

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
DE MINAS GERAIS (IFMG)
CAMPUS BAMBUÍ
BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

Henrique Augusto Gonçalves Fernandes

**FERRAMENTA DIDÁTICA PARA ENSINO DE USABILIDADE E ACESSIBILIDADE
PARA ESTUDANTES DE IHC DESENVOLVIDA NA GODOT ENGINE**

HENRIQUE AUGUSTO GONÇALVES FERNANDES

**FERRAMENTA DIDÁTICA PARA ENSINO DE USABILIDADE E ACESSIBILIDADE
PARA ESTUDANTES DE IHC DESENVOLVIDA NA GODOT ENGINE**

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 (IFMG) – *Campus Bambuí*, como requisito parcial para obtenção do grau de Bacharel em Engenharia de Computação.

Orientador: Prof. Me. Cláudio Ribeiro de Sousa

Catálogo na Fonte Biblioteca IFMG - Campus Bambuí

F363f Fernandes, Henrique Augusto Gonçalves.

Ferramenta didática para ensino de usabilidade e acessibilidade para estudantes de IHC desenvolvida na Godot Engine / Henrique Augusto Gonçalves Fernandes. – 2026.
111 f. : il. ; color.

Orientador: Cláudio Ribeiro de Sousa.
Trabalho de Conclusão de Curso (Bacharelado em Engenharia de Computação) – Instituto Federal de Minas Gerais. Campus Bambuí, 2026.

1. Interação humano-computador. 2. Usabilidade. 3. Avaliação Heurística. 4. Godot Engine. 5. Ferramenta Didática.
I. Sousa, Cláudio Ribeiro de. II. Instituto Federal de Minas Gerais – Campus Bambuí. III. Título

CDD 005.437

Catálogo: João Batista Rodrigues - CRB-6/2022

Henrique Augusto Gonçalves Fernandes

FERRAMENTA DIDÁTICA PARA ENSINO DE USABILIDADE E ACESSIBILIDADE PARA ESTUDANTES DE IHC DESENVOLVIDA NA GODOT ENGINE

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 (IFMG) – *Campus Bambuí*, como requisito parcial para obtenção do grau de Bacharel em Engenharia de Computação.

Aprovado em 20/05/2026 pela banca examinadora:

Prof. Me. Cláudio Ribeiro de Sousa – IFMG – *Campus Bambuí* – (Orientador)

Prof. Me. Natalia Camillo do Carmo – IFMG – *Campus Bambuí*

Prof. Dr. Bruno Alberto Soares Oliveira – IFMG – *Campus Bambuí*



Documento assinado eletronicamente por **Claudio Ribeiro de Sousa, Professor EBT**, em 20/05/2026, às 15:52, conforme Decreto nº 10.543, de 13 de novembro de 2020.



Documento assinado eletronicamente por **Natalia Camillo do Carmo, Professora Substituta**, em 20/05/2026, às 15:53, conforme Decreto nº 10.543, de 13 de novembro de 2020.



Documento assinado eletronicamente por **Bruno Alberto Soares Oliveira, Professor EBT**, em 20/05/2026, às 15:53, 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 **2736556** e o código CRC **3E2C61B8**.

Aos educadores que compreendem a tecnologia como ferramenta transformadora do ensino e aos estudantes que buscam conhecimento prático em Interação Humano-Computador.

AGRADECIMENTOS

Aos meus pais, Wellington Aparecido Fernandes e Rúbia Fernanda Gonçalves Fernandes, agradeço por todo o apoio, carinho e confiança depositados em mim ao longo desta jornada.

Agradeço ao meu orientador Prof. Cláudio Ribeiro de Sousa pelo apoio e direcionamento durante o desenvolvimento deste projeto.

Expresso minha gratidão aos professores do curso de Engenharia de Computação do IFMG – *Campus* Bambuí, cujos ensinamentos fundamentaram minha formação acadêmica e profissional.

Reconheço também a contribuição dos colegas e da comunidade acadêmica que direta ou indiretamente colaboraram para a realização deste trabalho.

“A tecnologia é melhor quando aproxima as pessoas.”
Matt Mullenweg

RESUMO

O *UX Detective* é uma ferramenta didática digital interativa para o ensino prático das dez heurísticas de usabilidade de Nielsen. Por meio dela, estudantes e profissionais da área de Computação atuam como avaliadores de usabilidade: analisam interfaces realistas com violações heurísticas intencionalmente injetadas, identificam e marcam os problemas encontrados, e recebem *feedback* formativo imediato com explicações didáticas individualizadas por violação. O ensino de Interação Humano-Computador nos cursos de Computação apresenta lacunas na aplicação prática dos conceitos de usabilidade e acessibilidade, demanda que motivou o desenvolvimento desta ferramenta sobre a *Godot Engine* 4.x utilizando GDScript como linguagem principal de programação. A metodologia adotada caracterizou-se como pesquisa aplicada, de natureza exploratória e descritiva, com abordagem qualitativa. O desenvolvimento seguiu etapas metodológicas estruturadas, compreendendo análise bibliográfica, modelagem de requisitos pedagógicos, implementação incremental de protótipos funcionais e documentação técnica abrangente. Adotou-se uma arquitetura de sistemas atômicos comunicantes por sinais, composta por seis *autoloads* (EventBus, SceneManager, GameSession, ProgressTracker, ThemeManager e CursorManager), um catálogo de interfaces acessíveis aderentes ao padrão WAI-ARIA, um catálogo de cenários baseado em recursos *.tres*, além de *managers*, *controllers* e *helpers* responsáveis pela composição dinâmica da interface de avaliação. Como resultado, entregou-se uma ferramenta funcional com dez cenários correspondentes às heurísticas H1–H10 no modo Normal (dificuldade Fácil), sistema de dicas progressivas, tela de resultados com *feedback* diagnóstico por violação e recursos de acessibilidade conformes à WCAG 2.1. A ferramenta encontra-se devidamente documentada e alinhada às práticas pedagógicas para o ensino de IHC, contribuindo para a formação de estudantes de cursos superiores de Computação e para a qualificação de profissionais atuantes nas áreas de usabilidade e acessibilidade.

Palavras-chave: Interação Humano-Computador. Usabilidade. Avaliação Heurística. Godot Engine. Ferramenta Didática. Heurísticas de Nielsen. GDScript. Acessibilidade.

ABSTRACT

UX Detective is an interactive digital didactic tool for the practical teaching of Nielsen's ten usability heuristics. Through it, students and computing professionals act as usability evaluators: they analyze realistic interfaces with intentionally injected heuristic violations, identify and mark the problems found, and receive immediate formative feedback with individualized didactic explanations per violation. The teaching of Human-Computer Interaction in Computer courses presents gaps in the practical application of usability and accessibility concepts, a demand that motivated the development of this tool on Godot Engine 4.x using GDScript as its main programming language. The adopted methodology was characterized as applied research of an exploratory and descriptive nature, with a qualitative approach. The development followed structured methodological stages, including bibliographic analysis, pedagogical requirements modeling, incremental implementation of functional prototypes, and comprehensive technical documentation. An atomic-systems architecture communicating through signals was adopted, comprising six autoloads (EventBus, SceneManager, GameSession, ProgressTracker, ThemeManager, and CursorManager), a catalog of accessible interfaces compliant with the WAI-ARIA standard, a scenario catalog based on .tres resources, as well as managers, controllers, and helpers responsible for the dynamic composition of the evaluation interface. As a result, a functional tool was delivered with ten scenarios corresponding to Nielsen's heuristics H1–H10 in Normal mode (Easy difficulty), a progressive hint system, a results screen providing per-violation diagnostic feedback, and accessibility resources aligned with WCAG 2.1. The tool is duly documented and aligned with pedagogical practices for HCI teaching, contributing to the training of undergraduate computing students and to the qualification of professionals working in usability and accessibility.

Keywords: Human-Computer Interaction. Usability. Heuristic Evaluation. Godot Engine. Didactic Tool. Nielsen's Heuristics. GDScript. Accessibility.

LISTA DE FIGURAS

Figura 1 - Tendências de busca por “ <i>Gamification</i> ”, “Tecnologia educacional”, “EAD” e “ <i>Godot Engine</i> ” (últimos cinco anos)	19
Figura 2 - Arquitetura de nós e cenas da <i>Godot Engine</i>	27
Figura 3 - Exemplo de interface do <i>SQLatch</i> para ensino interativo de SQL	36
Figura 4 - Posicionamento do <i>GDScript</i> em índices de admiração e desejabilidade (Stack Overflow Developer Survey 2025)	37
Figura 5 - Interface do jogo educacional <i>UsabiliCity</i>	38
Figura 6 - Estrutura hierárquica de nós no projeto desenvolvido na <i>Godot Engine</i>	45
Figura 7 - Estrutura de <i>sprints</i> adotada no desenvolvimento	57
Figura 8 - Interface de <i>checkout</i> em estado de protótipo funcional, com lógica implementada antes do refinamento visual	59
Figura 9 - Dica de nível 1 em exibição sobre a interface de testes utilizada no desenvolvimento	59
Figura 10 -Diagrama de caso de uso simplificado da ferramenta didática	60
Figura 11 -Composição de cenas da interface de <i>gameplay</i> a partir de sub-cenas autocontidas	65
Figura 12 -Diagrama de estados do ciclo de vida do cenário	69
Figura 13 -Mapeamento MDA (<i>Mechanics, Dynamics, Aesthetics</i>) do UX Detective	73
Figura 14 -Menu de pausa da interface de <i>gameplay</i> (liberdade e controle do usuário)	75
Figura 15 - <i>Mockup</i> do modo Desafio com <i>power-ups</i> e sistema de vidas no painel esquerdo	76
Figura 16 - <i>Mockup</i> do modo Desafio com obstáculos dinâmicos (<i>pop-ups</i> e <i>CAPTCHA</i>) sobre o cenário	77
Figura 17 -Tela do modo Score: seleção de submodo e <i>leaderboard</i> Global e Pessoal	78
Figura 18 -Fluxograma simplificado dos três modos de jogo principais (Normal, Desafio e Score)	78
Figura 19 -Diagrama de sequência do fluxo de submissão e carregamento da tela de resultados	82
Figura 20 -Tela inicial da ferramenta didática	83
Figura 21 -Menu principal com acesso aos diferentes modos de jogo	84
Figura 22 -Menu do modo Normal: seleção de heurística e dificuldade, com progresso e estrelas	85
Figura 23 -Cenário em execução no modo Normal	86
Figura 24 -Esquema da <i>GameplayInterface</i> com definição de cada componente	86

Figura 25 -Interface de <i>login</i> em avaliação, com violação injetada no botão “Lembre-se de mim”, marcação do usuário e <i>badge</i> no painel esquerdo	88
Figura 26 -Tela de resultados com análise de pontuação, resultados do cenário e <i>feedback</i> didático	88
Figura 27 -Fluxo do assistente virtual do tutorial interativo	90
Figura 28 -Recorte representativo da implementação em GDScript da ferramenta	92
Figura 29 -Grafo agregado de <i>publish/subscribe</i> do <i>EventBus</i> : emissores, sinais e ouvintes	105
Figura 30 -Modelo entidade-relacionamento dos <i>Resources</i> .tres (visão <i>data-driven</i>)	106
Figura 31 -Modelo entidade-relacionamento completo da ferramenta (conteúdo didático e persistência de usuário)	107
Figura 32 -Diagrama de rastreabilidade de requisitos para artefatos de código e cena	109
Figura 33 - <i>Loop</i> de <i>gameplay</i> e ciclo de avaliação heurística no UX Detective . .	111

LISTA DE QUADROS

Quadro 1 - Síntese das dez heurísticas de Nielsen com aplicação educacional .	25
Quadro 2 - Estrutura metacognitiva de GDD para aplicações de ensino de usabilidade	29
Quadro 3 - Correspondências entre princípios POUR e heurísticas de Nielsen .	31
Quadro 4 - Correspondências entre Taxonomia de Bloom e mecânicas de jogo para ensino de usabilidade	34
Quadro 5 - Estado da arte: Síntese de ferramentas didáticas digitais para ensino de IHC	39
Quadro 6 - Ferramentas didáticas para ensino técnico e contexto de uso	40
Quadro 7 - Relação entre objetivos do projeto e resultados obtidos	40
Quadro 8 - Glossário técnico: termos da Godot Engine, padrões arquiteturais e matriz de confusão	43
Quadro 9 - Pilares conceituais da Godot Engine e sua aplicação no UX Detective	48
Quadro 10 -Autoloads e Resources da arquitetura de sistemas atômicos	53
Quadro 11 -Síntese dos principais requisitos funcionais e seu estado de implementação	56
Quadro 12 -Distribuição dos <i>sprints</i> quinzenais entre macrofases do desenvolvimento	58
Quadro 13 -Padrões ARIA APG implementados na ferramenta didática	64
Quadro 14 -Estrutura de dados para representação de violações em componentes ARIA	66
Quadro 15 -Exemplo de configuração de violação em recurso <i>ViolationData</i> . . .	67
Quadro 16 -Catálogo de interfaces por heurística, tema e serviço análogo de referência	71
Quadro 17 -Estratégias de implementação das heurísticas de Nielsen	72
Quadro 18 -Responsabilidades centrais do <i>GameSession</i>	79
Quadro 19 -Responsabilidades centrais do <i>ValidationManager</i>	80
Quadro 20 -Injeção de violação H4 (Consistência) via <i>ViolationData</i>	81
Quadro 21 -Fluxo principal do modo Normal e componentes responsáveis	87
Quadro 22 -Cobertura do catálogo de cenários por heurística e dificuldade . . .	91
Quadro 23 -Síntese consolidada do estado de entrega do UX Detective	93

LISTA DE SIGLAS

- ABNT – Associação Brasileira de Normas Técnicas
- APG – *ARIA Authoring Practices Guide*
- API – *Application Programming Interface*
- eMAG – Modelo de Acessibilidade em Governo Eletrônico
- GDD – *Game Design Document*
- FN – *False Negative* (matriz de confusão)
- FP – *False Positive* (matriz de confusão)
- GDScript – *Godot Script*
- GUI – *Graphical User Interface*
- GUT – *Godot Unit Test*
- HCI – *Human-Computer Interaction*
- IFMG – Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais
- IHC – Interação Humano-Computador
- INPI – Instituto Nacional da Propriedade Industrial
- ISO – *International Organization for Standardization*
- LBI – Lei Brasileira de Inclusão da Pessoa com Deficiência
- MDA – *Mechanics, Dynamics, Aesthetics*
- MIT – *Massachusetts Institute of Technology*
- NIT – Núcleo de Inovação Tecnológica
- POUR – *Perceivable, Operable, Understandable, Robust*
- SOLID – *Single responsibility, Open-closed, Liskov substitution, Interface segregation, Dependency inversion*
- TCC – Trabalho de Conclusão de Curso
- TN – *True Negative* (matriz de confusão)
- TP – *True Positive* (matriz de confusão)
- UI – *User Interface*
- UX – *User Experience*
- W3C – *World Wide Web Consortium*
- WAI-ARIA – *Web Accessibility Initiative – Accessible Rich Internet Applications*
- WCAG – *Web Content Accessibility Guidelines*

SUMÁRIO

1	INTRODUÇÃO	17
1.1	Objetivo Geral	17
1.2	Objetivos Específicos	17
1.3	Justificativa	18
1.4	Resultados Esperados	19
2	FUNDAMENTAÇÃO TEÓRICA	21
2.1	Fundamentos Conceituais	21
2.1.1	<i>Interação Humano-Computador: Histórico e Importância</i>	21
2.1.2	<i>Heurísticas de Nielsen para Usabilidade</i>	22
2.1.3	<i>Programação Visual e Ferramentas de Desenvolvimento</i>	25
2.1.4	<i>Ferramentas de Desenvolvimento de Jogos e Godot Engine</i>	26
2.1.5	<i>Game Design Document (GDD)</i>	28
2.1.6	<i>Acessibilidade Digital e Princípios POUR</i>	29
2.1.7	<i>Metodologias Ativas de Ensino</i>	31
2.1.8	<i>Aprendizagem Baseada em Jogos e Fundamentos Pedagógicos</i>	32
2.2	Revisão Bibliográfica	35
2.2.1	<i>Ensino de Interação Humano-Computador</i>	35
2.2.2	<i>Ferramentas Didáticas Digitais em IHC</i>	35
2.2.3	<i>Abordagens Visuais de Programação na Educação</i>	36
2.2.4	<i>Gamificação e Jogos Educacionais</i>	38
2.2.5	<i>Estado da Arte em Ferramentas Didáticas para IHC</i>	39
3	METODOLOGIA	41

3.1	Caracterização da Pesquisa	41
3.2	Convenções e Glossário Técnico	42
3.3	Ferramentas Tecnológicas	44
3.3.1	<i>Godot Engine como Ambiente de Desenvolvimento</i>	44
3.3.1.1	Hierarquia de Nós e Árvore de Cenas	44
3.3.1.2	Control Nodes e Containers	44
3.3.1.2	Controles de interação.	44
3.3.1.2	Containers de layout.	46
3.3.1.3	Sistema de Sinais (<i>Signals</i>)	46
3.3.1.4	Cenas como Unidade Modular	46
3.3.1.5	<i>Resources</i> para Dados Padronizados	47
3.3.2	<i>GDScript como Linguagem de Desenvolvimento</i>	47
3.3.2.1	Mecanismos de Extensibilidade	48
3.3.2.2	Documentação <i>Inline</i> e <i>Rich Text</i> no Editor	49
3.3.2.3	Nós Únicos (%UniqueName)	49
3.3.2.4	Convenções de Codificação Adotadas	49
3.3.3	<i>Ferramentas de Design e Prototipação</i>	50
3.3.4	<i>Arquitetura de Sistemas Atômicos</i>	51
3.3.4.1	Autoloads (Singletons Globais)	51
3.3.4.2	<i>Resources</i> para Conteúdo Didático	52
3.3.4.3	<i>Managers</i> , <i>Controllers</i> e <i>Helpers</i>	54
3.3.5	<i>Framework de Testes GUT</i>	54
3.4	Métodos Aplicados	55
3.4.1	<i>Engenharia de Requisitos</i>	55

3.4.2 Metodologia Ágil Adaptada	57
3.4.3 Diagramação e Modelagem	60
3.4.4 Design Educacional e Framework Pedagógico	61
3.4.5 Arquitetura de Componentes ARIA	62
3.4.5.1WAI-ARIA: Conceitos Fundamentais	63
3.4.5.2ARIA Authoring Practices Guide (APG)	63
3.4.5.3Adaptação para Godot Engine	64
3.4.5.4Arquitetura com Violações Intencionais	65
3.4.5.5Fluxo de Carregamento de Cenários	67
3.5 Desenvolvimento	70
3.5.1 Seleção e Diversidade Temática das Interfaces	70
3.5.2 Implementação das Heurísticas de Nielsen	70
3.5.3 Estruturação dos Modos de Jogo	73
3.5.3.1Modo Tutorial	74
3.5.3.2Modo Normal	74
3.5.3.3Modo Desafio	75
3.5.3.4Modo Score	76
3.5.4 Implementação com GDScript	79
3.5.4.1GameSession	79
3.5.4.2GameplayInterfaceController	79
3.5.4.3MarkerTool	80
3.5.4.4ValidationManager	80
3.5.5 Processo de Avaliação Formativa	81
4 RESULTADOS	83

4.1	Interface e Design Visual	83
4.1.1	<i>Tela Inicial e Menu Principal</i>	83
4.1.2	<i>Elementos Visuais e Identidade Gráfica</i>	83
4.2	Funcionalidades Implementadas	84
4.2.1	<i>Modo Normal: Estado Funcional</i>	84
4.2.2	<i>Fluxo Principal da Aplicação</i>	87
4.2.3	<i>Tutorial Interativo: Estado Parcial</i>	89
4.2.4	<i>Modos Desafio e Score: Estado Projetado</i>	89
4.2.5	<i>Cobertura das Heurísticas e Catálogo de Cenários</i>	91
4.2.6	<i>Implementação com GDScript</i>	91
4.3	Síntese Consolidada dos Resultados	92
5	CONCLUSÃO	94
5.1	Trabalhos Futuros	95
5.2	Continuidade e Estratégia de Propriedade Intelectual	96
	REFERÊNCIAS	97
	APÊNDICES	101
	APÊNDICE A - Protocolo de Estudo Controlado para Validação Empírica	101
A.1	Objetivos e Hipóteses	101
A.2	Desenho Experimental	101
A.3	Participantes	102
A.4	Instrumentos	102
A.5	Procedimento Experimental	103
A.6	Métricas e Análise Estatística	103
A.7	Considerações Éticas	103

A.8	Cronograma Estimado	104
APÊNDICE B	- Grafo de Sinais do EventBus	105
APÊNDICE C	- Modelo ER dos <i>Resources</i> .tres	106
APÊNDICE D	- Modelo ER Completo da Ferramenta	107
APÊNDICE E	- Diagrama de Rastreabilidade de Requisitos	108
APÊNDICE F	- <i>Loop</i> de <i>Gameplay</i> e Ciclo de Avaliação	110

1 INTRODUÇÃO

A Interação Humano-Computador (IHC) é disciplina essencial na formação de profissionais da computação, ao abordar princípios de usabilidade, acessibilidade e *design* de interfaces que impactam diretamente a qualidade dos produtos digitais. Segundo Nielsen (1994), a usabilidade é o pilar que garante sistemas eficazes, eficientes e satisfatórios para o usuário.

No contexto da graduação em Engenharia de Computação, predominam abordagens teóricas no ensino de IHC, com limitações quanto à aplicação prática dos conceitos (FERREIRA *et al.*, 2014). Para suprir essa lacuna, adotam-se crescentemente recursos didáticos digitais, jogos sérios e ambientes interativos no ensino de usabilidade, acessibilidade e avaliação heurística, complementando as abordagens expositivas tradicionais (JOMSRI; PRANGCHUMPOL; POONSILP, 2025; BENITTI, 2015).

A Godot Engine destaca-se como plataforma contemporânea para desenvolvimento de recursos didáticos digitais, notadamente por sua arquitetura orientada a nós e suporte ao GDScript, linguagem de *scripting* com sintaxe similar ao Python (RANAWEERA; MAHMOUD, 2024). A maturidade atual do *engine*, aliada ao seu caráter gratuito e de código aberto, oferece fundamentos sólidos para a construção de artefatos educacionais de alta interatividade em contexto acadêmico (GONÇALVES, 2023).

Este projeto visa contribuir para o ensino prático de usabilidade e acessibilidade por meio do desenvolvimento de uma ferramenta didática digital na Godot Engine, fundamentada nas dez heurísticas de Nielsen, buscando proporcionar experiências de aprendizagem ativa tanto a estudantes de cursos superiores de Computação quanto a profissionais atuantes nas áreas de usabilidade e acessibilidade.

1.1 Objetivo Geral

Desenvolver uma ferramenta didática digital baseada na Godot Engine, utilizando GDScript como linguagem principal de programação, para apoiar o ensino prático dos fundamentos de usabilidade e acessibilidade, com ênfase nas dez heurísticas de Nielsen, direcionada a estudantes de IHC em cursos de ensino superior da área de Computação e a profissionais atuantes em usabilidade e acessibilidade.

1.2 Objetivos Específicos

- a) Realizar levantamento bibliográfico sobre metodologias de ensino de usabilidade, ferramentas pedagógicas digitais e aplicação das heurísti-

- cas de Nielsen em contextos educacionais;
- b) Projetar cenários didáticos interativos e exemplos práticos de violações e implementações corretas dos princípios de usabilidade, estruturados pedagogicamente para progressão de aprendizagem;
- c) Modelar e implementar a ferramenta na Godot Engine utilizando GDScript e sua arquitetura orientada a nós e sinais;
- d) Documentar detalhadamente o processo de desenvolvimento, incluindo arquitetura do sistema, funcionalidades implementadas, guia de utilização pedagógica e exemplos de aplicação em contextos educacionais de Computação.

1.3 Justificativa

A proposta justifica-se pela necessidade de recursos práticos e dinâmicos para o ensino de usabilidade, alinhando-se às tendências pedagógicas contemporâneas e à demanda do mercado profissional por engenheiros de computação capacitados em *design* de interfaces e experiência do usuário. Metodologias ativas de ensino, suportadas por ferramentas digitais interativas, promovem maior engajamento discente e melhor retenção de conhecimento em disciplinas técnicas, especialmente em IHC (JOMSRI; PRANGCHUMPOL; POONSILP, 2025).

A utilização da Godot Engine fundamenta-se em sua acessibilidade, arquitetura modular e comunidade ativa, além da linguagem GDScript que apresenta sintaxe simplificada e curva de aprendizado favorável (PERIN; SILVA; VALENTIM, 2023). Ferramentas didáticas digitais, como a proposta neste trabalho, podem complementar o ensino de IHC, ampliando o repertório de estratégias pedagógicas disponíveis a docentes e discentes.

Experiências de trabalhos anteriores, como o UsabiliCity (FERREIRA *et al.*, 2014) e o UsabilityGame (BENITTI, 2015), demonstram a viabilidade de recursos computacionais lúdicos para o ensino de usabilidade. A Figura 1 apresenta uma comparação de tendências de busca por quatro termos diretamente correlacionados ao escopo deste trabalho, coletada no Google *Trends* e abrangendo os últimos cinco anos: “*Gamification*” (linha em azul), “Tecnologia educacional” (linha em vermelho), “EAD” (linha em amarelo) e “*Godot Engine*” (linha em verde). Observa-se tendência geral de crescimento, com destaque para a *Godot Engine* e para tecnologias educacionais, cujo interesse intensificou-se no último ano.

Como síntese comparativa de ferramentas didáticas digitais para IHC, apresenta-se o Quadro 5.

A principal contribuição deste trabalho é o *UX Detective*: uma ferramenta didática digital que combina gamificação, *feedback* formativo progressivo e caráter

Figura 1 – Tendências de busca por “Gamification”, “Tecnologia educacional”, “EAD” e “Godot Engine” (últimos cinco anos)



Fonte: GOOGLE TRENDS, adaptado pelo autor, 2026.

autoexemplificante. Diferencia-se das ferramentas correlatas identificadas na literatura — como o UsabiliCity (FERREIRA *et al.*, 2014), o UsabilityGame (BENITTI, 2015) e o HCI Gamification App (JOMSRI; PRANGCHUMPOL; POONSILP, 2025) — por implementar, na própria interface, os princípios de usabilidade e acessibilidade que se propõe a ensinar, seguindo padrões WAI-ARIA e WCAG 2.1. Além disso, o sistema de dicas progressivas com custo variável e a tela de resultados com análise diagnóstica por violação oferecem um ciclo de avaliação formativa que vai além da simples pontuação, conectando a experiência lúdica diretamente ao desenvolvimento de competências analíticas de avaliação heurística.

1.4 Resultados Esperados

Como resultado principal do trabalho, entregou-se uma ferramenta didática digital funcional, documentada e alinhada às boas práticas pedagógicas para o ensino de usabilidade em IHC. A aplicação proporcionou um ambiente interativo para experimentação prática dos conceitos de usabilidade fundamentados nas dez heurísticas de Nielsen, apoiando-se em avaliação direta de interfaces realistas com marcação de violações, *feedback* formativo e dicas progressivas.

Adicionalmente, produziu-se documentação técnica abrangente, incluindo manual de utilização pedagógica, guia de arquitetura, catálogo de cenários didáticos aplicáveis ao ensino de Engenharia de Computação e diagramas de fluxo. A ferramenta foi concebida para ser utilizada como recurso complementar em disciplinas de IHC, promovendo atualização e inovação no ensino da área.

Espera-se contribuição para a comunidade acadêmica por meio do registro de *software* junto ao Núcleo de Inovação Tecnológica (NIT) do IFMG e do Instituto

Nacional da Propriedade Industrial (INPI), com posterior abertura do repositório de código-fonte sob licença alinhada à estratégia institucional de propriedade intelectual, ampliando o impacto dos resultados para além do contexto institucional. A estratégia de continuidade detalhada no Capítulo 5 prevê a manutenção do repositório como privado durante o processo formal de registro e sua subsequente publicização.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Fundamentos Conceituais

2.1.1 Interação Humano-Computador: Histórico e Importância

A Interação Humano-Computador (IHC) consolidou-se como campo interdisciplinar na década de 1980, dedicado ao projeto, avaliação e implementação de sistemas computacionais interativos para uso humano (NIELSEN, 1994). Foi um período marcado pela transição dos computadores de ambientes corporativos restritos para contextos pessoais e domésticos, demandando interfaces mais intuitivas e acessíveis (NORMAN, 2013).

O desenvolvimento histórico da IHC pode ser compreendido em três fases principais, conforme sintetiza Myers (1998):

- a) **Fase dos sistemas em lote (1950-1970):** Caracterizada pela interação indireta através de cartões perfurados e processamento diferido, com foco na eficiência computacional em detrimento da experiência do usuário;
- b) **Fase das interfaces de linha de comando (1970-1980):** Introdução de terminais interativos que permitiam comunicação direta com o sistema, embora ainda exigindo memorização de comandos sintáticos complexos;
- c) **Fase das interfaces gráficas (1980-presente):** Emergência de interfaces visuais baseadas em metáforas do mundo real, utilizando elementos como janelas, ícones e menus, democratizando o acesso à computação.

A norma ISO 9241-11:2018 (ISO, 2018) define usabilidade como o grau em que um sistema pode ser usado por usuários específicos para atingir objetivos específicos com eficácia, eficiência e satisfação em um contexto de uso especificado. Esta definição estabelece três pilares fundamentais para avaliação de interfaces:

- **Eficácia:** Precisão e completude com que usuários atingem objetivos específicos;
- **Eficiência:** Recursos gastos em relação à precisão e completude dos objetivos alcançados;
- **Satisfação:** Ausência de desconforto e presença de atitudes positivas em relação ao uso do sistema.

No contexto brasileiro, o ensino de IHC em cursos de Engenharia de Computação e áreas correlatas ganhou relevância a partir dos anos 2000, motivado pela crescente demanda do mercado por profissionais capacitados em *design* de interfaces

e experiência do usuário. Segundo Ferreira *et al.* (2014), o ensino de usabilidade no Brasil tradicionalmente privilegia abordagens teóricas, apresentando lacunas na aplicação prática dos conceitos, justificando a necessidade de ferramentas didáticas que promovam aprendizagem ativa e contextualizada.

A importância da IHC estende-se para além da qualidade de produtos digitais, impactando diretamente aspectos como inclusão digital, acessibilidade, produtividade organizacional e satisfação dos usuários. Norman (2013) argumenta que sistemas mal projetados não apenas frustram usuários, mas também geram custos econômicos significativos decorrentes de erros, retrabalho e abandono de sistemas. No cenário contemporâneo, onde a interação com sistemas computacionais permeia praticamente todos os aspectos da vida cotidiana, a formação de engenheiros de computação capacitados em IHC constitui imperativo tanto acadêmico quanto profissional.

2.1.2 Heurísticas de Nielsen para Usabilidade

Originalmente propostas por Nielsen e Molich (1990) e consolidadas em Nielsen (1994), as dez heurísticas de usabilidade constituem o *framework* conceitual fundamental desta pesquisa. Essas heurísticas representam princípios gerais de *design* de interfaces que, quando aplicados, resultam em sistemas mais utilizáveis e satisfatórios.

As heurísticas de Nielsen são amplamente reconhecidas e utilizadas na avaliação de interfaces devido à sua abrangência, simplicidade de aplicação e eficiência na identificação de problemas de usabilidade.

- a) **Visibilidade do status do sistema:** O sistema deve sempre manter os usuários informados sobre o que está acontecendo, fornecendo *feedback* apropriado dentro de um tempo razoável. Esta heurística enfatiza a importância da comunicação contínua entre sistema e usuário, por meio de indicadores de progresso, mensagens de confirmação e sinais visuais que confirmem ações realizadas.
- b) **Correspondência entre o sistema e o mundo real:** O sistema deve falar a linguagem dos usuários, com palavras, frases e conceitos familiares, em vez de termos orientados ao sistema. Deve-se seguir convenções do mundo real, apresentando informações em uma ordem lógica e natural. Esta heurística ressalta a importância de utilizar metáforas adequadas e vocabulário compreensível ao contexto cultural e cognitivo dos usuários.
- c) **Controle e liberdade do usuário:** Usuários frequentemente escolhem funções do sistema por engano e precisam de uma “saída de emergência” claramente marcada para deixar o estado indesejado sem passar

por um processo estendido. Esta heurística enfatiza a importância de opções de desfazer e refazer, bem como caminhos alternativos para atingir um objetivo, evitando que o usuário se sinta encurralado pelo sistema.

- d) **Consistência e padrões:** Os usuários não devem ter que se perguntar se diferentes palavras, situações ou ações significam a mesma coisa. Esta heurística destaca a necessidade de padronização na interface, garantindo que elementos similares funcionem e apareçam de maneira similar, criando coerência visual e funcional que reduz a carga cognitiva do usuário.
- e) **Prevenção de erros:** Melhor que boas mensagens de erro é um *design* cuidadoso que previne que um problema ocorra. Esta heurística promove a eliminação de condições propensas a erros e a confirmação antes de ações críticas, antecipando possíveis equívocos do usuário e oferecendo mecanismos para evitá-los.
- f) **Reconhecimento em vez de memorização:** Minimizar a carga de memória do usuário tornando objetos, ações e opções visíveis. O usuário não deve ter que lembrar informações de uma parte da interface para outra. Esta heurística enfatiza a importância de disponibilizar visualmente as opções e comandos, evitando que o usuário precise memorizar procedimentos ou comandos específicos.
- g) **Flexibilidade e eficiência de uso:** Aceleradores — invisíveis para usuários novatos — podem acelerar a interação para usuários experientes, permitindo que o sistema atenda tanto usuários inexperientes quanto experientes. Esta heurística destaca a importância de oferecer múltiplos caminhos para realizar uma tarefa, com níveis progressivos de complexidade e otimização.
- h) **Design estético e minimalista:** As interfaces não devem conter informações irrelevantes ou raramente necessárias. Cada unidade extra de informação compete com as unidades relevantes e diminui sua visibilidade relativa. Esta heurística promove a simplicidade e clareza visual, removendo elementos desnecessários que possam distrair o usuário da informação essencial.
- i) **Ajudar os usuários a reconhecer, diagnosticar e recuperar-se de erros:** Mensagens de erro devem usar linguagem simples (sem códigos), indicar com precisão o problema e sugerir construtivamente uma solução. Esta heurística ressalta que erros são oportunidades de aprendizado quando adequadamente apresentados, fornecendo orientação específica e contextual.

- j) **Ajuda e documentação:** Embora seja preferível que o sistema possa ser usado sem documentação, pode ser necessário fornecer ajuda e documentação. Tais informações devem ser fáceis de pesquisar, focadas na tarefa do usuário, concretas quanto aos passos a serem realizados e concisas em sua extensão. Esta heurística reconhece que o suporte complementar, quando necessário, deve ser acessível e relevante ao contexto de uso.

A aplicação sistemática das heurísticas orienta o *design* de sistemas centrados no usuário, reduz a fricção cognitiva e aumenta a satisfação na interação. No contexto educacional, elas servem simultaneamente como conteúdo de estudo e como princípios de *design* da própria ferramenta didática, criando meta-aprendizado em que o meio exemplifica o conteúdo.

Aplicações contemporâneas das heurísticas de Nielsen evidenciam sua robustez e adaptabilidade a contextos tecnológicos emergentes. Estudos recentes, como o de Jomsri, Prangchumpol e Poonsilp (2025), demonstram que as heurísticas permanecem relevantes para avaliação de interfaces móveis, aplicações em nuvem e sistemas com componentes de inteligência artificial, embora possam requerer interpretações contextualizadas. Por exemplo, a heurística de “Visibilidade do status do sistema” aplica-se a *chatbots* através de indicadores de digitação, enquanto a “Prevenção de erros” manifesta-se em sistemas de recomendação através de confirmações antes de ações irreversíveis.

Críticas às heurísticas de Nielsen também enriquecem sua compreensão e aplicação. Norman (2013) argumenta que, embora as heurísticas forneçam *framework* útil para identificação de problemas de usabilidade, sua aplicação mecânica sem consideração do contexto de uso específico pode resultar em avaliações superficiais. Similarmente, pesquisadores contemporâneos apontam que as heurísticas, desenvolvidas originalmente para interfaces desktop, necessitam adaptação para contextos como dispositivos vestíveis, realidade aumentada e interfaces por voz. Estas críticas não diminuem o valor das heurísticas, mas evidenciam a necessidade de interpretação contextualizada e evolução contínua dos princípios.

No contexto da ferramenta didática desenvolvida neste trabalho, as heurísticas de Nielsen cumprem duplo papel: (1) são o conteúdo pedagógico principal, estudado por meio de cenários interativos; e (2) orientam o *design* da ferramenta, fazendo com que sua interface exemplifique as boas práticas que ensina. Esta autoexemplificação oferece ao estudante experiência direta dos princípios em uso.

O Quadro 1 sintetiza as dez heurísticas com suas aplicações educacionais na ferramenta. A organização dos cenários didáticos na ferramenta segue estrutura progressiva, apresentando cada heurística através de pares contrastantes de interfaces: uma versão que viola o princípio (*bad UX*) e outra que o implementa corretamente

(*good UX*). Esta abordagem comparativa, fundamentada em princípios de aprendizagem por contraste (PAIVA *et al.*, 2022), facilita a compreensão conceitual ao tornar as diferenças explícitas e tangíveis. Adicionalmente, sistemas de dicas progressivas e *feedback* formativo guiam o estudante na identificação e compreensão das violações heurísticas, promovendo desenvolvimento gradual de expertise em avaliação de usabilidade.

Quadro 1 – Síntese das dez heurísticas de Nielsen com aplicação educacional

Heurística	Princípio Fundamental	Aplicação Educacional
1	Visibilidade do status do sistema	<i>Feedback</i> imediato sobre progresso de aprendizagem
2	Correspondência sistema-mundo real	Interface familiar com metáforas educacionais
3	Controle e liberdade do usuário	Possibilidade de exploração sem consequências
4	Consistência e padrões	Interface uniforme entre diferentes atividades
5	Prevenção de erros	Design que evita confusão conceitual
6	Reconhecimento vs. memorização	Informações visíveis conforme necessidade
7	Flexibilidade e eficiência de uso	Adaptação a diferentes níveis de conhecimento
8	Design estético e minimalista	Foco nos elementos essenciais da aprendizagem
9	Recuperação de erros	Mensagens educativas sobre correções
10	Ajuda e documentação	Suporte contextual para conceitos de usabilidade

Fonte: Elaborado pelo autor com base em (NIELSEN, 1994), 2026.

2.1.3 Programação Visual e Ferramentas de Desenvolvimento

A evolução das linguagens de programação visual está fundamentalmente ligada à busca por modelos cognitivos mais intuitivos para representação e manipulação de algoritmos. Conforme Perin, Silva e Valentim (2023), abordagens visuais de programação operam no limite entre o pensamento concreto e o pensamento formal abstrato, proporcionando representações tangíveis de conceitos computacionais abstratos.

O desenvolvimento histórico deste paradigma tem evoluído constantemente, manifestando-se atualmente em diversos ambientes educacionais e profissi-

onais, incluindo motores de jogos como a Godot Engine (GONÇALVES, 2023). Estudos sobre abordagens visuais de programação sugerem três principais benefícios cognitivos:

- a) **Redução de carga cognitiva extrínseca:** a simplificação sintática permite maior foco em aspectos algorítmicos;
- b) **Suporte à visualização mental:** representações visuais facilitam a formação de modelos mentais sobre fluxos de execução;
- c) **Scaffolding cognitivo:** estruturas pré-definidas funcionam como apoio que auxilia na construção de conhecimentos mais complexos (WÖRISTER; KNOBELSDORF, 11/2023).

No contexto específico do ensino de usabilidade, ferramentas de desenvolvimento de jogos oferecem a vantagem de serem, em si mesmas, exemplos de *design* centrado no usuário, incorporando princípios de *affordance*, *feedback* imediato e prevenção de erros que são objetos de estudo na disciplina de IHC (NIELSEN, 1994). Esta meta-aplicação dos conceitos facilita a assimilação dos princípios de Nielsen.

2.1.4 Ferramentas de Desenvolvimento de Jogos e Godot Engine

O ecossistema de ferramentas para desenvolvimento de jogos eletrônicos tem experimentado significativa democratização na última década, com motores de jogos tornando-se progressivamente mais acessíveis e versáteis. Segundo Ranaweera e Mahmoud (2024), esta tendência tem alterado não apenas o perfil da indústria de jogos, mas também viabilizado aplicações educacionais, científicas e culturais anteriormente inviáveis devido às barreiras técnicas e financeiras.

Gonçalves (2023) classifica as ferramentas contemporâneas para desenvolvimento de jogos em quatro categorias principais, com base em seus propósitos e requisitos de especialização técnica:

- a) **Motores de jogos de propósito geral:** como Unreal Engine, Unity e Godot, que oferecem funcionalidades abrangentes para desenvolvimento de diversos gêneros e plataformas;
- b) **Frameworks especializados:** como Phaser (para jogos HTML5), PyGame (para Python) e MonoGame, que se concentram em contextos de desenvolvimento específicos;
- c) **Ferramentas orientadas a designers:** como GameMaker Studio, Construct e RPG Maker, que priorizam fluxos de trabalho visuais com menor necessidade de programação;
- d) **Plataformas educacionais:** como Scratch, Kodu Game Lab e MakeCode Arcade, projetadas especificamente para fins didáticos.

A Godot Engine distingue-se no panorama dos motores de jogos por ca-

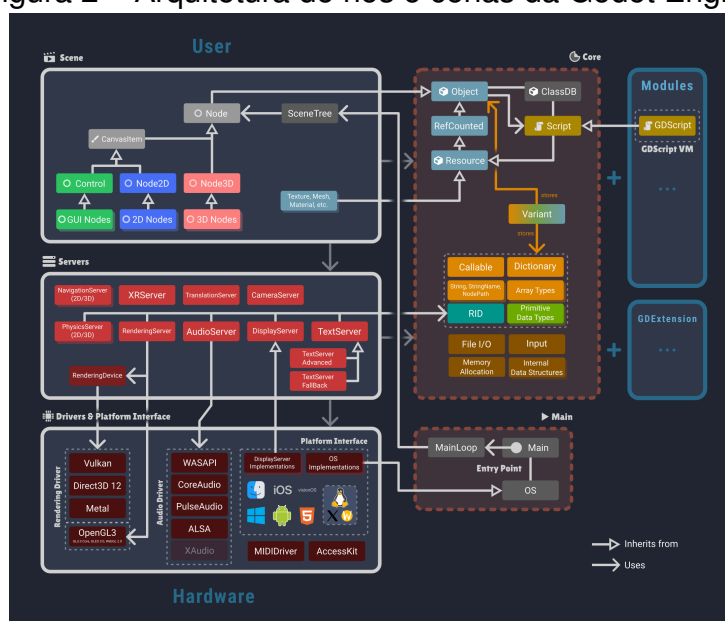
racterísticas particulares que a tornam especialmente adequada para contextos educacionais. Sua arquitetura orientada a cenas e nós, licença MIT permissiva, tamanho compacto e comunidade colaborativa configuram um ecossistema com significativo potencial para aplicações pedagógicas (RANAWEERA; MAHMOUD, 2024).

Uma análise comparativa realizada por Gonçalves (2023) identificou na Godot Engine vantagens competitivas em cinco dimensões críticas:

- Acessibilidade financeira:** sendo completamente gratuita e de código aberto, sem restrições ou taxas de licenciamento;
- Portabilidade:** exigindo requisitos de sistema modestos e permitindo desenvolvimento em múltiplas plataformas;
- Interface intuitiva:** oferecendo um ambiente de desenvolvimento integrado (IDE) com baixa fragmentação de ferramentas;
- Curva de aprendizado otimizada:** particularmente com GDScript, linguagem projetada especificamente para o *engine* com sintaxe inspirada em Python;
- Extensibilidade:** permitindo a integração de *plugins* e extensões, ampliando suas capacidades.

A arquitetura da Godot Engine, baseada em um sistema hierárquico de nós organizados em cenas, reflete naturalmente princípios de *design* modular e encapsulamento, conceitos fundamentais em Engenharia de Software, conforme ilustrado na Figura 2 (PERIN; SILVA; VALENTIM, 2023). Esta estrutura facilita a visualização de conceitos abstratos, tornando-os mais tangíveis para estudantes em fase inicial de formação.

Figura 2 – Arquitetura de nós e cenas da Godot Engine



Fonte: Documentação oficial da Godot Engine:

https://docs.godotengine.org/en/stable/engine_details/architecture/godot_architecture_diagram.html.

O sistema de sinais da Godot implementa o padrão de projeto *Observer*, estabelecendo um paradigma de programação orientada a eventos particularmente relevante para o desenvolvimento de interfaces interativas (SELVESAKIS; KAZANLIDIS, 2024). Esta abordagem proporciona um modelo mental intuitivo para conceitos de IHC como *feedback*, responsividade e interatividade, facilitando a compreensão de princípios de usabilidade por meio de demonstrações práticas e experimentação direta.

2.1.5 Game Design Document (GDD)

O *Game Design Document* (GDD) constitui um artefato fundamental no processo de desenvolvimento de jogos e aplicações interativas, funcionando como *blueprint* que articula visão, mecânicas, narrativa e aspectos técnicos em um documento coeso e abrangente (HOLM; HUKE, 2020). No contexto de aplicações educacionais, o GDD assume dimensão adicional ao incorporar objetivos pedagógicos e estratégias de aprendizagem, estabelecendo pontes entre *design* de jogos e teorias educacionais (PAIVA *et al.*, 2022).

Benitti (2015) identifica cinco componentes essenciais em um GDD para aplicações educacionais, argumentando que a integração coerente destes elementos é determinante para o sucesso pedagógico da ferramenta:

- a) **Visão conceitual:** articulando propósito, público-alvo e objetivos de aprendizagem;
- b) **Mecânicas de jogo:** descrevendo sistemas interativos e sua correlação com conceitos a serem aprendidos;
- c) **Progressão e feedback:** detalhando estratégias de *scaffolding* e avaliação formativa;
- d) **Design de interface:** especificando *layouts*, fluxos de navegação e princípios de UX;
- e) **Arquitetura técnica:** documentando requisitos, tecnologias e estratégias de implementação.

A adaptação do formato tradicional de GDD para ferramentas didáticas de IHC apresenta desafios particulares, uma vez que o próprio documento deve exemplificar os princípios de usabilidade que serão objeto de estudo (FERREIRA *et al.*, 2014). Inspirando-se na proposta de aprendizagem aplicada de Jomsri, Prangchumpol e Ponsilp (2025), adota-se neste trabalho um modelo metacognitivo de GDD educacional, no qual cada seção do documento não apenas descreve aspectos do *software*, mas também explicita quais princípios de usabilidade estão sendo aplicados em sua concepção, conforme ilustrado no Quadro 2.

No desenvolvimento de ferramentas didáticas para o ensino de IHC, o GDD

Quadro 2 – Estrutura metacognitiva de GDD para aplicações de ensino de usabilidade

Seção do GDD	Conteúdo Convencional	Camada Metacognitiva
Visão Geral	Descrição do conceito e objetivos	Aplicação das heurísticas 2 (correspondência com mundo real) e 8 (design estético e minimalista)
Mecânicas	Sistemas de interação e regras	Aplicação das heurísticas 3 (controle e liberdade) e 7 (flexibilidade e eficiência)
Interface	Wireframes e fluxos de navegação	Aplicação das heurísticas 1 (visibilidade), 4 (consistência) e 6 (reconhecimento)
Feedback	Sistemas de resposta e progressão	Aplicação das heurísticas 5 (prevenção de erros) e 9 (recuperação de erros)
Suporte	Tutoriais e ajuda contextual	Aplicação da heurística 10 (ajuda e documentação)

Fonte: Adaptado de (NIELSEN, 1994) e (JOMSRI; PRANGCHUMPOL; POONSILP, 2025).

assume relevância particular ao documentar não apenas o produto final, mas também a arquitetura de interação, suas categorias e fluxos de navegação (PERIN; SILVA; VALENTIM, 2023), constituindo uma metadocumentação essencial para aplicações educacionais.

2.1.6 Acessibilidade Digital e Princípios POUR

A acessibilidade digital, entendida como a capacidade de sistemas computacionais serem utilizados por pessoas com diferentes habilidades e em diversos contextos, constitui princípio fundamental de *design* inclusivo e responsável (W3C, 2018). No contexto contemporâneo, onde a interação com sistemas digitais tornou-se essencial para acesso a serviços, educação, trabalho e participação social, garantir acessibilidade não configura apenas conformidade normativa, mas imperativo ético e social.

O W3C (World Wide Web Consortium), por meio das *Web Content Accessibility Guidelines* (WCAG 2.1) (W3C, 2018), estabeleceu os princípios POUR como *framework* conceitual para acessibilidade digital. Estes princípios organizam-se em quatro pilares fundamentais:

- a) **Perceptível (*Perceivable*)**: A informação e os componentes da interface devem ser apresentados de forma que os usuários possam

percebê-los. Isto implica oferecer alternativas textuais para conteúdo não textual, disponibilizar legendas e audiodescrições, criar conteúdo que possa ser apresentado de diferentes formas sem perda de informação, e facilitar a visualização e audição do conteúdo através de contraste adequado e controles de volume.

- b) **Operável (*Operable*)**: Os componentes da interface e a navegação devem ser operáveis. Este princípio enfatiza que toda funcionalidade deve estar disponível via teclado, que os usuários devem ter tempo suficiente para ler e usar o conteúdo, que o *design* não deve induzir convulsões ou reações físicas, e que mecanismos de navegação devem ser claros e consistentes.
- c) **Compreensível (*Understandable*)**: A informação e a operação da interface devem ser compreensíveis. Isto requer que textos sejam legíveis e compreensíveis, que páginas web apareçam e operem de forma previsível, e que usuários sejam auxiliados a evitar e corrigir erros através de mensagens claras e suporte contextual.
- d) **Robusto (*Robust*)**: O conteúdo deve ser robusto o suficiente para ser interpretado de forma confiável por uma ampla variedade de tecnologias assistivas. Este princípio assegura que, à medida que tecnologias evoluem, o conteúdo permanece acessível, exigindo conformidade com padrões estabelecidos e compatibilidade com *software* assistivo.

No contexto brasileiro, a acessibilidade digital é respaldada por marcos legais e normativos que estabelecem obrigatoriedade e diretrizes para implementação. A Lei Brasileira de Inclusão da Pessoa com Deficiência (LBI – Lei nº 13.146/2015) (BRASIL, 06/07/2015), conhecida como Estatuto da Pessoa com Deficiência, estabelece em seu Artigo 63 que é obrigatória a acessibilidade nos *sites* da internet mantidos por empresas com sede ou representação comercial no Brasil e por órgãos de governo, garantindo à pessoa com deficiência o acesso às informações disponíveis.

Complementarmente, o Modelo de Acessibilidade em Governo Eletrônico (eMAG) (BRASIL, 2014), desenvolvido pelo Governo Federal, constitui um conjunto de recomendações específicas para o contexto brasileiro, harmonizando-se com os padrões internacionais do WCAG enquanto contempla particularidades culturais, linguísticas e tecnológicas nacionais. O eMAG versão 3.1 organiza suas diretrizes em torno dos mesmos princípios POUR, adaptando-os à realidade dos serviços públicos digitais brasileiros.

A relação entre usabilidade e acessibilidade é intrinsecamente complementar, conforme argumenta Nielsen (1994). Enquanto a usabilidade preocupa-se com a facilidade de uso para todos os usuários, a acessibilidade foca especificamente em garantir que pessoas com deficiências possam utilizar sistemas de forma equivalente.

Práticas de *design* que promovem acessibilidade frequentemente resultam em melhor usabilidade geral, beneficiando todos os usuários através de navegação mais clara, *feedback* mais explícito e interfaces mais robustas.

No contexto educacional, o ensino de acessibilidade digital em cursos de Engenharia de Computação assume importância estratégica, preparando futuros profissionais para desenvolver sistemas inclusivos desde a concepção. A integração de princípios de acessibilidade em ferramentas didáticas para ensino de IHC, como a proposta neste trabalho, cria oportunidade de meta-aprendizado, onde estudantes não apenas estudam acessibilidade teoricamente, mas experimentam suas implicações práticas na própria ferramenta de ensino.

O Quadro 3 estabelece correspondências entre os princípios POUR de acessibilidade e as heurísticas de Nielsen, evidenciando a convergência conceitual entre ambos os *frameworks* e reforçando a complementaridade entre usabilidade e acessibilidade.

Quadro 3 – Correspondências entre princípios POUR e heurísticas de Nielsen

Princípio POUR	Heurísticas de Nielsen Relacionadas
Perceptível	H1 (Visibilidade do status), H8 (Design estético e minimalista)
Operável	H3 (Controle e liberdade), H7 (Flexibilidade e eficiência)
Compreensível	H2 (Correspondência sistema-mundo real), H4 (Consistência e padrões), H9 (Recuperação de erros)
Robusto	H4 (Consistência e padrões), H6 (Reconhecimento vs. memorização)

Fonte: Elaborado pelo autor com base em (W3C, 2018) e (NIELSEN, 1994), 2026.

2.1.7 Metodologias Ativas de Ensino

Metodologias ativas de ensino caracterizam-se pelo protagonismo do estudante no processo de aprendizagem, em oposição aos métodos tradicionais centrados na exposição do docente (JOMSRI; PRANGCHUMPOL; POONSILP, 2025). Entre as principais estratégias aplicáveis ao ensino de IHC destacam-se: aprendizagem baseada em problemas, projetos interdisciplinares, gamificação e avaliação entre pares.

A ferramenta proposta busca integrar elementos dessas metodologias, oferecendo ambiente propício à experimentação, à reflexão crítica e ao desenvolvimento de competências práticas em usabilidade e acessibilidade. Estas metodologias, quando combinadas com ferramentas tecnológicas apropriadas, facilitam o processo de ensino-aprendizagem em disciplinas técnicas, especialmente aquelas com forte

componente visual como IHC.

2.1.8 Aprendizagem Baseada em Jogos e Fundamentos Pedagógicos

A aprendizagem baseada em jogos (*game-based learning*) constitui abordagem pedagógica que utiliza jogos digitais como meio para promover aquisição de conhecimentos, desenvolvimento de habilidades e mudança de atitudes (HOLM; HUKE, 2020). Diferenciando-se da gamificação, que aplica elementos de jogos em contextos não lúdicos, a aprendizagem baseada em jogos envolve o uso de jogos completos – com narrativas, regras, desafios e sistemas de *feedback* – especificamente projetados ou adaptados para fins educacionais (RAMÍREZ RUIZ; VARGAS SANCHEZ; BOUDE FIGUEREDO, 2024).

O potencial pedagógico dos jogos digitais fundamenta-se em diversos princípios da teoria da aprendizagem contemporânea. Conforme Papert (1980), jogos proporcionam ambientes de aprendizagem construcionista, nos quais estudantes constroem conhecimento através da manipulação ativa de objetos e conceitos, recebendo *feedback* imediato sobre suas ações. Esta interatividade favorece ciclos rápidos de experimentação, reflexão e refinamento de estratégias, acelerando processos de aprendizagem.

A Taxonomia de Bloom (BLOOM *et al.*, 1956), amplamente utilizada em *design* instrucional, estabelece hierarquia de objetivos educacionais organizados em seis níveis cognitivos: lembrar, compreender, aplicar, analisar, avaliar e criar. Jogos educacionais bem projetados têm potencial de endereçar múltiplos níveis desta taxonomia simultaneamente. No contexto do ensino de usabilidade, por exemplo, um jogo pode requerer que estudantes lembrem as heurísticas de Nielsen, compreendam seu significado conceitual, apliquem-nas em cenários simulados, analisem interfaces reais identificando violações, avaliem a severidade dos problemas detectados e, eventualmente, criem propostas de redesign.

Tsay, Kofinas e Luo (2018) identifica cinco mecanismos pelos quais jogos digitais promovem aprendizagem efetiva:

- a) **Engajamento motivacional:** Elementos como desafios progressivos, sistemas de recompensa e narrativas envolventes aumentam a motivação intrínseca do estudante, sustentando esforço e persistência mesmo diante de dificuldades;
- b) **Prática contextualizada:** Jogos situam a prática de habilidades em contextos significativos e autênticos, facilitando transferência de conhecimento para situações reais;
- c) **Feedback imediato:** Respostas instantâneas às ações do jogador permitem ajustes rápidos de estratégias e consolidação de conhecimentos

corretos;

- d) **Progressão adaptativa:** Sistemas de dificuldade ajustável mantêm o estudante em zona de desenvolvimento proximal, onde desafios são suficientemente difíceis para estimular aprendizagem, mas não tão difíceis a ponto de gerar frustração;
- e) **Espaço seguro para erros:** Jogos criam ambientes onde falhar é parte natural do processo de aprendizagem, removendo ansiedades associadas a avaliações tradicionais e encorajando experimentação.

No contexto específico do ensino de IHC, jogos educacionais oferecem vantagens adicionais ao permitir simulação de cenários de interação diversos, visualização de consequências de decisões de *design* e experimentação de princípios de usabilidade em ambientes controlados (FERREIRA *et al.*, 2014). A natureza interativa dos jogos alinha-se naturalmente com o objeto de estudo da IHC, criando coerência metodológica entre meio e conteúdo.

A avaliação de aprendizagem em ambientes baseados em jogos requer abordagens específicas que considerem tanto resultados (*outcomes*) quanto processos. Paiva *et al.* (2022) propõe modelo de avaliação formativa integrada, no qual a performance do estudante no jogo – medida através de métricas como tempo de conclusão, número de tentativas, padrões de navegação e decisões tomadas – fornece dados objetivos sobre progressão de aprendizagem. Complementarmente, atividades reflexivas pós-jogo, como discussões e produções escritas, permitem avaliar compreensão conceitual profunda e capacidade de transferência de conhecimento.

O Quadro 4 estabelece correspondências entre os níveis cognitivos da Taxonomia de Bloom e mecânicas de jogo aplicáveis ao ensino de usabilidade, ilustrando como *design* de jogos educacionais pode ser sistematicamente estruturado para promover aprendizagem em diferentes níveis de complexidade cognitiva.

Resultados de estudos empíricos sobre efetividade de jogos educacionais para ensino de usabilidade têm sido consistentemente positivos. Benitti (2015), em pesquisa com estudantes de graduação em Ciência da Computação, identificou ganhos significativos em compreensão conceitual e capacidade de aplicação das heurísticas de Nielsen após uso de jogo educacional, comparado a grupo controle que recebeu apenas instrução tradicional. Similarmente, Jomsri, Prangchumpol e Poon-silp (2025) reportou aumento de engajamento e satisfação discente em cursos de IHC que incorporaram elementos de aprendizagem baseada em jogos.

No desenvolvimento da ferramenta didática proposta neste trabalho, os princípios de aprendizagem baseada em jogos orientam decisões de *design* pedagógico, desde a estruturação de cenários didáticos até a implementação de sistemas de progressão e *feedback*. A integração destes princípios com as heurísticas de Nielsen e os princípios de acessibilidade digital constitui abordagem holística que busca

Quadro 4 – Correspondências entre Taxonomia de Bloom e mecânicas de jogo para ensino de usabilidade

Nível de Bloom	Objetivo Cognitivo	Mecânica de Jogo Aplicável
1. Lembrar	Recordar heurísticas de Nielsen	Desafios de identificação, <i>flashcards</i> interativos
2. Compreender	Explicar significado de princípios	Narrativas com exemplos, tutoriais contextualizados
3. Aplicar	Usar heurísticas em cenários	Análise de interfaces simuladas, identificação de violações
4. Analisar	Comparar interfaces e identificar padrões	Cenários contrastantes (bom vs. mau UX), avaliações comparativas
5. Avaliar	Julgar severidade de problemas	Sistemas de pontuação, classificação de prioridades
6. Criar	Propor soluções de <i>design</i>	Modos criativos, <i>sandbox</i> de redesign

Fonte: Elaborado pelo autor com base em (BLOOM *et al.*, 1956) e (NIELSEN, 1994), 2026.

maximizar efetividade educacional da ferramenta.

2.2 Revisão Bibliográfica

2.2.1 Ensino de Interação Humano-Computador

A literatura sobre ensino de IHC evidencia avanços metodológicos na formação de profissionais aptos a projetar sistemas centrados no usuário. Segundo Jomsri, Prangchumpol e Poonsilp (2025), abordagens tradicionais, excessivamente teóricas, são insuficientes para o desenvolvimento das competências práticas necessárias à aplicação dos princípios de usabilidade. O uso de recursos digitais lúdicos e interativos tem se destacado como estratégia para fortalecer o ensino e a aprendizagem em IHC.

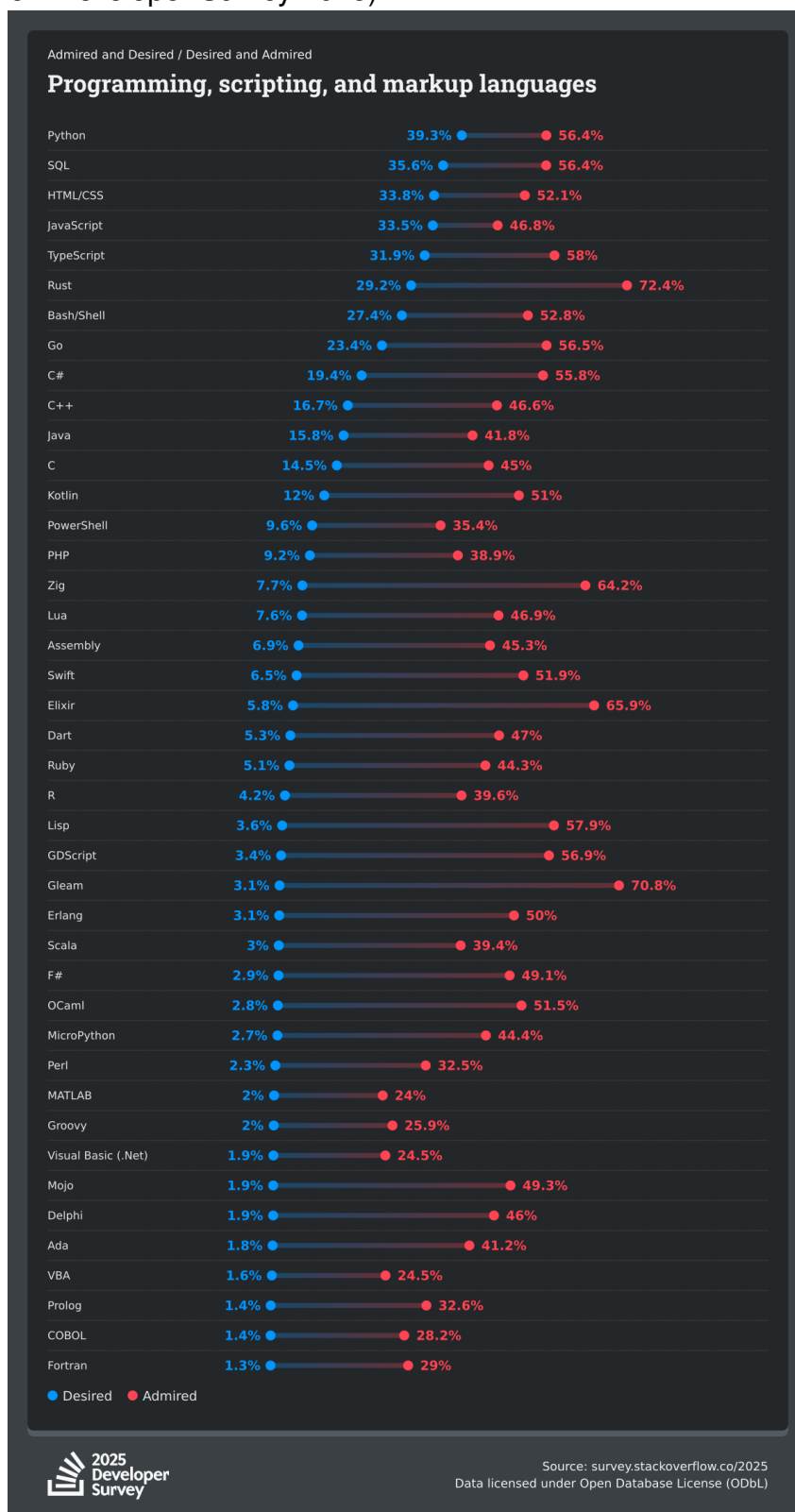
No Brasil, iniciativas como o UsabiliCity (FERREIRA *et al.*, 2014) demonstram a eficácia de jogos educacionais para comunicar propriedades de usabilidade por meio de analogias e dinâmicas interativas. Outros autores, como Paiva *et al.* (2022), ressaltam a importância da experiência do usuário para a aceitação de plataformas gamificadas de aprendizagem, reforçando a necessidade de aplicar os próprios princípios de IHC no desenvolvimento de ferramentas didáticas.

2.2.2 Ferramentas Didáticas Digitais em IHC

Diversos estudos apontam que o emprego de ferramentas digitais no ensino de IHC resulta em ganhos significativos de engajamento e aprendizagem (TSAY; KOFINAS; LUO, 2018). Segundo Ramírez Ruiz, Vargas Sanchez e Boude Figueredo (2024), elementos de gamificação aplicados ao contexto educacional aumentam o envolvimento dos estudantes, especialmente em conteúdos de maior abstração conceitual.

O desenvolvimento de sistemas de aprendizagem interativa, como o SQ-Latch (SELVESAKIS; KAZANIDIS, 2024), evidencia a efetividade de abordagens pedagógicas mediadas por ambientes visuais em contextos técnicos (Figura 3). Dados do *Stack Overflow Developer Survey 2025* (STACK OVERFLOW, 2025) indicam que o GDScript, linguagem nativa da Godot, apresentou índice de “desejabilidade” (*desired*) de 3,4% e de “admiração” (*admired*) de 56,9% entre os profissionais da área, conforme ilustrado na Figura 4. Embora se trate de linguagem relativamente recente, os índices observados evidenciam crescimento consistente na adoção e valorização do GDScript pela comunidade, reforçando a pertinência de sua utilização como linguagem principal em aplicações educacionais orientadas à inovação e à experimentação.

Figura 4 – Posicionamento do GDScript em índices de admiração e desejabilidade (Stack Overflow Developer Survey 2025)



Fonte: STACK OVERFLOW, 2025.

senta um desafio significativo, descrito por Jomsri, Prangchumpol e Poonsilp (2025) como uma transição crítica na formação técnica. Para mitigar este problema, sistemas que combinam interfaces visuais intuitivas com linguagens de *script* acessíveis, como GDScript da Godot Engine, têm demonstrado resultados promissores em ambientes educacionais (GONÇALVES, 2023).

2.2.4 Gamificação e Jogos Educacionais

A gamificação, entendida como a aplicação de elementos de jogos em contextos não lúdicos, tem se mostrado eficaz para aumentar o engajamento, a motivação e o desempenho de estudantes em ambientes de aprendizagem (RAMÍREZ RUIZ; VARGAS SANCHEZ; BOUDE FIGUEREDO, 2024). Para Holm e Huke (2020), técnicas como pontos, níveis, recompensas e *feedback* imediato contribuem para a criação de experiências educacionais dinâmicas, favorecendo a aprendizagem ativa e colaborativa.

No ensino de IHC, a gamificação pode ser utilizada para simular desafios reais de usabilidade, promover a experimentação de diferentes estratégias de *design* e estimular a reflexão crítica sobre escolhas de interface e acessibilidade.

O desenvolvimento de jogos educacionais para o ensino de usabilidade e acessibilidade constitui abordagem estratégica para a formação de competências em IHC (FERREIRA *et al.*, 2014; BENITTI, 2015). Jogos digitais permitem a contextualização de problemas, a experimentação controlada de soluções e a avaliação formativa do desempenho dos estudantes.

Projetos como UsabiliCity demonstram que a ludicidade e a interatividade favorecem a internalização de conceitos abstratos e a transferência de conhecimentos para situações práticas. A Figura 5 ilustra um exemplo de interface de jogo educacional voltado para o ensino de heurísticas de usabilidade.

Figura 5 – Interface do jogo educacional UsabiliCity



Fonte: Adaptado de Ferreira *et al.* (2014).

Uma análise conduzida por Ferreira *et al.* (2014) sobre a eficácia de ferra-

mentas visuais para o ensino de usabilidade revelou ganhos significativos em engajamento estudantil e retenção de conhecimento em comparação com métodos tradicionais baseados exclusivamente em exposição teórica. Entretanto, o mesmo estudo alerta para a necessidade de estratégias de transição que permitam aos estudantes evoluir gradualmente para ambientes de programação textual quando necessário.

2.2.5 Estado da Arte em Ferramentas Didáticas para IHC

Diversos trabalhos recentes abordam o desenvolvimento de jogos, ferramentas e recursos digitais para o ensino de IHC, cada qual com diferentes propostas metodológicas e tecnológicas. O Quadro 5 apresenta uma síntese comparativa dos principais projetos identificados na literatura, destacando aspectos como paradigma de desenvolvimento, heurísticas abordadas e estratégias pedagógicas utilizadas.

Quadro 5 – Estado da arte: Síntese de ferramentas didáticas digitais para ensino de IHC

Ferramenta	Paradigma	Recursos	Estratégia Pedagógica
UsabiliCity (FERREIRA <i>et al.</i> , 2014)	Jogo digital	Heurísticas de Nielsen	Analogias entre problemas de cidade e de interface
UsabilityGame (BENITTI, 2015)	Simulação	Requisitos, prototipação e avaliação	Ciclo completo de engenharia de usabilidade em empresa fictícia
HCI Gamification App (JOMSRI; PRANG-CHUMPOL; POONSILP, 2025)	Gamificação	Princípios de IHC	Missões, níveis e simulações de interfaces
UX Detective (este trabalho)	GDScript (Go-dot)	Heurísticas de Nielsen e acessibilidade	Marcação direta de violações em interfaces realistas com dificuldade progressiva

Fonte: Elaborado pelo autor com base na literatura, 2026.

O Quadro 6 compara ferramentas didáticas digitais para o ensino de conceitos técnicos, situando o UX Detective em relação a alternativas consolidadas na literatura.

O Quadro 7 sintetiza a correspondência entre os objetivos específicos estabelecidos no Capítulo 1 e os resultados obtidos ao longo do desenvolvimento da

Quadro 6 – Ferramentas didáticas para ensino técnico e contexto de uso

Ferramenta	Público-alvo	Contexto	Principais recursos
Scratch	Ensino fundamental e médio	Introdução à lógica e programação	Programação visual, gamificação
SQLatch	Ensino superior	Ensino de SQL	Interface visual, <i>feedback</i> automático
UsabiliCity	Ensino superior	Usabilidade por analogia	Analogias cidade-interface, <i>feedback</i> imediato
UX Detective (este trabalho)	Ensino superior e profissionais de usabilidade/acessibilidade	IHC, usabilidade e acessibilidade	Godot Engine + GDScript, avaliação heurística direta em interfaces WAI-ARIA

Fonte: Adaptado de Perin, Silva e Valentim (2023) e Selvesakis e Kazanidis (2024).

ferramenta.

Quadro 7 – Relação entre objetivos do projeto e resultados obtidos

Objetivo	Resultado Obtido
Levantamento bibliográfico	Fundamentação teórica consolidada sobre IHC, heurísticas de Nielsen, acessibilidade WAI-ARIA e <i>game-based learning</i>
Modelagem de cenários didáticos	Catálogo de 10 interfaces associadas a <i>violations</i> H1–H10, com dificuldade Fácil implementada
Implementação da ferramenta	Modo Normal funcional (fluxo menu → avaliação → resultados) e modos Desafio/Score projetados
Arquitetura de <i>software</i>	Arquitetura de sistemas atômicos com 6 <i>autoloads</i> e catálogos baseados em <i>Resources</i> .tres
Documentação do processo	Monografia, diagramas de fluxo e arquitetura, e planejamento de registro no NIT

Fonte: Elaborado pelo autor, 2026.

3 METODOLOGIA

Esta pesquisa adota uma abordagem de desenvolvimento experimental para a criação de uma ferramenta didática digital, fundamentada em técnicas de *design* centrado no usuário e metodologias ágeis. O processo metodológico foi estruturado em etapas interdependentes, garantindo rigor científico e alinhamento com os objetivos pedagógicos propostos. A elaboração do trabalho observou as diretrizes do Manual de Normalização de Trabalhos Acadêmicos do IFMG (INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE MINAS GERAIS (IFMG), 2020) e o regulamento institucional vigente para Trabalhos de Conclusão de Curso (INSTITUTO FEDERAL DE MINAS GERAIS, 13/01/2025).

3.1 Caracterização da Pesquisa

A investigação é pesquisa aplicada (GIL, 2019), voltada à solução de um problema educacional concreto: o ensino das heurísticas de usabilidade de Nielsen por métodos ativos e contextualizados.

Quanto aos objetivos, classifica-se como exploratória-descritiva (PRODANOV; FREITAS, 2013). O caráter exploratório se evidencia na coleta e análise de interfaces existentes que servem de base para os cenários didáticos, identificando violações reais das heurísticas. O aspecto descritivo está no registro sistemático do processo de desenvolvimento e nas decisões de *design* pedagógico e arquitetural.

Do ponto de vista da abordagem, a pesquisa é predominantemente qualitativa, pois, conforme Lakatos e Marconi (2021), preocupa-se com aspectos da realidade que não podem ser quantificados, centrando-se na compreensão e explicação da dinâmica de aprendizagem proporcionada pela ferramenta. A análise qualitativa permeia tanto o processo de desenvolvimento quanto a avaliação da adequação pedagógica da solução proposta.

Quanto aos procedimentos técnicos, a pesquisa adota o método de desenvolvimento tecnológico, combinando revisão bibliográfica sobre IHC e aprendizagem baseada em jogos, desenvolvimento experimental de *software* educacional, e avaliação formativa iterativa. O desenvolvimento segue o paradigma construcionista, no qual o próprio processo de criação da ferramenta constitui oportunidade de aprendizagem e investigação dos princípios de usabilidade que se pretende ensinar, estabelecendo um meta-processo reflexivo onde o meio (ferramenta didática) incorpora e exemplifica seu próprio fim (ensino de usabilidade).

3.2 Convenções e Glossário Técnico

A descrição metodológica e arquitetural deste capítulo recorre a termos técnicos de três domínios: a *Godot Engine* e a linguagem GDScript, padrões clássicos de projeto de *software* e a terminologia da matriz de confusão usada na avaliação de marcações de violação. O Quadro 8 reúne as definições; os termos não serão redefinidos no restante do documento.

Quadro 8 – Glossário técnico: termos da Godot Engine, padrões arquiteturais e matriz de confusão

Termo	Definição
<i>Godot Engine</i>	
autoload	<i>Script</i> ou cena registrada globalmente que se mantém ativa durante toda a execução, equivalente ao padrão <i>singleton</i> .
signal	Mecanismo de comunicação assíncrona entre nós, implementação nativa do padrão <i>Observer</i> .
node	Unidade básica da árvore de cenas, com responsabilidade específica (renderização, lógica, entrada).
.tscn	Extensão de arquivos de cena (<i>scene</i>) do Godot, contendo a árvore de nós serializada.
.tres	Extensão de arquivos de recurso (<i>resource</i>) do Godot, contendo dados serializados que parametrizam o comportamento do código.
Vector2	Tipo nativo do Godot que representa um par ordenado de números de ponto flutuante, usado para posições, escalas e direções.
modulate	Propriedade de nós visuais que multiplica a cor de renderização nos canais RGBA, usada para controlar opacidade e tinta.
PackedScene	Recurso serializável que representa uma cena instanciável em tempo de execução.
<i>Padrões arquiteturais</i>	
<i>Observer</i>	Padrão de projeto no qual um sujeito notifica observadores sobre mudanças de estado por meio de eventos. No UX Detective, materializa-se pelo sistema de <code>signal</code> .
<i>Singleton</i>	Padrão que garante instância única de uma classe acessível globalmente; no Godot, implementado por meio de <code>autoload</code> .
APG	<i>ARIA Authoring Practices Guide</i> : catálogo do W3C de padrões de componentes acessíveis, definindo papéis, propriedades e teclas de navegação esperadas.
<i>Avaliação de marcações (matriz de confusão)</i>	
TP (<i>True Positive</i>)	Violação real corretamente marcada pelo usuário.
FP (<i>False Positive</i>)	Marcação do usuário em ponto da interface que não corresponde a uma violação.
FN (<i>False Negative</i>)	Violação real que o usuário deixou de marcar.
TN (<i>True Negative</i>)	Ponto da interface sem violação que o usuário corretamente não marcou.

Fonte: Elaborado pelo autor, 2026.

3.3 Ferramentas Tecnológicas

3.3.1 Godot Engine como Ambiente de Desenvolvimento

A Godot Engine foi selecionada como plataforma base de desenvolvimento devido à sua arquitetura flexível e adequação ao contexto educacional. A *engine* é organizada em torno de cinco pilares conceituais que estruturam toda a construção de aplicações: **nós** (*Nodes*), **cenas** (*Scenes*), **sinais** (*Signals*), **Resources** e a hierarquia **pai-filho** expressa na árvore de cenas. Estes pilares foram explorados integralmente no desenvolvimento do UX Detective, conforme detalhado a seguir.

3.3.1.1 Hierarquia de Nós e Árvore de Cenas

Os nós da Godot organizam-se em árvore hierárquica pai-filho, estrutura particularmente vantajosa para *software* educacional: (i) transformações aplicadas ao nó pai (posição, escala, visibilidade, *disabled*) propagam-se aos filhos, simplificando a manipulação em lote; (ii) a árvore pode ser inspecionada e manipulada visualmente no editor, oferecendo modelo mental concreto ao estudante; e (iii) cada nó executa função específica, favorecendo composição de responsabilidades. A Figura 6 ilustra um recorte da árvore de nós utilizada na interface de avaliação do UX Detective.

3.3.1.2 Control Nodes e Containers

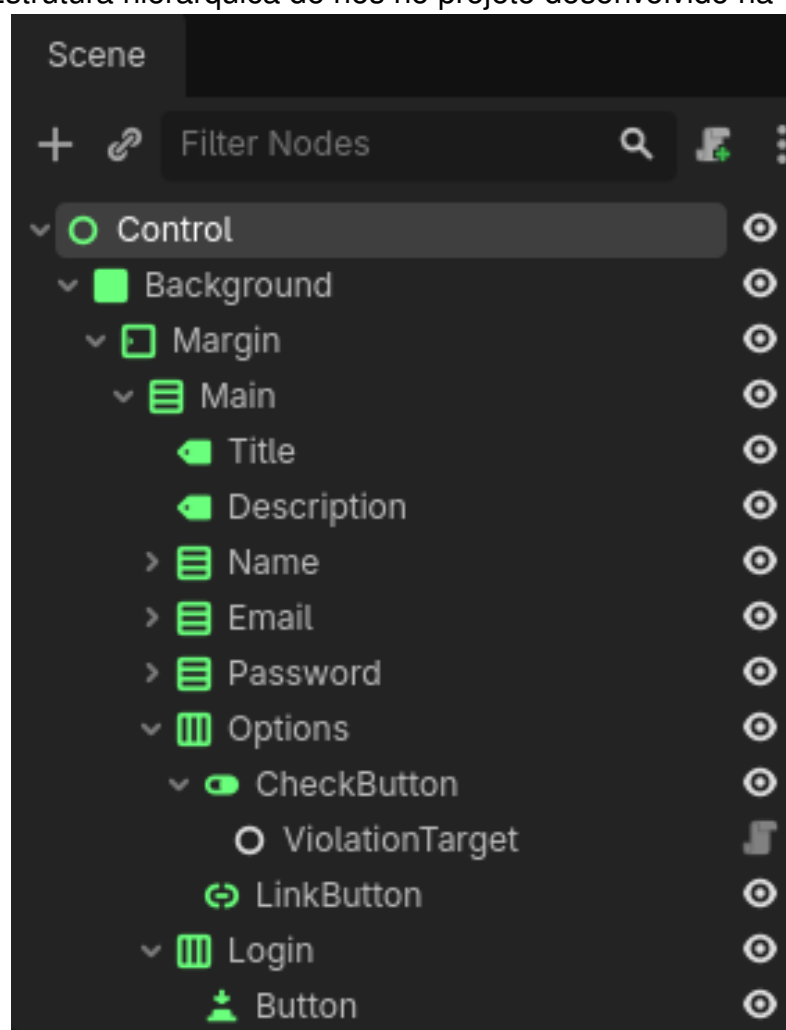
A Godot Engine oferece uma família dedicada de nós denominada *Control*, cuja finalidade é a construção de interfaces de usuário bidimensionais. Destes, dois grupos foram amplamente utilizados na ferramenta.

3.3.1.2.1 Controles de interação.

Estes nós implementam os elementos com os quais o usuário interage diretamente e são, por consequência, partes integrais da experiência de usuário que o UX Detective se propõe a ensinar. Os *controls* mais recorrentes no projeto são:

- **Button**: botão padrão, utilizado em menus, botões de ação e seleção de heurísticas na prancheta;
- **CheckButton** e **CheckBox**: alternadores binários para opções de configuração e alternância de estados;
- **LineEdit** e **TextEdit**: campos de entrada de texto simples e multilinha, empregados nos formulários de autenticação e nos formulários de cenários;

Figura 6 – Estrutura hierárquica de nós no projeto desenvolvido na Godot Engine



Fonte: Elaborado pelo autor, 2026.

- **ProgressBar**: indicador de progresso, utilizado no painel esquerdo para comunicar avanço na avaliação;
- **Label** e **RichTextLabel**: exibição de textos simples e com formatação BBCode, utilizada nas explicações didáticas e na tela de resultados;
- **OptionButton** e **SpinBox**: seleção de opções em lista e entrada numérica, utilizados nas configurações;
- **ScrollContainer** e **TabContainer**: contêineres interativos para organização de conteúdo extenso e navegação entre abas.

3.3.1.2.2 Containers de layout.

Os *Container nodes* fornecem *layouts* flexíveis e escalonáveis que se adaptam automaticamente a diferentes resoluções e proporções de tela. Entre os mais utilizados no projeto destacam-se `VBoxContainer` e `HBoxContainer` (organização vertical e horizontal com distribuição automática), `GridContainer` (arranjo tabular), `MarginContainer` (margens consistentes), `CenterContainer` (centralização responsiva) e `AspectRatioContainer` (preservação de proporção). Esta abordagem automatiza a adaptação da interface, alinhando-se aos princípios de *design* responsivo e favorecendo a manutenção evolutiva.

3.3.1.3 Sistema de Sinais (*Signals*)

Os sinais conectam-se em escopo local (dentro da mesma cena) ou global (via *autoload* `EventBus`). O uso intensivo de sinais no UX Detective foi decisão arquitetural deliberada: elimina o acoplamento direto via `get_node()`, facilita testes unitários com o *framework* GUT e permite que cada componente seja desenvolvido, substituído e testado isoladamente.

Para tornar visível o grafo completo de *publish/subscribe* da aplicação e permitir raciocínio sobre efeitos em cascata, o Apêndice B apresenta um diagrama agregado dos tópicos de sinais do `EventBus`, seus emissores e seus ouvintes. O diagrama distingue emissores, canais de sinal e ouvintes (*listeners*), funcionando como antídoto ao chamado *observer spaghetti* ao tornar rastreáveis as cadeias de efeitos a cada emissão.

3.3.1.4 Cenas como Unidade Modular

Na Godot, uma cena é uma árvore de nós reutilizável, salva em disco como um arquivo `.tscn`. Uma cena pode ser *instanciada* como filha de outra cena, formando composições dinâmicas. Esta propriedade foi amplamente explorada no projeto: a *Ga-*

meplayInterface, por exemplo, é composta pela instanciação de múltiplas cenas menores (*top_bar.tscn*, *left_panel.tscn*, *bottom_bar.tscn*, *right_panel.tscn* e *zoom/*), cada uma com responsabilidades específicas. Essa modularidade permite que diferentes interfaces de cenário e diferentes modos de jogo reutilizem a mesma estrutura, alterando apenas as partes necessárias e respeitando o princípio da segregação de responsabilidades (MARTIN, 2003).

3.3.1.5 *Resources* para Dados Padronizados

Os *Resources* são objetos serializáveis (armazenados em arquivos *.tres* ou *.res*) que permitem definir estruturas de dados tipadas, editáveis diretamente no inspetor da Godot e versionáveis em sistemas de controle de versão como o Git. No UX Detective, os *Resources* constituem a espinha dorsal do conteúdo didático: cada **violação** é um *ViolationData*, cada **cenário** é um *ScenarioData*, e as **dicas progressivas** são recursos vinculados a cada cenário. Esta abordagem separa o conteúdo pedagógico da lógica de código, viabilizando que docentes e pesquisadores possam criar, editar ou traduzir cenários inteiramente pelo inspetor visual, sem escrever uma única linha de GDScript.

O Apêndice C apresenta o diagrama entidade-relacionamento dos *Resources* da ferramenta, evidenciando sua natureza *data-driven*: o *ScenarioData* agrega *HintData* e referencia uma *PackedScene* com *ViolationTarget* contendo a biblioteca de *ViolationData* injetáveis por cenário.

O Quadro 9 sintetiza os cinco pilares conceituais da Godot Engine e seu emprego direto no UX Detective.

3.3.2 *GDScript* como *Linguagem de Desenvolvimento*

O desenvolvimento da ferramenta utiliza GDScript como linguagem principal de programação. GDScript é uma linguagem de *script* de alto nível, com tipagem opcional e sintaxe inspirada em Python, projetada especificamente para a Godot Engine. Suas principais vantagens para o contexto educacional incluem:

- a) **Sintaxe acessível:** a sintaxe clara e concisa do GDScript reduz a carga cognitiva associada à programação, permitindo foco nos conceitos de usabilidade e *design* de interfaces;
- b) **Integração nativa:** por ser a linguagem nativa da Godot Engine, oferece acesso direto a todas as funcionalidades do motor, incluindo o sistema de nós, sinais e *containers*;
- c) **Tipagem opcional:** GDScript oferece suporte a anotações de tipo (*type hints*), permitindo detecção de erros em tempo de desenvolvimento e

Quadro 9 – Pilares conceituais da Godot Engine e sua aplicação no UX Detective

Pilar	Conceito	Uso no UX Detective
Nós (<i>Nodes</i>)	Unidade atômica de funcionalidade, organizada em árvore pai-filho	Composição de interfaces, controladores e <i>managers</i>
Cenas (<i>Scenes</i>)	Árvore de nós reutilizável salva em <code>.tscn</code>	<i>GameplayInterface</i> composta por cenas menores (barras, painéis, prancheta)
Sinais (<i>Signals</i>)	Comunicação assíncrona baseada no padrão <i>Observer</i>	Desacoplamento entre <i>managers</i> , <i>controllers</i> e UI via <i>EventBus</i>
<i>Resources</i>	Dados tipados, serializáveis e editáveis no inspetor	<i>ViolationData</i> , <i>ScenarioData</i> e <i>HintData</i>
Hierarquia pai-filho	Propagação automática de transformações e estados	Interfaces compostas que herdam configuração do <i>container</i>

Fonte: Elaborado pelo autor, 2026.

melhorando a manutenibilidade do código, conforme recomendado por Pressman e Maxim (2016) para projetos de *software* educacional;

- d) **Documentação abrangente:** a documentação oficial da Godot Engine fornece exemplos detalhados e tutoriais para GDScript, facilitando o aprendizado e a resolução de problemas.

3.3.2.1 Mecanismos de Extensibilidade

Um aspecto particularmente relevante do GDScript é a possibilidade de **estender** nós e recursos nativos da Godot Engine por meio da palavra-chave `extends`. Qualquer *script* pode herdar de um tipo nativo (como `Control`, `Node`, `Resource` ou qualquer `Control` específico, como `Button`) e personalizar sua lógica e seus atributos. Este mecanismo foi essencial no UX Detective para construir a hierarquia de componentes acessíveis `AriaComponent`, que estende `Control` e é, por sua vez, estendida por componentes especializados como `AriaButton`, `AriaCheckbox` e `AriaDialog`.

Complementarmente, GDScript disponibiliza anotações (*annotations*) que integram o *script* ao editor da Godot:

- `@export`: torna uma propriedade visível e editável no inspetor da Godot. Esta anotação é fundamental para separar código e conteúdo: `@export var scenario: ScenarioData` permite que um cenário seja atribuído visualmente, sem necessidade de modificar o *script*;
- `@export_range`, `@export_enum`, `@export_file`: variantes que restringem o

- tipo de valor aceitável, habilitando *widgets* de inspetor adequados (*sliders*, *combos*, *seletores de arquivo*);
- `@onready`: garante que uma referência seja resolvida apenas após a cena estar pronta, simplificando inicialização;
- `@tool`: permite que o *script* seja executado no próprio editor, viabilizando ferramentas de autoria personalizadas.

3.3.2.2 Documentação *Inline* e *Rich Text* no Editor

GDScript suporta comentários de documentação (iniciados por `##`) que são renderizados pelo editor da Godot como documentação em *rich text*. Ao posicionar um comentário `##` imediatamente antes de uma classe, função, sinal ou variável exportada, o texto passa a ser exibido no *tooltip* do inspetor e no painel de ajuda do editor, incluindo formatação em Markdown e referências cruzadas a outros membros. Esta capacidade foi explorada no UX Detective para documentar `class_name`, `signal` e `@export` de forma auto-explicativa, acelerando a familiarização de novos contribuidores com a base de código e servindo como modelo pedagógico de boas práticas de documentação.

3.3.2.3 Nós Únicos (%UniqueName)

Na Godot 4.x, um nó pode ser marcado como *unique* dentro de sua cena, passando a ser referenciável pela sintaxe `%NomeDoNo` a partir de qualquer outro *script* contido na mesma cena. Este recurso elimina dependências frágeis de caminhos relativos (`$../.../Panel/Button`) e reduz a probabilidade de quebras quando a árvore de nós é reorganizada. No UX Detective, nós críticos como `%ClipboardController`, `%MarkerTool` e `%ProgressBar` são marcados como únicos, permitindo que *controllers* e *helpers* acessem-nos de forma declarativa e resiliente a refatorações de *layout*.

3.3.2.4 Convenções de Codificação Adotadas

No desenvolvimento deste projeto, adotaram-se convenções rigorosas de codificação para garantir legibilidade e manutenibilidade:

- **Tipagem estática**: todas as variáveis, parâmetros de função e valores de retorno são explicitamente tipados (`var score: int = 0, func calculate_result() -> bool`);
- **Nomenclatura de classes**: uso de `class_name` para identificação explícita de classes reutilizáveis;
- **Documentação *inline***: utilização de comentários `##` para produzir docu-

mentação em *rich text* diretamente no editor;

- **Padrões de nomenclatura:** *snake_case* para funções e variáveis, *PascalCase* para classes e *UPPER_SNAKE_CASE* para constantes.

Estas práticas alinham-se aos princípios de Engenharia de Software propostos por Martin (2003), promovendo código limpo, testável e de fácil compreensão.

3.3.3 Ferramentas de Design e Prototipação

Para concepção dos *mockups* de interface, criação dos *assets* visuais e documentação gráfica da arquitetura, foram utilizadas as seguintes ferramentas:

- **Canva:** ferramenta *online* de *design* gráfico utilizada para criação de *mockups* de tela que ilustraram o *user flow* da aplicação ainda na fase de concepção. Esta etapa mostrou-se decisiva por três razões: (i) possibilitou validar, antes da implementação em Godot, o posicionamento e a hierarquia dos elementos de interface; (ii) forneceu subsídios concretos para a definição da arquitetura modular (identificação de *panels*, *bars*, *controllers*); e (iii) consolidou a identidade visual da aplicação (paleta cromática, tipografia, estilo dos ícones). Os *mockups* foram posteriormente adotados como referência durante a implementação das cenas Godot correspondentes.
- **Mermaid Chart:** utilizado para elaboração de diagramas conceituais, fluxogramas de interação, diagramas de classes e diagramas de *sequence*. A sintaxe declarativa baseada em texto facilita o versionamento dos diagramas junto ao código-fonte, a integração com ambientes de desenvolvimento e a geração de visualizações consistentes, permitindo rápidas iterações nos modelos da aplicação.
- **Kenney Cursor Pack e Kenney Emotes Pack** (licenças Creative Commons Zero — CC0) (KENNEY, 2024): bibliotecas de *assets* utilizadas para os cursores personalizados e ícones de emoção/*feedback* empregados na interface. A escolha justifica-se pela gratuidade, ausência de restrições de licenciamento e identidade visual consistente, compatível com o contexto educacional da aplicação.
- **Lucide** (LUCIDE, 2024): biblioteca de ícones de código aberto que fornece mais de 1.600 arquivos vetoriais (SVG) destinados a projetos digitais e impressos. Foi empregada para os ícones das dez heurísticas de Nielsen exibidos na prancheta de avaliação, bem como para diversos ícones de navegação e ação da aplicação. A uniformidade estética e a cobertura ampla de símbolos contribuíram para a consistência visual entre componentes, requisito da heurística 4 (Consistência e Padrões).

- **Prototipação em papel:** técnica tradicional de esboço manual utilizada nas fases iniciais de *design* para exploração rápida de conceitos de interface e interação. Esta abordagem permitiu iterações ágeis e de baixo custo antes da migração para os *mockups* digitais no Canva.

3.3.4 Arquitetura de Sistemas Atômicos

A arquitetura de *software* adotada no projeto fundamenta-se no padrão de Sistemas Atômicos (*Atomic Systems*), caracterizado pela decomposição da aplicação em componentes autocontidos, de responsabilidade única, comunicando-se por sinais. Esta abordagem alinha-se aos princípios SOLID propostos por Martin (2003), especialmente ao princípio da responsabilidade única (*Single Responsibility Principle*).

Os princípios fundamentais desta arquitetura incluem:

- a) **Responsabilidade única:** cada sistema ou componente possui finalidade claramente definida, facilitando compreensão, teste e manutenção isolada;
- b) **Comunicação por sinais:** a comunicação entre sistemas ocorre exclusivamente através de sinais, implementando o padrão *Observer* (GAMMA *et al.*, 1994) de forma nativa por meio do mecanismo de *signals* da Godot Engine;
- c) **Ausência de acoplamento direto:** os sistemas não referenciam-se mutuamente através de chamadas diretas (`get_node()`) ou armazenamento de referências, eliminando dependências rígidas;
- d) **Composição sobre herança:** funcionalidades complexas são construídas pela composição de componentes simples, promovendo reutilização e flexibilidade.

A arquitetura do UX Detective aplica esses princípios em três camadas complementares: *autoloads* globais, *Resources* de conteúdo e *managers/controllers/helpers* de interface.

3.3.4.1 Autoloads (Singletons Globais)

O UX Detective define seis *autoloads*, cada um com responsabilidade exclusiva:

- **EventBus:** barramento de eventos global que centraliza a declaração e o roteamento de sinais entre sistemas. Implementa o padrão *Mediator* (GAMMA *et al.*, 1994) e atua como canal de comunicação assíncrona sem armazenar estado;
- **SceneManager:** gerenciador centralizado de ciclo de vida de cenas, res-

- ponsável por carregamento, transição e histórico de navegação, oferecendo transições visuais (*fade*, *slide*);
- **GameSession**: mantém o estado da sessão de jogo corrente (usuário, modo ativo, cenário em curso, configuração de dificuldade) e coordena o início e o encerramento de avaliações;
- **ProgressTracker**: registra o progresso de aprendizagem do usuário (heurísticas completadas, classificação por estrelas, desbloqueio de níveis), persistindo dados entre sessões;
- **ThemeManager**: centraliza a aplicação de temas visuais (cores, fontes, estilos) e alterna entre modos visuais, mantendo consistência estética entre cenas e possibilitando preferências de acessibilidade;
- **CursorManager**: controla o cursor ativo (padrão, *marker*, *eraser*) durante as avaliações, refletindo o modo de interação selecionado na prancheta.

3.3.4.2 *Resources* para Conteúdo Didático

O conteúdo pedagógico foi encapsulado em *Resources* .tres tipados, editáveis no inspetor da Godot, sem necessidade de manipulação de código. Os principais tipos de recurso são:

- **ViolationData**: descreve uma violação heurística, incluindo identificador, heurística associada (H1–H10), severidade, alterações de propriedade a aplicar sobre o componente-alvo, descrição educacional e critério WCAG relacionado;
- **ScenarioData**: define um cenário de avaliação, referenciando a interface a ser carregada, a lista de *ViolationData* aplicáveis, as dicas progressivas associadas, a dificuldade e o título exibido ao usuário;
- **HintData**: descreve dicas progressivas vinculadas a um cenário ou violação, organizadas em níveis cumulativos de especificidade;
- **ScenarioCatalog** e **InterfaceCatalog**: recursos agregadores que listam, respectivamente, todos os cenários disponíveis e todas as interfaces implementadas, permitindo seleção dinâmica em tempo de execução.

O Quadro 10 sintetiza os *autoloads* e *Resources* e seus papéis arquiteturais.

A integração entre conteúdo didático estático (descrito por *Resources*) e estado dinâmico de execução (mantido por *autoloads*) materializa-se em um modelo de dados unificado apresentado no Apêndice D. Esse modelo incorpora as entidades de persistência que sustentam os modos de jogo (USER_DATA, PROFILE_ENTRY, PROGRESS_ENTRY, MATCH) e as tabelas de mapeamento *many-to-many* (SCENARIO_VIOLATION_MAP, SCENARIO_HINT_MAP), viabilizando reuso de violações entre

Quadro 10 – Autoloads e Resources da arquitetura de sistemas atômicos

Componente	Tipo	Responsabilidade
EventBus	Autoload	Roteamento global de sinais entre sistemas
SceneManager	Autoload	Carregamento, transição e histórico de cenas
GameSession	Autoload	Estado da sessão corrente (usuário, modo, cenário, dificuldade)
ProgressTracker	Autoload	Registro e persistência do progresso de aprendizagem
ThemeManager	Autoload	Aplicação e alternância de temas visuais
CursorManager	Autoload	Troca de cursor conforme ferramenta de interação
ViolationData	Resource	Dados de uma violação heurística injetável
ScenarioData	Resource	Configuração de um cenário (interface, violações, dicas)
HintData	Resource	Dicas progressivas vinculadas a cenário/violação
ScenarioCatalog	Resource	Agregador de cenários disponíveis
InterfaceCatalog	Resource	Agregador de interfaces ARIA implementadas

Fonte: Elaborado pelo autor, 2026.

cenários sem duplicação de dados.

3.3.4.3 *Managers, Controllers e Helpers*

A camada de apresentação é composta por três categorias de objetos, cada qual com responsabilidade arquitetural distinta:

- **Managers:** coordenam ciclos de vida e regras de negócio transversais. São exemplos o `ValidationManager` (classificação de marcações em verdadeiros/falsos positivos e negativos) e o `AssessmentSystem` (cálculo de pontuação, estrelas e níveis de desempenho);
- **Controllers:** intermediam a lógica da interface com os sistemas subjacentes. O `GameplayInterfaceController` orquestra a composição da interface de avaliação, enquanto o `ClipboardController` gerencia o estado da prancheta e a seleção de heurísticas;
- **Helpers:** fornecem funções utilitárias reutilizáveis, como carregamento e povoamento das telas de resultados com dados vindos dos recursos de cenário, transição entre fases de avaliação e formatação de *feedback* didático.

Esta arquitetura facilita a compreensão do código por estudantes, permite testes unitários isolados e simplifica a manutenção evolutiva e a adição de novos cenários sem alterar as camadas estáveis.

3.3.5 *Framework de Testes GUT*

Para garantir qualidade e confiabilidade do *software* desenvolvido, adotou-se a prática de testes unitários automatizados utilizando o *framework* GUT (*Godot Unit Test*). Conforme Pressman e Maxim (2016), testes automatizados constituem prática essencial em projetos de Engenharia de Software, permitindo detecção precoce de defeitos e facilitando manutenção evolutiva.

O GUT é um *framework* de testes específico para Godot Engine, oferecendo:

- **Sintaxe declarativa:** Estrutura de testes baseada em convenções claras (`test_`, `before_each`, `after_each`);
- **Asserções especializadas:** Métodos de validação otimizados para tipos de dados da Godot Engine (`assert_eq`, `assert_true`, `assert_signal_emitted`);
- **Isolamento de testes:** Cada teste executa em contexto isolado, evitando efeitos colaterais entre casos de teste;
- **Relatórios detalhados:** Saída formatada identificando testes bem-

sucedidos, falhos e ignorados.

No desenvolvimento deste projeto, estabeleceu-se meta de cobertura de testes de no mínimo 85% do código funcional, priorizando:

- a) Lógica de negócio dos sistemas educacionais (ScenarioManager, AssessmentSystem, NarrationController, ProgressTracker);
- b) Validação de entradas e processamento de respostas do usuário;
- c) Comunicação via sinais entre componentes;
- d) Funcionalidades críticas de acessibilidade.

A abordagem de desenvolvimento orientada a testes (*Test-Driven Development*) foi parcialmente adotada, especialmente para componentes de lógica complexa, garantindo que requisitos funcionais sejam validados sistematicamente ao longo do desenvolvimento.

3.4 Métodos Aplicados

3.4.1 Engenharia de Requisitos

O levantamento e especificação de requisitos seguiu uma abordagem centrada no usuário, priorizando as necessidades educacionais do público-alvo (estudantes de IHC). Os requisitos foram categorizados em:

- a) **Requisitos funcionais:** especificação das funcionalidades necessárias para o ensino dos princípios de usabilidade, incluindo a componentização da interface de avaliação, os sistemas de violação, cenários, pontuação e os modos de jogo (Tutorial, Normal, Desafio, Score);
- b) **Requisitos não funcionais:** definição de parâmetros de usabilidade, acessibilidade, desempenho e portabilidade da ferramenta, garantindo que ela própria exemplifique as boas práticas que pretende ensinar;
- c) **Requisitos pedagógicos:** estabelecimento de critérios para avaliação da eficácia didática, como progressão de aprendizagem, *feedback* formativo e avaliação diagnóstica.

De forma sintética, os requisitos funcionais contemplaram: (i) a criação das principais telas de navegação (menus) e da *GameplayInterface*, interface de avaliação heurística; (ii) a componentização da *GameplayInterface* como composição de cenas menores dedicadas a funções específicas (barra superior, painel esquerdo, barra inferior e painel direito/prancheta); (iii) a implementação do sistema de violações para armazenar os dados, identificar os componentes a serem violados e associá-los à prancheta; (iv) a implementação do sistema de cenários, funcionando como catálogo que associa interfaces a violações e as carrega dinamicamente na *GameplayInterface*; (v) o sistema de pontuação e *feedback* baseado em cenário e desempenho do

usuário; (vi) o sistema de dicas progressivas baseadas em *Resource*; (vii) o catálogo de componentes aderentes à ARIA para construção das interfaces; (viii) o catálogo de interfaces utilizáveis na avaliação; (ix) o modo Normal com as três categorias de dificuldade (Fácil, Médio, Difícil); (x) o tutorial interativo sequencial com cinco lições didáticas sobre as heurísticas e o uso da aplicação; (xi) o sistema de obstáculos e *power-ups* para os modos Desafio e Score; (xii) a implementação dos modos Desafio e Score com salvamento de pontuações e *leaderboard*; (xiii) o sistema de criação de contas, com *login* e *sign-up*, permitindo múltiplos usuários em um mesmo ambiente; e (xiv) os recursos de acessibilidade e configurações.

O Quadro 11 sintetiza os principais requisitos funcionais e seu estado de implementação ao término deste trabalho.

Quadro 11 – Síntese dos principais requisitos funcionais e seu estado de implementação

ID	Requisito	Descrição	Estado
RF01	Telas de navegação	Menu principal, seleção de modo e tela inicial	Implementado
RF02	<i>Gameplay/Interface</i> componentizada	Composição em cenas: barra superior, painel esquerdo, barra inferior e painel direito (prancheta)	Implementado
RF03	Sistema de violações	Armazenamento via <i>ViolationData</i> , <i>ViolationTarget</i> e vinculação à prancheta	Implementado
RF04	Sistema de cenários	<i>ScenarioCatalog</i> associando interfaces a violações e carregando dinamicamente a avaliação	Implementado
RF05	Sistema de pontuação e <i>feedback</i>	Cálculo via <i>AssessmentSystem</i> e exibição didática por violação na tela de resultados	Implementado
RF06	Sistema de dicas progressivas	<i>HintData</i> com níveis cumulativos, revelados sob demanda	Implementado
RF07	Catálogo de componentes ARIA	Biblioteca de componentes <i>AriaComponent</i> aderentes ao APG para autoria de interfaces	Implementado
RF08	Catálogo de interfaces	Dez interfaces associadas a H1–H10, armazenadas como cenas <i>.tscn</i> e referenciadas no catálogo	Implementado
RF09	Modo Normal com três dificuldades	Estrutura de progressão Fácil → Médio → Difícil	Parcial (Fácil)
RF10	Tutorial interativo sequencial	Cinco lições introdutórias sobre heurísticas e fluxo de avaliação	Parcial
RF11	Obstáculos e <i>power-ups</i> (Desafio/Score)	Elementos que aumentam ou reduzem a dificuldade	Projetado
RF12	Modos Desafio e Score	<i>Leaderboard</i> , pontuações, submodos (Infinito, Alvo, Tempo)	Projetado
RF13	Sistema de contas (<i>login/sign-up</i>)	Autenticação e múltiplos perfis em um mesmo ambiente	Projetado
RF14	Recursos de acessibilidade e Configurações	Navegação por teclado, alternância de tema, cursores personalizados	Implementado

Fonte: Elaborado pelo autor, 2026.

Complementarmente ao Quadro 11, o Apêndice E apresenta o diagrama de rastreabilidade de requisitos, conectando, na primeira camada, as funcionalidades e restrições (requisitos funcionais e não funcionais) aos artefatos de código e de

cena correspondentes (segunda camada) por meio de relações de satisfação (terceira camada). Esta estrutura segue o padrão adotado por trabalhos de conclusão brasileiros, como Benitti (2015), e viabiliza a construção de tabelas de rastreabilidade para a defesa da monografia. Os requisitos sem elementos correspondentes representam trabalhos futuros ou práticas de desenvolvimento (RNF-05, RNF-06 e RNF-07).

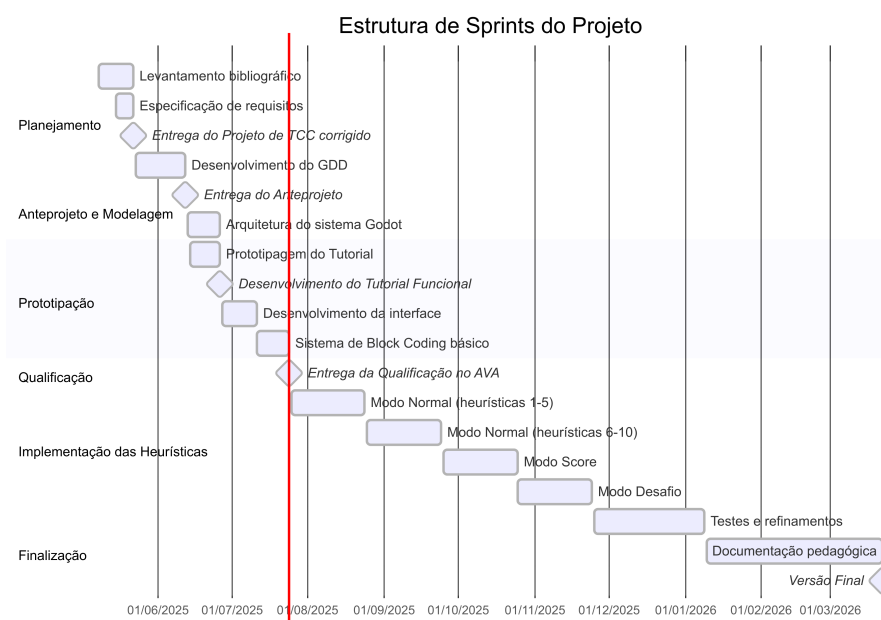
3.4.2 Metodologia Ágil Adaptada

O processo de desenvolvimento adotou princípios da metodologia ágil *Scrum*, adaptados para o contexto acadêmico individual. Foram implementados:

- **Sprints quinzenais:** Ciclos de desenvolvimento com objetivos específicos e entregáveis definidos;
- **Product backlog:** Lista priorizada de funcionalidades a serem implementadas;
- **Revisão e retrospectiva:** Análise crítica ao final de cada *sprint* para avaliação do progresso e ajustes metodológicos.

A Figura 7 apresenta a estrutura de *sprints* adotada durante o processo de desenvolvimento.

Figura 7 – Estrutura de *sprints* adotada no desenvolvimento



Fonte: Elaborado pelo autor, 2026.

O desenvolvimento ativo do UX Detective estendeu-se por aproximadamente doze meses, organizados em vinte e quatro *sprints* quinzenais. A condução do projeto seguiu cadência regular de planejamento, execução, revisão e retrospectiva por *sprint*, com objetivos incrementais alinhados ao *backlog* consolidado no Game

Design Document. O Quadro 12 apresenta a distribuição aproximada dos *sprints* entre quatro macrofases sequenciais, com sobreposição parcial entre fases adjacentes: levantamento bibliográfico e definição de escopo; projeto da arquitetura de sistemas atômicos e do modelo de dados em *Resources*; implementação dos sistemas centrais, do catálogo de cenários e da camada de componentes ARIA; e refinamento, testes automatizados com o *framework* GUT e redação da documentação técnica.

Quadro 12 – Distribuição dos *sprints* quinzenais entre macrofases do desenvolvimento

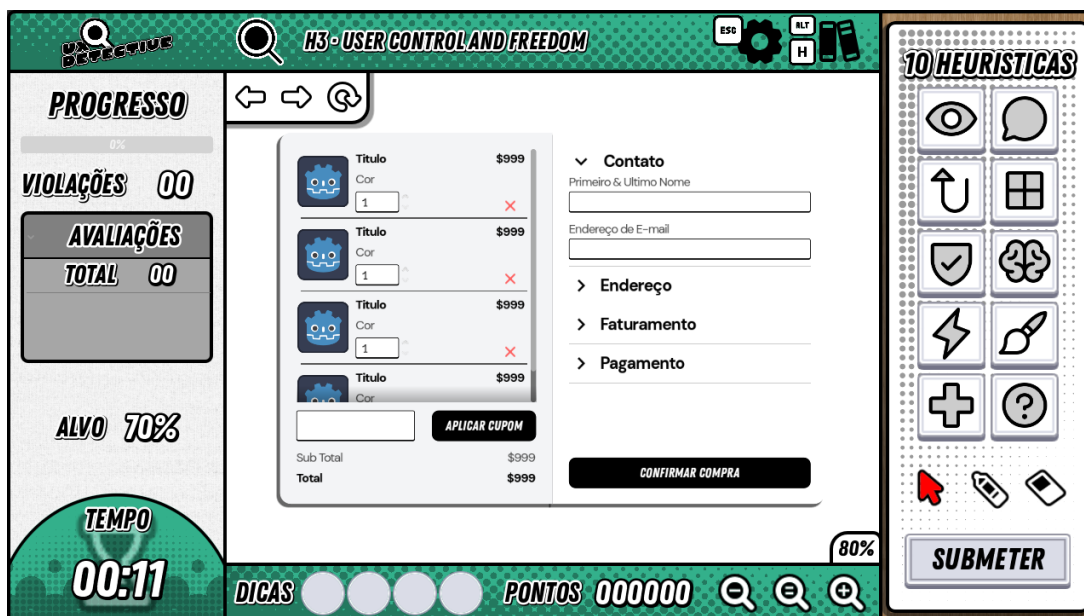
Fase	Macroatividade	<i>Sprints</i>	Entregáveis principais
I	Levantamento bibliográfico e escopo	3	Revisão da literatura, requisitos iniciais, GDD em versão preliminar
II	Arquitetura e modelo de dados	4	Arquitetura de sistemas atômicos, modelo de <i>Resources</i> , diagramas de sinal
III	Implementação central	12	<i>Autoloads</i> , <i>managers</i> , componentes ARIA e dez cenários H1–H10
IV	Refinamento, testes e documentação	5	Suíte de testes em GUT, ajustes de UX e redação da monografia
Total		24	—

Fonte: Elaborado pelo autor, 2026.

Uma prática deliberada adotada ao longo dos *sprints* foi a de construir primeiro o comportamento e a lógica das interfaces e, somente depois, refinar sua aparência. A Figura 8 apresenta uma interface de *checkout* em estado de protótipo funcional, na qual os campos, listas e botões já respondem corretamente aos sinais e às interações do usuário antes da incorporação dos elementos gráficos mais expressivos. Esta abordagem de “lógica antes de visual” minimiza o retrabalho e valida decisões de arquitetura em contexto funcional, seguindo as práticas de prototipação evolutiva descritas por Pressman e Maxim (2016).

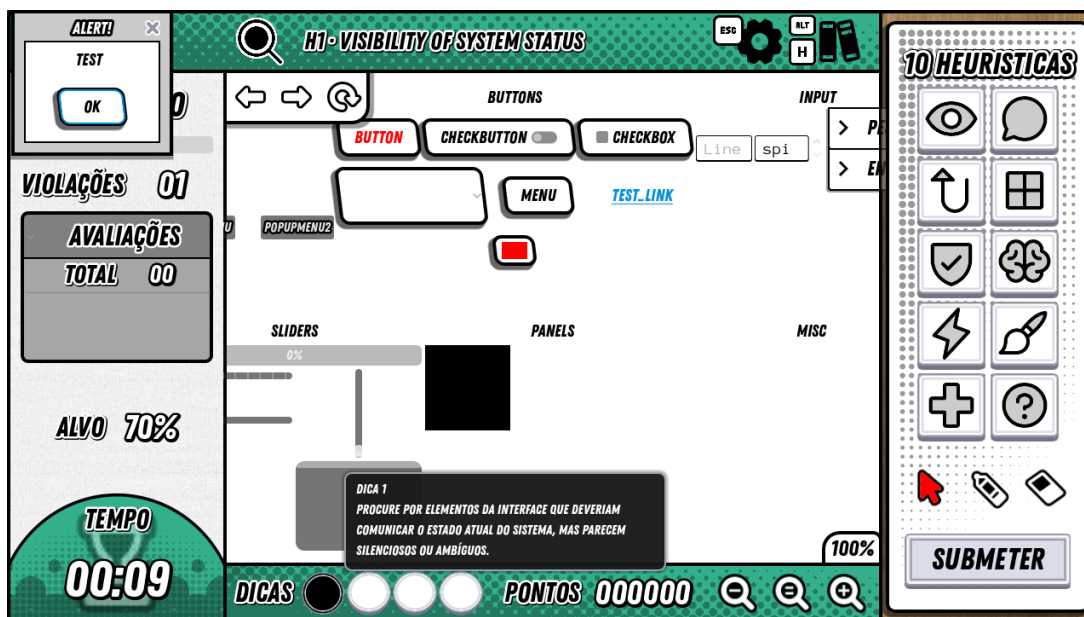
Adicionalmente, durante o desenvolvimento foi construída uma **interface de testes** voltada à validação em massa dos componentes *AriaComponent* e do sistema de violações. Esta interface, usada como ambiente controlado, permitiu exercitar rapidamente diversos tipos de *Control* (botões, *checkboxes*, *sliders*, painéis, *line edits*, *popup menus*) e injetar violações em lote, acelerando significativamente a implementação e os testes. A Figura 9 apresenta um recorte desta interface de testes com uma dica de nível 1 em exibição, ilustrando simultaneamente o ambiente de testes e o funcionamento do sistema progressivo de dicas descrito nas seções seguintes.

Figura 8 – Interface de *checkout* em estado de protótipo funcional, com lógica implementada antes do refinamento visual



Fonte: Elaborado pelo autor, 2026.

Figura 9 – Dica de nível 1 em exibição sobre a interface de testes utilizada no desenvolvimento



Fonte: Elaborado pelo autor, 2026.

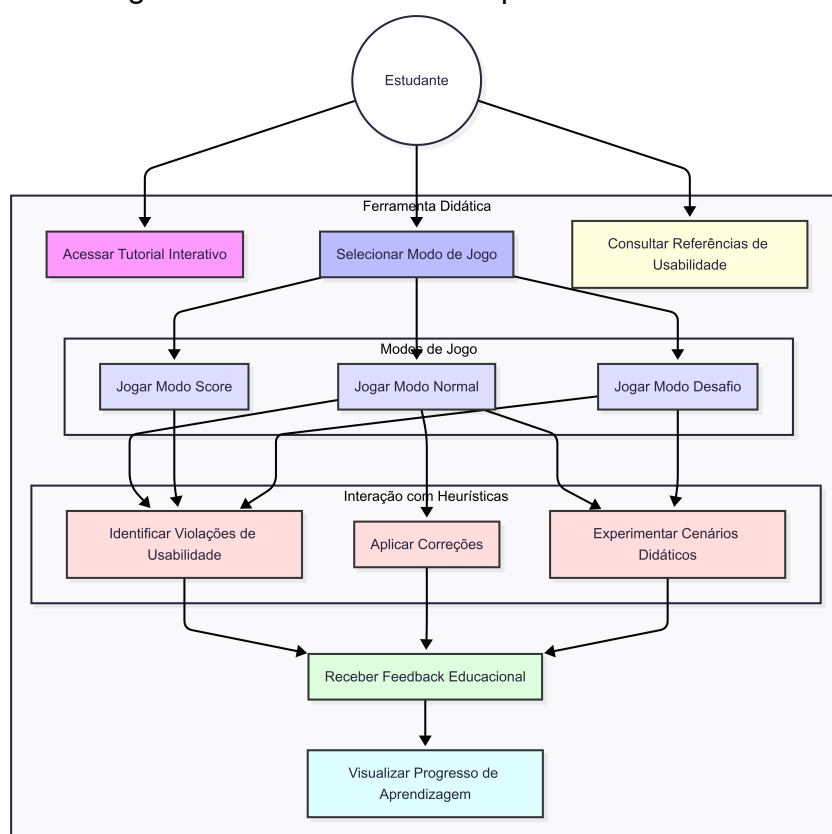
3.4.3 Diagramação e Modelagem

Para documentar e planejar os aspectos estruturais e comportamentais da ferramenta, foram utilizados:

- Diagramas de caso de uso:** Modelagem das interações entre usuário e sistema, destacando os fluxos principais de interação para cada modo de jogo.
- Diagramas de atividade:** Representação dos fluxos de execução das principais funcionalidades, especialmente para os cenários didáticos relacionados às heurísticas de Nielsen.
- Mapas conceituais:** Estabelecimento de relações entre os conceitos de usabilidade e as mecânicas de jogo que os representam.

A Figura 10 apresenta o diagrama de caso de uso simplificado, modelando as interações entre o estudante e os principais módulos da ferramenta.

Figura 10 – Diagrama de caso de uso simplificado da ferramenta didática



Fonte: Elaborado pelo autor, 2026.

3.4.4 Design Educacional e Framework Pedagógico

A estrutura educacional da ferramenta fundamenta-se em princípios de aprendizagem ativa e avaliação formativa, conforme discutido no Capítulo 2. O *design* pedagógico articula as dez heurísticas de Nielsen com objetivos de aprendizagem estruturados segundo a Taxonomia de Bloom, promovendo progressão cognitiva do conhecimento factual à aplicação crítica.

Para operacionalizar esta abordagem, foram desenvolvidos quatro sistemas educacionais integrados, cada um com responsabilidade específica no processo de ensino-aprendizagem:

- a) **ScenarioManager (Gerenciador de Cenários Pedagógicos):** Módulo responsável pelo carregamento, apresentação e gerenciamento do ciclo de vida dos cenários de avaliação heurística. Implementa máquina de estados finitos (GAMMA *et al.*, 1994) para controlar transições entre fases de observação, identificação, validação e *feedback*, rastreando violações identificadas pelo estudante e mantendo contexto do cenário ativo;
- b) **AssessmentSystem (Sistema de Avaliação Formativa):** Componente de avaliação baseada em critérios, com *feedback* diagnóstico. Gerencia banco de questões de múltipla escolha e verdadeiro-falso relacionadas às heurísticas, calcula pontuação segundo critérios de correção (identificações corretas, omissões e falsos positivos), e gera relatórios de desempenho individualizados. Implementa rubrica de avaliação alinhada aos níveis de dificuldade (Fácil, Médio, Difícil);
- c) **NarrationController (Controlador de Narrativa Pedagógica):** Módulo de mediação instrucional que guia o estudante através dos conteúdos educacionais. Responsável pela entrega de explicações didáticas sobre cada heurística, gerenciamento do sistema de dicas progressivas (com custo variável conforme dificuldade) e apresentação de soluções recomendadas após avaliação. Adapta linguagem e nível de detalhe conforme perfil do estudante;
- d) **ProgressTracker (Sistema de Acompanhamento de Progresso):** Componente de rastreamento de desempenho e evolução do aprendiz. Implementa sistema de desbloqueio por níveis de dificuldade (Easy → Medium → Hard), registra classificação por estrelas (1-3 estrelas conforme precisão e uso de dicas), e persiste histórico de tentativas e melhores resultados. Emite sinais de conquista quando estudante completa todas as 10 heurísticas em determinado nível de dificuldade.

A comunicação entre estes sistemas ocorre exclusivamente através de si-

nais do EventBus, garantindo baixo acoplamento e facilitando testes isolados. Esta arquitetura modular permite, por exemplo, que diferentes estratégias de avaliação sejam experimentadas sem impactar o gerenciamento de cenários ou o rastreamento de progresso.

Do ponto de vista pedagógico, cada um desses sistemas tem papel específico no processo de aprendizagem. O sistema de dicas progressivas em quatro níveis, gerenciado pelo `NarrationController`, opera como suporte gradual (*scaffolding*), conceito fundamentado na Zona de Desenvolvimento Proximal de Vygotsky (VYGOTSKY, 1978): o estudante ou profissional recebe apoio proporcional à sua necessidade, com custo crescente por dica para incentivar a autonomia. A classificação por estrelas — máxima apenas quando o cenário é concluído sem uso de dicas — configura um mecanismo de *mastery learning* que motiva a revisão deliberada até atingir desempenho pleno. Já a tela de resultados do `AssessmentSystem`, estruturada em três painéis (análise qualitativa, pontuação quantitativa e *feedback* diagnóstico por violação), implementa avaliação formativa nos moldes propostos por Benitti (2015): não apenas informa a pontuação final, mas explicita os erros cometidos e as violações não identificadas, permitindo que o estudante ou profissional compreenda o raciocínio correto antes de avançar ao próximo cenário.

Do ponto de vista pedagógico, a estrutura de modos de jogo (Tutorial, Normal, Desafio, Score) representa diferentes estratégias instrucionais:

- **Tutorial:** Aprendizagem guiada com andaimes (*scaffolding*), introduzindo conceitos progressivamente com suporte máximo;
- **Normal:** Prática deliberada com *feedback* formativo, permitindo consolidação do conhecimento através de dificuldade progressiva;
- **Desafio:** Aplicação sob pressão, testando automatização de competências adquiridas com limitações de tempo e obstáculos;
- **Score:** Gamificação competitiva, motivando aperfeiçoamento contínuo através de classificações e recordes pessoais.

Adicionalmente, todos os componentes de interface implementam requisitos de acessibilidade conforme WCAG 2.1 (W3C, 2018) e eMAG (BRASIL, 2014), utilizando padrões ARIA do WAI-ARIA APG (W3C, 2024) para garantir compatibilidade com tecnologias assistivas. Esta decisão alinha-se ao meta-princípio do projeto: a ferramenta de ensino de usabilidade deve, ela própria, exemplificar excelência em usabilidade e acessibilidade.

3.4.5 Arquitetura de Componentes ARIA

A implementação de acessibilidade na ferramenta didática fundamenta-se na especificação WAI-ARIA (*Web Accessibility Initiative – Accessible Rich Internet Ap-*

plications) do W3C, adaptada para o contexto de aplicações desenvolvidas na Godot Engine. Esta seção descreve a arquitetura de componentes de interface acessíveis e o sistema de violações intencional projetado para fins educacionais.

3.4.5.1 WAI-ARIA: Conceitos Fundamentais

O WAI-ARIA constitui uma especificação técnica desenvolvida pelo W3C para aprimorar a acessibilidade de conteúdo dinâmico e interfaces de usuário complexas (W3C, 2023). O padrão estabelece três conceitos fundamentais que orientam a implementação de componentes acessíveis:

- a) **Papéis (*Roles*)**: Definem o tipo semântico de um elemento de interface, comunicando sua função para tecnologias assistivas. Papéis como `button`, `checkbox`, `dialog` e `progressbar` permitem que leitores de tela anunciem adequadamente o propósito de cada componente, independentemente de sua representação visual;
- b) **Estados (*States*)**: Representam condições dinâmicas que podem mudar durante a interação do usuário. Estados como `aria-checked`, `aria-disabled`, `aria-expanded` e `aria-selected` informam tecnologias assistivas sobre a condição atual de elementos interativos;
- c) **Propriedades (*Properties*)**: Fornecem informações semânticas adicionais sobre elementos, geralmente estáticas ou que mudam com menor frequência. Propriedades como `aria-label`, `aria-describedby` e `aria-required` complementam o papel do elemento com contexto adicional para usuários de tecnologias assistivas.

A relação entre WAI-ARIA e as WCAG 2.1 (W3C, 2018) é complementar: enquanto as WCAG estabelecem critérios de sucesso para conteúdo acessível, o WAI-ARIA fornece mecanismos técnicos para implementar acessibilidade em interfaces dinâmicas. No contexto de software educacional, esta fundamentação técnica é particularmente relevante, pois permite demonstrar concretamente como problemas de acessibilidade se manifestam em componentes de interface, conectando teoria (princípios WCAG/POUR) à prática (implementação WAI-ARIA).

3.4.5.2 ARIA Authoring Practices Guide (APG)

O *ARIA Authoring Practices Guide (APG)* (W3C, 2024) consiste em documentação oficial do W3C que fornece padrões de implementação detalhados para widgets de interface comum. O guia especifica, para cada tipo de componente:

- **Estrutura semântica**: Papéis ARIA obrigatórios e opcionais, hierarquia de elementos, e relações entre componentes;

- **Navegação por teclado:** Padrões de interação esperados, incluindo teclas de atalho, navegação com Tab e setas, e comportamentos em contextos compostos;
- **Gerenciamento de foco:** Estratégias para movimentação de foco entre elementos, *focus trapping* em modais, e indicadores visuais de foco;
- **Estados e propriedades:** Atributos ARIA necessários, valores válidos, e condições para atualização dinâmica.

No desenvolvimento da ferramenta didática, a adoção do APG como referência canônica para implementação de componentes garante conformidade com expectativas estabelecidas de usuários de tecnologias assistivas. O Quadro 13 sintetiza os padrões APG implementados na ferramenta, relacionando-os aos componentes GDScript correspondentes.

Quadro 13 – Padrões ARIA APG implementados na ferramenta didática

Padrão APG	Componente	Papel ARIA	Navegação Principal
Button	AriaButton	button	Enter/Space ativa ação
Form Field	AriaFormField	textbox	Entrada de texto padrão
Checkbox	AriaCheckbox	checkbox	Space alterna estado
Progress Bar	AriaProgress	progressbar	Apenas leitura
Combobox	AriaComboBox	combobox	Setas navegam opções
Switch	AriaSwitch	switch	Space alterna on/off
Tabs	AriaTabs	tablist	Setas alternam abas
Accordion	AriaAccordion	region	Enter expande/colapsa
Breadcrumb	AriaBreadcrumb	navigation	Tab navega links
Dialog (Modal)	AriaDialog	dialog	Escape fecha, Tab circula
Alert	AriaAlert	alert	Anunciado automaticamente

Fonte: Elaborado pelo autor com base em (W3C, 2024), 2026.

3.4.5.3 Adaptação para Godot Engine

A transposição dos padrões WAI-ARIA, originalmente especificados para tecnologias web, para o contexto da Godot Engine demandou adaptações arquiteturais significativas. A engine não possui suporte nativo a ARIA, sendo necessário implementar uma camada de abstração que emule os comportamentos esperados.

A estratégia adotada baseou-se nos seguintes princípios:

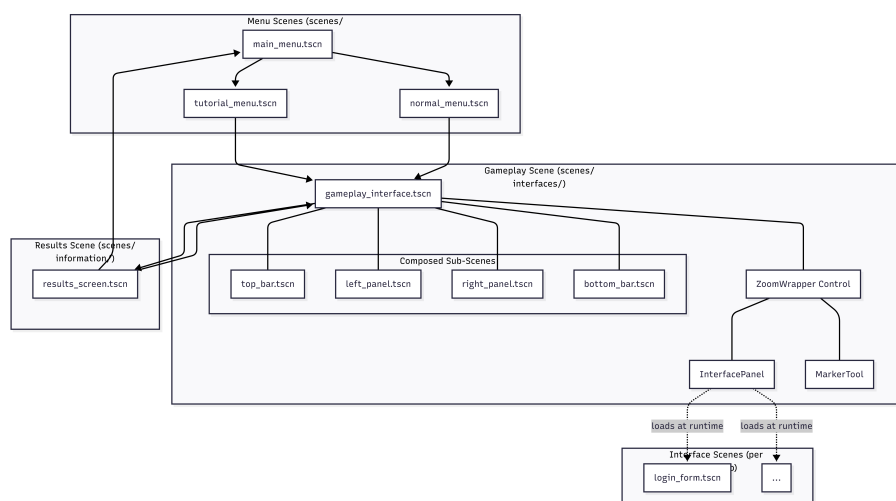
- a) **Extensão da classe Control:** Todos os componentes ARIA estendem a classe base `Control` da Godot, herdando funcionalidades de posicionamento, renderização e gerenciamento de foco;
- b) **Sistema de sinais para estados:** O mecanismo nativo de sinais da Godot implementa o padrão `Observer` (GAMMA *et al.*, 1994), permitindo comunicação assíncrona de mudanças de estado sem acoplamento di-

reto entre componentes;

- c) **Propriedades exportadas:** Atributos ARIA como `aria_name`, `aria_description` e `disabled` são expostos como propriedades editáveis no inspetor da Godot, facilitando configuração visual;
- d) **Métodos virtuais para extensão:** *Hooks* como `_handle_key_event()` e `_on_disabled_changed()` permitem que subclasses implementem comportamentos específicos sem sobrescrever lógica base.

Esta arquitetura possibilita que a ferramenta didática demonstre conceitos de acessibilidade de forma equivalente a aplicações *web*, apesar das diferenças de plataforma. A Figura 11 ilustra como as cenas voltadas ao usuário são compostas dentro da cena de *gameplay*: cada sub-cena é autocontida e pode ser aberta isoladamente no editor, expressando a mesma relação de composição-por-herança observada na hierarquia dos componentes ARIA.

Figura 11 – Composição de cenas da interface de *gameplay* a partir de sub-cenas autocontidas



Fonte: Elaborado pelo autor, 2026.

3.4.5.4 Arquitetura com Violações Intencionais

O diferencial pedagógico está no sistema de violações intencionais (*violation-aware architecture*), que permite a injeção controlada de problemas de acessibilidade em componentes de interface. Esta capacidade é fundamental para os cenários de avaliação heurística, onde estudantes devem identificar violações em interfaces simuladas.

A classe base `AriaComponent` implementa as seguintes estruturas para suporte a violações:

- `violated_flag`: Propriedade booleana computada que indica se o componente possui violações ativas. O valor é derivado automaticamente do

estado do dicionário de violações;

- `violations_dictionary`: Dicionário que mapeia identificadores de violação para seus dados descritivos, incluindo severidade, heurística relacionada e alterações de propriedade;
- `apply_violation(violation_data: Dictionary)`: Método que aplica uma violação ao componente, armazenando valores originais para posterior restauração e modificando propriedades conforme especificado;
- `remove_violation(violation_id: String)`: Método que remove uma violação específica, restaurando propriedades aos valores originais;
- `clear_all_violations()`: Método que remove todas as violações ativas, retornando o componente ao estado inicial.

O Quadro 14 apresenta a estrutura de dados utilizada para representar violações, incluindo campos obrigatórios e opcionais.

Quadro 14 – Estrutura de dados para representação de violações em componentes ARIA

Campo	Tipo	Descrição
<code>violation_id</code>	String	Identificador único da violação
<code>property_changes</code>	Dictionary	Alterações de propriedade a aplicar
<code>severity</code>	String	Severidade: “low”, “medium” ou “high”
<code>heuristic_id</code>	int	Heurística de Nielsen relacionada (1-10)
<code>description</code>	String	Descrição educacional da violação
<code>wcag_criterion</code>	String	Critério WCAG relacionado (opcional)

Fonte: Elaborado pelo autor, 2026.

O trecho de código a seguir exemplifica a aplicação de uma violação que oculta o *feedback* visual de um botão, violando a Heurística 1 (Visibilidade do Status do Sistema):

O Quadro 15 ilustra o mecanismo de injeção de violação em um componente *AriaButton*, demonstrando como o *ViolationTarget* modifica propriedades em tempo de execução.

O *ViolationTarget* aplica as alterações de propriedade ao componente pai via o método `set()` nativo da Godot Engine, armazenando os valores originais para restauração posterior. Ao detectar que o componente implementa o método `on_violation_applied()`, o *ViolationTarget* o invoca, permitindo que o componente reaja à violação além da simples troca de propriedade.

Este mecanismo permite que o *ScenarioManager* construa cenários de avaliação de forma declarativa, especificando quais componentes devem apresentar quais violações, sem necessidade de criar variantes separadas de cada interface.

Quadro 15 – Exemplo de configuração de violação em recurso *ViolationData*

Campo	Valor
violation_id	"h1_hidden_feedback_btn"
property_changes	{"modulate:a": 0.3, "disabled": true}
severity	"high"
heuristic_id	1
description	"Botão não fornece <i>feedback</i> visual de estado"
wcag_criterion	"2.4.7" (Focus Visible)

Fonte: Elaborado pelo autor, 2026.

3.4.5.5 Fluxo de Carregamento de Cenários

A integração entre os catálogos, os *Resources* de cenário, as interfaces ARIA e os sistemas de avaliação é orquestrada por meio de um fluxo bem definido de carregamento de cenários, executado a cada início de avaliação. Este fluxo é a manifestação concreta da arquitetura atômica, pois cada passo é executado por um componente especializado, comunicando-se exclusivamente por sinais e por passagem de *Resources*.

O fluxo típico é composto pelas seguintes etapas:

- a) **Seleção do cenário:** a partir do menu, o usuário seleciona uma heurística e uma dificuldade; o *GameSession* consulta o *ScenarioCatalog* e recupera o *ScenarioData* correspondente;
- b) **Transição de cena:** o *SceneManager* carrega a cena *gameplay_interface.tscn* e comunica a transição via sinal do *EventBus*;
- c) **Instanciação da interface-alvo:** o *GameplayInterfaceController* lê o *ScenarioData.interface*, consulta o *InterfaceCatalog* e instancia a cena de interface (*.tscn*) correspondente dentro do painel central de avaliação;
- d) **Injeção de violações:** o *ScenarioBuilder* percorre a lista de *ViolationData* do cenário e, para cada entrada, localiza o *ViolationTarget* declarado na interface alvo e aplica as alterações de propriedade via o método *set()* nativo da Godot, armazenando os valores originais para posterior restauração;
- e) **Povoamento da interface de avaliação:** o *GameplayInterfaceController*, com auxílio de *helpers*, popula a barra superior (título do cenário, pausa, guia), o painel esquerdo (progresso, violações marcadas, tempo decorrido), a barra inferior (dicas, pontuação, *zoom*) e o painel direito/prancheta (ferramentas,

heurísticas e botão de submissão);

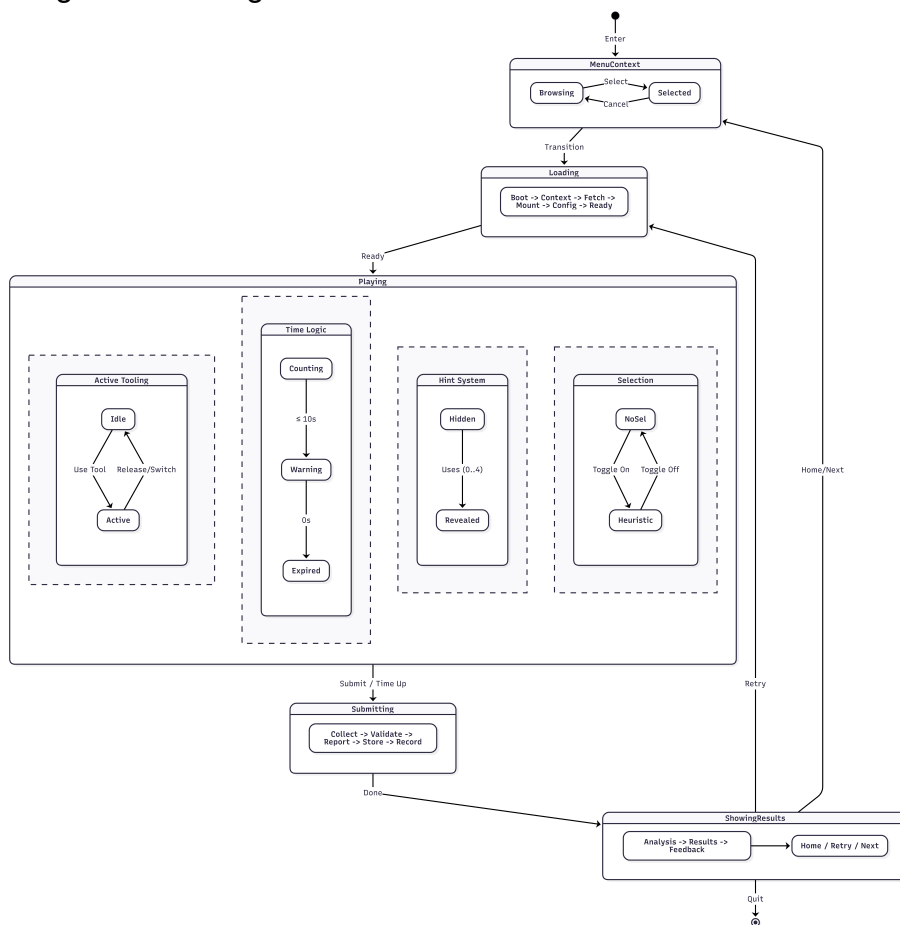
- f) **Interação do usuário:** o `MarkerTool` captura *clicks* do usuário; o `ClipboardController` registra a heurística selecionada; marcações são armazenadas na sessão;
- g) **Submissão e validação:** ao submeter, o `ValidationManager` compara as marcações do usuário com os `ViolationTarget` reais, classificando as respostas em verdadeiros positivos, falsos positivos, falsos negativos e verdadeiros negativos;
- h) **Avaliação e feedback:** o `AssessmentSystem` calcula pontuação, estrelas e nível de desempenho, e encaminha os dados à tela `results_screen.tscn`, que, por meio de *helpers*, popula os três painéis de resultado (Análise, Resultados e Feedback) a partir dos `ViolationData` associados ao cenário;
- i) **Registro de progresso:** o `ProgressTracker` consome o sinal de conclusão e persiste o melhor resultado do usuário para a heurística e dificuldade em questão, podendo desbloquear a próxima dificuldade.

O Apêndice F ilustra o *loop* de *gameplay* vivenciado pelo jogador durante um cenário, traduzindo as nove etapas descritas acima em um fluxograma. O nó central `PlayLoop` concentra cada ação do jogador — ler narração, selecionar heurística, trocar ferramenta, marcar, remover marcação, aplicar *zoom* ou solicitar dica — até que a submissão (*submit*) ou o esgotamento do tempo (*time up*) conduzam o fluxo à validação, avaliação e apresentação dos resultados. Pontos de persistência, marcados em cilindros verdes, ocorrem exatamente duas vezes por cenário: no registro de progresso e no armazenamento do relatório final; todo o restante é processado em memória. Os modos com bordas tracejadas indicam funcionalidades projetadas mas ainda não totalmente implementadas (os modos Desafio e Score).

Complementarmente, a Figura 12 apresenta o ciclo de vida do cenário na forma de um diagrama de estados: o fluxo de alto nível é linear (`MenuContext` → `Loading` → `Playing` → `Submitting` → `ShowingResults`), e durante o estado `Playing` quatro subestados concorrentes rastreiam, respectivamente, a ferramenta ativa, a lógica do temporizador, o sistema de dicas e a heurística selecionada. A árvore de nós da `Godot` suporta esta modelagem de forma natural, pois cada subestado é tratado por um *listener* distinto, conectado a sinais específicos do `EventBus`.

Esta orquestração declarativa — na qual cenários, violações e dicas são expressos como dados em *Resources* e a lógica permanece genérica — conecta os conceitos teóricos de acessibilidade e usabilidade às suas manifestações práticas em interfaces de usuário e permite que docentes expandam o catálogo sem modificar o código.

Figura 12 – Diagrama de estados do ciclo de vida do cenário



Fonte: Elaborado pelo autor, 2026.

3.5 Desenvolvimento

3.5.1 Seleção e Diversidade Temática das Interfaces

Na ausência de plataforma ou repositório consolidado de interfaces de referência para o ensino prático das heurísticas de Nielsen, a seleção das dez interfaces seguiu critério pragmático: modelar contextos de uso cotidianos ao público-alvo — estudantes e profissionais da área de Computação —, priorizando a familiaridade com o domínio retratado em cada cenário. Em paralelo, buscou-se diversidade temática entre os dez cenários, abrangendo entretenimento, autenticação, comunicação, educação, turismo, busca, finanças, configuração de sistema e comércio eletrônico, para evitar viés de domínio único e exercitar a transferência do conhecimento heurístico a contextos distintos.

O Quadro 16 sintetiza a correspondência entre heurística, interface implementada, tema e exemplo de serviço análogo de referência. A natureza autoral desta seleção constitui limitação reconhecida deste trabalho, considerada no protocolo de validação empírica detalhado no Apêndice A, em que validadores externos avaliarão a representatividade do catálogo.

3.5.2 Implementação das Heurísticas de Nielsen

Cada uma das dez heurísticas de Nielsen foi operacionalizada na ferramenta através de cenários didáticos específicos, permitindo aos estudantes identificar tanto exemplos positivos quanto violações de cada princípio. O Quadro 17 detalha as estratégias de implementação para cada heurística.

Quadro 16 – Catálogo de interfaces por heurística, tema e serviço análogo de referência

H	Heurística	Interface	Tema	Serviço análogo
H1	Visibilidade do status	Reprodutor de música	Entretenimento	Plataforma de <i>streaming</i> de música
H2	Correspondência sistema–mundo	Formulário de <i>login</i>	Autenticação	Tela de autenticação de serviço <i>web</i>
H3	Controle e liberdade	Compositor de <i>e-mail</i>	Comunicação	Cliente de <i>e-mail web</i>
H4	Consistência e padrões	Submissão acadêmica	Educação	Sistema de submissão de trabalhos
H5	Prevenção de erros	Reserva de hospedagem	Turismo	Plataforma de reserva de hotéis
H6	Reconhecimento vs. memorização	Busca com filtros	Busca/Varejo	Sistema de busca com filtros facetados
H7	Flexibilidade e eficiência	Painel de negociação	Finanças	Plataforma de negociação financeira
H8	Estética e <i>design</i> minimalista	Painel analítico	<i>Analytics</i>	Painel de visualização de dados
H9	Recuperação de erros	Configurações do sistema	Sistema	Tela de configuração de aplicativo
H10	Ajuda e documentação	Finalização de compra	Comércio eletrônico	Fluxo de <i>checkout</i> de <i>e-commerce</i>

Fonte: Elaborado pelo autor, 2026.

Quadro 17 – Estratégias de implementação das heurísticas de Nielsen

Heurística	Cenários Positivos	Cenários Negativos (Violações)
1. Visibilidade do status do sistema	Barras de progresso, indicadores de carregamento, <i>feedback</i> visual	Ausência de <i>feedback</i> após ações do usuário, estados indefinidos
2. Correspondência sistema-mundo real	Metáforas visuais familiares, linguagem do usuário	Terminologia técnica, ícones não intuitivos
3. Controle e liberdade	Botões de desfazer e refazer, opções de cancelamento	Ações irreversíveis sem confirmação, ausência de saídas claras
4. Consistência e padrões	Interfaces padronizadas, comportamentos previsíveis	Controles que mudam de posição, funções inconsistentes
5. Prevenção de erros	Validação de entradas, mensagens de confirmação	Padrões de <i>design</i> propensos a erros, armadilhas de interface
6. Reconhecimento vs. memorização	Ícones significativos, ajuda contextual	Comandos ocultos, códigos a memorizar
7. Flexibilidade e eficiência	Atalhos para usuários avançados, personalização	Fluxos rígidos, processos desnecessariamente longos
8. Design estético e minimalista	Interfaces limpas, hierarquia visual clara	Poluição visual, informações irrelevantes
9. Recuperação de erros	Mensagens de erro construtivas	Mensagens genéricas ou técnicas, sem orientação
10. Ajuda e documentação	Tutoriais contextuais, dicas integradas	Documentação inacessível ou inexistente

Fonte: Elaborado pelo autor com base em (NIELSEN, 1994), 2026.

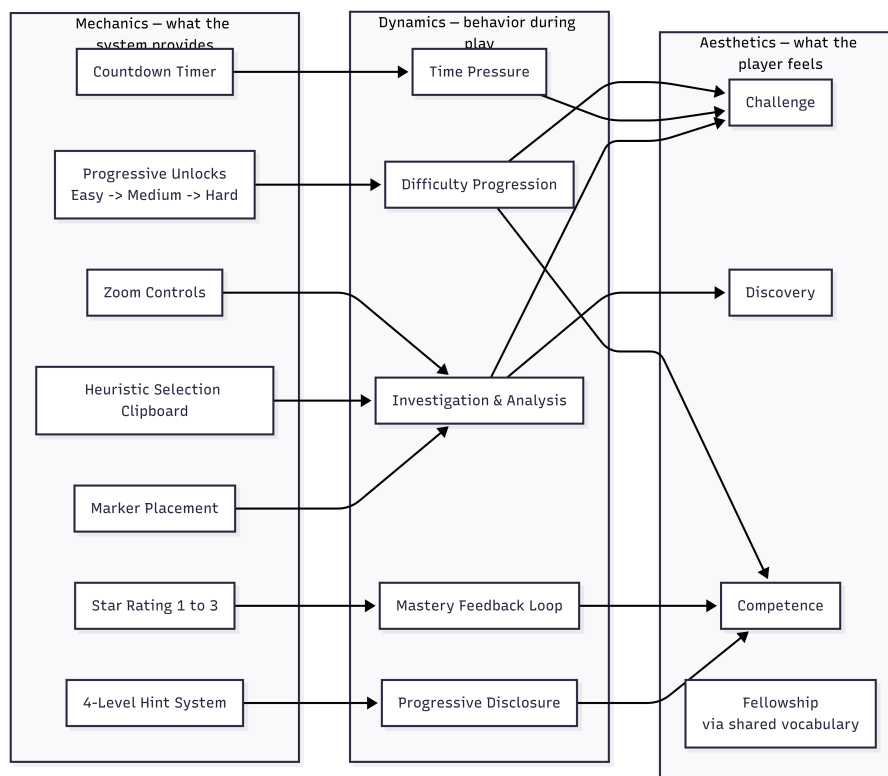
3.5.3 Estruturação dos Modos de Jogo

A ferramenta oferece quatro modos de jogo, cada um enfatizando aspectos distintos da aprendizagem de usabilidade. A concepção dos modos foi orientada pelo *framework* MDA (*Mechanics, Dynamics, Aesthetics*) (HUNICKE; LEBLANC; ZUBEK, 2004), que decompõe o *design* de jogos em três camadas:

- **Mecânicas** — elementos disponibilizados pelo sistema: cronômetro regressivo, desbloqueio progressivo, controles de *zoom*, prancheta de heurísticas, ferramentas de marcação, sistema de estrelas e dicas em quatro níveis.
- **Dinâmicas** — comportamentos emergentes do uso destas mecânicas: pressão temporal, progressão de dificuldade, investigação e análise da interface, ciclo de *feedback* sobre maestria e divulgação progressiva de informação.
- **Estéticas** — experiências afetivas e cognitivas pretendidas: desafio, descoberta, competência e pertencimento a uma comunidade que compartilhe vocabulário técnico.

A Figura 13 sintetiza esta articulação no UX Detective, evidenciando como cada mecânica gera uma dinâmica didática e contribui para a experiência estética pretendida.

Figura 13 – Mapeamento MDA (*Mechanics, Dynamics, Aesthetics*) do UX Detective



Fonte: Elaborado pelo autor, 2026, com base em (HUNICKE; LEBLANC; ZUBEK, 2004).

3.5.3.1 Modo Tutorial

O Modo Tutorial constitui a porta de entrada recomendada para novos usuários, apresentando os conceitos fundamentais de forma estruturada e sequencial. Embora opcional, o tutorial é fortemente incentivado após a criação de conta, permanecendo acessível via menu principal para consulta posterior.

Um diferencial do tutorial é a presença de um **assistente virtual**, implementado como *overlay* interativo sobre a *GameplayInterface*, cuja função é guiar o usuário por meio de destaques visuais (*highlights*) sobre os elementos relevantes e mensagens contextuais dependentes da etapa em curso. O assistente é responsável por direcionar a atenção do usuário, explicar o propósito de cada componente e fornecer *feedback* imediato sobre suas ações, caracterizando uma abordagem de aprendizagem guiada.

Estrutura pedagógica do tutorial (cinco lições):

- a) **Lição 1 — Boas-vindas e Objetivos:** introduz o conceito de heurísticas de usabilidade, sua relevância para IHC e o funcionamento da ferramenta, com o assistente destacando a estrutura da tela principal;
- b) **Lição 2 — Apresentação das Heurísticas:** apresenta visão geral de H1–H10, com definições, exemplos práticos e demonstrações interativas de violações comuns;
- c) **Lição 3 — Demonstração do Processo de Avaliação:** explica o funcionamento das ferramentas da prancheta (Cursor de Interação, Marcador de Violações, Removedor), o uso da lista de heurísticas e o processo de submissão, com o assistente marcando cada botão na ordem esperada de uso;
- d) **Lição 4 — Prática Guiada:** exercício supervisionado com assistência passo-a-passo e dicas obrigatórias para garantir compreensão;
- e) **Lição 5 — Primeira Avaliação:** interface simples com violações evidentes, *feedback* imediato e garantia de sucesso com o uso de dicas.

O sistema de desbloqueio sequencial (1 → 2 → 3 → 4 → 5) assegura progressão ordenada, com as lições 4 e 5 permanecendo bloqueadas até a conclusão das anteriores. Os indicadores visuais (✓) sinalizam as lições completadas.

3.5.3.2 Modo Normal

O Modo Normal constitui a experiência central de aprendizagem, permitindo exploração dos conceitos de usabilidade com suporte formativo adequado:

- **Estrutura:** 10 cenários (um por heurística H1-H10), cada um com 3 níveis de dificuldade (Fácil, Médio, Difícil);

- **Sistema de desbloqueio por nível:** Todas as 10 heurísticas iniciam desbloqueadas em dificuldade Fácil. Dificuldade Média desbloqueia apenas após completar todas as 10 heurísticas em Fácil. Dificuldade Difícil desbloqueia após completar todas em Médio;
- **Sistema de dicas progressivas:** Fácil (3 dicas), Médio (2 dicas), Difícil (1 dica), reveladas gradualmente;
- **Sistema de classificação por estrelas:** 1 estrela (✓, 60-79% correto), 2 estrelas (✓✓, 80-99%), 3 estrelas (★, 100% sem dicas);
- **Avaliação:** *Feedback* formativo detalhado sobre acertos, omissões e identificações incorretas, com explicações didáticas e recursos complementares.

Em atendimento à Heurística 3 (Controle e Liberdade do Usuário), o cenário em execução pode ser pausado a qualquer momento, abrindo um painel com as ações de retomar, reiniciar, acessar as configurações, consultar o guia e retornar ao menu principal, conforme ilustra a Figura 14. O recurso de pausa é compartilhado por todos os modos de jogo como elemento comum da *GameplayInterface*.

Figura 14 – Menu de pausa da interface de *gameplay* (liberdade e controle do usuário)



Fonte: Elaborado pelo autor, 2026.

3.5.3.3 Modo Desafio

O Modo Desafio testa o domínio das heurísticas sob pressão temporal e obstáculos dinâmicos. Está em estado de *design* documentado, ainda não implementado. O fluxo previsto começa com uma sequência contínua de rodadas, cada uma apresentando uma interface distinta.

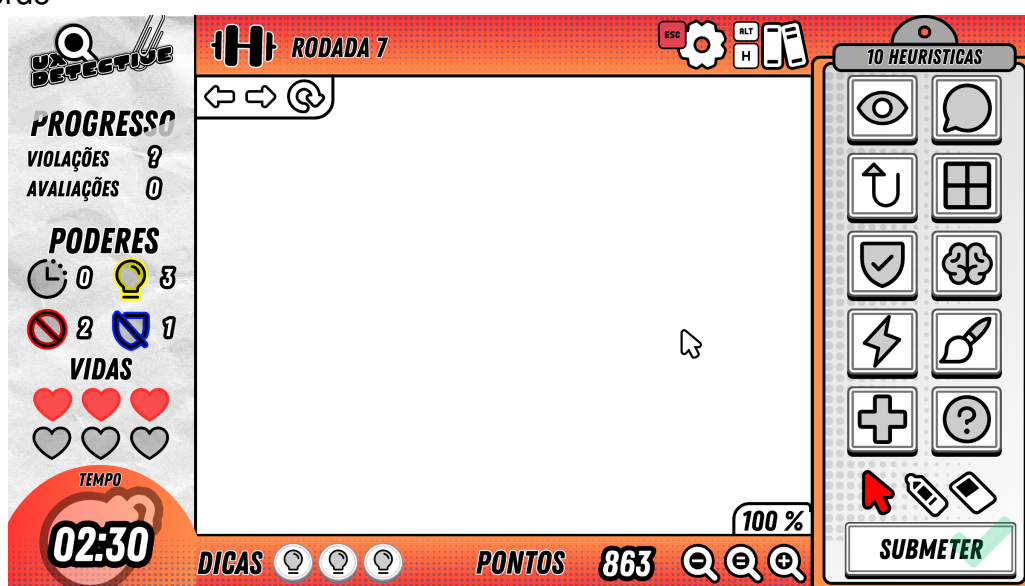
Um cronômetro regressivo limita o tempo de avaliação por rodada, iniciando em 5 minutos e convergindo para 2 minutos a cada cinco rodadas concluídas.

O sistema de vidas começa com três e admite até três adicionais recuperáveis em rodadas específicas; vidas são consumidas apenas ao esgotar o tempo, e avaliações incorretas geram penalidade temporal. Obstáculos dinâmicos simulam interrupções reais; *power-ups* podem ser obtidos para mitigar o desafio.

A condição de vitória é sobreviver a um número-alvo de rodadas sem perder todas as vidas. A pontuação combina pontos-base por identificação correta, bônus de tempo, multiplicadores de sequência e deduções por vidas perdidas ou dicas utilizadas.

A Figura 15 apresenta o *mockup* da interface do modo Desafio, com destaque para o painel esquerdo, no qual se observam os contadores de *power-ups* (congelamento de tempo, dicas, escudo e bloqueador de obstáculos) e o indicador de vidas do jogador. A Figura 16 evidencia a manifestação dos obstáculos dentro de um cenário em execução, incluindo *pop-ups*, desafios *CAPTCHA* e elementos explosivos interativos.

Figura 15 – *Mockup* do modo Desafio com *power-ups* e sistema de vidas no painel esquerdo



Fonte: Elaborado pelo autor, 2026.

3.5.3.4 Modo Score

O Modo Score, também em estado de *design* documentado, é projetado para incentivar competitividade e aperfeiçoamento por meio de pontuação maximizada. Seu fluxo previsto é: após seleção, o usuário escolhe um dos submodos — *Infinito* (jogo contínuo até falha), *Alvo* (atingir pontuação-alvo progressiva) ou *Tempo* (pontuação máxima em tempo limitado); rodadas sucessivas de avaliação heurística são apresentadas, pontuando ações como identificação correta de violação, acerto

Figura 16 – Mockup do modo Desafio com obstáculos dinâmicos (*pop-ups* e *CAPTCHA*) sobre o cenário



Fonte: Elaborado pelo autor, 2026.

de heurística, tempo restante e rodadas perfeitas (100% sem dicas); multiplicadores de sequência incentivam desempenho consistente; ao final da sessão, a pontuação é registrada em *leaderboards* (globais, pessoais e de desafios semanais). A duração é variável conforme o submodo selecionado. O sistema de pontuação e *leaderboard* depende do sistema de contas (RF13), também em estado projetado.

A Figura 17 apresenta a tela de seleção do modo Score, com os três submodos (Infinito, Alvo e Tempo) disponíveis no painel lateral e a tabela de classificação (*leaderboard*) apresentada simultaneamente em suas visões Global e Pessoal, permitindo ao jogador contextualizar seu desempenho em relação à comunidade e aos próprios recordes.

A Figura 18 sintetiza o fluxograma dos três modos de jogo principais da ferramenta (Normal, Desafio e Score), evidenciando as transições entre o menu principal e cada modo.

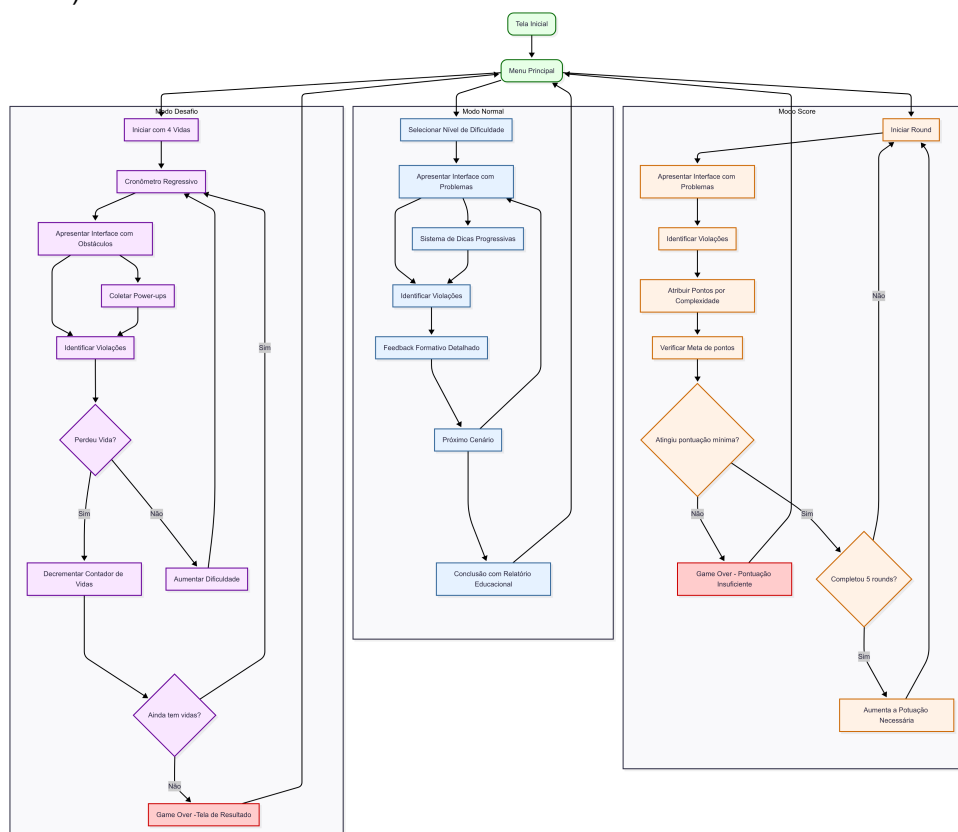
Figura 17 – Tela do modo Score: seleção de submodo e *leaderboard* Global e Pessoal

GLOBAL				PESSOAL		
POS	NOME	ROUND	PONTOS	POS	ROUND	PONTOS
1	PEDRO	42	14646	1	22	8746
2	JOAO	38	12458	2	23	8736
3	CARLA	37	11789	3	22	8363
4	KK	30	10063	4	19	8289
5	JOVEM	26	9753	5	18	7896
6	NAT	22	8746	6	18	7845

Buttons: INFINITO, ALVO, TEMPO, ESC VOLTAR

Fonte: Elaborado pelo autor, 2026.

Figura 18 – Fluxograma simplificado dos três modos de jogo principais (Normal, Desafio e Score)



Fonte: Elaborado pelo autor, 2026.

3.5.4 Implementação com GDScript

O desenvolvimento utilizando GDScript foi estruturado seguindo a arquitetura de nós e sinais da Godot Engine e a decomposição em sistemas atômicos descrita nas seções anteriores. Em particular, quatro componentes concentram a maior parte da lógica de execução do modo Normal: o `GameSession` (*autoload* de sessão), o `GameplayInterfaceController` (*controller* de tela), o `MarkerTool` (ferramenta de marcação) e o `ValidationManager` (*manager* de validação). Esta subseção descreve o papel arquitetural de cada um.

3.5.4.1 *GameSession*

O `GameSession`, registrado como *autoload*, mantém o estado da sessão ativa e coordena o início e o encerramento de avaliações. Suas responsabilidades incluem armazenar referências ao usuário corrente, ao modo de jogo ativo (Tutorial, Normal, Desafio, Score), ao cenário em execução (`ScenarioData`) e ao nível de dificuldade selecionado; emitir sinais como `scenario_started` e `scenario_finished` através do `EventBus`; e fornecer métodos utilitários para *controllers* e *managers* consultarem o contexto da sessão sem depender de referências cruzadas. O Quadro 18 sintetiza sua interface.

Quadro 18 – Responsabilidades centrais do `GameSession`

Elemento	Responsabilidade
<code>var active_scenario: ScenarioData</code>	Cenário em execução no momento
<code>var difficulty: Difficulty</code>	Dificuldade selecionada (Fácil, Médio, Difícil)
<code>func start_scenario(scenario)</code>	Emite <code>scenario_started</code> e prepara sistemas dependentes
<code>func finish_scenario(result)</code>	Encerra avaliação, dispara <code>scenario_finished</code> e encaminha resultado ao <code>AssessmentSystem</code>
<code>signal scenario_started(data)</code>	Permite que UI, <i>MarkerTool</i> e outros sistemas reajam ao início

Fonte: Elaborado pelo autor, 2026.

3.5.4.2 *GameplayInterfaceController*

O `GameplayInterfaceController` é o *controller* central da *GameplayInterface*, responsável por orquestrar a composição da tela de avaliação. Ao receber um `ScenarioData` via `GameSession`, ele: (i) consulta o `InterfaceCatalog` para ob-

ter a cena da interface a ser avaliada e a instancia no painel central; (ii) delega ao *ScenarioBuilder* a injeção das violações declaradas; (iii) conecta os sinais das cenas componentes (*top_bar*, *left_panel*, *bottom_bar*, *right_panel/clipboard*) ao *EventBus*; e (iv) atualiza painéis em tempo real com dados da sessão (tempo decorrido, violações marcadas, heurística selecionada, dicas restantes). A organização desta classe expressa o padrão *Mediator* (GAMMA *et al.*, 1994), mantendo os componentes visuais desacoplados entre si.

3.5.4.3 *MarkerTool*

O *MarkerTool* é o componente de ferramenta de marcação, que captura cliques do usuário sobre a interface-alvo e desenha as marcações visuais. Ao receber um clique, o *MarkerTool* identifica o componente sob o cursor, verifica a heurística atualmente selecionada no *ClipboardController* e cria um *badge* de marcação vinculado ao componente. Cada marcação contém o identificador do componente, a heurística associada e coordenadas. O *MarkerTool* atua em conjunto com o *ZoomWrapper* para preservar a consistência visual das marcações mesmo após *zoom* e *pan* da interface.

3.5.4.4 *ValidationManager*

O *ValidationManager* compara cada marcação do usuário com os *ViolationTarget* da interface, classificando-as conforme a matriz de confusão (TP, FP, FN, TN). A heurística associada à marcação participa da classificação: uma marcação sobre um componente realmente violado, mas com a heurística incorreta, é tratada como falso positivo. O resultado classificado é emitido via sinal e consumido pelo *AssessmentSystem*, que calcula pontuação, estrelas e nível de desempenho. O Quadro 19 resume a interface do *ValidationManager*.

Quadro 19 – Responsabilidades centrais do *ValidationManager*

Elemento	Responsabilidade
<code>func validate(markings, targets)</code>	Classifica marcações em TP, FP, FN e TN
<code>func compute_metrics()</code>	Calcula precisão, revocação e acurácia por heurística
<code>signal validation_completed(report)</code>	Emite o relatório final para o <i>AssessmentSystem</i>

Fonte: Elaborado pelo autor, 2026.

Adicionalmente, no contexto do sistema de cenários, as violações são expressas como *Resources ViolationData* e associadas a um *ViolationTarget* decla-

rado na interface alvo. O Quadro 20 exemplifica a configuração de uma violação da Heurística 4 (Consistência e Padrões).

Quadro 20 – Injeção de violação H4 (Consistência) via *ViolationData*

Campo	Valor
violation_id	"h4_button_position_swap"
property_changes	{"position": Vector2(220, 0)} aplicado ao botão "Sim"
heuristic_id	4
description	"Posição dos botões Confirmar/Cancelar invertida em relação ao padrão da plataforma"

Fonte: Elaborado pelo autor, 2026.

A Figura 19 apresenta o diagrama de sequência da submissão: desde o clique do jogador no botão até a exibição da tela de resultados. O diagrama é útil para compreender o acoplamento assíncrono entre *GameplayInterfaceController*, *ValidationManager*, *AssessmentSystem*, *ProgressTracker* e a tela de resultados.

A combinação destes quatro componentes, articulada por *signals* do *EventBus* e por *Resources* tipados, permite que o modo Normal opere de forma genérica para qualquer cenário do catálogo, sem necessidade de código específico por heurística. A utilização de GDScript — com tipagem, anotações `@export`, comentários `##` para documentação e referências por `%NomeUnico` — garante, simultaneamente, clareza do código e integração nativa com o editor da Godot.

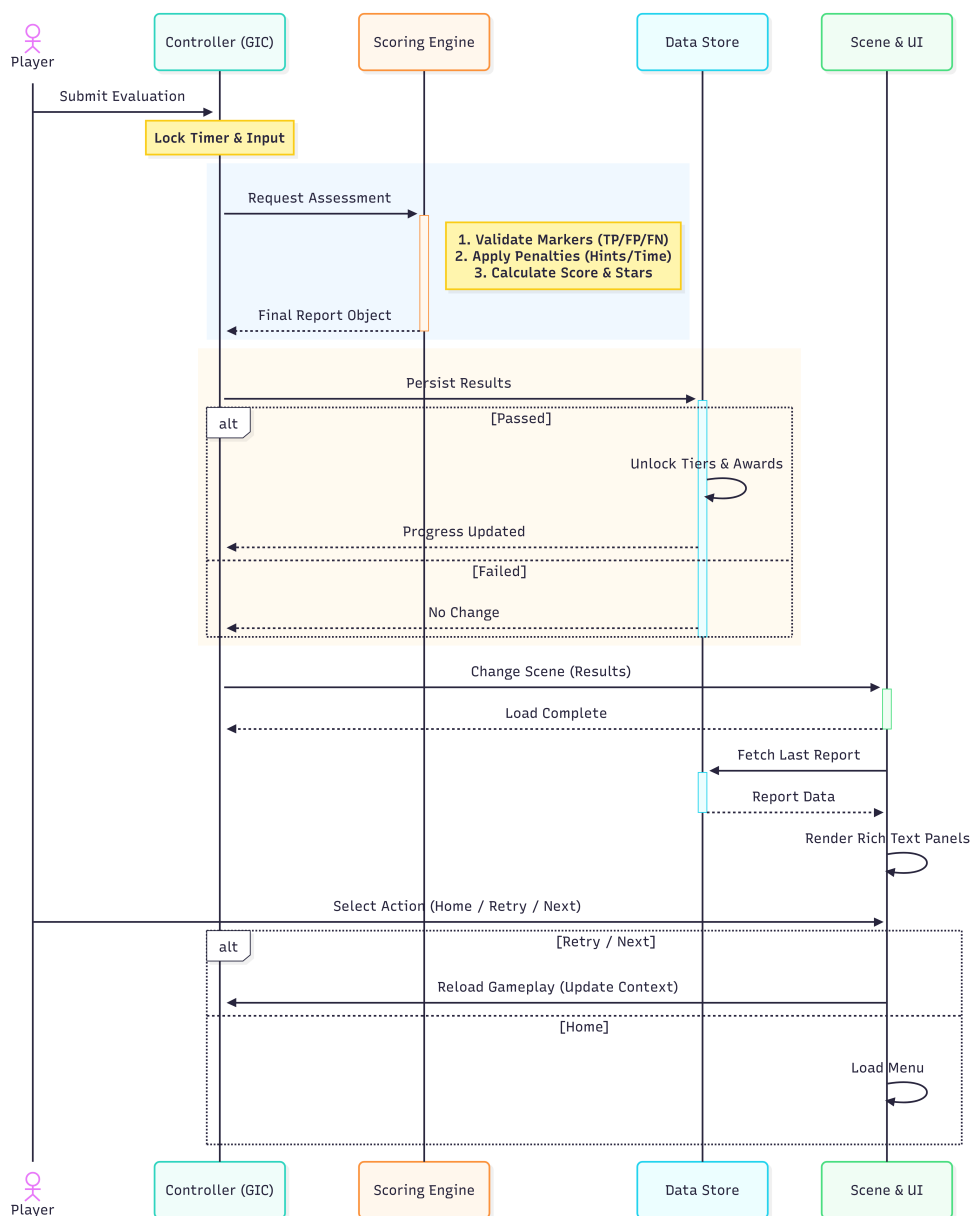
3.5.5 Processo de Avaliação Formativa

Para orientar o desenvolvimento da ferramenta, foi adotada uma abordagem de avaliação formativa baseada em:

- Revisão por pares:** Consulta a colegas da área de computação para coleta de impressões e sugestões sobre a estrutura e funcionamento da ferramenta;
- Aplicação de heurísticas:** Utilização das próprias heurísticas de Nielsen como instrumento de avaliação da ferramenta em desenvolvimento;
- Estudo de documentação:** Revisão sistemática de documentação *online* e boas práticas de desenvolvimento de jogos educacionais, incorporando princípios consolidados na literatura especializada.

Esta abordagem permitiu identificar oportunidades de melhoria de forma iterativa durante o processo de desenvolvimento, refinando a ferramenta antes mesmo da avaliação com usuários finais.

Figura 19 – Diagrama de sequência do fluxo de submissão e carregamento da tela de resultados



Fonte: Elaborado pelo autor, 2026.

4 RESULTADOS

Este capítulo apresenta os resultados do desenvolvimento da ferramenta didática UX Detective, documentando as implementações, as decisões de *design* e o estado funcional alcançado ao término do trabalho.

4.1 Interface e Design Visual

4.1.1 Tela Inicial e Menu Principal

A interface foi desenvolvida seguindo os próprios princípios de usabilidade que a ferramenta ensina. A Figura 20 apresenta a tela inicial, com elementos que orientam o usuário e reforçam a identidade visual.

Figura 20 – Tela inicial da ferramenta didática



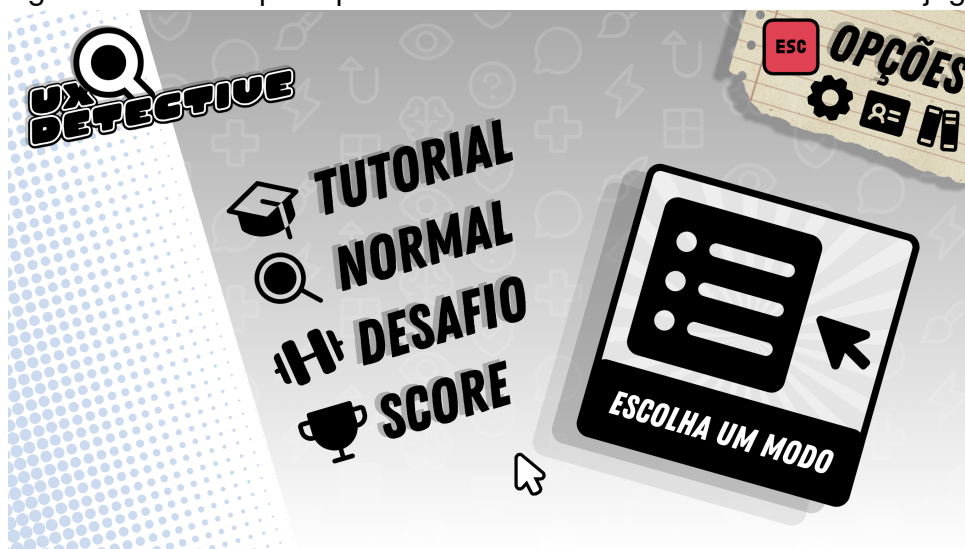
Fonte: Elaborado pelo autor, 2026.

O menu principal foi estruturado de forma hierárquica, priorizando clareza e facilidade de navegação, com acesso direto aos modos de jogo, tutorial, biblioteca de heurísticas e configurações. A Figura 21 ilustra a organização das opções disponíveis.

4.1.2 Elementos Visuais e Identidade Gráfica

A identidade visual da aplicação foi consolidada a partir dos *mockups* produzidos no Canva, que orientaram a definição da paleta cromática, tipografia e hierarquia visual. Os elementos gráficos funcionais (cursors personalizados, ícones de emoção e *feedback*) foram obtidos do Kenney Cursor Pack e do Kenney Emotes Pack, ambos sob licença Creative Commons Zero (CC0) (KENNEY, 2024). Os ícones das

Figura 21 – Menu principal com acesso aos diferentes modos de jogo



Fonte: Elaborado pelo autor, 2026.

dez heurísticas, bem como ícones de navegação e ação da ferramenta, foram extraídos da biblioteca Lucide (LUCIDE, 2024), que fornece mais de 1.600 *assets* SVG em *open-source*. O uso destes acervos garantiu consistência estética entre componentes, em linha com a heurística 4 (Consistência e Padrões), e eliminou barreiras de licenciamento para a disponibilização aberta do código-fonte.

4.2 Funcionalidades Implementadas

4.2.1 Modo Normal: Estado Funcional

O modo Normal é a principal entrega funcional deste trabalho. Todos os sistemas previstos para sua operação foram implementados, distribuídos em três camadas:

- **autoloads:** `GameSession`, `SceneManager`, `EventBus`, `ThemeManager`, `CursorManager` e `ProgressTracker`;
- **systems:** `ScenarioBuilder`, `NarrationController`, `ValidationManager`, `AssessmentSystem`, `MarkerTool` e `ViolationBadgeSystem`;
- **controllers:** `GameplayInterfaceController` e `ClipboardController`.

O `InterfaceCatalog` reúne dez interfaces ARIA, cada uma associada pelo `ScenarioCatalog` a `ViolationData` específicas e dicas progressivas, cobrindo as dez heurísticas de Nielsen (H1 a H10). A dificuldade Fácil tem todos os dez cenários jogáveis de ponta a ponta; as dificuldades Média e Difícil permanecem em estado projetado, dependendo apenas de novas configurações de `ScenarioData`, sem alteração no código.

A Figura 22 apresenta a tela de seleção do modo Normal, na qual o usuário

visualiza os dez cenários correspondentes às heurísticas H1–H10, o percentual de progresso e as estrelas conquistadas em cada dificuldade. Os botões das dificuldades Média e Difícil permanecem acinzentados por ainda não terem sido desbloqueados pelo usuário, comunicando visualmente a regra de progressão.

Figura 22 – Menu do modo Normal: seleção de heurística e dificuldade, com progresso e estrelas



Fonte: Elaborado pelo autor, 2026.

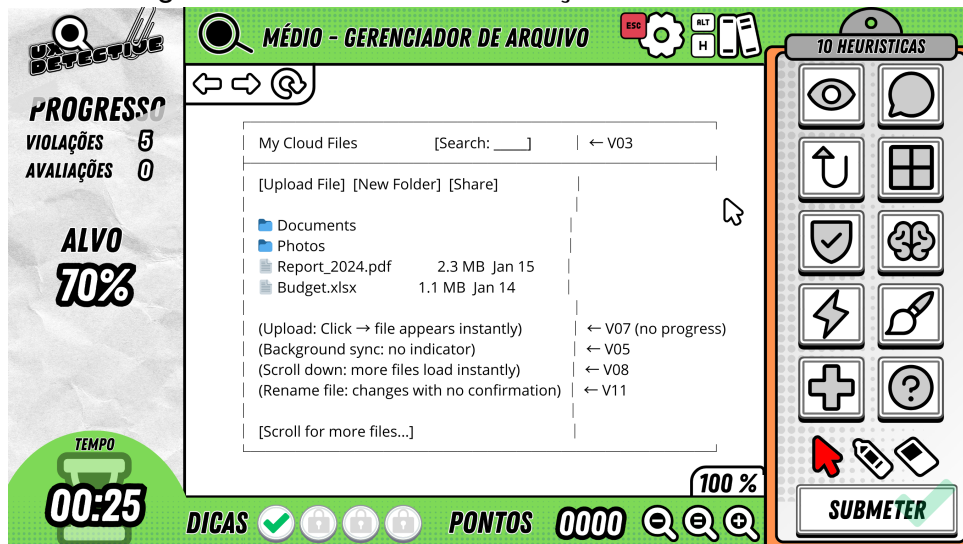
Do ponto de vista pedagógico, o desbloqueio progressivo das dificuldades implementa o princípio da aprendizagem adaptativa (TSAY; KOFINAS; LUO, 2018): o estudante avança da dificuldade Fácil para Média e Difícil conforme demonstra domínio no nível anterior, evitando sobrecarga cognitiva e consolidando os fundamentos heurísticos antes da exposição a desafios mais complexos. A exibição das estrelas conquistadas por cenário reforça a percepção de progresso e estimula a revisão deliberada até atingir desempenho pleno.

A Figura 23 apresenta um exemplo de cenário em execução no modo Normal, com a interface-alvo no painel central e a prancheta de avaliação no painel direito.

A Figura 24 detalha o esquema da *GameplayInterface* implementada, evidenciando a composição em barra superior, painel esquerdo, painel central, barra inferior e painel direito (prancheta).

A *GameplayInterface* configura um ambiente de aprendizagem por descoberta (PAPERT, 1980): o estudante ou profissional é colocado diante de uma interface real sem indicações sobre onde estão os problemas, precisando mobilizar o conhecimento das heurísticas para identificar as violações. A separação entre painel central (interface-alvo) e painel direito (prancheta de marcação) distingue observação de análise, exercitando os níveis superiores da Taxonomia de Bloom (BLOOM *et al.*, 1956) — análise e avaliação.

Figura 23 – Cenário em execução no modo Normal



Fonte: Elaborado pelo autor, 2026.

Figura 24 – Esquema da *Gameplay/Interface* com definição de cada componente

Fonte: Elaborado pelo autor, 2026.

4.2.2 Fluxo Principal da Aplicação

Além dos sistemas e do catálogo de cenários, foi implementado o fluxo principal da aplicação: do menu principal o usuário acessa o modo Normal, seleciona uma heurística e uma dificuldade, é conduzido à *GameplayInterface* onde realiza a avaliação, submete as marcações e é redirecionado à tela de resultados, que exibe *feedback* diagnóstico por violação. O Quadro 21 sintetiza as etapas deste fluxo e os componentes responsáveis por cada uma.

Quadro 21 – Fluxo principal do modo Normal e componentes responsáveis

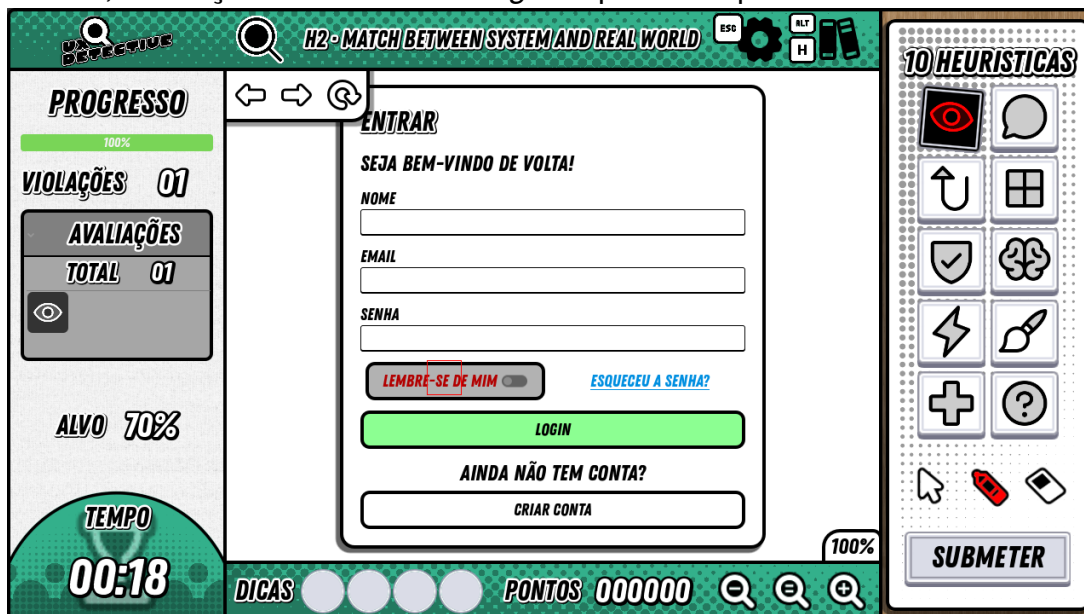
#	Etapa	Componente / Responsabilidade
1	Menu principal	MainMenu: navegação para modos, tutorial e configurações
2	Seleção de heurística e dificuldade	ScenarioSelector: consulta o ScenarioCatalog e entrega um ScenarioData
3	Transição de cena	SceneManager: carrega <code>gameplay_interface.tscn</code>
4	Povoamento da interface	GameplayInterfaceController: instancia interface-alvo e distribui dados entre painéis
5	Injeção de violações	ScenarioBuilder: aplica ViolationData aos ViolationTarget
6	Marcação	MarkerTool e ViolationBadgeSystem: registram e exibem as marcações do usuário
7	Submissão	ValidationManager: classifica em TP/FP/FN/TN
8	Avaliação e tela de resultados	AssessmentSystem + <code>results_screen.tscn</code> : pontuação, estrelas e <i>feedback</i> didático
9	Registro de progresso	ProgressTracker: persistência do melhor resultado

Fonte: Elaborado pelo autor, 2026.

Para ilustrar concretamente este fluxo, a Figura 25 apresenta um cenário de avaliação sobre a interface de *login*, na qual uma violação foi intencionalmente injetada no botão “Lembre-se de mim”. É possível observar a marcação realizada pelo usuário sobre o componente e a *badge* correspondente no painel esquerdo, indicando qual heurística foi associada àquela marcação. Após a submissão, o usuário é conduzido à tela de resultados (Figura 26), que organiza o *feedback* em três painéis: análise da pontuação obtida, resultados quantitativos do cenário e *feedback* didático sobre as violações identificadas.

O ciclo completo de avaliação — analisar a interface, marcar as violações, submeter e receber *feedback* diagnóstico — implementa o *loop* de avaliação formativa proposto por Benitti (2015): o estudante não recebe apenas uma pontuação, mas

Figura 25 – Interface de *login* em avaliação, com violação injetada no botão “Lembre-se de mim”, marcação do usuário e *badge* no painel esquerdo



Fonte: Elaborado pelo autor, 2026.

Figura 26 – Tela de resultados com análise de pontuação, resultados do cenário e *feedback* didático



Fonte: Elaborado pelo autor, 2026.

compreende quais violações identificou, quais omitiu e quais marcou indevidamente. Esse retorno detalhado promove reflexão sobre o próprio raciocínio heurístico, distinguindo a ferramenta de abordagens que limitam o *feedback* a um escore numérico final.

4.2.3 Tutorial Interativo: Estado Parcial

O modo Tutorial encontra-se em estado **parcialmente implementado**. Até o término do trabalho, foi entregue o **assistente virtual** responsável por conduzir o usuário com *highlights* visuais e mensagens contextuais sobre a *GameplayInterface*, bem como a **primeira lição** (“Boas-vindas e Objetivos”), que apresenta o conceito de heurísticas e o papel do usuário como avaliador. As lições 2 a 5 permanecem projetadas, com conteúdos didáticos definidos mas ainda não instanciados como cenários tutoriais completos. A Figura 27 apresenta o estado atual do fluxo do assistente.

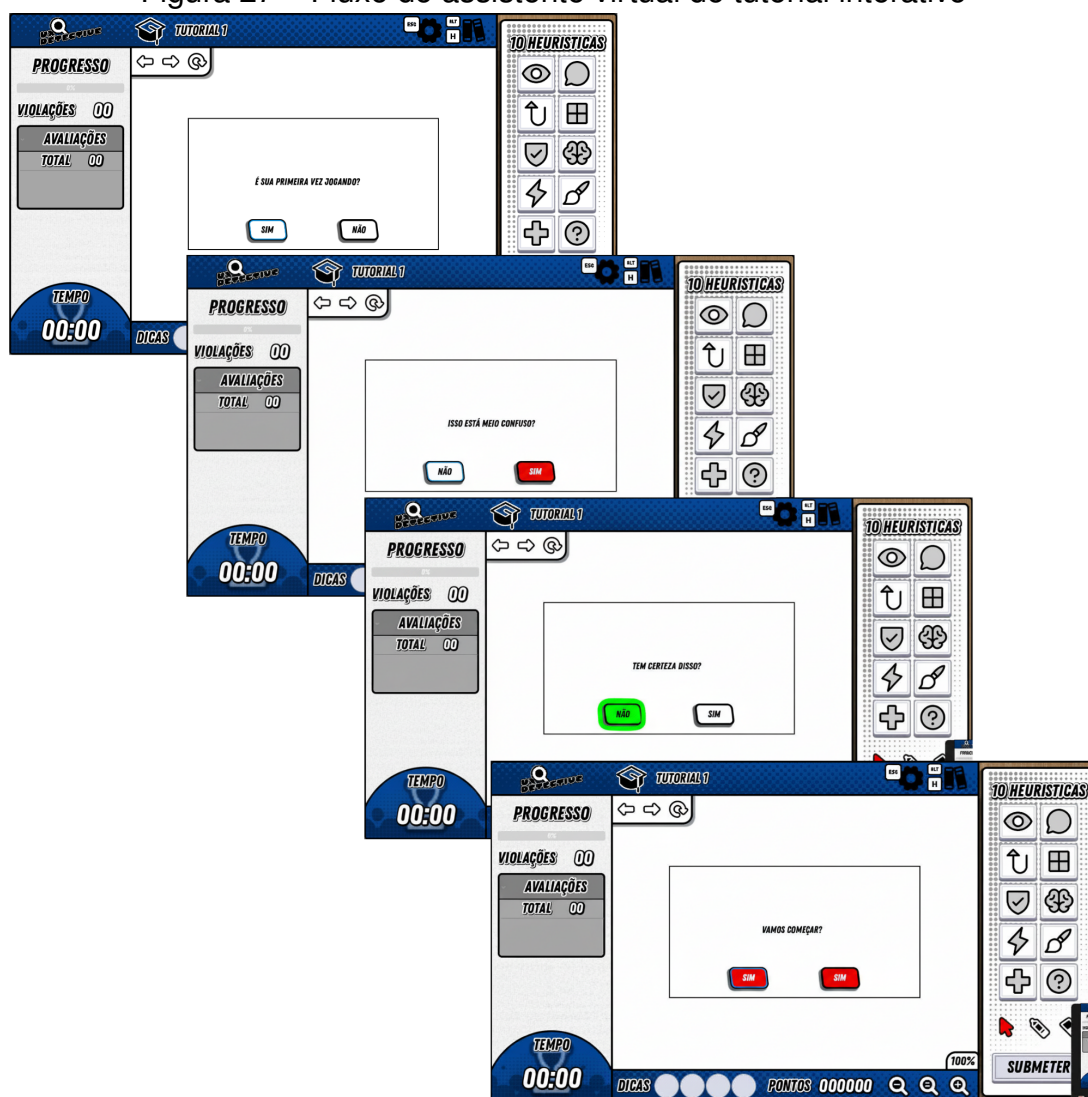
Do ponto de vista pedagógico, o Tutorial representa a camada de maior suporte instrucional na progressão de modos do UX Detective. A primeira lição entregue cumpre função de *onboarding* cognitivo: reduz a carga inicial ao apresentar o vocabulário das heurísticas e o papel do avaliador antes de qualquer cenário de avaliação propriamente dito, alinhando-se ao conceito de mediação instrucional discutido na literatura de aprendizagem baseada em jogos (JOMSRI; PRANGCHUMPOL; PO-ONSILP, 2025). As lições projetadas seguem a mesma lógica de *scaffolding* progressivo, preparando o estudante ou profissional para operar com autonomia crescente no modo Normal.

4.2.4 Modos Desafio e Score: Estado Projetado

Conforme indicado no Capítulo 3, os modos Desafio e Score permanecem em estado de *design* documentado, ainda não implementados. A arquitetura de sistemas atômicos e o sistema de cenários foram concebidos de forma a acomodá-los sem refatoração significativa: o *GameSession* já prevê a indicação de modo ativo, e tanto o sistema de obstáculos quanto os *power-ups* e o *leaderboard* podem ser expressos como extensões dos *Resources* já existentes.

A divisão entre o modo Normal e os modos Desafio e Score endereça o risco de memorização: em cenas determinísticas, o estudante poderia apenas decorar a localização das violações em vez de generalizar os princípios heurísticos. O modo Normal opera com cenários curados e progressão controlada, favorecendo a compreensão dos critérios de cada heurística por meio de *feedback* formativo imediato. Já os modos Desafio e Score foram projetados com sorteio de cenários, obstáculos dinâmicos e pressão temporal, exigindo aplicação do conhecimento a configurações

Figura 27 – Fluxo do assistente virtual do tutorial interativo



Fonte: Elaborado pelo autor, 2026.

inéditas. A separação reflete a progressão da Taxonomia de Bloom da compreensão para a aplicação e a análise (BLOOM *et al.*, 1956), promovendo transferência do conhecimento a contextos não vistos.

4.2.5 Cobertura das Heurísticas e Catálogo de Cenários

O Quadro 22 apresenta a cobertura atual do catálogo de cenários, indicando, para cada heurística, a existência de interface dedicada, a quantidade de violações modeladas e o estado de implementação nas três dificuldades previstas. A correspondência entre cada heurística e a interface temática associada está consolidada no Quadro 16 do Capítulo 3.

Quadro 22 – Cobertura do catálogo de cenários por heurística e dificuldade

H	Heurística	Interface	Fácil	Médio	Difícil
H1	Visibilidade do status do sistema	Sim	Implementado	Projetado	Projetado
H2	Correspondência sistema-mundo real	Sim	Implementado	Projetado	Projetado
H3	Controle e liberdade	Sim	Implementado	Projetado	Projetado
H4	Consistência e padrões	Sim	Implementado	Projetado	Projetado
H5	Prevenção de erros	Sim	Implementado	Projetado	Projetado
H6	Reconhecimento em vez de memorização	Sim	Implementado	Projetado	Projetado
H7	Flexibilidade e eficiência de uso	Sim	Implementado	Projetado	Projetado
H8	<i>Design</i> estético e minimalista	Sim	Implementado	Projetado	Projetado
H9	Recuperação de erros	Sim	Implementado	Projetado	Projetado
H10	Ajuda e documentação	Sim	Implementado	Projetado	Projetado

Fonte: Elaborado pelo autor, 2026.

4.2.6 Implementação com GDScript

A base de código em GDScript implementa os componentes descritos no Capítulo 3 (Seções sobre arquitetura de sistemas atômicos e sobre implementação). A camada de *autoloads* concentra seis *singletons* globais (EventBus, SceneManager, GameSession, ProgressTracker, ThemeManager, CursorManager); a camada de *systems* reúne ScenarioBuilder, NarrationController, ValidationManager,

AssessmentSystem, MarkerTool, ViolationBadgeSystem e ViolationTarget; a camada de *components* organiza os componentes ARIA, os *controllers* (GameplayInterfaceController, ClipboardController) e os elementos de UI reutilizáveis. O conteúdo didático é expresso como *Resources* .tres em src/resources/, abrangendo cenários, violações, dicas, catálogos e interfaces.

A Figura 28 ilustra um recorte representativo da base de código, evidenciando as convenções de uso de `class_name`, `extends`, `@export`, comentários `##` para documentação e referências via `%UniqueName` discutidas no Capítulo 3.

Figura 28 – Recorte representativo da implementação em GDScript da ferramenta

```
extends Control

## Main controller for gameplay_interface.tscn.
##
## Owns the full gameplay flow: loads scenario → mounts interface → applies
## violations → collects markers on submit → runs validation → records progress
## → navigates to results. Other systems (clipboard, marker tool, badges) are
## pure components that this controller wires together.

#region Node references
@onready var _right_panel: Control = %RightPanel
@onready var _left_panel: Control = %LeftPanel
@onready var _interface_host: Control = %Panel
@onready var _marker_tool: MarkerTool = %MarkerTool
@onready var _validation_manager: ValidationManager = %ValidationManager
@onready var _zoom_label: Label = %ZoomLabel
@onready var _zoom_wrapper: ZoomWrapper = %ZoomWrapper

#endregion

#region Runtime systems
var _assessment_system: AssessmentSystem = null
var _narration_controller: NarrationController = null
var _clipboard_controller: Node = null
var _badge_system: ViolationBadgeSystem = null
var _badges_container: GridContainer = null
var _scenario_helper: ScenarioHelper = null
```

Fonte: Elaborado pelo autor, 2026.

4.3 Síntese Consolidada dos Resultados

O Quadro 23 apresenta, de forma consolidada, o estado de entrega de cada macro-componente do UX Detective ao término deste trabalho, distinguindo entregas completas, parciais e projetadas.

Quadro 23 – Síntese consolidada do estado de entrega do UX Detective

Macro-componente	Estado	Observação
Arquitetura de sistemas atômicos (6 <i>autoloads</i>)	Implementado	EventBus, SceneManager, GameSession, ProgressTracker, ThemeManager, CursorManager
<i>GameplayInterface</i> componentizada	Implementado	Barras (superior, inferior) e painéis (esquerdo, central, direito/prancheta)
Catálogo de 10 interfaces ARIA	Implementado	Uma interface por heurística (H1–H10)
Sistema de violações (ViolationData/ViolationTarget)	Implementado	Injeção declarativa via <i>Resources</i>
Sistema de cenários e catálogos	Implementado	ScenarioCatalog e InterfaceCatalog baseados em <i>.tres</i>
Modo Normal — dificuldade Fácil	Implementado	Dez cenários jogáveis, cobrindo H1–H10
Modo Normal — dificuldades Média/Difícil	Projetado	Depende apenas da criação de novos ScenarioData
ValidationManager e AssessmentSystem	Implementado	Classificação TP/FP/FN/TN, pontuação e estrelas
Tela de resultados com <i>feedback</i> diagnóstico	Implementado	Três painéis (Análise, Resultados, Feedback) populados a partir de ViolationData
Sistema de dicas progressivas	Implementado	HintData acoplado ao NarrationController
Modo Tutorial	Parcial	Assistente virtual e primeira lição entregues
Modos Desafio e Score	Projetado	Fluxos documentados, mecânicas especificadas
Sistema de contas e <i>leaderboard</i>	Projetado	Necessário para o modo Score
Acessibilidade (WAI-ARIA/WCAG 2.1)	Implementado	Componentes AriaComponent aderentes ao APG

Fonte: Elaborado pelo autor, 2026.

5 CONCLUSÃO

Este trabalho apresentou o desenvolvimento do UX Detective, uma ferramenta didática digital para o ensino prático das dez heurísticas de usabilidade de Nielsen (1994) a estudantes de graduação em cursos de Computação e a profissionais atuantes nas áreas de usabilidade e acessibilidade. A ferramenta foi construída sobre a *Godot Engine 4.x* utilizando GDScript como linguagem principal de programação, sob uma arquitetura de sistemas atômicos comunicantes por sinais, conforme detalhado nos Capítulos 3 e 4.

Os objetivos específicos do Capítulo 1 foram atendidos:

- Realizou-se levantamento bibliográfico sistemático sobre metodologias de ensino de usabilidade, jogos sérios e gamificação na educação, com análise de trabalhos correlatos como UsabilityGame (BENITTI, 2015), UsabilityCity (FERREIRA *et al.*, 2014) e HCI Gamification App (JOMSRI; PRANG-CHUMPOL; POONSILP, 2025).
- Projetou-se e implementou-se uma arquitetura modular baseada em seis *autoloads*, catálogos de *Resources* .tres e camadas de *managers*, *controllers* e *helpers*, garantindo desacoplamento e manutenção evolutiva.
- Entregaram-se dez cenários didáticos correspondentes às heurísticas H1 a H10, jogáveis de ponta a ponta na dificuldade Fácil, com infraestrutura preparada para expansão sem alteração no código-fonte.
- Documentou-se o processo de desenvolvimento ao longo de doze meses e vinte e quatro *sprints* quinzenais, distribuídos entre as macrofases de levantamento, arquitetura, implementação e refinamento descritas no Capítulo 3.

A ferramenta tem aplicabilidade direta em disciplinas de Interação Humano-Computador, Engenharia de Software, Experiência do Usuário e Acessibilidade Digital. Complementa abordagens expositivas tradicionais ao colocar estudantes para identificar violações em interfaces realistas, exercitando análise e avaliação — níveis superiores da Taxonomia de Bloom (BLOOM *et al.*, 1956) — em vez de apenas memorizar definições. Sua interface segue os padrões WAI-ARIA (W3C, 2024) e WCAG 2.1 (W3C, 2018), exemplificando os princípios que ensina. A definição de cenários em arquivos de recurso externos permite que docentes adaptem a ferramenta a contextos específicos, adicionando cenários, violações ou idiomas sem modificar o código.

A combinação metodológica adotada — desenvolvimento tecnológico, revisão bibliográfica e práticas ágeis do *Scrum* (PRESSMAN; MAXIM, 2016) adaptadas ao contexto individual — mostrou-se eficaz para conduzir o projeto de um escopo inicial exploratório até um artefato funcional, absorvendo mudanças de entendimento sem comprometer a conclusão no prazo estabelecido pelo curso. A arquitetura de

sistemas atômicos com comunicação por sinais, fundamentada no padrão *Observer* de Gamma *et al.* (1994), provou-se adequada para ferramentas didáticas: a independência dos componentes reduz a complexidade cognitiva do código, facilita a adição de novos cenários por contribuidores futuros e permite testar cada subsistema isoladamente com o *framework* GUT. A estrutura de quatro modos (Tutorial, Normal, Desafio e Score) progride do reconhecimento à aplicação automatizada das heurísticas, evitando memorização específica dos cenários e favorecendo a transferência dos princípios de usabilidade a interfaces inéditas.

Pretende-se proceder com o registro do *software* UX Detective junto ao Núcleo de Inovação Tecnológica (NIT) do IFMG, órgão responsável por apoiar a comunidade acadêmica no processo de proteção da propriedade intelectual, incluindo o registro junto ao Instituto Nacional da Propriedade Industrial (INPI), conforme a Lei nº 9.609/1998. A iniciativa segue a política institucional do IFMG de proteger a produção intelectual discente e permitir adoção controlada do artefato por outras instituições, ampliando o impacto dos resultados para além do contexto institucional.

5.1 Trabalhos Futuros

A principal limitação deste trabalho é a ausência de validação empírica com estudantes em sala de aula. A ferramenta foi submetida a análise heurística pelo próprio autor e a revisão formativa junto ao orientador; um estudo controlado com grupo experimental e grupo de controle, nos moldes propostos por Benitti (2015), permitiria medir objetivamente o ganho de aprendizado.

Como linhas de trabalho futuro identificam-se: a implementação completa dos modos Desafio e Score com toda a mecânica descrita, incluindo *leaderboard*, sistema de vidas, obstáculos dinâmicos e *power-ups*; a expansão do conjunto de cenários para contemplar múltiplas configurações em cada nível de dificuldade (Médio e Difícil), aproveitando a arquitetura *data-driven* já consolidada; a execução do estudo controlado para validação empírica detalhado no Apêndice A; a avaliação da ferramenta junto a usuários de tecnologias assistivas, como leitores de tela, validando na prática os padrões WAI-ARIA implementados; o desenvolvimento de módulo de edição de cenários integrado à própria ferramenta, permitindo que docentes criem materiais personalizados sem necessidade de conhecimento em *Godot Engine*; e a tradução da interface e dos conteúdos didáticos para o inglês, ampliando o alcance internacional do projeto.

5.2 Continuidade e Estratégia de Propriedade Intelectual

A continuidade do desenvolvimento do UX Detective após a conclusão deste trabalho está prevista no âmbito institucional do IFMG, em colaboração com o Núcleo de Inovação Tecnológica (NIT). O registro de *software* junto ao INPI, ao recair sobre a expressão literal do código-fonte, é compatível com a publicação científica de artigos que descrevam a arquitetura, as decisões de *design*, o catálogo de cenários didáticos e os resultados de futuras validações empíricas, sem comprometer a proteção da propriedade intelectual. Pretende-se submeter os resultados deste trabalho e do estudo controlado proposto no Apêndice A a periódicos e eventos da área de Educação em Computação e de Interação Humano-Computador.

Quanto à manutenibilidade da ferramenta, o código-fonte segue convenções de estilo consolidadas na comunidade GDScript, incluindo nomenclatura *Pascal-Case* para classes e *snake_case* para métodos e propriedades, tipagem estática nos contratos de *managers* e *controllers* e separação explícita entre camadas de *autoloads*, sistemas e componentes. A documentação do projeto contempla *Game Design Document* próprio descrevendo a arquitetura, os fluxos de comunicação por sinais e o catálogo de *Resources*, favorecendo a evolução incremental por contribuidores externos. A expressão de cenários e violações como arquivos *.tres* permite a expansão do catálogo didático sem alterações no código-fonte.

Em conformidade com a orientação institucional do NIT, o repositório de código permanecerá privado durante o processo formal de registro de *software* junto ao IFMG e ao INPI. Após o início ou a conclusão do registro, o repositório será publicizado sob licença alinhada à estratégia de propriedade intelectual definida pela instituição, permitindo o uso educacional por outras escolas e cursos, bem como o recebimento de contribuições da comunidade interessada na adoção e na evolução do artefato.

REFERÊNCIAS

- BENITTI, F. B. V. Evaluation of a Game Used to Teach Usability to Undergraduate Students in Computer Science. Versão inglesa. **Journal of Usability Studies**, v. 11, n. 1, p. 21–39, 2015. Disponível em: <http://www.upassoc.org>.
- BLOOM, B. S. *et al.* **Taxonomy of Educational Objectives: The Classification of Educational Goals – Handbook I: Cognitive Domain**. New York: David McKay Company, 1956.
- BRASIL. **eMAG: Modelo de Acessibilidade em Governo Eletrônico**: Versão 3.1. Governo Federal do Brasil, Departamento de Governo Digital. 2014. Disponível em: <https://www.gov.br/governodigital/pt-br/acessibilidade-e-usuario/emag>. Acesso em: 15/01/2025.
- BRASIL. **Lei nº 13.146, de 6 de julho de 2015**: Institui a Lei Brasileira de Inclusão da Pessoa com Deficiência (Estatuto da Pessoa com Deficiência). Brasília, DF, 06/07/2015. Diário Oficial da União, Brasília, DF, 7 jul. 2015. Disponível em: https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2015/lei/l13146.htm. Acesso em: 15/01/2025.
- FERREIRA, B. M. *et al.* UsabiliCity: Um Jogo de Apoio ao Ensino de Propriedades de Usabilidade de Software Através de Analogias. *In: ANAIS do XXV Simpósio Brasileiro de Informática na Educação (SBIE 2014)*. Sociedade Brasileira de Computação, 2014. v. 1, p. 1273. DOI: 10.5753/cbie.sbie.2014.1273.
- GAMMA, E. *et al.* **Design Patterns: Elements of Reusable Object-Oriented Software**. Reading: Addison-Wesley, 1994. ISBN 0-201-63361-2.
- GIL, A. C. **Como elaborar projetos de pesquisa**. 6. ed. São Paulo: Atlas, 2019. ISBN 978-8597012613.
- GONÇALVES, L. S. **Desenvolvimento de um Jogo Educacional para o Ensino de Algoritmos com Godot Engine**. 2023. Trabalho de Conclusão de Curso – Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais – Campus Bambuí, Bambuí.
- HOLM, U.; HUKE, V. **Gamification in a Workshop Environment: A Design for Evaluating the Effect of Gamification on Learning**. 2020. Master's Thesis – Chalmers University of Technology, Gothenburg.
- HUNICKE, R.; LEBLANC, M.; ZUBEK, R. MDA: A Formal Approach to Game Design and Game Research. Versão inglesa. *In: PROCEEDINGS of the AAAI Workshop on Challenges in Game AI*. San Jose, CA: AAAI Press, 2004. p. 1–5.
- INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE MINAS GERAIS (IFMG). **Manual de normalização de trabalhos acadêmicos**. Belo Horizonte, 2020.
- INSTITUTO FEDERAL DE MINAS GERAIS. **Resolução nº 2, de 13 de janeiro de 2025**: Dispõe sobre diretrizes gerais para Trabalho de Conclusão de Curso (TCC) no

âmbito do IFMG. Belo Horizonte, 13/01/2025. Disponível em: https://www.bambui.ifmg.edu.br/portal/images/2025/Conselho_Academico/Resolucoes/Resolucao_2_de_13.01.2025.pdf. Acesso em: 04/04/2025.

ISO. **ISO 9241-11:2018**: Ergonomics of human-system interaction — Part 11: Usability: Definitions and concepts. Versão inglesa. Geneva, 2018.

JOMSRI, P.; PRANGCHUMPOL, D.; POONSILP, K. Enhancing Student Engagement in HCI Courses through Gamified Mobile Learning Interfaces in Higher Education. Versão inglesa. **International Journal of Information and Education Technology**, International Journal of Information e Education Technology, v. 15, n. 4, p. 812–819, 2025. ISSN 20103689. DOI: 10.18178/ijiet.2025.15.4.2287.

KENNEY. **Kenney Game Assets**: Cursor Pack e Emotes Pack. Versão inglesa. Licença Creative Commons Zero (CC0). 2024. Disponível em: <https://kenney.nl/assets>. Acesso em: 01/10/2025.

LAKATOS, E. M.; MARCONI, M. d. A. **Fundamentos de metodologia científica**. 9. ed. São Paulo: Atlas, 2021. ISBN 978-8597026580.

LUCIDE. **Lucide Icons**: Open-source icon library. Versão inglesa. Mais de 1.600 ícones SVG sob licença ISC. 2024. Disponível em: <https://lucide.dev/>. Acesso em: 01/10/2025.

MARTIN, R. C. **Agile Software Development**: Principles, Patterns, and Practices. Upper Saddle River: Prentice Hall, 2003. ISBN 0-13-597444-5.

MYERS, B. A. A Brief History of Human-Computer Interaction Technology. Versão inglesa. *In*: 2. ACM interactions. New York, NY, USA: ACM, 1998. v. 5, p. 44–54. DOI: 10.1145/274430.274436.

NIELSEN, J. **Usability Engineering**. San Francisco: Morgan Kaufmann, 1994. ISBN 9780080520292.

NIELSEN, J.; MOLICH, R. Heuristic Evaluation of User Interfaces. Versão inglesa. *In*: PROCEEDINGS of the SIGCHI Conference on Human Factors in Computing Systems (CHI '90). New York, NY, USA: ACM, 1990. p. 249–256. ISBN 0-201-50932-6. DOI: 10.1145/97243.97281.

NORMAN, D. A. **The Design of Everyday Things**: Revised and Expanded Edition. New York: Basic Books, 2013. ISBN 978-0465050659.

PAIVA, L. *et al.* To the Next Level! An Exploratory Study on the Influence of User Experience on the Acceptance of a Gamified Learning Platform. Versão inglesa. *In*: INTERNATIONAL Conference on Computer Supported Education, CSEDU – Proceedings. Science e Technology Publications, Lda, 2022. v. 1, p. 281–288. ISBN 9789897585623. DOI: 10.5220/0011067900003182.

PAPERT, S. **Mindstorms**: Children, Computers, and Powerful Ideas. New York: Basic Books, 1980. ISBN 9780465046744.

PAUCAR-CURASMA, R. *et al.* Strengthening Computational Thinking in Engineering and Health Students Through IoT Activities and Block-Based Programming. Versão inglesa. **Revista de Gestão Social e Ambiental**, v. 18, n. 11, p. e09714, 2024. ISSN 1981-982X. Disponível em:

<https://rgsa.openaccesspublications.org/rgsa/article/view/9714>.

PERIN, A. P. J.; SILVA, D. E. D. S.; VALENTIM, N. M. C. Investigating Block Programming Tools in High School to Support Education 4.0: A Systematic Mapping Study. Versão inglesa. **Informatics in Education**, Vilnius University Institute of Data Science e Digital Technologies, v. 22, n. 3, p. 463–498, 2023. ISSN 16485831. DOI: 10.15388/infedu.2023.21.

PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de Software**: Uma Abordagem Profissional. 8. ed. Porto Alegre: AMGH, 2016. ISBN 978-8580555349.

PRODANOV, C. C.; FREITAS, E. C. d. **Metodologia do trabalho científico**: Métodos e técnicas da pesquisa e do trabalho acadêmico. 2. ed. Novo Hamburgo: Feevale, 2013. ISBN 978-8577171583. Disponível em:

<http://www.feevale.br/Comum/midias/8807f05a-14d0-4d5b-b1ad-1538f3aef538/E-book%20Metodologia%20do%20Trabalho%20Cientifico.pdf>.

RAMÍREZ RUIZ, J. J.; VARGAS SANCHEZ, A. D.; BOUDE FIGUEREDO, O. R. Impact of Gamification on School Engagement: A Systematic Review. Versão inglesa. **Frontiers in Education**, Frontiers Media, v. 9, 2024. DOI: 10.3389/feduc.2024.1466926.

RANAWEERA, M.; MAHMOUD, Q. H. Deep Reinforcement Learning with Godot Game Engine. Versão inglesa. **Electronics**, MDPI, v. 13, n. 5, 2024. ISSN 20799292. DOI: 10.3390/electronics13050985.

SELVESAKIS, M.; KAZANIDIS, I. SQLatch: Enhancing SQL Learning through Interactive Block-Based Programming. Versão inglesa. **ResearchGate Preprint**, 2024. Disponível em: <https://www.researchgate.net/publication/386210981>. Acesso em: 01/10/2025.

STACK OVERFLOW. **Stack Overflow Developer Survey 2025**: Admired and Desired Technologies. Versão inglesa. 2025. Disponível em: <https://survey.stackoverflow.co/2025/>. Acesso em: 10/02/2026.

TSAY, C. H. H.; KOFINAS, A.; LUO, J. Enhancing Student Learning Experience with Technology-Mediated Gamification: An Empirical Study. Versão inglesa. **Computers & Education**, Elsevier, v. 121, p. 1–17, 2018. ISSN 03601315. DOI: 10.1016/j.compedu.2018.01.009.

VYGOTSKY, L. S. **Mind in Society**: The Development of Higher Psychological Processes. Cambridge: Harvard University Press, 1978. ISBN 9780674576292.

W3C. **Accessible Rich Internet Applications (WAI-ARIA) 1.2**. Versão inglesa. World Wide Web Consortium. 2023. Disponível em: <https://www.w3.org/TR/wai-aria-1.2/>. Acesso em: 15/01/2025.

W3C. **ARIA Authoring Practices Guide (APG)**. Versão inglesa. World Wide Web Consortium. 2024. Disponível em: <https://www.w3.org/WAI/ARIA/apg/>. Acesso em: 15/01/2025.

W3C. **Web Content Accessibility Guidelines (WCAG) 2.1**: W3C Recommendation. Versão inglesa. World Wide Web Consortium. 2018. Disponível em: <https://www.w3.org/TR/WCAG21/>. Acesso em: 15/01/2025.

WÖRISTER, F.; KNOBELSDORF, M. A Block-Based Programming Environment for Teaching Low-Level Computing. Versão inglesa. *In*: ACM International Conference Proceeding Series. Association for Computing Machinery, 11/2023. ISBN 9798400716539. DOI: 10.1145/3631802.3631825.

APÊNDICE A - PROTOCOLO DE ESTUDO CONTROLADO PARA VALIDAÇÃO EMPÍRICA

O protocolo a seguir adapta o referencial metodológico de Jomsri, Prangchumpol e Poonsilp (2025), que comparou três protótipos de interfaces *mobile* gamificadas em um curso de Interação Humano-Computador no ensino superior, ao público-alvo do UX Detective — estudantes e profissionais da área de Computação — e responde à limitação reconhecida no Capítulo 5.

A.1 Objetivos e Hipóteses

O estudo tem por objetivo geral mensurar a efetividade pedagógica do UX Detective no ensino das heurísticas de usabilidade de Nielsen, comparando o ganho de aprendizagem e o nível de satisfação obtidos por meio da ferramenta com aqueles proporcionados por instrução tradicional. As hipóteses formuladas são:

- **H1:** O ganho de aprendizagem dos grupos que utilizam o UX Detective é estatisticamente superior ao do grupo submetido apenas a instrução tradicional.
- **H2:** O grupo que utiliza os modos Normal, Desafio e Score em conjunto apresenta maior capacidade de transferência do conhecimento heurístico a cenários inéditos do que o grupo restrito ao modo Normal, atendendo à progressão da Taxonomia de Bloom (BLOOM *et al.*, 1956).
- **H3:** A satisfação dos usuários com o UX Detective, medida por questionário de experiência de usuário, supera valores de referência relatados na literatura para ferramentas didáticas similares (BENITTI, 2015; JOMSRI; PRANGCHUMPOL; POONSILP, 2025).

A.2 Desenho Experimental

Adota-se desenho quase-experimental com três grupos independentes, formados por alocação aleatória estratificada conforme o nível de conhecimento prévio em IHC, aferido em questionário de *baseline*:

- **Grupo A (Controle):** Instrução tradicional, baseada em aula expositiva sobre as heurísticas de Nielsen e exercícios de identificação de violações em capturas de tela estáticas.
- **Grupo B (Experimental Restrito):** Instrução tradicional complementada pelo uso supervisionado do modo Normal do UX Detective.
- **Grupo C (Experimental Completo):** Instrução tradicional complementada pelo uso supervisionado dos modos Normal, Desafio e Score do UX

Detective.

A.3 Participantes

A amostra é composta por estudantes e profissionais da área de Computação vinculados ao IFMG e a instituições parceiras, matriculados em disciplinas de Interação Humano-Computador, Engenharia de Software ou áreas correlatas, ou atuantes profissionalmente em desenvolvimento e *design* de interfaces. Visa-se um tamanho amostral mínimo de sessenta participantes (vinte por grupo), considerando a magnitude de Jomsri, Prangchumpol e Poonsilp (2025) (oitenta e sete participantes em três grupos) e a viabilidade local. São critérios de inclusão: matrícula em disciplina elegível ou atuação profissional na área, familiaridade básica com plataformas de aprendizagem e ausência de experiência prévia com ferramentas gamificadas voltadas ao ensino de heurísticas. A participação é voluntária mediante Termo de Consentimento Livre e Esclarecido, com submissão prévia do protocolo ao Comitê de Ética em Pesquisa.

A.4 Instrumentos

A coleta de dados utiliza quatro instrumentos complementares:

- **Pré-teste e pós-teste de conhecimento:** Itens equivalentes que aferem reconhecimento de violações em capturas de tela, classificação correta segundo as heurísticas H1 a H10 e justificativa textual da violação identificada. A equivalência entre versões é assegurada por validação prévia por especialistas.
- **Questionário de satisfação de experiência de usuário:** Réplica do instrumento de onze itens em escala *Likert* de cinco pontos empregado por Jomsri, Prangchumpol e Poonsilp (2025), contemplando clareza de uso, facilidade de navegação, consistência do *design*, responsividade do sistema, estética e apelo visual, experiência de gamificação, compatibilidade com dispositivos, disponibilidade de ajuda e informação, customização e controle, adequação para fins educacionais e clareza de apresentação do conteúdo.
- **Instrumentação interna de comportamento:** Registro automático, pelo *AssessmentSystem*, das classificações TP, FP, FN e TN por cenário, do tempo decorrido por cenário, do número de dicas consultadas e do número de tentativas, conforme já implementado na ferramenta.
- **Mapas de calor (*heatmaps*) de interação:** Coleta de coordenadas de cliques e tempo de permanência nas regiões da interface, instrumentada

no Godot, replicando o procedimento de Jomsri, Prangchumpol e Poonsilp (2025).

A.5 Procedimento Experimental

Cada sessão experimental é padronizada em trinta minutos por participante, em ambiente laboratorial controlado quanto à iluminação, ao nível de ruído e à configuração do equipamento, conforme procedimento adotado por Jomsri, Prangchumpol e Poonsilp (2025). A sessão é estruturada em quatro etapas: aplicação do pré-teste; intervenção pedagógica correspondente ao grupo de alocação; aplicação do pós-teste; e preenchimento do questionário de satisfação. Todos os participantes recebem instruções idênticas previamente à intervenção, registradas em roteiro escrito, com vistas a minimizar viés do facilitador.

A.6 Métricas e Análise Estatística

O ganho de aprendizagem é calculado como ganho normalizado, definido pela razão entre o ganho efetivo (pós-teste menos pré-teste) e o ganho máximo possível (pontuação máxima do pré-teste subtraída do escore obtido no pré-teste). Aplica-se análise de variância de um fator (ANOVA) entre os três grupos, complementada por comparações múltiplas pelo procedimento de Tukey HSD, ao nível de significância $\alpha = 0,05$. A comparação intra-grupo entre pré-teste e pós-teste utiliza teste t pareado. A satisfação é analisada pelos mesmos procedimentos comparativos, e as métricas comportamentais (tempo, dicas, tentativas, taxas de acerto) são reportadas descritivamente e correlacionadas com o ganho de aprendizagem por meio de coeficientes de correlação de Pearson.

A.7 Considerações Éticas

O protocolo prevê submissão prévia ao Comitê de Ética em Pesquisa da instituição proponente, anonimização integral dos dados coletados e armazenamento em meio seguro, com acesso restrito aos pesquisadores. O Termo de Consentimento Livre e Esclarecido explicita finalidade da pesquisa, voluntariedade da participação, possibilidade de desistência a qualquer momento sem prejuízo e destino dos dados após o encerramento do estudo.

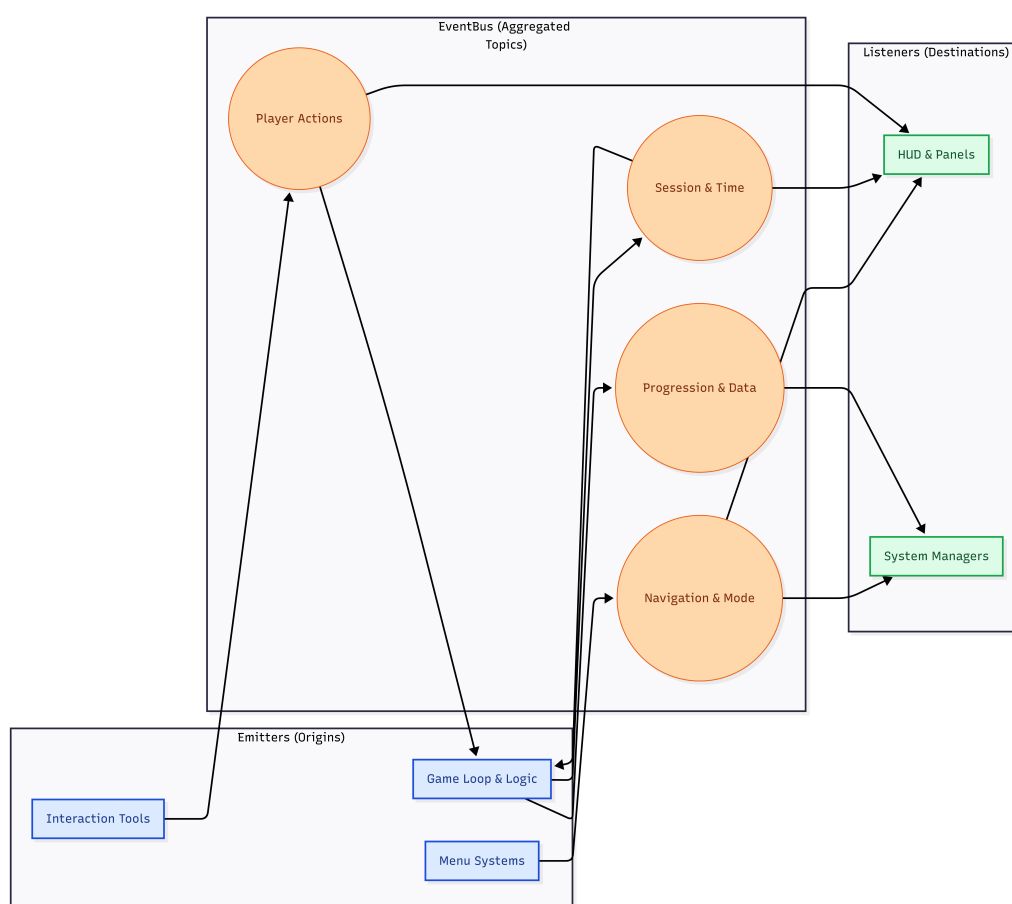
A.8 Cronograma Estimado

A execução completa do protocolo é estimada em seis meses, distribuídos em três fases: dois meses para preparação (validação dos instrumentos, submissão ao CEP, recrutamento e definição de cronograma com instituições parceiras); dois meses para coleta de dados (execução das sessões com os três grupos); e dois meses para análise estatística, interpretação dos resultados e redação dos artigos científicos derivados.

APÊNDICE B - GRAFO DE SINAIS DO EVENTBUS

O diagrama a seguir apresenta o grafo agregado de *publish/subscribe* do EventBus do UX Detective. Nós azuis representam os emissores de sinais, círculos laranjas representam os canais de sinal agrupados por tema, e nós verdes representam os ouvintes (*listeners*) conectados nos métodos `_ready()`. O diagrama permite rastrear cadeias de efeitos a cada emissão, servindo como antídoto ao *observer spaghetti*.

Figura 29 – Grafo agregado de *publish/subscribe* do EventBus: emissores, sinais e ouvintes

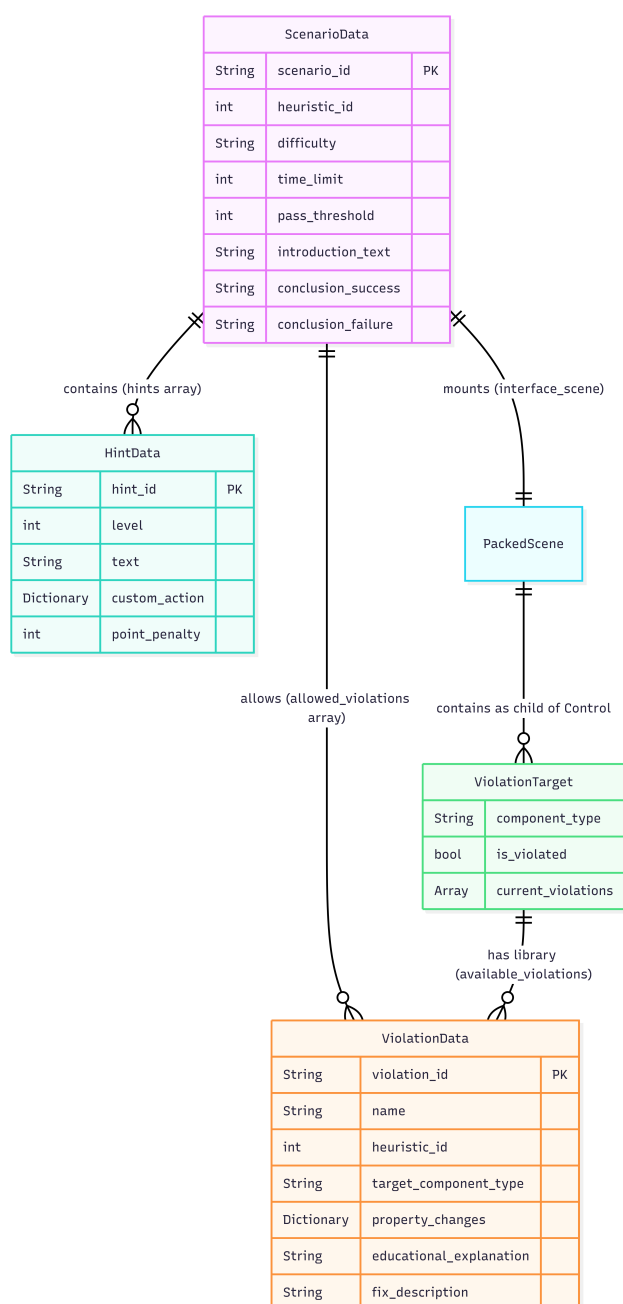


Fonte: Elaborado pelo autor, 2026.

APÊNDICE C - MODELO ER DOS *RESOURCES* .tres

O diagrama a seguir apresenta o modelo entidade-relacionamento dos *Resources* serializáveis da ferramenta, evidenciando a arquitetura *data-driven*. O *ScenarioData* agrega *HintData* e referencia uma *PackedScene* instanciada em tempo de execução, cujos *ViolationTarget* possuem bibliotecas de *ViolationData* injetáveis conforme o cenário ativo.

Figura 30 – Modelo entidade-relacionamento dos *Resources* .tres (visão *data-driven*)

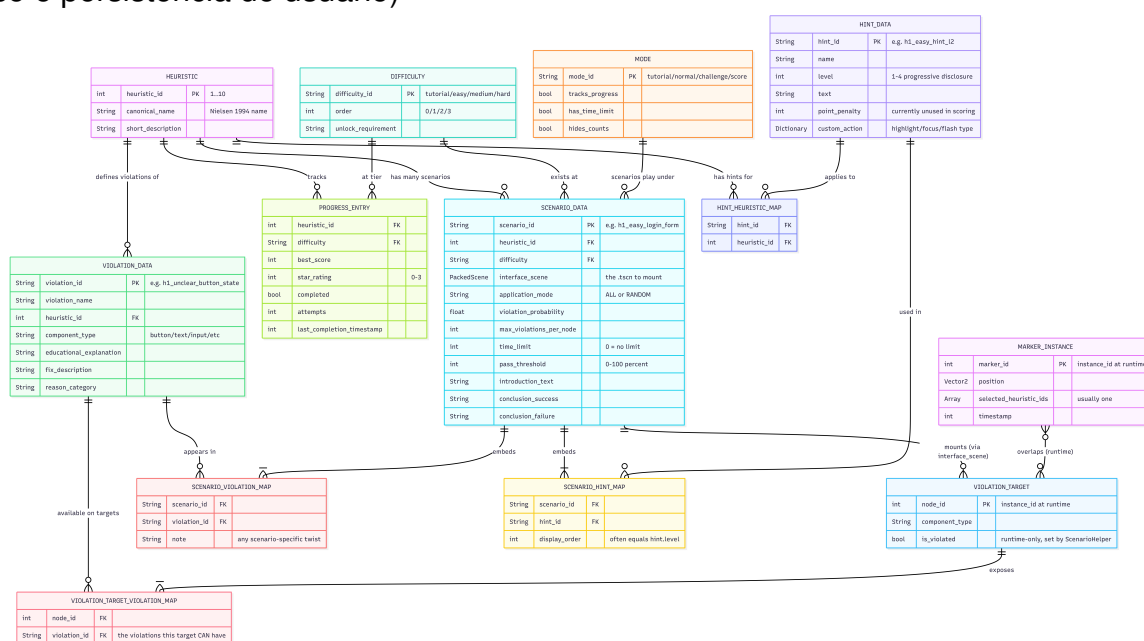


Fonte: Elaborado pelo autor, 2026.

APÊNDICE D - MODELO ER COMPLETO DA FERRAMENTA

O diagrama a seguir apresenta o modelo entidade-relacionamento completo do UX Detective, integrando o conteúdo didático estático (descrito pelos *Resources*) e o estado dinâmico de execução (mantido pelos *autoloads*). Além das entidades de conteúdo, inclui as entidades de persistência de usuário (USER_DATA, PROFILE_ENTRY, PROGRESS_ENTRY, MATCH) e as tabelas de mapeamento *many-to-many* (SCENARIO_VIOLATION_MAP, SCENARIO_HINT_MAP, VIOLATION_TARGET_VIOLATION_MAP).

Figura 31 – Modelo entidade-relacionamento completo da ferramenta (conteúdo didático e persistência de usuário)

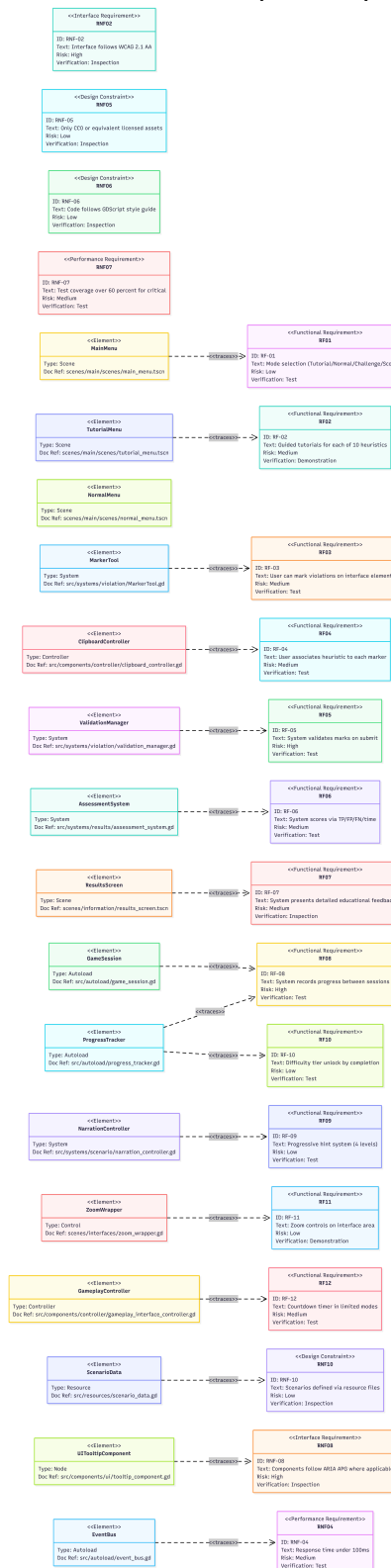


Fonte: Elaborado pelo autor, 2026.

APÊNDICE E - DIAGRAMA DE RASTREABILIDADE DE REQUISITOS

O diagrama a seguir apresenta a rastreabilidade entre requisitos funcionais e não funcionais, artefatos de código e cenas correspondentes, expressa em três camadas conectadas por relações de satisfação. Esta estrutura permite verificar, para cada requisito declarado no Quadro 11, quais artefatos do projeto o concretizam. Os requisitos sem elementos correspondentes (RNF-05, RNF-06 e RNF-07) representam práticas de desenvolvimento ou trabalhos futuros.

Figura 32 – Diagrama de rastreabilidade de requisitos para artefatos de código e cena

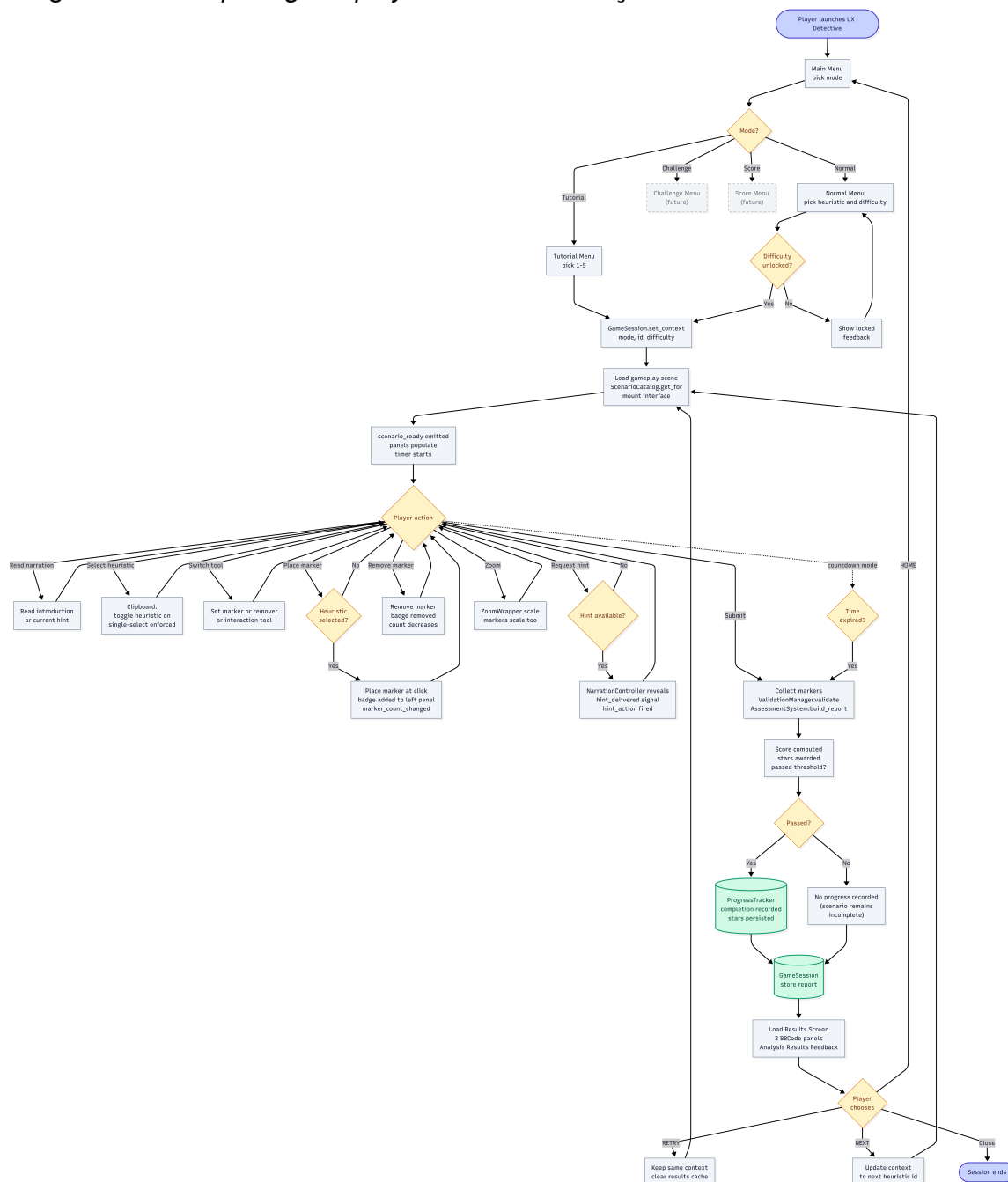


Fonte: Elaborado pelo autor, 2026.

APÊNDICE F - *LOOP DE GAMEPLAY* E CICLO DE AVALIAÇÃO

O fluxograma a seguir traduz, em forma visual, as nove etapas do ciclo de avaliação heurística descritas na Seção 3.3.5.4. O nó central *PlayLoop* concentra cada ação do jogador — ler narração, selecionar heurística, trocar ferramenta, marcar, remover marcação, aplicar *zoom* ou solicitar dica — até que a submissão ou o esgotamento do tempo conduzam o fluxo à validação, avaliação e apresentação dos resultados. Os pontos de persistência, marcados em cilindros verdes, ocorrem exatamente duas vezes por cenário: no registro de progresso e no armazenamento do relatório final; todo o restante é processado em memória. Os modos com bordas tracejadas indicam funcionalidades projetadas mas ainda não totalmente implementadas (os modos *Desafio* e *Score*).

Figura 33 – Loop de gameplay e ciclo de avaliação heurística no UX Detective



Fonte: Elaborado pelo autor, 2026.