moedas. Clicando em “Continuar” o jogador será redirecionado para a tela da turma.

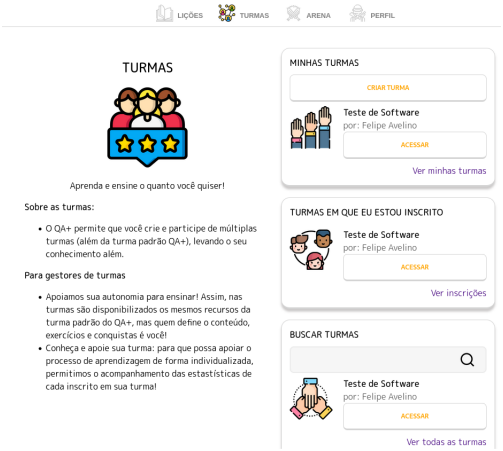
Figura 17 – Tela de quiz: quiz concluído



Fonte: Autoria própria (2024).

No QA+ também há a tela de turmas (Figura 18), que o jogador acessa clicando em “Turmas” na barra de navegação. Nela é possível acessar as turmas em que o jogador está inscrito (Figura 19), buscar por outras turmas para se inscrever nelas e, se o jogador for um gestor de turmas, visualizar as turmas criadas por ele.

Figura 18 – Tela de turmas



Fonte: Autoria própria (2024).

Figura 19 – Tela de inscrições

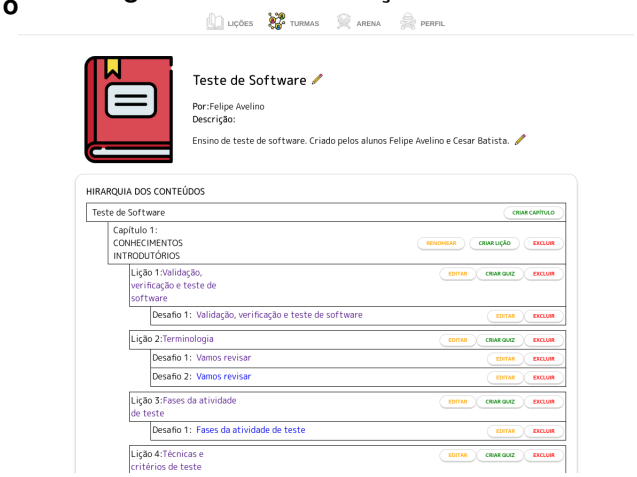
Fonte: Autoria própria (2024).

Se o gestor clicar em alguma das suas turmas, na tela de sumário da turma (Figura 20) serão exibidas opções para editar ou excluir a turma, bem como retornar para a tela anterior. Se ele clicar em “Editar Turma” será redirecionado para a tela de edição da turma (Figura 21), na qual existem opções para renomear a turma e associar uma descrição a ela, bem como criar, editar ou excluir capítulos, lições e quizzes.

Figura 20 – Sumário da turma (disponível para o gestor da turma)

Fonte: Autoria própria (2024).

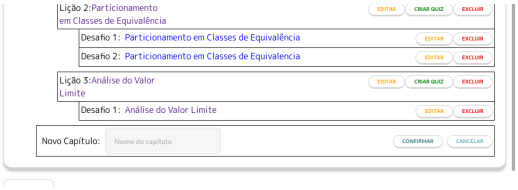
Figura 21 – Tela de edição da turma



Fonte: Autoria própria (2024).

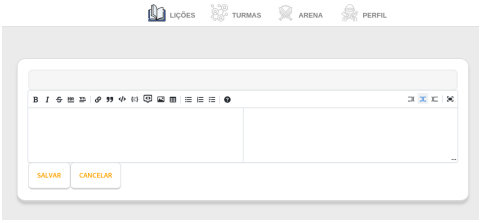
Para criar um capítulo deve-se clicar em “Criar capitulo”, inserir no respectivo campo o nome do capítulo e depois clicar em “Confirmar” (Figura 22). Por sua vez, para criar uma lição deve-se clicar em “Criar lição”. Ao fazer isso será exibida a tela de criação de lição (Figura 23), na qual há um editor de *markdown* para a inserção de conteúdo. Já para criar um quiz deve-se clicar em “Criar quiz”. Na tela de criação de quiz (Figura 24) também há um editor de *markdown* para inserção do enunciado do quiz, campos para inserção das alternativas de resposta e um componente para indicar qual é a alternativa correta.

Figura 22 – Criando capítulo



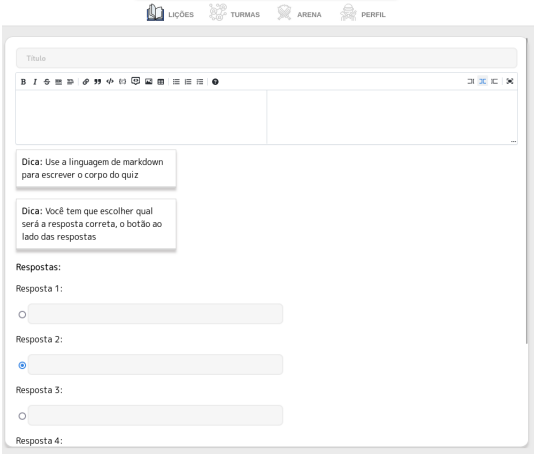
Fonte: Autoria própria (2024).

Figura 23 – Criando lição



Fonte: Autoria própria (2024).

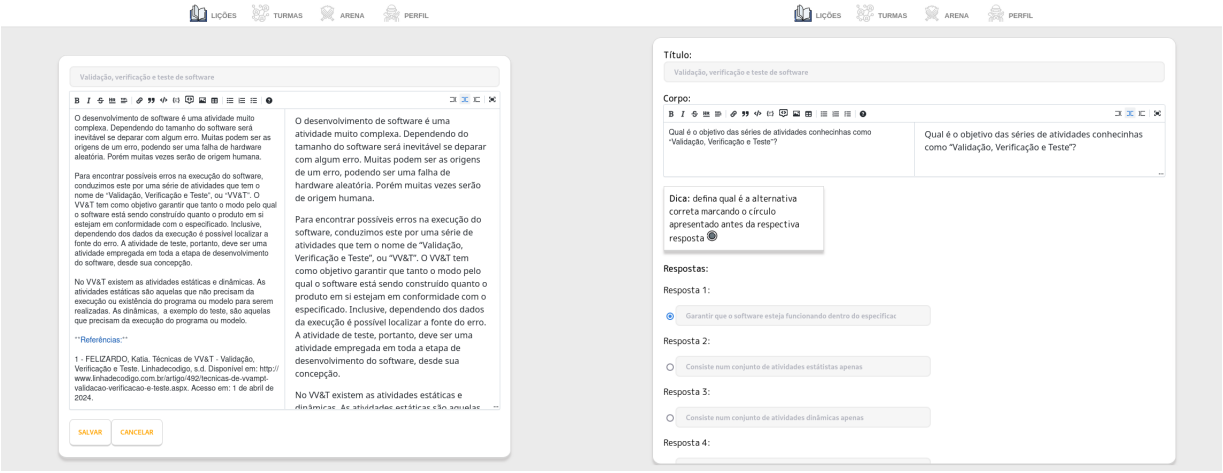
Figura 24 – Criando quiz



Fonte: Autoria própria (2024).

Caso seja necessário editar uma lição ou quiz isso poderá ser realizado por meio das opções de edição existentes no sumário da turma. As figuras 25 e 26 apresentam as telas de edição de lição e quiz, respectivamente.

Figura 25 – Tela de edição de lições



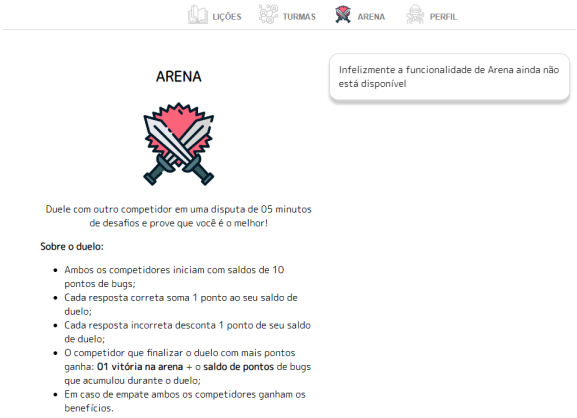
Fonte: Autoria própria (2024).

Figura 26 – Tela de edição de quizzes

Fonte: Autoria própria (2024).

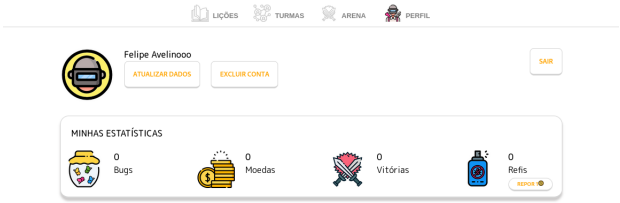
Por fim, na barra de navegação também existem as opções de acesso às telas de arena e perfil. Ao clicar em “Arena” o usuário acessará a tela inicial da arena (Figura 27), na qual será indicado que essa funcionalidade ainda não se encontra disponível. Por sua vez, ao clicar em “Perfil” o usuário acessará a tela de perfil (Figura 28), a qual mostra as estatísticas do usuário e permite com que esse atualize seus dados (troque de senha), exclua sua conta ou saia do jogo (ação que o redicionará para a tela de login).

Figura 27 – Tela de arena



Fonte: Autoria própria (2024).

Figura 28 – Tela de perfil do usuário



Fonte: Autoria própria (2024).

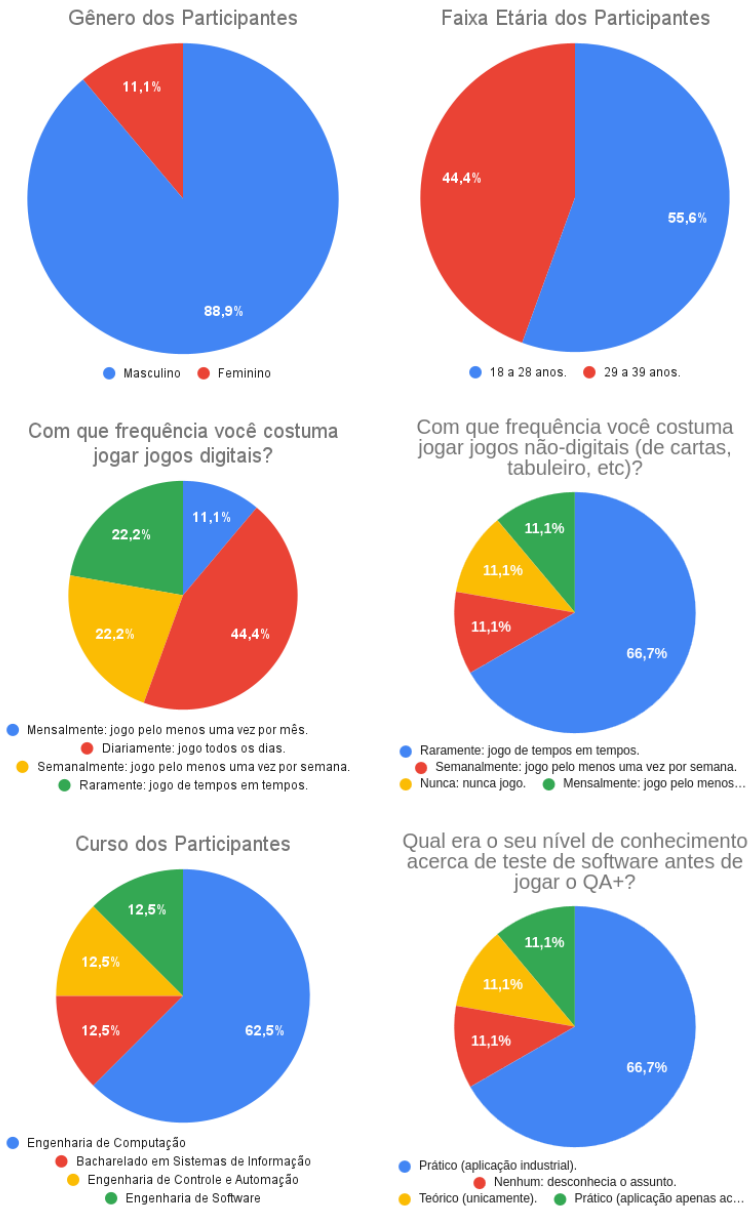
5.2 Fase 2 — Avaliação do QA+

Para a avaliação do QA+ conduzimos um estudo de caso, no qual participaram 9 alunos de graduação da UTFPR de forma online e remota entre 29/05/24 e 05/06/24.

Durante a execução do estudo os participantes foram guiados pelas informações de um formulário criado no *Google Forms* (vide Apêndice A). Inicialmente, os alunos acessaram o link do formulário e leram o conteúdo de sua primeira seção (Figura 33): um Termo de Consentimento Livre e Esclarecido (TCLE) explicando a pesquisa e com uma opção para aceite ou recusa de participação. Caso o aluno não aceitasse participar era apresentada uma página de agradecimento (Figura 38). Caso contrário, era apresentada uma seção com instruções do que deveria ser realizado (Figura 34): (1) acessar o jogo e explorar os recursos da turma padrão do QA+, completando ao menos uma lição e seu quiz; e (2) após isso, retornar para o formulário online e responder os questionários de caracterização do participante e avaliação do QA+.

O primeiro questionário apresentado (vide Figuras 35 e 36) visou coletar informações de caracterização dos participantes. Os resultados obtidos para esse questionário podem ser visualizados na Figura 29, por meio da qual é possível evidenciar que a maior parte dos respondentes foi do sexo masculino, estando suas idades distribuídas quase que igualmente entre as faixas etárias de “18 a 28 anos” e “29 a 39 anos”. Ainda, a maioria dos respondentes informou jogar diariamente jogos digitais (44,4%), mas raramente jogos não-digitais (66,7%). Isso indica uma provável familiaridade e preferência dos participantes por jogos digitais (caso do QA+). Já acerca dos cursos dos respondentes, todos foram da área de tecnologia da informação (como esperado), sendo a maioria dos respondentes do curso de Engenharia de Computação. Por fim, acerca do nível de conhecimento prévio em teste de software, a maioria dos respondentes (66,7%) já possuía conhecimento prático (industrial) sobre o assunto.

Figura 29 – Resultados das questões de caracterização dos participantes



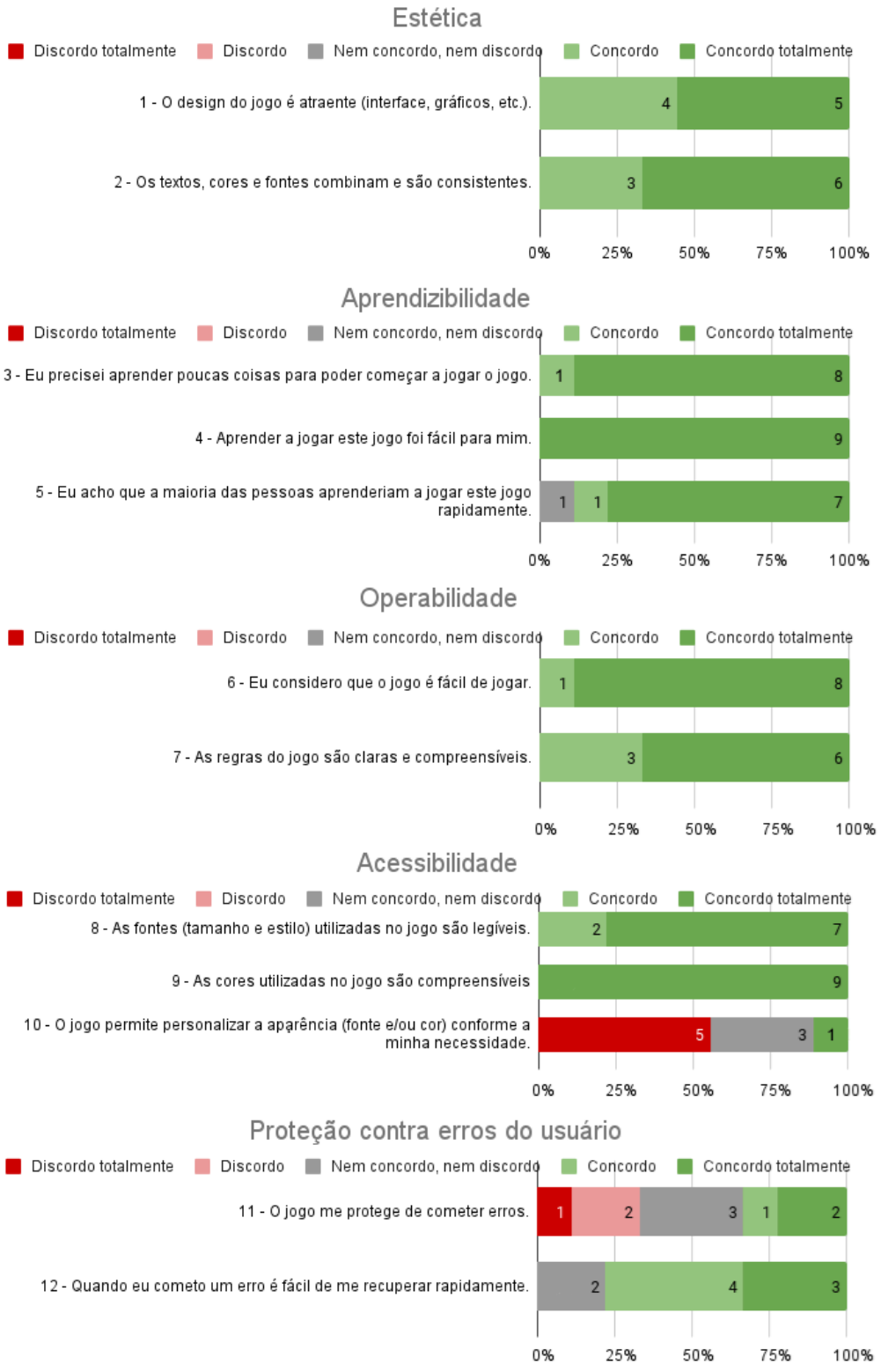
Fonte: Autoria própria (2024).

Por sua vez, o segundo questionário (Figura 37) foi composto pelas questões dispostas no Quadro 1 e, adicionalmente, uma questão aberta (para a qual não foram obtidas respostas) para informe de considerações adicionais dos participantes sobre o QA+.

Na Figura 30 são apresentados os resultados obtidos para as subdimensões de usabilidade do MEEGA+. Analisando os resultados evidenciou-se que: para as subdimensões “estética”, “aprendizibilidade” e “operabilidade” as respostas foram majoritariamente positivas, o que indica que o QA+ é esteticamente atraente, sendo também de fácil compreensão e uso; os resultados de “acessibilidade” foram positivos, com exceção do item 10, o que decorreu da não implementação de recursos de personalização de aparência nesta versão do QA+; para a subdimensão “proteção contra erros do usuário” foi evidenciado que, apesar de ter sido indicado

pela maioria dos participantes que a recuperação de erros é fácil e rápida (aspecto positivo), o jogo ainda precisa ser aprimorado no que tange a proteger o usuário de cometer erros.

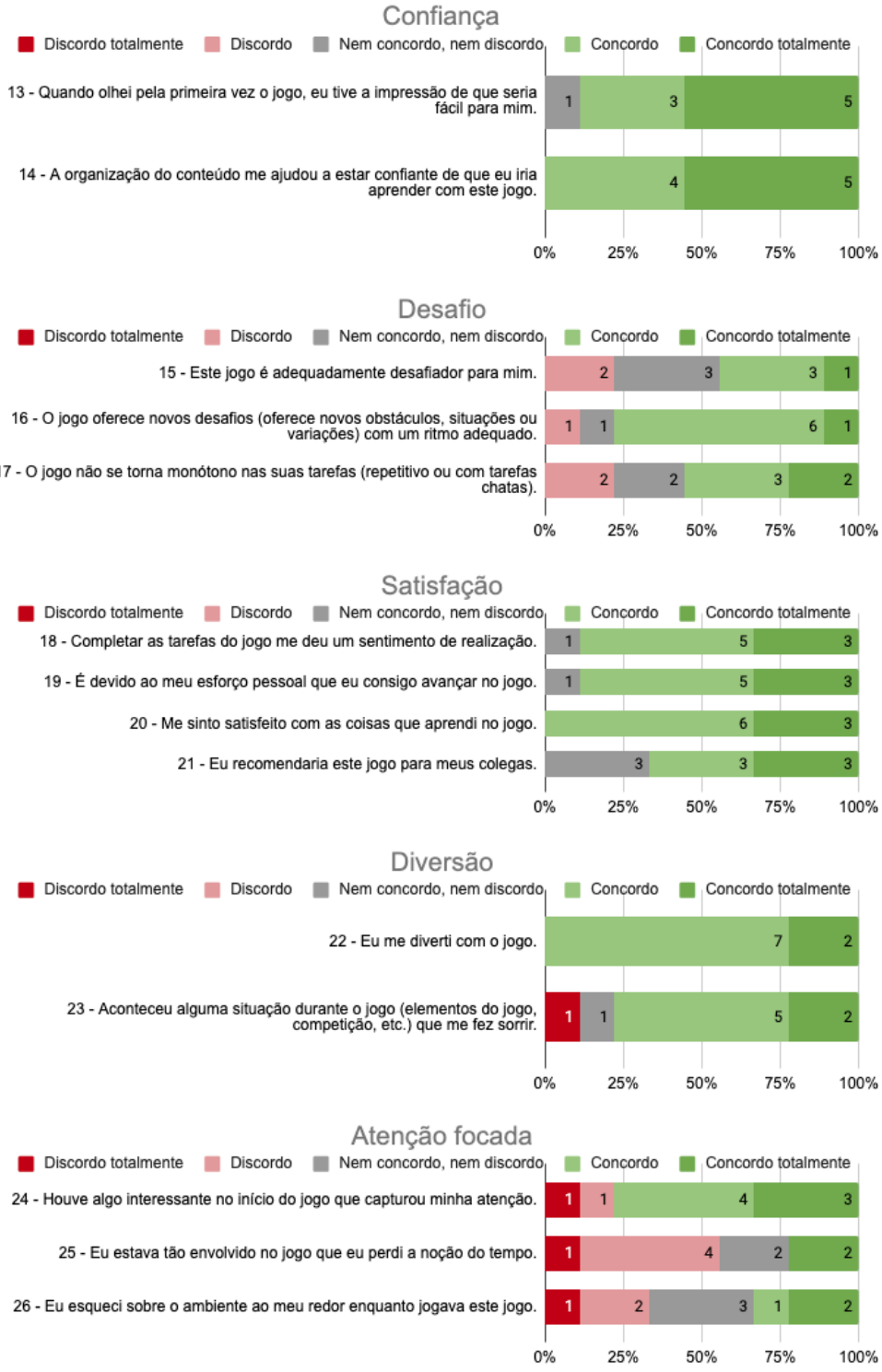
Figura 30 – Resultados da avaliação das subdimensões de usabilidade



Fonte: Autoria própria (2024).

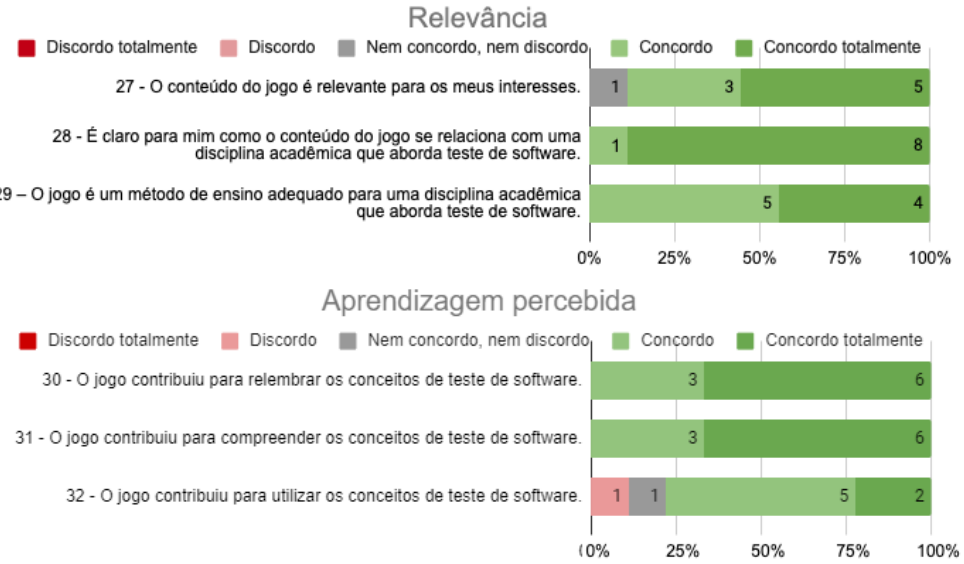
Por sua vez, nas Figuras 31 e 32 são apresentados os resultados obtidos para as subdimensões de experiência do jogador.

Figura 31 – Resultados da avaliação das subdimensões de experiência do jogador (parte 1)



Fonte: Autoria própria (2024).

Figura 32 – Resultados da avaliação das subdimensões de experiência do jogador (parte 2)



Fonte: Autoria própria (2024).

Analisando os resultados das Figuras 31 e 32 é possível evidenciar que: (1) para as subdimensões “desafio” e “atenção focada” as respostas se dividiram entre positivas, neutras e negativas, indicando a oportunidade de realização de melhorias no QA+ para que este se torne mais desafiador, bem como capture mais a atenção dos jogadores; e (2) para as subdimensões “confiança”, “satisfação”, “diversão”, “relevância” e “aprendizagem percebida” os resultados foram em sua maioria positivos, indicando que o QA+ desperta nos jogadores sentimentos como confiança, satisfação e diversão. Ainda, os resultados evidenciam que, na percepção dos participantes, o jogo é relevante para o ensino, contribuindo para que seus usuários relembrem, compreendam e utilizem conceitos de teste de software.

Por meio dos resultados expostos acima pode-se concluir que a versão inicial do QA+ foi bem aceita por alunos de graduação, o que — respondendo a questão de pesquisa deste trabalho, apresentada na Seção 1.2 — corresponde a um indício de resultado positivo em relação à aplicabilidade do QA+ como recurso educacional para apoiar o ensino de teste e qualidade de software no nível superior acadêmico.

5.3 Considerações finais do capítulo

Este capítulo apresentou e discutiu os resultados obtidos em cada fase do desenvolvimento do trabalho de conclusão de curso. O capítulo seguinte apresenta a conclusão do trabalho.

6 CONCLUSÃO

Neste trabalho desenvolvemos e avaliamos uma versão inicial do QA+, um jogo projetado para apoiar o ensino de teste e qualidade de software no nível superior acadêmico. Ao longo do estudo exploramos o potencial da gamificação como ferramenta pedagógica, buscando mitigar os desafios enfrentados por alunos e educadores no processo de ensino-aprendizagem.

A avaliação da versão inicial do QA+ revelou que o jogo foi considerado esteticamente atraente e de fácil compreensão e uso pelos estudantes, lhes tendo causado igualmente sentimentos como confiança, satisfação e diversão, aspectos essenciais para a manutenção do engajamento e interesse dos alunos. Ainda, os dados coletados mostraram que o jogo foi apontado como relevante pelos discentes, lhes tendo auxiliado a relembrar, compreender e aplicar conceitos de teste de software. Tais resultados consistem em um indício positivo em relação à aplicabilidade do QA+ como recurso educacional para apoiar o ensino de teste e qualidade de software no nível superior acadêmico.

Ao mesmo tempo, por meio da avaliação do jogo também foi possível evidenciar que há espaço para tornar as atividades do QA+ mais desafiadoras aos alunos, capturar ainda mais a sua atenção e lhes auxiliar a não cometer erros durante a utilização do jogo.

Por fim, conclui-se que este trabalho consiste em uma contribuição relevante para o contexto de ensino-aprendizagem de teste e qualidade de software. Os resultados iniciais do QA+ sugerem que há viabilidade para a expansão e refinamento do jogo, bem como para novas pesquisas envolvendo teste de software, educação e gamificação.

6.1 Limitações e dificuldades encontradas

A condução desta pesquisa encontrou as seguintes limitações e dificuldades:

- Devido a questões de complexidade técnica e limitação de tempo algumas funcionalidades originalmente previstas para o QA+ não puderam ser implementadas. Ainda, foram enfrentadas dificuldades na criação de lições com conteúdo multimídia, principalmente no que se refere à inclusão e ao manuseio de imagens via links externos e na inserção da lista de referências na lateral da lição (a qual acabou sendo inserida textualmente no final das lições). Essas dificuldades exigiram adaptações que podem ter comprometido a experiência do usuário.
- Devido a restrições de tempo, a avaliação do QA+ sofreu limitações em relação à quantidade de respostas obtidas e diversificação da amostra.
- A análise dos resultados das questões 28 e 29 — relacionadas à aplicabilidade do jogo em uma disciplina acadêmica — pode ter sofrido limitações, uma vez que essas questões foram respondidas apenas por discentes (e não docentes).

6.2 Oportunidades de trabalhos futuros

Identificamos como oportunidades de trabalhos futuros:

- A continuação do desenvolvimento do QA+ por meio de:
  - Implementação dos requisitos originalmente previstos por Belinski (2021), mas não atendidos (total ou parcialmente) neste estudo, a exemplo das funcionalidades de: (a) desafios diários, de modo a incentivar o uso contínuo do jogo, promovendo uma prática constante e reforçando o aprendizado ao longo do tempo; e (b) duelos entre jogadores, para manter o engajamento e o interesse dos usuários.
  - Melhorias em aspectos para os quais foram obtidos resultados não tão positivos na avaliação da versão inicial do QA+. Isso pode ser realizado a partir de: (a) disponibilização de questões e desafios mais complexos e variados, que estimulem o pensamento crítico e a resolução de problemas; (b) implementação de opções de personalização, para adaptar o jogo às preferências individuais dos usuários (isso pode incluir a personalização de avatares, temas e níveis de dificuldade); e (c) aprimoramento dos recursos de proteção do usuário contra erros.
  - Incorporação de mais animações, ilustrações e outros elementos visuais que tornem a experiência de aprendizagem ainda mais atraente e envolvente.
- A condução de novos estudos de avaliação do QA+ (visando obter percepções adicionais e a generalização dos resultados):
  - Com amostras mais diversificadas (em relação a sexo, conhecimento prévio do assunto, curso, instituição, etc) e numerosa de alunos.
  - Com docentes da área de Engenharia de Software.
  - Com pedagogos (para fins de avaliação de método e didática).
  - Comparando os resultados do QA+ (em relação a seu impacto na motivação e aprendizado dos alunos) com os obtidos por outras soluções voltadas ao ensino de teste e qualidade de software (gamificadas ou não).

REFERÊNCIAS

BELINSKI, V. **QA+: Um Jogo Para o Ensino de Teste e Qualidade de Software**. Curitiba, 2021. Disponível em: http://dx.doi.org/10.13140/RG.2.2.12510.45125. Acesso em: 05 jun. 2024.

COLLINGS, T. **Controller-Service-Repository**. 2021. Disponível em: https://tom-collings.medium.com/controller-service-repository-16e29a4684e5. Acesso em: 20 jun. 2024.

COSTA, I.; OLIVEIRA, S. Uma avaliação do uso de gamificação para apoiar o ensino e aprendizagem de testes exploratórios. *In*: **XII Computer on the Beach**. [*S.l.*: *s.n.*], 2021. p. 027–034.

DELAMARO, M. E.; MALDONADO, J. C.; JINO, M. **Introdução ao teste de software**. 2. ed. Rio de Janeiro: Elsevier, 2016. ISBN 978-85-352-8352-5.

ELBAUM, S. *et al.* Bug hunt: Making early software testing lessons engaging and affordable. *In*: **29th International Conference on Software Engineering (ICSE'07)**. [*S.l.*: *s.n.*], 2007. p. 688–697.

FADEL, L. M. *et al.* **Gamificação na educação**. 1. ed. São Paulo: Pimenta Cultural, 2014. ISBN 978-85-66832-13-6.

FEBBRARO, O. *et al.* Unit testing in aspide. **Applications of Declarative Programming and Knowledge Management**, Springer, 2013.

GAROUSI, V. *et al.* Software-testing education: A systematic literature mapping. **Journal of Systems and Software**, v. 165, 2020.

MACêDO, K. S. **As aventuras de Jack Test: jogo educacional para o apoio ao ensino de teste de software**. 2014. Tese (Trabalho de Conclusão de Curso) — Universidade Estadual do Sudoeste da Bahia, 2014.

MELO, S. M. *et al.* Testing education: a survey on a global scale. *In*: **Proceedings of the 34th Brazilian Symposium on Software Engineering**. New York: Association for Computing Machinery, 2020. p. 554–563.

MYERS, G. J.; BADGETT, T.; SANDLER, C. **The Art of Software Testing**. 3. ed. New Jersey: John Wiley & Sons, Inc, 2012. 254 p. ISBN 978-1-118-03196-4.

PETRI, G.; WANGENHEIM, C. G. von; BORGATTO, A. F. MEEGA+: Um Modelo para a Avaliação de Jogos Educacionais para o ensino de Computação. **Revista Brasileira de Informática na Educação**, v. 27, n. 3, 2019.

PRINCE, M. Does active learning work? A review of the research. **Journal of Engineering Education**, v. 93, n. 3, p. 223–231, 2004.

SOUZA, J.; BORGES, S.; DURELLI, V. Gamificação aplicada à aprendizagem de critérios de teste de software. *In*: **Anais Estendidos do XXI Simpósio Brasileiro de Jogos e Entretenimento Digital**. Porto Alegre, RS, Brasil: SBC, 2022. p. 775–784.

VALLE, P. **Jogos educacionais: uma contribuição para o ensino de teste de software**. 2017. Dissertação (Mestrado) — Universidade de São Paulo, 2017.

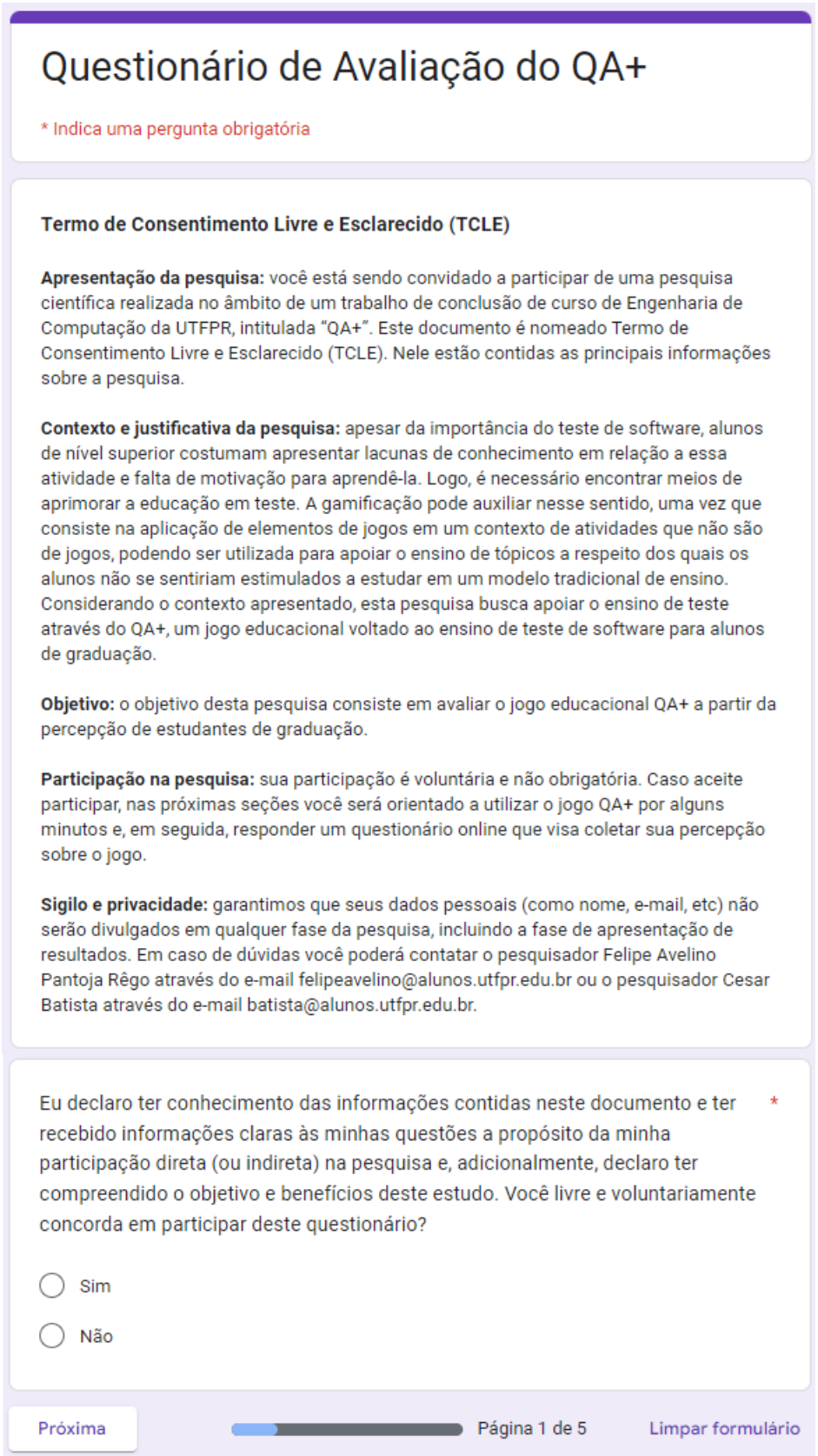
VALLE, P. H. D.; BARBOSA, E. F.; MALDONADO, J. C. Um Mapeamento Sistemático Sobre Ensino de Teste de Software. **Anais do XXVI Simpósio Brasileiro de Informática na Educação**, v. 26, 2015.

WAZLAWICK, R. S. **Engenharia de Software: conceitos e praticas**. [*S.l.*]: Elsevier, 2013.

ZICHERMANN, G.; CUNNINGHAM, C. **Gamification by Design: Implementing Game Mechanics in Web and Mobile Apps**. [*S.l.*]: O'Reilly Media, Inc., 2011.

APÊNDICE A – Formulário de Coleta de Dados

Figura 33 – Formulário de Coleta de Dados — Página 1



Questionário de Avaliação do QA+

* Indica uma pergunta obrigatória

Termo de Consentimento Livre e Esclarecido (TCLE)

Apresentação da pesquisa: você está sendo convidado a participar de uma pesquisa científica realizada no âmbito de um trabalho de conclusão de curso de Engenharia de Computação da UTFPR, intitulada "QA+". Este documento é nomeado Termo de Consentimento Livre e Esclarecido (TCLE). Nele estão contidas as principais informações sobre a pesquisa.

Contexto e justificativa da pesquisa: apesar da importância do teste de software, alunos de nível superior costumam apresentar lacunas de conhecimento em relação a essa atividade e falta de motivação para aprendê-la. Logo, é necessário encontrar meios de aprimorar a educação em teste. A gamificação pode auxiliar nesse sentido, uma vez que consiste na aplicação de elementos de jogos em um contexto de atividades que não são de jogos, podendo ser utilizada para apoiar o ensino de tópicos a respeito dos quais os alunos não se sentiriam estimulados a estudar em um modelo tradicional de ensino. Considerando o contexto apresentado, esta pesquisa busca apoiar o ensino de teste através do QA+, um jogo educacional voltado ao ensino de teste de software para alunos de graduação.

Objetivo: o objetivo desta pesquisa consiste em avaliar o jogo educacional QA+ a partir da percepção de estudantes de graduação.

Participação na pesquisa: sua participação é voluntária e não obrigatória. Caso aceite participar, nas próximas seções você será orientado a utilizar o jogo QA+ por alguns minutos e, em seguida, responder um questionário online que visa coletar sua percepção sobre o jogo.

Sigilo e privacidade: garantimos que seus dados pessoais (como nome, e-mail, etc) não serão divulgados em qualquer fase da pesquisa, incluindo a fase de apresentação de resultados. Em caso de dúvidas você poderá contatar o pesquisador Felipe Avelino Pantoja Rêgo através do e-mail felipeavelino@alunos.utfpr.edu.br ou o pesquisador Cesar Batista através do e-mail batista@alunos.utfpr.edu.br.

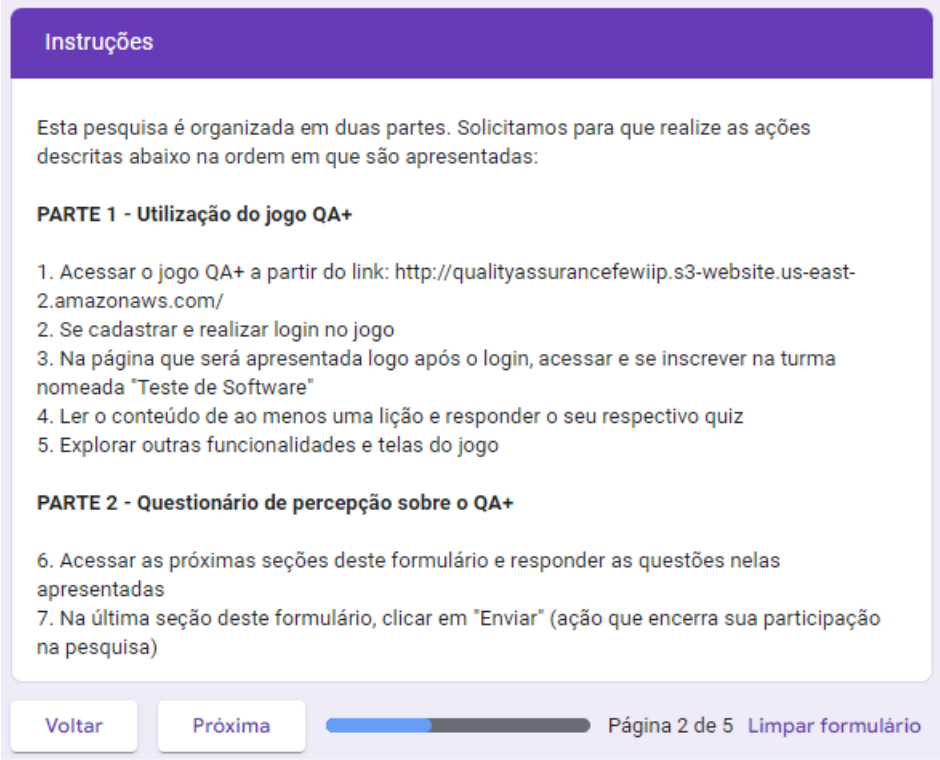
Eu declaro ter conhecimento das informações contidas neste documento e ter recebido informações claras às minhas questões a propósito da minha participação direta (ou indireta) na pesquisa e, adicionalmente, declaro ter compreendido o objetivo e benefícios deste estudo. Você livre e voluntariamente concorda em participar deste questionário? *

- ◯ Sim
- ◯ Não

Próxima Página 1 de 5 Limpar formulário

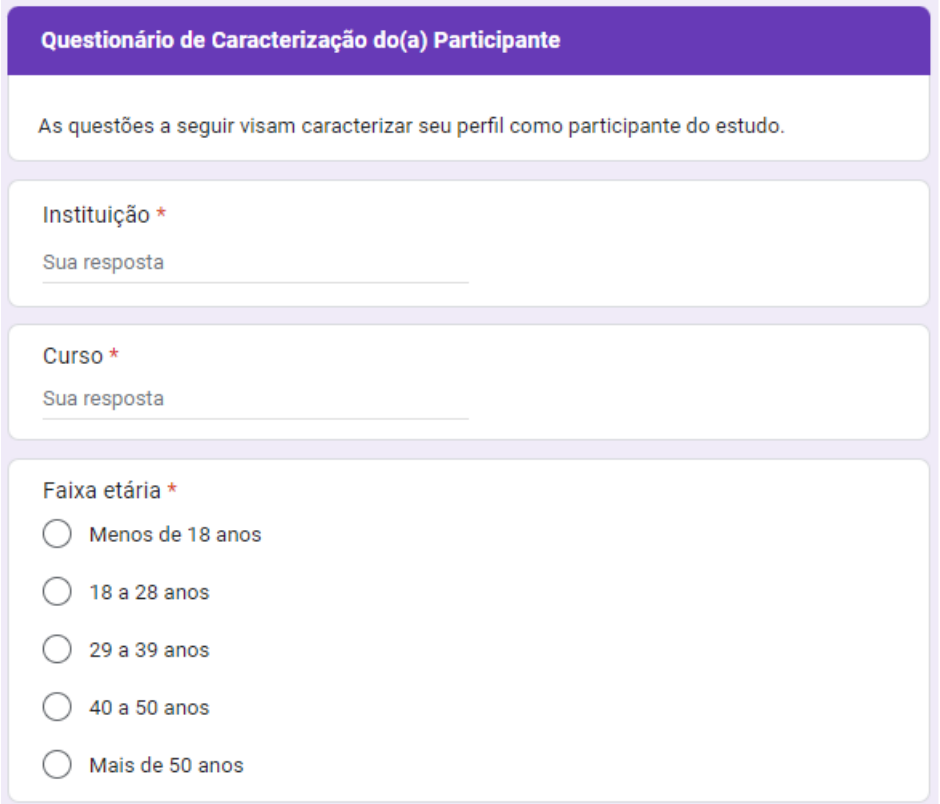
Fonte: Autoria própria (2024).

Figura 34 – Formulário de Coleta de Dados — Página 2



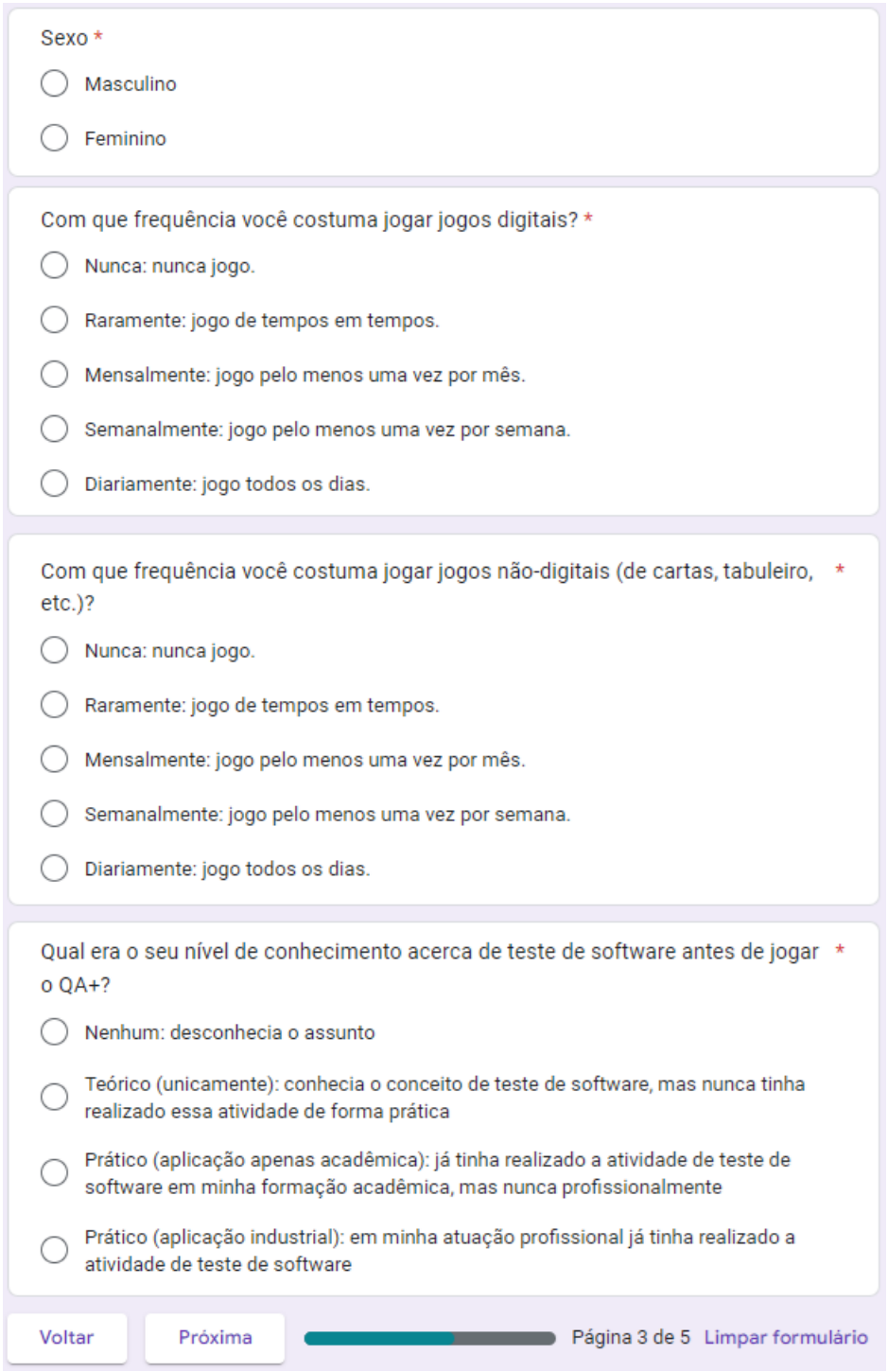
Fonte: Autoria própria (2024).

Figura 35 – Formulário de Coleta de Dados — Página 3 (parte 1)



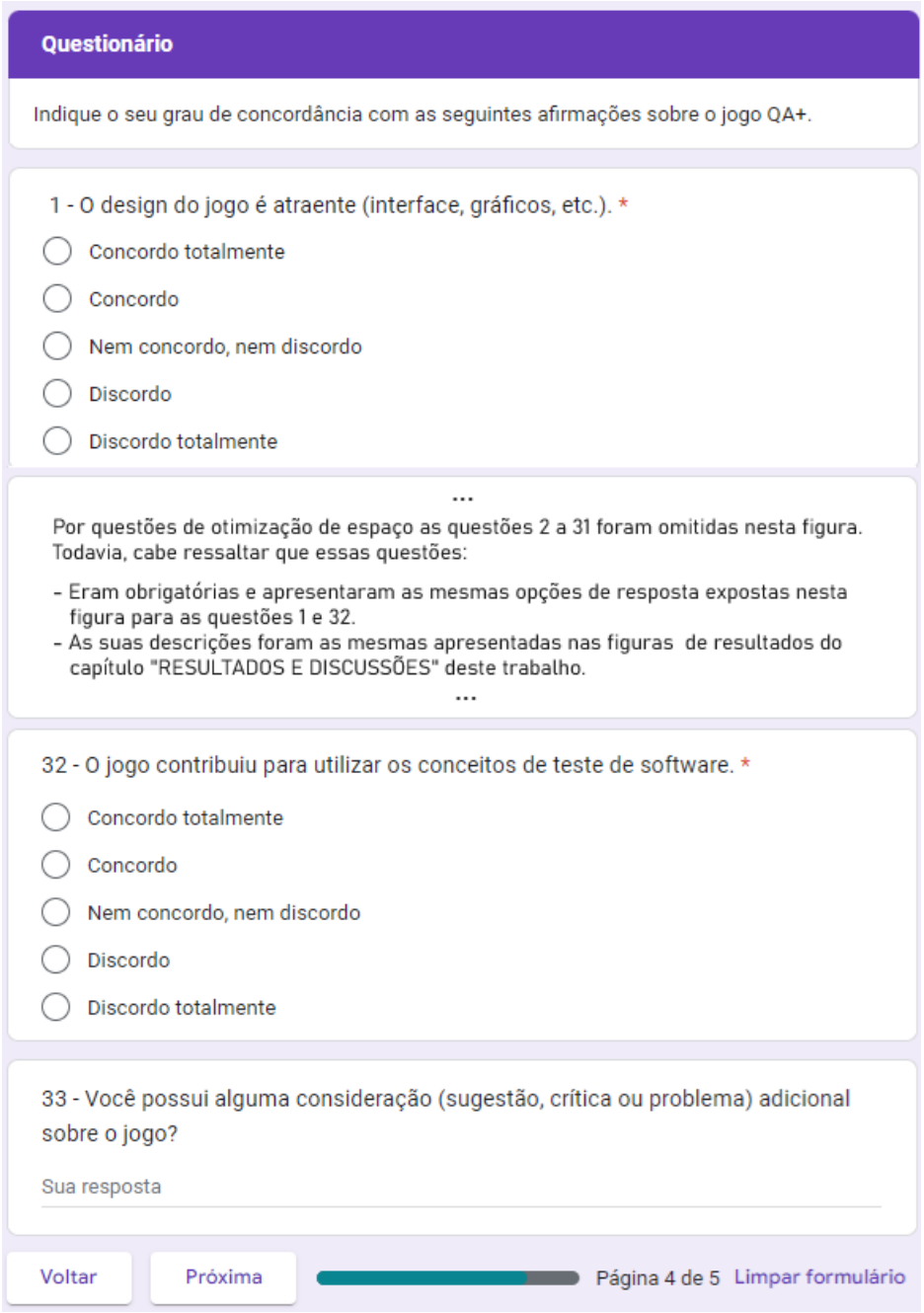
Fonte: Autoria própria (2024).

Figura 36 – Formulário de Coleta de Dados — Página 3 (parte 2)



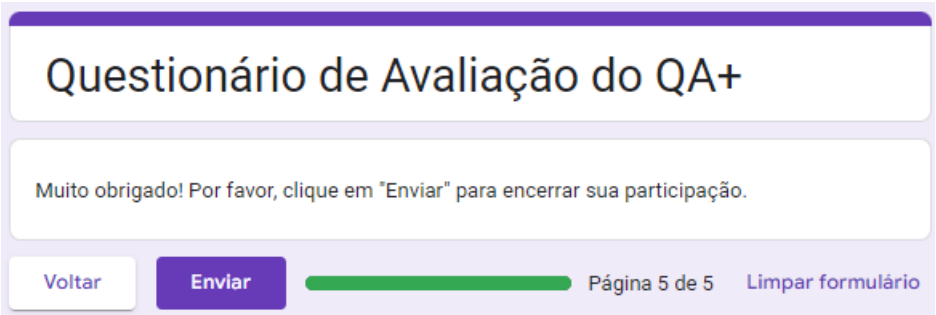
Fonte: Autoria própria (2024).

Figura 37 – Formulário de Coleta de Dados — Página 4



Fonte: Autoria própria (2024).

Figura 38 – Formulário de Coleta de Dados — Página 5



Fonte: Autoria própria (2024).