
**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO PIAUÍ**
**CAMPUS PICOS**
**TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS**

**JOÃO PEDRO RUIDIVALLE MEDEIROS DE AMORIM**

# QUESTANSWER: UM JOGO DE TRÍVIA COM GERAÇÃO PROCEDURAL DE CONTEÚDO BASEADA EM INTELIGÊNCIA ARTIFICIAL GENERATIVA

**PICOS, PIAUÍ**
**2026**

JOÃO PEDRO RUIDIVALLE MEDEIROS DE AMORIM

# QUESTANSWER: UM JOGO DE TRÍVIA COM GERAÇÃO PROCEDURAL DE CONTEÚDO BASEADA EM INTELIGÊNCIA ARTIFICIAL GENERATIVA

Trabalho de Conclusão de Curso (Relatório Técnico de Software) apresentado como exigência parcial para obtenção do diploma do Curso de Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia do Piauí, Campus Picos.

**Orientador:** Prof. Dr. Daniel de Sousa Luz

PICOS, PIAUÍ
2026

JOÃO PEDRO RUIDIVALLE MEDEIROS DE AMORIM

# QUESTANSWER: UM JOGO DE TRÍVIA COM GERAÇÃO PROCEDURAL DE CONTEÚDO BASEADA EM INTELIGÊNCIA ARTIFICIAL GENERATIVA

Trabalho de Conclusão de Curso (Relatório Técnico de Software) apresentado como exigência parcial para obtenção do diploma do Curso de Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia do Piauí, Campus Picos.

Aprovada em 04 de Agosto de 2026.

BANCA EXAMINADORA:

---

**Prof. Dr. Daniel de Sousa Luz(Orientador)**
Instituto Federal do Piauí (IFPI)

---

**Prof. M$^{e}$. Anthony Irlan Marques Luz**
Instituto Federal do Piauí (IFPI)

---

**Prof. M$^{e}$. Manoel Messias Pereira Medeiros**
Instituto Federal do Piauí (IFPI)

PICOS, PIAUÍ
2026

Dedico esse trabalho a Deus, à minha família e aos meus amigos, pela presença, incentivo e por tornarem esta jornada possível.

## AGRADECIMENTOS

Agradeço a Deus, primeiramente, pela vida, força e sabedoria concedidas em cada etapa desta jornada. A Ele dedico toda a minha gratidão, pois sem a Sua presença e amparo diário, a realização deste trabalho não seria possível.

À minha família, pelo amor, suporte incondicional e por todo o esforço dedicado a garantir que eu pudesse focar integralmente na minha carreira acadêmica e profissional. O trabalho e a ajuda de vocês foram os pilares fundamentais para a concretização deste ciclo.

Aos meus amigos e colegas de jornada, em especial ao grupo conhecido como "Mantinhas", que me acompanharam ao longo desses anos, alguns desde os tempos de ensino médio e outros desde o início da graduação em Análise e Desenvolvimento de Sistemas. Agradeço por estarem presentes, por tornarem a rotina mais leve e divertida mesmo diante das dificuldades e, principalmente, por terem sido a centelha de inspiração que deu origem à ideia central deste Trabalho de Conclusão de Curso.

Aos professores do curso de Análise e Desenvolvimento de Sistemas do Instituto Federal do Piauí (IFPI) - Campus Picos, por compartilharem seus conhecimentos e contribuírem diretamente para a minha formação profissional.

Agradeço de forma especial ao professor Juciê, pelas importantes contribuições e direcionamentos durante a etapa de pré-projeto, e ao meu orientador, professor Dr. Daniel de Sousa Luz, pela condução atenciosa, paciência, suporte técnico e orientações valiosas que tornaram viável a realização deste trabalho.

*"O coração do prudente adquire o conhecimento, e o ouvido dos sábios busca a sabedoria."*
*Provérbios 18:15*

## RESUMO

Os jogos digitais do gênero trívia enfrentam um recorrente desafio de longevidade devido ao rápido esgotamento de seus acervos de perguntas e respostas estáticas, o que compromete o fator de re-jogabilidade e o engajamento dos usuários. Para mitigar essa limitação, este trabalho detalha o projeto, a implementação e a validação do *QuestAnswer*, um ecossistema de *software* móvel alimentado por Geração Procedural de Conteúdo via Aprendizado de Máquina (PCGML). A solução adota uma arquitetura cliente-servidor distribuída e desacoplada, composta por um aplicativo móvel reativo desenvolvido na Godot Engine, um servidor de aplicação Web API em .NET 9 conteinerizado via Docker e hospedado na nuvem (Render), uma camada de persistência relacional em PostgreSQL (Supabase) e uma esteira de ingestão em lote operada por uma ferramenta de linha de comando (CLI) em Node.js. Essa esteira orquestra inferências em Grandes Modelos de Linguagem (LLMs) por meio da API da GroqCloud, utilizando engenharia de *prompts* com histórico de termos salvos para prevenir duplicidades na origem e aplicando rotinas probabilísticas de desestabilização lúdica com pistas ambíguas (*trick hints*). A validação empírica do sistema foi conduzida com uma amostra de 23 estudantes do curso de Análise e Desenvolvimento de Sistemas do IFPI Campus Picos. Os resultados apontaram elevada usabilidade da interface (média de 9,57/10), alta estabilidade técnica (médias de falhas e travamentos abaixo de 2,5/10) e ampla aceitação da coerência das dicas sintéticas (8,30/10) e do divertimento proporcionado pelas pegadinhas (8,87/10). Adicionalmente, a condução de um Teste A/B evidenciou que grupos com acervos estáticos maiores (70+ cartas) e menores (15 cartas) relataram percepções equivalentes de exaustão rápida de conteúdo (7,23 vs. 7,40), comprovando empiricamente a alta velocidade de consumo do gênero e a necessidade de uma esteira de suprimentos contínua aliada a diretrizes formais de *game design*. Conclui-se que o ecossistema atingiu os objetivos propostos, demonstrando a viabilidade técnica, econômica e lúdica da PCGML aplicada aos jogos de entretenimento móvel.



**Palavras-chave**: Geração Procedural de Conteúdo; Inteligência Artificial Generativa; Grandes Modelos de Linguagem; Jogos de Trívia; Engenharia de Software.

**ABSTRACT**

Digital trivia games face a recurring longevity challenge due to the rapid exhaustion of their static question and answer sets, which compromises replayability and user engagement. To mitigate this limitation, this paper details the design, implementation, and validation of *QuestAnswer*, a mobile software ecosystem powered by Procedural Content Generation via Machine Learning (PCGML). The solution adopts a distributed and decoupled client-server architecture, consisting of a reactive mobile application developed in the Godot Engine, a .NET 9 Web API application server containerized via Docker and hosted in the cloud (Render), a relational persistence layer in PostgreSQL (Supabase), and a batch ingestion pipeline operated via a Node.js Command Line Interface (CLI) tool. This pipeline orchestrates Large Language Model (LLM) inferences through the GroqCloud API, applying prompt engineering with historical term injection to prevent duplications at the source, as well as probabilistic playful destabilization routines using ambiguous clues (*trick hints*). Empirical system validation was conducted with a sample of 23 Systems Analysis and Development students at IFPI Campus Picos. Results indicated high interface usability (mean 9.57/10), strong technical stability (low bug/crash incidence below 2.5/10), and broad approval of synthetic clue coherence (8.30/10) and trick hints fun factor (8.87/10). Furthermore, an A/B Test revealed that groups using larger (70+ cards) and smaller (15 cards) static card sets reported equivalent perceptions of rapid content exhaustion (7.23 vs. 7.40), empirically proving the genre's rapid consumption speed and the essential need for a continuous supply pipeline combined with formal game design guidelines. It is concluded that the ecosystem achieved its goals, demonstrating the technical, economic, and playful feasibility of PCGML applied to mobile entertainment games.

**Keywords**: Procedural Content Generation; Generative Artificial Intelligence; Large Language Models; Trivia Games; Software Engineering.

## LISTA DE ILUSTRAÇÕES

## LISTA DE TABELAS

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| ABNT | Associação Brasileira de Normas Técnicas |
| ADS | Análise e Desenvolvimento de Sistemas |
| API | *Application Programming Interface* (Interface de Programação de Aplicações) |
| APK | *Android Package* (Pacote Android) |
| BaaS | *Backend as a Service* (*Backend* como Serviço) |
| BNCC | Base Nacional Comum Curricular |
| CLI | *Command Line Interface* (Interface de Linha de Comando) |
| HTTP | *Hypertext Transfer Protocol* (Protocolo de Transferência de Hipertexto) |
| HTTPS | *Hypertext Transfer Protocol Secure* (Protocolo de Transferência de Hipertexto Seguro) |
| IA | Inteligência Artificial |
| IDE | *Integrated Development Environment* (Ambiente de Desenvolvimento Integrado) |
| IFPI | Instituto Federal de Educação, Ciência e Tecnologia do Piauí |
| JSON | *JavaScript Object Notation* (Notação de Objetos JavaScript) |
| LLM | *Large Language Model* (Grande Modelo de Linguagem) |
| ORM | *Object-Relational Mapping* (Mapeamento Objeto-Relacional) |
| PaaS | *Platform as a Service* (Plataforma como Serviço) |
| PCG | *Procedural Content Generation* (Geração Procedural de Conteúdo) |
| PCGML | *Procedural Content Generation via Machine Learning* (Geração Procedural de Conteúdo via Aprendizado de Máquina) |
| PGB | Pesquisa Game Brasil |
| PX | *Player Experience* (Experiência do Jogador) |

| | |
|---|---|
| REST | *Representational State Transfer* (Transferência de Estado Representacional) |
| RF | Requisito Funcional |
| RNF | Requisito Não Funcional |
| RPG | *Role-Playing Game* (Jogo de Interpretação de Papéis) |
| SGBD | Sistema Gerenciador de Bancos de Dados |
| TCC | Trabalho de Conclusão de Curso |
| UX | *User Experience* (Experiência do Usuário) |
| XML | *eXtensible Markup Language* (Linguagem de Marcação Extensível) |

## LISTA DE SÍMBOLOS

| | |
|---|---|
| % | Porcentagem |
| $N$ | Tamanho total da amostra populacional (Utilizado na Tabela de Avaliação de Uso) |
| $n$ | Tamanho do subgrupo / subamostra (Utilizado na Tabela do Teste A/B) |
| $+$ | Mais / Superior a (Utilizado para denotar "70+ cartas") |

**SUMÁRIO**

# 1 INTRODUÇÃO

O segmento de jogos eletrônicos tem demonstrado uma expansão constante no cenário mundial recente, acompanhado pelo incremento no volume de usuários. Nesse contexto, o mercado brasileiro apresenta crescimento contínuo, estimando-se que cerca de 75% da população consuma jogos digitais com frequência (Fortim *et al.*, 2022). Essa conjuntura comprova a relevância desses *softwares* como um dos eixos principais do entretenimento contemporâneo.

Mais do que simples itens de consumo, os jogos digitais configuram-se como experiências interativas cuja longevidade e triunfo dependem de modo direto do engajamento do público. Para tornar essa vivência significativa, faz-se indispensável propor desafios equilibrados, capazes de instigar o interesse e a retenção do jogador no decorrer do tempo (Costa, 2016). Sob essa ótica, a superação de barreiras e metas é tida como um dos componentes primordiais vinculados ao divertimento e à permanência na atividade lúdica (Salen; Zimmerman, 2012).

Contudo, em vertentes específicas, tais como títulos casuais, produções narrativas ou jogos de trívia, os enigmas e conteúdos estruturados costumam ser finitos. À medida que o usuário explora tais elementos por completo, o fator novidade e a curva de aprendizado decrescem, culminando no desinteresse e no consequente abandono do ecossistema (Koster, 2004). Logo, a escassez de conteúdo impõe um obstáculo severo para criadores que pretendem estender o ciclo de vida de produtos baseados em estruturas estáticas de testes.

Como resposta a esse impasse, mecanismos de Geração Procedural de Conteúdo (Procedural Content Generation, PCG) vêm sendo investigados como ferramentas para diversificar metas e multiplicar o acervo disponível. Em tempos recentes, a evolução dos Grandes Modelos de Linguagem (LLM) abriram caminhos inéditos para a concepção automatizada de dinâmicas instáveis, ampliando substancialmente o escopo de aplicação da PCG em ambientes virtuais (Farrokhi Maleki; Zhao, 2024).

Diante do exposto, este Relatório Técnico detalha o ecossistema do QuestAnswer,um jogo de trívia direcionado a dispositivos móveis alimentado por conteúdo gerado sinteticamente via inteligência artificial como engrenagem de geração procedural. O sistema opera como uma prova de conceito prática para ilustrar a viabilidade da integração de algoritmos de PCG baseados em IA no fluxo de perguntas e respostas, servindo de alicerce para futuros estudos acerca do aumento de re-jogabilidade em plataformas digitais.

## 1.1 OBJETIVOS

### 1.1.1 Geral

Este trabalho tem como objetivo o projeto, implementação e validação de um ecossistema de software para dispositivos móveis baseado em um jogo de trívia que utilize a inteligência artificial generativa como vetor para a geração procedural de conteúdo *offline*. Para alcançar o objetivo geral, foram definidos os seguintes objetivos específicos:

### 1.1.2 Específicos

1. Mapear e documentar os requisitos funcionais e não funcionais do ecossistema móvel e de suas ferramentas de apoio
2. Modelar a arquitetura distribuída do sistema, delimitando as fronteiras entre o cliente do aplicativo móvel, o microsserviço de *backend* e o mecanismo de ingestão automatizada.
3. Construir a interface do jogo de trívia para dispositivos móveis integrando animações fluidas e elementos visuais reativos.
4. Acoplar a elaboração iterativa de prompts à esteira de geração procedural para alimentar o banco de dados com estruturas textuais padronizadas.
5. Efetuar testes funcionais de software para inspecionar a resiliência e a corretude dos fluxos de dados implementados no protótipo.

# 2 FUNDAMENTAÇÃO TEÓRICA

Esta seção apresenta o embasamento teórico que sustenta o desenvolvimento do ecossistema *QuestAnswer*, abordando os conceitos de jogos digitais, a experiência do jogador baseada em desafios e a evolução da geração procedural de conteúdos por meio de inteligência artificial.

## 2.1 EXPERIÊNCIA DO JOGADOR E O FLUXO LÚDICO

Diferente dos softwares convencionais, cujo objetivo primordial é funcional e utilitário (Pressman; Maxim, 2016), os jogos digitais constituem um subconjunto mediado por sistemas computacionais com finalidade essencialmente experiencial, lúdica e voluntária (Adams, 2014; Costa, 2016). Salen e Zimmerman (Salen; Zimmerman, 2012) definem o jogo como um sistema estruturado por regras no qual os participantes enfrentam conflitos artificiais que resultam em consequências mensuráveis e com as quais estabelecem um forte vínculo emocional (Juul, 2005).

Dessa forma, a qualidade da interação do usuário com esse ecossistema hedônico é definida como Experiência do Jogador (*Player Experience*, PX) (González-Sánchez; Padilla-Zea; Vela, 2009). A PX expande os critérios tradicionais de Usabilidade (*User Experience*, UX), focados em eficácia e eficiência (Garrett, 2010), ao incorporar dimensões psicossociais complexas como a imersão, a motivação e a flutuação de emoções (Costa, 2016).

Para guiar e sustentar essa experiência ótima de engajamento profundo, Csikszentmihalyi (Csikszentmihalyi, 1990) propôs a Teoria do *Flow*, posteriormente transposta para o design de jogos por Sweetser e Wyeth (Sweetser; Wyeth, 2005) através do modelo *GameFlow*. O pilar central dessa dinâmica reside no estrito equilíbrio entre o nível de desafio proposto pelo sistema e as habilidades do jogador: obstáculos excessivamente complexos geram estados de ansiedade e frustração, enquanto estímulos insuficientes guiam o usuário diretamente ao tédio, culminando na quebra da experiência lúdica.

## 2.2 A LIMITAÇÃO DE CONTEÚDO EM JOGOS DE TRÍVIA

Conforme apontado na introdução deste trabalho, a manutenção do jogador no chamado Canal do Flow depende diretamente do fornecimento contínuo de novos problemas e metas (Csikszentmihalyi, 1990). Esse esgotamento de conteúdo é particularmente crítico em gêneros altamente dependentes de estruturas textuais estáticas, como os jogos casuais, narrativos ou de trívia, situação em que a quebra do fluxo de novidade culmina no tédio e no abandono precoce do software (Koster, 2004).

Sob essa perspectiva, a limitação de conteúdo configura-se como um dos principais desafios para a longevidade na engenharia de jogos, justificando o questionamento de (Schell, 2008): 'Como meu jogo pode gerar novos problemas para que os jogadores voltem sempre a ele?'.

## 2.3 GERAÇÃO PROCEDURAL DE CONTEÚDO VIA APRENDIZADO DE MÁQUINA

Historicamente, a Geração Procedural apoiava-se em algoritmos matemáticos determinísticos, como Perlin Noise ou Autômatos Celulares, que eram excelentes para gerar mapas e terrenos geográficos, mas ineficazes para gerar narrativas ou desafios lúdicos baseados em linguagem (Togelius; Yannakakis *et al.*, 2011).

Como resposta ao impasse do esgotamento de acervos, a Geração Procedural de Conteúdo (PCG) surge como uma estratégia de automação algorítmica voltada à criação autônoma de elementos do jogo, sejam eles componentes espaciais, mecânicos ou cenários narrativos (Togelius; Kastbjerg *et al.*, 2011; Korn *et al.*, 2017). No contexto do *QuestAnswer*, essa dinâmica ocorre de forma *offline*, isto é, o acervo de desafios é gerado, validado e armazenado em lote na infraestrutura de persistência antes da interação direta e da distribuição ao cliente móvel (McGathey, 2014).

Em vez de se apoiar em métodos construtivos tradicionais limitados a regras rígidas codificadas manualmente (Smith, 2014), o sistema adota a Geração Procedural de Conteúdo via Aprendizado de Máquina (*Procedural Content Generation via Machine Learning*, PCGML) (Summerville *et al.*, 2017). O paradigma da PCGML transfere o encargo de modelagem de regras para modelos matemáticos previamente treinados com grandes volumes de dados, permitindo a síntese autônoma de novos artefatos (Summerville *et al.*, 2017).

Dentre as vertentes mais recentes do aprendizado de máquina, destacam-se os Grandes Modelos de Linguagem (LLMs). Enquanto abordagens iniciais de PCGML focavam em Redes Neurais convolucionais e Cadeias de Markov (Summerville *et al.*, 2017), os LLMs trouxeram a capacidade inédita de produzir sentenças textuais semanticamente coerentes e adaptáveis. Por processarem a linguagem natural com alta precisão, os LLMs mostram-se a evolução ideal para a expansão de jogos de perguntas e respostas, superando as antigas gerações baseadas em gramáticas rígidas (Farrokhi Maleki; Zhao, 2024). Embora a PCG enfrente o risco de falhas catastróficas (*catastrophic failures*), quando o algoritmo gera saídas inviáveis ou sem nexo (Togelius; Yannakakis *et al.*, 2011), sua aplicação dirigida mitiga custos de criação manual contínua e sustenta a oferta de novos desafios, mantendo o jogador em constante estado de imersão e divertimento (Oliveira *et al.*, 2023).

## 2.4 TECNOLOGIAS UTILIZADAS

Nesta seção são apresentadas as tecnologias de *software* empregadas no desenvolvimento do ecossistema *QuestAnswer*. As ferramentas foram selecionadas com base em critérios de interoperabilidade, ausência de custos de licenciamento (*open-source* ou camadas gratuitas), maturidade no mercado e alta performance. Para garantir clareza estrutural, as tecnologias estão agrupadas de acordo com o módulo de atuação no sistema.

### 2.4.1 Módulo Jogo para Dispositivo Móvel

No âmbito do desenvolvimento de jogos digitais para dispositivos móveis, os motores de jogos (*game engines*) atuam como núcleos de abstração que viabilizam a criação do projeto e sua exportação multiplataforma (Thorn, 2011). Essas ferramentas fornecem ambientes integrados para gerenciamento de cenas, renderização gráfica, execução de rotinas de *script* e controle de áudio.

Para a construção do cliente móvel do *QuestAnswer*, selecionou-se a *Godot Engine* acompanhada de sua linguagem nativa de *script*, o GDScript. A *Godot* destaca-se por ser uma ferramenta totalmente de código aberto (*open-source*) sob a licença MIT, o que elimina riscos e custos de licenciamento comercial em produções independentes ou acadêmicas (Hurst *et al.*, 2026). A adoção da *Godot Engine* não se deu por preferência empírica, mas sim por critérios arquiteturais: além de isentar o projeto de custos de licenciamento e royalties corporativos devido à licença MIT, seu motor de renderização 2D nativo, tornando-a a escolha técnica ideal para smartphones de baixo desempenho.

### 2.4.2 Módulo Web API e Backend

Para viabilizar a comunicação entre os componentes do ecossistema, utiliza-se o conceito de Interfaces de Programação de Aplicações Web (*Web Application Programming Interfaces*, Web API), que consistem em conjuntos de rotinas e padrões padronizados para o intercâmbio de dados entre *softwares* via protocolos de rede (Biehl, 2016). O projeto adota o padrão arquitetural REST (*Representational State Transfer*), que estabelece restrições para a criação de serviços *web* escaláveis, simples e fracamente acoplados (Fielding, 2000).

A implementação da Web API RESTful do *QuestAnswer* apoia-se na plataforma .NET e no *framework* ASP.NET Core, utilizando a linguagem de programação C#. O .NET provê um ambiente de execução (*runtime*) robusto, multiplataforma e de alto desempenho (Liberty, 2005), enquanto o ASP.NET oferece infraestrutura nativa para roteamento, serialização de dados e gerenciamento de requisições HTTP em *Web* APIs

(Lock, 2023). A escolha da linguagem C# justifica-se por sua tipagem forte, orientação a objetos e amplo suporte corporativo.

Na camada de persistência de dados, utiliza-se o Sistema Gerenciador de Banco de Dados (SGBD) PostgreSQL. Trata-se de um banco de dados relacional de código aberto, amplamente reconhecido por sua estabilidade, conformidade com os padrões ACID e maturidade no tratamento de consultas (Pereira; Torres, 2023; Knijnik, 2022).

Para mitigar o descompasso de impedância (*impedance mismatch*), conflito conceitual entre o paradigma orientado a objetos das linguagens de programação e a estrutura relacional de tabelas adotada pelos SGBDs (Wang, 2024), utilizam-se ferramentas de Mapeamento Objeto-Relacional (*Object-Relational Mapping*, ORM), como o *Entity Framework Core* (EF Core). Tais componentes abstraem o acesso aos dados, automatizam o mapeamento de entidades e otimizam a execução de rotinas de persistência no banco de dados (Yaganti, 2020).

### 2.4.3 Módulo Gerador de Cartas

A esteira de suprimentos e automação de conteúdo do *QuestAnswer* apoia-se no ambiente de execução *Node.js*. O *Node.js* executa código JavaScript do lado do servidor utilizando o motor V8 da Google, dispensando a dependência de navegadores (Poglia; Bavaresco, 2025). Sua arquitetura assíncrona e orientada a eventos torna-o adequado para a construção de ferramentas de linha de comando (CLI) eficientes para coordenação de tarefas de rede e manipulação de arquivos JSON (Springer *et al.*, 2025).

Para a síntese automatizada de perguntas e respostas, a ferramenta consome Modelos de Linguagem de Grande Escala (*Large Language Models* – LLMs). Os LLMs consistem em sistemas de inteligência artificial baseados em redes neurais profundas treinadas em vastos volumes de dados textuais, com capacidade de compreender contextos, seguir instruções e gerar textos coerentes em linguagem natural (Inuwa-Dutse, 2025).

Para viabilizar a inferência dos LLMs com alta performance, o gerador conecta-se à infraestrutura em nuvem da *GroqCloud* (To, 2025). A *GroqCloud* atua como um serviço externo especializado no processamento acelerado de modelos de linguagem em nuvem, disponibilizando acesso a arquiteturas de peso aberto como o *openai/gpt-oss-20b* — um modelo de linguagem otimizado para tarefas de raciocínio em ambientes computacionais com restrição de recursos (Inuwa-Dutse, 2025). Essa infraestrutura permite delegar o esforço computacional do processamento sintético de texto para serviços especializados em nuvem.

### 2.4.4 Ferramentas de Apoio e Ambientes de Desenvolvimento Integrado

O suporte ao ciclo de desenvolvimento, edição de código, controle de versão e testes do ecossistema apoia-se em Ambientes de Desenvolvimento Integrado (*Integrated Development Environments*, IDEs) e ferramentas auxiliares. As IDEs consistem em aplicações de *software* que consolidam editores de código-fonte, ferramentas de compilação, depuradores e recursos de automação em uma interface unificada, otimizando a produtividade do desenvolvedor e a manutenção do código (Pressman; Maxim, 2016). No ecossistema do *QuestAnswer*, empregaram-se as IDEs JetBrains Rider e Visual Studio Code para a construção dos módulos de *backend* e *scripts* de automação.

### 2.4.5 Consolidação da Especificação Técnica

Com o objetivo de mapear, caracterizar e consolidar a especificação técnica de todas as ferramentas de engenharia de *software* empregadas no ecossistema do *QuestAnswer*, a Tabela 1 apresenta uma visão unificada dos componentes detalhados nesta seção. A tabela categoriza cada tecnologia de acordo com o módulo de atuação no ecossistema, sua versão de implantação, o tipo de sua licença de uso e sua respectiva categoria.

Tabela 1 – Especificação e categorização das tecnologias utilizadas no *QuestAnswer*.

| **Tecnologia** | **Versão** | **Módulo** | **Licença** | **Categoria** |
|---|---|---|---|---|
| Godot Engine | 4.6/7 | Jogo Mobile | Open-Source | Motor de Jogo |
| GDScript | 4.6/7 | Jogo Mobile | Open-Source | Linguagem de Script |
| .NET | 9.0 | Backend | Open-Source | Plataforma de Desenvolvimento |
| ASP.NET | 9.0 | Backend | Open-Source | Framework para desenvolver Web APIs |
| C# | 13 | Backend | Open-Source | Linguagem Orientada Objetos |
| Entity Framework | 9.0.8 | Backend | Open-Source | Framework ORM |
| PostgreSQL | 16.14 | Backend | Open-Source | SGBD Relacional |
| Docker | 29.1.3 | Backend | Open-Source | Conteinerização |
| Render | - | Backend | Proprietária | PaaS (Plataforma como Serviço) |
| Supabase | - | Backend | Open-Core | BaaS (Backend como Serviço) |
| Node.js | 24.15.0 | Gerador de Cartas | Open-Source | Ambiente de Execução |
| GroqCloud | - | Gerador de Cartas | Proprietária | API de Inferência |

| **Tecnologia** | **Versão** | **Módulo** | **Licença** | **Categoria** |
|---|---|---|---|---|
| openai/gpt-oss-20b | - | Gerador de Cartas | Open-Source | LLM |
| Rider | 2026.1.3 | IDE | Proprietária | IDE de Desenvolvimento |
| Visual Studio Code | 1.124.0 | IDE | Proprietária | Editor de Código Extensível |
| Yaak | 2025.5.5 | Testes | Open-Source | Cliente de API HTTP/REST |
| Git | 2.43.0 | Versionamento | Open-Source | Controle de Versão Distribuído |
| GitHub | - | Versionamento | Proprietária | Plataforma de Repositórios |

Fonte: Elaborado pelo autor (2026)

# 3 TRABALHOS RELACIONADOS

A literatura em Engenharia de Software e Desenvolvimento de Jogos apresenta diversas abordagens para mitigar o gargalo da produção manual de conteúdo e ampliar o fator de replay em aplicações lúdicas. Para mapear o estado da arte e evidenciar as contribuições do ecossistema QuestAnswer, esta seção analisa três trabalhos correlatos que empregam técnicas de Geração Procedural de Conteúdo (PCG) em jogos educativos, geradores de mapas e jogos de puzzle.

## 3.1 ANÁLISE DOS TRABALHOS CORRELATOS

Voltado ao contexto de jogos educativos, Araújo e Aranha (2018) propuseram um modelo de geração procedural de fases utilizando gramáticas formais de reescrita. A abordagem dos autores estrutura-se em uma esteira de três etapas sequenciais: (i) geração da estrutura básica do cenário; (ii) injeção de componentes pedagógicos alinhados às diretrizes da Base Nacional Comum Curricular (BNCC); e (iii) inclusão de elementos adicionais de entretenimento e mecânicas de dificuldade. A técnica foi avaliada na criação de fases jogáveis para os jogos *Bomberman* e *Sokoban*.

O estudo de Araújo e Aranha (2018) aproxima-se do *QuestAnswer* ao buscar superar o alto custo da criação manual de conteúdo em jogos com propósitos específicos por meio de algoritmos automatizados.

Contudo, enquanto os autores utilizam gramáticas formais estáticas focadas em posicionamento espacial e geométrico 2D, o *QuestAnswer* emprega Geração Procedural de Conteúdo via Aprendizado de Máquina (PCGML) apoiada em Grandes Modelos de Linguagem (LLMs), direcionando a automação para a síntese semântica e textual de desafios acompanhados de rotinas probabilísticas de desestabilização lúdica (*trick hints*).

No âmbito da geração de cenários para jogos de ação e RPG, Leite e Lima (2012) desenvolveram um algoritmo recursivo para a construção procedural de mapas bidimensionais contínuos, tais como cavernas, calabouços e ilhas. A solução proposta opera em três fases: geração recursiva de terrenos com sorteio estocástico de direções cardeais, validação de tamanho mínimo da área útil e correção de coesão espacial por rotação de *tiles* e remoção de paredes inválidas.

O trabalho de Leite e Lima (2012) compartilha com esta pesquisa o objetivo central de estender o tempo de vida útil do *software* e prover experiências diversificadas a cada sessão sem dependência de intervenção humana contínua.

Todavia, enquanto a solução de Leite e Lima (2012) é voltada estritamente à topologia e disposição espacial de elementos locais exportados em arquivos XML, o

*QuestAnswer* opera sob uma arquitetura distribuída cliente-servidor (Godot + .NET + PostgreSQL), na qual o foco reside na geração e validação de acervos textuais na nuvem com barreira de prevenção de redundâncias na origem.

Investigando diretamente a limitação de re-jogabilidade em jogos do gênero *puzzle*, Santos (2022) projetou o jogo *Color Maze*, acompanhado de um gerador construtivo de desafios do tipo *path-building* e um solucionador (*solver*) baseado no algoritmo A* para validação prévia de solubilidade. O autor conduziu avaliações quantitativas e qualitativas com usuários para mensurar a percepção de consistência de dificuldade em diferentes tamanhos de tabuleiro e tipos de labirinto (perfeitos vs. trançados).

O trabalho de Santos (2022) possui forte alinhamento teórico com a fundamentação do *QuestAnswer*, uma vez que ambos partem do diagnóstico de que o interesse por jogos de enigma decai drasticamente assim que o jogador descobre as soluções de um acervo estático.

Entretanto, enquanto Santos (2022) foca na resolução de caminhos numéricos/visuais em grades 2D validados por um *solver* determinístico local na *engine* Unity, o *QuestAnswer* enfrenta o desafio da geração textual dedutiva por inteligência artificial generativa, contornando a previsibilidade do modelo de linguagem por meio de uma esteira de ingestão em lote desacoplada via CLI (Node.js) e filtragem de histórico no cliente móvel.

Para sintetizar as diferenças arquiteturais e conceituais discutidas ao longo desta seção e evidenciar a lacuna tecnológica preenchida por esta pesquisa, a Tabela 2 apresenta um quadro comparativo direto entre as soluções analisadas e o ecossistema proposto.

Tabela 2 – Comparativo entre os trabalhos correlatos e a solução proposta.

| **Trabalho** | **Autores** | **Contexto de Aplicação** | **Abordagem da PCG** |
|---|---|---|---|
| Geração Procedural de Conteúdo para Criação de Fases de Jogos Educativos usando Gramática | Araújo & Aranha (2018) | Jogos Educativos (Fases 2D) | Gramáticas formais de reescrita focadas em topologia espacial e injeção de regras pedagógicas. |
| Geração procedural de mapas para jogos 2D | Leite & Lima (2012) | Jogos de Ação e RPG (Mapas 2D) | Algoritmos recursivos com sorteio estocástico e exportação local em arquivos XML. |
| Geração procedural de puzzles para o desenvolvimento de jogos | Santos (2022) | Jogos de Enigma / *Puzzles* | Construção de caminhos (*path-building*) validada localmente por algoritmo determinístico (Solver A*). |
| ***QuestAnswer* (Presente Trabalho)** | **Amorim (2026)** | **Trívia Mobile (Enigmas Textuais)** | **PCGML via LLMs focada em semântica textual, processamento em nuvem e desestabilização lúdica (*trick hints*).** |

Fonte: Elaborado pelo autor (2026).

# 4 APLICAÇÃO PROPOSTA

Esta seção detalha como o projeto foi planejado e modelado, incluindo a estrutura e os processos que foram implementados.

## 4.1 LEVANTAMENTO DE REQUISITOS

A Tabela 3 apresenta os requisitos funcionais (RF) identificados para o ecossistema *QuestAnswer*. O mapeamento dessas necessidades fundamentou-se na observação das limitações e interrupções ocorridas durante partidas do jogo de tabuleiro tradicional *Perfil*, servindo de base para o projeto das funcionalidades essenciais e para a especificação da arquitetura do sistema.

Tabela 3 – Requisitos funcionais do ecossistema QuestAnswer.

| **Código** | **Requisito** | **Descrição** |
|---|---|---|
| RF01 | Geração procedural de conteúdo | O sistema deve automatizar a concepção de novos baralhos de trívia estruturados (JSON) por meio de modelos generativos de linguagem via GroqCloud API. |
| RF02 | Ingestão dirigida via CLI | O utilitário de linha de comando deve prover uma interface assistida para que o administrador selecione categorias e dispache novos lotes de cartas à Web API. |
| RF03 | Prevenção de redundâncias na origem | O módulo gerador deve interrogar os registros existentes no servidor via requisição HTTP GET e injetar os termos já salvos no prompt para mitigar duplicidades. |
| RF04 | Desestabilização de dificuldade | O algoritmo do gerador deve aplicar rotinas de desestabilização lúdica, substituindo aleatoriamente até três das dez pistas originais por sentenças ambíguas (*trick hints*). |
| RF05 | Filtragem dirigida pelo cliente | O aplicativo móvel deve registrar localmente os identificadores das cartas já exibidas e transmiti-los via parâmetro de consulta (*query string*) para isolar desafios inéditos. |
| RF06 | Sincronização e controle de partida | O cliente móvel deve gerenciar o ciclo de vida lúdico das rodadas, operando contadores temporizados para revelação progressiva de dicas e atualização do placar dinâmico. |

Fonte: Elaborado pelo autor (2026)

A Tabela 4 apresenta os requisitos não funcionais que o ecossistema do jogo precisa ter para garantir uma experiência agradável aos jogadores:

Tabela 4 – Requisitos não funcionais do ecossistema QuestAnswer.

| **Código** | **Requisito** | **Descrição** |
|---|---|---|
| RNF01 | Escalabilidade e disponibilidade | O conteúdo textual do banco de dados relacional deve ser expansível de forma assíncrona desacoplada, garantindo fluxo infinito de dados sem necessidade de recompilar o cliente. |
| RNF02 | Resiliência da interface de usuário | A aplicação móvel deve interceptar falhas ou esgotamento completo de registros ativos no servidor, suspendendo requisições em segundo plano para evitar loops de carregamento eterno ou congelamento de tela. |
| RNF03 | Otimização de renderização gráfica | A interface desenvolvida na Godot Engine deve empregar estruturas leves e emissores de partículas nativos (*CPUParticles2D*), mitigando gargalos de processamento em *smartphones* de baixo desempenho. |
| RNF04 | Autenticação e segurança de endpoints | Os canais de escrita da Web API RESTful (.NET) devem ser protegidos por meio de um *middleware* de validação simétrica, exigindo obrigatoriamente o metadado *X-API-KEY* nos cabeçalhos HTTP. |
| RNF05 | Interoperabilidade de dados | O tráfego de dados entre o gerador CLI (Node.js), o servidor de aplicação (.NET) e o cliente móvel (GDScript) deve ser padronizado rigidamente por meio de contratos de intercâmbio no formato JSON. |

Fonte: Elaborado pelo autor (2026)

## 4.2 ARQUITETURA DO SISTEMA

Para viabilizar o cumprimento dos requisitos levantados, o ecossistema do QuestAnswer foi projetado sob um modelo arquitetural cliente-servidor distribuído, focado no desacoplamento de responsabilidades e na interoperabilidade baseada em contratos de dados padronizados. Sendo estruturado em três diferentes módulos principais, Jogo Mobile, Web API e Gerador de Cartas. A infraestrutura integra processamento local, microsserviços na nuvem e o consumo de inteligência artificial de forma independente.

O Jogo Mobile, desenvolvido utilizando a Godot Engine, opera na ponta final do ecossistema e gerencia o ciclo de vida lúdico das rodadas, a interface reativa e a organização das pontuações e identificação dos jogadores. O sistema de rodadas foi estruturado intencionalmente para fornecer propósito e interatividade, transformando o acervo procedural em um meio dinâmico para a obtenção de vitórias. Para otimizar o armazenamento em smartphones de baixo desempenho e garantir flexibilidade, as cartas não são salvas localmente no dispositivo. Em vez disso, o cliente móvel realiza requisições assíncronas à Web API contendo os identificadores (IDs) das cartas já exibidas e armazenadas localmente na sessão, isolando desafios inéditos e eliminando a necessidade de publicar novas versões do aplicativo apenas para expandir o bancode

dados.

A Web API atua como o núcleo das regras de negócios e o ponto central de intermediação de mensagens entre os módulos. Desenvolvido em .NET, encapsulado via Docker e hospedado na plataforma Render, o servidor expõe as entidades de controle restritas à classe de cartas do jogo. Como barreira de segurança contra acessos maliciosos e tentativas de alteração ou exclusão indevida do acervo, implementou-se um middleware de autenticação simétrica simples. Através do metadado X-API-KEY inserido nos cabeçalhos HTTP, o servidor valida de forma direta se o emissor possui privilégios de escrita ou leitura antes de processar as transações.

A camada de persistência é isolada em um banco de dados relacional PostgreSQL gerenciado pela plataforma Supabase, sendo dedicada exclusivamente ao armazenamento estável das cartas estruturadas. A comunicação entre o servidor de aplicação e o banco de dados é integralmente mediada pelo framework ORM Entity Framework Core, que traduz as operações orientadas a objetos da API para o modelo relacional de forma nativa e segura.

O ecossistema de dados é alimentado de forma externa pelo Gerador de Cartas, um utilitário desenvolvido em Node.js que opera localmente como uma esteira de suprimentos controlada pelo administrador via ferramenta de linha de comando (CLI). O fluxo de geração e validação dos desafios é executado sob as seguintes etapas sequenciais:

1. O operador especifica via terminal a quantidade de cartas e as categorias desejadas. O utilitário interroga a Web API por meio de uma requisição HTTP GET para recuperar os títulos já existentes da categoria escolhida e injeta esses termos em um arquivo JSON de diretrizes. Esse processo garante que o modelo de linguagem (*openai/gpt-oss-20b*) receba o contexto no *prompt*, mitigando redundâncias e duplicidades na origem;

2. Após a montagem do *payload*, o utilitário dispara a solicitação de geração via HTTPS para a API da GroqCloud. Ao receber o retorno textual cru, o gerador efetua a tradução da resposta para objetos JavaScript;

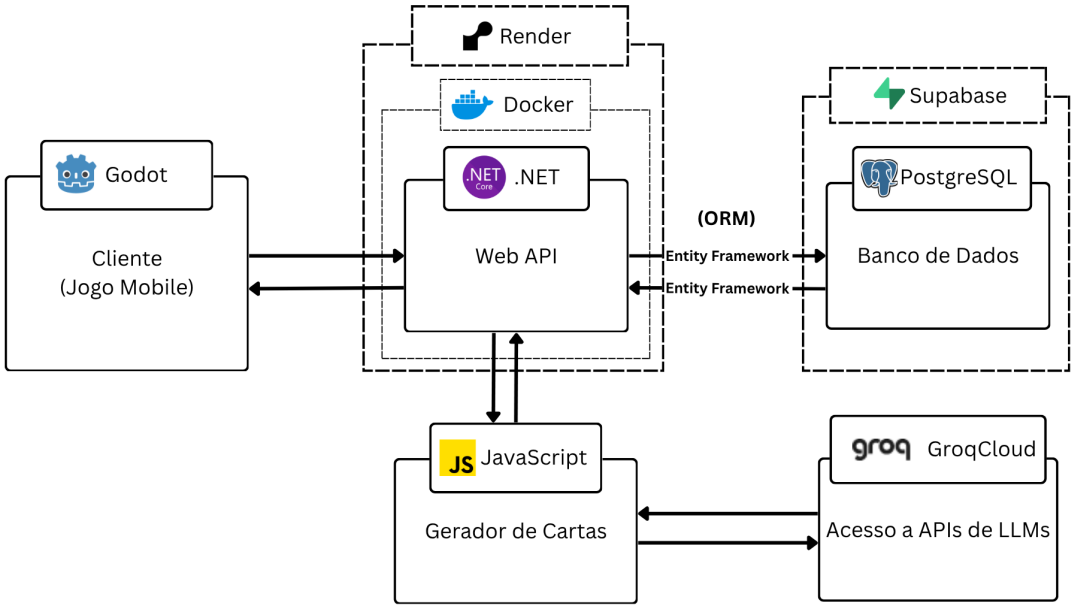
3. O algoritmo inspeciona a integridade estrutural da carta, avaliando o título, a categoria correspondente e a presença exata das dez pistas requeridas, e aplica rotinas probabilísticas para incrementar a diversão. Caso o sorteio interno determine a inserção de elementos ambíguos, o sistema substitui aleatoriamente até três das pistas originais por sentenças de duplo sentido (*trick hints*), que estão armazenadas dentro da aplicação;

4. Uma vez validada e modificada, a carta final em formato JSON é despachada via requisição HTTP POST autenticada para gravação permanente no banco de

dados através da Web API.

A Figura 1 ilustra graficamente o fluxo de integração e as conexões estabelecidas por esses módulos.

Figura 1 – Diagrama de arquitetura macro e fluxo de dados do ecossistema QuestAnswer.




Fonte: Elaborado pelo autor (2026).

Essa divisão de responsabilidades garante a total independência dos serviços: o cliente móvel consome apenas dados prontos e otimizados, o servidor de aplicação mantém alta disponibilidade focado exclusivamente nas regras de distribuição, e a alta carga computacional da inteligência artificial generativa fica isolada na esteira de suprimentos do gerador.

## 4.3 IMPLANTAÇÃO

Com o propósito de garantir que os serviços do ecossistema funcionem de forma remota e ininterrupta, viabilizando o acesso contínuo aos componentes do sistema, realizou-se a implantação da infraestrutura de *backend* em nuvem. Para isso, a Web API foi encapsulada utilizando a tecnologia de conteinerização *Docker*, assegurando a padronização do ambiente de execução independentemente da máquina hospedeira. O *deploy* da aplicação foi executado na plataforma *Render*, atuando como uma Plataforma como Serviço (*Platform as a Service*, PaaS), a qual foi integrada diretamente ao repositório do projeto no GitHub. Dessa forma, estabeleceu-se um fluxo de entrega contínua (*Continuous Deployment*), no qual cada novo *commit* na ramificação principal (*main branch*) engatilha automaticamente a compilação e a atualização

do serviço em nuvem. Paralelamente, a camada de persistência em *PostgreSQL* foi implantada na plataforma *Supabase*, operando como um ecossistema de *Backend as a Service* (BaaS). Essa arquitetura distribuída assegura a alta disponibilidade do ecossistema, permitindo que tanto o utilitário gerador de cartas quanto a aplicação *mobile* se comuniquem com a API em tempo real, sem dependência de servidores locais e isento de custos operacionais.

Por sua vez, a disponibilização do aplicativo *mobile* aos usuários finais foi realizada utilizando o recurso nativo de exportação da *Godot Engine* voltado ao sistema operacional *Android*. Esse processo consolida as cenas, scripts em GDScript, ativos gráficos e arquivos de áudio em um único pacote binário de instalação (*Android Package*, APK) de porte leve e otimizado. Em razão das taxas de licenciamento associadas à conta de desenvolvedor na loja oficial da plataforma (*Google Play Store*) e considerando o caráter experimental do protótipo acadêmico, optou-se pela distribuição direta do arquivo de instalação por meio do armazenamento em nuvem no *Google Drive*. Essa estratégia viabilizou o *download* e a instalação direta nos dispositivos móveis dos voluntários durante as etapas de teste de forma prática, controlada e sem custos adicionais.

## 4.4 TESTES

A verificação da corretude e da estabilidade do ecossistema *QuestAnswer* foi conduzida por meio de uma abordagem iterativa de testes funcionais e de integração, acompanhando continuamente o ciclo de desenvolvimento de cada módulo. Essa estratégia permitiu validar antecipadamente os contratos de intercâmbio de dados, a segurança dos canais de comunicação e a fluidez da experiência lúdica no aplicativo móvel.

Na camada de *backend*, os testes de integração das rotas da Web API RESTful foram realizados de forma isolada com o auxílio do cliente HTTP *Yaak*. Durante essa etapa, inspecionou-se o comportamento dos *endpoints* diante de requisições do tipo *GET* e *POST*, avaliando o correto processamento dos *payloads* estruturados em formato JSON. Além disso, validou-se a eficácia do *middleware* de autenticação simétrica, certificando que requisições desprovidas do metadado *X-API-KEY* nos cabeçalhos fossem prontamente rejeitadas com os códigos de status HTTP apropriados.

No entanto, os testes do módulo gerador de cartas ocorreu de forma iterativa por meio de execuções diretas e consecutivas na ferramenta de linha de comando (CLI). Durante esses testes operacionais, avaliou-se a comunicação com a API da *GroqCloud* para garantir a correta conversão do retorno sintético do modelo de linguagem em objetos válidos pelo *parser* em *Node.js*. Assegurou-se, assim, a conformidade do esquema de dados, a integridade da injeção de pistas ambíguas e o envio bem-sucedido das cartas para o servidor .NET antes de sua gravação definitiva no banco de

dados.

Por sua vez, a validação do aplicativo *mobile* ocorreu de forma funcional e dinâmica diretamente no ambiente de execução e depuração nativo da *Godot Engine*. Nessa fase, foram simuladas partidas completas para inspecionar o ciclo de vida das rodadas, a contagem temporizada dos pontos, a revelação progressiva das dicas e a atribuição correta das pontuações aos jogadores. Testou-se também o mecanismo de filtragem de cartas no cliente, confirmando que os identificadores das cartas jogadas fossem gravados localmente e anexados às requisições, isolando desafios inéditos sem comprometer a memória ou o desempenho de renderização do dispositivo.

# 5 RESULTADOS

Este capítulo apresenta os resultados práticos obtidos com o desenvolvimento e a validação do ecossistema *QuestAnswer*. As seções a seguir detalham a materialização do sistema por meio de um estudo de caso da interface gráfica, a avaliação quantitativa e qualitativa de uso conduzida com os estudantes, e a discussão dos achados experimentais sob a ótica da engenharia de *software* e do *game design*.

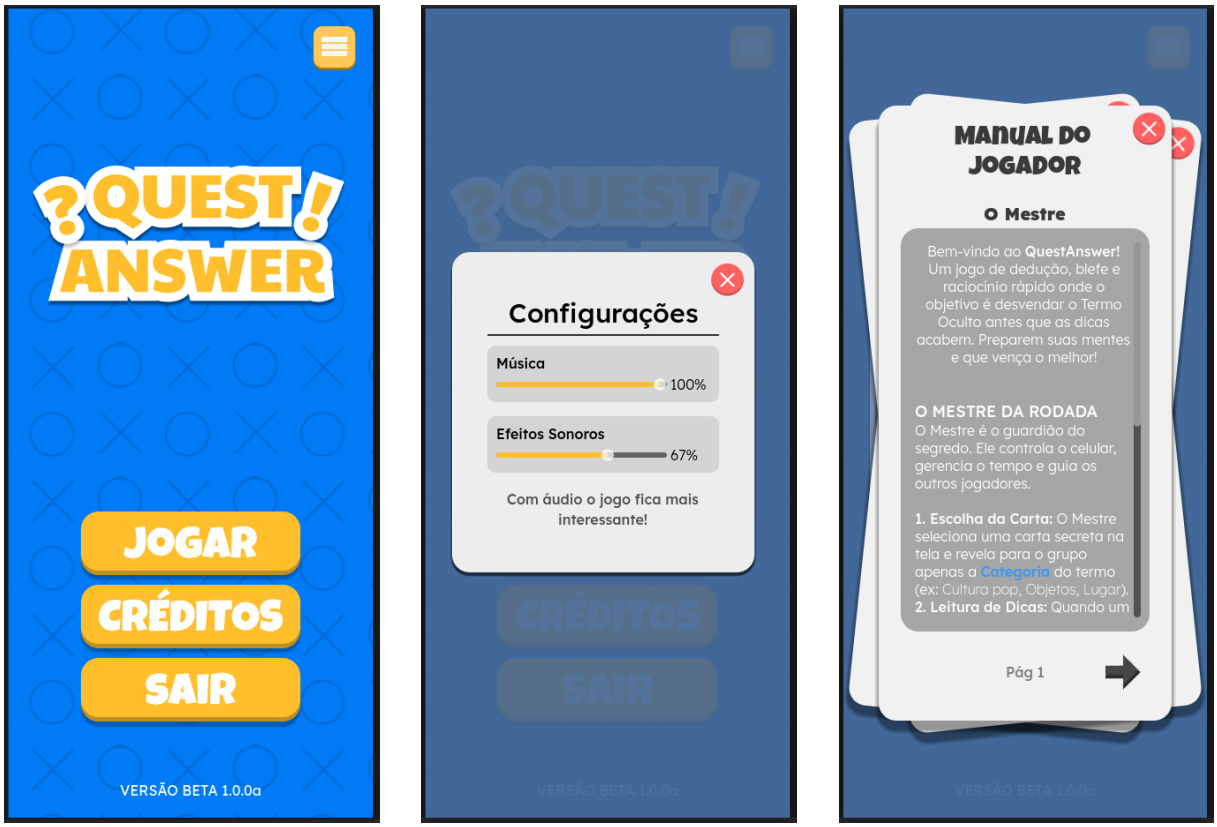
## 5.1 ESTUDO DE CASO: INTERFACE E NAVEGAÇÃO

Como resultado do desenvolvimento do projeto, o ecossistema *QuestAnswer* teve sua interface gráfica totalmente construída e integrada no cliente móvel, concretizando o Objetivo Específico 3. As telas finais resultantes entregam a identidade visual do jogo e garantem a usabilidade planejada na fase de concepção, cujos protótipos de alta fidelidade constam nos Apêndices B e C.

### 5.1.1 Tela Inicial, Configurações e Manual

A Tela Inicial desenvolvida (Figura 2a) consolida a porta de entrada do aplicativo, apresentando uma hierarquia visual clara, destaque para a marca do *QuestAnswer* e botões de ação direta. Por meio dessa interface funcional, o usuário inicia a partida, acessa as opções do sistema, visualiza os créditos ou encerra a aplicação. O menu de configurações resultante (Figura 2b) opera na forma de um painel sobreposto (*modal*), permitindo ao jogador ajustar independentemente os volumes da trilha e dos efeitos sonoros por meio de barras deslizantes. Adicionalmente, a navegação de suporte entrega o Manual do Jogador (Figura 2c), um componente interativo estruturado em páginas que disponibiliza as regras do jogo, a definição do papel do Mestre da rodada e a dinâmica de pontuação.

Figura 2 – Interface de acesso, menus de configurações e manual do ecossistema QuestAnswer.




(a) Tela Inicial (b) Menu de Configurações (c) Manual do Jogador

Fonte: Elaborado pelo autor (2026).

### 5.1.2 Fluxo de Preparação e Loop Principal de Jogabilidade

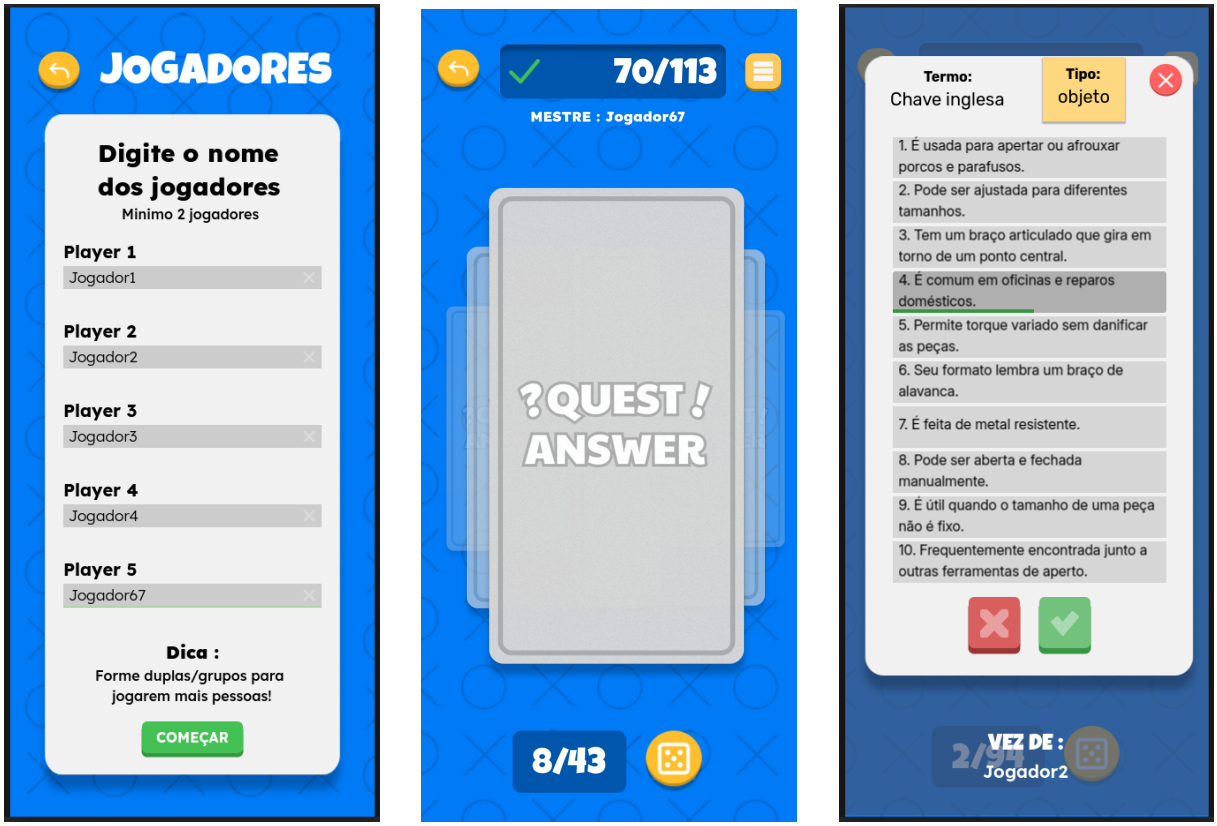
O ciclo principal de interação (*core loop*) do aplicativo foi materializado por meio de interfaces integradas de preparação e execução das rodadas, comprovando o atendimento ao Requisito Funcional RF06 (Sincronização e controle de partida). A interface de cadastro obtida (Figura 3a) realiza o registro dinâmico dos participantes da sessão, validando a restrição de no mínimo dois jogadores e exibindo orientações para a formação de duplas ou grupos.

Após a confirmação dos participantes, o aplicativo avança para a tela de escolha de carta (Figura 3b), que exibe o verso do baralho oculto, o identificador do Mestre atual da rodada, o progresso no acervo total (ex.: 70/113) e o índice da carta atual em relação à contagem de cartas inéditas disponíveis na sessão (ex.: 8/43), resultado que evidencia o funcionamento prático da filtragem no cliente estipulada pelo RF05.

Ao selecionar a carta, a interface de controle do Mestre (Figura 3c) é renderizada, apresentando o termo oculto e a categoria correspondente, acompanhados das dez pistas sequenciais. Esta tela entrega em tempo real barras temporizadas de

progresso, botões de validação de acerto e erro, e controles de revelação individual de dicas, exibindo de forma prática as sentenças ambíguas (*trick hints*) geradas em conformidade com o RF04.

Figura 3 – Interface de registro de jogadores, seleção de baralho e controle de dicas na partida.




(a) Tela de Jogadores (b) Escolha de Carta (c) Interface da Carta

Fonte: Elaborado pelo autor (2026).

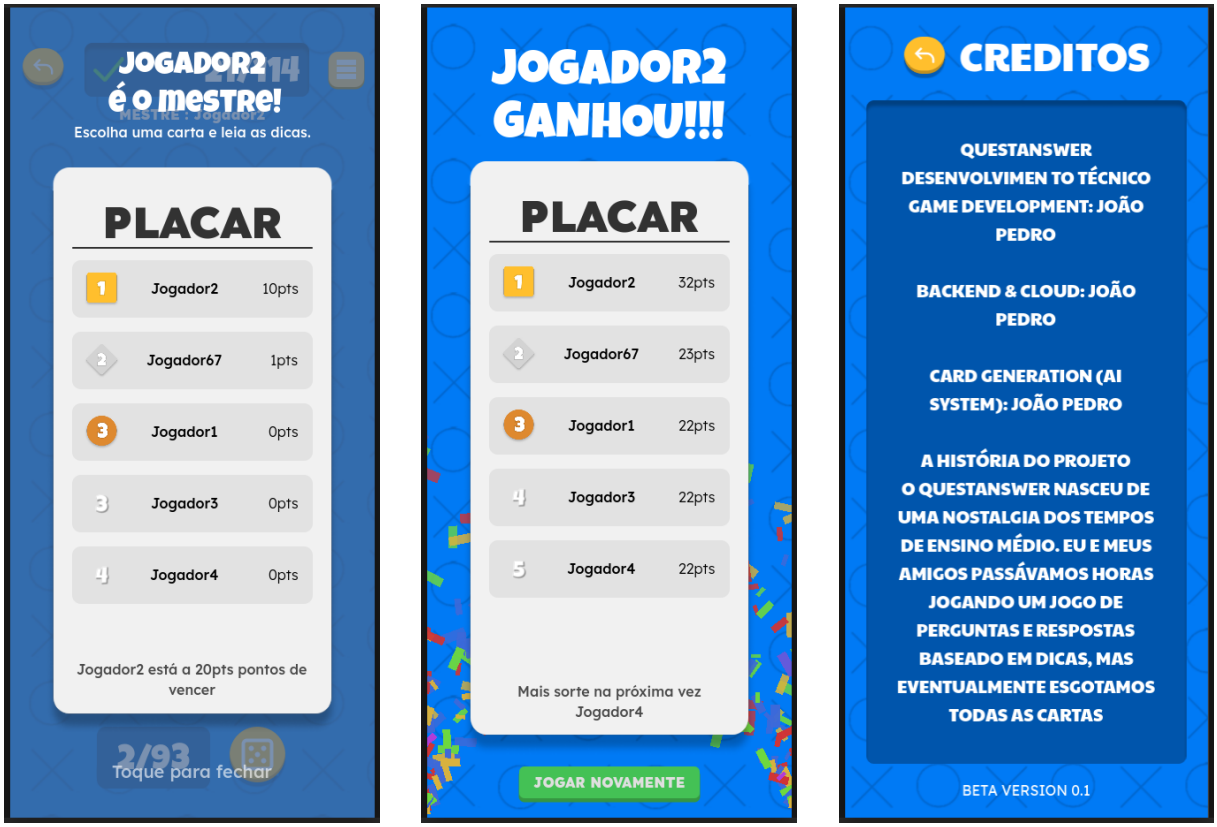
### 5.1.3 Acompanhamento, Vitória e Créditos

O acompanhamento da partida e o encerramento do jogo foram viabilizados por meio de interfaces reativas de *feedback* desenvolvidas no cliente móvel. Entre as rodadas, a tela de placar dinâmico construída (Figura 4a) exibe a classificação atualizada dos jogadores em ordem decrescente de pontuação, indica quem assumirá o papel de Mestre na rodada seguinte e calcula a margem de pontos necessária para atingir a condição de vitória.

Quando um jogador alcança a pontuação teto estipulada, o aplicativo aciona a tela de vitória (Figura 4b), executando um *feedback* visual enriquecido com emissores de partículas nativos para celebrar o campeão e disponibilizar o botão de reinício da sessão. A execução fluida e sem queda de quadros desses nós gráficos otimizados demonstra diretamente o cumprimento do Requisito Não Funcional RNF03 (Otimização

de renderização gráfica). Por fim, a tela de créditos (Figura 4c) formaliza a autoria do ecossistema, categorizando as responsabilidades de engenharia de *software* (*game development*, *backend*, nuvem e IA) e registrando a motivação da criação do projeto.

Figura 4 – Interface de placar dinâmico, tela de vitória final e créditos do desenvolvimento do jogo.




(a) Placar Dinâmico (b) Tela de Vitória (c) Tela de Créditos

Fonte: Elaborado pelo autor (2026).

## 5.2 AVALIAÇÃO DE USO

Esta seção apresenta os dados empíricos quantitativos e qualitativos coletados no formulário de validação do aplicativo, concretizando o **Objetivo Específico 5** do trabalho.

### 5.2.1 Amostragem e Metodologia de Coleta

Os testes de validação do ecossistema *QuestAnswer* foram realizados com uma amostra de 23 estudantes do curso de Análise e Desenvolvimento de Sistemas (ADS) do Instituto Federal do Piauí (IFPI) Campus Picos, distribuídos entre os cinco períodos letivos do curso. O instrumento de coleta de dados consistiu em um formulário de avaliação estruturado com 16 perguntas, mesclando métricas quantitativas em escala de 1 a 10 e campos qualitativos de texto livre. A amostragem abrangeu uma

frota heterogênea de *smartphones* móveis (com predominância de modelos de entrada e intermediários), permitindo avaliar o comportamento do aplicativo sob diferentes capacidades de processamento.

### 5.2.2 Resultados Quantitativos

Conforme os dados apurados, a interface do aplicativo apresentou elevada intuitividade (9,57) e clareza visual (9,65), enquanto a combinação de efeitos sonoros e visuais proporcionou maior imersão nas partidas. Mesmo diante de uma interface reativa enriquecida por animações, o jogo demonstrou alta estabilidade técnica, registrando médias baixíssimas para lentidões (2,48), travamentos (2,35) e *bugs* de lógica (2,17). Esses indicadores comprovam experimentalmente o atendimento às diretrizes de qualidade do RNF02 (Resiliência da interface de usuário) e do RNF03 (Otimização gráfica em dispositivos de baixo desempenho).

Sob a perspectiva do conteúdo, as dicas das cartas, sintetizadas pelo modelo de linguagem da *GroqCloud*, foram avaliadas como altamente coerentes e úteis (8,30), validando as técnicas de engenharia de *prompt* do Objetivo Específico 4 e a geração via API do RF01. Adicionalmente, a injeção probabilística de dicas falsas (*trick hints*) obteve nota 8,87, provando que a desestabilização lúdica prevista no RF04 atua como um elemento chave de engajamento e diversão.

Com o objetivo de sintetizar os dados quantitativos obtidos no questionário de avaliação aplicado aos 23 participantes, a Tabela 5 consolida os indicadores organizados por categorias de usabilidade, desempenho técnico, qualidade da inteligência artificial e satisfação geral.

Tabela 5 – Consolidação das métricas quantitativas obtidas na avaliação ($N = 23$).

| **Categoria** | **Métrica / Indicador** | **Média** | **Mediana** | **Desvio Padrão** |
|---|---|---|---|---|
| Usabilidade | Intuitividade da Interface | 9,57 | 10,0 | 0,66 |
| | Facilidade de Entendimento | 9,65 | 10,0 | 0,65 |
| | Efeitos Visuais e Fluidez | 9,74 | 10,0 | 0,69 |
| | Efeitos Sonoros e Imersão | 8,91 | 10,0 | 2,07 |
| Desempenho* | Ausência de Lentidão/Gargalos | 2,48 | 1,0 | 3,06 |
| | Ausência de Travamentos/Crashes | 2,35 | 1,0 | 2,84 |
| | Ausência de Bugs de Lógica | 2,17 | 1,0 | 2,69 |
| Mecânica e IA | Coerência das Dicas da IA | 8,30 | 9,0 | 1,66 |
| | Divertimento das Pegadinhas | 8,87 | 9,0 | 1,42 |
| | Ritmo e Dinâmica de Partida | 8,70 | 10,0 | 2,05 |
| Satisfação | Probabilidade de Recomendação | 9,17 | 10,0 | 1,95 |
| | Potencial de Re-jogabilidade | 8,83 | 10,0 | 1,61 |

Fonte: Elaborado pelo autor (2026).

**Notas técnicas de desempenho variam de 1 (excelente/sem falhas) a 10 (muitas falhas).*

### 5.2.3 Discussão

Com o propósito de avaliar experimentalmente a hipótese do esgotamento de acervos textuais em jogos de trívia, conduziu-se um Teste A/B entre as turmas participantes. As turmas de ADS I, III e V receberam a Versão A do aplicativo, caracterizada por um acervo dinâmico expandido (70+ cartas com injeção assíncrona, viabilizada pelo RNF01). Em contrapartida, as turmas de ADS II e IV utilizaram a Versão B, composta por um lote estático e limitado de 15 cartas, sem comunicação com a esteira de geração em nuvem.

Para analisar o impacto dessa variação de volume na percepção de longevidade do jogo, a Tabela 6 apresenta o comparativo das médias, medianas e desvios padrão obtidos para as métricas de exaustão e repetição de conteúdo entre os dois grupos de teste.

Tabela 6 – Comparativo da percepção de exaustão e repetição de conteúdo no Teste A/B.

| **Métrica Analisada** | **Estatística** | **Versão A** ($n = 13$) | **Versão B** ($n = 10$) |
|---|---|---|---|
| Percepção de que o conteúdo acabou rápido | Média / Mediana<br>Desvio Padrão | 7,23 / 8,0<br>3,03 | 7,40 / 7,5<br>2,76 |
| Percepção de que o jogo ficou repetitivo rápido | Média / Mediana<br>Desvio Padrão | 9,23 / 10,0<br>1,59 | 7,80 / 9,5<br>3,01 |

Fonte: Elaborado pelo autor (2026).

A análise dos dados expostos na Tabela 6 revela que a percepção de que o conteúdo "acabou rápido"permaneceu praticamente equivalente entre a Versão A (7,23) e a Versão B (7,40). Esse achado corrobora empiricamente as reflexões de (Schell, 2008) e (Koster, 2004) detalhadas na Seção 2, demonstrando que jogos baseados em trívia possuem uma velocidade de consumo de dados extremamente acelerada. Dessa forma, comprova-se que o simples aumento pontual no volume estático de cartas não resolve o gargalo de longevidade do *software*, pois o acervo é rapidamente superado pela dinâmica das partidas em grupo.

Outro fenômeno relevante observado diz respeito ao indicador de repetição, no qual a Versão A registrou uma média superior (9,23) em comparação à Versão B (7,80). Essa variação decorre do fato de que os jogadores da Versão A, ao realizarem sessões mais longas de jogo sustentadas pelo banco maior, passaram a reconhecer os padrões sintáticos e estruturais de redação inerentes aos *prompts* do modelo de linguagem. Esse resultado evidencia que o fornecimento de texto sintético não deve se limitar à quantidade de dados, exigindo rotinas contínuas de checagem na origem para mitigar redundâncias e aplicar variação contextual.

Nesse cenário, evidencia-se a importância de integrar diretrizes consolidadas de *Game Design* ao processo de geração. O texto bruto fornecido pelo LLM não

constitui, por si só, um jogo; para que o usuário se mantenha no Canal do *Flow* (Csikszentmihalyi, 1990), a informação precisa ser convertida em um desafio estruturado. No *QuestAnswer*, essa transformação é viabilizada pelo ecossistema de regras, como o papel do Mestre, a pontuação regressiva temporizada e a injeção probabilística de pistas ambíguas (*trick hints*), demonstrando que o utilitário CLI de geração via PCGML e a arquitetura distribuída desacoplada são indispensáveis para alimentar a esteira de suprimentos do jogo de forma sustentável e escalável.

# 6 CONCLUSÃO

O presente Trabalho de Conclusão de Curso detalhou o projeto, a implementação e a validação do ecossistema *QuestAnswer*, uma plataforma móvel de trívia desenvolvida para demonstrar a viabilidade da Geração Procedural de Conteúdo via Aprendizado de Máquina (PCGML) como solução para o rápido esgotamento de acervos textuais em jogos casuais.

A concepção do projeto tem suas raízes em uma motivação pessoal iniciada no ensino médio, período em que a escassez finita de cartas de jogos de mesa tradicionais interrompia a continuidade do entretenimento entre amigos. Todavia, a transformação dessa ideia, até então era um sonho abstrato.

Tornar essa ideia em um produto de engenharia funcional e distribuído só se tornou exequível a partir do amadurecimento técnico e teórico adquirido ao longo do curso de Análise e Desenvolvimento de Sistemas, com destaque para a consolidação de conhecimentos em arquitetura de *backend* com .NET/C#, bancos de dados relacionais e desenvolvimento de jogos na Godot Engine.

Todos os objetivos propostos para o trabalho foram plenamente alcançados. O Objetivo Geral foi atingido mediante a entrega do ecossistema completo em nuvem e sua validação prática. De forma específica:

1. O levantamento, mapeamento e documentação das restrições do sistema foram consolidados através dos Requisitos Funcionais (RF01 a RF06) e Não Funcionais (RNF01 a RNF05);

2. A arquitetura distribuída foi modelada e implantada de forma desacoplada, delimitando com clareza as fronteiras entre o cliente móvel (Godot), o servidor de aplicação (.NET/Render), a persistência (PostgreSQL/Supabase) e a esteira de ingestão via CLI (Node.js/GroqCloud);

3. A interface gráfica móvel foi construída com componentes reativos, animações fluidas e emissores de partículas nativos (*CPUParticles2D*);

4. A esteira de geração procedural acoplou com sucesso técnicas de engenharia de *prompt* para produzir arquivos JSON estruturados, aplicando rotinas probabilísticas de desestabilização lúdica (*trick hints*);

5. Os testes funcionais de integração e a avaliação de uso com 23 usuários confirmaram a resiliência dos canais de comunicação e a estabilidade da aplicação.

Sob a perspectiva dos resultados práticos e sociais, o jogo ultrapassou o escopo estrito dos testes laboratoriais. Observou-se a adesão orgânica da aplicação entre

os participantes, que continuaram utilizando o *QuestAnswer* em rodas de conversa e encontros sociais informais mesmo após o término da coleta de dados. Do ponto de vista acadêmico e metodológico, o desenvolvimento trouxe aprendizados valiosos sobre a importância da organização estruturada de projetos e evidenciou que os desafios da criação de jogos transcendem as barreiras puramente de programação, adentrando o campo do *Game Design*, área fundamental para converter dados brutos em experiências verdadeiramente desafiadoras e envolventes.

Com base nas sugestões registradas pelos participantes no formulário de avaliação, identificam-se oportunidades de melhorias futuras de curto e médio prazo. A principal delas reside na expansão e diversificação dos modelos de *prompts* e categorias, introduzindo dinâmicas de geração capazes de criar estruturas frasais heterogêneas e mitigar a percepção de repetição sintática em sessões prolongadas de jogo.

Por fim, aponta-se um horizonte promissor para trabalhos futuros no campo da inteligência artificial aplicada aos jogos digitais. Propõe-se, prioritariamente:

- **Expansão da Amostragem Empírica:** Uma vez que a amostra de 23 usuários, embora aponte tendências que corroboram a literatura, é estatisticamente limitada, sugere-se a aplicação de testes em larga escala para consolidar as métricas de usabilidade e retenção.
- **Otimização da Escala de Prompts:** Atualmente, a injeção de todos os títulos existentes no prompt mitiga duplicidades (RF03), porém não é uma solução infinitamente escalável devido ao limite de tokens do LLM. Estudos futuros devem implementar técnicas de busca vetorial (Embeddings/RAG) para contornar duplicidades semânticas sem sobrecarregar o prompt.
- **Arquitetura de Validação Factual (Multi-Agente):** Como o mecanismo atual avalia apenas o formato e o título, trabalhos futuros devem implementar um segundo agente de IA atuando como auditor para checar a veracidade factual e gramatical das dicas antes de sua persistência.
- **Publicação do Aplicativo:** Adequar as chaves de API, o modelo de financiamento na nuvem e os termos de uso para a viabilização da publicação oficial do aplicativo na Google Play Store.
- **Estudo de Impacto de Diretrizes de *Game Design*:** Conduzir um novo Teste A/B que não restrinja o volume de cartas, mas compare diretamente a experiência do usuário em duas versões do sistema: uma que apenas exiba o conteúdo gerado de forma crua e outra equipada com o envelope lúdico completo, isolando cientificamente a contribuição do *Game Design* na manutenção do jogador no Canal do *Flow*.

## DECLARAÇÃO DE USO DE INTELIGÊNCIA ARTIFICIAL GENERATIVA

Durante a elaboração deste trabalho, foram utilizadas ferramentas de inteligência artificial generativa para apoio em tarefas específicas. O modelo **Google Gemini 3.6 Flash Estendido** foi utilizado para revisão gramatical e de estilo do texto, bem como para o auxílio na organização de referências bibliográficas e na discussão de conceitos técnicos durante a fase de pesquisa. Todo o conteúdo técnico, as análises, os resultados e as conclusões apresentados são de responsabilidade do autor, que revisou e validou integralmente as contribuições geradas com o apoio dessa ferramenta.

# REFERÊNCIAS

ADAMS, E. **Fundamentals of Game Design**. [*S. l.*]: New Riders, 2014. (Always learning Pearson). ISBN 9780321929679. Disponível em: https://books.google.com.br/books?id=L6pKAgAAQBAJ.

ARAÚJO, Wendell Oliveira de; ARANHA, Eduardo Henrique da Silva. Geração Procedural de Conteúdo para Criação de Fases de Jogos Educativos usando Gramática. *In:* ANAIS do XXIX Simpósio Brasileiro de Informática na Educação (SBIE). [*S. l.*]: SBC, 2018. p. 725–734. DOI: 10.5753/cbie.sbie.2018.725.

BIEHL, Matthias. **RESTful API design**. [*S. l.*]: API-University Press, 2016. v. 3.

COSTA, André Febeliano da. **Avaliação de experiência de jogador aplicada ao desenvolvimento de jogos**. 2016. Dissertação (Mestrado em Ciências) – Escola Politécnica da Universidade de São Paulo, São Paulo. Área de Concentração: Engenharia de Computação. Orientador: Ricardo Nakamura. Disponível em: https://www.teses.usp.br/teses/disponiveis/3/3141/tde-29082016-134726/publico/AndreFebelianodaCostaCorr16.pdf.

CSIKSZENTMIHALYI, Mihaly. Flow: The Psychology of Optimal Experience*. In:* [*s. l.: s. n.*], jan. 1990.

FARROKHI MALEKI, Mahdi; ZHAO, Richard. Procedural Content Generation in Games: A Survey with Insights on Emerging LLM Integration. **Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment**, Association for the Advancement of Artificial Intelligence (AAAI), v. 20, n. 1, p. 167–178, nov. 2024. ISSN 2326-909X. DOI: 10.1609/aiide.v20i1.31877. Disponível em: http://dx.doi.org/10.1609/aiide.v20i1.31877.

FIELDING, Roy Thomas. **Architectural Styles and the Design of Network-based Software Architectures**. 2000. Tese (Doutorado) – University of California, Irvine, Irvine, CA, USA. Dissertation Committee: Richard N. Taylor (Chair), Mark S. Ackerman, David S. Rosenblum.

FORTIM, Ivelise *et al.* Pesquisa indústria brasileira de games 2022. **TIC CULTURA**, v. 113, 2022.

GARRETT, J.J. **The Elements of User Experience: User-Centered Design for the Web and Beyond**. [*S. l.*]: Pearson Education, 2010. (Voices That Matter). ISBN 9780321624642. Disponível em: https://books.google.com.br/books?id=9QC6r5OzCpUC.

GONZÁLEZ-SÁNCHEZ, José; PADILLA-ZEA, Natalia; VELA, Francisco Luis. From Usability to Playability: Introduction to Player-Centred Video Game Development

Process. *In:* v. 5619, p. 65–74. ISBN 978-3-642-02805-2. DOI: 10.1007/978-3-642-02806-9_9.

HURST, William *et al.* Game engines for sustainable open science software: a case study. **Entertainment Computing**, Elsevier, p. 101123, 2026.

INUWA-DUTSE, Isa. OpenAI's GPT-OSS-20B Model and Safety Alignment Issues in a Low-Resource Language. **arXiv preprint arXiv:2510.01266**, 2025.

JUUL, Jesper. **Half-real: Video games between real rules and fictional worlds**. Cambridge, Mass: MIT Press, 2005. ISBN 9780262516518.

KNIJNIK, David Mees. Comparação de SGBDs mongoDB e postgreSQL para jogos digitais, 2022.

KORN, Oliver *et al.* Procedural Content Generation for Game Props? A Study on the Effects on User Experience. **Computers in Entertainment**, v. 15, p. 1–15, abr. 2017. DOI: 10.1145/2974026.

KOSTER, Raph. **A Theory Of Fun**. [*S. l.*]: O'Reilly Media, 2004.

LEITE, GDOB; LIMA, Edirlei Soares de. Geração procedural de mapas para jogos 2d. **SBGames**, 2012.

LIBERTY, Jesse. **Programming C#: building. NET applications with C**. [*S. l.*]: "O'Reilly Media, Inc.", 2005.

LOCK, Andrew. **ASP. NET core in Action**. [*S. l.*]: Simon e Schuster, 2023.

MCGATHEY, Ian L. Search-Based Procedural Content Generation in Games. *In:* disponível em: https://api.semanticscholar.org/CorpusID:229165731.

OLIVEIRA, Gonçalo *et al.* A Rule Based Procedural Content Generation System. *In:* [*s. l.: s. n.*], jan. 2023. p. 107–113. ISBN 978-3-031-23235-0. DOI: 10.1007/978-3-031-23236-7_8.

PEREIRA, Ricardo Timarán; TORRES, Anívar Chaves. Integración de técnicas supervisadas de aprendizaje automático con el SGBD POSTGRESQL en una arquitectura medianamente acoplada. **Encuentro Internacional de Educación en Ingeniería**, 2023.

POGLIA, Pedro F; BAVARESCO, Me Jorge Luis Boeira. Benchmark de Frameworks Node. js para Desenvolvimento de APIs: Avaliação de Desempenho com Express, NestJS, Fastify e Next. js, 2025.

PRESSMAN, R.S.; MAXIM, B. **Engenharia De Software: UMA ABORDAGEM PROFISSIONAL**. [*S. l.*]: MCGRAW HILL - ARTMED, 2016. ISBN 9788580555332. Disponível em: https://books.google.com.br/books?id=smNFvgAACAAJ.

SALEN, Katie; ZIMMERMAN, Eric. **Rules of Play - Volume 3**. [*S. l.*]: Blucher, 2012.

SANTOS, Renan de Souza dos. Geração procedural de puzzles para o desenvolvimento de jogos. Universidade Federal do Rio Grande do Norte, 2022.

SCHELL, J. **The Art of Game Design: A book of lenses**. [*S. l.*]: CRC Press, 2008. ISBN 9780080919171. Disponível em: https://books.google.com.br/books?id=ZHfMBQAAQBAJ.

SMITH, Gillian. Understanding procedural content generation: A design-centric analysis of the role of PCG in games. **Conference on Human Factors in Computing Systems - Proceedings**, abr. 2014. DOI: 10.1145/2556288.2557341.

SPRINGER, Sebastian *et al.* **Node. js: the comprehensive guide**. [*S. l.*]: Packt Publishing Ltd, 2025.

SUMMERVILLE, Adam *et al.* Procedural Content Generation via Machine Learning (PCGML). **IEEE Transactions on Games**, PP, fev. 2017. DOI: 10.1109/TG.2018.2846639.

SWEETSER, Penelope; WYETH, Peta. GameFlow: A Model for Evaluating Player Enjoyment in Games. **Computers in Entertainment**, v. 3, p. 3, jul. 2005. DOI: 10.1145/1077246.1077253.

THORN, Alan. **Game engine design and implementation**. [*S. l.*]: Jones & Bartlett Learning, 2011.

TO, Minh. Zero-trust networks for LLM inference in edge-cloud computing, 2025.

TOGELIUS, Julian; KASTBJERG, Emil *et al.* What is Procedural Content Generation? Mario on the borderline, jun. 2011. DOI: 10.1145/2000919.2000922.

TOGELIUS, Julian; YANNAKAKIS, Georgios *et al.* Search-Based Procedural Content Generation: A Taxonomy and Survey. **Computational Intelligence and AI in Games, IEEE Transactions on**, v. 3, p. 172–186, out. 2011. DOI: 10.1109/TCIAIG.2011.2148116.

WANG, Ken. Cause and Solution of Object-Relational Impedance Mismatch, 2024.

YAGANTI, Dheerendra. Performance-Driven Design of Scalable Web APIs Using. Net Core 3.1, Entity Framework Core, and SQL Server on Azure App Services, 2020.

# APÊNDICE A – FORMULÁRIO DE AVALIAÇÃO DA EXPERIÊNCIA DO USUÁRIO

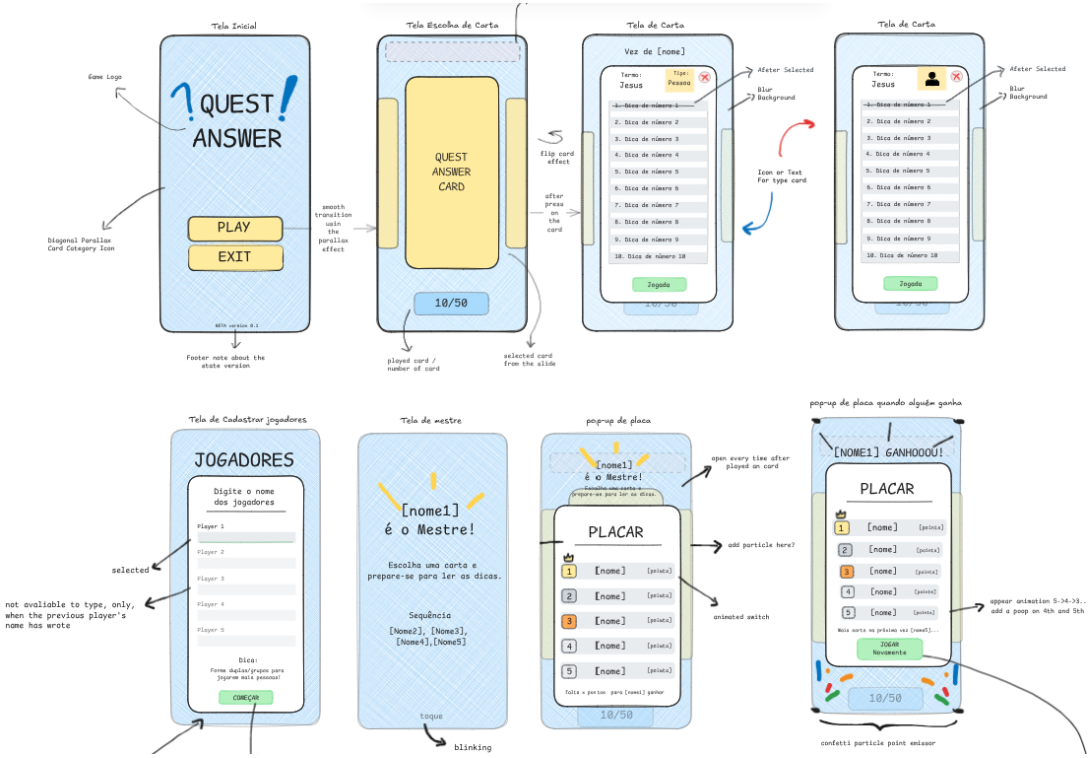
Este apêndice apresenta o instrumento utilizado para a coleta de dados de usabilidade e recepção lúdica com os participantes durante os testes do ecossistema *QuestAnswer*. O formulário completo e as respostas obtidas podem ser consultados por meio do seguinte endereço eletrônico:

- Link de Acesso ao Formulário:
  `https://forms.gle/vFgexDwP2SaNjASU9`

## APÊNDICE B – ESBOÇOS INICIAIS DE INTERFACE

Este apêndice documenta os esboços manuais de baixa fidelidade, esboços produzidos durante as sessões iniciais de concepção do *QuestAnswer*. A Figura 5 ilustra o planejamento inicial para a disposição dos botões e contadores da interface principal.

Figura 5 – Esboço conceitual da interface de controle do jogo.

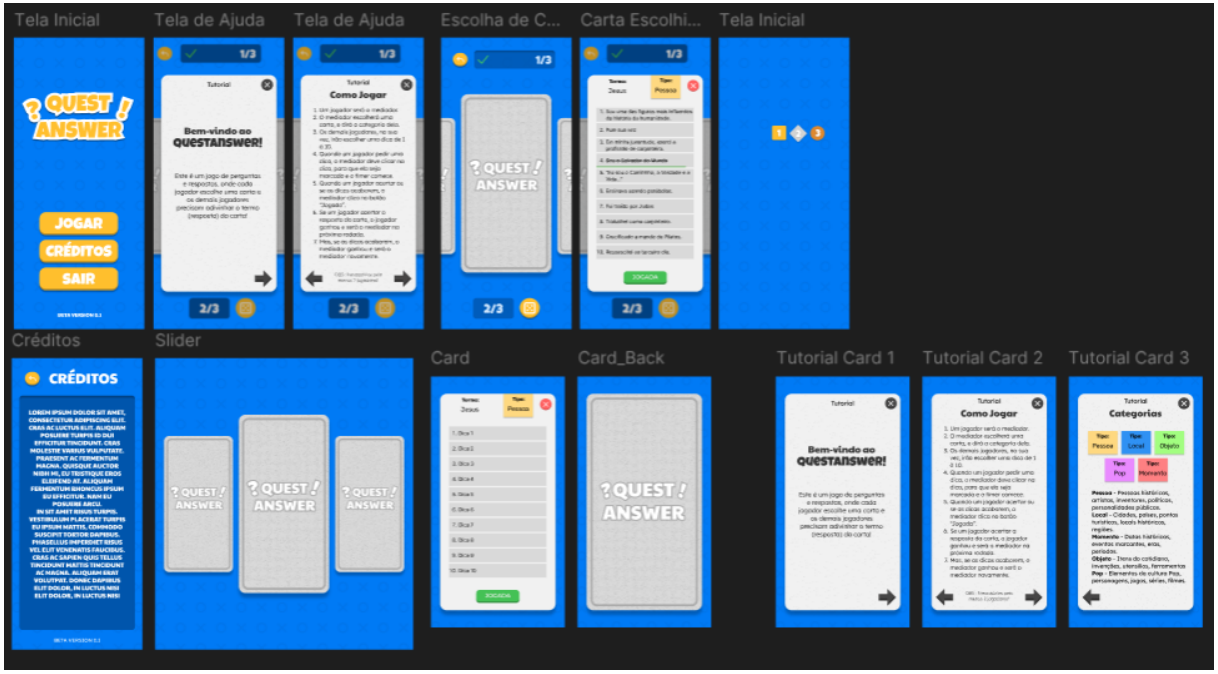



Fonte: Elaborado pelo autor.

## APÊNDICE C – PROTÓTIPOS DE ALTA FIDELIDADE (FIGMA)

A Figura 6 ilustra o mapeamento do fluxo de navegação e a paleta de cores concebida no software de prototipagem Figma. O objetivo deste artefato não é a leitura do texto das telas, mas sim demonstrar a evolução visual do projeto desde o rascunho de baixa fidelidade B até a estruturação moderna, reativa e baseada em componentes (UI/UX) que foi posteriormente implementada no cliente final da *Godot Engine*.

Figura 6 – Protótipo de alta fidelidade da interface de controle do jogo.




Fonte: Elaborado pelo autor.

# APÊNDICE D – ESTRUTURA DO PROMPT DE GERAÇÃO

Este apêndice apresenta o modelo base de instrução (*prompt*) enviado à API da GroqCloud para a geração de uma nova carta. Os campos entre chaves indicam as variáveis preenchidas dinamicamente pela ferramenta de linha de comando (CLI) antes do disparo da requisição:

```
{
  "gerar-carta": "Você é um gerador de cartas de um jogo de
  perguntas e respostas chamado QuestAnswer. Sua tarefa é criar
  uma carta com as seguintes características:

  A carta deve ter um termo oculto em português brasileiro
  (ou a versão mais famosa) (answer) que o jogador deve adivinhar.
  A carta deve pertencer à categoria {Categoria_Selecionada}.
  A carta deve conter 10 dicas que ajudem o jogador a descobrir
  o termo.
  Não use o termo da resposta nas dicas, nem variações próximas.
  As dicas devem ser curtas, claras, informativas, e variadas
  entre si (evite repetições ou ambiguidades).
  Evite dicas com datas muito específicas ou linguagem
  excessivamente técnica.
  A carta deve ser interessante para um público geral, casual.

  Responda em JSON no seguinte formato (sem explicações extras):
  {
    \"answer\": \"Albert Einstein\",
    \"category\": \"Pessoa\",
    \"tips\": [
      \"Foi um grande cientista.\",
      \"Revolucionou a física.\",
      \"É conhecido pela Teoria da Relatividade.\",
      \"Recebeu um Prêmio Nobel.\",
      \"Viveu parte da vida nos Estados Unidos.\",
      \"Ficou famoso por seus cabelos bagunçados.\",
      \"Nasceu na Alemanha.\",
      \"Inspirou debates filosóficos.\",
      \"Seu trabalho impactou a ciência moderna.\",
```

```
      \"É um ícone da genialidade.\"
    ]
  }

  Não gere essas cartas: [{Lista_de_Termos_Existentes}]"
}
```

# APÊNDICE E – CÓDIGO-FONTE E REPOSITÓRIOS DO ECOSSISTEMA

Com o propósito de assegurar a transparência, a auditabilidade e o caráter *open-source* da pesquisa, os códigos-fonte desenvolvidos para os três módulos que integram o ecossistema *QuestAnswer* estão disponibilizados nos seguintes repositórios públicos na plataforma GitHub:

- Módulo Cliente Móvel:

  `https://github.com/Juawo/questanswer-client`

- Módulo Web API e Backend:

  `https://github.com/Juawo/questanswer-api`

- Módulo Gerador de Cartas:

  `https://github.com/Juawo/questanswer-cardgen`