AniCard Battle: Arquitetura Mista em Flutter para o Desenvolvimento de um Jogo Mobile Educativo

Lucas Gabriel Teixeira do Nascimento[1], Andrey Rodrigues Moreira[2], Ana Paula Abrantes de Castro e Shiguemori[2]

Instituto Federal de Educação, Ciência e Tecnologia de São Paulo (IFSP) (teixeira.nascimento@aluno.ifsp.edu.br)[1]
Instituto Federal de Educação, Ciência e Tecnologia de São Paulo (IFSP) , Jacareí, Brasil[2]

Resumo: O AniCard é um jogo mobile, desenvolvido em Flutter, inspirado no Super Trunfo para ensino de zoologia. O projeto tem arquitetura mista, usando Firebase Firestore para os dados dinâmicos do usuário e JSON local para os atributos das cartas de animais. Ao integrar engenharia de software, gamificação e os conteúdos da biodiversidade, o app atua como ferramenta pedagógica lúdica de apoio. A abordagem adotada otimiza o desempenho geral da aplicação, reduz leituras no backend e amplia escalabilidade.

Palavras-chave: Jogos Educativos; Zoologia; Flutter; Firebase; Engenharia de Software

INTRODUÇÃO

O uso de metodologias ativas e da gamificação no ambiente educacional tem se mostrado uma estratégia relevante para favorecer o engajamento dos estudantes, especialmente no ensino de Ciências e Biologia. A literatura acadêmica recente destaca que a gamificação pode contribuir para despertar o interesse dos alunos, estimular a curiosidade e favorecer uma aprendizagem mais significativa ao abordar temas como o Reino Animalia de forma dinâmica, lúdica e acessível (Santos, 2021). Em contextos de ensino nos quais o envolvimento do estudante precisa ser estimulado, a aplicação de jogos pode aproximar o aluno do conteúdo, transformando a aula em uma experiência mais investigativa e participativa (Revista Mirante, 2022). Nesse cenário, jogos de cartas digitais que resgatam mecânicas de comparação de atributos, como as do clássico “Super Trunfo”, apresentam potencial pedagógico para apoiar o ensino de zoologia e biodiversidade.

Para materializar essas mecânicas de forma fluida e ampliar o acesso dos estudantes por meio de dispositivos móveis, o desenvolvimento de jogos educativos tem se apoiado em frameworks modernos, como o Flutter (Moura, 2025. Aplicativos construídos com essa tecnologia podem alcançar bom desempenho gráfico devido à sua arquitetura, que utiliza uma base central escrita em C/C++ para otimizar a renderização (Nanavati et al., 2024). No ecossistema Flutter, a interface do usuário é construída por meio de widgets, o que permite traduzir o estado mutável da partida diretamente para os elementos visuais de forma reativa (Nanavati et al., 2024), favorecendo a atualização dinâmica das telas durante a interação do usuário com o jogo.

Apesar do apelo visual e pedagógico, o desenvolvimento de um jogo de cartas digital educativo impõe desafios arquiteturais importantes. É necessário coordenar um ciclo contínuo de ações de jogo, como embaralhar cartas, distribuir mãos e processar a lógica de comparação dos atributos dos animais, sem comprometer o desempenho da aplicação. A interface precisa ser responsiva, exigindo gerenciamento de estado adequado e separação clara de responsabilidades no código. Além disso, o jogo precisa integrar essas regras a um sistema de dados eficiente, capaz de gerenciar tanto o progresso dinâmico do aluno, como coleções de cartas e moedas, quanto os atributos estáticos e estruturais de cada espécie.

Nesse contexto, foi desenvolvido o AniCard, um jogo de cartas móvel educativo, implementado em Flutter, com o propósito de apoiar o ensino de zoologia. O projeto estrutura sua base de código separando a interface responsiva, às lógicas de turno e os serviços de dados, criando um núcleo de funcionamento do jogo encapsulado e modular.

O objetivo deste trabalho é relatar o desenvolvimento do jogo educativo AniCard, com ênfase na construção de sua arquitetura mista e na

organização das principais funcionalidades da partida. O artigo apresenta a implementação da lógica de turnos e do gerenciamento de estado durante o jogo, utilizando princípios de Arquitetura Limpa para separar a interface visual da camada de dados. Essa camada é composta pelo Firestore, responsável pelo armazenamento do progresso do aluno, e por um arquivo JSON local, utilizado para armazenar os atributos dos animais. Essa abordagem busca melhorar o desempenho da aplicação, reduzir a dependência de leituras constantes no backend e tornar a ferramenta mais adequada ao contexto educacional.

MATERIAL E MÉTODOS

A metodologia adotada no desenvolvimento do AniCard foi organizada em três etapas principais: definição da arquitetura do aplicativo, implementação da lógica de turnos e gerenciamento de estado, e estruturação do fluxo de dados misto. Essa organização permitiu separar os aspectos relacionados à interface, às regras da partida e ao armazenamento das informações utilizadas pelo jogo.

1. Arquitetura do Aplicativo e Divisão em Camadas

O desenvolvimento da estrutura principal de funcionamento do AniCard foi realizado utilizando o framework multiplataforma Flutter e a linguagem Dart, aproveitando a organização hierárquica de widgets para a construção de uma interface reativa. No ecossistema Flutter, a atualização da interface ocorre a partir da reconstrução de componentes visuais em resposta às alterações de estado da aplicação. Esse funcionamento está associado ao princípio de que a interface visual pode ser compreendida como uma função do estado atual do sistema, conforme representado na Equação (Nanavati et al., 2024).

$$UI = f(state) \qquad (1)$$

Para assegurar a manutenibilidade, a legibilidade e a escalabilidade do código da tela de batalha, foi adotada uma abordagem baseada em princípios de Arquitetura Limpa. O sistema do jogo foi segmentado em quatro camadas de software distintas:

1. Interface principal: atua como ponto de entrada da tela de combate, controlando o ciclo de vida da partida e coordenando a disposição do layout responsivo.
2. Componentes visuais: correspondem ao conjunto de widgets responsáveis por renderizar elementos específicos da aplicação, como cartas dos animais, imagens, fichas técnicas, indicadores de pontuação e área central de confronto.
3. Funções auxiliares: representam a camada de lógica responsável pelo processamento das regras do jogo, incluindo validação, sorteio e comparação dos atributos zoológicos entre duas cartas.
4. Serviços de banco de dados: correspondem ao módulo responsável por encapsular o acesso aos dados locais e às chamadas de rede, separando a lógica do jogo dos detalhes de infraestrutura.

O fluxo de navegação geral e o acesso às funcionalidades pedagógicas do jogo foram concentrados em uma tela inicial, projetada para guiar o usuário de forma simples e intuitiva pelas principais divisões do sistema, conforme apresentado na Figura 1.

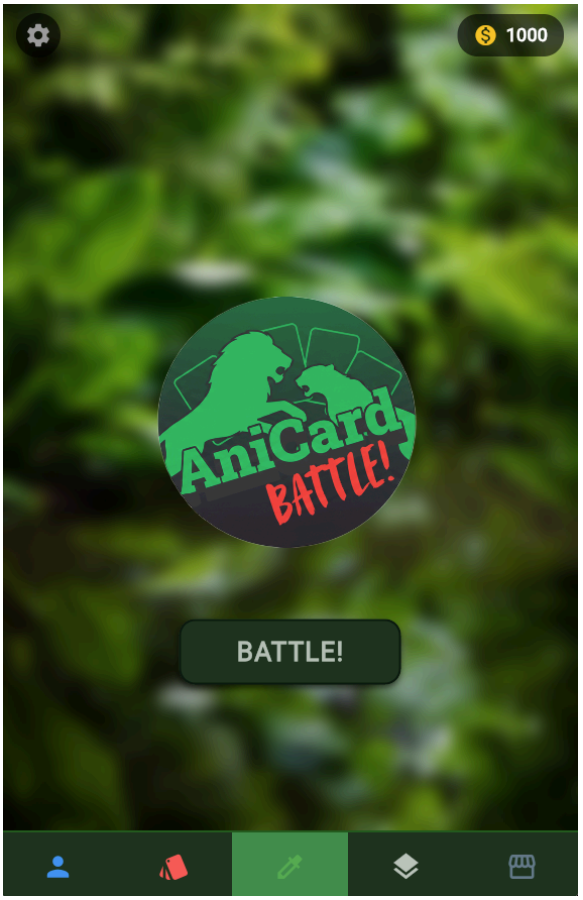


Figura 1. Interface estrutural do menu principal do aplicativo AniCard.
Fonte: Elaborado pelos autores (2026).

2. Lógica de Turnos e Gerenciamento de Estado

A lógica da partida precisa responder a eventos dinâmicos contínuos para manter o envolvimento do estudante durante o jogo. Para evitar a reconstrução desnecessária de toda a árvore de *widgets* e otimizar o tempo de renderização da interface, o gerenciamento de estado foi implementado com o uso de componentes do tipo *StatefulWidget* (Nanavati et al., 2024). Essa abordagem permite atualizar seletivamente apenas as partes da interface

modificadas após uma ação do jogador, contribuindo para a fluidez da experiência interativa.

O ciclo de vida da partida educativa foi organizado em etapas que envolvem o embaralhamento e a distribuição das cartas, o sorteio dos atributos biológicos, a comparação matemática entre os valores das cartas e a definição das ações de pontuação ou descarte, conforme ilustrado na Figura 2.

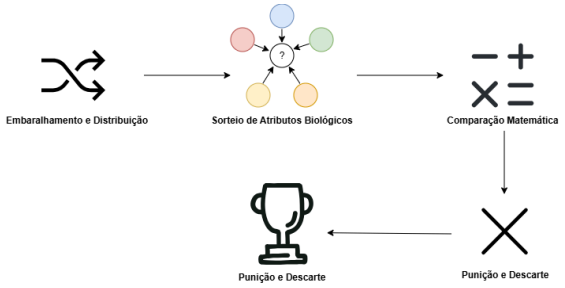


Figura 2. Fluxograma do ciclo de vida da partida AniCard.
Fonte: Elaborado pelos autores (2026).

3. Integração e Fluxo de Dados Misto

Para suportar as mecânicas educativas sem adicionar latência ao ciclo de aprendizagem, a camada de serviços utiliza uma lógica híbrida de dados. Os dados dinâmicos do aluno, como inventário de cartas desbloqueadas, decks equipados e progresso no jogo, são obtidos de forma assíncrona a partir do Firebase Firestore.

Por outro lado, para reduzir a dependência de leituras constantes em nuvem durante as partidas, o conteúdo didático, composto pelos atributos biológicos estáticos e pelos caminhos das imagens de cada espécie do Reino Animalia, é armazenado localmente em um arquivo JSON estruturado. Os identificadores do Firestore são relacionados aos dados presentes nesse arquivo local por meio do *rootBundle* do Flutter, permitindo que o jogo apresente os dados dos animais de forma rápida durante a batalha, sem depender de múltiplas requisições ao servidor.

A visualização das cartas obtidas e gerenciadas pelo usuário é concentrada em um painel específico de coleção, conforme apresentado na Figura 3.

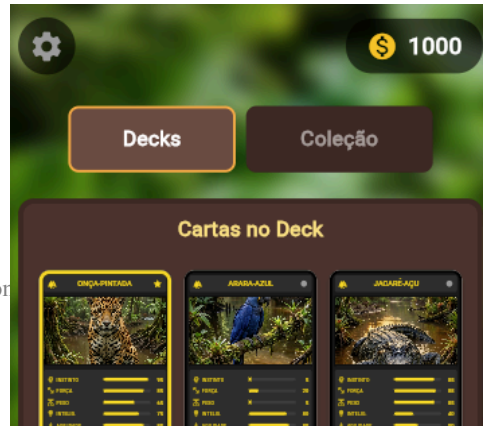


Figura 3.Interface da tela de Coleção de cartas zoológicas e inventário do jogador.
Fonte: Elaborado pelos autores (2026)

RESULTADOS E DISCUSSÃO

A implementação da tela de batalha do AniCard indicou a viabilidade da arquitetura proposta para a construção de uma ferramenta educativa interativa baseada em gamificação. O emprego de princípios de Arquitetura Limpa favoreceu a separação de responsabilidades no código da aplicação, permitindo distinguir os elementos de interface, os componentes visuais, as funções auxiliares e os serviços de dados. A camada de apresentação ficou responsável pela organização do layout responsivo em modo vertical e pela captura das interações do aluno, conforme observado na interface ativa da tela de batalha, apresentada na Figura 4.

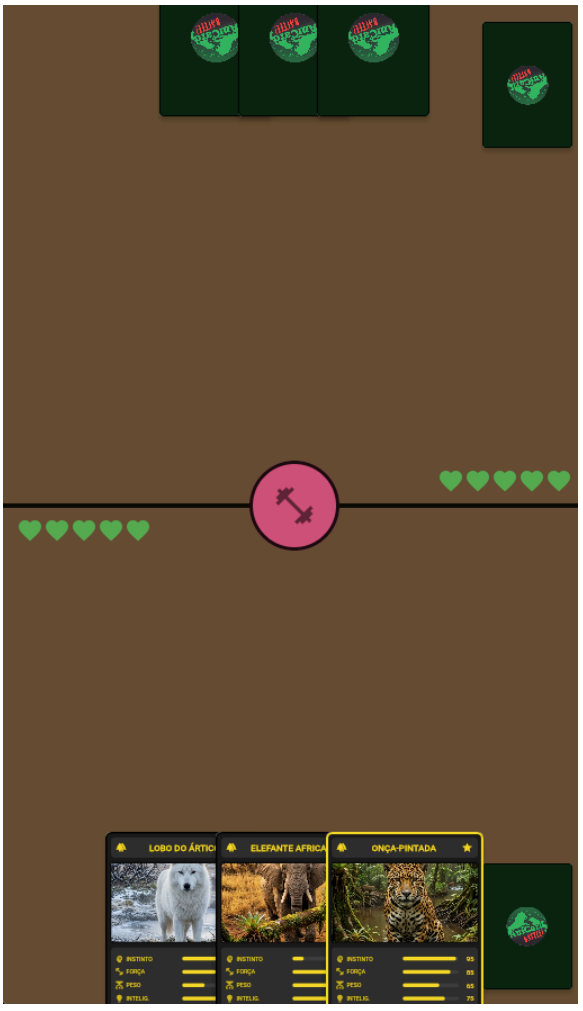


Figura 4. Interface ativa da tela de batalha e comparação de atributos no AniCard.
Fonte: Elaborado pelos autores (2026)

As lógicas de turno e de validação dos atributos zoológicos foram concentradas nas funções auxiliares, o que contribuiu para uma organização mais clara do código e para a possibilidade de expansão futura do jogo. Essa modularidade permite que novas informações biológicas, como taxonomia, habitat ou estado de conservação das espécies, sejam incorporadas ao sistema sem a necessidade de reformular toda a interface.

Em relação ao gerenciamento de estado, a estratégia de encapsular partes específicas da interface utilizando StatefulWidget mostrou-se adequada para a atualização dinâmica dos elementos visuais durante a partida. A literatura técnica indica que a reconstrução desnecessária da árvore completa de widgets no Flutter pode prejudicar o tempo de renderização e de construção da interface, o que, em um contexto educativo, pode reduzir a fluidez da interação do estudante com o jogo (Nanavati et al., 2024). Ao aplicar atualizações de estado localizadas, como a alteração da barra de vida após uma rodada ou a exibição de um novo atributo biológico sorteado, o sistema evitou o recarregamento completo da tela a cada turno, contribuindo para uma experiência de uso mais fluida (Nanavati et al., 2024).

Por fim, o modelo híbrido de integração de dados mostrou-se adequado para reduzir a dependência de leituras constantes em nuvem e organizar os dados utilizados durante a partida. Em arquiteturas dependentes de plataformas Backend as a Service (BaaS), a precificação pode estar relacionada à quantidade de documentos lidos e transferidos (Airbyte, 2025). Caso a aplicação realizasse requisições em nuvem para obter atributos biológicos, textos descritivos e imagens a cada nova carta retirada do deck, o volume de leituras poderia elevar os custos operacionais em um projeto educativo em expansão (Firebase, 2026). A abordagem mista reduz esse problema ao utilizar o Firestore principalmente para sincronizar o progresso do aluno, enquanto os dados científicos das cartas são consultados localmente em um arquivo JSON incorporado à aplicação. Dessa forma, o sistema reduz a latência de acesso às informações durante a partida e contribui para maior previsibilidade de custos no backend.

CONCLUSÃO

O desenvolvimento do AniCard permitiu a construção de um jogo de cartas móvel educativo voltado ao apoio do ensino de zoologia. A utilização do framework Flutter, associada aos princípios de Arquitetura Limpa, contribuiu para a organização modular da aplicação, separando a interface gráfica, a lógica da partida e os serviços de dados. Essa estrutura favoreceu a implementação de uma experiência interativa, com atualização dinâmica dos elementos visuais e maior clareza na organização do código.

Do ponto de vista da arquitetura de dados, a abordagem híbrida adotada mostrou-se adequada para o contexto da aplicação, ao combinar o armazenamento do progresso do aluno no Firebase Firestore com a consulta local dos atributos biológicos das cartas em arquivos JSON. Essa separação reduziu a dependência de leituras constantes em nuvem durante as partidas e contribuiu para melhorar o desempenho percebido da aplicação, além de favorecer maior previsibilidade no uso de recursos do backend.

Assim, o AniCard apresenta-se como uma ferramenta pedagógica lúdica, capaz de integrar conteúdos de zoologia, mecânicas de jogo e princípios de engenharia de software. A proposta contribui para o uso da gamificação no ensino de Ciências, ao aproximar conceitos biológicos de uma experiência interativa e acessível em dispositivos móveis.

Como trabalhos futuros, pretende-se ampliar a experiência do usuário por meio da implementação de partidas multiplayer, com avaliação de alternativas baseadas em WebSockets ou streams do Firestore. Também se prevê a inclusão de animações adicionais, o aprimoramento da experiência visual do jogo e a integração do sistema de recompensas ao perfil persistente do estudante no Firestore.

REFERÊNCIAS

AIRBYTE. **Google Firestore Pricing Guide**: Real-World Costs & Optimization Tips. *Airbyte Pricing Insights*, [s. l.], 2025.

FIREBASE. **Optimize query performance | Firestore**. *Google Cloud Documentation*, [s. l.], 2026.

GAMIFICAÇÃO aplicada no ensino de biologia. **Revista Mirante**, Anápolis, v. 15, n. 1, jun. 2022.

MOURA, Isac Maximo Alves de. **Explorando o uso de prompts para IA generativa no auxílio aos estudos**: desenvolvimento de um aplicativo educacional com Flutter. 2025. Trabalho de Conclusão de Curso (Graduação em Engenharia de Software) – Universidade Federal do Ceará, Russas, 2025.

NANAVATI, J.; PATEL, A.; PATEL, S.; PATEL, U. Critical Review and Fine-Tuning Performance of Flutter Applications. *In*: INTERNATIONAL CONFERENCE ON MOBILE COMPUTING AND SUSTAINABLE INFORMATICS (ICMCSI), 5., 2024. **Proceedings** [...]. [S. l.]: IEEE, 2024. p. 131-136.

RUSSO, Caique Antonio. **A ascensão do modelo de jogos online gratuitos**. 2024. 61 f. Trabalho de Conclusão de Curso (Graduação em Ciências Econômicas) – Instituto de Economia, Universidade Estadual de Campinas, Campinas, 2024.

SANTOS, J. S. *et al*. A gamificação no ensino de biologia: contribuições do bingo como recurso de aprendizagem. *In*: ENCONTRO NACIONAL DAS LICENCIATURAS (ENALIC), 8., 2021. **Anais** [...]. [S. l.: s. n.], 2021. v. 1.