UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de São José do Rio Preto

Sabrina de Oliveira Guizilini

CodeGuardians:

Um jogo educacional para estimular o aprendizado de testes funcionais de software

São José do Rio Preto
2024

Sabrina de Oliveira Guizilini

CodeGuardians:

Um jogo educacional para estimular o aprendizado de testes funcionais de software

Trabalho de Conclusão de Curso (TCC) apresentado como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação, junto ao Conselho de Curso de Bacharelado em Ciência da Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista “Júlio de Mesquita Filho”, Câmpus de São José do Rio Preto.

Orientadora: Profa. Dra. Rogéria Cristiane Gratão de Souza

São José do Rio Preto
2024

G969c

Guizilini, Sabrina de Oliveira

CodeGuardians: Um jogo educacional para estimular o aprendizado de testes funcionais de software / Sabrina de Oliveira Guizilini. -- São José do Rio Preto, 2024

117 p.

Trabalho de conclusão de curso (Bacharelado - Ciência da Computação) - Universidade Estadual Paulista (UNESP), Instituto de Biociências Letras e Ciências Exatas, São José do Rio Preto

Orientadora: Rogéria Cristiane Gratão de Souza

1. Testes de software. 2. Jogo educacional. 3. Técnica funcional de testes. I. Título.

Sabrina de Oliveira Guizilini

CodeGuardians:

Um jogo educacional para estimular o aprendizado de testes funcionais de software

Trabalho de Conclusão de Curso (TCC) apresentado como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação, junto ao Conselho de Curso de Bacharelado em Ciência da Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista “Júlio de Mesquita Filho”, Câmpus de São José do Rio Preto.

Comissão Examinadora

Profª. Drª. Rogéria Cristiane Gratão de Souza
UNESP – Câmpus de São José do Rio Preto
Orientadora

Profª. Drª. Adriana Barbosa Santos
UNESP – Câmpus de São José do Rio Preto

Prof. Dr. Diego Renan Bruno
UNESP – Câmpus de São José do Rio Preto

São José do Rio Preto
2024

AGRADECIMENTOS

Agradeço a Deus por ter me dado força e sabedoria para conseguir concluir este projeto.

RESUMO

O teste de software é uma atividade essencial para garantir o correto funcionamento e a qualidade dos produtos de software. Porém, há uma escassez de profissionais qualificados nessa área, decorrente da desmotivação dos estudantes e da falta de atenção adequada por parte do sistema educacional. Nesse contexto, o uso de jogos educacionais se apresenta como uma abordagem eficaz para auxiliar o processo de ensino, uma vez que estes jogos podem desempenhar um papel fundamental no aumento da motivação e na facilitação do processo de aprendizado. Sendo assim, uma maneira de estimular e facilitar o aprendizado de testes de software é utilizar jogos educacionais. Considerando que a técnica funcional de testes é a mais utilizada nas empresas brasileiras, foram analisados os jogos educacionais existentes que a incorporam em seu conteúdo. Por meio das carências identificadas nos jogos analisados e dos requisitos almejados para um jogo educacional de testes funcionais de software, foi desenvolvido um jogo educacional, denominado *CodeGuardians*, que aborda as estratégias fundamentais da técnica funcional de testes de software aplicadas aos diferentes níveis de teste, em um ambiente de simulações práticas e realistas para a elaboração e execução dos testes. Para avaliar sua adequação, 17 estudantes do ensino superior matriculados em cursos da área da computação, que estão cursando ou já cursaram a disciplina de Engenharia de Software, jogaram o jogo educacional e responderam a um questionário online. Os resultados indicam que o jogo, além de proporcionar uma experiência lúdica e envolvente, aumentou a motivação dos jogadores para o aprendizado de testes de software e facilitou a compreensão tanto da teoria quanto da aplicação prática dos critérios da técnica funcional e dos diferentes níveis de teste. Isso confirma a eficácia do jogo como ferramenta educacional e sua contribuição significativa para o ensino de testes de software.

Palavras-chave: Testes de software. Jogo educacional. Técnica funcional de testes.

ABSTRACT

Software testing is an essential activity to ensure the correct functioning and quality of software products. However, there is a shortage of qualified professionals in this area, stemming from the lack of motivation among students and the inadequate attention given by the educational system. In this context, the use of educational games emerges as an effective approach to support the teaching process, as these games can play a key role in increasing motivation and facilitating learning. Therefore, one way to stimulate and facilitate the learning of software testing is through the use of educational games. Considering that the functional testing technique is the most commonly used in Brazilian companies, existing educational games incorporating it in their content were analyzed. Through the gaps identified in the analyzed games and the desired requirements for an educational game focused on functional software testing, an educational game called CodeGuardians was developed, addressing the fundamental strategies of the functional testing technique applied to different levels of testing in a practical and realistic simulation environment for the design and execution of tests. To assess its effectiveness, 17 higher education students enrolled in computer science-related courses, who are currently taking or have already taken the Software Engineering course, played the educational game and answered an online questionnaire. The results indicate that the game, in addition to providing a fun and engaging experience, increased the players' motivation to learn software testing and facilitated the understanding of both the theory and practical application of functional technique criteria and different testing levels. This confirms the effectiveness of the game as an educational tool and its significant contribution to the teaching of software testing.

Keywords: Software testing. Educational game. Functional testing technique.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CISQ	*Consortium for Information & Software Quality*
IEEE	*Institute of Electrical and Eletronics Engineers*
HTML	*HyperText Markup Language*
CSS	*Cascading Style Sheets*
SPA	*Single Page Application*
MEEGA	*Model for evaluating educational games*
MVVM	*Model-View-ViewModel*

SUMÁRIO

1 INTRODUÇÃO

Neste capítulo, são delineados os detalhes do trabalho, com o propósito de destacar sua relevância. Na seção 1.1, é apresentado o contexto do estudo, incluindo as hipóteses adotadas para sua realização. Na seção 1.2, tem-se a justificativa e motivação, destacando a importância e o embasamento acadêmico que fundamentam a escolha do tema específico. Na seção 1.3 é descrito o objetivo, tanto geral, quanto específico, definindo as metas do projeto. A metodologia empregada para o desenvolvimento do trabalho é discutida na seção 1.4. Na seção 1.5 é abordada a exequibilidade do trabalho. Por fim, na seção 1.6, tem-se a estrutura da monografia.

1.1 Contexto do trabalho

O *software* é encontrado em praticamente todos os dispositivos utilizados nos dias de hoje, o que evidencia sua ampla presença na sociedade. Dessa forma, o *software* desenvolvido atualmente tem um impacto significativo na vida das pessoas e das empresas, podendo influenciar positivamente na eficiência das operações ou causar frustrações e prejuízos (Myers, Sandler, Badgett, 2012). Com o aumento da integração de *software* em todos os aspectos da vida diária, a preocupação com a qualidade dos sistemas de *software* também cresceu. Nesse contexto, existem diversas atividades que podem ser incorporadas ao processo de desenvolvimento de *software* com o objetivo de garantir a qualidade do produto, sendo uma delas o teste de *software* (Pressman, Maxim, 2021).

O propósito do teste de *software* é demonstrar a funcionalidade pretendida de um programa e identificar defeitos antes de sua utilização. Durante o teste, o programa é executado com dados simulados e os resultados são analisados em busca de falhas e aspectos não funcionais (Sommerville, 2011). Quando os testes identificam falhas e os defeitos que causaram tais falhas são corrigidos adequadamente, a qualidade do *software* é sempre aprimorada (Graham *et al*., 2008).

De acordo com Pressman, Maxim (2021), as diversas estratégias de teste de *software* apresentadas na literatura compartilham algumas características genéricas, como a importância de se iniciar os testes no nível de componentes avançando para a integração do sistema completo, de forma a abranger tanto testes de baixo nível,

para validar a implementação de pequenas funções e algoritmos, quanto testes de alto nível, para verificar a conformidade com os requisitos do cliente.

Ademais, outra característica importante mencionada por Pressman, Maxim (2021), é que diferentes técnicas de teste são adequadas para variadas abordagens de engenharia de *software* e em momentos distintos durante o processo de desenvolvimento. Existem duas principais técnicas de teste: funcional (caixa-preta) e estrutural (caixa-branca). Na abordagem funcional o *software* é testado com base em suas especificações, sem considerar sua estrutura interna, enquanto na abordagem estrutural, os testes se baseiam na análise da estrutura interna do programa (Myers, Sandler, Badgett, 2012). Vale ressaltar a relevância da técnica funcional de teste, pois, embora todas as técnicas de teste sejam úteis e complementares (Graham *et al*., 2008), estudos sugerem que a abordagem funcional é mais eficiente na detecção de defeitos que a abordagem estrutural (Basili, Selby, 1987; Kamsties, Lott, 1995). Além disso, o trabalho de Santos *et al.* (2022) mostra que a abordagem funcional é a mais utilizada entre as empresas de *software* brasileiras.

Atualmente, para exercer atividades relacionadas a testes de *software*, a indústria requer profissionais de teste devidamente capacitados, cuja formação é responsabilidade do sistema educacional. Contudo, uma lacuna persiste entre as exigências da indústria em termos de conhecimento de teste e o que é ensinado nas instituições de ensino superior (Blanco *et al.*, 2023). Nesse contexto, Valle, Barbosa, Maldonado (2015) afirmam que a indústria de testes de *software* depara-se com uma escassez de profissionais qualificados, o que decorre principalmente da ineficácia das abordagens teóricas tradicionais no ensino, havendo a necessidade de ambientes práticos que incentivem os estudantes a se envolverem em atividades relacionadas à essa área. Diante disso, os autores ressaltam a importância da adoção de abordagens de ensino inovadoras a fim de facilitar o processo de ensino e aprendizagem de testes de *software*, destacando o uso de jogos educacionais como uma abordagem eficaz para auxiliar nesse processo.

Jogos educacionais são uma subcategoria de jogos sérios, que por sua vez são jogos que além de entreterem, buscam atingir um propósito adicional, como a aprendizagem de conteúdos específicos, como é o caso da subcategoria educacional (Dörner *et al.*, 2016). Wangenheim e Shull (2009) e Stege, Lankvelde e Sprock (2011), enfatizam que a integração de jogos na educação pode ser um estímulo motivacional no processo de aprendizagem. Segundo Fraser (2017), jogos educativos estão sendo

utilizados comumente para ensinar conceitos complexos de várias áreas e o uso desses jogos pode transformar tarefas tediosas ou complicadas em experiências divertidas e motivadoras, de forma a facilitar o processo de ensino. Diante deste cenário, é possível definir duas hipóteses para a condução do presente projeto:

- Hipótese 1: O uso de um jogo educacional para o ensino da técnica funcional de testes de *software* contribui para aumentar a motivação e o engajamento dos estudantes no aprendizado dos conceitos relacionados à essa área;
- Hipótese 2: Um jogo educacional para o ensino da técnica funcional de testes, que aborde também os diferentes níveis de teste e que possua simulações práticas de elaboração e execução de testes, facilita, de maneira abrangente, a compreensão e aplicação de princípios e técnicas de testes de *software*.

1.2 Justificativa e motivação

O teste de *software* é uma atividade essencial e amplamente difundida para assegurar a qualidade dos sistemas de *software* (Garousi *et al.*, 2020) e, geralmente, é a etapa mais dispendiosa do processo de desenvolvimento, podendo representar mais de 70% dos custos totais de projetos de *software* (Abdullahi *et al.*, 2020). Embora a etapa de testes possa acarretar custos consideráveis, pesquisas apontam que o custo da má qualidade, falhas e remoção de defeitos em *softwares* pode ser muito mais significativo. O *Consortium for Information & Software Quality* (CISQ) estima que em 2022, o custo da má qualidade de *software* nos Estados Unidos foi de aproximadamente 2,41 trilhões de dólares e que, desse total, 1,81 trilhões de dólares foram devido a falhas de *software* (Krasner, 2022). Ademais, de acordo com uma pesquisa conduzida em 2013 pela Universidade de *Cambridge* (Britton et al., 2013), o custo total para localizar e corrigir defeitos em *softwares* aumentou para 312 bilhões de dólares anualmente. Considerando esse panorama, o ensino de testes de *software* em cursos de graduação na área da computação desempenha um papel fundamental na capacitação dos futuros profissionais, que deverão ter os conhecimentos necessários para atender às demandas do mercado (Beppe *et al.*, 2018) e, sobretudo, para contribuir com a melhoria contínua da qualidade e confiabilidade dos sistemas de *software*.

Apesar de toda a importância dos testes no processo de desenvolvimento de *software*, essa é uma área que carece de atenção, tanto por parte do sistema educacional, quanto por parte dos estudantes. Em cursos de graduação em computação, o ensino de testes é frequentemente negligenciado em detrimento de atividades de design e implementação e, além disso, está defasado, por conta da falta de ênfase adequada no ensino das habilidades de teste necessárias para a indústria (Jesus *et al.*, 2020). Jia, Yang (2013) afirmam que atualmente, em muitas faculdades, os cursos de teste de *software* são integrados à disciplina de Engenharia de *Software* e que, com regularidade, esses cursos priorizam a teoria e a introdução de métodos de teste, ao invés de promover a prática efetiva de testes de *software*. Nesse contexto, Clarke *et al.* (2014) apontam que, devido ao extenso número de tópicos abordados em uma disciplina de Engenharia de *Software*, há pouca atenção direcionada ao teste de *software*. Por outro lado, há ainda a desmotivação por parte dos alunos no aprendizado de testes de *software* (Jia, Yang, 2013), uma vez que esta é considerada uma atividade entediante (Jesus *et al.*, 2018) e de caráter psicologicamente destrutivo (Neto, 2007; Myers, Sandler, Badgett, 2012; Pressman, Maxim, 2021). Em consequência disso, os alunos veem o teste como algo menos estimulante do que o *design* ou a programação, por exemplo, e assim, apenas 20%, em média, considera uma futura carreira na área de testes (Silvis-Cividjian, 2021).

Ainda dentro do contexto do ensino, os estudantes tendem a considerar tedioso o processo de aprendizagem tradicional baseado em aulas teóricas e leitura de materiais, mas, por outro lado, jogar jogos os mantém focados e envolvidos por mais tempo. Tendo isso em vista, e também o fato de que a geração atual, imersa no mundo digital, passa considerável tempo jogando e empregando suas habilidades de resolução de problemas com alta motivação, recentemente, têm surgido cada vez mais jogos que apoiam a aprendizagem (Šošić, Ristic, Milosevic, 2020). No tocante ao ensino de engenharia de *software*, existe uma tendência promissora para a incorporação de jogos e elementos de jogos nessa área (Kosa *et al.*, 2016; Rodrigues, Souza, Figueiredo, 2018). Segundo Kosa *et al.* (2016), existem diversos subtópicos da área de engenharia de *software* que podem ser abordados em jogos educacionais, incluindo testes de *software*. Os autores também enfatizam o fator da prática que os jogos podem trazer, afirmando que simular uma situação do mundo real pode proporcionar aos alunos a experiência que necessitam, contribuindo para integrar a prática com o conhecimento teórico.

No contexto dos testes de *software*, Šošić, Ristic, Milosevic (2020), afirmam que o uso de jogos para o ensino desta área pode ser visto como um recurso adicional na aprendizagem e que diversos jogos educacionais já foram desenvolvidos com o intuito de facilitar o aprendizado e ressaltar a relevância dos testes. Em relação à técnica funcional de testes, também existem vários jogos educacionais que a incorporam em seu conteúdo, tais como *Bug Hunt* (Elbaum *et al.*, 2007), *UTest* (Silva, 2010), Jogo das 7 Falhas (Diniz, Dazzi, 2011; Diniz, 2011), *BlackBox* (Ribeiro *et al.*, 2017), *Testing Game* (Valle, Rocha, Maldonado, 2017) e *Testing Maze* (Severo, Lelli, 2023). Embora todos esses jogos possuam sua contribuição para o ensino da área e de alguma maneira abordem conteúdos de testes funcionais, não há nenhum jogo que explore a aplicação de estratégias da técnica funcional em diferentes níveis de teste. Além disso, nem todos os jogos desenvolvidos possuem como parte de sua *gameplay,* uma simulação realista e prática do processo de elaboração e execução dos testes. Diante dessas lacunas, tem-se a motivação para o desenvolvimento do presente projeto.

1.3 Objetivo

O objetivo geral deste trabalho é estimular e facilitar o aprendizado de testes funcionais de software entre os estudantes universitários da área de computação, visando tornar o processo de aprendizagem mais eficaz e, por consequência, mitigar a falta de motivação e de conhecimento requerido pela indústria nesse campo específico.

Para isto, tem-se como objetivos específicos o desenvolvimento e avaliação da adequação e usabilidade de um jogo educacional, denominado *CodeGuardians*, abordando as estratégias fundamentais da técnica funcional de testes de *software* aplicadas aos diferentes níveis de teste, em um ambiente de simulações práticas e realistas para a elaboração e execução dos testes.

1.4 Metodologia

O desenvolvimento deste trabalho foi dividido nas seguintes etapas:

- Revisão bibliográfica – estudo e pesquisa sobre temas centrais ao trabalho, como qualidade, testes de *software* e jogos sérios, bem como o levantamento dos critérios necessários para o desenvolvimento de um jogo educacional focado em testes funcionais de *software*, a partir da análise dos pontos fortes e fracos de jogos educacionais já existentes com o mesmo propósito;
- Estabelecimento dos requisitos do protótipo proposto – obtenção e estudo dos requisitos desejados para um jogo educacional de testes funcionais de *software*;
- Estabelecimento das tecnologias utilizadas e arquitetura de *software* – definição das tecnologias e arquitetura que serão utilizadas e irão nortear o desenvolvimento do protótipo, a partir dos requisitos especificados;
- Prototipação – implementação do protótipo;
- Avaliação – avaliação do protótipo por estudantes de cursos da área da computação que já cursaram ou estão cursando a disciplina de Engenharia de *Software*, de forma a verificar e avaliar a usabilidade e adequação da aplicação.

1.5 Organização da monografia

O presente trabalho é composto por quatro capítulos, além deste introdutório, conforme segue:

Capítulo 2 – Fundamentação teórica: neste capítulo são apresentados fundamentos teóricos pertinentes ao trabalho, incluindo qualidade e testes de *software*, jogos sérios e análise de trabalhos correlatos;

Capítulo 3 – Detalhamento do *CodeGuardians*: neste capítulo são expostas a arquitetura e as tecnologias utilizadas para o desenvolvimento do jogo educacional, bem como a descrição de suas mecânicas e regras, juntamente com a ilustração das telas correspondentes;

Capítulo 4 – Resultados do processo de avaliação: neste capítulo são mostrados os resultados obtidos com a avaliação do jogo educacional desenvolvido, realizada por meio de um questionário online disponibilizado juntamente com o link para o *CodeGuardians* a estudantes de cursos da área da computação que já cursaram ou estão cursando a disciplina de Engenharia de *Software*;

Capítulo 5 – Conclusão: neste capítulo são mostradas as conclusões que podem ser vistas a partir dos resultados obtidos, bem como os possíveis trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Com o objetivo de garantir uma compreensão precisa por parte do leitor, este capítulo se destaca como um alicerce fundamental para a estrutura do estudo. Nele, busca-se apresentar e elucidar a base teórica necessária para o desenvolvimento do presente projeto, visando também auxiliar na assimilação dos argumentos e conclusões apresentados ao longo do trabalho. Para tanto, na seção 2.1 é abordada a qualidade de *software*, onde sua importância e características são descritas, a fim de expor o contexto da Engenharia de *Software* onde os testes de *software* estão inseridos. Na seção 2.2, tem-se a base teórica acerca de testes de *software*, incluindo conceitos fundamentais, níveis e técnicas de teste, com ênfase na técnica funcional. Já na seção 2.3, os principais conceitos, classificações e elementos essenciais de jogos sérios são explanados, assim como seu uso na educação. Na seção 2.4 é realizada uma análise comparativa de trabalhos correlatos, expondo seus pontos fortes e deficiências. Por fim, tem-se a seção 2.5 com as considerações finais.

2.1 Qualidade de *Software*

Os desafios relacionados à qualidade de *software* surgiram na década de 1960 com o desenvolvimento dos primeiros sistemas de grande porte. Ao longo do século XX, o software entregue enfrentava problemas de desempenho, confiabilidade, manutenção e reutilização. Em resposta a essa insatisfação, técnicas formais de gerenciamento de qualidade de *software* foram gradualmente adotadas. O aprimoramento dessas técnicas, aliado a avanços em tecnologias de *software* e processos de testes mais eficientes, resultou em melhorias notáveis no patamar geral de qualidade de *software* (Sommerville, 2011). Entretanto, nos dias de hoje, a indústria de *software* continua a enfrentar desafios significativos relacionados à qualidade, devido à persistente prática de lançar produtos sem testes adequados (Pressman, Maxim, 2021) e à falta de compreensão de que implementar um processo de garantia de qualidade é uma parte essencial de uma estratégia para sobreviver em um mercado cada vez mais exigente e competitivo (Bartié, 2002).

Existem diferentes perspectivas acerca da qualidade de *software*. Segundo Bartié (2002) a qualidade de *software* refere-se a um processo sistemático que abrange todas as fases e artefatos produzidos no processo de desenvolvimento de

software, visando garantir a conformidade dos processos e produtos, com o intuito de prevenir e eliminar defeitos. Já Andrade (2015) considera a qualidade de *software* como um conjunto de atributos que devem ser cumpridos para atender às necessidades do usuário, considerando variáveis como o domínio da aplicação, tecnologias utilizadas, características do projeto e demandas do usuário e da organização. Além disso, o autor afirma que a qualidade do *software* também é subjetiva e varia de acordo com a perspectiva de quem a avalia – usuários, desenvolvedores e organizações possuem diferentes necessidades.

A qualidade de um *software* é diretamente influenciada pela qualidade de seu processo de desenvolvimento, logo, é fundamental que o processo de garantia da qualidade abranja tanto o produto tecnológico, quanto o processo de desenvolvimento do *software* (Bartié, 2002; Andrade, 2015). Dessa forma, é possível identificar duas dimensões essenciais para alcançar a qualidade do *software*: a dimensão da qualidade do produto e a dimensão relacionada à qualidade do processo. A qualidade do produto está centrada na garantia da qualidade do produto tecnológico gerado durante o ciclo de desenvolvimento, incorporando fases específicas para testes no cronograma do projeto. Por outro lado, a qualidade do processo envolve a criação de uma cultura que não tolera erros, estabelecendo procedimentos que impedem falhas. Isso abrange a avaliação da qualidade de todos os artefatos gerados ao longo do ciclo de desenvolvimento, indo além dos produtos tecnológicos, de forma a incluir requisitos, modelos, especificações, arquitetura, e demais documentos, garantindo a qualidade em cada etapa do processo (Bartié, 2002).

2.2 Teste de *Software*

O teste de *software* é um processo fundamental de avaliação e execução do *software*, cujo objetivo é encontrar erros (Myers, Sandler, Badgett, 2012; Hooda, Chhillar, 2015). Durante esse processo, os requisitos implementados e os componentes do sistema são exercitados e avaliados de forma a garantir a conformidade com os requisitos especificados e a identificação de discrepâncias entre os resultados esperados e os resultados reais (Hooda, Chhillar, 2015). De maneira simplificada, testar um *software* consiste em verificar, por meio de uma execução controlada, se o seu comportamento está de acordo com o especificado (Neto, 2007). O fator constante que torna os testes necessários é o erro humano (Wazlawick, 2013).

Neste sentido, Delamaro, Maldonado, Jino (2016) afirmam que a construção de *software* é uma tarefa complexa sujeita a diversos problemas, muitos dos quais derivam unicamente do erro humano, visto que o desenvolvimento de *software* é profundamente influenciado pela habilidade, interpretação e execução dos indivíduos envolvidos em sua construção. É importante mencionar que os testes provam apenas a presença de erros, e não sua ausência (Sommerville, 2011; Myers, Sandler, Badgett, 2012). Dessa forma, a área de testes de *software* concentra-se em estabelecer conjuntos de teste finitos e realizáveis, que, embora não garantam a ausência de defeitos no *software*, são capazes de identificar os mais prováveis, permitindo assim sua correção (Wazlawick, 2013).

2.2.1 Técnicas de Teste de *Software*

A criação de casos de teste eficazes é o aspecto mais importante no processo de testes de *software* (Myers, Badgett, Sandler, 2012). Um caso de teste consiste em um par formado por um elemento do domínio de entrada de um programa mais o resultado esperado para a execução do programa para aquela entrada (Delamaro, Maldonado, Jino, 2016). A relevância de criar casos de teste eficazes se deve ao fato de que testar absolutamente tudo em um programa é impossível – o teste de qualquer programa é necessariamente incompleto. Considerando restrições de tempo e custo, a melhor estratégia se torna selecionar subconjuntos de casos de teste, entre todos os casos de teste possíveis, que possuam a maior probabilidade de detectar a maioria dos erros. Nesse sentido, existem técnicas que permitem a criação de casos de teste de maneira mais estratégica (Myers, Badgett, Sandler, 2012).

As técnicas de teste são classificadas com base na origem das informações utilizadas para definir os requisitos de teste. Elas abrangem diversas perspectivas do *software*, exigindo a definição de uma estratégia de teste que leve em consideração suas vantagens e perspectivas complementares. As principais técnicas incluem: técnica funcional e técnica estrutural (Neto, 2007).

A técnica funcional de testes é uma abordagem em que os casos de teste são projetados considerando o programa ou sistema como uma caixa-preta. Os detalhes de implementação não são considerados, e o *software* é avaliado sob a perspectiva do usuário (Delamaro, Maldonado, Jino, 2016). Os testes funcionais são derivados das especificações do sistema (Sommerville, 2011), que descrevem suas funções, ou

seja, as ações realizadas pelo software (Graham *et al.*, 2008). O testador fornece entradas ao sistema e valida as saídas. Se as saídas diferirem das previstas na especificação do programa, o teste é bem-sucedido, pois identificou uma falha no *software* (Fernandes, 2017). Ao aplicar técnicas de teste funcional, é derivado um conjunto de casos de teste que reduzem o número necessário de testes adicionais para alcançar uma cobertura adequada, e que identificam a presença ou ausência de classes de erros, ao invés de um erro associado somente ao teste específico que está sendo realizado (Myers, Badgett, Sandler, 2012).

Já na técnica estrutural de testes, também conhecida como teste caixa-branca, o conhecimento do código-fonte é utilizado para projetar casos de teste visando a detecção de defeitos. O objetivo do uso dessa técnica é criar testes que proporcionem uma cobertura abrangente do programa, garantindo que todos os seus caminhos lógicos sejam exercitados, resultando na execução de cada declaração do programa pelo menos uma vez (Sommerville, 2011). Os testes baseados na técnica estrutural são reconhecidos por sua eficácia em detectar erros, embora também sejam conhecidos por serem desafiadores de se implementar (Bartié, 2002). Além disso, essa técnica é considerada complexa de ser automatizada e sua aplicabilidade não abrange todos os níveis de testes, sendo, na maioria das vezes, limitada ao teste de unidade. Não obstante essas desvantagens, os testes estruturais complementam os testes funcionais ao identificar falhas que não podem ser verificadas apenas com essa técnica (Fernandes, 2017).

Cada técnica possui seus critérios de teste, que são regras estabelecidas para determinar como os elementos do domínio de entrada de um programa – conhecidos como dados de teste – serão agrupados em subdomínios. Esses critérios orientam a seleção dos casos de teste, definindo quais aspectos do programa serão testados (Delamaro, Maldonado, Jino, 2016). Considerando o foco deste trabalho, apenas os principais critérios relacionados à técnica funcional são descritos a seguir.

2.2.2 Critérios da Técnica Funcional

Os principais critérios da técnica funcional de teste incluem: particionamento de equivalência; análise do valor limite; grafo causa-efeito e *error-guessing* (Delamaro, Maldonado, Jino, 2016). No entanto, considerando que a abordagem grafo causa-efeito resulta na geração de tabelas de decisão e que a maioria dos profissionais

considera mais conveniente e útil utilizar exclusivamente essa tabela (Graham *et al.*, 2008), e também que o critério *error-guessing* é uma abordagem informal em que o testador usa sua intuição e experiência para pressupor possíveis tipos de erros (Delamaro, Maldonado, Jino, 2016), os critérios apresentados a seguir para serem abordados no jogo educacional são: particionamento de equivalência, análise do valor limite e tabela de decisão.

2.2.2.1 Particionamento de Equivalência

Um princípio fundamental do teste funcional é a identificação de situações equivalentes. Por exemplo, quando um programa aceita um conjunto específico de dados (condição normal) e rejeita outro conjunto (condição excepcional), pode-se identificar duas classes de equivalência para os dados de entrada do programa: os dados aceitos e os dados rejeitados. Devido à potencial infinitude desses conjuntos, torna-se impraticável testar todos os seus elementos. Portanto, o particionamento de equivalência visa garantir que pelo menos um elemento representativo de cada conjunto seja testado (Wazlawick, 2013). Uma vez estabelecidas as classes de equivalência, é razoável assumir que qualquer elemento dentro de uma classe possa ser considerado representativo dela, pois todos os elementos de uma mesma classe devem se comportar de maneira semelhante. Essa estratégia de dividir o domínio de entrada em classes de equivalência tem como principal consequência e também vantagem, tornar a quantidade de dados de entrada finita e viável para a atividade de teste (Delamaro, Maldonado, Jino, 2016).

De acordo com Myers *et al*. (2004) as classes podem ser estabelecidas seguindo as diretrizes:

1. Se a condição de entrada especifica um intervalo de valores (por exemplo, “a idade deve ser de 10 a 100 anos”), são definidas uma classe válida (idade entre 10 e 100 anos) e duas classes inválidas (idade menor do que 10 e maior do que 100 anos);
2. Se a condição de entrada especifica uma quantidade de valores (por exemplo, “a senha deve ter de 1 a 6 caracteres”), são definidas uma classe válida (senha com 1 a 6 caracteres) e duas classes inválidas (senha com nenhum e senha com mais de 6 caracteres);

3. Se a condição de entrada especifica um conjunto de valores determinados e o programa pode manipulá-los de forma diferente (por exemplo, “o tipo de veículo deve ser carro, moto ou caminhão”), é definida uma classe válida para cada um desses valores e uma classe inválida com outro valor qualquer (ônibus, por exemplo);
4. Se a condição de entrada especifica uma situação do tipo “deve ser de tal forma” (por exemplo, “o primeiro caractere do identificador deve ser uma letra”), são definidas uma classe válida (primeiro caractere é uma letra) e uma inválida (primeiro caractere não é uma letra).

Na literatura, são encontradas pequenas variações dessas diretrizes. Por exemplo, as diretrizes definidas em Pressman e Maxim (2021) e Neto (2007) diferem ligeiramente das descritas em Myers *et al.* (2004). Elas mantêm a diretriz de intervalo de valores, mas também acrescentam que se uma condição de entrada exige um valor específico, são definidas uma classe de equivalência válida e duas inválidas. Além disso, se uma condição de entrada especifica um membro de um conjunto ou é booleana (verdadeira ou falsa), são definidas uma classe de equivalência válida e uma inválida. Ademais, o domínio de saída do programa também pode ser considerado na partição das classes, com o objetivo de identificar possíveis alternativas que poderiam determinar classes de equivalência no domínio de entrada (Delamaro, Maldonado, Jino, 2016).

Após serem identificadas as condições/variáveis de entrada e suas classes de equivalência, observando-se as entradas e as saídas, os casos de teste devem ser determinados (Delamaro, Maldonado, Jino, 2016). O passo a passo deste processo é (Myers *et al.*, 2004):

1. Atribuir um número único a cada classe de equivalência;
2. Até que todas as classes de equivalência válidas tenham sido cobertas por casos de teste, escrever um novo caso de teste que cubra o maior número possível de classes de equivalência válidas não cobertas;
3. Até que os casos de teste tenham coberto todas as classes de equivalência inválidas, escrever um caso de teste que cubra uma, e apenas uma, das classes de equivalência inválidas não cobertas.

A razão pela qual devem ser gerados casos de teste exclusivos para classes inválidas é que um elemento de uma classe inválida pode mascarar a validação do elemento de outra classe inválida (Myers *et al.*, 2004; Delamaro, Maldonado, Jino, 2016).

2.2.2.2 Análise do Valor Limite

Pressman e Maxim (2021) observam que uma parcela significativa de erros ocorre nas fronteiras do domínio de entrada de um programa, em vez do “centro”. Diante disso, uma prática recomendada para a seleção de casos de teste é escolher aqueles que abrangem os limites das partições, juntamente com casos próximos ao ponto médio de cada partição. Isso se deve ao fato de que, durante o desenvolvimento de um sistema, os projetistas e programadores geralmente consideram os valores típicos de entrada. Testar os valores no ponto médio da partição pode ajudar a abranger esses casos típicos. No entanto, os valores-limite são frequentemente atípicos e podem ser negligenciados pelos desenvolvedores (Sommerville, 2011).

Logo, o critério de teste denominado análise do valor limite complementa o particionamento de equivalência, uma vez que seu enfoque é a seleção de casos de teste nos limites das classes de equivalência, ao invés de levar à escolha de qualquer elemento dentro de uma classe. Além de se concentrar nas condições de entrada, a análise do valor limite também considera o domínio de saída como fonte para a seleção de casos de teste (Pressman, Maxim, 2021).

Assim como a técnica de particionamento de equivalência, a análise do valor limite também possui algumas diretrizes gerais (Myers *et al*., 2004):

1. Se uma condição de entrada especificar um intervalo de valores, devem ser definidos casos de teste para os limites do intervalo e casos de teste de entrada inválida, imediatamente subsequentes. Por exemplo, se o domínio válido de um valor de entrada for de -1,0 a +1,0, devem ser definidos casos de teste para as entradas -1,0; 1,0; -1,001 e 1,001;
2. Se uma condição de entrada especifica uma quantidade de valores, devem ser definidos casos de teste para o número mínimo e máximo de valores e um abaixo e além desses valores. Por exemplo, se um arquivo de entrada pode

conter de 1 a 255 registros, devem ser definidos casos de teste para 0, 1, 255 e 256 registros;

3. Usar a diretriz 1 para as condições de saída;
4. Usar a diretriz 2 para as condições de saída;
5. Se a entrada ou saída de um programa é um conjunto ordenado (um arquivo sequencial, por exemplo), deve ser dada maior atenção nos primeiros e últimos elementos do conjunto;
6. Usar a intuição para definir outras condições limites.

2.2.2.3 Tabela de Decisão

As técnicas de particionamento de equivalência e análise do valor limite são comumente aplicadas a situações ou entradas específicas. No entanto, se diferentes combinações de entradas resultarem em ações distintas, pode ser mais desafiador demonstrar isso usando essas técnicas, que geralmente se concentram na interface do usuário (Graham *et al.*, 2008). Nesse contexto, a tabela de decisão é uma maneira adequada para lidar com combinações de entradas (Graham *et al*., 2008), sendo empregada para registrar regras de negócio complexas que o sistema deve obedecer, além de servir como um guia para a criação de casos de teste (Copeland, 2004). Ademais, as tabelas de decisão podem ter o benefício adicional de detectar problemas e ambiguidades nas especificações e complementar o particionamento de equivalência. Ao explorar diferentes combinações de condições, é possível abranger diversas classes de equivalência (Graham *et al*., 2008).

Para encontrar casos de teste utilizando tabelas de decisão, as condições são interpretadas como entradas e as ações como saídas. Em alguns casos, as condições podem referir-se a classes de equivalência de entradas, enquanto as ações representam as principais partes do processamento funcional do item testado. As regras são então tratadas como casos de teste. Como as tabelas de decisão podem ser mecanicamente forçadas a serem completas, é proporcionada uma garantia razoável de um conjunto abrangente de casos de teste como resultado (Jorgensen, 2014).

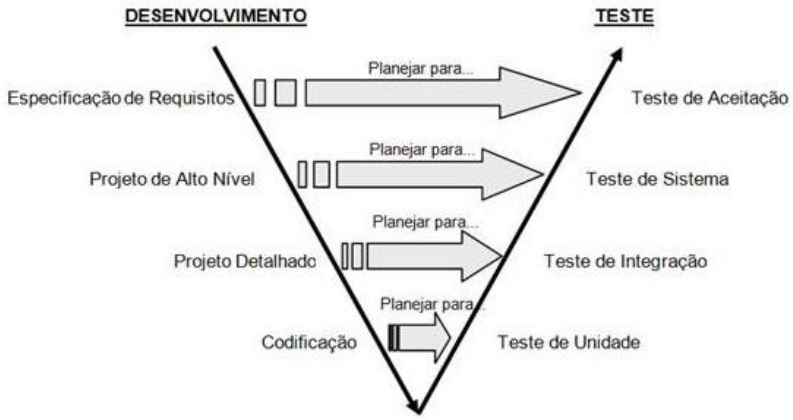
2.2.3 Níveis de Teste de *Software*

A organização dos testes deve ser alinhada ao ciclo de vida do desenvolvimento de *software* para que possa proporcionar efetivamente seus benefícios (Graham *et al.*, 2008). O Modelo Cascata, um dos primeiros modelos de ciclo de vida de desenvolvimento a ser projetado, tem como característica principal a execução sequencial das tarefas. Isso significa que apenas no final do ciclo de vida do projeto, é que os testes tendem a ocorrer (Wazlawick, 2013). Esse padrão pode resultar na descoberta tardia de possíveis defeitos (Graham *et al.*, 2008).

Em resposta a esse e outros problemas encontrados na metodologia tradicional em cascata, surgiu o Modelo V (Graham *et al.*, 2008): uma derivação do Modelo Cascata, mantendo sua natureza sequencial e dirigida por documentação (Wazlawick, 2013). No entanto, sua principal diferença reside no estabelecimento de relações entre as fases do ciclo de vida do desenvolvimento e suas fases de teste correspondentes (Mathur, Malik, 2010), conforme ilustrado na Figura 1.

O modelo também evidencia que o teste não se resume apenas a uma atividade baseada em execução. Existem diversas atividades que precisam ser realizadas antes do término da fase de codificação. Essas atividades devem ocorrer em paralelo com as atividades de desenvolvimento, e os testadores devem colaborar com os desenvolvedores e analistas de negócios para realizar essas tarefas e produzir um conjunto de entregáveis de teste. Os artefatos produzidos pelos desenvolvedores e analistas de negócios durante o desenvolvimento servem como base para os testes em um ou mais níveis. Essa abordagem antecipada, muitas vezes, leva à detecção precoce de defeitos nos documentos de base para os testes (Graham *et al.*, 2008).

Figura 1 – Modelo V



Fonte: https://testesw.wordpress.com/processo-de-testes/ Acesso em: 20 de março de 2024.

Os níveis de testes introduzidos pelo Modelo V são abordados com mais detalhes nas subseções seguintes.

2.2.3.1 Teste de Unidade

O teste de unidade, também conhecido como teste de módulo ou teste de componente, consiste em testar os subprogramas individuais, sub-rotinas, classes ou procedimentos em um programa. Ao invés de iniciar o teste com o programa como um todo, o foco inicial está nos componentes menores do programa. Existem três motivações principais para essa abordagem. Primeiramente, o teste de unidade auxilia na gestão dos elementos de teste, uma vez que a atenção é inicialmente concentrada em unidades menores do programa. Em segundo lugar, testar unidades isoladas facilita a tarefa de depuração, já que, ao encontrar um erro, é possível identificar o módulo específico onde ele ocorre. Por fim, o teste de unidade introduz paralelismo no processo de teste do programa, permitindo testar vários módulos simultaneamente (Myers, Badgett, Sandler, 2012).

O objetivo dos testes de unidade é garantir a qualidade de um componente tecnológico específico, avaliando sua estrutura interna e sua conformidade com os requisitos estabelecidos (Bartié, 2002). Em geral, as falhas encontradas em testes de unidade são provocadas por defeitos de lógica e implementação do módulo (Neto, 2007). Além disso, os testes de unidade, geralmente, são realizados pelos próprios desenvolvedores (Wazlawick, 2013). De acordo com Sommerville (2011), os processos de desenvolvimento de componentes e testes de unidade são frequentemente integrados. Os programadores elaboram dados de teste e testam incrementalmente o código durante o desenvolvimento. Essa abordagem é economicamente viável, pois o programador possui um conhecimento aprofundado do componente, sendo a pessoa mais indicada para gerar casos de teste.

Nos testes de unidade, tanto a técnica funcional, quanto a técnica estrutural, podem ser utilizadas, de forma a se complementarem (Baresi, Pezzè, 2006).

2.2.3.2 Teste de Integração

O teste de integração é um método sistemático para construir a estrutura de um programa enquanto, simultaneamente, testes são conduzidos para identificar erros relacionados às interfaces. O objetivo é combinar os componentes testados individualmente e criar uma estrutura de programa conforme projetado (Sawant, Bari, Chawan, 2012). Os testes de integração são necessários pois, ao integrar dois ou mais módulos, mesmo que esses módulos já estejam aprovados nos testes de unidade, diversos problemas podem ocorrer: um componente pode causar efeitos inesperados ou adversos em outros; perdas de dados podem acontecer por meio de interfaces; a combinação de subfunções pode não resultar na função principal desejada; imprecisões toleráveis individualmente podem se tornar inaceitáveis em níveis mais elevados e até problemas com estruturas de dados globais podem surgir (Pressman, Maxim, 2021).

Recomenda-se que a integração seja feita de maneira incremental, com o programa sendo construído e testado em pequenos incrementos, facilitando o isolamento e correção de erros (Pressman, Maxim, 2021). As estratégias comuns para teste de integração incremental são (Pressman, Maxim, 2021):

- Integração descendente (*top-down*): os módulos são integrados começando pelo módulo de controle principal e descendo pela hierarquia de controle. Os módulos subordinados são incorporados à estrutura, seguindo uma abordagem primeiro-em-profundidade ou primeiro-em-largura. Durante os testes de integração, o módulo de controle principal atua como testador (test driver), enquanto os componentes diretamente subordinados a ele substituem os pseudocontrolados (stubs). Dependendo da abordagem escolhida, os pseudocontrolados são gradualmente substituídos pelos componentes reais, um de cada vez, durante os testes de integração. Os testes são realizados à medida que cada componente é integrado, e ao final de cada conjunto de testes, outro pseudocontrolado é substituído pelo componente real. Um teste de regressão é executado para garantir a ausência de novos erros;
- Integração ascendente (*bottom-up*): a construção e o teste começam com os módulos nos níveis mais baixos na estrutura do programa. Essa abordagem garante que a funcionalidade dos componentes subordinados esteja sempre disponível, reduzindo a necessidade de pseudocontrolados. Componentes de baixo nível são agrupados em módulos que executam funções específicas. Um

pseudocontrolador é criado para gerenciar entrada e saída do caso de teste. O módulo, sendo um conjunto de componentes, é testado. Em seguida, os pseudocontroladores são removidos e os módulos são combinados, seguindo para níveis superiores na estrutura do programa;

- Teste Fumaça: O teste de fumaça é uma abordagem de teste de integração utilizada em equipes ágeis, caracterizada como uma estratégia de integração contínua. Essa abordagem é projetada para projetos com prazos críticos, permitindo avaliações frequentes do projeto. Consiste na integração de componentes de *software* em uma construção (*build*), seguida por testes para identificar erros “bloqueadores” que podem atrasar o cronograma. O produto completo é testado diariamente, utilizando uma abordagem de integração descendente ou ascendente.

2.2.3.3 Teste de Sistema

Os testes de unidade e integração podem assegurar a qualidade dos módulos individuais e suas interações, porém não garantem o comportamento esperado do sistema como um todo. Nesse contexto, os testes de sistema se tornam essenciais, pois têm como propósito verificar se o sistema como um todo está em conformidade com suas especificações (Baresi, Pezzè, 2006). Para Wazlawick (2013), o teste de sistema visa verificar se a versão atual do sistema é capaz de executar processos ou casos de uso completos, do ponto de vista do usuário, realizando uma série de operações no sistema e obtendo os resultados esperados. O teste do sistema ainda pode abranger uma variedade de tipos especializados de testes, a fim de garantir que todos os requisitos funcionais e não funcionais foram atendidos (Yadav, 2012).

O teste do sistema é tipicamente a última etapa de testes em nome da equipe de desenvolvimento. A principal meta nesta fase é identificar o maior número possível de defeitos. Sendo assim, em geral, os testes de sistema são conduzidos por testadores especializados, que compõem uma equipe dedicada e, às vezes, até independente, dentro do processo de desenvolvimento (Graham *et al.,* 2008). Ademais, os testes estruturais têm limitada aplicabilidade nos testes de sistema, devido à extensão do *software*. Avaliar a cobertura de código em milhões de linhas ou identificar caminhos não percorridos é inviável nesse caso. Por outro lado, é crucial identificar o conjunto completo de funcionalidades oferecidas e abranger todos os

casos possíveis para revelar erros no sistema. Tendo isso em vista, os testes de sistema são predominantemente baseados em técnicas funcionais (Baresi, Pezzè, 2006).

2.2.3.4 Teste de Aceitação

O teste de aceitação avalia o sistema com base nos requisitos do usuário. Embora seja semelhante ao teste de sistema, onde todo o sistema é verificado, a principal diferença está na mudança de foco. Enquanto os testes de sistema garantem que o sistema entregue corresponda às especificações, os testes de aceitação garantem que o sistema atenda às necessidades específicas do usuário. Os testes de aceitação devem ser realizados pelo cliente, pois este é quem conhece as necessidades do sistema para agregar valor ao negócio (Yadav, 2012). Após a conclusão do teste de aceitação, o cliente tem a opção de aprovar a versão testada do sistema ou solicitar modificações (Wazlawick, 2013). Existem duas variantes de teste de aceitação (Wazlawick, 2013):

- Teste alfa: realizado pelo cliente ou seu representante de maneira informal e livre, sem o planejamento formal do teste de sistema;
- Teste beta: menos controlado pela equipe de desenvolvimento, onde versões operacionais do *software* são disponibilizadas para vários usuários explorarem sem supervisão direta. Essas versões geralmente expiram após um período definido, quando os usuários são convidados a avaliar o sistema.

2.3 Jogos sérios

Os jogos podem ser definidos como um tipo de atividade lúdica em que os participantes se envolvem em uma realidade fictícia, buscando alcançar pelo menos um objetivo arbitrário e não trivial, seguindo um conjunto de regras estabelecidas (Adams, 2014).

Jogos que utilizam algum tipo de maquinário computacional, como computadores e celulares, são conhecidos como jogos digitais. Os jogos digitais têm alcançado sucesso, atraindo uma ampla gama de usuários, desde crianças até idosos, abrangendo todos os grupos sociais. Assim como os jogos tradicionais, os jogos

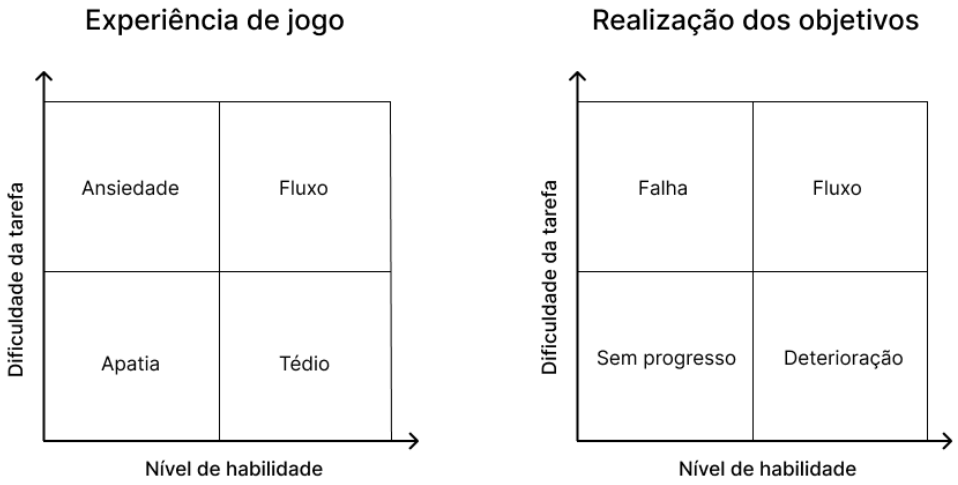
digitais também podem ser intrinsecamente motivadores, uma vez que as pessoas se envolvem por interesse próprio, sem buscar recompensas externas, além de serem capazes de fazer com que seus jogadores se sintam imersos e absorvidos pela atividade. Tendo em vista esses fatores, o uso de jogos digitais que possam ser capazes de contribuir para o alcance de algum propósito ou aprendizado do mundo real tornou-se uma abordagem vantajosa em diversas áreas. Isso ocorre porque tais jogos podem, por meio de uma experiência agradável, fornecer a motivação e o engajamento necessários para que o jogador alcance objetivos predeterminados da vida real. Os jogos digitais que possuem essa capacidade são denominados jogos sérios (Döner *et al.*, 2016).

Os jogos sérios devem ser tanto atrativos quanto eficazes: eles devem atingir seus objetivos sem prejudicar a experiência de jogo. A experiência de jogo (*game experience*) é uma experiência subjetiva de envolvimento completo no jogo, incluindo diversão, desafio, imersão emocional e absorção (Döner *et al.*, 2016). Uma dimensão importante da experiência de jogo é o fluxo do jogo (*game flow*), que se refere à experiência de concentração exclusiva e sensação de controle sobre o jogo, imersão, definição clara de metas e *feedback* imediato e consistente. O fluxo do jogo ocorre quando a dificuldade da tarefa corresponde adequadamente às habilidades do jogador (Prensky, 2001; Döner et *al.*, 2016). Nesse contexto, Sinclair (2011) desenvolveu um modelo de fluxo duplo (*dual flow*), que envolve o equilíbrio entre atratividade e eficácia. Este modelo mostra que é possível manter um equilíbrio adequado entre a dificuldade das tarefas apresentadas e o nível de habilidade dos jogadores. Isso garante que os jogos sérios sejam eficazes em alcançar seus objetivos enquanto também são atrativos para os jogadores. O modelo de fluxo duplo é apresentado na Figura 2.

O desenvolvimento de jogos sérios é composto por dois elementos fundamentais: o *design* de jogos e a produção de jogos. O *design* de jogos engloba todos os aspectos relevantes para a estrutura interna e aparência externa de um jogo, enquanto a produção de jogos refere-se à fase de construção do jogo. O *design* de um jogo sério, em geral, começa com a definição dos seus objetivos caracterizadores e a determinação de seu público-alvo, a fim de manter todo o restante do processo focado nos objetivos e possibilitar que elementos de jogo atraentes para seu público-alvo possam ser investigados. Além disso, o processo de *design* de jogos sérios deve ser abordado de forma holística desde o início: o jogo não deve ser simplesmente um

acréscimo ao conteúdo sério, tampouco o conteúdo sério deve ser apenas adicionado a um jogo de entretenimento sem alterações (Döner *et al.*, 2016).

Figura 2 – Modelo de Fluxo Duplo



Fonte: Adaptado de Sinclair (2011)

No contexto de *design* de jogos sérios, regras, mecânica de jogo e jogabilidade (*gameplay*) são conceitos fundamentais. As regras são normas ou configurações que restringem o jogo. Elas podem incluir diretrizes sobre o que é permitido ou não durante o jogo. Já as mecânicas de um jogo referem-se à forma com que os jogadores podem interagir com um jogo, de acordo com as regras estabelecidas e a situação particular, como um nível ou cenário. Por fim, a jogabilidade pode ser definida como o processo externo que se desenvolve entre o jogador e o jogo enquanto o jogo é jogado (Döner *et al.*, 2016). Para Adams (2014), a jogabilidade depende de desafios e ações. Um desafio é uma tarefa, proposta ao jogador, que não é facilmente realizável. Superar um desafio exige esforço mental ou físico. A maioria dos desafios em um jogo são obstáculos diretos para alcançar um objetivo, e, inevitavelmente, para superar esses desafios e atingir o objetivo de um jogo, é necessário realizar ações, que são especificadas nas regras. Posto isso, a jogabilidade, de acordo com essa perspectiva, consiste nos desafios que o jogador deve enfrentar para atingir o objetivo do jogo e nas ações que ele pode realizar para enfrentar esses desafios.

2.3.1 Classificações

Um jogo, seja sério ou apenas de entretenimento, não requer necessariamente dois ou mais participantes (Dempsey, Rasmussen, Lucassen, 1994). Se houver apenas um participante, o modo de jogo é denominado *single-player*, já se houver mais de um participante é denominado *multi-player* (Döner *et al.*, 2016).

Os jogos também podem ser classificados quanto ao seu gênero. No processo de *design* de jogos digitais, definir a classificação do jogo em um gênero desde o início do processo pode contribuir para a qualidade do produto final, uma vez que cada gênero possui premissas e restrições já apresentadas em jogos de mesma classificação (Herz, 1997). Herz (1997) propôs a classificação dos jogos digitais em oito gêneros, a saber:

- Ação: concentram-se na interação física do jogador com o ambiente virtual, desafiando-os a superar obstáculos através do controle do personagem;
- Aventura: priorizam a narrativa e a resolução de quebra-cabeças, incentivando os jogadores a explorar o mundo do jogo enquanto acumulam itens e progridem na história;
- Luta: enfatizam a competição entre personagens, oferecendo combates intensos e dinâmicos entre jogadores altamente treinados;
- Quebra-Cabeça (Puzzle): requerem a resolução de exercícios lógicos como objetivo principal, desafiando os jogadores a usar estratégias e raciocínio para resolver os enigmas apresentados;
- Interpretação de papéis (RPG): imersão em narrativas épicas onde os jogadores assumem papéis específicos, acumulam experiência e habilidades, completam missões e avançam na história;
- Simulação: reproduzem situações do mundo real, permitindo que os jogadores explorem e interajam com ambientes virtuais complexos de forma imersiva;
- Esportes: recriam atividades esportivas e competições, desde esportes populares até os menos convencionais, em um ambiente virtual;

- Estratégia: exigem tomada de decisões complexas para alcançar a vitória, envolvendo desde gestão de recursos até batalhas militares e manobras diplomáticas.

Dempsey, Rasmussen e Lucassen (1994) destacam que os jogos de simulação, em específico, são alinhados à teoria da cognição situada, a qual postula que o conhecimento é inseparável do seu contexto, com base na noção de que o cérebro humano é naturalmente projetado para aprender por meio da experiência, e que essa experiência só tem significado quando inserida em um contexto específico. Diante dessa perspectiva, na ausência de experiência da vida real, os jogos de simulação podem suprir essa lacuna, mostrando-se como uma poderosa ferramenta (Dempsey, Rasmussen, Lucassen, 1994). Ademais, esse gênero de jogo que apresenta características básicas tanto de jogos, quanto de simulações puras, ainda pode envolver estudos de caso, uma vez que um estudo de caso consiste em uma análise detalhada de uma situação real ou simulada, conduzida com o propósito de exemplificar e/ou destacar características específicas ou gerais (Ellington, Addinall, Percival, 1981).

2.3.2 Elementos fundamentais

Malone (1981), em seu estudo, identificou três fatores que contribuem para tornar tanto os jogos, quanto o aprendizado em ambientes instrucionais computadorizados, mais interessantes e motivadores. Tais fatores incluem: desafio, fantasia e curiosidade.

Primeiramente, para que um ambiente seja desafiador, objetivos cuja realização é incerta devem ser fornecidos (Malone, 1981). De acordo com Adams (2014), as regras de um jogo geralmente definem o objetivo final do jogo como uma condição de vitória. Além dos objetivos finais, que direcionam os jogadores para as metas finais do jogo, os jogos também devem possuir objetivos intermediários, representando marcos ao longo da interação do jogador com o jogo (Crawford, 2003). Ademais, um bom objetivo deve ser pessoalmente significativo e, além disso, algum tipo de *feedback* de desempenho deve ser fornecido, para que os jogadores saibam se estão ou não alcançando suas metas e se sintam motivados (Malone, 1981).

Existem quatro formas pelas quais a realização de um objetivo pode se tornar incerta: aleatoriedade, informações ocultas, nível de dificuldade variável e metas de vários níveis. Grande parte dos jogos de computador que fazem sucesso podem ser jogados em diferentes níveis de dificuldade e possuem diversos níveis diferentes de objetivos. Essas características fazem com que os desafios sejam fornecidos em um nível de dificuldade apropriado e os jogadores cujo resultado é certo em um nível de meta ainda possam ser desafiados por outro nível de meta, o que torna a competição mais motivadora (Malone, 1981).

As fantasias podem enriquecer os ambientes instrucionais e jogos, tornando-os mais atrativos e educacionais. Um ambiente indutor de fantasia é aquele que evoca imagens mentais de coisas não presentes nos sentidos ou na experiência real da pessoa envolvida. Essas imagens mentais podem ser de objetos físicos ou situações sociais, podendo ou não ser prováveis de ocorrer no ambiente do indivíduo. A fantasia é caracterizada por envolver aspectos emocionais, que contribuem significativamente para sua atratividade (Malone, 1981). O elemento de fantasia, muitas vezes, é negligenciado em jogos sérios ou de treinamento, porém, pode ser crucial para o sucesso global de um jogo (El-Shamy, 2001). Os jogos de computador que retratam fantasias envolvendo emoções como guerra, destruição e competição tendem a ser mais populares (Malone, 1981).

A narrativa apresentada conforme um jogador avança em um jogo, também contribui para sua reação emocional, de maneira a garantir que haja um vínculo emocional real vivenciado a cada etapa do jogo. Elementos emocionais como perda, vínculo com personagens, responsabilidade e conquista são componentes que o jogador experimenta enquanto “vive” a narrativa do jogo. Uma narrativa eficaz deve assegurar que os jogadores tenham a capacidade de fazer escolhas que impactem nos resultados dos eventos no jogo, tanto de forma intencional quanto não intencional, ativa ou passivamente. Dessa forma, a presença de uma narrativa forte em um jogo funciona como um meio ativo para afetar as escolhas do jogador, aumentando assim seu engajamento e imersão (Ike *et al.*, 2021).

Acerca da curiosidade, Malone (1981) afirma que os ambientes podem estimular a curiosidade dos indivíduos ao oferecer um nível adequado de complexidade informacional. Eles devem ser inovadores e surpreendentes, mas não completamente incompreensíveis. Existem dois tipos de curiosidade: curiosidade sensorial e curiosidade cognitiva. A curiosidade sensorial é despertada pela atração

de mudanças nos estímulos sensoriais, como luz, som ou outras sensações, em um ambiente. Já a curiosidade cognitiva é despertada ao apresentar informações que desafiam o conhecimento existente do indivíduo, levando-o a buscar mais conhecimento para aprimorar suas compreensões.

Outras características fundamentais para jogos sérios são a personalização e adaptação. Jogos, em geral, são jogados por uma ampla variedade de jogadores, com características e habilidades distintas uns dos outros. Assim, um dos requisitos essenciais para bons jogos é adaptar-se às características do jogador o máximo possível, buscando tanto a atração, quanto a eficácia. Nesse sentido, a personalização refere-se à adaptação estática única de um aspecto do jogo às necessidades ou preferências de um usuário. Isso envolve, por exemplo, permitir que o jogador escolha um avatar. Por outro lado, a adaptação refere-se ao ajuste contínuo do jogo com base nas ações e no desempenho do usuário, além do estado atual do jogo, visando alcançar um estado desejado. Apresentar tarefas mais fáceis ou mais difíceis, e fornecer suporte (dicas) quando o jogador enfrenta dificuldades, por exemplo, são formas de adaptação (Döner *et al.*, 2016).

Além das características anteriormente mencionadas, a literatura apresenta uma variedade de modelos que exploram aspectos fundamentais dos jogos de forma mais detalhada (Döner *et al.*, 2016). Um desses modelos é o de Prensky (2001), que apresenta doze elementos essenciais que devem ser incorporados pelos jogos, juntamente com seus proveitos proporcionados. De acordo com o autor, a combinação desses elementos é a razão pela qual os jogos de computador e *videogame* fazem imenso sucesso e são tão envolventes. Esses elementos e seus respectivos proveitos são apresentados, resumidamente, na Tabela 1.

2.3.3 Jogos educacionais

Os jogos educacionais são uma subcategoria de jogos sérios (Döner *et al.*, 2016), sendo conceituados como jogos criados e aplicados para propósitos de ensino e aprendizagem. Dentro desses jogos, é possível combinar elementos de diversão com conceitos educacionais para impulsionar a motivação e o envolvimento dos alunos (Al-Azawi, Al-Faliti, Al-Blushi, 2016).

Tabela 1 – Elementos de jogos e seus proveitos

Elemento	Proveito proporcionado
Diversão	Prazer e entretenimento
Brincadeira	Envolvimento intenso
Regras	Estrutura
Objetivos	Motivação
Interatividade	Ação
Adaptabilidade	Fluxo
Feedback	Aprendizado
Estados de Vitória	Gratificação do ego
Conflito/Competição/Desafio	Adrenalina
Resolução de Problemas	Criatividade
Interação	Grupos sociais
História e Representação	Emoção

Fonte: Elaborado pela autora

Em um estudo conduzido por Lozza e Rinaldi (2017), foi examinada a percepção e a utilização de metodologias ativas e jogos por parte de jovens universitários de diversos cursos e seus professores na disciplina de Metodologia da Pesquisa Científica. Os resultados indicaram que os professores participantes acreditavam que os jogos poderiam aprimorar os resultados de ensino na disciplina em questão, refletindo a opinião dos alunos, que também expressaram sua preferência por abordagens lúdicas que englobassem a aprendizagem e a aplicação prática do conhecimento.

O emprego de jogos digitais na educação não apenas motiva os alunos, mas também facilita o aprendizado, melhora a retenção do conteúdo ensinado e promove a construção de hábitos de persistência na resolução de problemas e desafios (Tarouco *et al.*, 2004).

Em comparação com a instrução tradicional por meio de aulas expositivas, a utilização de aprendizado baseado em jogos se mostra mais efetiva, gerando efeitos de aprendizado superiores (Al-Azawi, Al-Faliti, Al-Blushi, 2016).

Nos contextos educacionais, os jogos devem ser vistos como um recurso pedagógico complementar, fortalecendo os conceitos já ensinados em sala de aula. Eles devem ser instrutivos e divertidos, orientando ao aluno a direção correta para o aprendizado. A implementação de jogos na sala de aula deve estabelecer uma conexão com o processo de aprendizagem, permitindo que professor e aluno desenvolvam juntos uma experiência agradável de construção de conhecimento (Lozza, Rinaldi, 2017). Além disso, é essencial que os alunos não apenas se envolvam com o mundo do jogo, mas também adotem uma postura crítica em relação ao processo, permitindo que reflitam sobre sua interação com o jogo quando observado de fora. Isso indica que a aprendizagem criativa por meio dos jogos exige um esforço considerável por parte dos professores para garantir resultados positivos (de Freitas, Oliver, 2006). Também é papel dos professores avaliar cuidadosamente os jogos a serem utilizados, considerando os objetivos educacionais. O uso de recursos tecnológicos em sala de aula, incluindo jogos educacionais, sempre requer conhecimento prévio dos mesmos e que esse conhecimento esteja fundamentado em princípios teórico-metodológicos claros (Tarouco *et al.*, 2004).

Embora um jogo educacional bem concebido possa apresentar diversas vantagens, também é importante reconhecer que ele pode ter algumas desvantagens, tais como: se não for adequadamente aplicado, um jogo perde seu propósito educacional; nem todos os conceitos podem ser adequadamente transmitidos por meio de jogos; obrigar os alunos a jogar pode gerar resistência; se as regras não forem compreendidas claramente, os alunos podem se sentir desorientados; e uma avaliação inadequada do jogo pode comprometer o alcance dos objetivos educacionais (Falkembach, 2006). Nesse contexto, Grando (2004) identifica as principais vantagens e desvantagens que os professores devem considerar ao planejarem atividades pedagógicas com jogos. Dentre as vantagens identificadas, pode-se mencionar: (re) significação de conceitos já aprendidos de uma forma motivadora para o aluno; desenvolvimento de estratégias de resolução de problemas; participação ativa do aluno na construção do seu próprio conhecimento; desenvolvimento da criatividade e senso crítico e desenvolvimento de habilidades que os alunos necessitam. Em relação às desvantagens, complementando as que já foram citadas anteriormente, Grando (2004) afirma também que o tempo gasto com atividades de jogo em sala de aula é maior se o professor não estiver preparado e estas atividades podem perder a ludicidade pela interferência constante do professor.

2.4 Análise de trabalhos correlatos

Diversos jogos educacionais têm sido utilizados para apoiar o ensino em variadas áreas da computação (Wangenheim, Wangenheim, 2012). Especificamente no campo dos testes de *software*, essa tendência também é evidente, como destacado por Šošić, Ristic e Milosevic (2020). Vários jogos educacionais foram desenvolvidos com o objetivo de facilitar o aprendizado e enfatizar a importância dos testes (Šošić, Ristic, Milosevic, 2020). Dentro do escopo dos jogos voltados para a área de testes de *software*, que abordam a técnica funcional – foco do presente projeto – pode-se mencionar os seguintes trabalhos:

- *Bug Hunt* (Elbaum *et al.*, 2007): jogo de simulação em formato de tutorial desenvolvido para envolver os estudantes no aprendizado de estratégias de testes de *software*. O jogo é dividido em quatro lições que abordam diferentes conteúdos sobre testes, incluindo conceitos básicos e terminologia, teste funcional, teste estrutural e automação de testes. Na lição sobre teste funcional, em específico, são abordadas as técnicas de Particionamento de Equivalência e Análise do Valor Limite. Cada lição contém um conjunto de objetivos de aprendizagem, um desafio único e uma forma de *feedback* de interação distinta para incentivar a exploração das diferentes estratégias na identificação de falhas. Em cada desafio, são fornecidas todas as instruções necessárias, e o aluno deve elaborar casos de teste com base na abordagem de teste da lição para encontrar falhas em um programa fictício. Há também um botão de “dicas” que pode ser utilizado caso haja dificuldades no desafio. O progresso é avaliado com base no número de falhas detectadas pelo conjunto de testes do aluno e na comparação com outros colegas participantes. Após a conclusão de cada lição, o estudante recebe um *feedback* geral do seu desempenho, e ao professor é fornecido um resumo geral do desempenho de todos os alunos. O principal ponto positivo do jogo é proporcionar ao aluno uma experiência prática de elaboração de casos de teste, demonstrando como as perspectivas de cada uma das diferentes abordagens de testes de *software* se complementam, levando à identificação de diferentes falhas em um

mesmo programa. Por outro lado, o jogo poderia ser aprimorado ao oferecer simulações práticas da execução dos casos de teste em seu programa fictício, bem como abordar os diferentes níveis de teste nas lições, tornando a experiência do aluno mais imersiva e o aprendizado mais abrangente e profundo. Os *feedbacks* de interação e de desempenho, assim como as dicas, são elementos presentes que agregam positivamente para a interatividade e adaptabilidade do jogo. No entanto, o jogo carece de uma narrativa em um contexto fantasioso e lúdico, o que poderia torná-lo ainda mais atrativo e motivador para os estudantes;

- *Utest* (Silva, 2010): jogo de simulação que possui o foco em testes de unidade, abordando questões teóricas e práticas. Diferente do *Bug Hunt* (Elbaum *et al.*, 2007), este jogo apresenta uma narrativa fantasiosa, onde o jogador assume o papel de um programador *freelancer* em um contexto de mercado de trabalho que demanda habilidades em testes de unidade para garantir a qualidade dos produtos. Inicialmente, o jogador explora um jornal com anúncios de projetos disponíveis. Ao escolher um projeto, ele é então apresentado às suas informações relevantes e deve responder a questões teóricas sobre testes de *software* em uma simulação de entrevista de emprego. Se aprovado na entrevista, o jogador recebe a descrição de uma função ou método do projeto escolhido e enfrenta desafios práticos de teste de unidade. Esses desafios envolvem montar partições de equivalência, identificar valores-limite, selecionar dados de entrada para as partições e montar um grafo de causa-efeito. Uma vez resolvido um desafio, o jogo fornece um *feedback* sobre a resposta dada e o jogador ainda deve responder uma pergunta teórica adicional, representando um desafio bônus relacionado ao conteúdo do desafio resolvido. Durante o jogo as ações do jogador são contabilizadas por meio de pontos, que são fornecidos a cada resposta correta. Ao final de um projeto, o jogador recebe um *feedback* sobre seu desempenho, e sua pontuação é salva em um *ranking* junto às pontuações de todos os jogadores, o que promove a competição. O jogo ainda possui elementos de personalização e adaptação, possibilitando que o jogador escolha um avatar no início e fornecendo

dicas caso haja dificuldade. Apesar dos critérios da técnica funcional serem abordados de forma abrangente, o jogo não abrange a elaboração e execução de casos de teste e, além disso, como seu foco está apenas em testes de unidade, não são abordados todos os níveis de teste;

- Jogo das 7 Falhas (Diniz, Dazzi, 2011; Diniz, 2011): jogo educacional de simulação que se concentra na execução de casos de teste funcionais. Na narrativa do jogo, o jogador assume o papel de testador em uma equipe de teste de *software* de uma empresa fictícia, com o objetivo de identificar falhas nas funcionalidades de um *software* específico no menor tempo possível. O jogo oferece dois níveis de complexidade – baixo e médio. Cada nível apresenta uma funcionalidade com sete falhas a serem descobertas. O jogador é inicialmente apresentado aos requisitos da funcionalidade a ser testada e, em seguida, os campos e botões relacionados à funcionalidade são exibidos na tela para que possam ser inseridas entradas de casos de teste. Quando uma entrada identifica uma falha, o jogador recebe uma mensagem indicando a descoberta da falha e deve selecionar a classe de equivalência ou valor-limite que a originou, a partir de sete opções listadas. O jogador ganha pontos somente ao selecionar a opção correta, recebendo *feedback* positivo ou negativo após cada seleção. Assim como os jogos analisados anteriormente, o Jogo das 7 Falhas inclui um recurso de dicas, pelo qual o jogador pode receber uma única dica sobre a localização de uma das falhas. Embora o jogo possua o ponto positivo de envolver uma simulação prática e realista de execução de casos de teste baseados em critérios funcionais, não é abrangida a elaboração de casos de teste com base nesses critérios dentro do jogo; é sugerido que os alunos os elaborem previamente para executá-los durante o jogo. Adicionalmente, outra carência apresentada é o fato de nenhum nível de teste ser abordado;
- *BlackBox* (Ribeiro *et al.*, 2017): jogo *single-player* que desafia os jogadores a resolverem missões usando seus conhecimentos de teste funcional. Situado em um cenário de uma agência de espionagem governamental que se revela uma organização criminosa controlada por

invasores de sistemas, os jogadores devem utilizar suas habilidades para impedir os vilões de alcançarem seus objetivos malignos. No início do jogo, o jogador pode escolher um avatar para representá-lo. O jogo possui três missões (cenários) que são apresentadas de forma progressiva. Em cada missão, o jogador recebe informações sobre seu objetivo, que no contexto de testes, representam as especificações de uma determinada funcionalidade. Também são apresentadas informações relacionadas ao contexto de testes funcionais para aquela missão, como classes de equivalência e valores-limite. Diante disso, o jogador deve resolver um problema com base nas informações fornecidas, como selecionar as saídas esperadas para dadas entradas, de forma a completar os casos de teste, na primeira missão. Após completar uma missão, o jogador deve responder um questionamento que relaciona as informações sobre o teste com o objetivo da missão. Ao escolher a alternativa correta, o jogador terá sucesso na missão, e *feedbacks* positivos ou negativos são fornecidos com base no seu desempenho. O principal ponto forte do jogo é sua narrativa forte e envolvente. Dentre as carências identificadas, pode-se mencionar a falta de uma experiência/simulação prática de execução dos casos de teste e a não abordagem dos níveis de teste;

- *Testing Game* (Valle, Rocha, Maldonado, 2017): jogo de ação em 2D no qual os jogadores são representados por avatares com várias habilidades, como andar, correr, pular e atirar. Dividido em três níveis, cada um corresponde a uma técnica de teste de *software* – funcional, estrutural e baseada em defeitos – com fases relacionadas aos seus respectivos critérios de teste. Os jogadores avançam pelos níveis abrindo portas com chaves encontradas durante o jogo. Além disso, o jogo inclui dicas e módulos teóricos acessíveis por meio de um ícone no menu superior, o que auxilia na aprendizagem dos estudantes. A pontuação é baseada nos acertos dos jogadores. Apesar de haver um contexto de fantasia, em que os jogadores realizam ações e derrotam inimigos, não há a presença efetiva de uma narrativa. Acerca do nível em que conteúdos da técnica funcional são abordados, o jogador deve resolver desafios como: eliminar inimigos que representam classes de

equivalência inválidas de acordo com a especificação dada; preencher uma tabela de classes de equivalência; encontrar casos de teste válidos para satisfazer o critério de particionamento de equivalência, eliminando os inimigos que representam casos de teste inválidos e encontrar casos de teste que satisfaçam o critério análise do valor limite, também eliminando inimigos que representam casos de teste inválidos. Embora esses desafios sejam divertidos e motivadores, não é proporcionada nenhuma experiência direta de elaboração de casos de teste e nem prática da execução dos mesmos. Também não são abordados os diferentes níveis de teste, aspectos esses que poderiam tornar a aprendizagem por meio do jogo mais profunda e abrangente;

- *Testing Maze* (Severo, Lelli, 2023): jogo de quebra-cabeça em labirinto que desafia os jogadores a alcançar um objetivo específico enquanto aprendem sobre especificação de casos de teste funcionais. Ambientado em uma caverna com temática de tumbas egípcias, o jogador assume o papel de um gato arqueólogo em busca de tesouros seguindo notas de casos de teste deixadas por um engenheiro de *software*. O jogo consiste em um único nível com 10 casos de teste e 4 tesouros, onde o objetivo é completar todos os casos de teste com o menor número de tentativas possível e coletar os tesouros. Cada caso de teste é detalhadamente descrito, contendo requisitos associados, pré-condições, dados de entrada, instruções passo a passo e resultados esperados. Esses casos de teste são basicamente instruções para que o jogador as execute no labirinto, como andar até tal ponto e abrir um baú com uma senha para coletar um tesouro. Porém, as pré-condições de cada caso de teste fazem com que o jogador tenha que estruturar uma boa ordem de execução, já que alguns casos de teste possuem como pré-condição não ter completado determinado caso de teste anteriormente ou já possuir determinado tesouro. O labirinto ainda apresenta armadilhas que são ativadas em bifurcações, impedindo o retrocesso do jogador, que deve terminar um caso de teste ou usar o botão “Retornar”, perdendo uma de suas sete vidas a cada uso. O jogo termina quando o jogador completa todos os casos de teste, coleta todos os tesouros ou perde todas as vidas, recebendo um *feedback*

> juntamente com sua pontuação total ao final. Embora não englobe a elaboração e simulação prática de execução de casos de teste, nem conceitos de níveis de teste e critérios de técnica funcional, o jogo destaca-se por promover o aprendizado sobre especificações de testes funcionais, ao mesmo tempo que faz com que o raciocínio lógico do jogador seja exercitado.

Com base nas análises realizadas, torna-se evidente a falta de jogos educacionais que abordem não apenas a técnica funcional, mas também os diferentes níveis de teste. Essa deficiência persiste desde os trabalhos pioneiros até o atual estado da arte. Além disso, há uma escassez de jogos que integrem em sua jogabilidade a simulação da execução de casos de teste, visando proporcionar uma experiência prática e realista, e por conseguinte, um aprendizado mais profundo ao aluno. Dos jogos analisados, apenas o Jogo das 7 Falhas (Diniz, Dazzi, 2011; Diniz, 2011) incorpora essa abordagem, porém, o jogo não engloba, juntamente à essa simulação, a elaboração dos casos de teste, o que tornaria a experiência de jogo ainda mais imersiva e próxima à realidade. Os critérios funcionais mais abordados nos jogos analisados são o Particionamento de Equivalência e a Análise do Valor Limite. Nesse sentido, o *Utest* (Silva, 2010) destaca-se como o trabalho mais abrangente, incluindo também o critério de Grafo Causa-Efeito/Tabela de Decisão. Por fim, em relação aos elementos de jogo, a maioria dos trabalhos analisados incorpora aspectos importantes como narrativa, *feedback* de desempenho e elementos de adaptação e personalização, como dicas e escolha de avatares.

Na Tabela 2, é apresentada uma síntese da análise realizada considerando os trabalhos correlatos. Os aspectos avaliados foram divididos em dois subgrupos, um contendo os aspectos relacionados ao conteúdo educacional e outro contendo os aspectos relacionados a elementos de jogo. São consideradas três classificações possíveis para cada aspecto. A primeira delas indica que o jogo abrange/“atende” por completo determinado aspecto; se o jogo não abrange/”atende” por completo o aspecto definido, ele o atende “em partes”; já se o jogo não abrange nada em relação a determinado aspecto, é atribuída a classificação “não atende”.

Tabela 2 - Análise comparativa dos jogos educacionais correlatos

Jogo Educacional	Conteúdo					Elementos de jogo		
	Foco na técnica funcional	Critérios funcionais	Diferentes níveis de teste	Elaboração de casos de teste	Simulação de execução de casos de teste	Feedbacks	Narrativa	Adaptação e personalização
Bug Hunt								
Utest								
Jogos das 7 falhas								
Blackbox								
Testing Game								
Testing Maze								

	Atende		Atende em partes		Não atende

Fonte: Elaborado pela autora

Diante do cenário atual exposto na Tabela 2, evidencia-se que o desenvolvimento de um jogo educacional, capaz de abordar todos os principais critérios da técnica funcional e preencher as lacunas identificadas em relação à abordagem dos diferentes níveis de teste, elaboração e simulação de execução de casos de teste, representa uma solução abrangente e relevante para o aprendizado de testes funcionais, configurando uma significativa contribuição para o ensino.

2.5 Considerações finais

Neste capítulo foram expostos os conceitos teóricos fundamentais para alicerçar o trabalho. Os principais conceitos e aspectos acerca de qualidade e testes de *software* foram explorados, assim como os conceitos relacionados a jogos sérios e educacionais, destacando o potencial que há em seu uso e suas características essenciais. Além disso, a partir da análise de trabalhos relacionados, foi possível identificar as principais carências em jogos educacionais que abordam a técnica funcional de testes de *software*. Nesse contexto, as maiores deficiências identificadas foram a falta de jogos que abordem os diferentes níveis de testes e a escassez de jogos que proporcionem uma experiência prática e imersiva de elaboração e execução de casos de teste em um contexto de simulação, sendo essas as principais carências que se busca sanar com o presente projeto.

No próximo capítulo é apresentado o detalhamento sobre o desenvolvimento do jogo educacional e a descrição de suas mecânicas e regras, acompanhadas das ilustrações das telas correspondentes.

3 DETALHAMENTO DO CODEGUARDIANS

Neste capítulo são expostos detalhes sobre o desenvolvimento do jogo educacional *CodeGuardians*, realizado com base no levantamento bibliográfico e na análise de jogos educacionais correlatos. Para tanto, na seção 3.1, tem-se as tecnologias e arquitetura utilizadas para o desenvolvimento do jogo. Já na seção 3.2, tem-se a descrição geral do jogo, juntamente com a descrição detalhada de suas mecânicas e regras, acompanhada de ilustrações das telas. Por fim, na seção 3.3, tem-se as considerações finais sobre o capítulo.

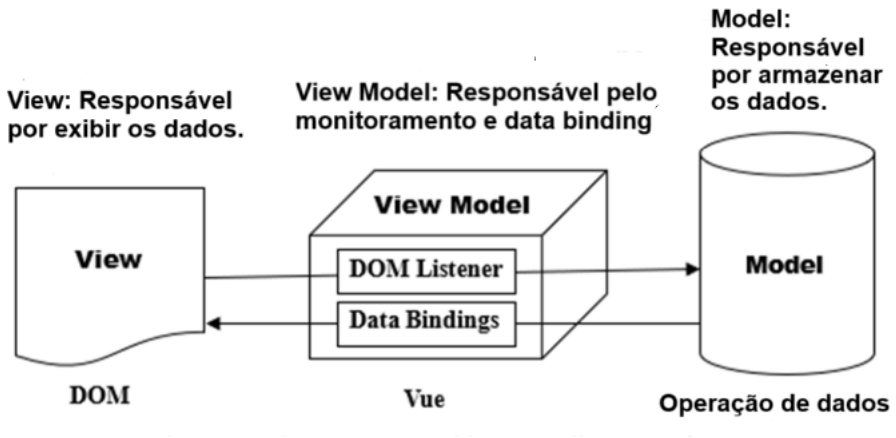
3.1 Tecnologias e arquitetura

O jogo educacional *CodeGuardians* é uma *Single Page Application* – SPA (Adobe Experience Cloud, 2023), focada exclusivamente na camada de *front-end*, sem a necessidade de *back-end* ou banco de dados. O desenvolvimento da aplicação foi realizado utilizando HTML, CSS e *JavaScript*, com base em uma abordagem de componentes por meio do *Vue.js* 3[1], um framework *JavaScript* voltado para a construção de interfaces de usuário. A utilização de componentes permite a divisão dos elementos da interface em partes independentes e reutilizáveis (Vue.Js, 2023).

A arquitetura utilizada segue o padrão MVVM (*Model-View-ViewModel*) do *Vue.js*[2], apresentado na Figura 4. Neste padrão, a "*View*" representa a interface do usuário (HTML e CSS), que é responsável pela exibição dos dados. O "*Model*" representa os dados da aplicação e a lógica de negócio (*JavaScript*). Já a "ViewModel" atua como uma ligação bidirecional (*two-way binding*) entre a *View* e o *Model*, utilizando a reatividade do Vue.js para sincronizar automaticamente as mudanças de dados entre a interface do usuário e o estado da aplicação (Li, Zhang, 2021). Essa abordagem permite que a interface do usuário responda de maneira dinâmica às alterações de estado e garante que a lógica de apresentação e a lógica de negócios permaneçam desacopladas, o que resulta em um código mais organizado e de fácil manutenção.

[1] <https://vuejs.org/> Último acesso em 15 de junho de 2024
[2] <https://012.vuejs.org/guide/> Último acesso em 15 de junho de 2024

Figura 3 - Arquitetura MVVM do Vue.js



Fonte: Adaptado de Li e Zhang (2021).

Cada seção da interface, que compõe as fases e as demais telas do jogo, foi implementada como um componente *Vue*, desenvolvido de forma modular, encapsulando sua própria estrutura, estilo e comportamento. Isso permite a reutilização de componentes em diferentes partes da aplicação, por meio da passagem de propriedades de um componente “pai” para um componente “filho”. Além disso, estados com o uso compartilhado por diversos componentes, como pontuação do jogador e dicas restantes, são gerenciados de forma global e centralizada por meio da *Reactivity API*[3], que permite criar um objeto reativo com estados compartilhados e importá-lo para vários componentes.

O *Vue.js* também inclui o *Vue Router*[4], utilizado para gerenciar a navegação na aplicação. Com o *Vue Router*, foi possível definir rotas para diferentes partes da aplicação e carregar os componentes correspondentes conforme necessário, sem a necessidade de recarregar páginas no servidor, mantendo assim a aplicação como uma *Single Page Application* (SPA).

Após o desenvolvimento, a aplicação foi implantada por meio da plataforma *Netlify*[5], que permite, em seu plano gratuito, hospedagem com alto desempenho e implantação contínua, por meio de repositórios do *GitHub*, de sites estáticos e SPAs. Dessa forma, o jogo é disponibilizado aos jogadores em um ambiente estável.

[3]<https://vuejs.org/guide/scaling-up/state-management.html#simple-state-management-from-scratch> Último acesso em 15 de junho de 2024
[4] <https://router.vuejs.org/> Último acesso em 15 de junho de 2024
[5] <https://www.netlify.com/> Último acesso em 15 de junho de 2024

3.2 Descrição geral e mecânicas do jogo

O CodeGuardians é um jogo *single-player* do gênero de simulação, envolvendo testes funcionais de *software*. O público-alvo do jogo são estudantes universitários de cursos da área de computação, sendo esperado que estes já tenham conhecimento prévio sobre conceitos básicos de testes de *software*. Logo, este jogo deve ser usado pelo docente para complementar o ensino, reforçando de maneira lúdica os conceitos teóricos abordados em sala de aula.

O jogo se situa em um contexto fantasioso de mundo pós-apocalíptico, centrando sua narrativa em torno de um sistema computacional essencial para a manutenção da civilização – o Aura379. Este sistema está atualmente em risco devido às ações de uma gangue conhecida como *ShadowHackers*, que busca desestabilizá-lo com o objetivo de dominar o mundo. Após surgir a informação de que um membro da gangue conseguiu infiltrar-se na sede de desenvolvimento do Aura379 e introduzir bugs no sistema, o professor Sr. Clarke fundou uma academia secreta denominada *CodeGuardians*. Esta academia é responsável por preservar a integridade do sistema e proteger a sociedade dos *ShadowHackers*. Neste contexto, o jogador assume o papel de um estudante prodígio recrutado pelo Sr. Clarke para integrar os *CodeGuardians*. A missão do jogador é testar os módulos afetados do sistema, identificar os *bugs* inseridos e reportá-los para que possam ser corrigidos pelos desenvolvedores, evitando assim o caos iminente que poderia afetar o mundo.

Em cada nível do jogo, que por sua vez, incorpora um nível de teste, o jogador deverá: resolver um desafio envolvendo a aplicação dos critérios da técnica funcional, elaborar casos de teste com base no artefato do desafio anterior e, por fim, executar os casos de teste no sistema fictício, em busca de falhas. Em cada uma dessas etapas, o jogador tem a chance de pontuar.

O objetivo final do jogo consiste em atingir uma pontuação mínima, para que o sistema Aura379 seja aprovado no teste de aceitação, pelo Sr. Clarke. Ao final do jogo, espera-se que o jogador: tenha compreendido os principais critérios da técnica funcional de testes, adquirindo também a capacidade de identificar as situações ideais para o uso de cada critério; tenha compreendido e saiba diferenciar os variados níveis de testes; saiba elaborar e executar casos de teste e se sinta motivado para aprofundar seu aprendizado em testes de *software*.

Nas subseções seguintes, são apresentadas as mecânicas e regras do jogo de forma detalhada.

3.2.1 Apresentação inicial

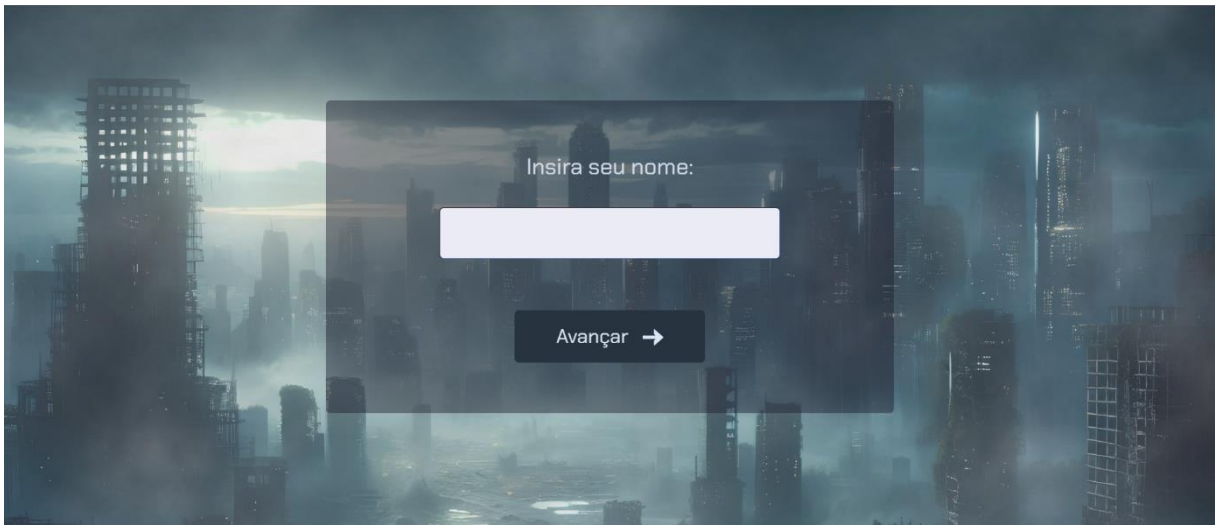
A tela inicial do jogo apresenta seu título e um botão para iniciar, conforme apresentado na Figura 4. A imagem de fundo, retratando uma cidade abandonada, combinada com o efeito de neblina que se move pela tela, cria uma atmosfera que imediatamente transporta o jogador para o cenário pós-apocalíptico do jogo, sendo essa uma característica que se mantém nas telas seguintes.

Figura 4 - Tela inicial

Fonte: Elaborado pela autora.

Na próxima tela, apresentada na Figura 5, é solicitado o nome do jogador, que não pode ser vazio e nem conter mais de 30 caracteres. O nome é solicitado para que haja, em alguns momentos do jogo, a personalização da interação, chamando o jogador pelo seu nome.

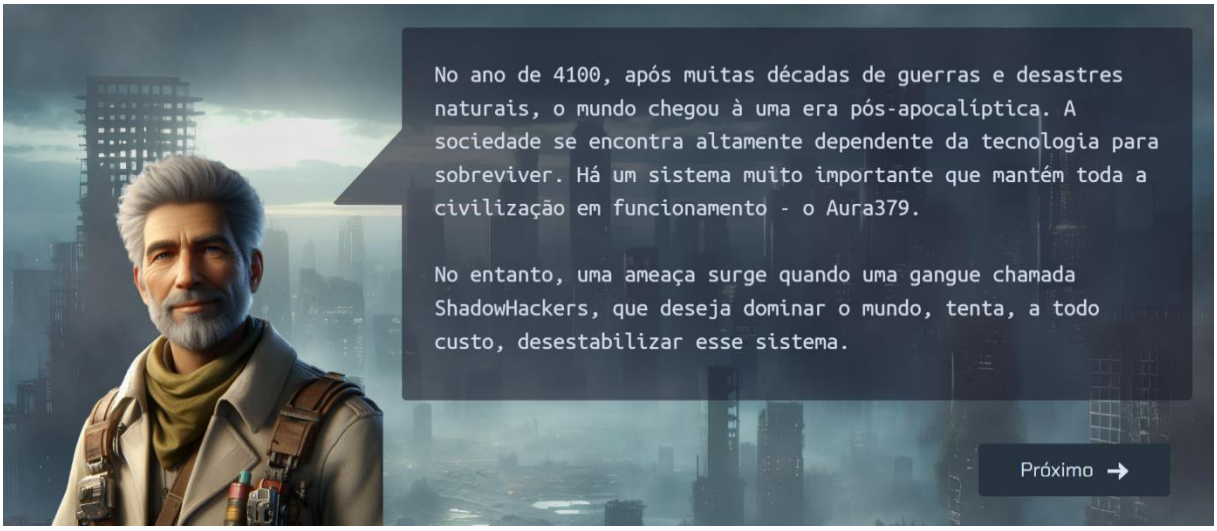
Figura 5 - Tela de inserção do nome do jogador



Fonte: Elaborado pela autora.

Após o jogador informar seu nome, a narrativa do jogo é apresentada. Na Figura 6, é exposta a primeira tela, a partir da qual a narrativa é apresentada ao jogador.

Figura 6 - Apresentação da narrativa do jogo



Fonte: Elaborado pela autora.

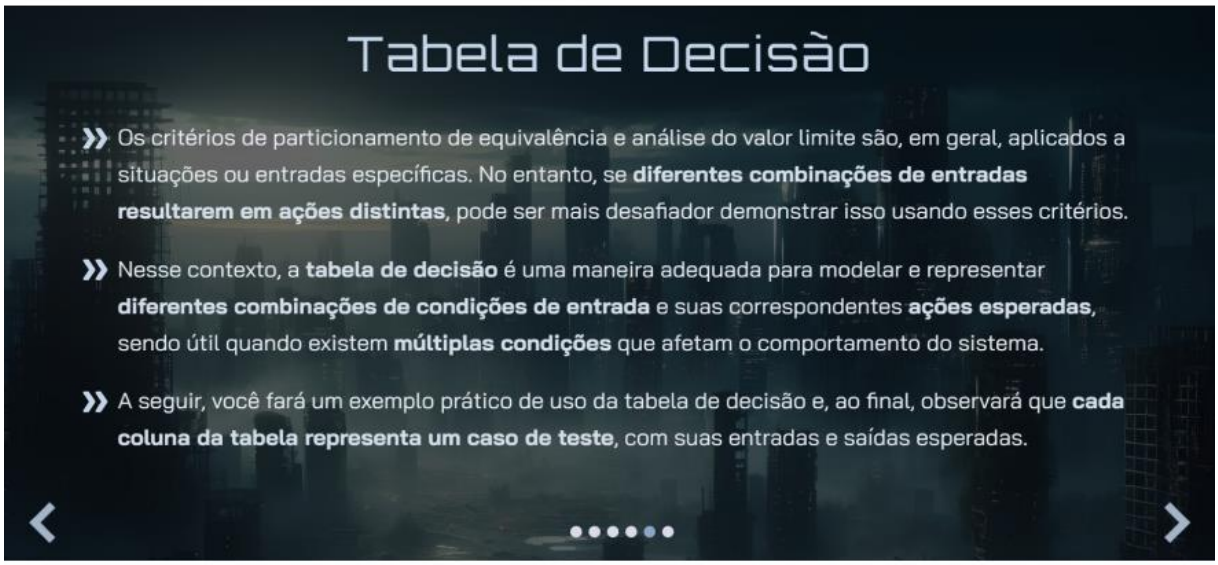
Após a apresentação da narrativa, o jogador pode optar por realizar uma fase de treinamento, que oferece explicações breves e objetivas sobre os principais conceitos abordados no jogo, como técnica funcional de testes, critérios dessa técnica, casos de teste e níveis de teste. Além disso, cada critério da técnica funcional

apresentado é acompanhado por um exemplo prático, semelhante aos desafios que serão encontrados no jogo, sendo possível conferir se as respostas estão corretas ou erradas. Essa fase oferece ao jogador a oportunidade de revisar os conceitos que serão abordados durante o jogo, assim ajudando-o a se preparar melhor e a ganhar confiança para enfrentar os desafios futuros dos níveis.

Na Figura 7a, é apresentada a tela da fase de treinamento que contém um resumo sobre o critério de Tabela de Decisão e na Figura 7b é apresentada a tela que contém o exemplo prático desse critério, com as respostas já conferidas.

Figura 7 - Fase de treinamento – Tabela de Decisão

a)

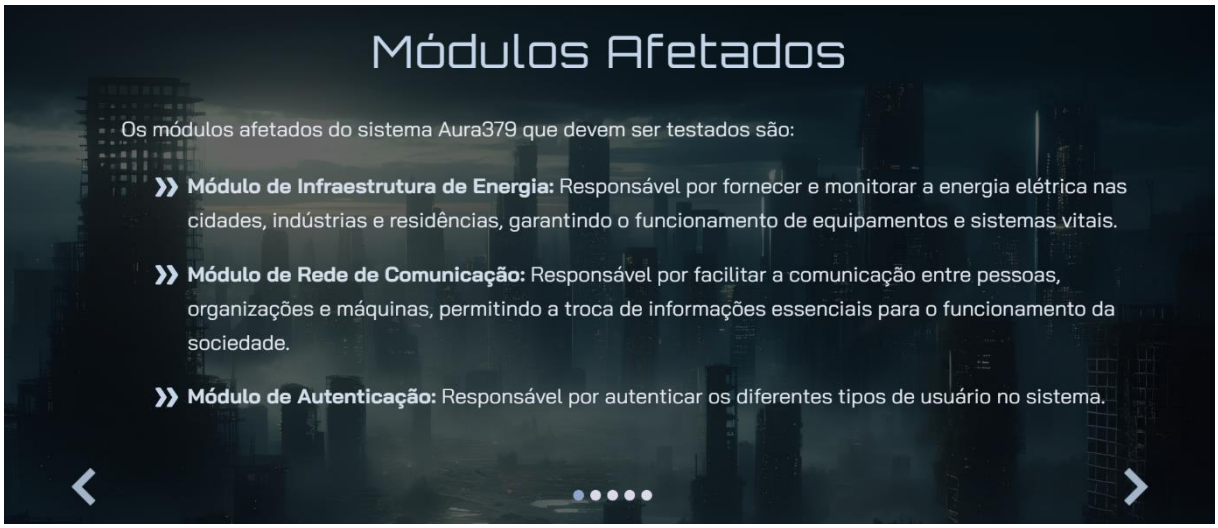


b)

Fonte: Elaborado pela autora.

Por fim, as regras e o funcionamento do jogo são apresentados ao jogador. Na tela ilustrada na Figura 8 (uma das telas da seção de funcionamento e regras), são apresentados os módulos afetados do sistema fictício que deverão ser testados durante o jogo.

Figura 8 - Regras do jogo – Módulos afetados



Fonte: Elaborado pela autora.

3.2.2 Controle de níveis

O jogo é composto pelos oito níveis apresentados no Apêndice A. Estes níveis, chamados de missões durante o jogo, incorporam diferentes níveis de teste. Os primeiros sete níveis correspondem a testes de unidade ou integração, seguindo a estratégia de integração ascendente (*bottom-up*), enquanto o último nível consiste em um teste de sistema. Em cada um desses níveis, o jogador enfrenta três desafios, correspondendo a três fases distintas:

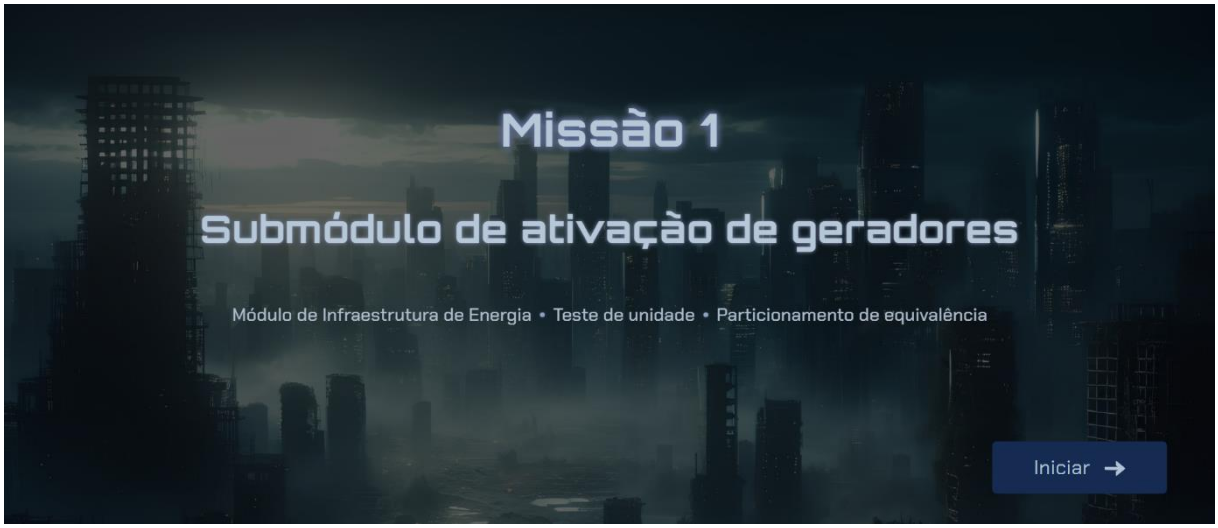
1. Aplicação de critérios da técnica funcional, onde deve identificar as classes de equivalência corretas, completar retas numéricas com os valores-limite apropriados ou preencher tabelas de decisão;
2. Elaboração de casos de teste, fundamentado no artefato resultante do uso dos critérios da técnica funcional;

3. Execução dos casos de teste, com o objetivo de identificar falhas no sistema fictício.

Em cada desafio, o jogador tem a oportunidade de pontuar, dependendo de suas resoluções e desempenho. Assim, o jogo apresenta níveis diferentes de objetivos para cada fase. Além disso, cada nível do jogo possui um grau de dificuldade, que aumenta à medida que o jogador avança no jogo. Essa progressão garante a correspondência entre a dificuldade das tarefas e a habilidade do jogador, promovendo o duplo fluxo, que por sua vez, possibilita uma aprendizagem eficaz e uma experiência de jogo agradável.

Na tela inicial de cada nível é apresentado: seu número; o nome do submódulo a ser testado; o nome do módulo ao qual pertence o submódulo a ser testado; o nível de teste; o critério da técnica funcional que será utilizado e um botão para iniciar, conforme ilustrado na Figura 9.

Figura 9 - Tela de apresentação dos níveis



Fonte: Elaborado pela autora.

Além disso, durante todo o jogo, após os níveis serem iniciados, são apresentadas informações de nível e fase atuais, progresso, pontuação total e dicas restantes na parte superior da tela (Figura 10). A barra de progresso é dividida em oito partes, representando os oito níveis, e cada parte é preenchida após cada nível ser finalizado.

Figura 10 - Informações de progresso, pontuação e dicas

Fonte: Elaborado pela autora.

Cabe mencionar que, quando o jogador finaliza uma fase de um nível ou um nível por completo, ele não pode refazê-lo(a), já que não há um tempo limite para finalizar os níveis e suas fases.

3.2.3 Atribuição de pontuação

O sistema de pontuação do jogo varia de acordo com o nível do jogo em que o jogador se encontra, a saber:

Níveis de 1 a 4:

- +30 pontos se a resolução completa da fase de uso dos critérios da técnica funcional estiver correta;
- +5 pontos por caso de teste correto na fase de elaboração de casos de teste;
- -5 pontos por caso de teste incorreto ou excedente na fase de elaboração de casos de teste;
- +15 pontos por falha encontrada na fase de execução dos casos de teste.

Níveis de 5 a 7:

- +40 pontos se a resolução completa da fase de uso dos critérios da técnica funcional estiver correta;
- +10 pontos por caso de teste correto na fase de elaboração de casos de teste;
- -10 pontos por caso de teste incorreto ou excedente na fase de elaboração de casos de teste;
- +20 pontos por falha encontrada na fase de execução dos casos de teste.

Nível 8:

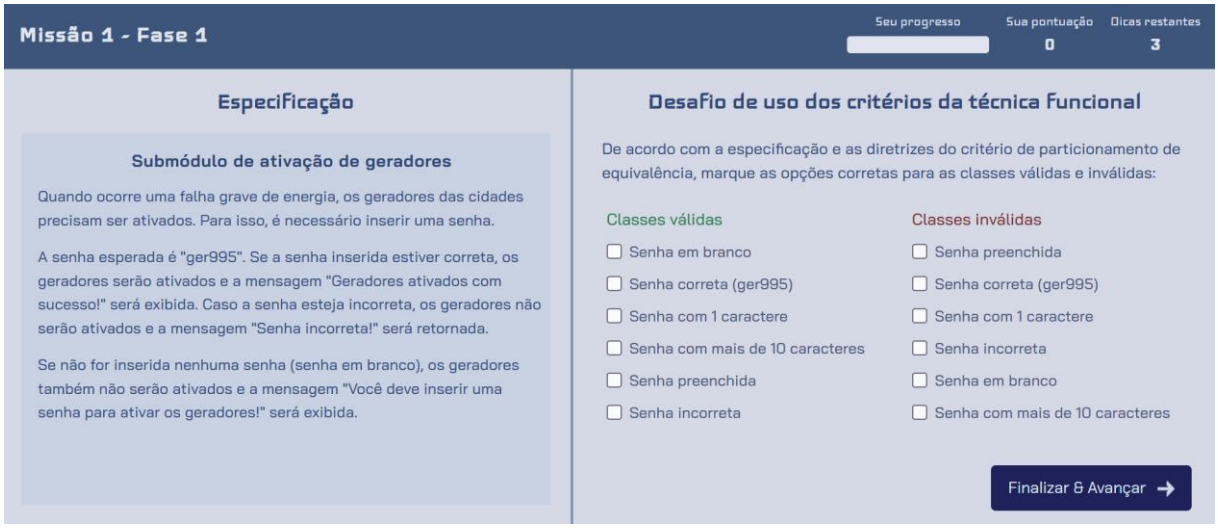
- +50 pontos se a resolução completa da fase de uso dos critérios da técnica funcional estiver correta;
- +15 pontos por caso de teste correto na fase de elaboração de casos de teste;
- -15 pontos por caso de teste incorreto ou excedente na fase de elaboração de casos de teste;
- +30 pontos por falha encontrada na fase de execução dos casos de teste.

Vale destacar que o jogador nunca terá uma pontuação total negativa, ou seja, mesmo que a soma da pontuação dos níveis realizados resulte em um valor negativo, sua pontuação total será de 0 pontos.

3.2.4 Fase de uso dos critérios da técnica funcional

A fase de uso dos critérios da técnica funcional contém a especificação do submódulo a ser testado e o desafio de uso dos critérios da técnica funcional, conforme apresentado na Figura 11.

Figura 11 - Fase de uso dos critérios da técnica funcional



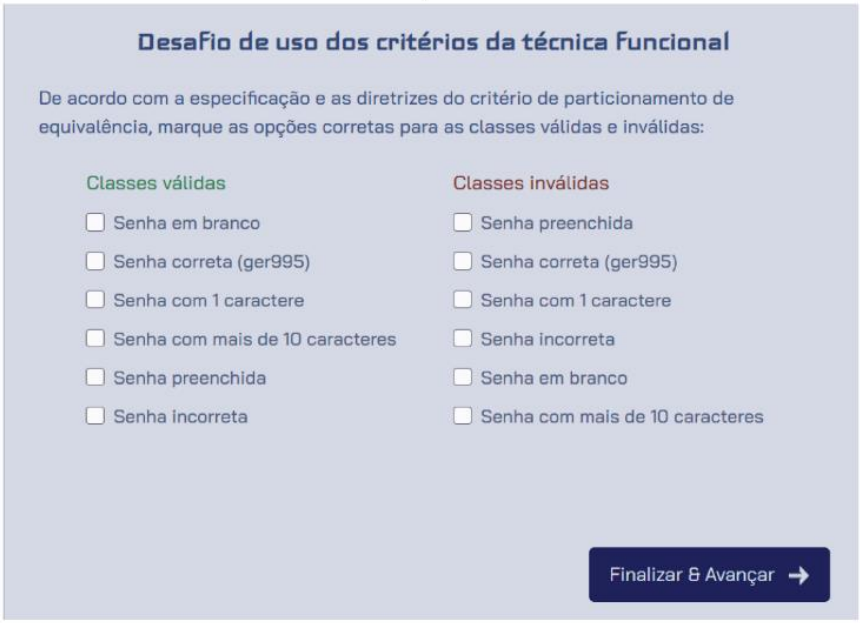
Fonte: Elaborado pela autora.

O desafio de uso dos critérios da técnica funcional varia de acordo com o critério abordado no nível, especificamente:

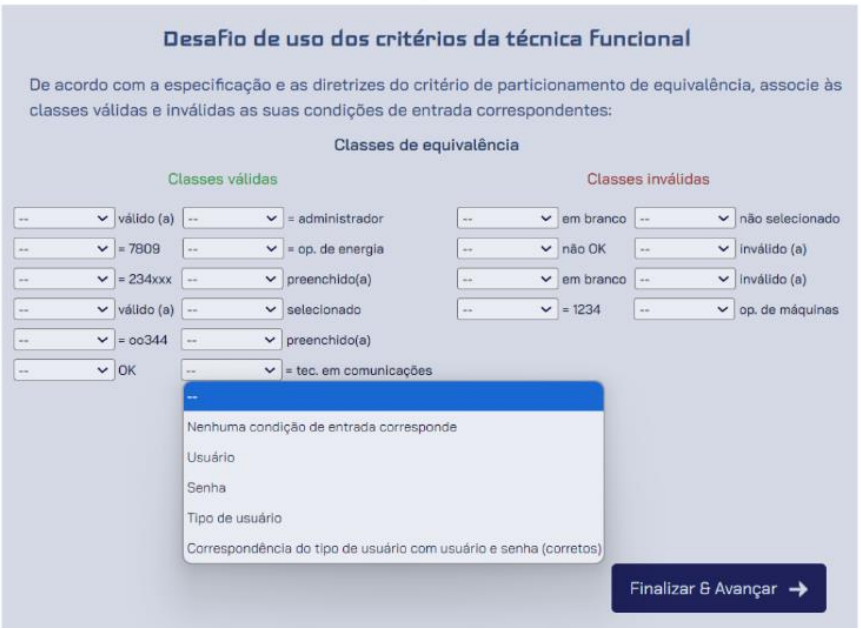
- Para o critério de Particionamento de Equivalência, o desafio consiste em selecionar as classes de equivalência corretas, em uma lista de *checkboxes* para classes válidas e em outra lista de *checkboxes* para classes inválidas, como ocorre no nível 1 (Figura 12a) ou associar às classes válidas e inválidas as suas condições de entrada/saída correspondentes, como ocorre no nível 7 (Figura 12b).

Figura 12 - Desafio de uso dos critérios da técnica funcional – Particionamento de Equivalência

a)



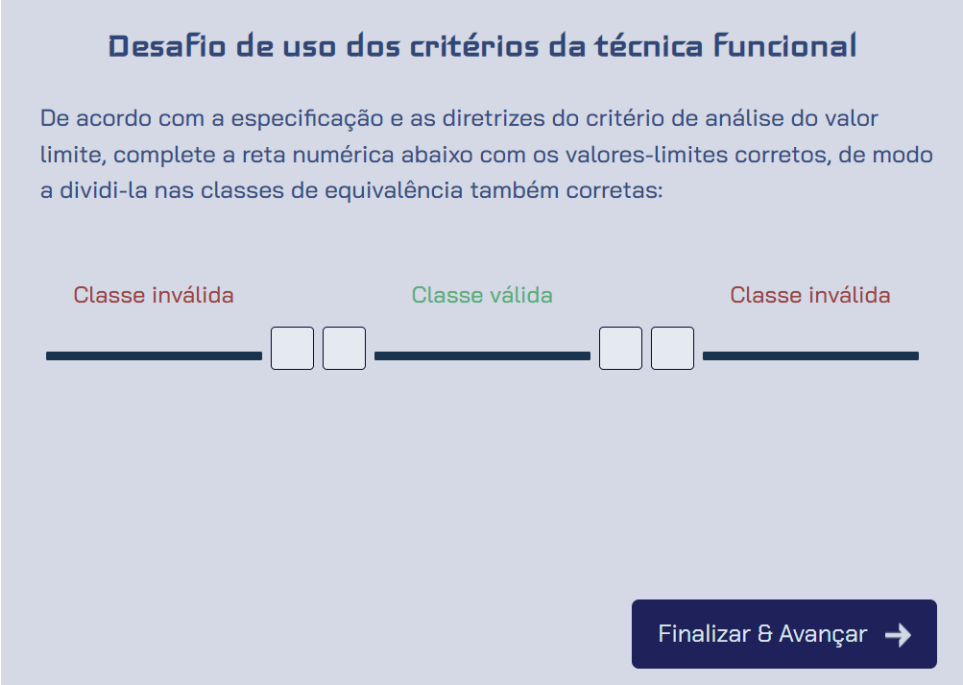
b)



Fonte: Elaborado pela autora.

- Para o critério de Análise do Valor Limite, o desafio consiste em inserir os valores limites corretos em uma reta numérica, de maneira a formar as classes de equivalência também corretas, como ocorre no nível 2 (Figura 13).

Figura 13 - Desafio de uso dos critérios da técnica funcional – Análise do Valor Limite

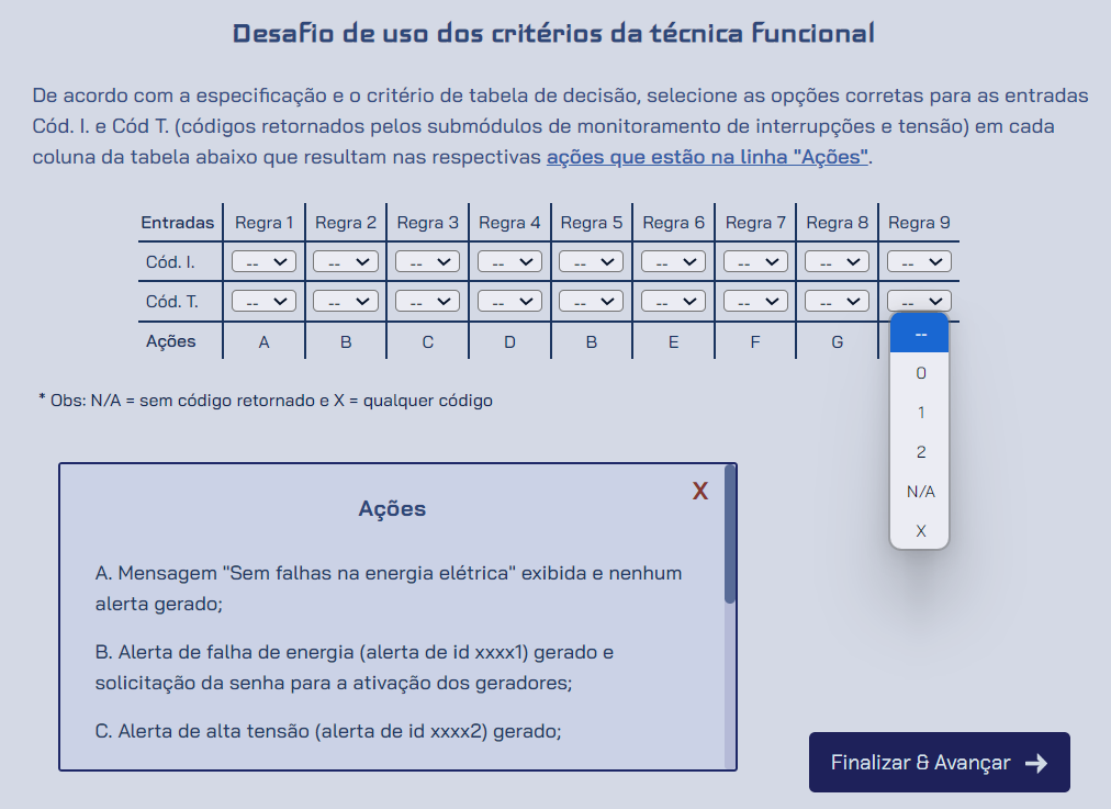


Fonte: Elaborado pela autora.

- Para o critério de Tabela de Decisão, o desafio consiste em completar os campos que faltam em uma tabela de decisão, como ocorre no nível 4 (Figura 14).

Cabe observar que nesta fase, o jogador acumula pontos apenas se a solução completa do desafio estiver correta. Além disso, ele pode encerrar a fase a qualquer momento e avançar para a fase de elaboração de casos de teste do nível, clicando no botão “Finalizar & Avançar” (Figura 11).

Figura 14 - Desafio de uso dos critérios da técnica funcional – Tabela de Decisão



Fonte: Elaborado pela autora.

3.2.5 Fase de elaboração de casos de teste

Na fase de elaboração de casos de teste, ilustrada na Figura 15, o artefato gerado na fase de uso dos critérios da técnica funcional é exibido para que o jogador o use como base para a elaboração dos casos de teste.

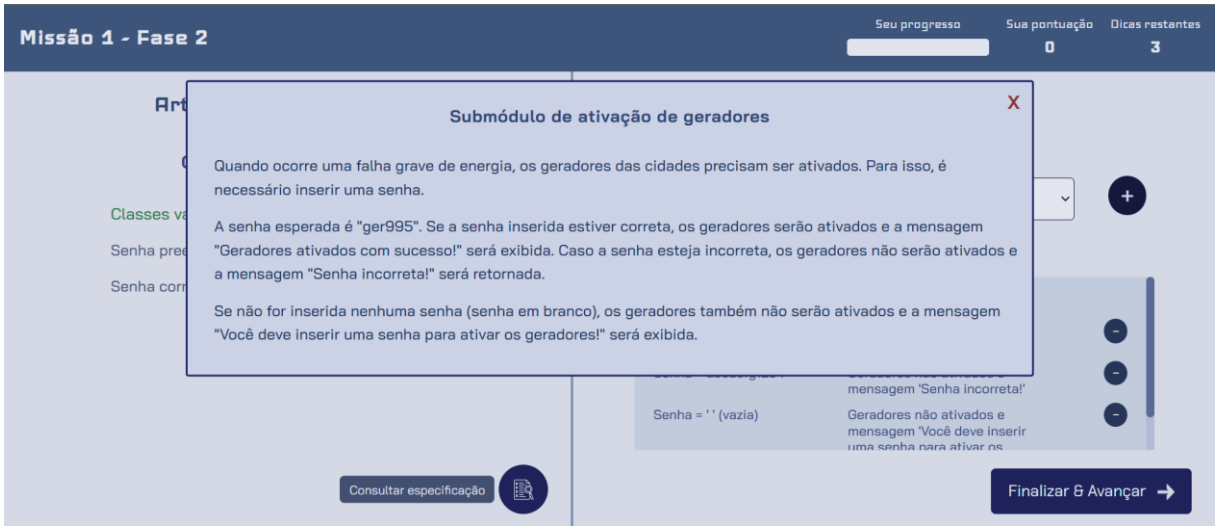
Para elaborar os casos de teste, o jogador deve, em alguns níveis inserir e, em outros níveis, selecionar, em listas de seleção com diversas opções, a(s) entrada(s) e saída(s) esperada(s) e clicar no botão com o símbolo “+”. Nesse contexto, é feita uma validação para que não sejam permitidos casos de teste repetidos ou com entrada e/ou saída esperada em branco/não selecionada. Se o caso de teste se encaixar em alguma dessas situações, será exibido um alerta ao jogador, senão, o par de entrada e saída esperada será considerado e exibido em uma lista de casos de teste gerados, a fim de que o jogador se lembre de quais casos de teste já gerou. Ainda na lista de casos de teste gerados, ao lado de cada par de entrada e saída esperada, há um botão com o símbolo “-”, que possibilita a exclusão de casos de teste.

Figura 15 - Fase de elaboração de casos de teste

Fonte: Elaborado pela autora.

Ademais, como é possível observar na Figura 15, na parte inferior da tela, há um botão que permite que a especificação do submódulo, exibida na primeira fase, seja consultada. Ao clicar no botão, um modal contendo a especificação é aberto em tela, conforme ilustrado na Figura 16.

Figura 16 - Consulta de especificação



Fonte: Elaborado pela autora.

Na fase de elaboração de casos de teste, o jogador ganha pontos para cada caso de teste correto e perde pontos para cada caso de teste incorreto ou excedente que gerar. Casos de teste excedentes, nesse contexto, são casos de teste corretos,

mas que repetem a validação de um aspecto já coberto por um caso de teste já gerado.

Por fim, ao clicar no botão “Finalizar & Avançar” (Figura 15), o jogador avança para a fase de execução dos casos de teste.

3.2.6 Fase de execução dos casos de teste

Na fase de execução dos casos de teste, ilustrada na Figura 17, o jogador deverá executar no sistema fictício os seus casos de teste elaborados na fase anterior. Para isso, ao lado esquerdo da tela, todos os casos de teste gerados na fase anterior são exibidos e, cada caso de teste, possui ao seu lado um *checkbox* para que o jogador possa marcar e ter controle dos casos de teste que já foram executados. Já, ao lado direito da tela, é(são) exibida(s) a(s) parte(s) do sistema fictício a ser(em) testada(s), para que o jogador execute seus casos de teste, inserindo as entradas e avaliando as saídas.

Logo abaixo do ambiente para execução dos casos de teste, há um botão chamado “Reportar Falha”, que o jogador pode acionar assim que encontrar uma falha. Para que o reporte da falha seja válido e forneça pontos ao jogador, as entradas correntes no sistema fictício devem corresponder a entradas que de fato revelam uma falha e, além disso, as entradas correntes não podem corresponder às entradas de alguma falha que já foi reportada anteriormente na fase em questão.

Figura 17 - Fase de execução dos casos de teste

Fonte: Elaborado pela autora.

Assim como na fase de elaboração dos casos de teste, nesta fase também há o botão que possibilita ao jogador consultar a especificação do submódulo que está sendo testado (Figura 17).

Ao clicar no botão “Finalizar & Avançar” (Figura 17), o jogador finaliza a fase e o nível por completo e é direcionado para uma tela de *feedback*.

É importante ressaltar que as falhas que o jogador não tiver encontrado na fase de execução dos casos de teste de determinado nível, permanecem na parte do sistema fictício que foi testada naquele nível durante todo o jogo, ou seja, se em um nível de teste de unidade, por exemplo, o jogador não encontrar determinada falha, em um nível posterior de teste de integração deste módulo com outro módulo, o módulo já testado continuará com a falha não encontrada.

3.2.7 Apresentação de dicas

O jogo inclui o recurso de dicas – um elemento adaptativo que auxilia o jogador em caso de dificuldades. O jogador possui acesso às dicas somente nas fases de execução dos casos de teste, a partir do nível 3, e, além disso, pode utilizar, no máximo, três dicas durante todo o jogo. Ademais, as dicas exibidas ao jogador sempre referem-se à falhas que ele ainda não encontrou. No caso de todas as falhas já terem sido reportadas, a dica exibida é um alerta de que não há mais falhas a serem encontradas na fase atual.

Na Figura 18, é ilustrada a abertura da dica na fase de execução dos casos de teste do nível 6. Para abrir a dica é necessário clicar no botão com o ícone de lâmpada na parte inferior da tela. Após isso, é aberto um modal para que seja confirmada a abertura da dica (Figura 18a). Ao clicar em “Sim”, o modal contendo a dica é aberto (Figura 18b).

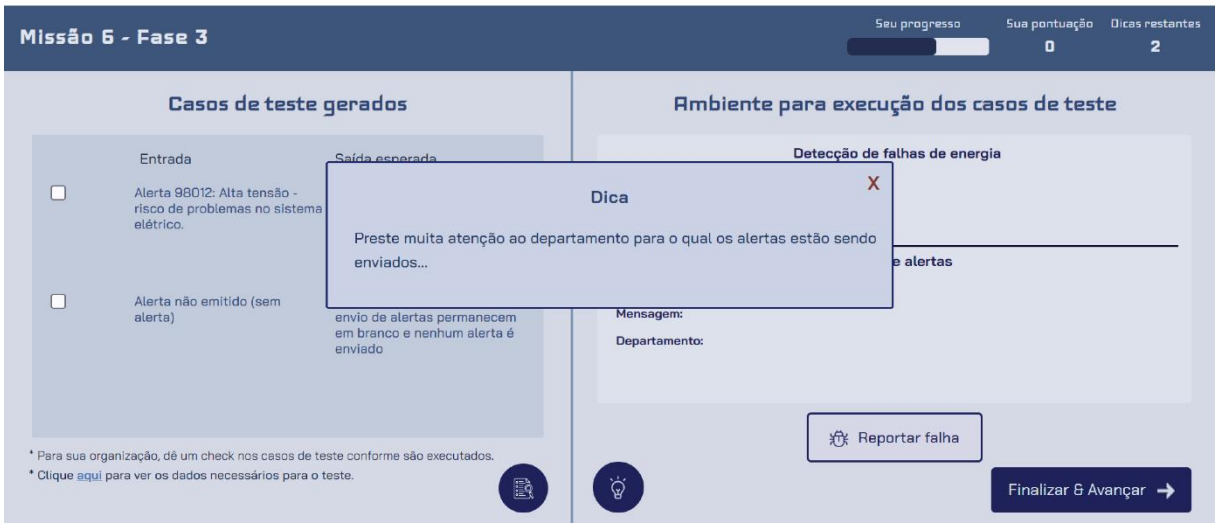
3.2.8 *Feedback* ao final dos níveis

Ao final de cada nível, o jogador recebe um *feedback* detalhado, que contém: uma mensagem personalizada de *feedback*; a pontuação total obtida no nível; a pontuação obtida em cada fase do nível; a resolução esperada de cada fase em conjunto com a resolução fornecida; e a regra da pontuação aplicada em cada fase.

Figura 18 - Abertura de dica

a)

b)



Fonte: Elaborado pela autora.

A mensagem personalizada de *feedback* varia de acordo com a pontuação obtida pelo jogador no nível em questão. Caso a pontuação obtida seja baixa, é exibida uma mensagem que motive o jogador a continuar se esforçando para se aprimorar, como “Você pode melhorar. Continue praticando!”. Já se a pontuação total obtida for alta, é exibida uma mensagem que ressalte o bom desempenho do jogador, como “Excelente! Você se saiu muito bem!”. Além disso, no *feedback* da fase de elaboração de casos de teste, é sinalizado, por meio de ícones de acerto, erro e aviso, ao lado de cada caso de teste gerado pelo jogador, se aquele caso de teste está correto, errado ou é excedente.

A tela de *feedback* do nível 1 é apresentada na Figura 19, contendo: *feedback* da fase de uso dos critérios da técnica funcional (Figura 19a); *feedback* da fase de elaboração de casos de teste (Figura 19b) e *feedback* da fase de execução dos casos de teste (Figura 19c).

3.2.9 Finalização e resultado do jogo

Após todos os níveis terem sido finalizados, há a etapa final do jogo, que representa o teste de aceitação realizado pelo Sr. Clarke. Nesta etapa, o sistema é considerado aprovado ou não para uso com base na pontuação total obtida pelo jogador.

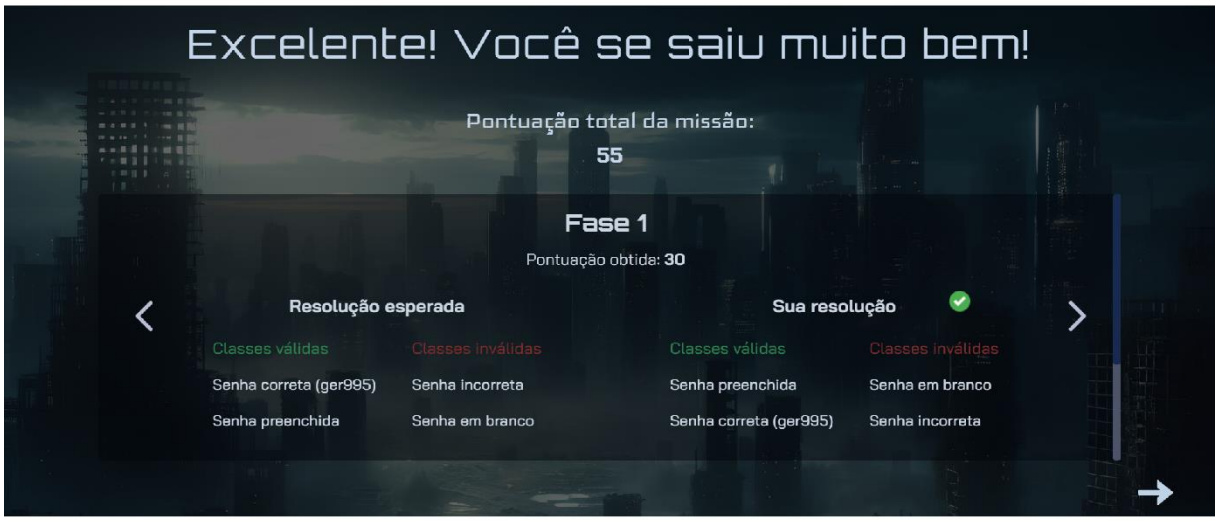
Considerando o sistema de pontuação, o número de fases de uso dos critérios da técnica funcional, a quantidade de casos de teste corretos por nível e a quantidade de falhas de todos os níveis, a pontuação máxima possível no jogo é de 1050 pontos. Para que o sistema seja considerado aprovado para uso, e o jogador alcance a vitória, é necessário obter no mínimo 550 pontos, o que excede em 25 pontos a metade da pontuação máxima, de modo a incorporar o elemento de desafio ao jogo.

Após a conclusão do último nível, a tela apresentada na Figura 20, com o Sr. Clarke avisando que o resultado do seu teste de aceitação será revelado, é exibida ao jogador.

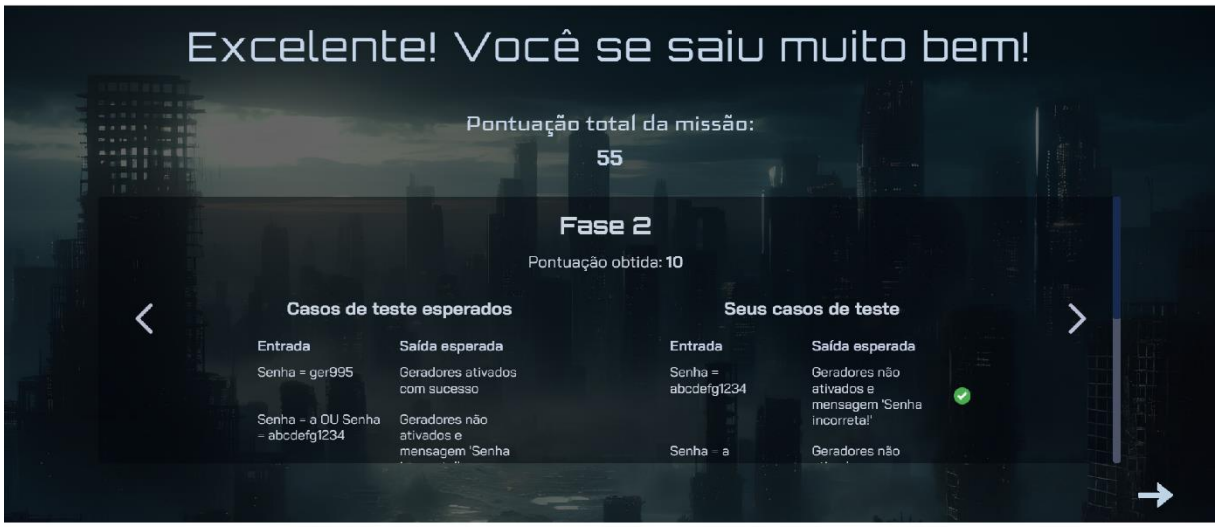
Em seguida, o personagem revela o resultado final do seu teste de aceitação (Figura 21). Caso a pontuação total do jogador seja maior ou igual a 550 pontos, ele é parabenizado, sua pontuação final é exibida e há uma animação de fogos de artifício explodindo no ar (Figura 21a). Caso contrário, uma fala motivacional é dita pelo Sr. Clarke e a pontuação final também é exibida (Figura 21b).

Figura 19 - Tela de feedback – nível 1

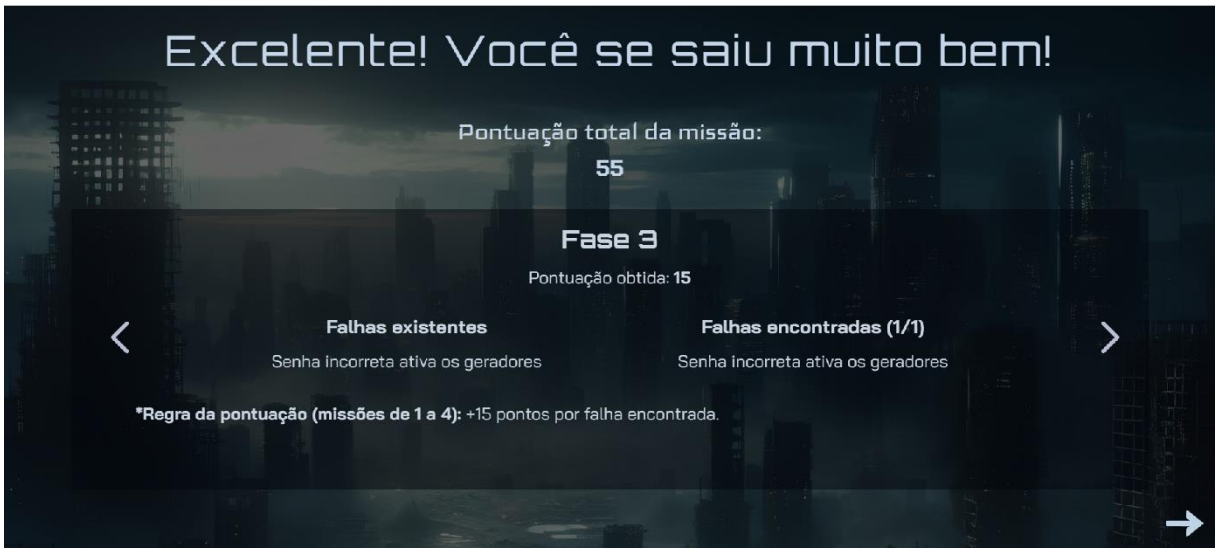
a)



b)

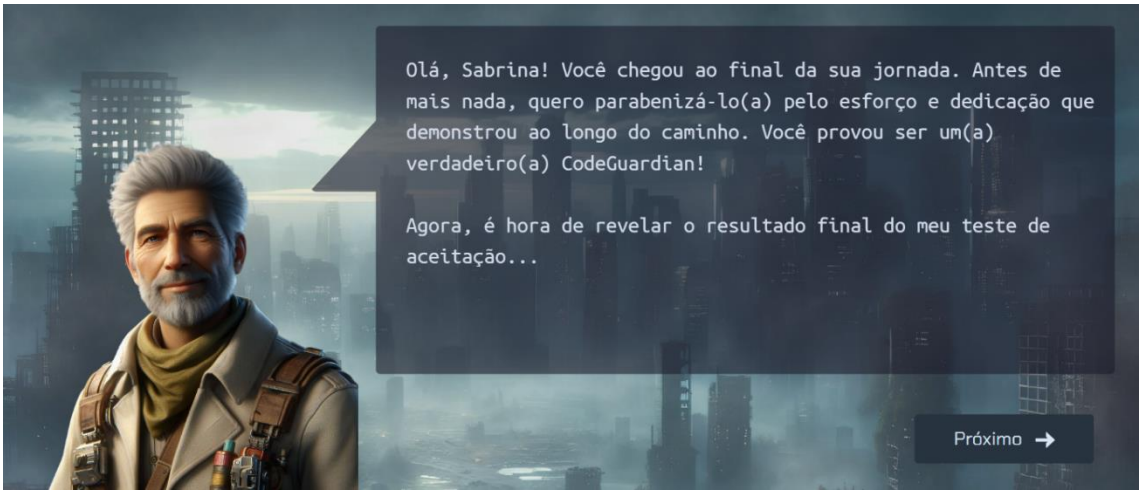


c)



Fonte: Elaborado pela autora.

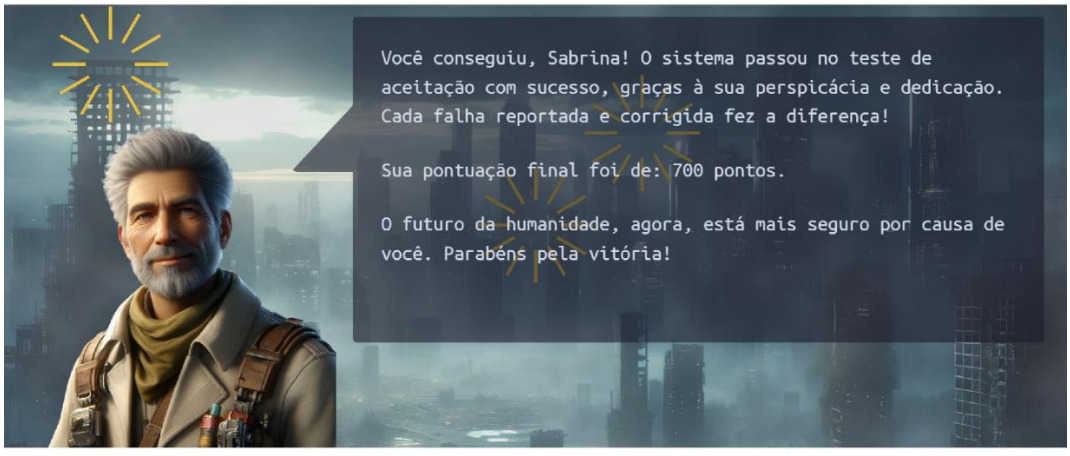
Figura 20 - Aviso de revelação do resultado final do jogo



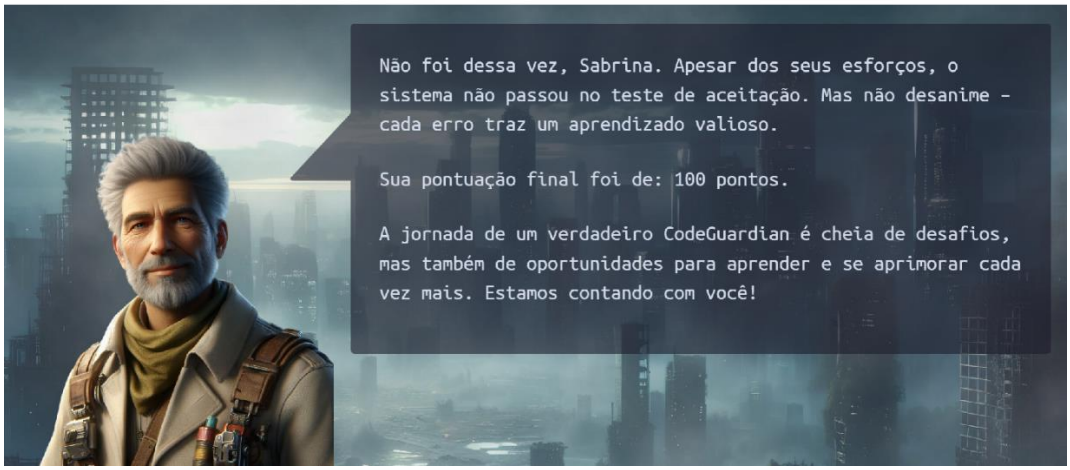
Fonte: Elaborado pela autora.

Figura 21 - Revelação do resultado final do jogo

a)



b)



Fonte: Elaborado pela autora.

3.3 Considerações finais

Neste capítulo foram expostos detalhes sobre a arquitetura e implementação do jogo educacional *CodeGuardians*, bem como sobre suas mecânicas e regras, com base na ilustração das telas correspondentes.

No próximo capítulo, são apresentados os resultados da validação do jogo educacional, ilustrados por meio de gráficos, e as conclusões obtidas.

4 RESULTADOS DO PROCESSO DE AVALIAÇÃO

Neste capítulo são expostos os resultados do processo de avaliação. Para tanto, na seção 4.1 é apresentado o perfil dos participantes, destacando suas principais características e seus conhecimentos prévios na área de testes de *software*, conforme coletado durante o processo de avaliação. Já na seção 4.2, tem-se a avaliação do *CodeGuardians*, apresentando os resultados da avaliação do jogo educacional. Por fim, na seção 4.3, tem-se as considerações finais, apresentando algumas considerações sobre o capítulo.

4.1 Perfil dos participantes

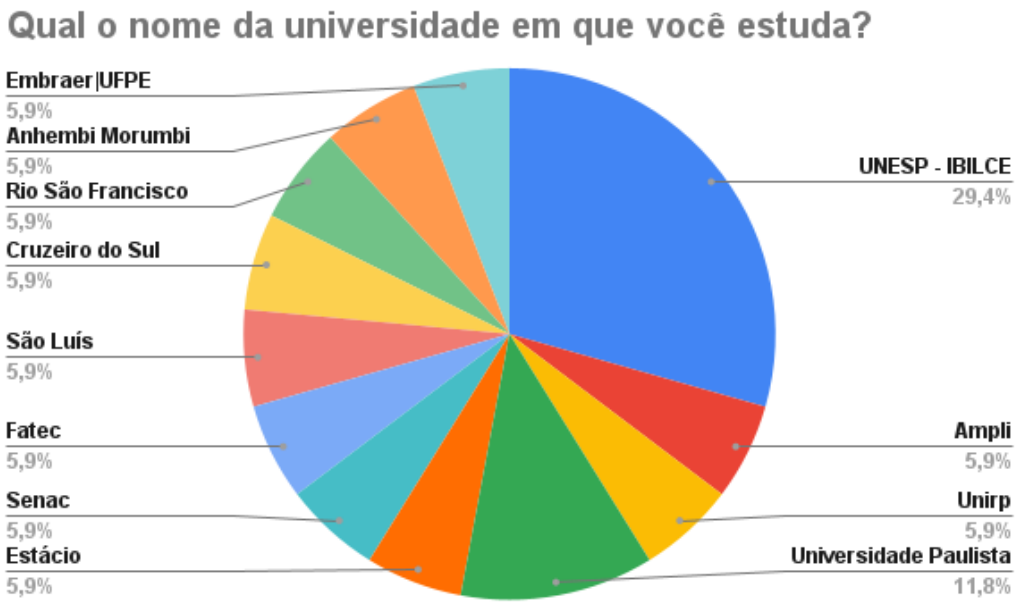
O jogo educacional ficou disponível *online* para avaliação, em conjunto com um formulário, previamente definido (Apêndice B), entre os dias 01 e 06 de outubro de 2024, via link disponibilizado em grupos de *Whatsapp* e redes sociais. O público-alvo foram estudantes de cursos da área da computação que estão cursando ou já cursaram a disciplina de Engenharia de *Software*.

O formulário teve 17 respostas, sendo ele composto por 33 questões e dividido em duas partes: a primeira parte, contendo perguntas sobre perfil e conhecimento prévio na área de testes de *software*, que deveria ser respondida antes do jogo ser iniciado, e a segunda parte, referente a avaliação do jogo, que deveria ser respondida após a conclusão do mesmo. Nesta subseção, são apresentados os resultados referentes à primeira parte do formulário (Apêndice B – Parte 1).

A partir das respostas coletadas sobre gênero, tem-se que 35,3% dos participantes da pesquisa são do gênero feminino, sendo os 64,7% restantes do gênero masculino. Já a faixa de idade dos participantes vai de 19 até 37 anos, sendo que a maioria (64,7%) têm de 21 a 25 anos de idade.

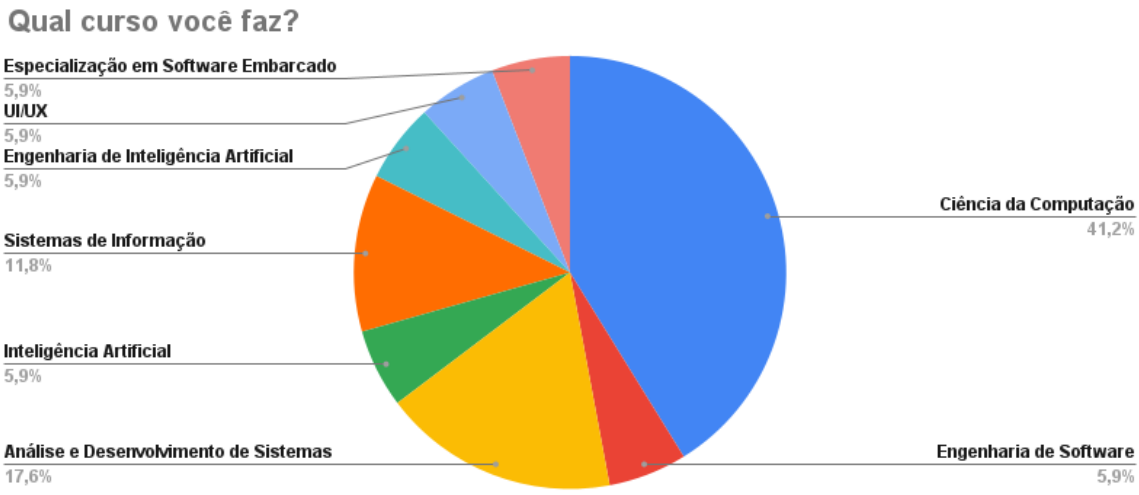
Com base nas Figuras 22, 23 e 24, tem-se que os participantes são estudantes de diversas universidades, cursando diferentes cursos na área de computação, distribuídos em variados semestres acadêmicos, sendo que a maioria estuda na UNESP – IBILCE (29,4%), faz o curso de Ciência da Computação (41,2%) e está no 10º semestre do curso (29,4%).

Figura 22 - Gráfico com resultado da avaliação sobre a pergunta "Qual o nome da universidade em que você estuda?".



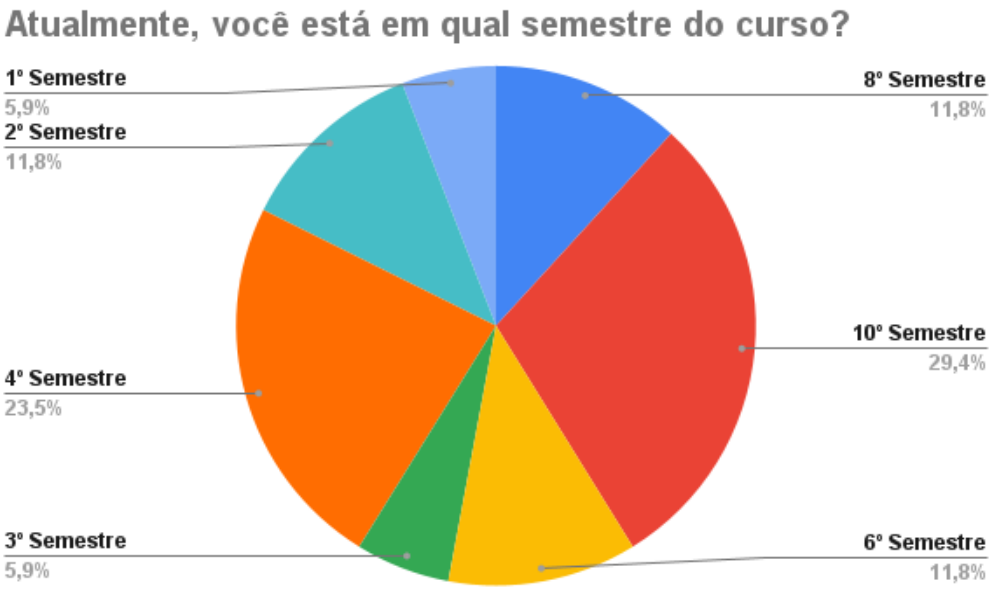
Fonte: Elaborado pela autora.

Figura 23 - Gráfico com resultado da avaliação sobre a pergunta "Qual curso você faz?".



Fonte: Elaborado pela autora.

Figura 24 - Gráfico com resultado da avaliação sobre a pergunta "Atualmente, você está em qual semestre do curso?".



Fonte: Elaborado pela autora.

Acerca do conhecimento prévio dos participantes na área de testes de *software*, a maior parte (76,4%) classificou seu nível de conhecimento na área de teste de *software*, em geral, como intermediário (52,9%) ou básico (23,5%).

Em relação à técnica funcional de testes de *software* e os diferentes níveis de teste, em específico, a maioria (70,6% em relação à técnica funcional e 82,4% em relação aos níveis de teste) também classificou seu conhecimento prévio como básico ou intermediário.

Por fim, o interesse dos participantes pela área de testes de *software* também foi avaliado antes de iniciarem o jogo. Os resultados mostraram que 17,6% estavam totalmente interessados, 29,4% tinham muito interesse, 35,3% apresentaram interesse moderado e 17,6% possuíam pouco interesse pela área. Esses dados revelam que a maioria dos participantes já possuía algum nível de interesse pela área de testes de *software* antes de começar o jogo, o que sugere um engajamento inicial com o tema.

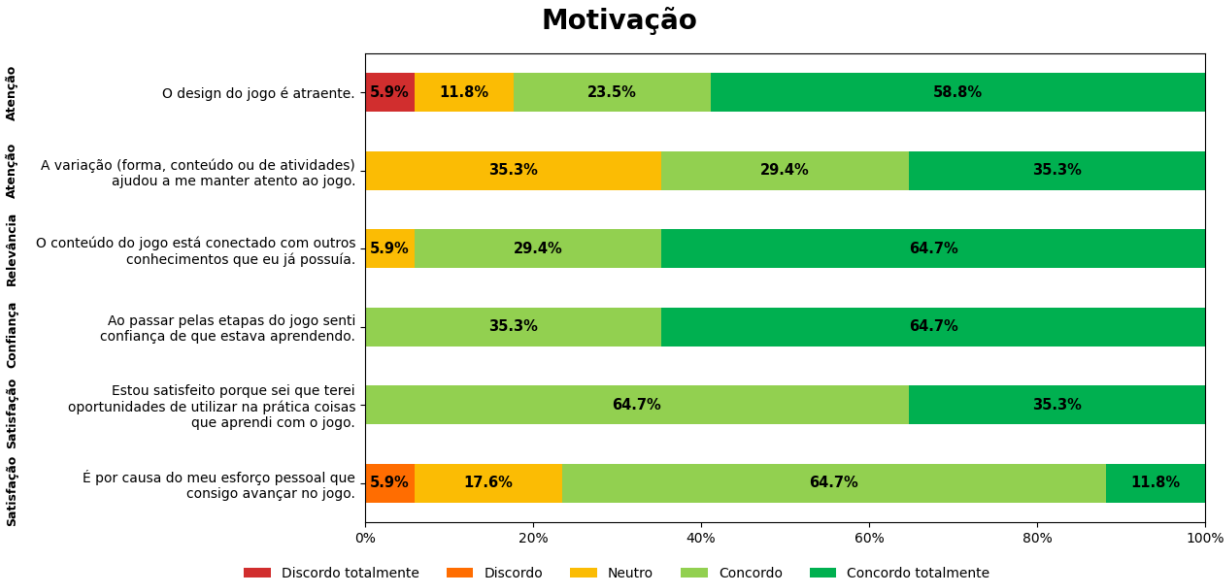
4.2 Avaliação do CodeGuardians

Para a avaliação do *CodeGuardians*, foram, inicialmente, definidas afirmações baseadas no Modelo de Avaliação de Jogos Educacionais – MEEGA, um modelo destinado à avaliação da qualidade de jogos educacionais da área de Engenharia de *Software* (Savi, Wangenheim, Borgatto, 2011). Essas afirmações foram respondidas em uma escala Likert de 5 pontos, que variava de “discordo totalmente” a “concordo totalmente” (Apêndice B – Parte 2). O MEEGA avalia três subcomponentes principais: motivação, experiência do usuário e aprendizagem. Cada um desses subcomponentes abrange diferentes dimensões, como atenção, relevância, divertimento, entre outras (Savi, Wangenheim, Borgatto, 2011).

As respostas coletadas e representadas na Figura 25, dizem respeito ao subcomponente motivação, dividido nas dimensões de atenção, relevância, confiança e satisfação. Na afirmação “O design do jogo é atraente”, 58,8% dos participantes concordam totalmente, enquanto 23,5% concordam, sugerindo que o jogo satisfez os jogadores na questão estética. Em relação à “variação no conteúdo”, observa-se que 35,3% concordam e 29,4% permanecem neutros, indicando uma possível área de melhoria para manter a atenção dos jogadores. A maioria, 64,7%, reconhece que o conteúdo do jogo está conectado com outros conhecimentos que já possuíam, destacando a relevância educacional do jogo. Quanto à confiança dos jogadores no aprendizado, 64,7% concordam totalmente que se sentiram mais confiantes de que estavam aprendendo ao passar pelas etapas do jogo. A satisfação geral é elevada, com 64,7% dos jogadores concordando que seu esforço pessoal promove o avanço no jogo e que veem oportunidades de aplicar o conhecimento adquirido na prática.

Tendo em vista esses resultados, é possível afirmar que o jogo promove uma experiência educativa engajante e relevante para a maioria dos jogadores, embora haja espaço para ajustes na variação de conteúdo para maximizar a atenção dos usuários.

Figura 25 - Gráfico com resultados da avaliação das afirmações referentes ao subcomponente de motivação.

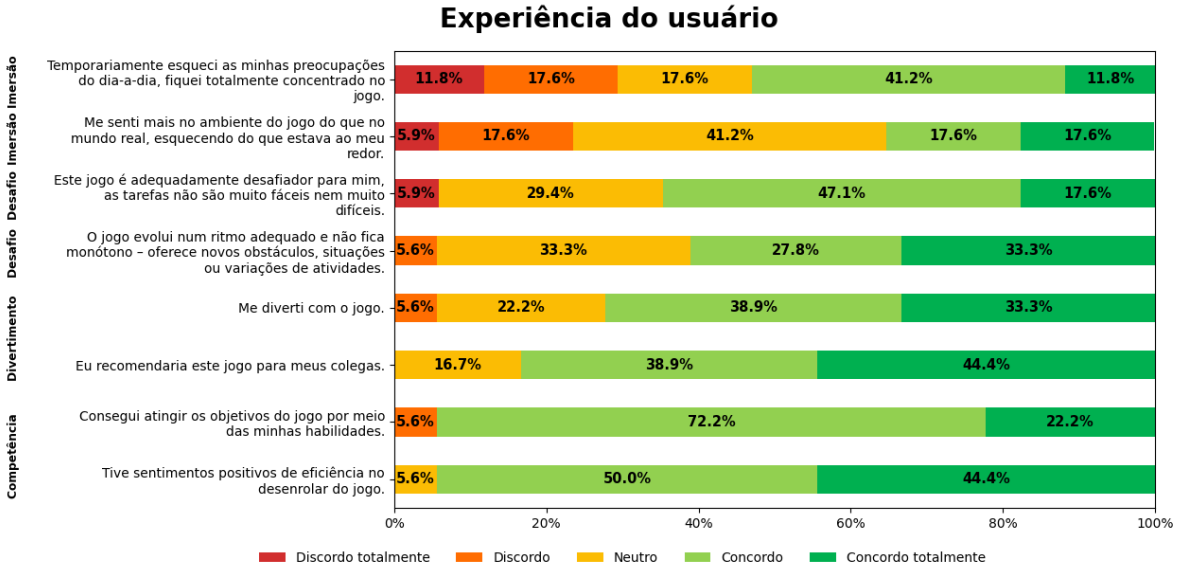


Fonte: Elaborado pela autora.

Os resultados da avaliação do subcomponente de experiência do usuário, conforme apresentado na Figura 26, revelam uma percepção predominantemente positiva dos participantes em relação a aspectos essenciais como imersão, desafio, divertimento e competência. Em termos de imersão, a maioria dos participantes (41,2%) concordou que se concentrou totalmente no jogo, esquecendo temporariamente as preocupações do dia a dia, enquanto 11,8% concordaram totalmente. No entanto, ainda há uma porcentagem significativa de respostas neutras (17,6%) e discordantes (29,4%), sugerindo que o jogo não conseguiu engajar plenamente todos os jogadores. Quanto ao desafio, o jogo parece ter alcançado um bom equilíbrio, com 64,7% dos jogadores concordando que as tarefas possuem um nível de dificuldade adequado e 61,1% percebendo que o ritmo de evolução do jogo foi satisfatório, com novos obstáculos e variações nas atividades. A diversão foi um aspecto amplamente reconhecido, com 72,2% dos jogadores afirmando que se divertiram durante o jogo. Além disso, 83,3% dos participantes declararam que recomendariam o jogo para seus colegas. No que diz respeito à competência, 94,4% dos jogadores concordaram que conseguiram atingir os objetivos propostos pelo jogo por meio de suas habilidades, reforçando a sensação de eficiência ao longo das atividades. Esses resultados indicam que o *CodeGuardians*, enquanto jogo educacional, cumpre bem seu papel de proporcionar uma experiência desafiadora e

envolvente, embora haja oportunidades para aprimorar o nível de imersão entre os jogadores que apresentaram respostas neutras ou discordantes.

Figura 26 - Gráfico com resultados da avaliação das afirmações referentes ao subcomponente de experiência do usuário.

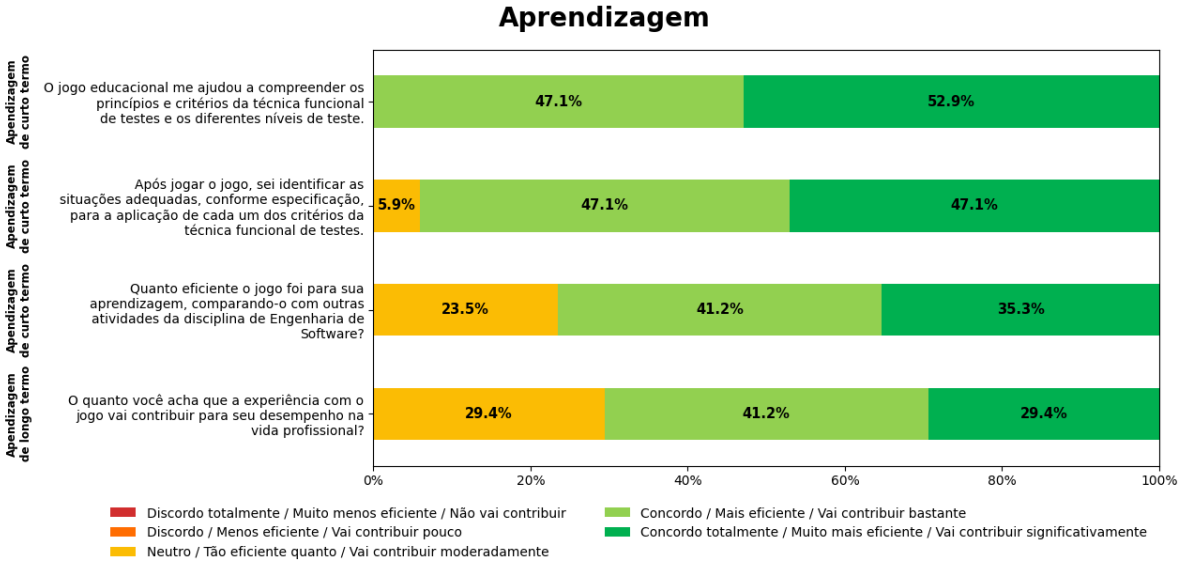


Fonte: Elaborado pela autora.

A avaliação do jogo educacional na categoria de aprendizagem apresentou resultados bastante positivos, indicando que o jogo contribuiu de forma significativa para o entendimento dos princípios e critérios da técnica funcional de testes. Conforme ilustrado na Figura 27, mais da metade dos jogadores (52,9%) concordaram totalmente com a afirmação de que o jogo ajudou a compreender esses conceitos, enquanto 47,1% concordaram. Além disso, a mesma proporção de 47,1% dos jogadores concordou que, após jogar o jogo, foi capaz de identificar as situações adequadas para a aplicação de cada critério, reforçando o aspecto prático para aprendizado que o jogo proporciona. Quando comparado a outras atividades da disciplina de Engenharia de *Software*, 41,2% dos participantes avaliaram o jogo como mais eficiente que outras abordagens, enquanto 35,3% consideraram-no muito mais eficiente. Em relação ao impacto de longo prazo, 29,4% acreditam que a experiência com o jogo contribuirá significativamente para o desempenho profissional, enquanto 41,2% esperam que irá contribuir bastante.

Esses dados evidenciam o potencial do jogo como uma ferramenta de aprendizado aplicada na área de Engenharia de *Software*, com capacidade de impactar diretamente o desenvolvimento profissional dos jogadores.

Figura 27 - Gráfico com resultados da avaliação das afirmações/perguntas referentes ao subcomponente de aprendizagem.



Fonte: Elaborado pela autora.

Além das afirmações baseadas no modelo MEEGA, os participantes também responderam, após terem concluído o jogo, às mesmas afirmações acerca do seu conhecimento e interesse na área de testes de *software*, respondidas antes de terem iniciado o jogo. A partir disso, foi possível depreender a contribuição do jogo *CodeGuardians* para o processo de ensino-aprendizagem, conforme descrito a seguir.

Na Tabela 3, observa-se um aumento no percentual de participantes que classificaram seu nível de conhecimento geral sobre testes de software como avançado, passando de 23,5% para 35,3%. Paralelamente, o percentual de participantes que classificaram seu conhecimento como intermediário e básico diminuiu de 52,9% para 47,1% e de 23,5% para 17,6%, respectivamente.

Em relação à técnica funcional e aos níveis de teste, especificamente, o conhecimento dos participantes também apresentou uma melhora (Tabelas 4 e 5). No que diz respeito à técnica funcional, o percentual de participantes que se classificaram com conhecimento intermediário cresceu de 29,4% para 41,2%, e aqueles que se consideraram avançados aumentaram de 17,6% para 35,3%. Em relação aos níveis

de teste, o percentual de participantes com conhecimento intermediário subiu de 41,2% para 64,7%, enquanto o percentual de participantes com conhecimento avançado passou de 11,8% para 17,6%.

Tabela 3 - Nível de conhecimento dos participantes em testes de software antes e após a utilização do jogo educacional.

Nível de conhecimento em testes de software (em geral)		
Classificação	**Antes de jogar o jogo**	**Após jogar o jogo**
Nenhum	0%	0%
Básico	23,5%	17,6%
Intermediário	52,9%	47,1%
Avançado	23,5%	35,3%
Especialista	0%	0%

Fonte: Elaborado pela autora.

Tabela 4 - Nível de conhecimento dos participantes sobre a técnica funcional de testes antes e após a utilização do jogo educacional.

Nível de conhecimento sobre a técnica funcional de testes		
Classificação	**Antes de jogar o jogo**	**Após jogar o jogo**
Nenhum	11,8%	0%
Básico	41,2%	23,5%
Intermediário	29,4%	41,2%
Avançado	17,6%	35,3%
Especialista	0%	0%

Fonte: Elaborado pela autora.

Além disso, após jogar o jogo, nenhum participante afirmou não possuir conhecimento sobre esses tópicos. Antes de iniciar o jogo, 11,8% dos participantes indicaram não ter conhecimento sobre a técnica funcional de testes (Tabela 4), e 5,9% afirmaram não ter conhecimento sobre os níveis de teste (Tabela 5). Esses resultados refletem uma melhora significativa no nível de conhecimento dos participantes sobre a técnica funcional, os níveis de teste e também sobre a área de testes de *software*,

em geral, após a interação com o jogo educacional. Com isso, é possível afirmar que o jogo facilitou, de maneira abrangente, a compreensão dos princípios e técnicas de teste de *software* em questão.

Tabela 5 - Nível de conhecimento dos participantes sobre os níveis de testes antes e após a utilização do jogo educacional.

Nível de conhecimento sobre os níveis de testes		
Classificação	**Antes de jogar o jogo**	**Após jogar o jogo**
Nenhum	5,9%	0%
Básico	41,2%	17,6%
Intermediário	41,2%	64,7%
Avançado	11,8%	17,6%
Especialista	0%	0%

Fonte: Elaborado pela autora.

Adicionalmente, após jogarem o jogo, os participantes reavaliaram seu nível de interesse pela área de testes de *software* (Tabela 6). Comparando com a classificação inicial, realizada antes do início do jogo, observou-se um aumento significativo no interesse dos participantes: o percentual daqueles que demonstraram muito interesse pela área cresceu de 29,4% para 41,2%, enquanto o percentual de participantes totalmente interessados aumentou de 17,6% para 23,5%. Paralelamente, o percentual de participantes com pouco interesse reduziu-se de 17,6% para 0%. Além disso, mais da metade dos participantes (64,7%) afirmou estar muito ou totalmente motivada a aprofundar seus conhecimentos em testes de *software* após a conclusão do jogo, e 35,3% afirmou estar moderadamente motivada, conforme apresentado na Figura 28. Esses resultados indicam que o jogo teve um impacto positivo tanto no aumento do interesse, quanto na motivação dos participantes para se aprofundarem na área de testes de *software*.

Por fim, foi disponibilizado um campo aberto de preenchimento opcional para que os participantes pudessem sugerir melhorias para o jogo, possibilitando a identificação de pontos a serem aprimorados. Dentre as sugestões recebidas, muitos participantes propuseram a inclusão de música de fundo e efeitos sonoros, argumentando que tais elementos tornariam a experiência de jogo mais imersiva.

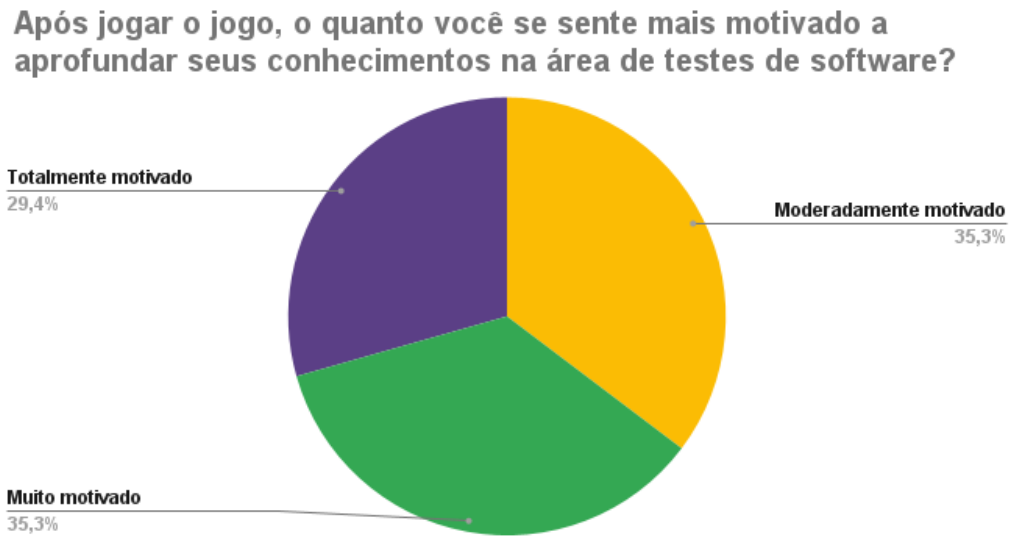
Outras sugestões incluem a adição de uma funcionalidade para salvar o progresso do jogador, a adição de um *ranking* de pontuação dos jogadores, a possibilidade de obter mais dicas ao longo das missões e implementar um "*fail-state*", onde algo negativo aconteça ao jogador em caso de pontuação muito baixa. Essas sugestões fornecem importantes *insights* que podem ser implementados para aprimorar a jogabilidade, aumentar a imersão e tornar o jogo uma ferramenta educacional ainda mais eficaz.

Tabela 6 - Nível de interesse dos participantes pela área de testes de software antes e após a utilização do jogo educacional.

Nível de interesse pela área de testes de *software*		
Classificação	**Antes de jogar o jogo**	**Após jogar o jogo**
Nenhum interesse	0%	0%
Pouco interesse	17,6%	0%
Interesse moderado	35,3%	35,3%
Muito interesse	29,4%	41,2%
Totalmente interessado	17,6%	23,5%

Fonte: Elaborado pela autora.

Figura 28 - Gráfico com resultado da avaliação sobre a pergunta "Após jogar o jogo, o quanto você se sente mais motivado a aprofundar seus conhecimentos na área de testes de software?".



Fonte: Elaborado pela autora.

4.3 Considerações finais

Neste capítulo, foram apresentados os resultados da avaliação do jogo educacional, realizada através de um questionário online, por estudantes de cursos da área de computação que estão cursando ou já cursaram a disciplina de Engenharia de *Software*. As respostas coletadas indicam que a maioria dos jogadores classificou o *CodeGuardians* como um jogo educacional com jogabilidade envolvente e divertida. Além disso, o jogo também foi reconhecido por sua contribuição significativa para o aprendizado e para o aumento do interesse pela área de testes de *software*.

No próximo capítulo, é apresentada uma conclusão baseada nos resultados mostrados, além de possíveis trabalhos futuros.

5 CONCLUSÃO

Neste capítulo, são expostas as conclusões que se tem a partir dos resultados obtidos. Para tanto, na seção 5.1, tem-se as contribuições do *CodeGuardians* para o ensino da área de testes de *software* e, por fim, na seção 5.2, tem-se os trabalhos futuros, apresentando todas as sugestões para melhoria do jogo educacional.

5.1 Contribuições do trabalho

Considerando, por um lado, a crescente importância dos testes de *software* para garantir a qualidade e o correto funcionamento dos sistemas, e, por outro, a escassez de profissionais qualificados nessa área, decorrente principalmente da desmotivação dos estudantes e da falta de atenção adequada no contexto educacional, abordagens de ensino inovadoras que integrem teoria e prática, como os jogos educacionais, podem desempenhar um papel fundamental no aumento da motivação e na facilitação do processo de aprendizado em testes de software, o que, por sua vez, contribuiria para a formação de profissionais mais qualificados, e, por consequência, para o aumento da qualidade dos produtos de *software* desenvolvidos na indústria. Nesse contexto, foi identificado que os jogos educacionais existentes que abordam a técnica de testes mais utilizada nas empresas brasileiras – a técnica funcional – carecem de alguns aspectos relevantes, como explorar a aplicação dos critérios da técnica funcional em todos os níveis de teste e incorporar em sua jogabilidade uma simulação realista e prática de elaboração e execução dos testes. Diante disso, verificou-se então a pertinência do presente trabalho que, conforme ilustrado na Tabela 7, atende todas as características de conteúdo e jogo analisadas previamente para os jogos educacionais correlatos (ver Capítulo 2 – Tabela 2).

Porém, é importante observar que, em razão da amostra de conveniência, usada no processo de avaliação qualitativo realizado, contendo uma quantidade baixa de participantes, onde uma parte significativa já possuía conhecimento e interesse prévios na área de testes de *software*, não é possível generalizar os resultados obtidos. Logo, ainda é preciso avaliar tais resultados por meio de uma pesquisa complementar quantitativa e com um maior número de participantes, principalmente que não tenham um engajamento e conhecimentos iniciais significativos com o tema.

Tabela 7 - Características de conteúdo e jogo abrangidas pelo *CodeGuardians*.

Jogo Educacional	Conteúdo					Elementos de jogo		
	Foco na técnica funcional	Critérios funcionais	Diferentes níveis de teste	Elaboração de casos de teste	Simulação de execução de casos de teste	Feedbacks	Narrativa	Adaptação e personalização
CodeGuardians								

	Atende		Atende em partes		Não atende

Fonte: Elaborado pela autora

O *CodeGuardians* foi avaliado por meio de um questionário *online* disponibilizado a estudantes de cursos da área de computação que estão cursando ou já cursaram a disciplina de Engenharia de *Software*. Com base nas respostas dos participantes, conclui-se que este trabalho apresenta uma contribuição significativa para o ensino de testes de *software*. Os resultados indicam que o jogo, além de proporcionar uma experiência lúdica e envolvente, aumentou a motivação dos jogadores para o aprendizado de testes de *software*. Além disso, o jogo facilitou a compreensão tanto da teoria quanto da aplicação prática dos critérios da técnica funcional e dos diferentes níveis de teste, validando assim as hipóteses inicialmente estabelecidas.

5.2 Trabalhos futuros

Por meio das sugestões de melhorias obtidas a partir dos participantes da avaliação do jogo, foi possível identificar pontos de melhorias para evolução do *CodeGuardians* e assim estabelecer objetivos para trabalhos futuros. Em primeiro lugar, vislumbra-se a criação da funcionalidade de cadastro e login no jogo, para que o progresso dos jogadores seja salvo e não seja mais preciso jogar todo o jogo de

uma única vez para chegar ao final. Com a funcionalidade de login implementada, outra implementação desejada é uma aba de *ranking*, onde os jogadores são classificados de acordo com a sua pontuação, podendo assim aumentar ainda mais o seu engajamento. Além disso, outras possíveis melhorias futuras são: adicionar uma música de fundo e efeitos sonoros ao jogo, implementar uma regra onde os jogadores são recompensados com mais dicas conforme sua pontuação aumenta e implementar um "*fail-state*", em que se a pontuação do jogador estiver muito baixa em relação ao seu nível atual, mais falhas serão inseridas no sistema fictício.

Deste modo, fica claro que os trabalhos futuros sugeridos visam a melhora da jogabilidade e experiência do usuário no jogo, fazendo que ele fique mais envolvente e imersivo. Por fim, perante o exposto, é possível afirmar que o objetivo do trabalho foi atingido.

REFERÊNCIAS

ABDULLAHI, S.; ZAKARI, A.; ABDU, H.; NURA, A.; ZAYYAD, M. A.; SULEIMAN, S.; ADAMU, A.; MASHASHA, A. S. Software testing: Review on tools, techniques and challenges. **International Journal of Advanced Research in Technology and Innovation**, v. 2, n. 2, p. 11-18, 2020.

ADAMS, E. **Fundamentals of game design**. 3. ed. Upper Saddle River, EUA: New Riders, 2014.

ADOBE EXPERIENCE CLOUD**. Single-page applications (SPAs) - what they are and how they work.** 2023. Disponível em: <https://business.adobe.com/blog/basics/learn-the-benefits-of-single-page-apps-spa>. Acesso em: 26 de junho de 2024.

AL-AZAWI, R.; AL-FALITI, F.; AL-BLUSHI, M. Educational gamification vs. game based learning: Comparative study. **International Journal of Innovation, Management and Technology**, v. 7, n. 4, p. 132-136, 2016.

ANDRADE, M. **Qualidade de Software**. 1.ed. Rio de Janeiro: Estácio, 2015.

ARAUJO, N.; MACHADO, R.; VIANA, D.; RIVERO, L. Avaliando a viabilidade do blackbox em sala de aula: Um jogo sério para ensino de teste funcional de software. In: **Anais do XXVIII Simpósio Brasileiro de Informática na Educação (SBIE 2017)**, p. 817-826, 2017.

BARESI, L.; PEZZE, M. An introduction to software testing. **Electronic Notes in Theoretical Computer Science**, v. 148, n. 1, p. 89-111, 2006.

BARTIÉ, A. **Garantia da Qualidade de Software.** 13.ed. Rio de Janeiro: Elsevier Editora, 2002.

BASILI, V. R.; SELBY, R. W. Comparing the effectiveness of software testing strategies. **IEEE Transactions on Software Engineering**, n. 12, p. 1278-1296, 1987.

BEPPE, T. A.; ARAÚJO, I. L.; ARAGÃO, B. S.; SANTOS, I. S.; XIMENES, D.; ANDRADE, R. M. C. Greatest: A card game to motivate the software testing learning. In: **Proceedings of the XXXII Brazilian Symposium on Software Engineering**, p. 298-307, 2018.

BLANCO, R.; TRINIDAD, M.; SUAREZ-CABAL, M. J.; CALDERÓN, A.; RUIZ, M.; TUYA, J. Can gamification help in software testing education? Findings from an empirical study. **Journal of Systems and Software**, v. 200, p. 111647, 2023.

BRITTON, T.; JENG, L.; CARVER, G.; CHEAK, P.; KATZENELLENBOGEN, T. Reversible debugging software. **Judge Bus. School, Univ. Cambridge, Cambridge, UK, Tech. Rep**, v. 229, 2013.

CLARKE, P. J.; DAVIS, D.; KING, T. M.; PAVA, J.; JONES, E. L. Integrating testing into software engineering courses supported by a collaborative learning environment. **ACM Transactions on Computing Education (TOCE)**, v. 14, n. 3, p. 1-33, 2014.

COPELAND, L. **A Practitioner's Guide to Software Test Design**. Norwood, MA, EUA: Artech House, 2004.

CRAWFORD, C. **Chris Crawford on game design**. Upper Saddle River, NJ, EUA: New Riders Publishing, 2003.

DE FREITAS, S.; OLIVER, M. How can exploratory learning with games and simulations within the curriculum be most effectively evaluated?. **Computers & Education**, v. 46, n. 3, p. 249-264, 2006.

DE JESUS, G. M.; FERRARI, F. C.; PORTO, D. P.; FABBRI, S. C. P. F. Gamification in software testing: A characterization study. In: **Proceedings of the III Brazilian Symposium on Systematic and Automated Software Testing**, p. 39-48, 2018.

DE JESUS, G. M.; FERRARI, F. C.; PASCHOAL, L. N.; SOUZA, S.; PORTO, D. P.; DURELLI, V. H. S. Is it worth using gamification on software testing education? An extended experience report in the context of undergraduate students. **Journal of Software Engineering Research and Development**, v. 8, p. 6: 1-6: 19, 2020.

DELAMARO, M. E.; MALDONADO, J. C.; JINO, M. **Introdução ao Teste de Software**. 2.ed. Rio de Janeiro: Elsevier Editora, 2016.

DEMPSEY, J.; RASMUSSEN, K.; LUCASSEN, B. Instructional gaming: implications for instructional technology. In: **Annual meeting of The Association for Educational Communications and Technology,** Nashville, EUA, 1994.

DINIZ, L. L.; DAZZI, R. LS. Jogo digital para o apoio ao ensino do teste de caixa-preta. In: **Anais do X Simpósio Brasileiro de Qualidade de Software**, p. 231-245, 2011.

DINIZ, L. **Jogo das 7 Falhas:** Um jogo educacional para o apoio ao ensino do teste de caixa-preta. Dissertação de Curso de Mestrado, Computação Aplicada, UNIVALI, São José, 2011.

DÖRNER, R.; GÖBEL, S.; EFFELSBERG, W.; WIEMEYER, J. **Serious games**: Foundations, concepts and practice. 1. ed. Cham, Suíça: Springer International Publishing, 2016.

ELBAUM, S.; PERSON, S.; DOKULIL, J.; JORDE, M. Bug hunt: Making early software testing lessons engaging and affordable. In: **29th International Conference on Software Engineering (ICSE'07)**. IEEE, p. 688-697, 2007.

ELLINGTON, H; ADDINALL, E; PERCIVAL, F. **Games and Simulations in Science Education**. Nova Iorque, EUA: Nichols Publishing Company, 1981.

EL-SHAMY, S. **Training games:** Everything you need to know about using games to reinforce learning. Virginia, EUA: Stylus Publishing, 2001.

FALKEMBACH, G. A. M. O lúdico e os jogos educacionais. **CINTED-Centro Interdisciplinar de Novas Tecnologias na Educação, UFRGS**, p. 911, 2006.

FERNANDES, R. **Engenharia de Software**. Curitiba: Editora Fael, 2017.

FRASER, G. Gamification of software testing. In: **2017 IEEE/ACM 12th International Workshop on Automation of Software Testing (AST)**. IEEE, p. 2-7, 2017.

GAROUSI, V.; RAINER, A.; LAUVAS JR, P.; ARCURI, A. Software-testing education: A systematic literature mapping. **Journal of Systems and Software**, v. 165, p. 110570, 2020.

GRAHAM, D.; VEENENDAAL, E.; EVANS, I.; BLACK, R. **Foundations of software testing**: ISTQB certification. Andover (Reino Unido): Cengage Learning Emea, 2008.

GRANDO, R. C. **O jogo e a Matemática no contexto da sala de aula**. São Paulo: Paulus Editora, 2004.

HERZ, J. C. **Joystick nation:** How videogames ate quarters, won our hearts, and rewired our minds. [s.l.] Little Brown and Company, 1997.

HOODA, I.; CHHILLAR, R. S. Software Test Process, Testing Types and Techniques. **International Journal of Computer Applications**, v. 111, n. 13, p. 10–14, 2015.

IKE, T. C.; HOE, T. W.; KIM, J. L. E.; Y'NG, N. Y. Exploring User Experience from an Emotional Context When Designing Immersive Games for Education. **Journal of ICT in Education**, v. 8, n. 1, p. 10-25, 2021.

JIA, S.; YANG, C. Teaching software testing based on cdio. **World Transactions on Engineering and Technology Education**, v. 11, n. 4, p. 476-479, 2013.

JORGENSEN, P. C. **Software testing: A craftsman's approach**. 4.ed. Filadélfia, PA, EUA: Auerbach, 2014.

KAMSTIES, E.; LOTT, C. M. An empirical evaluation of three defect-detection techniques. In: **European Software Engineering Conference**. Berlin, Heidelberg: Springer Berlin Heidelberg, p. 362-383, 1995.

KOSA, M.; YILMAZ, M.; O'CONNOR, R.; CLARKE, P. M. Software engineering education and games: a systematic literature review. **Journal of Universal Computer Science**, v. 22, n. 12, p. 1558-1574, 2016.

KRASNER, H. The cost of poor software quality in the US: A 2022 report. **Proc. Consortium Inf. Softw. QualityTM (CISQTM)**, p. 1-61, 2022.

LI, N; ZHANG, B. The research on single page application front-end development based on Vue. In: **Journal of Physics: Conference Series**, p. 012030, 2021.

LOZZA, R.; RINALDI, G. P. O uso dos jogos para a aprendizagem no ensino superior. **Caderno PAIC**, v. 18, n. 1, p. 575-592, 2017.

MALONE, T. W. Toward a theory of intrinsically motivating instruction. **Cognitive science**, v. 5, n. 4, p. 333-369, 1981.

MATHUR, S.; MALIK, S. Advancements in the V-Model. **International Journal of Computer Applications**, v. 1, n. 12, p. 29-34, 2010.

MYERS, G. J.; SANDLER, C.; BADGETT, T.; THOMAS, T. M. **The art of software testing**. 2. ed. Nashville, TN, EUA: John Wiley & Sons, 2004.

MYERS, G. J.; SANDLER, C.; BADGETT, T. **The art of software testing**. 3.ed. Nashville, TN, EUA: John Wiley & Sons, 2012.

NETO, A. Introdução a teste de software. **Engenharia de Software Magazine**, v. 1, p. 22, 2007.

PRENSKY, M. Fun, play and games: What makes games engaging. **Digital game-based learning**, v. 5, n. 1, p. 5-31, 2001.

PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de Software**: uma abordagem profissional. 9.ed. Porto Alegre: AMGH Editora, 2021.

RODRIGUES, P.; SOUZA, M.; FIGUEIREDO, E. Games and gamification in software engineering education: A survey with educators. In: **2018 IEEE Frontiers in Education Conference (FIE)**. IEEE, p. 1-9, 2018.

SANTOS, I.; MELO, S. M.; SOUZA, P. S. L.; SOUZA, S. R. S. A survey on the practices of software testing: a look into Brazilian companies. **Journal of Software Engineering Research and Development**, v. 10, p. 11: 1-11: 15, 2022.

SAVI, R.; WANGENHEIM, C.; BORGATTO, A. Um modelo de avaliação de jogos educacionais na engenharia de software. In: **Anais do XXV Simpósio Brasileiro de Engenharia de Software (SBES 2011), São Paulo**, 2011.

SAWANT, A. A.; BARI, P. H.; CHAWAN, P. M. Software testing techniques and strategies. **International Journal of Engineering Research and Applications (IJERA)**, v. 2, n. 3, p. 980-986, 2012.

SEVERO, J.; LELLI, V. Testing Maze: an educational game for teaching functional testing. In: **Proceedings of the XXXVII Brazilian Symposium on Software Engineering**, p. 407-415, 2023.

SILVA, A. C. **Jogo educacional para apoiar o ensino de técnicas para elaboração de testes de unidade**. Dissertação de Curso de Mestrado, Computação Aplicada, UNIVALI, São José, 2010.

SILVIS-CIVIDJIAN, N. Awesome bug manifesto: Teaching an engaging and inspiring course on software testing (position paper). In: **2021 Third International Workshop on Software Engineering Education for the Next Generation (SEENG)**. IEEE, p. 16-20, 2021.

SINCLAIR, J. **Feedback control for exergames**. Dissertação de Curso de Doutorado, Tecnologia da Informação, Edith Cowan University, Mount Lawley, Austrália, 2011.

SOMMERVILLE, I. **Engenharia de Software**. 9.ed. São Paulo: Pearson Prentice Hall, 2011.

ŠOŠIĆ, S.; RISTIC, O.; MILOSEVIC, M. Game-Based Learning of Software Testing. In: **8th International Scientific Conference Technics and Informatics in Education**, p. 151-155, 2020.

STEGE, L.; VAN LANKVELD, G.; SPRONCK, P. Serious games in education. **International Journal of Computer Science in Sport**, v. 10, n. 1, p. 1-9, 2011.

TAROUCO, L. M. R.; ROLAND, L. C.; FABRE, M. C. J. M.; KONRATH, M. L. P. Jogos educacionais. **Revista Novas Tecnologias na Educação - RENOTE**, v.2, n.1, 2004.

VALLE, P.; BARBOSA, E. F.; MALDONADO, J. Um mapeamento sistemático sobre ensino de teste de software. In: **Anais do XXVI Simpósio Brasileiro de Informática na Educação (SBIE 2015)**, p. 71, 2015.

VALLE, P.; ROCHA, R.; MALDONADO, J. Testing game: An educational game to support software testing education. In: **Proceedings of the XXXI Brazilian Symposium on Software Engineering**. p. 289-298, 2017.

VUE.JS. **Fundamentos dos Componentes**. 2023. Disponível em: <https://pt.vuejs.org/guide/essentials/component-basics#components-basics>. Acesso em: 26 de junho de 2024.

WANGENHEIM, C. G.; SHULL, F. To game or not to game?. **IEEE software**, v. 26, n. 2, p. 92-94, 2009.

WANGENHEIM, C. G.; WANGENHEIM, A. **Ensinando Computação com Jogos.** Florianópolis: Bookess Editora, 2012.

WAZLAWICK, R. **Engenharia de Software:** Conceitos e Práticas. 1.ed. Rio de Janeiro: Elsevier Editora, 2013.

YADAV, R. S. Improvement in the V-Model. **International Journal of Scientific & Engineering Research**, v. 3, n. 2, p. 1-6, 2012.

APÊNDICE A – NÍVEIS DO JOGO

Módulo de Infraestrutura de Energia

Nível 1 - Submódulo de ativação de geradores:

Nível de Teste: Teste de Unidade
Critério(s) da técnica funcional utilizado(s): Particionamento de Equivalência

Especificação: Quando ocorre uma falha grave de energia, os geradores das cidades precisam ser ativados. Para isso, é necessário inserir uma senha. A senha esperada é "ger995". Se a senha inserida estiver correta, os geradores serão ativados e a mensagem "Geradores ativados com sucesso!" será exibida. Caso a senha esteja incorreta, os geradores não serão ativados e a mensagem "Senha incorreta!" será retornada. Se não for inserida nenhuma senha (senha em branco), os geradores também não serão ativados e a mensagem "Você deve inserir uma senha para ativar os geradores!" será exibida.

Entrada(s): Senha
Saída(s): Ativação ou não dos geradores e mensagem informativa

Classes de equivalência:
1 - senha correta (ger995) (válida)
2 - senha preenchida (válida)
3 - senha incorreta (inválida)
4 - senha em branco (inválida)

Desafio da fase de uso dos critérios da técnica funcional: Selecionar as classes de equivalência corretas, em uma lista de checkboxes para classes válidas e em outra lista de checkboxes para classes inválidas.

Casos de teste esperados na fase de elaboração de casos de teste:

CT1 - Entrada: senha = ger995 (senha correta e preenchida) | Saída esperada: geradores ativados e mensagem "Geradores ativados com sucesso!".

CT2 - Entrada: senha = a (senha incorreta) | Saída esperada: geradores não ativados e mensagem “Senha incorreta!”.

CT3 - Entrada: senha em branco | Saída esperada: geradores não ativados e mensagem “Você deve inserir uma senha para ativar os geradores!”.

Falhas na fase de execução dos casos de teste: Senha incorreta ativa os geradores.

Nível 2 - Submódulo de monitoramento de interrupções de corrente:

Nível de Teste: Teste de Unidade
Critério(s) da técnica funcional utilizado(s): Particionamento de Equivalência e Análise do Valor Limite

Especificação: Existe um sensor que monitora a quantidade de interrupções de corrente que ocorrem nas cidades a cada intervalo de 5 minutos. De acordo com a quantidade de interrupções registrada pelo sensor, o sistema retorna um código de aviso que será usado para identificar falhas de energia.

Se a quantidade de interrupções nos últimos 5 minutos for maior ou igual a 0 e menor ou igual a 3, significa que a quantidade de interrupções está em um limite aceitável, e o sistema deve retornar o código 0. Se a quantidade de interrupções for maior que 3, isso indica que a quantidade de interrupções excedeu o limite, caracterizando uma situação anormal, e o sistema deve retornar o código 1. Se o sensor retornar uma quantidade negativa (menor que zero), isso indica que o sensor está quebrado. Neste caso, deve ser exibida a mensagem “Sensor quebrado!” e nenhum código deve ser retornado.

Obs: o sensor nunca retorna valores vazios ou que não sejam números inteiros.

Entrada(s): Quantidade de interrupções de corrente nos últimos 5 minutos (I)
Saída(s): Código de aviso ou mensagem informativa

Classes de equivalência:
1 - I < 0 (inválida)
2 - I >= 0 e I <= 3 (válida)
3 - I > 3 (inválida)

Valores-limite:
1 - I = -1
2 - I = 0
3 - I = 3
4 - I = 4

Desafio da fase de uso dos critérios da técnica funcional: Completar os valores-limites em uma reta numérica, de modo a dividi-la nas classes de equivalência corretas.

Casos de teste esperados na fase de elaboração de casos de teste:

CT1 - Entrada: I = -10 | Saída esperada: mensagem “Sensor quebrado!”

CT2 - Entrada: I = -1 (análise do valor limite) | Saída esperada: mensagem “Sensor quebrado!”

CT3 - Entrada: I = 0 (análise do valor limite) | Saída esperada: código 0

CT4 - Entrada: I = 2 | Saída esperada: código 0

CT6 - Entrada: I = 3 (análise do valor limite) | Saída esperada: código 0

CT7 - Entrada: I = 4 (análise do valor limite) | Saída esperada: código 1

CT8 - Entrada: I = 15 | Saída esperada: código 1

Falhas na fase de execução dos casos de teste: I = 4 retorna código 0.

Nível 3 - Submódulo de monitoramento de tensão:

Nível de Teste: Teste de Unidade

Critério(s) da técnica funcional utilizado(s): Particionamento de Equivalência e Análise do Valor Limite

Especificação: Assim como há um sensor que monitora a quantidade de interrupções de corrente, também existe um sensor que monitora a tensão. De acordo com a tensão em volts retornada por este sensor, o sistema gera um código de aviso que será utilizado para identificar falhas de energia.

Inicialmente, o sistema verifica se o sensor está funcionando corretamente. Se a tensão retornada for menor que 0V ou maior que 10.000V, isso indica que o sensor está quebrado e deve ser exibida a mensagem “Sensor quebrado!”. Se a tensão for maior ou igual a 0V e menor ou igual a 10.000V, o sensor está funcionando corretamente.

Com o sensor funcionando corretamente, diferentes códigos são retornados conforme a tensão:

- Se a tensão for maior ou igual a 100V e menor ou igual a 240V, a tensão está em uma faixa segura e deve ser retornado o código 0.
- Se a tensão for menor que 100V, ela está abaixo do limite inferior aceitável e deve ser retornado o código 1.
- Se a tensão for maior que 240V, ela está acima do limite superior aceitável e deve ser retornado o código 2.

Obs: o sensor nunca retorna valores vazios ou que não sejam números inteiros.

Entrada(s): Tensão em Volts (T)

Saída(s): Código de aviso ou mensagem informativa

Classes de equivalência:

1 - T < 0 (inválida)

2 - T > 10.000 (inválida)

3 - T >= 0 e T <= 10.000 (válida)

4 - T >= 0 e T < 100 (inválida)

5 - T > 240 e T <= 10.000 (inválida)

6 - T >= 100 e T <= 240 (válida)

Valores-limite:

1 - T = -1

2 - T = 0

3 - T = 10.000

4 - T = 10.001

5 - T = 100

6 - T = 99

7 - T = 240

8 - T = 241

Desafio da fase de uso dos critérios da técnica funcional: Completar os valores-limites em uma reta numérica, de modo a dividi-la nas classes de equivalência corretas.

Casos de teste esperados na fase de elaboração de casos de teste:

CT1 - Entrada: T < -1 | Saída esperada: mensagem “Sensor quebrado!”

CT2 - Entrada: T = -1 (análise do valor limite) | Saída esperada: mensagem “Sensor quebrado!”

CT3 - Entrada: T = 0 (análise do valor limite) | Saída esperada: código 1

CT4 - Entrada: T = 10.001 (análise do valor limite) | Saída esperada: mensagem “Sensor quebrado!”

CT5 - Entrada: T = 10.000 (análise do valor limite) | Saída esperada: código 2

CT6 - Entrada: T > 10001 | Saída esperada: mensagem “Sensor quebrado!”

CT7 - Entrada: T > 0 e T < 99 | Saída esperada: código 1

CT8 - Entrada: T = 100 (análise do valor limite) | Saída esperada: código 0

CT9 - Entrada: T = 99 (análise do valor limite) | Saída esperada: código 1

CT10 - Entrada: T > 100 e T < 240 | Saída esperada: código 0

CT11 - Entrada: T = 240 (análise do valor limite) | Saída esperada: código 0

CT12 - Entrada: T = 241 (análise do valor limite) | Saída esperada: código 2

CT13 - Entrada: T > 241 e T < 10000 | Saída esperada: código 2

Falhas na fase de execução dos casos de teste: T = 10.001 retorna código 2 e T = 100 retorna código 1.

Nível 4 - Submódulo de detecção de falhas na energia:

Nível de Teste: Teste de integração que envolve os três submódulos testados anteriormente e o submódulo de detecção de falhas na energia.

Critério(s) da técnica funcional utilizado(s): Tabela de Decisão.

Especificação: O submódulo de detecção de falhas de energia é responsável por emitir alertas e permitir que os geradores sejam ativados, com base nos códigos retornados pelos submódulos de monitoramento de interrupções de corrente e de monitoramento de tensão. As possibilidades são as seguintes:

1. Sem falhas:
    - Se ambos os submódulos retornarem código 0, nenhum alerta deve ser emitido e a mensagem “Sem falhas na energia elétrica” deve ser exibida.
2. Falha de energia com baixa tensão:

- Se o submódulo de monitoramento de interrupções de corrente retornar código 0 ou 1 e o submódulo de monitoramento de tensão retornar código 1, deve ser gerado um alerta de falha de energia. A senha para ativar os geradores deve ser solicitada. Inserindo a senha correta ("ger995"), os geradores devem ser ativados com sucesso.

3. Alta tensão:
    - Se o submódulo de monitoramento de interrupções de corrente retornar código 0 e o submódulo de monitoramento de tensão retornar código 2, deve ser gerado um alerta de alta tensão.
4. Instabilidade de energia:
    - Se o submódulo de monitoramento de interrupções de corrente retornar código 1 e o submódulo de monitoramento de tensão retornar código 0, deve ser gerado um alerta de instabilidade de energia.
5. Instabilidade de energia e alta tensão:
    - Se o submódulo de monitoramento de interrupções de corrente retornar código 1 e o submódulo de monitoramento de tensão retornar código 2, deve ser gerado um alerta de instabilidade de energia e alta tensão.
6. Erro no submódulo:
    - Se nenhum código for retornado por algum dos submódulos, nenhum alerta deve ser emitido e deve ser exibida a mensagem “Erro no submódulo xxx”.

Entrada(s): Código de monitoramento de tensão (T) e código do monitoramento de interrupções (I)

Saída(s): Alerta ou mensagem informativa

Tabela de Decisão:

I (código)	0	0	0	1	1	1	N/A	X	N/A
T (código)	0	1	2	0	1	2	X	N/A	N/A
Resultado	A	B	C	D	B	E	F	G	H

A. Mensagem "Sem falhas na energia elétrica" exibida e nenhum alerta gerado;

B. Alerta de falha de energia (alerta de id xxxx1) gerado e solicitação da senha para a ativação dos geradores;

C. Alerta de alta tensão (alerta de id xxxx2) gerado;

D. Alerta de instabilidade de energia (alerta de id xxxx3) gerado;

E. Alerta de instabilidade de energia e alta tensão (alerta de id xxxx2) gerado;

F. Mensagem "Erro no submódulo de monitoramento de interrupções" exibida;

G. Mensagem "Erro no submódulo de monitoramento de tensão" exibida;

H. Mensagem "Erro nos submódulos de monitoramento de interrupções e de tensão" exibida.

Obs: N/A = sem código e X = qualquer valor

Desafio da fase de uso dos critérios da técnica funcional: Completar a tabela de decisão (partes do I e T), selecionando o valor em uma caixa de seleção para cada célula.

Casos de teste esperados na fase de elaboração de casos de teste:

CT1 - Entrada: código I = 0 e código T = 0 | Saída esperada: Mensagem "Sem falhas na energia elétrica" exibida e nenhum alerta gerado

CT2 - Entrada: código I = 0 e código T = 1 | Saída esperada: Alerta de falha de energia (alerta de id xxxx1) gerado e solicitação da senha para a ativação dos geradores

CT3 - Entrada: código I = 0 e código T = 2 | Saída esperada: Alerta de alta tensão (alerta de id xxxx2) gerado

CT4 - Entrada: código I = 1 e código T = 0 | Saída esperada: Alerta de instabilidade de energia (alerta de id xxxx3) gerado

CT5 - Entrada: código I = 1 e código T = 1 | Saída esperada: Alerta de falha de energia (alerta de id xxxx1) gerado e solicitação da senha para a ativação dos geradores

CT6 - Entrada: código I = 1 e código T = 2 | Saída esperada: Alerta de instabilidade de energia e alta tensão (alerta de id xxxx2) gerado

CT7 - Entrada: código I = N/A e código T = X | Saída esperada: Mensagem - "Erro no módulo de monitoramento de interrupções" exibida

CT8 - Entrada: código I = X e código T = N/A | Saída esperada: Mensagem - "Erro no módulo de monitoramento de tensão" exibida

CT9 - Entrada: código I = N/A e código T = N/A | Saída esperada: Mensagem - " Erro nos submódulos de monitoramento de interrupções e de tensão " exibida

Falhas na fase de execução dos casos de teste: No caso de I = 0 e T = 1 os valores são invertidos na integração, T vira 0 e I vira 1 fazendo com que não tenha a resposta esperada. No caso de I = 1 e T = 1 não é pedida senha para ativar geradores e os geradores não conseguem ser ativados.

Módulo de Redes de Comunicação

Nível 5 - Submódulo de envio de alertas:

Nível de teste: Teste de Unidade
Critério(s) da técnica funcional utilizado(s): Particionamento de Equivalência e Análise do Valor Limite

Especificação: O submódulo de envio de alertas é responsável por processar e enviar alertas gerados por um usuário ou por outro submódulo do sistema. Cada alerta é composto por três elementos: um identificador (ID), uma mensagem e o departamento destinatário. Para que o alerta seja enviado corretamente, algumas regras de validação devem ser seguidas.

Primeiramente, o ID deve conter exatamente cinco dígitos. Caso essa condição não seja respeitada, o sistema retornará a mensagem de erro "O ID deve conter 5 dígitos!". Além disso, o último dígito do ID deve ser 1, 2 ou 3, representando a prioridade do alerta. O número 1 indica prioridade máxima, o número 2 representa prioridade média e o número 3 corresponde à prioridade baixa. Se o último dígito for diferente desses valores, será retornada a mensagem "O último dígito do ID pode ser somente 1, 2 ou 3!".

A mensagem do alerta também deve seguir um critério específico: ela deve ter entre 10 e 70 caracteres. Se a mensagem não atender a essa exigência, o sistema exibirá o erro "A mensagem deve conter entre 10 e 70 caracteres!". Além disso, o destinatário do alerta precisa ser um dos três departamentos disponíveis: Energia Elétrica, Trânsito ou Saúde. Caso não seja selecionado um departamento, o sistema retornará a mensagem "O departamento deve ser selecionado!".

Os campos de ID e mensagem também são obrigatórios. Portanto, ao tentar enviar um alerta sem preencher um desses campos, a mensagem de erro correspondente será "O campo [xxx] deve ser preenchido!", onde [xxx] representa o nome do campo que está vazio.

Se todas as validações forem cumpridas e o alerta for enviado com sucesso, o sistema exibirá a mensagem "Alerta enviado com sucesso para o departamento [selecionado]!". No entanto, se ocorrer algum problema durante o envio, será exibida a mensagem "Falha no envio do alerta!".

Para testar o cenário de falha no envio, há um ID específico que pode ser utilizado: o ID 00000. Esse ID é valido e foi configurado apenas para simular falhas no envio. Assim, ao utilizar o ID 00000, o sistema deverá retornar a mensagem de falha no envio, permitindo a verificação desse comportamento.

Entrada(s): ID do alerta (ID), mensagem (MSG) e departamento destinatário (DEP)

Saída(s): Mensagem informativa

Classes de equivalência:

ID

- **Número de dígitos (ND - ID):**

  1 - ND - ID = 5 (válida)
  2 – ND – ID = 0 (inválida)
  3 – ND – ID != 0 (válida)
  4 - ND – ID != 5 (inválida)

- **Dígito final (DF - ID):**

  1 – DF - ID = 1 (válida)
  2 - DF - ID = 2 (válida)
  3 - DF - ID = 3 (válida)
  4 - DF - ID = qualquer outro número (inválida)

Número de caracteres mensagem (NC - MSG):

1 – NC - MSG != 0 (válida)
2 - NC - MSG > = 10 e m <= 70 (válida)
3 - NC - MSG < 10 (inválida)
4 - NC - MSG > 70 (inválida)
5 - NC - MSG = 0 (inválida)

Departamento (DEP):

1 – DEP selecionado (válida)
2 - DEP = ele (válida)
3 - DEP = sau (válida)

4 - DEP = tra (válida)

5 - DEP não selecionado (inválida)

Mensagem de retorno do envio do alerta (MSG - RET):

1 - MSG – RET = sucesso no envio (válida)

2 - MSG – RET = falha no envio (inválida)

Valores-limite:

1 - NC - MSG = 9

2 - NC - MSG = 10

3 - NC - MSG = 70

4 - NC - MSG = 71

Desafio da fase de uso dos critérios da técnica funcional: Associar às classes válidas e inválidas as suas condições de entrada/saída correspondentes, selecionar a condição de entrada/saída a qual se aplica o critério de análise do valor limite e completar uma reta numérica com os valores-limite da classe selecionada.

Casos de teste esperados na fase de elaboração de casos de teste:

CT01 – Entrada: ID = 17891, Mensagem = '1234567890' (10 caracteres) e Departamento = Trânsito | Saída esperada: Alerta enviado e mensagem 'Alerta enviado com sucesso para o departamento de trânsito!'

CT02 – Entrada: ID em branco, Mensagem = 'testando alerta' e Departamento = Trânsito | Saída esperada: Alerta não enviado e mensagem 'O campo de ID deve ser preenchido!'

CT03 - Entrada: ID = 19897, Mensagem = 'testando alerta' e Departamento = Trânsito | Saída esperada: Alerta não enviado e mensagem 'O último dígito do ID pode ser somente 1, 2 ou 3!'

CT04 - Entrada: ID = 17891, Mensagem = '1234' (4 caracteres) e Departamento = Trânsito | Saída esperada: Alerta não enviado e mensagem 'A mensagem deve conter entre 10 e 70 caracteres!'

CT05 - Entrada: ID = 00000, Mensagem = 'testando alerta' e Departamento = Saúde | Saída esperada: Falha no envio do alerta e mensagem 'Falha no envio do alerta!'

CT06 - Entrada: ID = 17892, Mensagem = 'testando alerta' e Departamento = Energia elétrica | Saída esperada: Alerta enviado e mensagem 'Alerta enviado com sucesso para o departamento de energia elétrica!'

CT07 - Entrada: ID = 17891111, Mensagem = 'testando alerta' e Departamento = Trânsito | Saída esperada: Alerta não enviado e mensagem 'O ID deve conter 5 dígitos!'

CT08 - ID = 17891, Mensagem = 'teste mensagem alerta teste mensagem alerta teste mensagem alerta teste' (71 caracteres) e Departamento = Trânsito | Saída esperada: Alerta não enviado e mensagem 'A mensagem deve conter entre 10 e 70 caracteres!'

CT09 - Entrada: ID = 17891, Mensagem = 'testando alerta' e Departamento não selecionado | Saída esperada: Alerta não enviado e mensagem 'O departamento deve ser selecionado!'

CT10 - Entrada: 17891, Mensagem = 'teste mensagem alerta teste mensagem alerta teste mensagem alerta teste mensagem' (80 caracteres) e Departamento = Trânsito | Saída esperada: Alerta não enviado e mensagem 'A mensagem deve conter entre 10 e 70 caracteres!'

CT11 - Entrada: ID = 17891, Mensagem em branco e Departamento = Trânsito | Saída esperada: Alerta não enviado e mensagem 'O campo de mensagem deve ser preenchido!'

CT12 - Entrada: ID = 17891, Mensagem = 'teste mensagem alerta tes' (25 caracteres) e Departamento = Trânsito | Saída esperada: Alerta enviado e mensagem 'Alerta enviado com sucesso para o departamento de trânsito!'

CT13 - Entrada: ID = 17891, Mensagem = 'teste mensagem alerta teste mensagem alerta teste mensagem alerta test' (70 caracteres) e Departamento = Trânsito | Saída esperada: Alerta enviado e mensagem 'Alerta enviado com sucesso para o departamento de trânsito!'

CT14 - Entrada ID = 17893, Mensagem = 'testando alerta' e Departamento = Saúde | Saída esperada: Alerta enviado e mensagem 'Alerta enviado com sucesso para o departamento de saúde!'

CT15 - Entrada: ID = 17891, Mensagem = '123456789' (9 caracteres) e Departamento = Trânsito | Saída esperada: Alerta não enviado e mensagem 'A mensagem deve conter entre 10 e 70 caracteres!'

Falhas na fase de execução dos casos de teste: É permitido enviar alerta com mensagem em branco, é permitido enviar alerta com o ID com qualquer dígito no final e quando a mensagem possui 70 caracteres, o alerta não é enviado.

Nível 6 - Integração do submódulo de detecção de falhas na energia com o submódulo de envio de alertas

Nível de Teste: Teste de Integração
Critério(s) da técnica funcional utilizado(s): Particionamento de Equivalência

Especificação: Quando o submódulo de detecção de falhas de energia emitir um alerta, o submódulo de envio de alertas deve receber corretamente o seu ID e mensagem. O destinatário para esses alertas deve ser sempre o departamento de energia elétrica e o alerta deve ser enviado automaticamente. Por outro lado, quando não houver nenhum alerta emitido, todos os campos devem permanecer em branco e não deve haver envio de alerta.

Entrada(s): ID do alerta, mensagem e departamento destinatário
Saídas(s): Mensagem informativa

Classes de equivalência:
1 - Alerta emitido (válida)
2 - Alerta não emitido (inválida)

Desafio da fase de uso dos critérios da técnica funcional: Selecionar as classes de equivalência corretas, em uma lista de checkboxes para classes válidas e em outra lista de checkboxes para classes inválidas.

Casos de teste esperados na fase de elaboração de casos de teste:

CT01 - Entrada: situação que um alerta é emitido | Saída esperada: todas as informações do alerta devem ser recebidas corretamente pelo submódulo de envio de alertas e o alerta deve ser enviado com sucesso para o departamento de energia elétrica.

CT02 - Entrada: situação que um alerta não é emitido | Saída esperada: os campos devem permanecer em branco e nenhum alerta deve ser enviado pelo submódulo de envio de alertas.

Falhas na fase de execução dos casos de teste: Quando um alerta emitido pelo submódulo de detecção de falhas de energia é integrado com o submódulo de envio de alertas, o departamento para o qual é enviado é de trânsito.

Módulo de autenticação

Nível 7 - Submódulo de autenticação:

Nível de Teste: Teste de Unidade
Critério(s) da técnica funcional utilizado(s): Particionamento de Equivalência

Especificação: Para acessar o sistema, o usuário deve realizar autenticação com seu usuário, senha e tipo de usuário. Há 3 tipos de usuário no sistema: Operador de Energia, Técnico em Comunicação e Administrador.

Se todos os dados estiverem corretos e preenchidos, o login será feito com sucesso e o usuário será redirecionado para uma tela inicial onde será exibida uma mensagem de boas-vindas, a informação de seu tipo de usuário e um menu. Um tipo de usuário deve apenas conseguir logar no sistema como seu tipo de usuário, ou seja, se inserir usuário e senha corretos mas selecionar um tipo de usuário que não seja o seu, não poderá entrar no sistema. Além disso, se algum dos dados estiver incorreto ou em branco será exibida mensagem de erro e o login não será realizado.

Alguns usuários cadastrados e suas respectivas senhas e tipos de usuário para teste:

- usuário: def345, senha: 7809, tipo: operador de energia;
- usuário: aasbn5, senha: pol$444, tipo: técnico em comunicação;
- usuário: 234xxx, senha: oo344, tipo: administrador.

Entrada(s): usuário, senha e tipo de usuário
Saída(s): Login realizado como determinado tipo de usuário ou mensagem de erro

Classes de equivalência:
Usuário
1 - Usuário válido (válida)
2 – Usuário preenchido (válida)
3 - Usuário inválido (inválida)
4 - Usuário em branco (inválida)

Senha
1 - Senha válida (válida)
2 – Senha preenchida (válida)
3 - Senha inválida (inválida)
4 - Senha em branco (inválida)

Tipo de usuário:
1 – Selecionado (válida)
2 - Operador de energia (válida)

3 - Técnico em comunicações (válida)

4 - Administrador (válida)

5 - Não selecionado (inválida)

Correspondência do tipo de usuário com usuário e senha

1 - Usuário e senha corretos mas tipo de usuário corresponde (válida)

2 - Usuário e senha corretos mas tipo de usuário não corresponde (inválida)

Desafio da fase de uso dos critérios da técnica funcional: Associar às classes válidas e inválidas as suas condições de entrada/saída correspondentes.

Casos de teste esperados na fase de elaboração de casos de teste:

CT01 - Entrada: usuário: def345, senha: 7809, tipo: operador de energia | Saída esperada: usuário logado com sucesso com o tipo operador de energia

CT02 - Entrada: usuário incorreto, senha: 7809, tipo: operador de energia | Saída esperada: mensagem “Usuário inválido!”

CT03 - Entrada: usuário em branco, senha:7809, tipo: operador de energia | Saída esperada: mensagem “Preencha o campo de usuário!”

CT04 - Entrada: usuário: def345, senha incorreta, tipo: operador de energia | Saída esperada: mensagem “Senha inválida!”

CT05 - Entrada: usuário: def345, senha em branco, tipo: operador de energia | Saída esperada: mensagem “Preencha o campo de senha!”

CT06 - Entrada: usuário: def345, senha: 7809, tipo: sem estar selecionado | Saída esperada: mensagem “Escolha um tipo de usuário”

CT07 - Entrada: usuário: def345, senha: 7809, tipo: administrador (usuário e senha corretos mas com tipo de usuário incorreto) | Saída esperada: mensagem “O tipo de usuário não corresponde ao usuário e senha inseridos!”

CT08 - Entrada: usuário: aasbn5, senha: pol$444, tipo: técnico em comunicação | Saída esperada: usuário logado com sucesso com o tipo técnico em comunicação

CT09 - Entrada: usuário: 234xxx, senha: oo344, tipo: administrador | Saída esperada: usuário logado com sucesso com o tipo administrador

Falhas na fase de execução dos casos de teste: Usuário do tipo técnico em comunicação é logado como administrador e quando o usuário e a senha estão corretos, mas o tipo de usuário não corresponde, o login é realizado.

Sistema como um todo

Nível 8 - Uso habitual do sistema (módulos afetados) por tipo de usuário e tempo de serviço:

Nível de Teste: Teste de Sistema
Critério(s) da técnica funcional utilizado(s): Tabela de Decisão

Especificação: Como já mencionado anteriormente, o sistema possui três perfis de usuários: Operador de Energia, Técnico em Comunicação e Administrador, cada um com acessos e permissões específicos.

Operadores de energia têm acesso apenas aos submódulos de infraestrutura de energia (Submódulo de ativação de geradores, Submódulo de monitoramento de interrupções de corrente, Submódulo de monitoramento de tensão e Submódulo de detecção de falhas na energia) para monitorar interrupções, tensões e falhas, tendo a permissão de ativar geradores somente quando completam um ano de serviço. No cotidiano, operadores de energia podem monitorar interrupções, tensões e falhas, mas somente aqueles com tempo de serviço maior ou igual a um ano ativam os geradores conforme necessário, quando há falha de energia.

Técnicos em comunicação têm acesso apenas aos submódulos da infraestrutura de redes de comunicação (Submódulo de envio de alertas) e têm permissão para enviar alertas somente quando completam um ano de serviço. No seu dia a dia, técnicos em comunicação com tempo de serviço maior ou igual a um ano rotineiramente enviam alertas.

Administradores possuem acesso irrestrito (acessam todos os submódulos), mas apenas os que possuem tempo de serviço maior ou igual a um ano podem ativar geradores e enviar alertas. Administradores com menos de um ano de serviço podem, em suas atividades diárias, acessar todos os submódulos. Administradores com mais (ou igual) de um ano de serviço têm plenas funcionalidades no sistema, podendo inclusive, em suas rotinas, ativar geradores conforme necessário e enviar alertas gerados.

Logins para o teste:

- Operador de energia com menos de 1 ano de serviço - usuário:abc123; senha: 9989

- Operador de energia com tempo de serviço maior ou igual a 1 ano - usuário:opkvb; senha: gh56

- Técnico em comunicação com menos de 1 ano de serviço - usuário:lpo90; senha: sdfg

- Técnico em comunicação com tempo de serviço maior ou igual a 1 ano - usuário:k53gh; senha: 0098

- Administrador com menos de 1 ano de serviço - usuário:ffds3; senha: 124j

- Administrador com tempo de serviço maior ou igual a 1 ano - usuário:vndfd; senha: tp67

Entrada(s): usuário, senha e tipo de usuário
Saída(s): Login realizado com as permissões condizentes com o tipo de usuário e tempo de serviço.

Tabela de decisão:

Cargo	Op. De energia	Op. De energia	Tec. em comunicação	Tec. em comunicação	Administrador	Administrador
>= ou < que 1 ano de serviço	< 1 ano	>= 1 ano	< 1 ano	>= 1 ano	< 1 ano	>= 1 ano
Permissão	A	B	C	D	E	F

A. Possui acesso aos submódulos da infraestrutura de energia mas não possui permissão para ativar geradores

B. Possui acesso aos submódulos da infraestrutura de energia e possui permissão para ativar geradores

C. Possui acesso aos submódulos da infraestrutura de rede de comunicação mas não possui permissão para o envio de alertas

D. Possui acesso aos submódulos da infraestrutura de energia e possui permissão para o envio de alertas

E. Possui acesso a todos os submódulos do sistema mas não possui permissão para ativar geradores nem para envio de alertas

F. Possui acesso a todos os submódulos do sistema e possui permissão para ativar geradores e enviar alertas.

Desafio da fase de uso dos critérios da técnica funcional: Completar a tabela de decisão selecionando os valores corretos por meio de caixas de seleção em cada célula (cargo, tempo de serviço e permissão).

Casos de teste esperados na fase de elaboração de casos de teste:

CT01 - Entrada: Login como op. Energia < 1 ano de serviço | Saída esperada: pode acessar somente submódulos de infraestrutura de energia, tendo permissão para

monitorar interrupções, tensões e falhas de energia (ver alertas emitidos). Este usuário não pode conseguir ativar geradores.

CT02 - Entrada: Login como op. Energia >= 1 ano de serviço | Saída esperada: pode acessar somente submódulos de infraestrutura de energia, tendo permissão para monitorar interrupções, tensões e falhas de energia (ver alertas emitidos) e ativar geradores.

CT03 - Entrada: Login como tec. em comunicação < 1 ano de serviço | Saída esperada: pode acessar somente submódulos da infraestrutura de redes de comunicação e não pode conseguir enviar alertas.

CT04 - Entrada: Login como tec. em comunicação >= 1 ano de serviço | Saída esperada: pode acessar somente submódulos da infraestrutura de redes de comunicação, tendo permissão para enviar alertas.

CT05 - Entrada: Login como adm < 1 ano de serviço | Saída esperada: pode acessar todos os submódulos do sistema, incluindo verificar a emissão de alerta no módulo de detecção de falhas de energia e a integração do alerta com o módulo de envio de alertas, mas não pode ativar geradores nem enviar alertas.

CT06 - Entrada: Login como adm >= 1 ano de serviço | Saída esperada: pode acessar todos os submódulos do sistema, incluindo verificar emissão de alerta no módulo de detecção de falhas de energia, ativar geradores, verificar a integração do alerta com o módulo de envio de alertas e enviar o alerta.

Falhas na fase de execução dos casos de teste: Administradores com menos de 1 ano de serviço podem ativar geradores.

APÊNDICE B – FORMULÁRIO DISPONIBILIZADO AOS ESTUDANTES DE CURSOS DA ÁREA DA COMPUTAÇÃO

Bem-vindo(a) ao questionário de avaliação do jogo educacional *CodeGuardians*!
O questionário é composto por duas partes:

- Parte 1**:** Esta seção contém perguntas sobre o perfil do jogador e o conhecimento prévio em testes de software. É importante que seja respondida antes de iniciar o jogo.
- Parte 2**:** Contém perguntas baseadas no *Modelo MEEGA*, um modelo de avaliação para jogos educacionais na área de Engenharia de Software. Estas perguntas cobrem dimensões como motivação, experiência do usuário e aprendizagem. Além disso, inclui questões adicionais para avaliar o quão bem o jogo atingiu seus objetivos. A Parte 2 deve ser respondida apenas após a conclusão do jogo.

Parte 1 (deve ser respondida antes de iniciar o jogo)

1) Qual o nome da universidade em que você estuda?

2) Qual curso você faz?

3) Atualmente, você está em qual semestre do curso?

4) Qual sua idade?

5) Qual seu gênero?
Masculino
Feminino
Prefiro não dizer

6) Como você classifica seu nível de conhecimento sobre testes de software (em geral)?
Nenhum conhecimento

Conhecimento básico
Conhecimento intermediário
Conhecimento avançado
Especialista

7) Como você classifica seu nível de conhecimento sobre a técnica funcional (teste caixa-preta) de testes de software?
Nenhum conhecimento
Conhecimento básico
Conhecimento intermediário
Conhecimento avançado
Especialista

8) Como você classifica seu nível de conhecimento sobre os níveis de testes de software (teste de unidade, integração, etc.)?
Nenhum conhecimento
Conhecimento básico
Conhecimento intermediário
Conhecimento avançado
Especialista

9) Como você classifica seu nível de interesse pela área de testes de software?
Nenhum interesse
Pouco interesse
Interesse moderado
Muito interesse
Totalmente interessado

Parte 2 (deve ser respondida após a conclusão do jogo)

Motivação

10) O design do jogo é atraente.
Concordo totalmente

Concordo
Neutro
Discordo
Discordo totalmente

11) A variação (forma, conteúdo ou de atividades) ajudou a me manter atento ao jogo.
Concordo totalmente
Concordo
Neutro
Discordo
Discordo totalmente

12) O conteúdo do jogo está conectado com outros conhecimentos que eu já possuía.
Concordo totalmente
Concordo
Neutro
Discordo
Discordo totalmente

13) Ao passar pelas etapas do jogo senti confiança de que estava aprendendo.
Concordo totalmente
Concordo
Neutro
Discordo
Discordo totalmente

14) Estou satisfeito porque sei que terei oportunidades de utilizar na prática coisas que aprendi com o jogo.
Concordo totalmente
Concordo
Neutro
Discordo
Discordo totalmente

15) É por causa do meu esforço pessoal que consigo avançar no jogo.

Concordo totalmente
Concordo
Neutro
Discordo
Discordo totalmente

Experiência do usuário

16) Temporariamente esqueci as minhas preocupações do dia-a-dia, fiquei totalmente concentrado no jogo.

Concordo totalmente
Concordo
Neutro
Discordo
Discordo totalmente

17) Me senti mais no ambiente do jogo do que no mundo real, esquecendo do que estava ao meu redor.

Concordo totalmente
Concordo
Neutro
Discordo
Discordo totalmente

18) Este jogo é adequadamente desafiador para mim, as tarefas não são muito fáceis nem muito difíceis.

Concordo totalmente
Concordo
Neutro
Discordo
Discordo totalmente

19) O jogo evolui num ritmo adequado e não fica monótono – oferece novos obstáculos, situações ou variações de atividades.

Concordo totalmente
Concordo
Neutro
Discordo
Discordo totalmente

20) Me diverti com o jogo.

Concordo totalmente
Concordo
Neutro
Discordo
Discordo totalmente

21) Eu recomendaria este jogo para meus colegas.

Concordo totalmente
Concordo
Neutro
Discordo
Discordo totalmente

22) Consegui atingir os objetivos do jogo por meio das minhas habilidades.

Concordo totalmente
Concordo
Neutro
Discordo
Discordo totalmente

23) Tive sentimentos positivos de eficiência no desenrolar do jogo.

Concordo totalmente
Concordo
Neutro
Discordo

Discordo totalmente

Aprendizagem

24) O jogo educacional me ajudou a compreender os princípios e critérios da técnica funcional de testes e os diferentes níveis de teste.

Concordo totalmente
Concordo
Neutro
Discordo
Discordo totalmente

25) Após jogar o jogo, sei identificar as situações adequadas, conforme especificação, para a aplicação de cada um dos critérios da técnica funcional de testes.

Concordo totalmente
Concordo
Neutro
Discordo
Discordo totalmente

26) Quanto eficiente o jogo foi para sua aprendizagem, comparando-o com outras atividades da disciplina de Engenharia de Software?

Muito mais eficiente
Mais eficiente
Tão eficiente quanto
Menos eficiente
Muito menos eficiente

27) O quanto você acha que a experiência com o jogo vai contribuir para seu desempenho na vida profissional?

Vai contribuir significativamente
Vai contribuir bastante
Vai contribuir moderadamente
Vai contribuir pouco

Não vai contribuir

Perguntas adicionais

28) Após jogar o jogo, como você classifica seu nível de conhecimento sobre testes de software (em geral)?

Nenhum conhecimento
Conhecimento básico
Conhecimento intermediário
Conhecimento avançado
Especialista

29) Após jogar o jogo, como você classifica seu nível de conhecimento sobre a técnica funcional (teste caixa-preta) de testes de software?

Nenhum conhecimento
Conhecimento básico
Conhecimento intermediário
Conhecimento avançado
Especialista

30) Após jogar o jogo, como você classifica seu nível de conhecimento sobre os níveis de testes de software (teste de unidade, integração, etc.)?

Nenhum conhecimento
Conhecimento básico
Conhecimento intermediário
Conhecimento avançado
Especialista

31) Após jogar o jogo, como você classifica seu nível de interesse pela área de testes de software?

Nenhum interesse
Pouco interesse
Interesse moderado
Muito interesse

Totalmente interessado

32) Após jogar o jogo, o quanto você se sente mais motivado a aprofundar seus conhecimentos na área de testes de software?

Nada motivado

Pouco motivado

Moderadamente motivado

Muito motivado

Totalmente motivado

33) Você possui alguma(s) sugestão(ões) de melhoria(s) para o jogo?