UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

PEDRO HENRIQUE SARTORI URNAU

ANÁLISE DAS CONTRIBUIÇÕES DO BDD NO DESENVOLVIMENTO DE UM JOGO SÉRIO

PONTA GROSSA

2024

PEDRO HENRIQUE SARTORI URNAU

ANÁLISE DAS CONTRIBUIÇÕES DO BDD NO DESENVOLVIMENTO DE UM JOGO SÉRIO

Analysys of BDD contribuitions in the development of a serious game

Trabalho de conclusão de curso de graduação apresentado como requisito para obtenção do título de Bacharel em Ciência da Computação da Universidade Tecnológica Federal do Paraná (UTFPR).
Orientador(a): Prof.ª Dr.ª Simone Nasser Matos.

PONTA GROSSA

2024

PEDRO HENRIQUE SARTORI URNAU

ANÁLISE DAS CONTRIBUIÇÕES DO BDD NO DESENVOLVIMENTO DE UM JOGO SÉRIO

Trabalho de conclusão de curso de graduação apresentado como requisito para obtenção do título de Bacharel em Ciência da Computação da Universidade Tecnológica Federal do Paraná (UTFPR).

Data de aprovação: 13/maio/2024

_____

Simone Nasser Matos
Doutorado
Universidade Tecnológica Federal do Paraná

_____

Simone Bello Kaminski Aires
Doutorado
Universidade Tecnológica Federal do Paraná

_____

Tarcízio Alexandre Bini
Doutorado
Universidade Tecnológica Federal do Paraná

PONTA GROSSA

2024

AGRADECIMENTOS

Agradeço a Deus pela oportunidade honrosa de entrega e dedicação no meu propósito e à minha família, pelos momentos de ausência, por terem suportado o processo junto comigo e confiarem no meu potencial.

À professora Nasser, expresso minha sincera gratidão por sua orientação e ensinamentos.

RESUMO

Behavior-Driven Development, nomeado pela sigla BDD, permite que sejam escritos cenários de teste em uma fase anterior a definição do código. O cenário é escrito em três blocos (*given*, *when*, *then*). Existe uma escassez de trabalho na literatura que usam esse tipo de desenvolvimento para criar jogos educativos ou sérios. Este trabalho realizou uma análise da aplicação do BDD na criação de um jogo sério do tipo Quiz para avaliar suas contribuições. O processo metodológico envolveu etapas de criação do jogo e de aplicação do BDD. As etapas contempladas foram: elaboração dos requisitos, definição dos cenários, implementação do jogo, especificação executável, validação e automação dos testes e refatoração orientada por comportamento. Como resultado verificou-se que o BDD ajudou na validação das funcionalidades do jogo que foram identificadas em etapas iniciais do desenvolvimento, oportunizando maior legibilidade do código, facilitando as futuras atualizações e manutenção do jogo. Além disso, o processo pode ser reusado por pessoas que desenvolvem jogos sérios e que desejam fundamentar seu desenvolvimento em testes.

Palavras-chave: Behavior-Driven Development; BDD; jogos sérios, avaliação.

ABSTRACT

Behavior-Driven Development, named by the acronym BDD, allows test scenarios to be written in a phase prior to code definition. The scenario is written in three blocks (given, when, then). There is a scarcity of work in the literature that uses this type of development to create educational or serious games. Therefore, this work carried out an analysis of the application of BDD in the creation of a serious Quiz-type game to evaluate its contributions. The methodological process involved stages of creating the game and applying BDD. The steps covered are: requirements development, scenario definition, game implementation, executable specification, test validation and automation, and behavior-oriented refactoring. As a result, it was found that BDD helped to validate the game's features that were identified in the initial stages of development, providing greater readability of the code, facilitating future updates and maintenance of the game. Furthermore, the process can be reused by people who develop serious games and who want to base their development on testing.

Keywords: Behavior-Driven Development; BDD; serious game, assessment.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS

BDD	*Behavior-Driven Development*
CI	*Continuous Integration*
DSL	*Domain Specific Language*
HTML	*Hypertext Markup Language*
IDE	*Integrated Development Environment JSON JavaScript Object Notation*
ODS	Objetivos de Desenvolvimento Sustentável
POJO	*Plain Old Java Objects*
QA'S	*Quality Assurance*
TDD	*Test Driven Development*

SUMÁRIO

1 INTRODUÇÃO

O *Behavior-Driven Development* (BDD), uma evolução do *Test Driven Development* (TDD), é uma metodologia de processo na qual as pessoas envolvidas em um produto trabalham para discuti-lo e criá-lo. No BDD as funcionalidades são projetadas e incluem uma visão geral do projeto e metas onde a equipe se reúne para definir os cenários, que posteriormente serão implementados (ANDERLE, 2023).

Existem cerca de 40 *frameworks* de automação de testes disponíveis para aplicação do *Behavior Driven Development*, sendo um deles o *JBehave* (2023), que, assim como o *Cucumber* (2023), permite fácil comunicação entre a equipe de desenvolvimento e a equipe de negócios. A utilização dessas ferramentas consiste na implementação de uma classe *Plain Old Java Objects* (POJO), que é um *design* simplificado, com anotações para criar uma correspondência entre o texto do projeto e os métodos da classe (ROJAS, 2021). São criadas histórias (arquivo de texto com extensão, história) e implementadas para que sejam executadas por meio de um *framework* de testes (ANDERLE, 2023).

A adoção do BDD proporciona vantagens significativas, incluindo melhorias na comunicação, documentação dinâmica e uma visão abrangente do projeto. Além disso, possibilita o compartilhamento de conhecimento e o desenvolvimento de documentação de forma mais dinâmica (ANDERLE, 2023).

Ao utilizar o BDD, os profissionais podem criar cenários e realizar testes antes da finalização do projeto, identificando eventuais erros e desafios. Isso aprimora a comunicação da equipe, promovendo a colaboração e o compartilhamento de ideias (ANDERLE, 2023).

A adoção do BDD no desenvolvimento de jogos sérios traz benefícios ao promover interação e *feedback* contínuos, permitindo ajustes e melhorias ao longo do processo de criação do jogo. Além disso, o foco no valor do usuário final garante que as decisões de desenvolvimento estejam alinhadas com as necessidades e desejos dos jogadores, resultando em uma experiência de jogo mais gratificante e relevante (NASCIMENTO, 2023).

Pode-se destacar na literatura, trabalhos que aplicam o BDD no desenvolvimento de jogos sérios como o “Elaboração de um jogo sério sobre *Behavior Driven Development* para alunos de TI” (NASCIMENTO, 2023). Eles demonstram como essa abordagem pode ser aplicada de maneira eficaz para criar jogos

educacionais e de treinamento que atendam às necessidades específicas dos usuários e dos objetivos educacionais.

Outro trabalho é de Santos (2023), que afirma que ao utilizar o BDD, a equipe pode garantir que os recursos, a mecânica e os elementos do jogo são projetados com base nas necessidades e desejos dos jogadores. Isso significa que as decisões de desenvolvimento são orientadas pelo valor que fornecem aos jogadores, resultando em um jogo atraente e com maior probabilidade de atender às expectativas os e preferências do público-alvo.

Poucos são os trabalhos sobre BDD e jogos sérios na literatura, por isso, o presente trabalho aplicou o *Behavior-Driven Development* (BDD) na criação de um jogo Quiz, denominado *"QUIZBIIO*", para o ensino de biodiversidade. Segundo Smith e Jones (2020), a biodiversidade desempenha um papel fundamental na manutenção dos ecossistemas e na promoção da sustentabilidade ambiental. O tema biodiversidade foi escolhido porque atende os requisitos trabalhados por um projeto de extensão da Universidade Tecnológica Federal do Paraná na qual a orientadora desse trabalho é coordenadora.

O processo de criação do jogo incluiu etapas como elaboração de requisitos, definição de cenários, implementação do jogo, especificação executável, validação e automação de testes, e refatoração orientada por comportamento. Como resultado foi criado um processo que pode ser reutilizado por alunos que atuam no projeto de extensão Desenvolvimento de *Software* Educacional[1], na criação dos jogos para crianças ou pessoas com deficiência intelectual. Além disso, ao usar o BDD na criação do jogo *"QUIZBIIO"* conseguiu auxiliar na validação das funcionalidades, melhorar a legibilidade do código e facilitar o processo de manutenção.

1.1 Justificativa

A era digital trouxe consigo novos paradigmas educacionais e os jogos sérios destacam-se como uma ferramenta inovadora capaz de transcender os limites tradicionais da aprendizagem. Este segmento introdutório apresenta a relevância do tema, destacando a necessidade de repensar abordagens pedagógicas para atender às demandas contemporâneas.

[1] https://sites.google.com/view/lesic-softwareeducacional

Ao se relacionar o BDD ao desenvolvimento de jogos sérios, pode-se encontrar benefícios mútuos, como a colaboração multidisciplinar; pois a criação de jogos requer a colaboração entre diferentes especialidades, como os desenvolvedores, *designers*, artistas e escritores de narrativa (SOARES, 2011).

A utilização do BDD pode facilitar a comunicação e colaboração entre as equipes, pois todos podem contribuir com a definição do comportamento do jogo. O BDD também promove interação e *feedback* contínuos.

1.2 Objetivos

O objetivo geral dessa pesquisa foi aplicar o BDD no desenvolvimento de um jogo sério a fim de analisar suas contribuições.

Para atingir o objetivo geral, os seguintes objetivos específicos foram identificados:

- Adaptação de um processo de desenvolvimento para criação de jogos sérios usando BDD;
- Criação de um jogo sério do tipo Quiz sobre a conservação da biodiversidade, tema relacionado aos objetivos do desenvolvimento sustentável (ODS).
- Implementar os cenários de BDD em um *framework*.

1.3 Organização do trabalho

Este estudo científico é constituído de 5 (cinco) capítulos. O Capítulo 1 introduz os principais conceitos adotados para o desenvolvimento do trabalho, seguido dos objetivos e justificativas.

A abordagem do Capítulo 2 discorre sobre a fundamentação teórica, abordando definições relacionadas aos princípios fundamentais do *Behavior-Driven Development* (BDD) e sua aplicação no desenvolvimento de *software*.

Já, o Capítulo 3 do trabalho, apresenta o processo metodológico usado para criação e aplicação do BDD em jogos sérios.

Em consonância aos demais, o Capítulo 4 detalha os resultados obtidos da aplicação do BDD na criação de um jogo sério. Por fim, o Capítulo 5 traz a conclusão dessa pesquisa e indicações de trabalhos futuros que podem dar continuidade aos estudos.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta os conceitos e definições utilizadas no desenvolvimento do trabalho proposto. A Seção 2.1 introduz o desenvolvimento guiado por comportamento, fornecendo uma definição dessa abordagem e discutindo sua importância no contexto do desenvolvimento de *software*. São apresentados conceitos-chave, como especificações executáveis, cenários de teste e etapas para implementar o BDD. A Seção 2.2 explora os jogos sérios, com a finalidade de definir e destacar seu papel como ferramenta educacional. São abordadas também as características dos jogos sérios, incluindo objetivos educacionais, estrutura de *feedback* e desafios pedagógicos. A Seção 2.3, por sua vez, examina a relação entre BDD e jogos sérios, demonstrando como o BDD pode ser aplicado de forma eficaz no desenvolvimento de jogos sérios para garantir sua qualidade e eficácia pedagógica. Por fim, a Seção 3.4 apresenta exemplos práticos de como o BDD pode ser implementado em projetos de jogos sérios, fornecendo *insights* para desenvolvedores e educadores interessados nessa abordagem.

2.1 Desenvolvimento guiado por comportamento

O *Behavior Driven Development* (BDD), ou desenvolvimento guiado por comportamento, é uma abordagem que auxilia as equipes de desenvolvimento de *software* a se concentrarem na identificação e compreensão das histórias de usuários com foco na comunicação e entendimento dos requisitos no projeto (ANDERLE, 2023).

O BDD foi apresentado por Dan North, em 2003, como uma evolução do *Test Driven Development* (TDD), incentivando a colaboração de todos os envolvidos no projeto, incluindo desenvolvedores, *Product Owner*, analistas de negócios e arquitetos de *software*.

A importância do BDD reside em seu papel na melhoria da comunicação e compreensão dos requisitos entre as equipes, bem como na redução do desperdício de esforço de desenvolvimento. Ele ajuda a evitar que equipes criem funcionalidades que não estão alinhadas com os objetivos de negócios do projeto, garantindo que o desenvolvimento se concentre em funcionalidades que agregam valor (ANDERLE, 2023).

Focado na compreensão do comportamento desejado do sistema, o BDD utiliza uma linguagem natural para criar especificações executáveis, transformando-as posteriormente em testes automatizados. Alguns dos benefícios do BDD incluem (ANDERLE, 2023):

- Redução do desperdício: BDD direciona o esforço de desenvolvimento para funcionalidades que agregam valor ao negócio, evitando o desperdício de esforço em recursos de pouco valor.
- Comunicação: BDD encoraja a colaboração entre analistas de negócios, desenvolvedores e testadores, permitindo que eles expressem os requisitos de forma testável e compreensível para todas as partes interessadas.
- Redução de custos: Ao focar no desenvolvimento de aplicativos alinhados com os objetivos de negócios, o BDD reduz o custo de entrega de um produto viável e o custo associado a erros e atrasos.
- Mudanças mais fáceis e seguras: BDD torna as mudanças e extensões de aplicativos mais fáceis, graças à documentação gerada a partir das especificações executáveis e aos testes de aceitação automatizados.
- Lançamentos mais rápidos: Testes automatizados abrangentes aceleram o ciclo de lançamento, permitindo que os testadores gastem menos tempo em testes manuais.

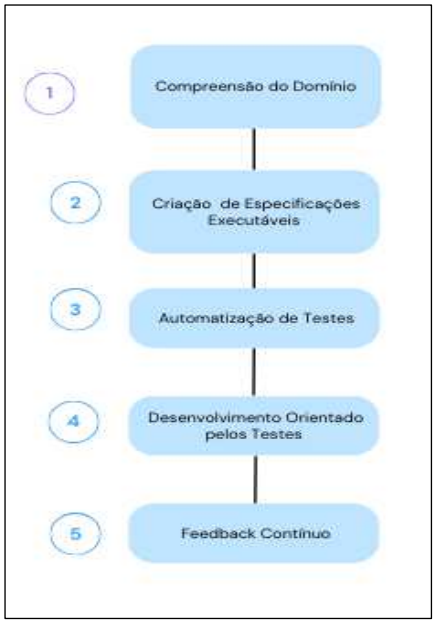
Além disso, o BDD promove uma linguagem única para análise e especificações, melhorando a compreensão e a comunicação entre equipes. Ele transforma os requisitos em comportamentos funcionais do *software,* o que ajuda na identificação do real valor a ser entregue ao cliente (NASCIMENTO, 2023).

Para implementar o BDD, histórias de usuários são escritas em formato de cenários que seguem o *template "Given* - Dado algum contexto inicial, *When* - Quando algum evento ocorrer, *Then* - Então verificar os resultados." Isso permite uma representação dos comportamentos esperados do *software* (NASCIMENTO, 2023). Uma das práticas centrais do BDD é escrever especificações executáveis em linguagem natural. Essas especificações são chamadas de cenários e seguem o formato "G*iven-When-Then*" (Dado-Quando-Então).

2.1.1 Etapas para o desenvolvimento do BDD

O desenvolvimento de BDD pode ser realizado em cinco etapas apresentadas na Figura 1.

Figura 1- Etapas para o desenvolvimento do BDD



Fonte: Autoria Própria (2024)

A compreensão do domínio (Etapa 1) envolve uma compreensão do sistema. O resultando resultado desta etapa é uma documentação sobre o sistema, estabelecendo um vocabulário comum para a equipe.

A criação de especificações executáveis (Etapa 2) é conduzida por meio da descrição do comportamento desejado em linguagem natural. Utilizando formatos como *Gherkin* (2024), são elaborados cenários de testes concretos, representando exemplos tangíveis de entrada e saída do sistema.

A automatização de testes (Etapa 3) visa a implementação dos códigos que validam os cenários descritos e integrados ao ciclo de integração contínua, garantindo uma validação regular e consistente do comportamento esperado.

No desenvolvimento orientado pelos testes (TDD) (Etapa 4), de forma interativa com a equipe, é criado o código-fonte para atender aos requisitos estabelecidos nos testes automatizados. O processo de TDD assegura que o código desenvolvido seja robusto, atendendo aos critérios de comportamento especificados.

Por fim, o *feedback* contínuo (Etapa 5) permite manter uma comunicação aberta, a equipe realiza revisões regulares dos resultados dos testes e do código,

promovendo correções rápidas de *bugs* ou ajustes. Essa abordagem proporciona um ciclo de desenvolvimento ágil e eficaz.

2.1.2 Ferramentas de BDD

Dentro do contexto do BDD, ferramentas específicas são frequentemente utilizadas para criar, executar e gerenciar testes de comportamento. Pode-se citar algumas como: *Cucumber* (2023)*, Setenity BDD* (2023)*, SpecFlow* (2023)*, Jasmine* (2023). No presente estudo, é detalhado duas ferramentas populares: *JBehave* e *Cucumber,* as quais são mais usadas para a implementação dos cenários em BDD na literatura.

O Cucumber Framework (2023) possibilita a implementação do BDD em projetos de testes ágeis e é utilizada por desenvolvedores e *QA's* (*Quality Assurance*) para a execução de testes automatizados. O *Cucumber* (2023) foi originalmente criado por membros da comunidade *Ruby* para apoiar o desenvolvimento de testes de aceitação automatizado utilizando a técnica BDD. Desde então, o *Cucumber* (2023) cresceu e foi traduzido em várias linguagens, inclusive o *Java*, permitindo assim, que vários desenvolvedores desfrutem de suas vantagens (SOARES, 2023).

O *framework Cucumber* se adapta para código aberto ou comercial e possibilita a implementação do BDD. Possui a versão paga chamada "*CucumberStudio*" e a versão *free* "*CucumberOpen*". A ferramenta faz uso da linguagem *Gherkin* (2023) e, sua implementação se dá por meio de *features*. Uma *feature* é uma descrição de um comportamento do sistema do ponto de vista do usuário. É uma forma de descrever as funcionalidades do *software* de uma maneira compreensível para o desenvolvedor.

No *Cucumber* (2023), as *features* são escritas na linguagem *Gherkin* (2023), que é uma linguagem de domínio específico (DSL) projetada para ser facilmente compreensível por todas as partes interessadas no desenvolvimento de software*.* Também é possível destacar a legibilidade dos s*teps* de testes, podendo estes serem marcados com *@tag*, como por exemplo, *@login* (SANTOS, 2012).

JBehave (2023) é uma biblioteca *Java* que facilita a implementação do desenvolvimento orientado por comportamento (BDD) em projetos de *software*. Ele fornece uma estrutura para expressar cenários de comportamento em linguagem natural e mapeá-los para testes automatizados escritos em *Java*. O *JBehave* (2023)

utiliza uma sintaxe semelhante à *Gherkin* (2023), uma linguagem específica de domínio (DSL) projetada para ser legível para pessoas. As histórias são escritas em formato de texto simples, usando palavras-chave como "*Given*" (Dado), "*When*" (Quando) e "*Then*" (Então) para descrever os passos do cenário.

Na ferramenta, cada história é representada por uma classe *Java* que estende *org.jbehave.core.steps.Story*. Essa classe contém os cenários e os passos associados a esses cenários. Os passos são implementados em métodos *Java*, além de fornecer suporte para a execução de testes de comportamento usando várias ferramentas de teste, como *JUnit* (SANTOS, 2012).

Os relatórios de execução podem ser gerados em formatos como console, *HTML, JSON*, entre outros e oferecer configurações flexíveis para adaptar a execução dos testes de acordo com as necessidades do projeto. Isso inclui a personalização de formatos de relatórios, configuração de passos globais, e outros aspectos relacionados à execução dos testes.

Em resumo, o *JBehave* (2023*)* é uma ferramenta robusta para a implementação de BDD em projetos Java, proporcionando uma estrutura para expressar e automatizar comportamentos do sistema. A abordagem baseada em *Gherkin* (2023) facilita a colaboração entre as equipes de desenvolvimento e de negócios, permitindo uma compreensão comum dos requisitos do software (SANTOS, 2012).

2.1.3 Especificação de cenários em BDD

A especificação dos comportamentos esperados do *software* é uma das etapas mais importantes do BDD. Isso é geralmente feito em colaboração com as partes interessadas, como os proprietários de produtos e os desenvolvedores. Durante a especificação, é importante criar uma linguagem comum que todos possam entender. Isso significa evitar jargões técnicos e usar uma linguagem natural acessível (ANDERLE, 2023).

Segundo North (2003), a sintaxe do BDD segue o formato "*Given-When-Then*" (Dado-Quando-Então), em que para criar cenários segue-se esse padrão:

- *Given* (Dado): Esta seção define o estado inicial ou contexto do cenário. Fornecendo as condições iniciais que devem ser atendidas antes que a ação ocorra. A palavra "Dado" será seguida por uma descrição do estado inicial.

- *When* (Quando): A seção "Quando" descreve a ação ou evento que está sendo testado. Isso representa a parte do cenário onde o sistema é estimulado de alguma forma. A palavra "Quando" será seguida por uma descrição da ação.
- *Then* (Então): A seção "Então" define o resultado esperado após a ação ser executada. Isso especifica o que o sistema deve fazer em resposta à ação. A palavra "Então" será seguida por uma descrição do resultado desejado.

O exemplo ilustrado no Quadro 1 representa um cenário em BDD, o qual foi escrito em linguagem acessível a todas as partes interessadas, incluindo desenvolvedores, testadores, analistas de negócios e partes não-técnicas. Eles ajudam a esclarecer os requisitos e a verificar se o software atende às expectativas.

Quadro 1- Saque em caixa eletrônico

Nome	**Exemplo saque de caixa**
Cenário	Saque em um caixa eletrônico com saldo suficiente dado que a conta tem saldo suficiente quando o cliente solicitar um saque, então o caixa eletrônico deve entregar o dinheiro.

Fonte: Adaptado de Anderle (2023).

O exemplo ilustrado no Quadro 2 representa uma aplicação do BDD baseando-se em técnicas como escrita de especificações executáveis, o uso de linguagem ubíqua, a integração de testes, a automação de testes de aceitação, o *feedback* contínuo e a refatoração orientada pelo comportamento. Essas práticas visam melhorar a comunicação, a qualidade e a agilidade no desenvolvimento de software, permitindo que as equipes entreguem um programa que atenda às expectativas do cliente.

Quadro 2- Reserva de quadros de hotel

Nome	**Exemplo reserva de quartos de hotel**
Cenário	O cliente entra no sistema para efetuar uma reserva de quarto de hotel *deluxe*, busca a data desejada e então o sistema confirma a reserva e mostra o preço total.

Fonte: Adaptado de Anderle (2023).

No entanto, se o código precisa ser refatorado, a equipe primeiro deve garantir que os cenários de testes continuem passando antes de realizar as alterações. Isso ajuda a manter a integridade do software.

2.2 Jogos sérios

Jogos sérios são usados para finalidade de aprendizagem e de treinamento de pessoas, tais como, nas áreas de meio ambiente, saúde, defesa. Eles vêm sendo criados e utilizados em diversos contextos educacionais, porém, seu desenvolvimento é um processo complexo, com alto custo (de recursos humanos, materiais, financeiros, espaço e tempo (NASCIMENTO, 2023).

> É importante ressaltar: (1) características do produto final - que requer balanceamento de aspectos pedagógicos (conteúdo, avaliação, *feedback*) e de jogabilidade (desafio, controle, imersão); (2) inclusão de competências a serem ensinadas, treinadas e avaliadas; (3) integração de diferentes profissionais em sua construção; (4) sistematização e padronização de artefatos e processos em seu desenvolvimento; (5) reuso e extensão desses artefatos; (6) avaliação e (7) validação, tanto do aprendizado/treinamento quanto do jogo sério. Para oferecer suporte para a construção de jogos sérios, superando estes desafios, diferentes metodologias emergiram, além das existentes nas áreas inter-relacionadas (SANTOS, 2012).

Os Jogos Sérios são conceituados como aplicativos digitais cujo propósito principal é educar, treinar ou informar (ADERLENE, 2022). Suas características distintivas, inclui objetivos educacionais claros, estrutura de *feedback*, simulações realistas e desafios pedagógicos que fomentam a resolução de problemas. A imersão é examinada como um componente importante para a eficácia desses jogos como ferramentas educacionais.

A implementação bem-sucedida de jogos sérios exige uma abordagem multidisciplinar. Detalhes de implementação são fornecidos sobre ferramentas de desenvolvimento específicas, como *unity* e *unreal engine*, enfatizando a importância da colaboração entre educadores, *designers* e desenvolvedores (ADERLENE, 2022).

Pesquisas são revisadas para destacar como esses jogos melhoram a retenção de informações, desenvolvem habilidades cognitivas e motivam os alunos. A discussão sobre a interatividade proporcionada pelos jogos sérios abrange exemplos práticos de como essa abordagem contribui para a aplicação prática do conhecimento, resultando em aprendizado mais efetivo e duradouro (ADERLENE, 2022).

Os jogos sérios oferecem uma promissora perspectiva para a construção de um ambiente de aprendizado mais eficaz, envolvente e adaptável às necessidades dos alunos na era digital (SANTOS, 2012). O compromisso contínuo com a pesquisa

e a prática aprimora ainda mais o papel desses jogos como ferramentas valiosas na promoção da educação inovadora e centrada no aluno.

Smith e Jones (2020), demonstram que, os jogos sérios, são ferramentas educacionais com o propósito principal de educar, treinar ou informar em diversas áreas, como meio ambiente, saúde e defesa. Embora seu potencial para transformar a aprendizagem seja reconhecido, o desenvolvimento desses jogos é um processo complexo e dispendioso, envolvendo recursos humanos, materiais, financeiros e tempo significativos. Dessa forma, para garantir sua eficácia, deve-se considerar características específicas do produto final, incluindo o equilíbrio entre aspectos pedagógicos, de jogabilidade e, a integração de competências, a serem ensinadas e avaliadas, bem como, a necessidade de envolvimento de diferentes profissionais em sua construção (SMITH; JONES, 2020).

Os jogos sérios desempenham um papel significativo na criação de conscientização e educação sobre tópicos ambientais complicados envolvendo biodiversidade (SMITH; JONES, 2020). Esses jogos não apenas oferecem entretenimento, mas também promovem discussões sobre conservação ambiental. Eles podem ser adaptados para diferentes públicos, desde crianças até formuladores de políticas, ampliando o alcance do conhecimento sobre biodiversidade e a compreensão dos desafios enfrentados pelos ecossistemas. Colaborações entre desenvolvedores de jogos e cientistas permitem a integração de dados reais e cenários autênticos, desafiando os jogadores a resolverem problemas complexos relacionados as mudanças climáticas, fragmentação de *habitats* e espécies invasoras.

Jogos como "*Fate of the World*" (SMITH; JONES, 2020), oferecem uma visão holística das interações entre humanos e meio ambiente, incentivando os jogadores a considerarem as consequências de suas escolhas.

Em resumo, os jogos sérios sobre biodiversidade desempenham um papel fundamental na educação, pesquisa e advocacia ambiental, contribuindo para o avanço do conhecimento científico e o desenvolvimento de políticas de conservação mais eficazes.

2.3 Trabalhos relacionados

A seguir são descritos os trabalhos similares ao proposto nesse trabalho que foram encontrados na literatura: "*BDD Assemble": A Paper-Based Game Proposal for Behavior Driven Development Design Learning* (SARINHO, 2019); Elaboração de um jogo sério sobre *Behavior Driven Development* para alunos de TI (NASCIMENTO, 2022).

O trabalho de Sarinho (2019) propõe um jogo de papel para auxiliar no aprendizado do *design* de desenvolvimento orientado a comportamento (BDD). O jogo foi desenvolvido como uma ferramenta para facilitar o entendimento e a aplicação dos conceitos de BDD, tornando o aprendizado mais envolvente e interativo. O jogo utiliza cartas e tabuleiros para simular situações reais de desenvolvimento de software, permitindo que os jogadores pratiquem a criação de cenários de teste, escrita de especificações e outras atividades associadas ao BDD. O objetivo é proporcionar uma abordagem prática e divertida para o ensino e aprendizado do BDD, promovendo uma compreensão mais profunda e eficaz dos conceitos envolvidos.

O estudo realizado por Nascimento (2022), propôs um método inovador para o ensino do desenvolvimento de jogos, centrado na metodologia de desenvolvimento orientado a comportamento (BDD). A estratégia principal delineada neste estudo envolveu a utilização de linguagem natural para a definição de requisitos de jogo, facilitando sua compreensão por parte dos envolvidos no processo de desenvolvimento, sejam eles alunos, professores ou desenvolvedores. Estes requisitos foram traduzidos em cenários de teste executáveis, possibilitando a verificação automatizada do comportamento do jogo.

Esse enfoque não apenas simplificou o aprendizado do desenvolvimento de jogos, mas também promoveu uma implementação mais precisa e dirigida, uma vez que os comportamentos esperados estavam delineados e passíveis de verificação contínua. Dessa forma, o método proporcionou uma compreensão mais clara dos requisitos do jogo e permitiu que os desenvolvedores se interessassem sobre o processo de desenvolvimento de forma mais eficiente, resultando em jogos que melhor atendiam às expectativas e necessidades dos usuários.

A implementação dessa abordagem multifacetada envolveu a definição detalhada de comportamentos esperados do jogo utilizando BDD, o que permitiu uma compreensão precisa e concisa dos requisitos educacionais a serem cumpridos. Além

disso, a integração contínua foi empregada para automatizar o processo de teste e verificação, garantindo que o código fosse integrado e testado repetidamente ao longo do ciclo de desenvolvimento. Isso permitiu uma detecção precoce de problemas e uma rápida interação sobre o software, resultando em uma entrega mais eficiente de funcionalidades educacionais de alta qualidade.

No tocante a colaboração interdisciplinar, desempenhou um papel importante, facilitando a comunicação entre diferentes equipes e garantindo que os objetivos educacionais fossem adequadamente traduzidos em funcionalidades de jogo. Assim, esse estudo demonstrou de forma abrangente os benefícios tangíveis de adotar uma abordagem BDD no desenvolvimento de jogos educacionais, em termos de qualidade do produto final considerando eficiência do processo de desenvolvimento.

O ciclo de desenvolvimento interativo é mostrado como a abordagem BDD permite ajustes contínuos com base no *feedback* dos educadores e resultados dos testes. Essa flexibilidade é fundamental para garantir a adaptação do jogo às necessidades em constante evolução dos alunos.

3 PROCESSO PARA O DESENVOLVIMENTO DO JOGO FUNDAMENTADO EM BDD

Este capítulo apresenta o processo de desenvolvimento de jogo sério que foi usado neste estudo de pesquisa. A Seção 3.1 discorre sobre a visão geral do processo. Na Seção 3.2, "elaboração dos requisitos", é detalhada a análise minuciosa dos requisitos, incluindo a definição do tema, objetivos educacionais, conceitos a serem explorados e requisitos funcionais do jogo.

Em seguida, na Seção 3.3, "definição dos cenários", o foco é na elaboração de cenários que atendam à lista de interatividade gráfica, permitindo testar diferentes interações de jogadores, objetivos de jogo e *feedbacks* esperados.

A implementação do jogo é abordada na Seção 3.4, onde os requisitos e cenários previamente identificados são utilizados para criar os elementos fundamentais do jogo e a interface gráfica, incluindo perguntas, opções de resposta, botões de interação e elementos visuais para *feedback*.

A Seção 3.5, "especificação executável", visou garantir que o desenvolvimento do jogo siga uma abordagem orientada pelo comportamento esperado, promovendo uma compreensão compartilhada e a implementação dos requisitos.

A validação e automação dos testes são discutidas na seção 3.6, destacando a importância de identificar os diferentes cenários de teste e os resultados esperados de cada interação do usuário com o jogo.

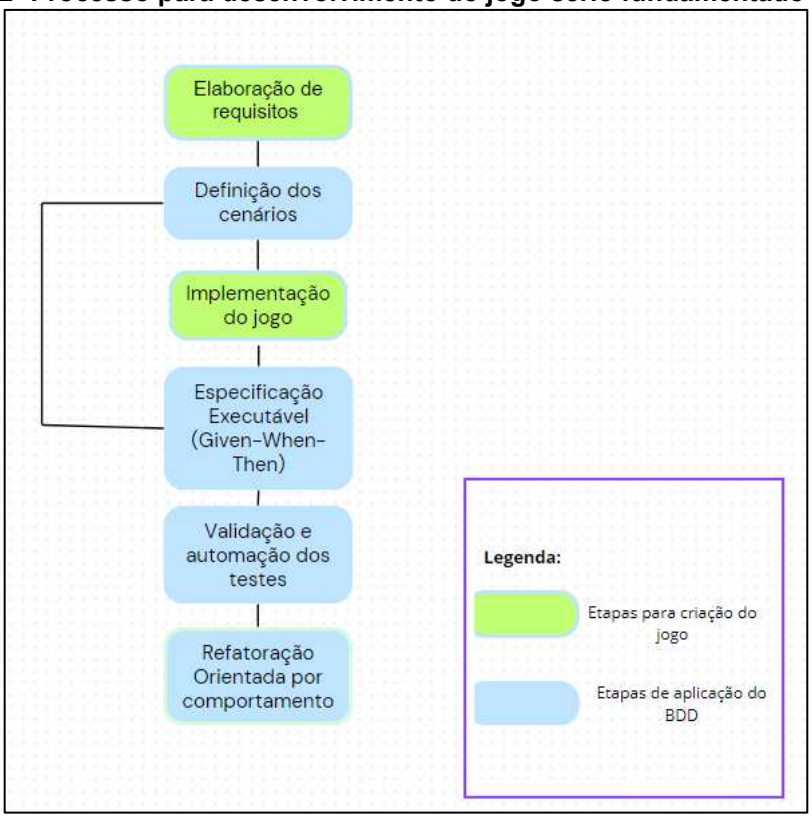
Por fim, a Seção 3.7, "refatoração orientada por comportamento", descreve a prática de melhoria contínua do código-fonte com base nos resultados dos testes de aceitação.

3.1 Visão geral do processo

O presente estudo apresenta como as etapas do desenvolvimento do jogo foram conduzidas em conjunto com a aplicação do desenvolvimento dirigido por comportamento (BDD). Ao adotar o *Behavior Driven Development* (BDD), o processo de criação do jogo é enriquecido com uma abordagem orientada à ação devido a criação de cenários. Isso permite uma interação contínua que promove a evolução constante do produto.

O processo utilizado para o desenvolvimento do jogo foi concebido em duas etapas: criação do jogo e a aplicação do BDD, conforme ilustra a Figura 2.

Figura 2- Processo para desenvolvimento do jogo sério fundamentado em BDD



Fonte: Adaptado de Anderle (2023).

Na etapa de elaboração de requisitos foi realizada uma análise detalhada dos requisitos do jogo sério. Para a definição dos cenários foram usados como base, os requisitos levantados na etapa anterior.

Na etapa de implementação do jogo, o jogo sério foi desenvolvido de acordo com os requisitos e cenários definidos nas etapas anteriores. Na etapa de especificação executável (*Given-When-Then*) foram criados os cenários em linguagem natural, seguindo o formato *Given-When-Then* do BDD.

Na etapa de validação e automação dos testes, os testes de aceitação foram automatizados para verificar se o jogo se comporta conforme especificado nas especificações executáveis.

Por fim, na etapa de refatoração orientada por comportamento, foi realizada para melhorar a qualidade do código-fonte do jogo. A seguir é descrito detalhadamente cada etapa do processo usado para a criação do jogo e análise do uso da abordagem BDD em seu desenvolvimento.

3.2 Elaboração dos requisitos

Na presente etapa, o desenvolvimento do jogo sério envolveu uma análise detalhada dos requisitos, contendo a definição do tema do jogo, os objetivos educacionais, os conceitos a serem explorados e levantamento dos requisitos funcionais.

A definição do tema foi estabelecida com base na conservação da biodiversidade (SMITH; JONES, 2020), pois é um tema que está sendo trabalhado pelo projeto de extensão de Letramento Digital[2] da Universidade Tecnológica Federal do Paraná. Tocante ao objetivo educacional, pode-se mencionar importância na educação dos jogadores sobre a biodiversidade e suas ramificações, uma vez que, os conceitos explorados no jogo, são: biodiversidade, o ecossistema e demais áreas de conservação.

Nesta etapa, foram definidos os requisitos funcionais como: jogar, receber *feedback*, emitir som por meio de um sintetizador, projetar dicas, exibir resultados e calcular o *ranking*. Ressalta-se que o tipo de jogo estabelecido foi do tipo *Quiz* (perguntas e respostas). Não foram criadas fases no jogo, pois o objetivo principal do presente estudo foi analisar as contribuições do BDD na criação de um jogo.

Com base nos dados anteriores e a interação que o usuário terá com o jogo, foram identificadas algumas funcionalidades de interatividade para proporcionar uma experiência de jogo educativa e envolvente, visando não apenas entreter, mas também educar os jogadores, conforme apresenta o Quadro 3.

Quadro 3- Interatividade gráfica para o jogador

Interatividade gráfica	• Três botões na home *page,* de “Jogar”; • “Configuração do som”; • “Ajuda”; • Sintetizador de voz; • Definição das perguntas / cada pergunta contendo alternativas e uma dica; • *Ranking* para mostrar a competência nas respostas do usuário; • Fundo envolvendo biodiversidade.

Fonte: Autoria própria (2024).

Os protótipos da tela foram definidos também nessa etapa, definidos tanto da *home page,* as opções de configurações do som, ajuda, amostra das perguntas e o

[2] https://sites.google.com/view/lesicpg/pagina-inicial

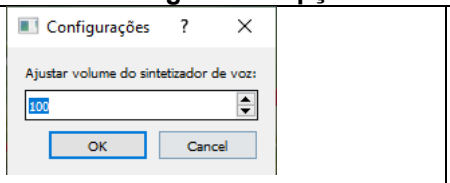
ranking final. A *home page* possui uma chamada e uma introdução do jogo, juntamente com as três opções de “jogar”, “configurações” e “ajuda” (Figura 3).

Figura 3- *home page* do jogo *QUIZBIIO*

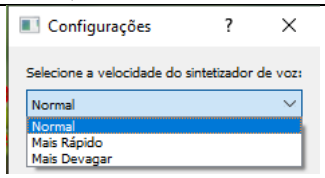
Fonte: Autoria própria (2024).

A aba de configurações (Figuras 4 (a) e (b)) permite alterar volume e velocidade do sintetizador de voz.

Figura 4 - Opções de configuração do *QUIZBIIO*



a. Ajuste de volume

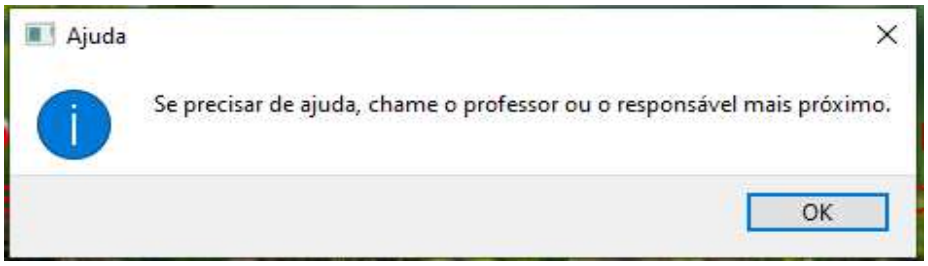


b. Velocidade do sintetizador

Fonte: Autoria própria (2024).

A aba de ajuda (Figura 5), caso o aluno não saiba como proceder durante o jogo, é exibida uma mensagem informando o que ele deve fazer.

Figura 5- Aba de ajuda



Fonte: Autoria própria (2024).

A interface das perguntas é apresentada na Figura 6. As perguntas foram criadas usando os materiais disponíveis na internet sobre biodiversidade (SMITH; JONES, 2020). As perguntas foram validadas por uma professora da instituição parceira dos projetos de extensão na qual a orientadora executa suas atividades. A relação das perguntas está apresentada na Seção 4.1.

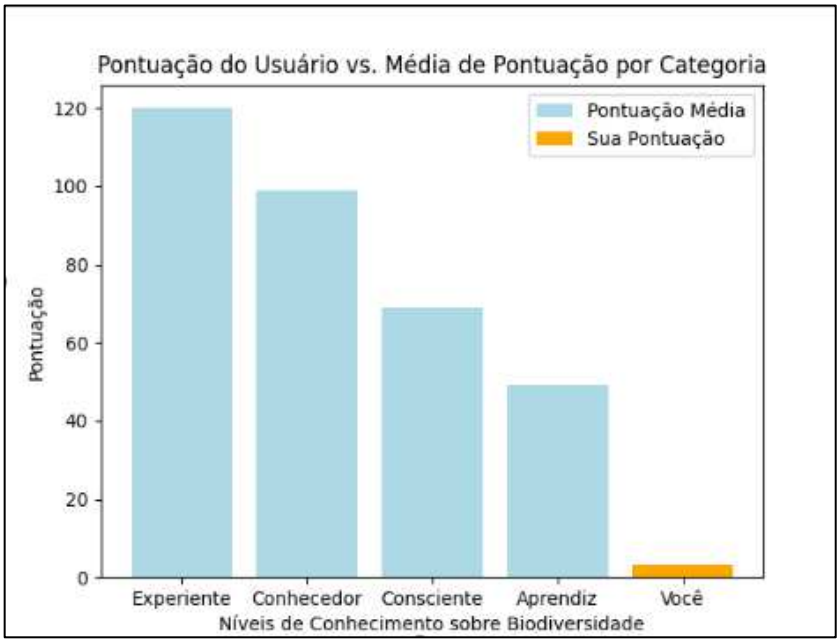
Figura 6 - Exemplo de uma pergunta do *QUIZBIIO*

Fonte: Autoria própria (2024).

E, por fim, o *ranking* final exibe como foi o desempenho do jogador durante o jogo (Figura 7), sendo que o *ranking* funciona da seguinte maneira: quando o usuário escolhe uma resposta, o código verifica se está certa ou errada. Se acertar de

primeira, o jogador ganha mais pontos, mas se errar, recebe uma dica na primeira tentativa. Se não acertar mesmo assim, passa para a próxima pergunta. No final, o código mostra um *ranking* que classifica o jogador de acordo com sua pontuação total. Se o jogador for bem, pode ser classificado como "experiente da biodiversidade", mas se não for tão bem, pode ser chamado de "aprendiz". Assim, comparando o desempenho dos gráficos no gráfico.

Figura 7 - Exemplo de um *ranking* do jogador



Fonte: Autoria própria (2024).

3.3 Definição dos cenários

Esta etapa contribuiu para que o jogo atendesse a lista de interatividade gráfica, fornecendo uma estrutura para testar diferentes interações dos jogadores, objetivos de jogo e *feedbacks* esperados. Nesta etapa o foco foi a definição dos cenários, desenvolvidos usando a lista de interação (Quadro 3).

Para os cenários de interatividade do jogador, pode-se incluir situações em que interage com a interface do jogo, como ao clicar no botão “jogar”, “configurações de áudio” ou “ajuda”. No qual o jogo implementa as respostas a essas interações, como ocultar a página inicial ao clicar em “jogar” e exibir configurações ou até mesmo a própria mensagem de ajuda. O resultado dessa etapa é detalhado na seção 4.1.2 do capítulo de resultados.

3.4 Implementação do jogo

Esta etapa foi realizada com base nos requisitos e cenários identificados em etapas anteriores, tendo como base a criação dos elementos fundamentais do jogo, assim como, a implementação da interface do jogo para apresentar as perguntas e opções de respostas, botões para interação e elementos visuais para fornecer *feedback.*

As mecânicas de jogo definem as regras e interações que governam a experiência do jogador. No código, as mecânicas incluem a apresentação de perguntas, verificação de respostas, atribuição de pontuações com base na precisão da resposta e avanço para a próxima pergunta ou exibição do *ranking* final.

Além da definição das interfaces gráficas, sistemas de pontuação e *feedbacks* visuais e sonoros, a implementação visou garantir que ele seja funcional e atenda aos objetivos pedagógicos estabelecidos, oferecendo uma experiência de jogo educativo para os jogadores.

A implementação foi realizada utilizando a linguagem *Python* (2024) como linguagem de programação e a *IDE PyCharm* (2024) para o desenvolvimento. A escolha do *Python* como linguagem deu-se pela sua simplicidade e flexibilidade, tornando-a ideal para a criação de jogo educativo, como proposto neste estudo. O ambiente de desenvolvimento oferecido pelo *PyCharm* (2024) facilitou a escrita e organização do código, além de fornecer ferramentas úteis para a depuração e testes. No que se refere às bibliotecas, foram utilizadas *Sys*, *PyQt5*, *Matplotlib.pyplot*, *Pyttsx3* e *Threading*, conforme apresenta o Quadro 4.

Quadro 4 - Bibliotecas

Nomenclatura	Bibliotecas
Sys	Biblioteca padrão do *Python* para interação com o sistema operacional.
PyQt5	Biblioteca para desenvolvimento de aplicativos *GUI (Graphical User* Interface), permitindo criar interfaces gráficas de forma fácil e intuitiva.
Matplotlib.pyplot	Biblioteca para criação de gráficos e visualizações de dados em *Python*.
Pyttsx3	Biblioteca de texto que permite ao *Python* falar textos usando diferentes motores de síntese de voz.
Threading	Biblioteca para trabalhar com threads em *Python*, permitindo a execução de múltiplas tarefas simultaneamente.

Fonte: Autoria própria (2024).

3.5 Especificação executável

Esta etapa visou garantir que o desenvolvimento do jogo siga uma abordagem orientada pelo comportamento esperado, promovendo uma compreensão compartilhada e uma implementação precisa dos requisitos do jogo.

Na presente etapa, foram criadas especificações que descrevem o comportamento esperado do jogo em resposta às ações do jogador, seguindo o formato *Given-When-Then* do BDD.

Foi criado um *arquivo.features* usando o *framework* de testes de comportamento *Cucumber* (2023), que permite escrever testes em uma linguagem de domínio específico *Gherkin* (2024). Nesse caso, os cenários de testes foram descritos em formato legível e são apresentados na subseção 4.1.3 do capítulo de resultados.

3.6 Validação e automação dos testes

A validação e automação dos testes é a análise das especificações executáveis que são fornecidas para identificar os diferentes cenários de testes que precisam ser cobertos, cada qual, descrito em uma interação do usuário com o jogo e os resultados esperados por meio dessa interação.

Foram criadas para cada interatividade, os testes para avaliar a funcionalidade da aplicação. Cada teste, foi elaborado para abordar diferentes aspectos da interatividade do usuário com a aplicação. Por exemplo, os testes "test_responder_corretamente" e "test_responder_incorretamente" verificam se o sistema reage adequadamente às respostas corretas e incorretas dos usuários, respectivamente. Por sua vez, o teste "test_ajustar_configuracoes_volume" analisa se as configurações de volume são ajustadas de forma precisa.

Além disso, há testes que abrangem a exibição correta da página de boas-vindas e do quiz, a finalização do quiz e a exibição de mensagens de ajuda. Juntos, esses testes garantem a integridade e o bom funcionamento da aplicação, assegurando uma experiência interativa satisfatória para os usuários finais.

Na sequência, cada interação do usuário foi verificada para constatar se o jogo responde conforme o esperado, incluindo a verificação do conteúdo exibido na interface, o estado dos elementos e a interação com outros sistemas como o sintetizador de voz. Se o teste falhar, o problema foi anotado, indicando qual

expectativa falhou e fornecendo informações detalhadas sobre a falha. Assim, foi investigado o problema e corrigido, garantindo que o jogo funcione como o esperado.

Os testes automatizados no *framework pytest,* foram executados regularmente após cada modificação significativa no código, sendo os resultados analisados para garantir que todas as funcionalidades do jogo estejam funcionando corretamente e que não haja regressões ou novos problemas. Cada interação do usuário foi verificada para garantir a integridade e o bom funcionamento da aplicação.

Cada teste, começa com um *start* inicial do programa, onde cria-se uma instância do QUIZBIIO. Em seguida, utilizando o mecanismo *'qtbot'*, que simula as interações do usuário, verificando se os diferentes aspectos da interface gráfica funcionam corretamente.

3.7 Refatoração orientada por comportamento

A refatoração orientada por comportamento é uma prática do BDD na qual envolve uma melhoria contínua do código-fonte com base nos resultados dos testes de aceitação.

Durante o desenvolvimento do jogo, a refatoração foi realizada de acordo com a validação e automação dos testes, definida na etapa anterior. Identificaram-se oportunidades de melhoria na legibilidade, manutenibilidade e eficiência do código, sem alterar o comportamento percebido pelo usuário.

Dessa forma, a refatoração orientada por comportamento garantiu que o código do jogo permaneça limpo e organizado ao longo do tempo, facilitando futuras atualizações e expansões do sistema.

Ao final dessa etapa, foram implementadas melhorias no código do jogo, como identificação de diminuição da repetição de códigos, além de uma melhora significativa na lógica das classes, alinhada com a validação e automação dos testes estabelecidos anteriormente.

Após a conclusão, foi realizada uma análise a fim de garantir que as mudanças não introduzissem regressões ou falhas no sistema, permitindo verificar se as melhorias foram efetivas e se o jogo continuava conforme o esperado.

4 RESULTADOS

Este capítulo apresenta os resultados alcançados com a aplicação do BDD no desenvolvimento do jogo sério QUIZBIIO. A Seção 4.1 descreve os principais resultados referentes a cada etapa do processo, tais como: casos de testes, relação das perguntas, criação de cenários, criação dos testes e validação dos requisitos. Por fim, a Seção 4.2 apresenta a análise das contribuições do BDD no desenvolvimento de jogos sérios, tais como a especificação executável.

4.1 Aplicação do processo

Esta seção apresenta os resultados de cada uma das etapas do processo de desenvolvimento de jogos sérios com aplicação do BDD que foi descrito no capítulo anterior.

4.1.1 Definição dos requisitos

Para a análise dos requisitos, foram definidas funcionalidades, como a exibição de perguntas, seleção de respostas, apresentação de dicas, cálculo de pontuação e exibição dos resultados.

As perguntas baseadas na conservação da biodiversidade estão apresentadas no Quadro 5. Nesse quadro também é apresentado as alternativas/ respostas para cada pergunta, assim como, dicas de modo que fossem compreensíveis para alunos mais novo.

Quadro 5- Relação das perguntas / alternativas e respostas

Perguntas	Alternativas e Respostas
Por que devemos conservar a biodiversidade?	a) Proteger todas as espécies. b) Importante para cientistas. c) Fornecer serviços essenciais para humanos. d) Sem impacto na vida diária. Resposta Correta: c) Fornecer serviços essenciais para humanos. Dica: Pense nos benefícios que os seres humanos recebem da natureza.
O que é biodiversidade?	a) Dinheiro em reservas naturais. b) Variedade de organismos em um local. c) Capacidade de sobrevivência de uma espécie. d) Velocidade de reprodução de espécies. Resposta Correta: b) Variedade de organismos em um local. Dica**:** Considere a variedade de vida em um determinado lugar.
	a) Usar produtos descartáveis. b) Reduzir, reutilizar e reciclar. c) Descartar lixo em rios e oceanos.

Como podemos ajudar a conservar a biodiversidade?	d) Aumentar a poluição do ar. Resposta Correta**:** b) Reduzir, reutilizar e reciclar. Dica: Pense em práticas que diminuam o desperdício e o uso excessivo de recursos.
Qual a importância das áreas protegidas?	a) Apenas para atividades recreativas. b) Sem impacto na biodiversidade. c) Fornecem habitats seguros. d) Aumentam urbanização e destruição do habitat. Resposta Correta: c) Fornecem habitats seguros. Dica: Considere o papel dessas áreas na preservação de habitats naturais.
O que é um ecossistema?	a) Comunidade de espécies interagindo. b) Computador para modelagem ambiental. c) Método de produção agrícola. d) Grupo de cientistas estudando animais selvagens. Resposta Correta: a) Comunidade de espécies interagindo. Dica: Pense nas relações entre os seres vivos e seu ambiente físico.
O que significa "espécie em perigo crítico"?	a) Ameaça iminente de extinção. b) Abundante em sua área. c) Comum globalmente. d) Não importante para conservação. Resposta Correta: a) Ameaça iminente de extinção. Dica: Considere o nível de risco que uma espécie enfrenta.
O que é uma área de conservação?	a) Onde caça é permitida. b) Proibido qualquer atividade humana. c) Protege biodiversidade e ecossistemas. d) Destinada ao desenvolvimento urbano. Resposta Correta: c) Protege biodiversidade e ecossistemas Dica: Pense no propósito dessas áreas em relação à preservação da natureza.
Como proteger a biodiversidade aquática?	a) Aumentar poluição dos oceanos. b) Reduzir plásticos descartáveis. c) Aumentar pesca predatória. d) Ignorar conservação de recifes de coral. Resposta Correta: b) Reduzir plásticos descartáveis. Dica: Considere uma das principais fontes de poluição nos oceanos.
O que é uma espécie endêmica?	a) Ocorre em muitos lugares. b) Comum globalmente. c) Encontrada apenas em uma área d) Não ameaçada de extinção. Resposta Correta**:** c) Encontrada apenas em uma área Dica: Pense em uma espécie restrita a uma região específica.
Por que conservar a biodiversidade marinha?	a) Oceanos sem importância para humanos. b) Oceanos muito grandes para serem afetados. c) Fornecem alimentos, regulação do clima e oxigênio. d) Oceanos sem vida. Resposta Correta: c) Fornecem alimentos, regulação do clima e oxigênio. Dica: Considere os serviços vitais que os oceanos prestam ao nosso planeta.

Fonte: Autoria própria (2024).

4.1.2 Definição dos cenários

Para os cenários de interatividade com o jogador (listados no Quadro 3 do Capítulo 3), foram incluídas situações para que os jogadores interagissem com a interface do jogo ao clicar nos botões "Jogar", "Configurações de áudio" ou "Ajuda".

Em relação aos objetivos de cada fase, o código foi projetado a fim de garantir uma resposta corretamente para avançar para a próxima pergunta. A lógica implementada no jogo, gerencia a apresentação das perguntas e a verificação das respostas e na sequência. Assim como, informa os *feedbacks* para os jogadores em relação as suas ações. Caso o jogador não acerte, recebe uma dica para ajudá-los no aprendizado e na assertividade da questão. Os *feedbacks* são fundamentais para manter os jogadores engajados e motivados ao longo do jogo.

Nessa etapa foram definidos sete cenários para o jogo. O primeiro cenário, denominado de iniciar o jogo (Quadro 6), permite que os jogadores sejam recepcionados com uma página de boas-vindas apresentando o título QUIZBIIO e uma mensagem de boas-vindas, convidando-os a explorar o jogo sobre a conservação da biodiversidade. São apresentados os botões para jogar, acessar configurações ou obter ajuda, enquanto uma imagem de fundo cria uma atmosfera relacionada ao tema. Esse cenário inicializa a experiência do jogador, oferecendo opções de interação.

Quadro 6- Cenário 1 - Iniciar o jogo

Nome	**Início do jogo**
Cenário	Quando o usuário inicia o jogo, a página de boas-vindas é exibida e então o título "*QUIZBIIO*" é exibido centralizado e a mensagem "Seja bem-vindo! Pronto para mais conhecimento sobre a conservação da nossa biodiversidade?" será exibida e os botões "Jogar", "Configurações" e "Ajuda" são exibidos e uma imagem de fundo será exibida.

Fonte: Autoria própria (2024).

No segundo cenário jogar o quiz (Quadro 7), é acionado quando o usuário clica no botão "jogar", e nesse caso, a primeira pergunta é imediatamente apresentada. Em seguida, a pergunta é também pronunciada em voz alta para oferecer uma experiência multimodal aos jogadores. As opções de resposta são exibidas para que o jogador possa fazer sua escolha e, o botão "próxima pergunta", é disponibilizado para avançar para a próxima etapa do jogo. Esse cenário mostra uma transição fluida do início do jogo para a interação com as perguntas, incentivando os jogadores a participarem ativamente, enquanto são guiados pelo jogo.

Quadro 7 - Jogar o quiz

Nome	Jogar o quiz
Cenário	O usuário clica no botão "Jogar", a primeira pergunta é apresentada, na sequência a pergunta é falada em voz alta e as opções de resposta serão exibidas e o botão "próxima pergunta" será exibida.

Fonte: Autoria própria (2024).

No terceiro cenário de responder a uma pergunta corretamente (Quadro 8), quando a pergunta é apresentada e o usuário seleciona a resposta correta, a pontuação do jogador é imediatamente atualizada. Em seguida, o jogo avança para a próxima pergunta, mantendo o ritmo da experiência e recompensando o jogador pelo seu conhecimento e precisão. Essa dinâmica de atualização da pontuação e progressão contínua por meio das perguntas, incentiva os jogadores a permanecerem envolvidos e focados em avançar no jogo.

Quadro 8 - Responder a uma pergunta corretamente

Nome	Responder a uma pergunta corretamente
Cenário	Quando a pergunta é exibida e o usuário seleciona a resposta correta, a pontuação do usuário é atualizada e a próxima pergunta será exibida.

Fonte: Autoria própria (2024).

No quarto cenário, denominado de responder a uma pergunta incorretamente, ao exibir a pergunta e o usuário selecionar uma resposta incorreta, uma dica é apresentada para auxiliá-lo a tentar novamente (Quadro 9). Após cada tentativa incorreta, o jogador é encorajado a persistir e melhorar sua resposta. Se a última tentativa for também incorreta, o jogo avança para a próxima pergunta. Esse cenário, demonstra uma abordagem de aprendizado adaptativo, oferecendo *feedback* construtivo e oportunidades para o jogador aprender com seus erros antes de prosseguir no jogo.

Quadro 9 - Responder a uma pergunta incorretamente

Nome	Responder a uma pergunta incorretamente
Cenário	Quando a pergunta é exibida e o usuário seleciona a resposta incorreta, uma dica será exibida para que o usuário tente responder novamente. Após a última tentativa incorreta, a próxima pergunta será exibida.

Fonte: Autoria própria (2024).

No quinto cenário, finalizar o quiz (Quadro 10), quando o usuário responde a todas as perguntas, o jogo exibe o *ranking* do usuário com base na pontuação final alcançada. Além disso, um gráfico comparativo é apresentado, mostrando a

pontuação do usuário em relação à média. Essa visualização oferece ao jogador uma perspectiva de seu desempenho em comparação com outros jogadores ou uma referência de pontuação média, incentivando a competição saudável e o desejo de melhorar. Essa conclusão gratificante da experiência de jogo reforça o engajamento do jogador e fornece um senso de realização.

Quadro 10 - Finalizar o jogo quiz

Nome	Finalizar o quiz
Cenário	Quando o usuário responde todas as perguntas, o ranking do usuário será exibido com base na pontuação final e um gráfico comparativo será exibido mostrando a pontuação do usuário e a média.

Fonte: Autoria própria (2024).

No sexto cenário, ajustar configurações de voz (Quadro 11), quando o usuário clica no botão "configurações", é apresentada uma opção para ajustar o volume do sintetizador de voz ou selecionar uma velocidade para o sintetizador. Após o usuário fazer suas seleções, as configurações são aplicadas imediatamente. Isso permite que o jogador personalize a experiência de acordo com suas preferências auditivas, garantindo que o jogo seja acessível e confortável para cada usuário. Essa capacidade de ajuste das configurações demonstra a preocupação do desenvolvedor em oferecer uma experiência personalizada e adaptável aos diferentes usuários.

Quadro 11- Ajustar configurações de voz

Nome	Ajustar configurações de voz
Cenário	Quando o usuário clica no botão "configurações" o usuário ajusta o volume do sintetizador de voz ou seleciona uma velocidade para o sintetizador de voz, logo, as configurações são aplicadas.

Fonte: Autoria própria (2024).

No sétimo cenário, obter ajuda (Quadro 12), quando o usuário clicar no botão "ajuda", indica que está precisando de assistência. Nesse momento, uma mensagem informativa é exibida, instruindo o usuário sobre como obter ajuda. Essa mensagem, pode incluir orientações sobre onde encontrar recursos de suporte, como contatar o serviço de atendimento ao cliente ou acessar um guia de jogo. Essa abordagem garante que os jogadores tenham acesso rápido as informações úteis e suporte quando necessário, melhorando sua experiência de jogo e reduzindo possíveis frustrações.

Quadro 12 - Obter ajuda

Nome	Obter ajuda
Cenário	Quando o usuário clica no botão "ajuda", o usuário precisa de assistência e então uma mensagem informativa é exibida instruindo o usuário a procurar ajuda.

Fonte: Autoria própria (2024).

4.1.3 Especificação Executável

Na Especificação Executável, os cenários apresentados na seção anterior foram implementados. Cada cenário foi estruturado em *Given-When-Then*, delineando o estado inicial, a ação do usuário e o resultado esperado.

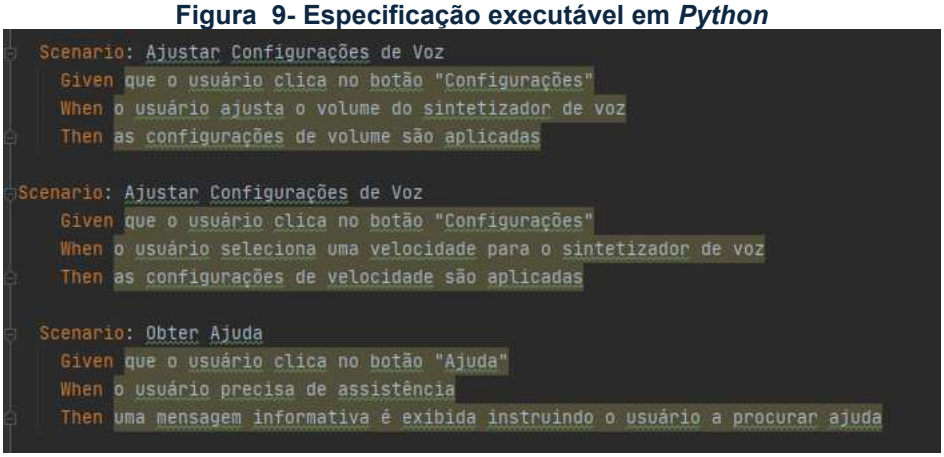
As Figuras 8 e 9 apresentam as implementações dos cenários que foram definidos na seção anterior. Note que neste caso, o cenário contém: o nome de cada cenário e a especificação executável de cada cenário.

Figura 8 - Especificação executável em *Python*

```
Feature: OrangeHRM Logo

  Scenario: Iniciar o Jogo
    Given que o usuário inicia o jogo
    When a página de boas-vindas é exibida
    Then o título "QUIZBIIO" é exibido centralizado
    And a mensagem "Seja bem-vindo! Pronto para mais conhecimento sobre a conservação da nossa biodiversidade?" é exibida
    And os botões "Jogar", "Configurações" e "Ajuda" são exibidos
    And uma imagem de fundo é exibida

  Scenario: Jogar o Quiz
    Given que o usuário clica no botão "Jogar"
    When a primeira pergunta é apresentada
    Then a pergunta é falada em voz alta
    And as opções de resposta são exibidas
    And o botão "Próxima Pergunta" é exibido

  Scenario: Responder a uma Pergunta Corretamente
    Given que uma pergunta está sendo exibida
    When o usuário seleciona a resposta correta
    Then a pontuação do usuário é atualizada
    And a próxima pergunta é exibida

  Scenario: Responder a uma Pergunta Incorretamente
    Given que uma pergunta está sendo exibida
    When o usuário seleciona uma resposta incorreta
    Then uma dica é exibida
    And o usuário pode tentar responder novamente
    And após a última tentativa incorreta, a próxima pergunta é exibida

  Scenario: Finalizar o Quiz
    Given que o usuário responde a todas as perguntas
    When a última pergunta é respondida
    Then o ranking do usuário é exibido com base na pontuação total
    And um gráfico comparativo é exibido mostrando a pontuação do usuário em relação à média
```

Fonte: Autoria própria (2024).

Figura 9- Especificação executável em *Python*



Fonte: Autoria própria (2024).

4.1.4 Validação e Automação dos Testes

Esse processo envolveu a análise das especificações executáveis fornecidas, identificando os cenários de teste que precisavam ser cobertos, cada qual descrito em uma interação específica do usuário com o jogo e os resultados esperados dessa interação.

A automação e validação dos testes foram realizadas utilizando o *pytest*, seguindo uma estrutura de projeto que incluiu um arquivo *test_main.py*. Os testes foram escritos de acordo com as especificações descritas, abrangendo cenários de interação do usuário e resultados esperados.

Os testes permitiram a detectação precoce dos erros, viabilizando assim, as correções antes que causassem problemas maiores. Além disso, ajudavam a documentar como o programa deveria se comportar em diferentes situações, contribuindo para a manutenção da qualidade do software e facilitando o desenvolvimento contínuo do jogo.

Foram feitos 6 testes de forma geral e 7 testes separados. No seguinte exemplo da Figura 10, o teste verifica se o usuário responde incorretamente a uma pergunta de um questionário. Ele cria uma instância do questionário, exibe-o e verifica se a pergunta atual é a primeira (índice 0) e se a pontuação é 0. Em seguida, simula uma resposta incorreta selecionando uma opção de resposta incorreta e clicando no botão "próximo". Após cada tentativa incorreta, verifica se o número de tentativas aumentou corretamente. Após a quarta tentativa incorreta, verifica se o quiz avançou para a próxima questão.

Figura 10 - *Def* da análise de respostas incorretas

```python
def test_responder_incorretamente(qtbot, app):
    """Testa se o usuário responde incorretamente a uma pergunta."""
    quiz = Quiz()
    qtbot.addWidget(quiz)
    quiz.show()

    assert quiz.current_question == 0
    assert quiz.score == 0

    # Seleciona uma resposta incorreta
    quiz.option_buttons[1].setChecked(True)
    quiz.next_button.click()

    assert quiz.attempts == 1  # Uma tentativa foi feita
    assert quiz.current_question == 0  # Continua na mesma pergunta

    quiz.next_button.click()  # Segunda tentativa incorreta
    assert quiz.attempts == 2

    quiz.next_button.click()  # Terceira tentativa incorreta
    assert quiz.attempts == 3

    quiz.next_button.click()  # Quarta tentativa incorreta, avança para a próxima pergunta
    assert quiz.current_question == 1
```

Fonte: Autoria própria (2024).

Os testes realizados, foram focados em testar a lógica interna da aplicação, como a interação entre os diferentes componentes do *PyQt5* e o comportamento do *QUIZBIIO*. Embora o BDD possa ser útil para especificar o comportamento esperado do sistema de uma forma mais legível e colaborativa, traduzir essas especificações diretamente em testes executáveis que pode não ser a abordagem mais prática ou eficaz, especialmente quando se trata de testes de unidades internas ou de integração no *Python*.

O resultado após fazer o *pytest* no terminal foi o ilustrado na Figura 11. No resultado é demonstrado que o teste foi aceito com sucesso, ao relatar no terminal que foram coletados 8 itens, significando que os 8 testes foram testados e concluídos com sucesso, como mostra a Figura 11.

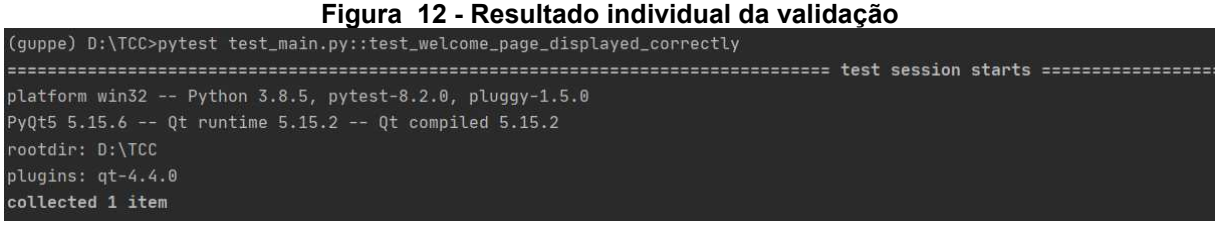
Figura 11 - Resultado geral da validação

```
============================= test session starts ==============================
platform win32 -- Python 3.8.5, pytest-8.2.0, pluggy-1.5.0
PyQt5 5.15.6 -- Qt runtime 5.15.2 -- Qt compiled 5.15.2
rootdir: D:\TCC
plugins: qt-4.4.0
collected 8 items
```

Fonte: Autoria própria (2024).

De forma isolada, rodando um *'def'* em específico, garantiu uma que cada passo do jogo esteja funcionando conforme deveria. Dessa forma, em um projeto de larga escala, poderia ser mais confiável fazer testes dessa maneira individual para garantir a funcionalidade do código. No exemplo da Figura 12, foi feito um teste que também foi executado com sucesso.

Figura 12 - Resultado individual da validação



Fonte: Autoria própria (2024).

Conforme realizado os testes, o desempenho do código 't*est_main'*, resultou na aprovação dos testes realizados sobre o QUIZBIIO.

4.1.5 Refatoração Orientada

Para a etapa de refatoração orientada por comportamento, é importante melhorar a qualidade do código-fonte do jogo, com base nos resultados dos testes de aceitação automatizados descritos na seção anterior.

Alguns pontos puderam ser melhorados tais como: i) clareza e organização do código, tendo a revisão de nomes de variáveis; ii) métodos e classes, agrupamentos de funcionalidades relacionadas em classes; iii) existência de códigos repetitivos; e, iv) consolidação dessas partes em funções ou métodos reutilizáveis, garantindo a legibilidade e manutenibilidade. Assim como, a adição de comentários e documentos onde for necessário e, a própria testabilidade do código.

A revisão da clareza e organização do código, incluiu a renomeação de variáveis, métodos e classes para melhorar a compreensão do código. Funcionalidades relacionadas foram agrupadas em classes para uma melhor organização. Além disso, foram identificados e consolidados trechos de código repetitivos em funções ou métodos reutilizáveis, visando aumentar a legibilidade e a manutenibilidade do código.

Após conferir as áreas de melhoria e realizar as refatorações, foram executados novamente os testes automatizados para garantir que as alterações não tenham introduziu novos problemas ou *bugs*, para que o comportamento geral do jogo

permaneça consistente conforme foi identificado anteriormente nos primeiros testes. Essa abordagem gradual de refatoração orientada por comportamento, visou garantir que o código do jogo esteja limpo, organizado e eficiente, facilitando futuras atualizações e manutenções.

A análise de desempenho dos códigos e as devidas melhorias, seguidos dos testes automatizados para compreender o funcionamento de como o BDD funciona na prática, foi alterada o código-fonte no *'def: main'* para utilizar a biblioteca *cProfile* para calcular o desempenho do código e salvar em um arquivo de texto.

A compilação do primeiro código realizou 204.678 chamadas de função em 3.183 segundos, indicando um número total de chamadas de função realizadas durante a execução do código no *benchmarking* realizado, como mostrado na figura 13.

Figura 13 - Bechamarking dos testes

```
         204678 function calls (204670 primitive calls) in 3.183 seconds

   Ordered by: internal time

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
     1961    1.385    0.001    1.385    0.001 {method 'acquire' of '_thread.lock' objects}
       16    0.732    0.046    0.732    0.046 {built-in method _thread.start_new_thread}
     3923    0.062    0.000    0.062    0.000 {method 'append' of 'list' objects}
     1961    0.060    0.000    0.060    0.000 {method 'release' of '_thread.lock' objects}
      937    0.041    0.000    0.041    0.000 {method 'acquire' of '_multiprocessing.SemLock' objects}
      523    0.040    0.000    0.040    0.000 {method 'write' of '_io.StringIO' objects}
     1961    0.027    0.000    0.027    0.000 {method 'remove' of 'collections.deque' objects}
     1961    0.022    0.000    0.022    0.000 {built-in method _imp.create_dynamic}
     1961    0.022    0.000    0.022    0.000 {built-in method builtins.exec}
     1961    0.021    0.000    0.021    0.000 {method 'format' of 'str' objects}
     1961    0.020    0.000    0.020    0.000 {method 'encode' of 'str' objects}
     1961    0.016    0.000    0.038    0.000 codecs.py:319(decode)
     1961    0.014    0.000    0.014    0.000 {built-in method _codecs.utf_8_decode}
     1176    0.013    0.000    0.013    0.000 {built-in method marshal.loads}
     1961    0.013    0.000    0.013    0.000 {built-in method builtins.isinstance}
     3921    0.013    0.000    0.013    0.000 {method 'close' of '_io.BytesIO' objects}
        1    0.011    0.011    0.027    0.027 {built-in method builtins.print}
     1961    0.010    0.000    0.010    0.000 {built-in method _imp.is_builtin}
     1961    0.009    0.000    0.009    0.000 {built-in method posix.fspath}
     3921    0.009    0.000    0.009    0.000 {built-in method builtins.hasattr}
     3921    0.009    0.000    0.009    0.000 {built-in method builtins.len}
     3921    0.009    0.000    0.009    0.000 {method 'getvalue' of '_io.BytesIO' objects}
     3923    0.008    0.000    0.008    0.000 {built-in method builtins.getattr}
      240    0.007    0.000    0.011    0.000 __init__.py:406(__getitem__)
        1    0.006    0.006    3.183    3.183 teste.py:40(main)
     1961    0.006    0.000    0.006    0.000 {method 'write' of '_io.TextIOWrapper' objects}
      120    0.006    0.000    0.024    0.000 <frozen importlib._bootstrap_external>:1233(find_spec)
        8    0.005    0.001    0.005    0.001 {built-in method matplotlib._image.resample}
     1961    0.005    0.000    0.005    0.000 {method 'find' of 'str' objects}
      362    0.005    0.000    0.005    0.000 {method 'copy' of 'dict' objects}
     1961    0.005    0.000    0.008    0.000 sre_parse.py:253(get)
      120    0.005    0.000    0.007    0.000 <frozen importlib._bootstrap>:1009(_handle_fromlist)
      120    0.004    0.000    0.007    0.000 <frozen importlib._bootstrap_external>:143(_path_isfile)
        4    0.004    0.001    0.025    0.006 __init__.py:2958(_handle_ns)
      185    0.004    0.000    0.004    0.000 {method 'match' of '_sre.SRE_Pattern' objects}
     1961    0.004    0.000    0.004    0.000 {method 'join' of 'str' objects}
      120    0.004    0.000    0.012    0.000 <frozen importlib._bootstrap_external>:57(_path_join)
      362    0.004    0.000    0.020    0.000 sre_parse.py:307(__init__)
     1961    0.004    0.000    0.004    0.000 {method 'pop' of 'dict' objects}
      120    0.004    0.000    0.005    0.000 {built-in method _imp._fix_co_filename}
     1961    0.004    0.000    0.004    0.000 {method 'format_map' of 'str' objects}
      481    0.003    0.000    0.004    0.000 {method 'read' of '_io.StringIO' objects}
        4    0.003    0.001    0.004    0.001 {built-in method builtins.compile}
     1961    0.003    0.000    0.003    0.000 {method 'rstrip' of 'str' objects}
      120    0.003    0.000    0.017    0.000 <frozen importlib._bootstrap_external>:824(get_data)
      144    0.003    0.000    0.003    0.000 {built-in method builtins.any}
      346    0.003    0.000    0.015    0.000 <frozen importlib._bootstrap_external>:75(_path_stat)
      120    0.003    0.000    0.013    0.000 <frozen importlib._bootstrap_external>:63(_path_split)
```

Fonte: Autoria própria (2024).

Após a realização das melhorias, o tempo total de execução durou 2.512 segundos, com um número total de 171.953 chamadas e as funções que mais

consomem tempo são as: *‘show_question’* e *‘check_answer’.* As funções que consumiram mais tempo mudaram após as melhorias, indicando que diferentes partes do código foram otimizadas com sucesso.

As etapas de refatoração do código e as análises realizadas foram fundamentais para identificar e corrigir os pontos fracos que afetavam o desempenho do programa, resultando em um código mais eficiente e responsivo. Este processo mostra a importância da análise de desempenho como parte integrante do desenvolvimento de software, visando melhorar a experiência do usuário e a eficiência do sistema.

4.2 Análise do uso do BDD no desenvolvimento de jogos sérios

Desenvolver jogos sérios é uma tarefa complexa que envolve não apenas a criação de uma experiência de entretenimento, mas a integração de elementos educacionais ou informativos. Nesse contexto, o *Behaviour-Driven Development* (BDD), ou desenvolvimento orientado por comportamento, emerge como uma metodologia eficaz para garantir a qualidade desses jogos.

Na fase inicial de estratégia do processo, a definição de requisitos foi fundamentalmente importante. A partir dessa definição, o BDD permitiu uma análise detalhada dos requisitos do jogo sério, enfatizando a compreensão dos objetivos educacionais e conceitos a serem ensinados. Isso facilitou a comunicação entre o autor deste trabalho e a orientadora, garantindo que todos estejam alinhados quanto às metas do projeto. No entanto, pode haver dificuldades na identificação precisa dos requisitos, especialmente quando se trata de integrar elementos educacionais complexos com a jogabilidade do programa.

A definição dos cenários, baseada nos requisitos estabelecidos, foi importante para orientar o desenvolvimento do jogo sério. Com o BDD, os cenários de teste foram elaborados de forma clara e concisa, descrevendo diferentes interações do jogador e os resultados esperados, garantindo que o jogo atenda às expectativas dos usuários finais e que os objetivos pedagógicos sejam alcançados. No entanto, a elaboração de cenários abrangentes e representativos de todas as situações possíveis pode ser desafiadora, especialmente em jogos sérios com mecânicas complexas.

Na fase da implementação do jogo sério, o BDD forneceu uma estrutura para desenvolver os elementos de acordo com os requisitos e cenários definidos. Isso promove uma abordagem centrada no comportamento, onde as funcionalidades são

projetadas para atender às expectativas dos usuários. A integração de mecânicas de jogo, interfaces gráficas e sistemas de pontuação é facilitada pela clareza dos requisitos estabelecidos pelo BDD. No entanto, a complexidade do código pode aumentar à medida que novas funcionalidades são adicionadas, exigindo uma gestão cuidadosa para garantir a manutenção da qualidade do jogo.

A criação de especificações executáveis em linguagem natural, seguindo o formato *Given-When-Then* do BDD, foi uma etapa importante para garantir a compreensão e validação das funcionalidades do jogo sério. Essas especificações descrevem o comportamento esperado do jogo em resposta às ações do jogador, facilitando a comunicação entre desenvolvedores, testadores e partes interessadas. A clareza e a simplicidade das especificações contribuíram para melhor compreensão dos requisitos e implementação mais precisa. No entanto, manter as especificações atualizadas à medida que o jogo evolui pode ser um desafio, especialmente em projetos de longo prazo.

A automação dos testes de aceitação, como parte da validação do jogo sério, foi simplificada pelo uso do BDD. A criação de *scripts* de teste que simulem as interações do jogador permitiu verificar se o jogo se comporta conforme as especificações executáveis. Isso ajudou a garantir a qualidade e consistência do jogo, acelerando o processo de desenvolvimento e facilitando a detecção de *bugs*. Para isso, a criação e manutenção de *scripts* de teste automatizados pode exigir recursos significativos, especialmente em projetos de grande escala.

A refatoração orientada por comportamento, realizada após a validação dos testes, foi essencial para garantir a qualidade contínua do código-fonte do jogo sério. O BDD forneceu uma base sólida para identificar e corrigir problemas de código com base nos resultados dos testes de aceitação automatizados. Isso promoveu a manutenção de um código limpo, organizado e eficiente, facilitando futuras atualizações e melhorias no jogo, além da legibilidade do código-fonte para manutenção, com a melhora na escrita das variáveis, comentários onde foram necessários e a explicação da lógica para quem rever o código. Entretanto, a refatoração pode ser um processo demorado e requer uma compreensão profunda das interações entre as diferentes partes do jogo.

Em síntese, o uso do BDD no desenvolvimento de jogos sérios ofereceu uma abordagem sistemática e orientada por comportamento para garantir a qualidade e eficácia da interface final. A integração, a definição de requisitos, criação de cenários,

implementação do jogo, especificação executável, validação de testes e refatoração, têm o condão de promover uma colaboração eficaz entre as partes interessadas e uma entrega consistente de funcionalidades. É salutar mencionar a importância de reconhecer os desafios associados à aplicação do BDD em projetos de jogos sérios, especialmente no tocante à complexidade dos requisitos educacionais e técnicos.

Há poucos estudos na literatura que exploraram o uso do BDD no desenvolvimento de jogos sérios mencionados no Capítulo 2, porém, destacam a eficácia na melhoria da qualidade e usabilidade dos jogos.

Os dois estudos Sarinho (2019) e Nascimento (2022), corroboram com o presente trabalho no tocante a aplicabilidade do uso do BDD, onde demonstram a importância da integração de elementos educativos ou informativos nos jogos sérios, bem como a necessidade de uma abordagem sistemática para garantir a qualidade e eficácia da experiência final.

O presente estudo destaca-se pela aplicação do BDD nas fases do processo de desenvolvimento, desde a definição de requisitos até à refatoração orientada para o comportamento, promovendo uma colaboração mais eficaz entre as partes interessadas e uma entrega consistente de funcionalidades.

Já o estudo “Criando um jogo sério sobre Desenvolvimento Orientado a Comportamento para estudantes de TI” (2022) menciona especificadamente sobre a criação de um jogo voltado para um público específico. Enquanto o estudo “BDD Assemble!” (2019) propõem um jogo de aprendizagem em papel para o design do *Behavior Driven Development*, que sugere uma abordagem mais conceitual e menos prática.

Uma das principais dificuldades ao usar o BDD foi a mudança na forma de pensar no desenvolvimento de um jogo sério. Enquanto a programação do dia a dia muitas vezes se concentra na lógica do código e na estruturação técnica, o BDD requer um desenvolvimento mais centrado no comportamento do usuário e nos resultados desejados. Essa transição exigiu um período de adaptação maior, tanto em termos de compreensão dos conceitos quanto na prática do desenvolvimento.

Outra dificuldade está na criação de cenários abrangentes que representem adequadamente a variedade de interações possíveis dentro do jogo sério. Pois, o BDD promove a especificação detalhada de comportamentos esperados ao garantir que os requisitos fossem cobertos de forma adequada.

Em última análise de resultados, embora o BDD ofereça uma estrutura valiosa para o desenvolvimento de jogos sérios, a transição para essa abordagem e a integração de requisitos educacionais foram desafiadores.

O uso do BDD no desenvolvimento do jogo sério QUIZBIIO permitiu uma maior legibilidade e clareza do código. Primeiramente, ao definir os comportamentos esperados do jogo em linguagem natural, tanto os desenvolvedores quanto outras partes interessadas conseguem entender melhor o que o código deve fazer. Além disso, escrever testes antes de começar a programar ajuda a criar um código mais organizado e fácil de manter no futuro. A automação de testes também foi importante, pois permitiu encontrar e corrigir problemas rapidamente, diminuindo os *bugs* e tornando o jogo mais estável. O Quadro 13 apresenta uma comparação do estudo para criação do QUIZBIIO e os trabalhos da literatura.

Quadro 13 - Comparação do QUIZBIIO e os trabalhos da literatura

Critério	Sarinho (2019)	Nascimento (2022)	QUIZBIIO
Objetivo	Facilitar o aprendizado do *design* de BDD através de um jogo de papel.	Ensinar desenvolvimento de jogos utilizando BDD, com foco em requisitos claros e verificáveis	Desenvolver jogos sérios usando BDD para garantir qualidade e eficácia educacional e analisar o uso do BDD.
Método de Ensino	Jogo de cartas e tabuleiros simulando situações reais de desenvolvimento de software.	Definição de requisitos de jogo em linguagem natural, traduzidos em cenários de teste executáveis	Uso de BDD para detalhar requisitos, definir cenários de teste e implementar funcionalidades de acordo com expectativas dos usuários.
Ferramentas Utilizadas	Cartas e tabuleiros.	Cenários de teste automatizados, integração contínua (CI).	Especificações executáveis em formato *Given-When-Then*, *scripts* de teste automatizados, refatoração orientada por comportamento.
Benefícios	Abordagem prática e divertida para o aprendizado de BDD, promovendo uma compreensão profunda	Simplificação do aprendizado do desenvolvimento de jogos, implementação precisa e dirigida, melhor comunicação entre equipes.	Estrutura clara para requisitos e cenários, comunicação facilitada entre partes interessadas, qualidade e consistência por meio de testes automatizados.
Resultados Esperados	Compreensão mais profunda dos conceitos de BDD.	Jogos que atendem melhor às expectativas e necessidades dos usuários, detecção precoce de problemas.	Qualidade contínua do código-fonte, entrega consistente de funcionalidades, manutenção facilitada, colaboração eficaz entre partes interessadas.

5 CONCLUSÃO

Este trabalho analisou as contribuições do BDD no desenvolvimento de um jogo sério do tipo *quiz*. O BDD não apenas fornece uma estrutura sistemática para especificações e testes automatizados, mas também promove uma mudança fundamental na cultura e na dinâmica de desenvolvimento de jogos sérios.

O BDD permitiu definir os comportamentos esperados do jogo de forma clara e acessível. Ao criar cenários e realizar testes antes da conclusão do jogo, pode-se identificar e corrigir erros de forma rápida e eficiente, garantindo a qualidade do jogo sério.

O processo do desenvolvimento do jogo sério adotado por este trabalho tornou-se relevante, pois além de contemplar a etapas reduzidas para criação de jogos, aborda também a adoção do BDD.

O levantamento de requisitos iniciais em linguagem natural dos cenários projetados para descrever interações específicas dos jogadores, foram necessários para garantir que os testes automatizados garantissem a conformidade com esses cenários, contribuindo para um desenvolvimento mais eficiente e alinhado com os objetivos do projeto.

O uso do BDD melhorou a legibilidade do código ao definir os comportamentos em linguagem natural, o que simplificou a manutenção e otimizou as chamadas de função no jogo sério. Ao escrever testes antes de implementar o código, o desenvolvedor é incentivado a criar um código mais modular, coeso e claro, tornando-o mais fácil de entender e modificar no futuro. Além disso, a automação de testes permitiu identificar e corrigir problemas de forma proativa, reduzindo a incidência de *bugs* e melhorando a estabilidade geral do jogo.

Destaca-se que o processo simplificado e utilizado na criação do jogo QUIZBIIO pode ser uma vantagem significativa para desenvolvedores de jogos sérios que desejam basear seu desenvolvimento em testes de alto e baixo nível. Ao seguir uma abordagem estruturada e baseada em comportamento, os desenvolvedores conseguem garantir uma entrega mais consistente de funcionalidades, uma melhor compreensão dos requisitos do projeto e maior confiança na qualidade do produto final.

Por fim, a adoção do BDD no desenvolvimento de jogos sérios não apenas melhora a qualidade do produto ao fornecer uma estrutura para especificações clara

de testes automatizados e *feedbacks* contínuos, como também tem o condão de servir como uma metodologia importante para o desenvolvimento de jogos sérios que visam oferecer uma experiência de aprendizado relevante e envolvente.

5.1 Trabalhos futuros

Considerando os resultados deste trabalho sobre o uso do BDD no desenvolvimento de jogos sérios, existem várias direções para pesquisas futuras. Uma delas é explorar como o BDD pode ser aplicado a diferentes tipos de jogos sérios, especialmente aqueles com mecânicas mais complexas ou objetivos educacionais específicos.

Além disso, pode-se investigar como o BDD combinado com outras metodologias ágeis contribui para criar processos de desenvolvimento mais automatizados e eficientes. Outra pesquisa é analisar como o BDD pode melhorar a colaboração entre equipes multidisciplinares, incluindo desenvolvedores, *designers* e educadores.

É importante que novas pesquisas possam ser realizadas para analisar como a adoção do BDD afeta a experiência dos utilizadores finais, por meio de estudos de usabilidade e *feedback* dos jogadores para melhor compreensão do seu impacto e refiná-lo ainda mais.

REFERÊNCIAS

ANDERLE, A. **Introdução ao BDD como melhoria de processo no desenvolvimento ágil de software.** 2015. Tese de Pós-Graduação. Universidade do Vale do Rio dos Sinos.

JBEHAVE. 2024. Disponível em: https://jbehave.org. Acesso em: 31 março 2024.

GHERKIN. 2008. Disponível em: https://cucumber.io/docs/gherkin/. Acesso em: 17 maio 2023.

LAWRENCE, R.; RAYNER, P. **Behavior-Driven Development with Cucumber**. Editora Addison-Wesley professional, 2019.

NASCIMENTO, F. R. O. do. **Elaboração de um jogo sério sobre behavior driven development para alunos de TI.** 2022. Tese de Graduação. Universidade Federal do Ceará.

NORTH, D. 2000. **Cucumber: behavior driven development (BDD**). Disponível em: https://cucumber.io/. Acesso em: 17 maio 2023.

NORTH, D. 2003. **Introducing BDD**. Disponível em: http://dannorth.net/introducing-dbb. Acesso em: 4 fevereiro 2024.

ROJAS, P. *et al*. Generation of pojos and daos classes from metadata database. **IEEE Latin America Transaction**. v.18, p. 1547-1554, 2020.

SANTOS, W. Ferramentas de Behavior Driven Development (BDD): uma revisão na literatura cinzenta. v.7, n.1, 2023. Universidade de Pernambuco.

SMITH, J.; JONES, A. The role of serious games in biodiversity education. **Journal of Environmental Education**. v.45, n.3, 2020.

SOARES, I. Desenvolvimento orientado por comportamento (BDD) - Um novo olhar sobre o TDD. **Java Magazinne**, v.I, n.91, 2011. Rio de Janeiro.

SARINHO, V. **BDD Assemble**: A paper-based Game proposal for behavior driven development design learning. 2019.