INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE MINAS GERAIS – *CAMPUS* BAMBUÍ
BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

Gabriela Dâmaso Rezende

TESTHUNTER*: UM JOGO DE CARTAS CRIADO PARA O ENSINO DE TESTE DE *SOFTWARE

Bambuí - MG
2024

GABRIELA DÂMASO REZENDE

TESTHUNTER: UM JOGO DE CARTAS CRIADO PARA O ENSINO DE TESTE DE *SOFTWARE*

Trabalho de conclusão de curso apresentado ao Curso de Bacharelado em Engenharia de Computação do Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais – *Campus* Bambuí como requisito parcial para obtenção do grau de Bacharel em Engenharia de Computação.

Orientador: Prof. Me. Cláudio Ribeiro de Sousa

Bambuí - MG
2024

Catalogação na Fonte Biblioteca IFMG - Campus Bambuí

R467t Rezende, Gabriela Dâmaso.
Testhunter: um jogo de cartas criado para o ensino de teste de software. / Gabriela Dâmaso Rezende. – 2024.
48 f. : il. ; color.

Orientador: Me. Cláudio Ribeiro de Sousa.
Trabalho de Conclusão de Curso (graduação) - Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais – Campus Bambuí, MG, Curso Bacharelado em Engenharia de Computação, 2024.

1. Teste de software. 2. Gamificação. 3. Desenvolvimento de jogos educativos. I. Sousa, Cláudio Ribeiro de. II. Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais – Campus Bambuí, MG. III. Título.

CDD 004.9

Elaborada por Douglas Bernardes de Castro- CRB-6/2802

Gabriela Dâmaso Rezende

TESTHUNTER: UM JOGO DE CARTAS CRIADO PARA O ENSINO DE TESTE DE *SOFTWARE*

Trabalho de conclusão de curso apresentado ao Curso de Bacharelado em Engenharia de Computação do Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais – *Campus* Bambuí como requisito parcial para obtenção do grau de Bacharel em Engenharia de Computação.

Aprovado em 28 de Novembro de 2024 pela banca examinadora:

Prof. Me. Cláudio Ribeiro de Sousa – IFMG – *Campus* Bambuí – (Orientador)
Prof. Dr. Ciniro Aparecido Leite Nametala – IFMG - *Campus* Bambuí
Prof. Dr. Marcos Roberto Ribeirol – IFMG - *Campus* Bambuí

Documento assinado eletronicamente por **Claudio Ribeiro de Sousa**, **Professor EBTT**, em 28/11/2024, às 16:50, conforme Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Ciniro Aparecido Leite Nametala**, **Professor**, em 28/11/2024, às 16:51, conforme Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Marcos Roberto Ribeiro**, **Professor**, em 28/11/2024, às 16:51, conforme Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade do documento pode ser conferida no site https://sei.ifmg.edu.br/consultadocs informando o código verificador **2112837** e o código CRC **2171AA0F**.

Bambuí - MG
2024

Para a minha família.

AGRADECIMENTOS

Agradeço primeiramente aos meus pais, José Antônio e Patrícia, que sempre desejaram o melhor para mim, me dando todo apoio e base. Com muito amor e zelo, moldaram a pessoa que sou hoje. Mesmo com todas as dificuldades que a vida traz e com as reivindicações que às vezes precisam ser feitas, eles sempre me motivaram a seguir meus sonhos, a estudar e a me tornar uma pessoa cada dia melhor.

Também gostaria de agradecer ao Higor, meu melhor amigo e companheiro, que, durante toda a jornada, não saiu do meu lado, estando sempre por perto para me apoiar e amparar. Eu agradeço profundamente a vocês; muito obrigada por acreditarem em mim, mesmo quando nem eu acreditei.

Além disso, gostaria de agradecer a todos os meus professores, que me ajudaram a chegar até aqui e que, com muita cautela, me guiaram pelo caminho. Em especial, eu gostaria de agradecer ao Professor Cláudio, meu orientador, por ter me guiado nessa jornada de escrita e aprendizado de maneira tão leve e didática, sendo sempre um exemplo de profissional que um dia eu também espero ser.

Por fim, gostaria de agradecer a mim, que, com muito suor e lágrimas, consegui chegar até o fim de uma jornada muito longa, porém proveitosa, sem perder a essência. A todos os citados e aos não citados aqui, mas que sempre estiveram presentes em minha jornada, deixo aqui o meu muito obrigada.

"Às vezes, são as pessoas das quais ninguém espera nada que fazem as coisas que ninguém consegue imaginar."

(Alan Turing)

RESUMO

O teste de *software* é uma prática essencial na engenharia de *software*, utilizado para identificar defeitos e garantir a qualidade dos produtos antes de seu lançamento. Este trabalho explora a importância do ensino de teste de *software*, destacando como ele capacita futuros profissionais com habilidades técnicas e promove uma mentalidade crítica para a detecção precoce de falhas. A falta de atividades práticas envolvendo teste de *software* na graduação representa um desafio para o ensino da disciplina e evidencia que as formações tradicionais, frequentemente, não priorizam a prática na elaboração e execução de casos de teste. Para enfrentar essa lacuna, este trabalho propõe o desenvolvimento de um jogo de cartas em formato virtual, utilizando conceitos de gamificação para revisar os principais conceitos de teste de *software*. O objetivo é criar uma ferramenta de aprendizado prática e dinâmica que auxilie os alunos na absorção do conteúdo, complementando as aulas dos professores e contribuindo para a formação de profissionais mais preparados para os desafios do mercado de trabalho. Este trabalho pode ser aplicado em cursos de graduação, como engenharia de *software*, ciência da computação, sistemas de informação, análise e desenvolvimento de sistemas e engenharia de computação, visto que todos esses cursos abordam tópicos relacionados à qualidade e ao desenvolvimento de *software*.

Palavras-chave: Teste de *software*. Gamificação. Desenvolvimento de jogos educativos. Aprendizado prático.

ABSTRACT

Software testing is an essential practice in software engineering, used to identify defects and ensure product quality before release. This work explores the importance of teaching software testing, highlighting how it equips future professionals with technical skills and fosters a critical mindset for early fault detection. The lack of practical activities involving software testing in undergraduate courses poses a challenge to the discipline's education, showing that traditional programs often do not prioritize hands-on experience in designing and executing test cases. To address this gap, this work proposes the development of a virtual card game using gamification concepts to review key software testing principles. The goal is to create a practical and dynamic learning tool that helps students absorb the content, complementing teachers' lectures and contributing to the training of professionals better prepared for the challenges of the job market. This work can be applied to undergraduate programs such as software engineering, computer science, information systems, systems analysis and development, and computer engineering, as these fields encompass topics related to software quality and development.

Keywords: Software testing. Gamification. Educational game development. Practical learning.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

ACM - *Association for Computing Machinery*

CEEinf - Diretrizes Curriculares de Cursos da Área de Computação e Informática

IEEE - *Institute of Electrical and Electronics Engineers*

IFMG - Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais

SBC - Sociedade Brasileira de Computação

SUMÁRIO

1 INTRODUÇÃO

Na engenharia de *software*, uma prática essencial é o teste. Esta etapa é fundamental para identificar defeitos e garantir a qualidade dos produtos, assegurando que eles atendam a um padrão específico antes de serem lançados no mercado ou entregues ao usuário final. Testar um *software* é executar um programa à procura de erros (MYERS, 2006). A relevância do teste é reconhecida na comunidade de desenvolvimento devido aos riscos substanciais relacionados a falhas nos sistemas.

O ensino de teste de *software* não apenas capacita os futuros profissionais com as habilidades técnicas necessárias para identificar defeitos e garantir a qualidade do *software*, mas também cultiva uma mentalidade crítica e proativa em relação à detecção de falhas. Ao integrar o ensino de testes nos programas de educação, as instituições de ensino preparam os estudantes para enfrentar os desafios reais do desenvolvimento de *software*, capacitando-os a contribuir significativamente para a mitigação dos riscos associados a falhas em sistemas de *software*, conforme destacado por Myers (2006).

Segundo Valle, Barbosa e Maldonado (2015), em seu trabalho que elaborou um mapeamento sistemático sobre o ensino de teste de *software*, "o ensino de teste de *software* é pouco explorado no contexto de cursos de graduação, sendo pouco abordado na maioria dos cursos de computação e apenas ensinado no final da graduação". Nesse sentido, o presente trabalho propõe a criação de um jogo de cartas em formato virtual que visa relembrar os principais conceitos e procedimentos a serem adotados na realização de um teste. Ao utilizar os conceitos de gamificação, espera-se proporcionar uma ferramenta de aprendizado que seja prática e dinâmica, que possibilite o auxílio na absorção do conteúdo por parte dos alunos e complemente as aulas do professor.

1.1 Objetivos

Nesta seção, são apresentados o objetivo geral e os específicos, norteadores para a realização do trabalho.

1.1.1 Objetivo geral

O presente trabalho visa proporcionar uma experiência de aprendizado, com aplicação de gamificação, sobre o ensino de conceitos de teste de *software* com a criação de um jogo de cartas virtual.

1.1.2 Objetivos específicos

Para se alcançar o objetivo geral, foram necessários os seguintes objetivos específicos:

a) fazer uma revisão bibliográfica sobre o tema, buscando trabalhos com foco na gamificação e no ensino de conceitos de teste de *software*;

b) selecionar os principais temas a serem abordados no jogo a ser desenvolvido;

c) desenvolver o jogo de cartas em formato digital por meio de uma dinâmica inovadora, com o objetivo de aprimorar o processo de ensino-aprendizagem dos temas selecionados, documentando todo o processo de desenvolvimento deste jogo.

1.2 Resultados esperados

Com o desenvolvimento da pesquisa proposta, foi elaborado um trabalho que contribui para a aplicação de uma nova ferramenta pedagógica que, ao adotar conceitos de gamificação, estimule alunos de graduação na área de computação a aprender e praticar conhecimentos sobre teste de *software* de maneira engajada e eficiente. Além disso, a pesquisa pode influenciar positivamente o ensino de testes de *software* ao oferecer uma abordagem lúdica e interativa, potencializando a fixação de conceitos cruciais e promovendo o interesse pela área. Por fim, o jogo desenvolvido pode se tornar uma referência prática para professores e estudantes, integrando-se ao processo educacional de forma significativa e inovadora.

1.3 Justificativa

A pesquisa documental realizada por Benitti e Albano (2012) relata como são ensinadas as técnicas de teste de *software* tendo como base os currículos CEEinf, 1999; SBC, 2003; SBC, 2005; IEEE, 2004; ACM; IEEE-CS, 2006. Benitti e Albano (2012) apontaram que a ausência de atividades práticas representa um desafio no ensino de teste de *software* nos cursos de graduação. Além disso, foi explorada a hipótese de que as capacitações tradicionais, frequentemente, não priorizam a prática na elaboração de casos de testes em diversos contextos, nem a execução desses casos, ambas essenciais para que os profissionais aprendam, de fato, a aplicar esses conceitos em situações reais.

Considerando a metodologia adotada em salas de aula, que tende a se concentrar em exposições por parte do professor, a estratégia de promover um ensino prático, no qual o aluno assume o papel central em seu processo de aprendizagem, surge como uma abordagem benéfica para aprimorar a qualidade do ensino. Os jogos educativos com propósitos pedagógicos demonstram sua relevância ao fomentar situações de ensino-aprendizagem, contribuindo para a construção do conhecimento (FIALHO, 2008).

Nesse sentido, a proposta de desenvolver um jogo de cartas digital educativo visa transformar a experiência de aprendizagem do aluno em algo acessível e envolvente. Ao incorporar elementos lúdicos e interativos, o jogo busca estimular o interesse e a

participação ativa do aluno, proporcionando-lhe uma experiência de aprendizagem dinâmica e significativa.

Ao assumir a liderança em sua própria jornada de aprendizagem dentro do contexto do jogo, o aluno se torna o protagonista de sua experiência educacional. Isso significa que ele não apenas absorve passivamente informações, mas também toma decisões, resolve problemas e enfrenta desafios que são relevantes para o seu processo de aprendizagem. Os jogos, representando uma ferramenta pedagógica crucial, promovem aspectos como socialização, atenção e concentração, o que resulta em benefícios significativos para o desenvolvimento e a aprendizagem, ao mesmo tempo em que atraem os alunos para o processo de aprendizado dos conteúdos didáticos de maneira diferenciada e lúdica (BATISTA; DIAS, 2012).

A sensação de controle e autonomia sobre o próprio aprendizado estimula a curiosidade e o desejo de buscar conhecimento de forma mais profunda e significativa. Dessa forma, o jogo se torna uma ferramenta poderosa para incentivar a aprendizagem ativa e o desenvolvimento de habilidades cognitivas, sociais e emocionais essenciais para o sucesso acadêmico e pessoal do aluno. De acordo com Possatto e Jagnow (2022), pesquisas bibliográficas de autores renomados, juntamente com a prática em sala de aula, indicam que os jogos didáticos programados são uma ferramenta poderosa para o desenvolvimento de habilidades e fixação dos conteúdos de maneira agradável.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, apresentam-se as principais referências e conceitos importantes utilizados para o desenvolvimento de presente trabalho.

2.1 Fundamentos conceituais

Neste tópico, são apresentados os principais conceitos que fundamentam o desenvolvimento deste trabalho, fornecendo a base teórica necessária para sua compreensão e contextualização.

2.1.1 Qualidade de software

No contexto geral, qualidade pode ser entendida como a conformidade aos requisitos estabelecidos, ou seja, a capacidade de um produto ou serviço de atender às expectativas e necessidades definidas previamente (KOSCIANSKI; SANTOS SOARES, 2007). A qualidade é medida pela diferença entre as características observadas e as características especificadas para a construção do produto. Em outras palavras, quanto menor for essa diferença, maior será a qualidade percebida do produto.

Tratando-se de *software*, a qualidade é um reflexo direto de sua conformidade aos requisitos definidos para ele. Esses requisitos são especificados por um especialista, o que implica que a qualidade depende das escolhas feitas durante a fase de definição dos requisitos. Portanto, a qualidade de um *software* não pode ser tratada de forma dogmática ou simplista (KOSCIANSKI; SANTOS SOARES, 2007). Em vez disso, exige uma abordagem técnica e criteriosa, que considere diversos fatores e que utilize métodos e ferramentas adequadas de engenharia de *software* para alcançar os objetivos de qualidade. Satisfazer o cliente é o propósito final, e isso só é possível por meio de uma compreensão clara e precisa do que significa qualidade para aquele contexto específico.

2.1.2 Teste de software

O teste de *software* é um processo fundamental no desenvolvimento de sistemas, cujo objetivo principal é identificar e corrigir defeitos que possam afetar a funcionalidade, o desempenho e a segurança do produto final. Essa atividade não se limita apenas a verificar se as especificações ou requisitos do negócio foram implementados corretamente, mas também a garantir a qualidade geral do *software*, minimizando riscos tanto para o negócio quanto para a imagem da empresa (BASTOS *et al.*, 2007).

O projeto de teste de *software* deve começar paralelamente ao projeto de desenvolvimento. Isso significa que, desde o início, devem ser considerados os aspectos de teste, permitindo a identificação precoce de defeitos. A correção de defeitos encontrados nos requisitos ou modelos custa menos do que a correção de defeitos identificados em

estágios mais avançados do desenvolvimento. Isso está alinhado com a "regra 10"de Myers (2006), que estabelece que o custo de correção de defeitos tende a aumentar quanto mais tarde o defeito é detectado.

Portanto, faz-se necessário que o teste de *software* seja realizado de maneira organizada e sistemática, seguindo uma metodologia adequada. Isso inclui o uso de técnicas e ferramentas apropriadas, ambientes de teste configurados corretamente e profissionais qualificados para conduzir os testes. Dessa forma, pode-se realizar uma boa execução dos testes, garantindo que as especificações do negócio sejam atendidas e contribuindo para a qualidade geral do *software*, reduzindo riscos e custos de manutenção.

2.1.2.1 Testes funcionais

Os testes funcionais são uma abordagem de teste de *software* que avalia as funcionalidades de um sistema de acordo com os requisitos especificados. Esse tipo de teste foca em verificar se a aplicação realiza corretamente as operações esperadas, independentemente de sua implementação interna. Myers (2006) destaca que os testes funcionais são feitos a partir da perspectiva do usuário final e têm como objetivo validar se todas as funções do *software* operam conforme o esperado, garantindo a qualidade e a conformidade do sistema com os requisitos.

2.1.2.2 Testes estruturais

Os testes estruturais, também conhecidos como testes de caixa branca, examinam a estrutura interna do *software*, avaliando a lógica e o fluxo de controle dos códigos e algoritmos implementados. Esse tipo de teste permite que o testador tenha acesso ao código-fonte e é realizado para garantir que todas as partes do *software* sejam acessadas e testadas, incluindo caminhos, condições e fluxos. De acordo com Myers (2006), os testes estruturais são fundamentais para identificar falhas internas e assegurar que o comportamento do *software* esteja em conformidade com sua estrutura e *design*.

2.1.2.3 Teste caixa-preta

O teste de caixa-preta, um tipo de teste funcional, é uma técnica de teste de *software* que avalia o funcionamento de um programa sem considerar sua estrutura interna. Em vez disso, o teste se concentra na entrada e na saída do *software*, tratando-o como uma caixa-preta na qual o testador não precisa conhecer sua implementação interna para realizar os testes (DIAS NETO, 2007). Dados de entrada são fornecidos, o teste é executado e o resultado obtido é comparado a um resultado esperado previamente conhecido. Haverá sucesso no teste se o resultado obtido for igual ao resultado esperado.

O elemento de *software* a ser examinado pode abranger um método, uma função interna, um programa, um componente, uma coleção de programas e/ou componentes,

ou, ainda, uma funcionalidade. A abordagem de teste funcional é aplicável em todos os estágios de testes (PRESSMAN, 2005).

2.1.2.4 Teste caixa branca

O teste de caixa branca, um tipo de teste estrutural, é uma técnica de teste de *software* que avalia a estrutura interna de um programa, incluindo sua lógica, estrutura de dados e caminhos de execução (DIAS NETO, 2007). Ao contrário do teste de caixa-preta, que não considera sua implementação interna, o teste de caixa branca examina o código-fonte do *software* para criar e executar os testes.

Este método opera diretamente no código-fonte do componente de *software* para avaliar aspectos como: teste de condição, teste de fluxo de dados, teste de ciclos e teste de caminhos lógicos (PRESSMAN, 2005). O testador tem acesso ao código-fonte da aplicação e pode construir códigos para efetuar a ligação de bibliotecas e componentes. Esse tipo de teste é criado por meio da análise do código-fonte e da elaboração de casos de teste que abranjam todas as possibilidades do componente de *software* (DIAS NETO, 2007). Dessa maneira, todas as variações originadas por estruturas de condições são testadas.

2.1.3 Gamificação

A gamificação incorpora elementos estéticos, estruturais e padrões de pensamento encontrados nos jogos. Tem o objetivo de motivar ações, promover aprendizado e resolver problemas, ao utilizar estratégias que tornam os jogos envolventes (MURR; FERRARI, 2020).

Quando aplicada no ensino do teste de *software*, ela desempenha um papel relevante, tornando o aprendizado mais envolvente e eficaz para os alunos. A utilização da gamificação é aconselhável em contextos educacionais, visando aprimorar o desempenho dos alunos, a concentração e a sensação de realização (LI *et al.*, 2023). Essa abordagem não apenas aumenta a motivação e o engajamento, mas também promove o aprendizado ativo, onde os alunos estão constantemente aplicando conceitos em situações práticas. Pesquisas recentes indicam aumento na utilização de abordagens gamificadas no ensino de teste de *software* para enfrentar questões ligadas à falta de motivação dos estudantes (NASCIMENTO, 2019).

Ao incorporar conceitos-base da gamificação, como pontos, níveis, rankings, missões, conquistas, personalização, feedback, regras e narrativa (MURR; FERRARI, 2020), é possível oferecer ao aluno uma compreensão mais clara e objetiva de seus pontos fortes e áreas a serem melhoradas.

Além disso, a gamificação se manifesta por meio de jogos educacionais sérios e simulações, proporcionando aos aprendizes a oportunidade de praticar habilidades e

adquirir conhecimento em um ambiente protegido (SZETO *et al.*, 2021). Ao contextualizar os conceitos em uma narrativa envolvente e promover a colaboração entre os alunos, a gamificação contribui para tornar o aprendizado de testes de *software* mais significativo e prepara os alunos para enfrentar os desafios da prática profissional.

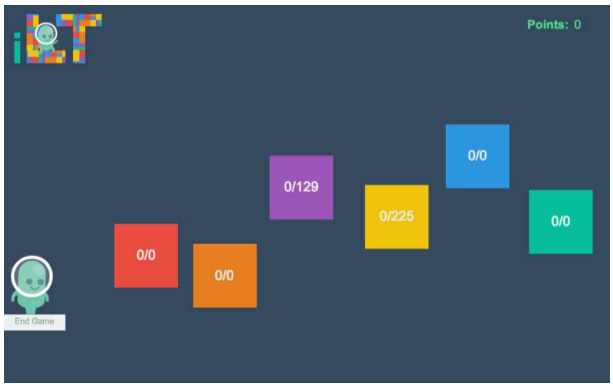
2.2 Revisão bibliográfica

Alguns trabalhos exploram o uso de jogos como ferramenta de ensino, por exemplo, Ribeiro (2014) em sua dissertação de mestrado. Nesse trabalho, foi explorada a importância dos jogos na educação, desenvolvendo um jogo específico para o ensino de teste de *software*. Este jogo visa complementar o ensino de modo a ajudar a aumentar o interesse do estudante e a preparar para a realização do exame de certificação ISTQB (*International Software Testing Qualification Board*), que é o primeiro nível do programa de certificação internacional em testes de *software*.

Esse trabalho consistiu em um jogo hospedado numa página *web* com a característica de ser multidinâmico. Alguns desafios do jogo são: adivinhar os conceitos, *drag-and-drop* para duas caixas, apanhar os conceitos corretos e descobrir os caminhos. A validação desse trabalho se deu ao submeter 3 grupos de alunos a um exame e comparar os resultados entre o grupo que não utilizou o jogo e os grupos que o utilizaram. Concluiu-se que o jogo alcançou o objetivo de elevar a motivação dos alunos na aprendizagem de testes de *software*.

Em suma, esse estudo mostrou como elementos lúdicos podem ser integrados ao currículo, promovendo uma aprendizagem mais interativa e eficaz. Destacou-se, também, que os jogos aumentam significativamente a absorção de conteúdos, tornando os conceitos de teste de *software* mais acessíveis e menos intimidadores, promovendo uma melhor compreensão e aplicação prática.

Figura 1 – Jogo ilearntest



Fonte: Ribeiro (2014)

Por outro lado, o trabalho de Diniz e Dazzi (2011) ofereceu uma perspectiva

adicional sobre o uso de jogos para ensinar conceitos específicos de teste de *software*. Em seu artigo, apresentou-se um jogo focado na prática dos conceitos de teste de caixa-preta, onde o jogador deve encontrar sete falhas em cada desafio. Esta abordagem difere de outras metodologias, oferecendo um método complementar e eficaz para o ensino de teste de *software*.

Figura 2 – Jogo das 7 falhas

Fonte: Diniz e Dazzi (2011)

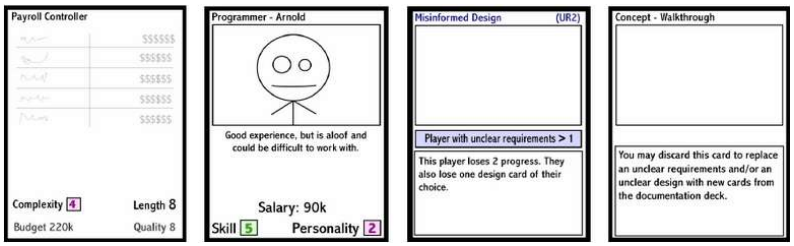
Nesse trabalho, destacou-se que os jogos aumentam a absorção de conteúdos e proporcionam uma maneira divertida de explorar conceitos técnicos. O artigo detalhou os benefícios dos jogos no ensino de teste de *software*, mostrando como a gamificação motiva os alunos. Diniz e Dazzi (2011) concluíram que o jogo é um método complementar eficaz para o ensino de teste de caixa-preta, motivando os alunos e apoiando o instrutor.

Da mesma forma, o trabalho de Fernandes *et al.* (2010) explorou a utilização de um jogo para complementar o aprendizado. Nesse caso, se tratou de um jogo de cartas visando ao aprimoramento do conhecimento de técnicas de *scrum* e métodos ágeis de desenvolvimento de *software*. Esse estudo demonstrou que o *PlayScrum* abordou muitas das fraquezas de abordagens tradicionais de aprendizagem e traz benefícios adicionais na forma de aprendizagem presencial. O autor, com base em seus resultados na etapa de validação, acreditou que o *PlayScrum* permitiu aos estudantes desenvolverem um sólido conhecimento e compreensão da utilização das técnicas no mundo real, complementando e aprimorando o aprendizado visto anteriormente. Segundo o autor, devido à dinâmica oferecida pelos jogos de cartas, o *PlayScrum* teve uma característica visual que o tornou simples de ser usado e proporcionou um aprendizado colaborativo. Com isso, foi capaz de fornecer um *feedback* quase imediato aos jogadores a respeito dos conteúdos a serem aprendidos.

Ainda na perspectiva de jogos de cartas, o trabalho de Navarro, Baker e Hoek (2004) é mais um exemplo da utilização dessa dinâmica. O jogo intitulado *Problems and*

Programmers é um jogo de cartas físico, jogado em turnos, que permite a um jogador atrasar o progresso do outro para alcançar o objetivo. Para ganhar o jogo, o jogador deve ser o primeiro a completar a integração do projeto e ter todas as cartas de código inspecionadas sem bugs. Caso o código final esteja livre de *bugs* e satisfaça o cliente, há um vencedor.

Figura 3 – Jogo *Problems and Programmers*



Fonte: Navarro, Baker e Hoek (2004)

O jogo percorre o ciclo de vida de um projeto e desafia os jogadores a administrar seus recursos e desafios da melhor forma possível. O autor realizou a validação com 28 estudantes de graduação do curso de engenharia de *software*, e, no fim, percebeu-se uma impressão positiva sobre o jogo. Os resultados obtidos com esse jogo e sua dinâmica serviram para inspirar a criação do jogo *PlayScrum* mencionado anteriormente. Com isso, se viu como um jogo de cartas pode não só proporcionar bons resultados para os alunos, mas também auxiliar na criação de novos jogos em áreas diferentes.

Por fim, Ribeiro (2014) e Diniz e Dazzi (2011) evidenciaram os benefícios dos jogos no ensino de teste de *software*, enquanto Fernandes *et al.* (2010) e Navarro, Baker e Hoek (2004) exploraram a gamificação por meio de jogos de cartas. Então, pôde-se perceber uma abordagem que integra elementos lúdicos no currículo, destacando a gamificação como um meio eficaz de aumentar a absorção de conteúdos e a motivação dos alunos. Evidenciou-se também que, ao analisar os trabalhos apresentados, estabeleceu-se o precedente da utilização de jogos como complemento ao ensino de testes de *software* e que jogos de cartas puderam ser eficazes mesmo em temas diferentes. Desse modo, um jogo de cartas voltado para o ensino de teste de *software* poderia explorar e praticar os conceitos vistos anteriormente pelo aluno.

O presente trabalho buscou explorar os precedentes estabelecidos na utilização de jogos para o ensino de teste de *software*, além de ter pesquisado e analisado conceitos já explorados anteriormente por outros autores, de modo a se ter uma base de tópicos relevantes que foram abordados no jogo *TestHunter*. Este trabalho teve foco em elaborar um jogo de cartas digital que complemente o aprendizado de teste de *software* e que possa auxiliar o professor ao ministrar a disciplina de qualidade de *software* ou similares.

3 METODOLOGIA

Este capítulo apresenta os aspectos metodológicos e abordagens utilizados no desenvolvimento deste trabalho, incluindo a classificação da pesquisa, a plataforma de *design* e as etapas que o compuseram, com as ferramentas, estratégias de ensino dos conceitos e requisitos.

3.1 Classificação do trabalho

O presente trabalho foi classificado conforme diferentes aspectos metodológicos que refletem a natureza e os objetivos da investigação, os procedimentos adotados, bem como a abordagem utilizada na análise dos dados. A pesquisa desenvolvida adotou uma abordagem qualitativa, que busca avaliar o processo e a compreensão do jogo educativo na aprendizagem. Em relação à natureza, enquadra-se como pesquisa aplicada, pois teve como objetivo principal aprofundar o conhecimento sobre o uso de jogos no auxílio ao aprendizado de práticas de teste de *software*. Este estudo busca contribuir para o conhecimento teórico na área, explorando como jogos podem ser utilizados de forma eficaz no ensino de testes de *software*.

O trabalho também se caracteriza como uma pesquisa exploratória, uma vez que visa proporcionar maior familiaridade com o tema e esclarecer certos conceitos. A pesquisa envolveu um levantamento bibliográfico sobre o uso de jogos no ensino de testes de *software*. Segundo Wazlawick (2009), a pesquisa não experimental consiste no estudo de fenômenos sem a intervenção sistemática do pesquisador. Dito isso, este trabalho enquadra-se na pesquisa não experimental, pois não envolveu a manipulação direta das variáveis relacionadas ao objeto de estudo.

3.2 Solução proposta

Conforme destacado por Peffers *et al.* (2007), é essencial definir claramente o problema de pesquisa e a importância de encontrar uma solução adequada. No Capítulo 1, foi identificado que a ausência de atividades práticas compromete a compreensão, por parte dos alunos, de conceitos fundamentais de teste de *software*. Isso gera desafios no processo de aprendizagem e pode levar à desmotivação.

Para abordar essa questão, este trabalho propõe a criação de um jogo de cartas que visa facilitar o ensino de teste de *software*. Este jogo não só pretende ajudar os alunos a entender e aplicar conceitos essenciais de testes de forma mais interativa e envolvente, mas também pode ser utilizado em sala de aula para promover uma aprendizagem colaborativa e dinâmica, auxiliando os docentes em sua prática pedagógica. A solução proposta é relevante porque visa permitir que os alunos enfrentem os desafios do aprendizado de teste de *software* de maneira prática e divertida, melhorando sua compreensão e motivação.

3.3 Materiais e métodos

Para o desenvolvimento deste trabalho, seguiram-se algumas etapas, listadas nas próximas subseções.

3.3.1 Etapa 1 - Escolha da plataforma de design

Para modelar e estruturar o jogo, optou-se pela utilização da plataforma Canva, que é uma ferramenta *online* voltada para *design* e comunicação visual. A plataforma se destaca por oferecer uma variedade de recursos gratuitos e acessíveis, permitindo que qualquer pessoa crie e publique suas próprias produções de forma prática e intuitiva (CANVA, 2024).

3.3.2 Etapa 2 - Identificação dos conteúdos

A identificação e seleção dos conteúdos a serem abordados no jogo *TestHunter* foram diretamente baseadas nos livros *Base de Conhecimento em Teste de Software*, de Bastos *et al.* (2007), e *Qualidade de Software*, de Koscianski e Santos Soares (2007). Após a leitura, identificaram-se treze tipos de testes selecionados e, com base nisso, elaboraram-se as cartas de conteúdo do jogo. Entre os testes abordados, foram classificados como sendo do tipo estrutural ou funcional. A seguir, é mostrada uma breve descrição de cada teste presente no jogo. Porém, nas cartas, é possível encontrar mais detalhes de cada teste:

- Testes Estruturais:
  - Teste de conformidade: Verifica se a aplicação foi desenvolvida conforme os padrões, procedimentos e guias de TI.
  - Teste de contingência: deve ser realizado quando a continuidade da operação do sistema seja essencial para as operações da organização.
  - Teste de estresse: avalia o comportamento do *software* sob condições críticas.
  - Teste de execução: avalia o comportamento do sistema no ambiente de produção.
  - Teste de operação: visa verificar, antes da entrada da aplicação em produção real, se os procedimentos podem ser executados adequadamente.
  - Teste de segurança: os testes de segurança são desenhados para avaliar a adequação dos procedimentos de proteção e as contramedidas projetadas.
- Testes funcionais:
  - Teste de controle: as técnicas deste teste são desenhadas para assegurar o funcionamento dos mecanismos que supervisionam o funcionamento do sistema de aplicações.

- Teste de integração: é desenhado para garantir que a interconexão entre os *softwares* de aplicação funcione corretamente.
- Teste de paralelo: serve para determinar se os resultados de um novo *software* de aplicação são consistentes com o processamento do antigo sistema ou da versão antiga do sistema.
- Teste de regressão: avalia novamente partes já testadas após a implementação de uma mudança em outra parte do sistema.
- Teste de requisitos: visa verificar se o sistema executa corretamente as funcionalidades.
- Teste de suporte manual: tem como objetivo verificar se os procedimentos de suporte manual estão documentados e completos.
- Teste de tratamento de erro: determina a capacidade do sistema de tratar apropriadamente transações incorretas.

3.3.3 Etapa 3 - Modelagem do jogo

Esta seção apresenta a modelagem do jogo de cartas digital, que teve como objetivo desenvolver uma dinâmica baseada nos conceitos de gamificação. O foco foi o processo de elaboração da mecânica e das interações do jogo.

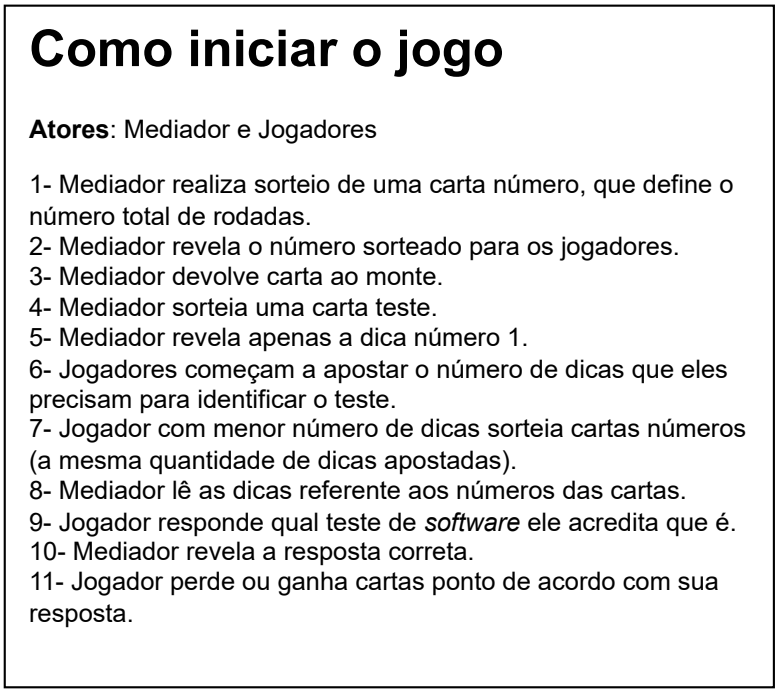
O processo de levantamento de requisitos envolve a identificação e coleta das necessidades e restrições, com o objetivo de assegurar que as expectativas dos usuários sejam atendidas. De acordo com Sommerville (2011), a especificação de requisitos abrange a definição dos requisitos dos usuários e do sistema, que devem ser claros, completos e consistentes. Sendo assim, o presente trabalho abordou este conceito para o levantamento de requisitos.

Após esse levantamento de requisitos, outro passo importante foi a confecção de um diagrama de caso de uso, que consiste em uma representação gráfica das interações entre os usuários ou atores e o jogo. Essa representação facilita a compreensão das funcionalidades e dos requisitos do sistema (VALENTE, 2020).

Um caso de uso, referente à dinâmica do jogo e que especifica os procedimentos iniciais, pode ser visto na Figura 4. Nela, é possível identificar os primeiros passos do início do jogo. Já a Figura 5 apresenta o diagrama de caso de uso, onde é possível ver graficamente a situação.

Com isso, foi possível começar a elaborar a dinâmica do *TestHunter*. No início do jogo, sorteia-se, pelo mediador, uma carta com um número de dois a dez, que indica o número total de rodadas do jogo. Logo após, a carta é devolvida ao monte e, aleatoriamente, se escolhe uma carta de teste. O mediador deve ler em voz alta a dica número um (que revelará se o teste em questão é do tipo funcional ou estrutural). O restante da carta deve ser de conhecimento apenas do mediador.

Figura 4 – Caso de uso



Fonte: Elaborado pela autora, 2024.

Em seguida, os jogadores (em sentido horário) devem dizer com quantas dicas acreditam ser capazes de adivinhar o nome do teste. Cada jogador deve apostar um número menor que o anterior, com um máximo de 10 e mínimo de 2. Quando um jogador indicar um número e o outro acreditar que é muito baixo para adivinhar o teste com essa quantidade de dicas, ele deve solicitar que o último a fazer a aposta tente adivinhar com o número de dicas que apostou.

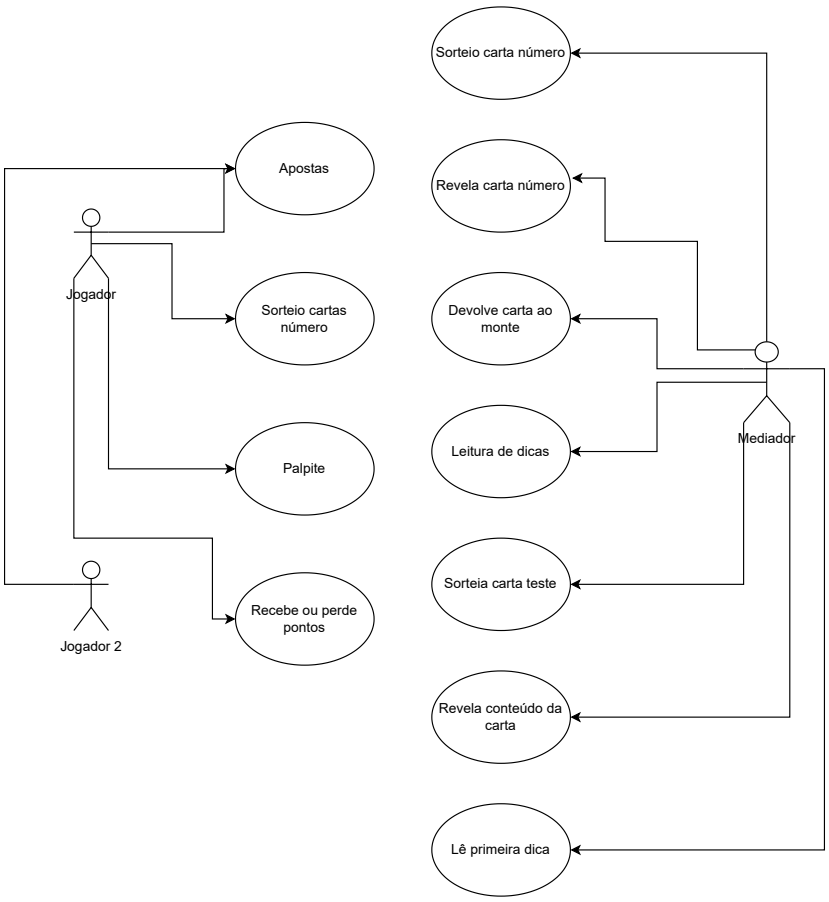
Se o jogador acertar, ele ganha cartas-ponto. Se não conseguir acertar, perde cartas. Um exemplo da dinâmica pode ser visto na Figura 6 , e o manual contendo as regras completas e exemplos de casos está disponível na Seção 4.1, Figura 7, 8 e 9.

3.3.4 Etapa 4 - Disponibilização do jogo

O jogo está disponível no repositório do GitHub[1]. Ao clicar no *link*, o usuário encontra todas as cartas, um manual contendo as regras e dicas do jogo, além das instruções iniciais para sua obtenção. Ademais, todas as cartas estão também disponíveis no apêndice deste trabalho.

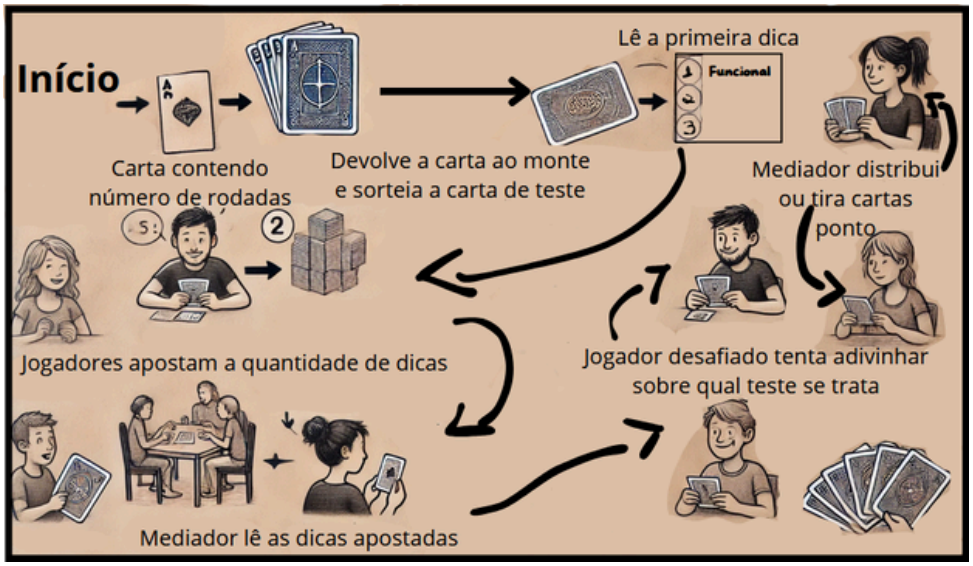
[1] *Link* para o repositório no GitHub: https://github.com/GabrielaDamaso/TestHunter

Figura 5 – Diagrama de caso de uso



Fonte: Elaborado pela autora, 2024.

Figura 6 – Dinâmica do jogo



Fonte: Elaborado pela autora, 2024.

4 DESENVOLVIMENTO

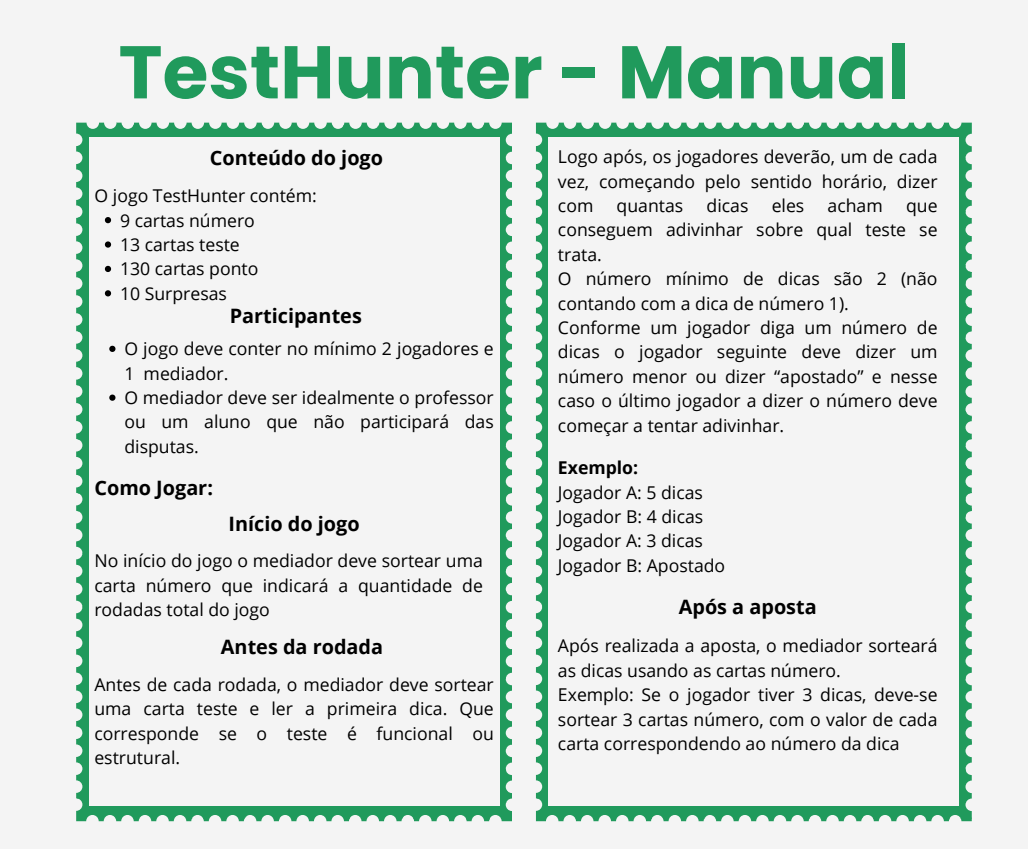
Este capítulo aborda o processo de criação do jogo *TestHunter*. As seções seguintes descrevem detalhadamente cada fase do projeto.

4.1 Manual

Esta seção visa explicar o funcionamento da dinâmica do jogo e os possíveis casos de interação entre jogadores. Nas Figuras 7, 8 e 9, é possível ver, na íntegra, o manual contendo as regras do jogo.

Com base na fase de levantamento de requisitos, foi possível elaborar a dinâmica, 4.2, e regulamentar o processo. O manual apresenta o passo a passo do funcionamento do jogo, assim como exemplo de situações, quantidade de cartas pertencentes, distribuição de pontos e como o jogo chega ao fim.

Figura 7 – Manual primeira parte

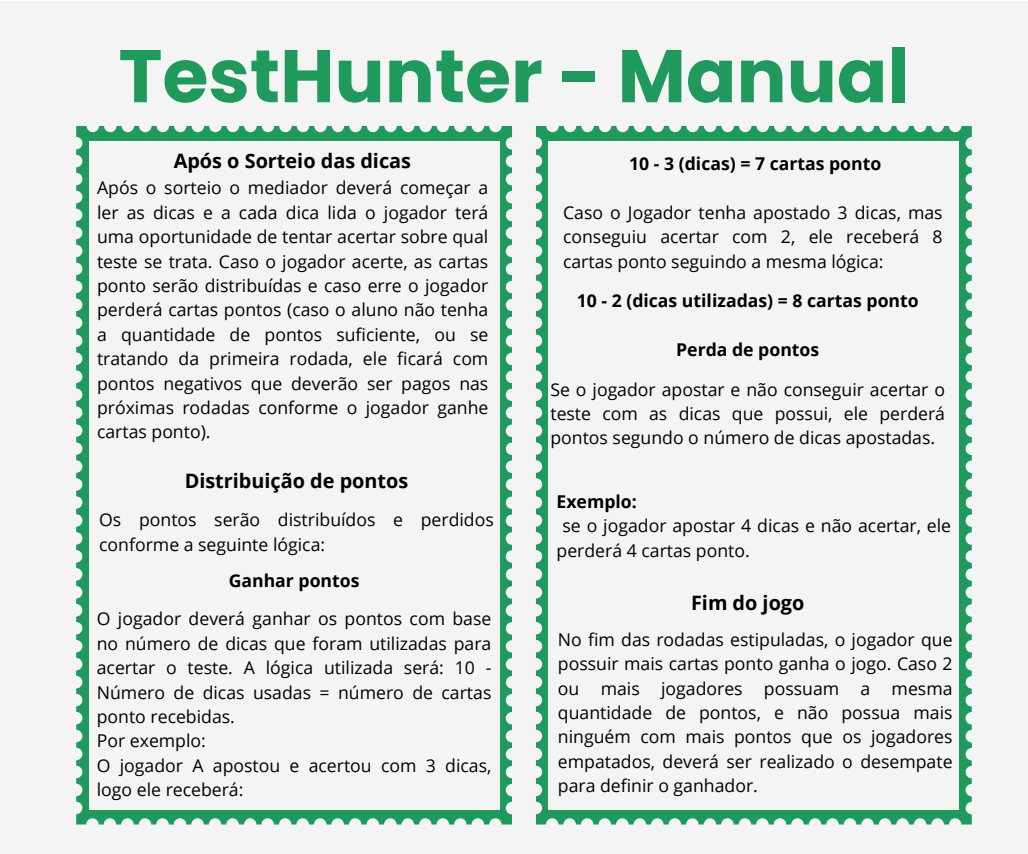


Fonte: Elaborado pela autora, 2024.

4.2 Dinâmica

Nesta seção, é descrita a dinâmica do jogo, detalhando as etapas necessárias para sua execução.

Figura 8 – Manual segunda parte



Fonte: Elaborado pela autora, 2024.

4.2.1 Etapa 1: Início do jogo

O jogo começa com a definição do mediador (recomenda-se que seja o professor ou um aluno disposto a não participar das disputas). Em seguida, o mediador pega uma carta-número, que representará a quantidade de rodadas a serem realizadas. Após o sorteio, o mediador devolve a carta ao monte, realiza o sorteio de uma carta de teste e lê, para os jogadores, a dica número 1 (a qual revela se o teste em questão é funcional ou estrutural).

4.2.2 Etapa 2: Apostas

A partir desse ponto, em sentido horário, os jogadores começam a apostar quantas dicas eles acreditam serem capazes de usar para adivinhar qual é o teste. O palpite de cada jogador deve ser sempre menor que o palpite do jogador anterior, porém não pode ser inferior a dois. A dica número 1, que é lida antes das apostas, não é considerada nessa etapa.

Quando o número mínimo de palpites for atingido ou um jogador não acreditar que consegue adivinhar com menos dicas do que o jogador anterior, ele deve informar o mediador. Nesse momento, o jogador anterior terá a oportunidade de adivinhar o teste.

Figura 9 – Manual terceira parte

Fonte: Elaborado pela autora, 2024.

4.2.3 Etapa 3: Adivinhação

Na etapa de adivinhação, o mediador realiza um sorteio de cartas de número de acordo com a quantidade de palpites apostados pelo jogador. Os números das cartas sorteadas indicam quais dicas serão lidas. O mediador lê uma dica por vez e, após a leitura de cada dica, o jogador tem uma oportunidade de adivinhar o teste. Portanto, o número de tentativas para adivinhar corresponde ao número de palpites apostados.

4.2.4 Etapa 4: Distribuição de pontos

Caso o jogador acerte o nome do teste em questão, ele receberá cartas-ponto de acordo com a quantidade de dicas usadas. A distribuição de pontos segue a seguinte fórmula:

$$10 - \text{Quantidade de Dicas} = \text{Pontos} \tag{1}$$

Onde 10 representa a quantidade máxima de pontos que podem ser distribuídos por rodada; "Quantidade de Dicas" representa o número total de dicas utilizadas pelo jogador; e "Pontos"é o resultado final de cartas-ponto a serem obtidas. Nota-se que a quantidade de dicas a ser utilizada na equação é a quantidade de dicas realmente usadas

pelo jogador; então, mesmo que tenham apostado uma quantidade maior de dicas, só será computada a quantidade efetivamente usada. Por exemplo: o jogador pode ter apostado 5 dicas, porém acertou o teste usando 2 dicas; neste caso, teremos a seguinte equação:

$$10 - 2 = 8 \tag{2}$$

Outro critério, também relevante na dinâmica do jogo, é a perda de pontos. Caso o jogador, em sua vez de adivinhar o teste, não consiga acertar, ele perderá cartas-ponto de acordo com a quantidade de dicas apostadas. Ou seja, caso tenham sido apostadas 3 dicas e o jogador não acerte o teste, ele perderá 3 pontos. Caso seja a primeira rodada ou o jogador não tenha pontos suficientes para perder, ele entregará os pontos que já possui, e o restante ficará como saldo negativo, devendo ser pago ao mediador assim que esses pontos forem conquistados em outras rodadas.

4.2.5 Etapa 5: Finalização e desempate

Após o término das rodadas preestabelecidas, faz-se a contabilização dos pontos, e o jogador que tiver mais cartas-ponto ganha o jogo. Porém, caso dois ou mais jogadores possuam a mesma quantidade de pontos, serão executadas rodadas de desempate. As rodadas de desempate devem seguir a seguinte regra:

- Apenas os jogadores empatados e com a maior quantidade de pontos podem participar dessas rodadas.

Nesta rodada, o mediador irá sortear uma carta surpresa e ler para todos os jogadores participantes. Somente após o fim da leitura, os jogadores podem responder. O jogador que levantar a mão primeiro, após o término da leitura, terá direito a responder. Caso o jogador acerte, ele ganhará uma carta-ponto e vencerá o jogo; caso erre, o jogador estará fora da competição. Se houver mais jogadores empatados, deve-se sortear outra carta e repetir o processo até que um jogador seja o vencedor.

4.3 Elaboração do jogo

Para atender aos conceitos de gamificação previamente mencionados 2.1.3, foram definidas algumas estratégias que o jogo deveria incorporar. Os principais conceitos de gamificação trabalhados são: competição, recompensas, desafios e *feedback* imediato. A incorporação desses elementos ao jogo foi detalhada nas seções: 4.3.1, 4.3.2, 4.3.3 e 4.3.4.

4.3.1 Quantidade de jogadores e o conceito de competição

O jogo foi projetado para ser jogado com, no mínimo, 2 e, no máximo, 8 jogadores, excluindo-se o mediador. Como uma das estratégias de gamificação adotadas

é a competição, é necessário um número mínimo de 2 jogadores para que haja disputa. O limite máximo de 8 jogadores foi estabelecido para evitar que o jogo se torne longo e enfadonho, o que comprometeria a dinâmica das rodadas.

4.3.2 Sistema de pontos e o conceito de recompensa

Para atender ao conceito de gamificação relacionado a recompensas, foi implementado um sistema de pontuação no jogo. A lógica utilizada para determinar a quantidade de cartas-ponto é a seguinte:

$$\text{Pontos} \times \text{Cartas} = \text{Quantidade} \tag{3}$$

Onde "Pontos"representa a quantidade máxima de pontos que pode ser obtida por rodada; "Cartas"significa o número total de cartas-teste; e "Quantidade"representa a quantidade de cartas-ponto disponíveis no jogo. Com base nesses dados, chega-se aos seguintes valores:

$$10 \times 13 = 130 \tag{4}$$

4.3.3 Cartas-teste e o conceito de feedback imediato

O conceito de *feedback* imediato é aplicado por meio das cartas-teste, nas quais, ao receber uma dica, o jogador pode fazer um palpite e imediatamente obter a resposta sobre se está certo ou não. Caso o palpite esteja incorreto, o teste em questão é revelado ao final da rodada, permitindo ao jogador conhecer o desempenho e aprender com a experiência. A quantidade de cartas-teste foi determinada com base na análise do livro de Bastos *et al.* (2007), que enfatiza os treze temas abordados nas cartas.

4.3.4 Cartas-surpresa e o conceito de desafio

Para explorar o conceito de desafio e introduzir um critério de desempate, foram criadas as cartas-surpresa, que são utilizadas apenas no final do jogo e em situações específicas. Assim, quando o jogo chega ao final, o participante que deseja vencer é desafiado a responder uma pergunta em um formato diferente do utilizado no restante do jogo, o que gera um desejo adicional de ganhar e a sensação de superação de um desafio.

5 CONCLUSÃO

O presente trabalho propôs o desenvolvimento de um jogo de cartas, o *TestHunter*, com o intuito de auxiliar no ensino dos conceitos de teste de *software*, tornando o aprendizado mais dinâmico e envolvente. A partir de uma revisão bibliográfica e da análise de abordagens existentes, como as de Ribeiro (2014) e Fernandes *et al.* (2010), foi possível identificar a necessidade de práticas educativas mais interativas no ensino de testes de *software*, especialmente em cursos de graduação. Os estudos mostraram que a gamificação pode ser uma estratégia eficaz para motivar os alunos e aumentar o engajamento por meio de elementos lúdicos. Com base nessas evidências, a gamificação foi incorporada ao projeto para promover uma experiência de aprendizado ativa, fundamentada em desafios, recompensas e *feedbacks* imediatos.

Durante o desenvolvimento do *TestHunter*, foram seguidas etapas que incluíram a escolha da plataforma de *design*, a identificação dos conteúdos de teste de *software* a serem abordados e a modelagem das mecânicas de jogo. O jogo foi projetado com uma dinâmica que simula cenários de apostas e competição, em que os jogadores devem adivinhar o tipo de teste de *software* a partir de dicas fornecidas em rodadas sucessivas. A estrutura do jogo inclui diferentes tipos de cartas, como cartas-teste, cartas-ponto e cartas-surpresa, que permitem aos jogadores experimentar conceitos de teste funcional e estrutural de forma lúdica e educativa. Além disso, as mecânicas de pontuação e penalização foram elaboradas para incentivar a participação ativa e o aprendizado contínuo, proporcionando aos alunos a oportunidade de rever conceitos teóricos.

Os objetivos estabelecidos no início do projeto foram atendidos, destacando-se a seleção dos conteúdos relevantes, baseada na literatura e em referências da área de teste de *software*. A solução proposta representa uma alternativa prática e inovadora às metodologias tradicionais de ensino, complementando as aulas teóricas e ajudando os alunos a fixar os conceitos de forma efetiva. O uso de uma abordagem gamificada mostrou-se promissor para superar a falta de atividades práticas, com potencial para enriquecer o currículo acadêmico e fomentar o interesse pela área de qualidade de *software*.

Entretanto, algumas limitações foram identificadas ao longo do processo. A necessidade de testar o jogo com diferentes grupos de estudantes ainda se faz presente para avaliar de forma mais abrangente a eficácia do *TestHunter* em contextos variados e em instituições com diferentes perfis de alunos. Para trabalhos futuros, sugere-se expandir o jogo para incluir uma gama maior de tópicos relacionados a testes de *software*, como testes automatizados e técnicas avançadas de verificação. Além disso, explorar outras mecânicas de jogo, como modos colaborativos ou integração com plataformas digitais, pode potencializar ainda mais o impacto educativo. Implementar um *software* do jogo pode contribuir para uma experiência mais imersiva e acessível aos estudantes, favorecendo a aplicação prática dos conceitos. Realizar estudos comparativos com outros métodos de ensino ajudará a analisar o efeito do *TestHunter* no desempenho acadêmico e na retenção

do conhecimento dos alunos.

Assim, espera-se que este projeto contribua significativamente para o ensino de qualidade de *software*, promovendo uma formação mais completa e prática dos futuros profissionais, e que se estabeleça como uma ferramenta educacional útil para docentes e alunos da área.

REFERÊNCIAS

BASTOS, A. *et al.* **Base de conhecimento em teste de software**. 2. ed. São Paulo: Martins Fontes, 2007.

BATISTA, D. A.; DIAS, C. L. O processo de ensino e de aprendizagem através dos jogos educativos no ensino fundamental. In: COLLOQUIUM humanarum. 2012. v. 9, p. 975–982.

BENITTI, F.; ALBANO, E. Teste de Software: o que e como é ensinado? In: ANAIS do XX Workshop sobre Educação em Computação. Curitiba/PR: SBC, 2012. p. 180–189. Disponível em: https://sol.sbc.org.br/index.php/wei/article/view/29080.

CANVA. **Sobre o Canva**. Acesso em: 15 ago. 2024. 2024. Disponível em: https://www.canva.com/pt_br/about/.

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

DINIZ, L. L.; DAZZI, R. L. S. Jogo das sete falhas: Um jogo educacional para apoio ao ensino do teste caixa preta. **Anais do Computer on the Beach**, p. 1–10, 2011.

FERNANDES, J. M. *et al.* PlayScrum - A Card Game to Learn the Scrum Agile Method. **null**, 2010. DOI: 10.1109/vs-games.2010.24.

FIALHO, N. N. Os jogos pedagógicos como ferramentas de ensino. In: 1. CONGRESSO nacional de educação. 2008. v. 6, p. 12298–12306.

KOSCIANSKI, A.; SANTOS SOARES, M. dos. **Qualidade de Software-2ª Edição: Aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software**. Novatec Editora, 2007.

LI, Q. *et al.* Evaluation of gamification techniques in learning abilities for higher school students using FAHP and EDAS methods. **Soft Computing**, Springer, p. 1–19, 2023.

MURR, C. E.; FERRARI, G. Entendendo e aplicando a gamificação: o que é, para que serve, potencialidades e desafios. **UFSC. E-BOOK**, 2020.

MYERS, G. J. **The art of software testing**. John Wiley & Sons, 2006.

NASCIMENTO, E. H. R. d. **Aplicando gamificação no ensino de teste de software**. 2019. Diss. (Mestrado) – Brasil.

NAVARRO, E.; BAKER, A.; HOEK, A. van der. Teaching software engineering using simulation games. In.

PEFFERS, K. *et al.* A design science research methodology for information systems research. **Journal of Management Information Systems**, v. 24, p. 45–77, 2007. Acesso em: 16 ago. 2024. Disponível em: https://www.researchgate.net/publication/284503626.

POSSATTO, L. B.; JAGNOW, C. A Contribuição Dos Jogos No Processo Ensino/Aprendizagem. **Revista Científica Multidisciplinar Núcleo do Conhecimento. Ano**, v. 3, n. 11, p. 144–165, 2022.

PRESSMAN, R. S. **Software engineering: a practitioner's approach**. Palgrave macmillan, 2005.

RIBEIRO, T. P. B. ilearntest: Jogo educativo para aprendizagem de testes de software, 2014.

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

SZETO, M. D. *et al.* Gamification and game-based strategies for dermatology education: narrative review. **JMIR dermatology**, JMIR Publications Toronto, Canada, v. 4, n. 2, p. e30325, 2021.

VALENTE, M. T. Engenharia de software moderna. **Princípios e Práticas para Desenvolvimento de Software com Produtividade**, v. 1, n. 24, 2020.

VALLE, P.; BARBOSA, E. F.; MALDONADO, J. Um mapeamento sistemático sobre ensino de teste de software. In: 1. BRAZILIAN Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE). 2015. v. 26, p. 71.

WAZLAWICK, R. S. **Metodologia de pesquisa para ciência da computação**. Elsevier Rio de Janeiro, 2009. v. 2.

APÊNDICES

Teste de estresse

1 - É um tipo de teste estrutural
2 - É realizado colocando o software sob condições mínimas de operação
3 - Visa identificar o comportamento do software quando submetido a volumes de dados acima do esperado
4 - O teste deve simular o mais próximo possível do ambiente de produção
5 - Um dos objetivos é determinar que volumes de transações normais e acima do normal podem ser processados num período esperado
6 - Um dos objetivos é determinar se o sistema é capaz de processar grande volume de dados
7 - Um dos objetivos é determinar se a capacidade do sistema tem recursos suficientes disponíveis para garantir os tempos de resposta adequado
8 - Um dos objetivos é determinar se as pessoas podem executar suas tarefas e manter o tempo de resposta adequado
9 - Um dos objetivos é determinar se há restrições quanto ao ambiente em que o software vai operar
10 - Esse teste sobrecarrega o sistema e tenta levá-lo a falhar, com grande volume de transações

Teste de execução

1 - É um tipo de teste estrutural
2 - É usado para avaliar o comportamento do sistema no ambiente de produção e verificar se são atendidas as premissas de desempenho estabelecidas
3 - Esse teste verifica o tempo de resposta, os tempos de processamento e o desempenho (performance)
4 - Esse teste pode ser usado no sistema inteiro ou em partes
5 - Tem como objetivo determinar se o sistema pode atingir os critérios de desempenho específicos
6 - Tem como objetivo verificar o nível de utilização de hardware e do software
7 - Tem como objetivo determinar o tempo de processamento das transações
8 - Pode ser usado de modo a simular o funcionamento de todas as partes ou apenas de uma determinada seção de um sistema, usando um sistema de simulação
9 - Pode ser usado na criação de programas temporários para avaliar o desempenho
10 - Esse teste pode ser realizado na própria instalação ou em terceiros

Teste de contigência

1 - É um tipo de teste estrutural
2 - Tem como objetivo garantir a continuidade das operações após um desastre
3 - Esse teste verifica o processo de recuperação e a eficácia das partes componentes do processo
4 - Um de seus objetivos é manter um backup de dados
5 - Um de seus objetivos é documentar o procedimento de recuperação
6 - Um de seus objetivos é nomear pessoas responsáveis pela recuperação e verificar se foram devidamente treinadas
7 - Um de seus objetivos é disponibilizar ferramentas para a recuperação
8 - O teste pode ser desenhado para avaliar a capacidade dos usuários de continuar operando e de operações retornarem ao normal depois de ocorrer um problema
9 - Esse teste deve ser realizado sempre que a continuidade da operação do sistema for essencial para as operações da área de negócio ou da organização
10 - A quantidade de perda potencial determina a quantidade de recursos a ser alocada no planejamento de contingência e no planejamento desse teste.

Teste de operação

1 - É um tipo de teste estrutural.
2 - Este teste é desenhado para verificar, antes da entrada da aplicação em produção real, se os procedimentos da produção e os operadores podem executar adequadamente a aplicação.
3 - Este teste é desenhado para estabelecer se o sistema é executável durante a operação normal.
4 - Este teste tem como objetivo: determinar que a documentação da operação está completa.
5 - Este teste tem como objetivo: garantir que os mecanismos de suporte foram corretamente preparados e funcionam de modo adequado.
6 - Este teste tem como objetivo: testar que os operadores, usando a documentação preparada, conseguem efetivamente operar o sistema.
7 - Durante a fase de desenho do software os procedimentos operacionais identificados são projetados e avaliados .
8 - Durante a realização desse teste, os operadores não devem ser avisados nem obter ajuda externa durante o processo.
9 - Os testes precisam ser executados como se fizessem parte da operação normal do computador de modo que seja avaliada a efetividade da operação em processar a aplicação no ambiente real.
10 - Importante que as falhas de operação sejam identificadas ainda como falhas de implementação e antes de se iniciar a produção.

Teste de conformidade

1 - É um tipo de teste estrutural
2 - Verifica se a aplicação foi desenvolvida de acordo com os padrões, procedimentos e guias de ti
3 - Pode ser mais importante executar este tipo de teste durante a fase de requisitos do que nos estágios finais do ciclo de vida.
4 - Este teste é realizado para garantir a conformidade com as metodologias para encorajar e auxiliar os profissionais de ti a adotá-las
5 - Um dos objetivos é verificar se as metodologias de desenvolvimento de software e de manutenção são seguidas
6 - Um dos objetivos é garantir a conformidade aos padrões, procedimentos e guias de ti
7 - Um dos objetivos é avaliar se a documentação do sistema de aplicação é racional e está completa
8 - Pode ser detalhado e demorado, exigindo uma revisão minuciosa de documentação e processos.
9 - Muitas vezes, este teste faz parte do processo de integração contínua.
10 - Pode ser utilizado para garantir o cumprimento de políticas definidas pelo usuário

Teste de segurança

1 - É um tipo de teste estrutural
2 - É um processo necessário para garantir a confidencialidade das informações e a proteção dos dados contra o acesso indevido de terceiros
3 - Este teste é desenhado com o intuito de avaliar a adequação dos procedimentos de proteção e as contra medidas projetadas
4 - Um dos objetivos é determinar se foi dada a atenção adequada a identificação de riscos de segurança
5 - Um dos objetivos é determinar se foi preparada uma definição realista das regras de acesso ao sistema e se estas foram implementadas de acordo com as definições
6 - Um dos objetivos é conduzir testes racionais para garantir que as medidas de segurança tenham sido corretamente implementadas
7 - Primeiro passo é identificar os riscos de segurança e a potencial perda associada a esses riscos
8 - Se os riscos são baixos ou as maneiras de evitar as invasões são as usuais, o pessoal de ti pode conduzir os testes necessários. Senão precisa contratar especialistas
9 - Este tipo de teste pode inicialmente ser dividido em segurança física e lógica
10 - Esse tipo de teste deve ser empregado quando as informações protegidas pelo sistema de aplicação são de alto valor para a organização.

Teste de requisitos

1 - É um tipo de teste funcional
2 - Este tipo de teste visa verificar se o sistema executa corretamente as funcionalidades e se é capaz de sustentar essa correção após sua utilização por um período de tempo contínuo
3 - Este teste deve ser considerado formalmente realizado após os programas se tornarem operacionais
4 - Um dos objetivos deste teste é verificar se os requisitos dos usuários estão implementados
5 - Um dos objetivos deste teste é verificar se o processamento da aplicação está em conformidade com as políticas e os procedimentos da organização
6 - Uma boa prática é realizar revisões regulares dos requisitos para garantir que as condições de teste permaneçam relevantes e completas.
7 - Esse tipo de teste é realizado basicamente através da criação de condições de testes e checklists de funcionalidades
8 - As condições de teste são preparadas inicialmente de maneira genérica durante a fase de requisitos
9 - Recomenda-se que as condições de teste sejam derivadas diretamente dos requisitos e não das documentações preparadas para o sistema
10 - Toda aplicação deve ter seus requisitos testados

Teste de regressão

1 - É um tipo de teste funcional
2 - Esse tipo de teste volta a testar segmentos já testados após a implementação de uma mudança em outra parte do software
3 - Sempre que Mudanças são efetuadas num segmento de código, problemas podem ocorrer em outros segmentos já testado
4 -Um dos objetivos é determinar se os dados e as condições de teste permanecem atuais
5 - Um dos objetivos é determinar se as funções previamente testadas continuam funcionando corretamente após a introdução de mudanças no sistema
6 - Este teste deve ser executado sempre que o software sofrer uma alteração
7 - Quando houver uma alteração no software deve-se aplicar os mesmos testes realizados anteriormente para garantir que os resultados não tenham sido afetados pelas mudanças
8 - Um dos problemas deste teste é o gasto excessivo de tempo e a cansativa repetição das operações
9 - Este teste é empregado quando é muito alto o risco de novas mudanças afetarem áreas não alteradas da aplicação
10 - No processo de desenvolvimento, este teste deve ser aplicado após a realização de um número predeterminado de alterações no sistema

Figura 10 – Cartas teste A

Teste de tratamento de erros

1 - É um tipo de teste estrutural
2 - Este teste determina a capacidade do sistema de tratar apropriadamente transações incorretas.
3 - Um dos objetivos é determinar se todas as condições de erro esperadas são reconhecidas pelo sistema.
4 - Um dos objetivos é determinar se foi atribuída responsabilidade para processar os erros identificados.
5 - Um dos objetivos é determinar se é mantido um controle razoável sobre os erros durante o processo de correção.
6 - Bom método para desenvolver esse teste é realizar um brainstorm, procurando identificar o que pode dar errado com o sistema.
7 - Devem ser testados a introdução do erro, o processamento do erro, a condição de controle e a nova introdução da condição apropriadamente corrigida.
8 - Este teste é um processo interativo no qual os erros são inicialmente introduzidos, identificados e corrigidos, e depois novamente introduzidos para que o ciclo se complete.
9 - Este teste deve ocorrer durante todo o ciclo de vida de desenvolvimento do sistema.
10 - Em todos os pontos do processo de desenvolvimento, o impacto dos erros deve ser identificado.

Teste de controle

1 - É um tipo de teste funcional.
2 - As técnicas deste teste são desenhadas para assegurar o funcionamento dos mecanismos que supervisionam o funcionamento do sistema de aplicações.
3 - Um dos seus objetivos é garantir que os dados estejam completos e corretos.
4 - Um dos seus objetivos é garantir que as transações sejam autorizadas.
5 - Um de seus objetivos é garantir que a manutenção das informações da trilha de auditoria seja realizada.
6 - Um de seus objetivos é garantir que os processamentos sejam eficientes, eficazes e econômicos.
7 - Os controles são desenhados para reduzir riscos e para isto estes devem ser identificados.
8 - Responsável pelo teste cria situações de risco para verificar se os controles são eficazes em reduzi-las a um nível aceitável.
9 - Um método que pode ser empregado é o desenvolvimento da Matriz de riscos.
10 - Os controles devem ser vistos como um sistema dentro do software de aplicação e testados em paralelo com os testes de outros software.

Teste de suporte manual

1 - É um tipo de teste funcional.
2 - Um dos objetivos é Verificar se os procedimentos de suporte manual estão documentados e completos.
3 - Um dos objetivos é determinar se as responsabilidades pelo suporte manual foram estabelecidas.
4 - Um dos objetivos é determinar se o pessoal que dará o suporte manual está adequadamente treinado.
5 - Dos objetivos é determinar se o suporte manual e o segmento automatizado estão interligados apropriadamente.
6 - Esse tipo de teste envolve, a princípio, a avaliação da adequação do processo e, posteriormente, sua execução que pode ser feita juntamente com o teste normal do sistema.
7 - Teste pode ocorrer sem o apoio do pessoal de sistemas, bastando que tenha equipe de apoio manual e forneça os procedimentos a serem executados.
8 - Os resultados dos Testes devem ser avaliados pelo pessoal de sistemas que verificará se os procedimentos foram executados adequadamente.
9 - Este teste não deve ser deixado totalmente para os últimos estágios do ciclo de desenvolvimento de software.
10 - Este tipo de teste requer coordenação entre a equipe de suporte manual e o pessoal de sistemas.

Teste de integração

1 - É um tipo de teste funcional.
2 - Esse teste é desenhado para garantir que a interconexão entre os software de aplicação funcione corretamente.
3 - Este teste ajuda a identificar problemas de comunicação entre os diferentes componentes do sistema.
4 - Este teste aumenta a confiança na integração e interação dos sistemas.
5 - Um dos objetivos é determinar se os parâmetros e dados são transferidos corretamente entre os software.
6 - Um dos objetivos é garantir o momento certo de execução e a existência de coordenação das funções entre os software.
7 - Um dos objetivos é determinar se a documentação pertinente é correta e completa.
8 - Este teste envolve a operação de vários softwares em teste, o que pode ser custoso e demorado.
9 - Processo consiste basicamente em passar os dados previstos entre os diversos software envolvidos e verificar se a transferência foi bem-sucedida.
10 - Teste deve ser conduzido sempre que existir uma mudança nos parâmetros entre software de aplicações.

Teste paralelos

1 - É um tipo de teste funcional.
2 - Este teste serve para determinar se os resultados de um novo software de aplicação são consistentes com o processamento do antigo sistema ou da versão antiga do sistema.
3 - Este tipo de teste requer um bom entendimento de ambos os sistemas para ajustar os dados de entrada e interpretar os resultados.
4 - Um dos objetivos é assegurar que a nova versão do software de aplicação Execute corretamente.
5 - Um dos objetivos é demonstrar consistências e inconsistências entre duas versões do mesmo software de aplicação.
6 - Este teste exige que os mesmos dados de entrada rodem em duas versões da mesma aplicação.
7 - Teste pode ser realizado no software inteiro ou com apenas um ou mais segmentos.
8 - Se a nova aplicação mudar o formato dos dados Então os dados de entrada deverão ser previamente modificados para rodar na nova aplicação.
9 - Caso apresente resultados diferentes entre as versões, faz-se necessário conciliar manualmente os resultados dos dois softwares para garantir a correção do processamento do novo sistema.
10 - Este teste deverá ser usado Quando existirem incertezas com relação à correção do processamento da nova aplicação e se as duas versões foram similares.

Figura 11 – Cartas teste B

Carta surpresa

Múltipla escolha:

Qual é o principal objetivo do teste de regressão?
a) Verificar se sistema funciona sobre grande volume de dados.
b) Garantir que alterações no software não introduziram novos defeitos em partes que já estavam funcionando.
c) Avaliar a documentação do software para conformidade com os padrões.

Resposta:
B

Carta surpresa

Verdadeiro ou falso:

O custo da correção de defeitos tende a ser maior quanto mais tarde o defeito é corrigido.

Resposta:
Verdadeiro

Carta surpresa

Múltipla escolha:

Em que momento o teste de regressão é mais indicado?
a) Após a implementação de novas funcionalidades
b) Durante a definição de requisitos
c) Antes de lançar a versão inicial do software

Resposta:
A

Carta surpresa

Múltipla escolha:

Qual das alternativas a seguir descreve corretamente o teste de unidade?
a) Verifica a interação entre diferentes módulos do sistema
b) Avalia se uma funcionalidade atende aos requisitos do usuário
c) Testa partes isoladas e específicas do código para garantir que funcionam corretamente

Resposta:
C

Carta surpresa

Verdadeiro ou falso:

As técnicas de teste de software só podem ser realizadas após o software estar totalmente finalizado

Resposta:
Falso

Carta surpresa

Múltipla escolha:

O que é um bug no contexto de testes de software?
a) Uma melhoria solicitada pelo cliente
b) Um defeito ou falha no código que causa um comportamento incorreto do software
c) Uma atualização de versão do software

Resposta:
B

Figura 12 – Cartas surpresa A

Carta surpresa

Múltipla escolha:

Qual é a principal diferença entre teste de desempenho e teste de estresse?
a) O teste de desempenho mede a resposta do sistema sob carga normal, enquanto o teste de estresse avalia o comportamento sob carga extrema
b) O teste de estresse verifica a conformidade do software com os requisitos, enquanto o teste de desempenho avalia a usabilidade
c) Ambos são realizados da mesma forma, apenas variando o tipo de sistema testado

Resposta:
A

Carta surpresa

Múltipla escolha:

Qual é o principal objetivo do teste de integração?
a) Verificar a integração de diferentes módulos do sistema
b) Testar funcionalidades isoladas do software
c) Avaliar o desempenho do software em condições extremas

Resposta:
A

Carta surpresa

Múltipla escolha:

Qual técnica de teste é usada para validar o comportamento do software a partir das especificações de entrada e saída?
a) Teste de caixa branca
b) Teste de caixa preta
c) Teste de aceitação

Resposta:
B

Carta surpresa

Múltipla escolha:

O que significa automação de testes?
a) Realizar testes manualmente de forma mais eficiente
b) Utilizar ferramentas e scripts para executar testes de forma automatizada, sem intervenção manual
c) Delegar a execução de testes para os usuários finais

Resposta:
B

Figura 13 – Cartas surpresa B

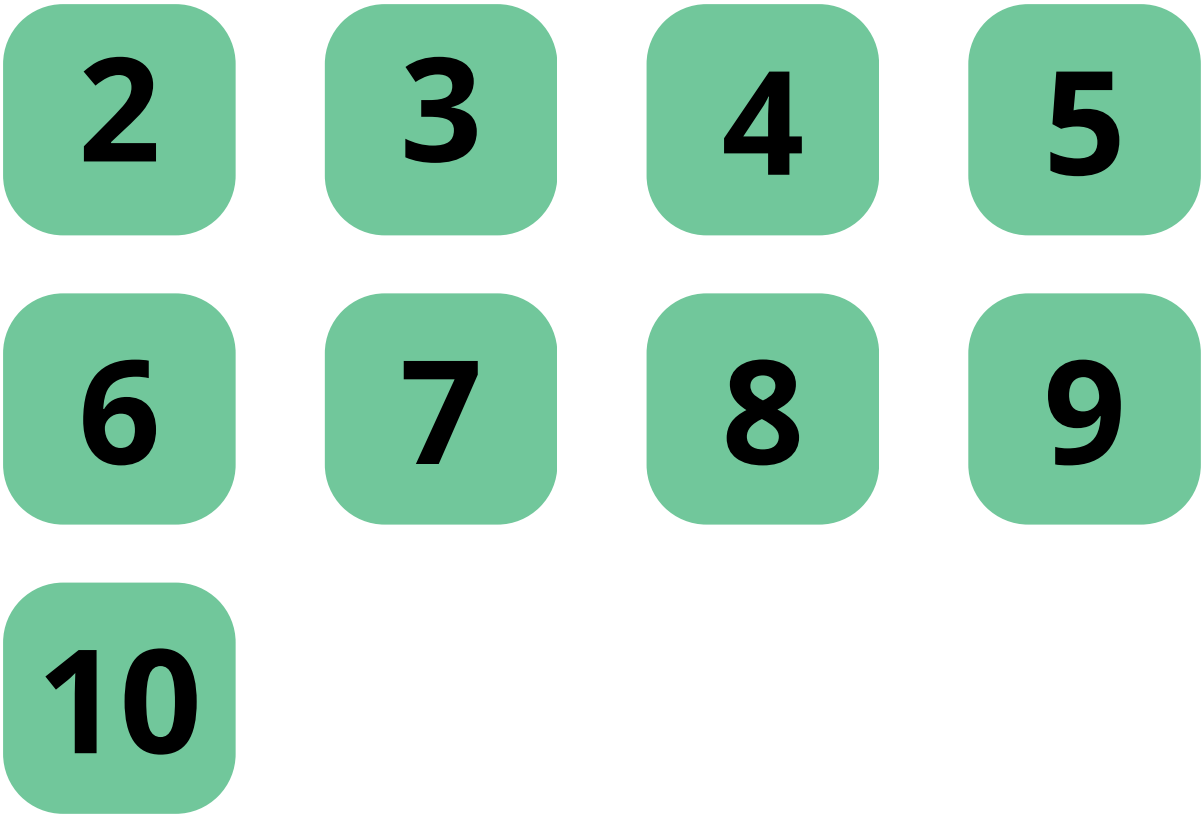


Figura 14 – Cartas número

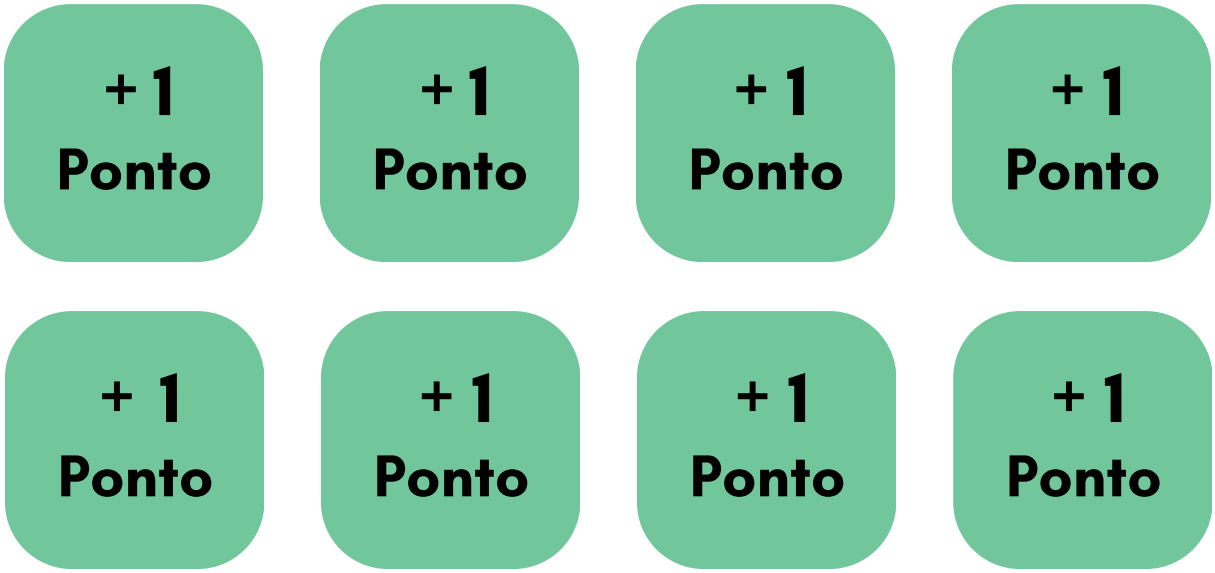


Figura 15 – Cartas ponto A

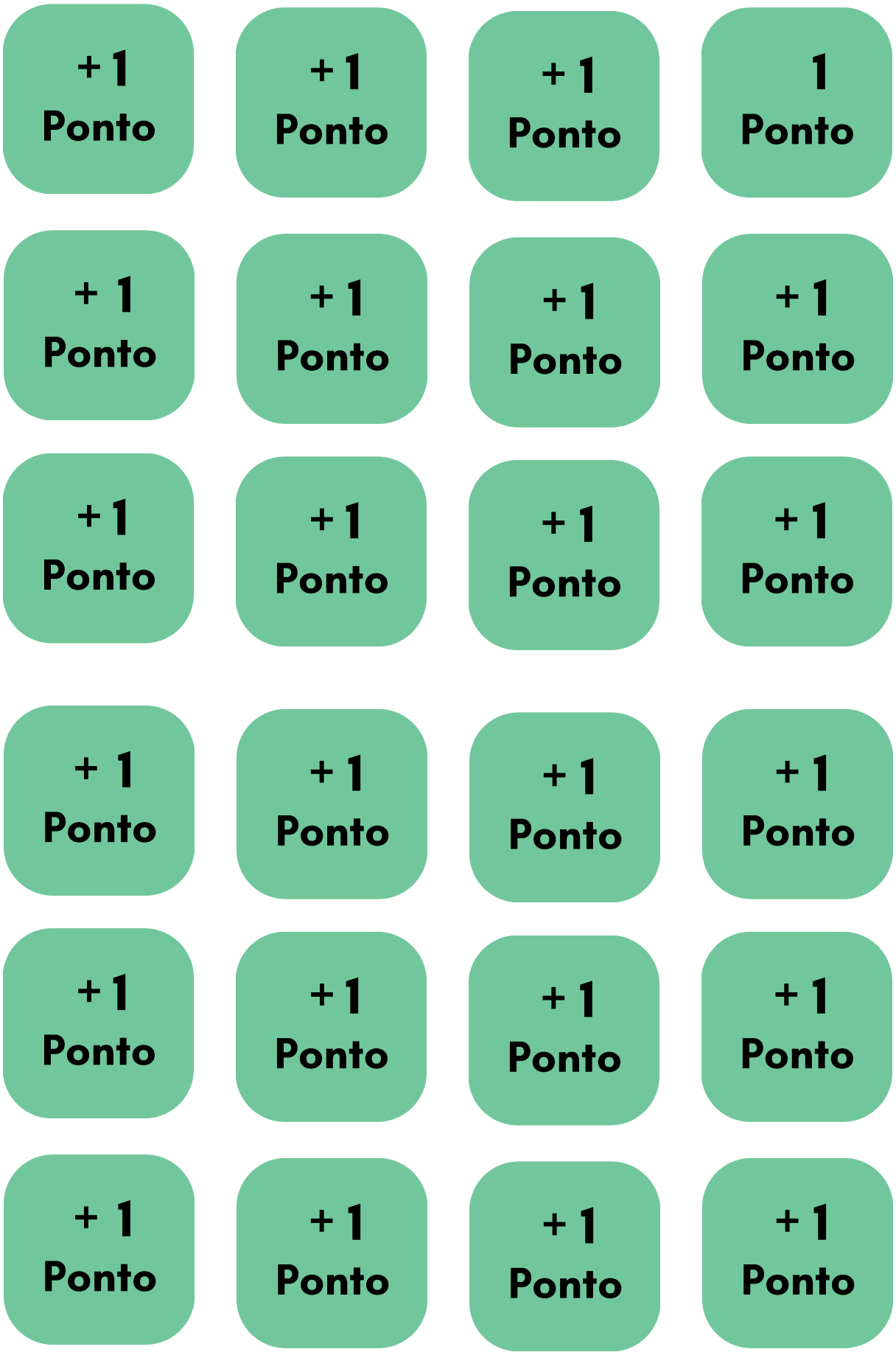


Figura 16 – Cartas ponto B

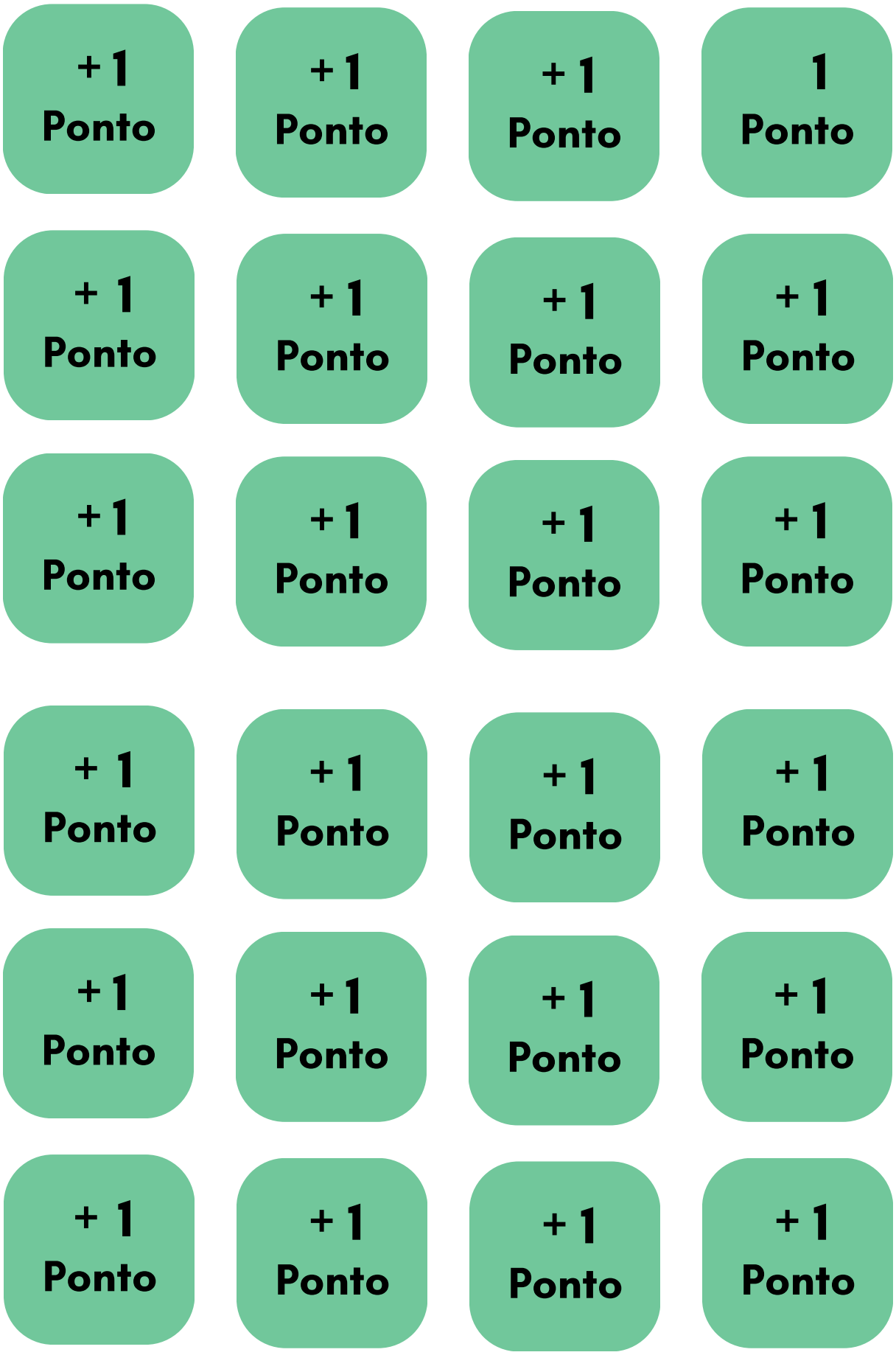


Figura 17 – Cartas ponto C

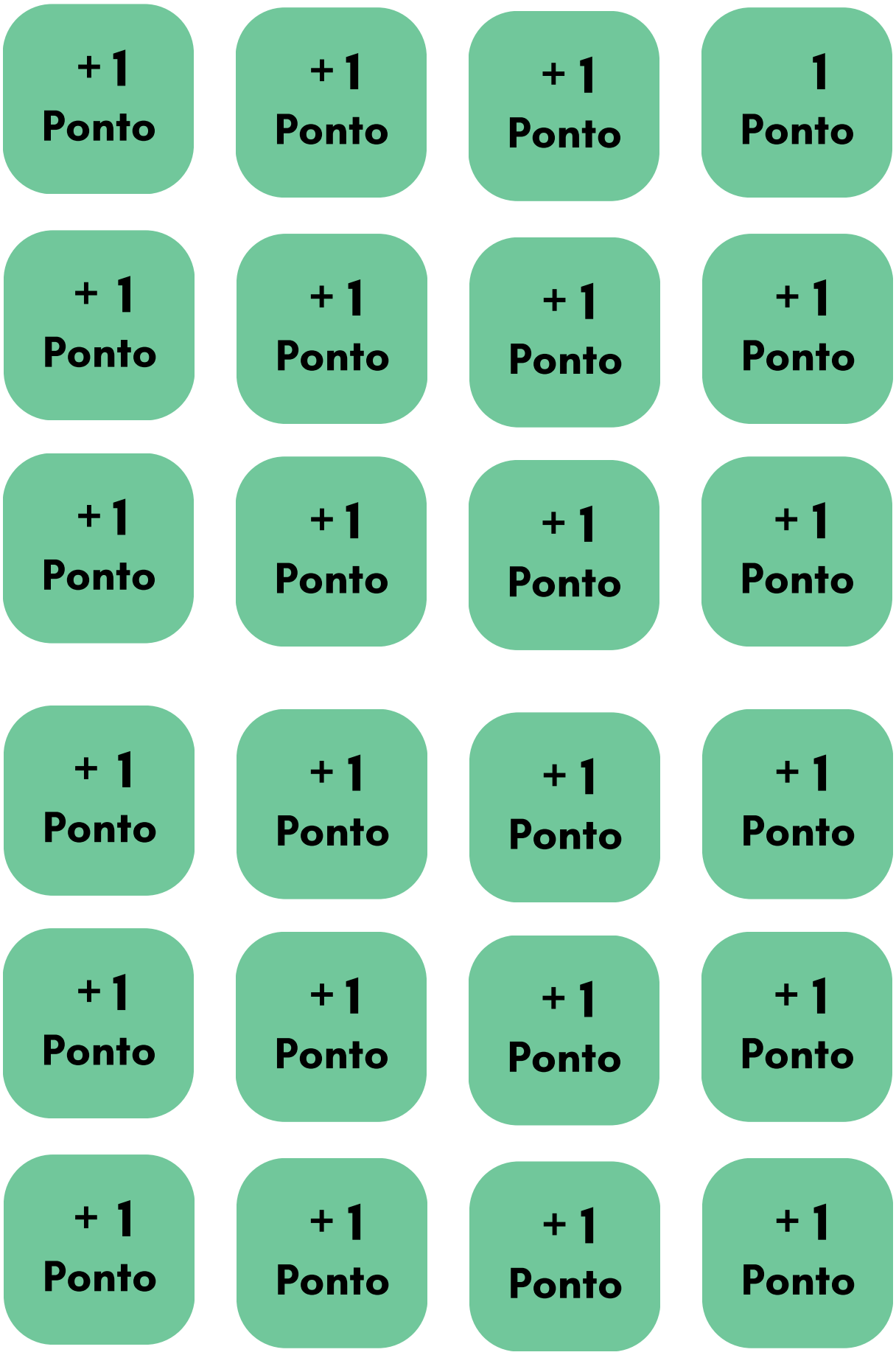


Figura 18 – Cartas ponto D

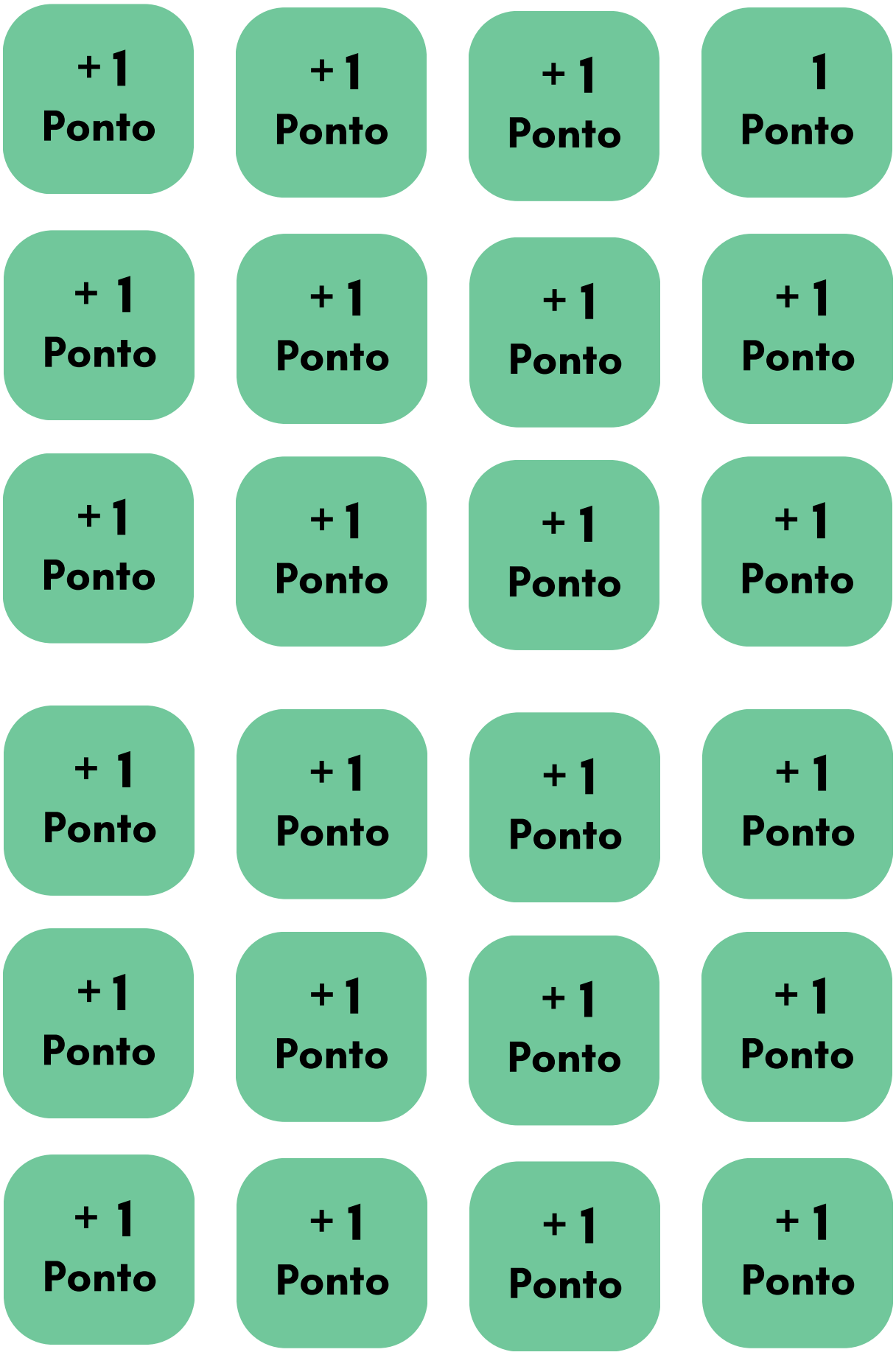


Figura 19 – Cartas ponto E

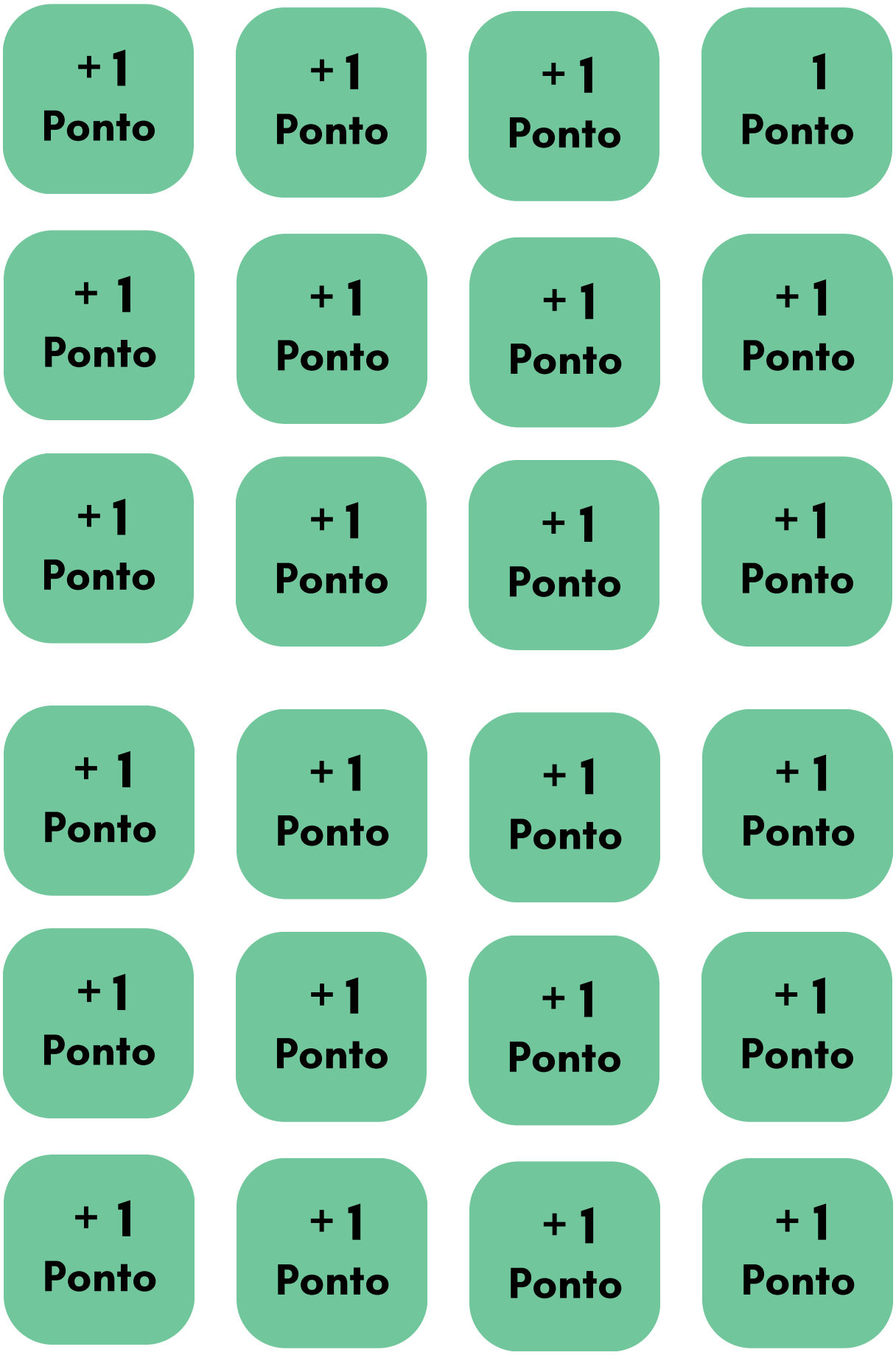


Figura 20 – Cartas ponto F

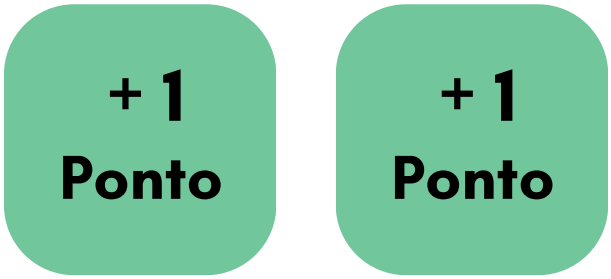


Figura 21 – Cartas ponto G