Documentação do Sistema - DS

IDENTIFICAÇÃO DO PROJETO

Projeto

- Slide Puzzle - ODSlize

Disciplina do Projeto

- Gerência de Configuração e Mudanças

Equipe do Projeto

- GEISA MORAIS GABRIEL - 2024012594;
- LEONARDO INACIO GUILHERME DANTAS - 2024012595;
- TIAGO AMARO NUNES - 2024012611.

HISTÓRICO DE REGISTROS

Versão	***Data***	***Autor(es)***	***Descrição***
0.1	*17/09/2025*	*GEISA MORAIS GABRIEL* *LEONARDO INACIO GUILHERME DANTAS*	*- Apresentação* *- Lista inicial de requisitos* *- Regras de negócio*
0.2	*22/09/2025*	*GEISA MORAIS GABRIEL* *TIAGO AMARO NUNES*	*- Regras de negócio (update)* *- Diagrama de classes* *- Visão geral do sistema* *- Lista de requisitos (update)*
0.3	*24/09/2025*	*LEONARDO INACIO GUILHERME DANTAS*	*- Arquitetura do sistema*
0.4	*28/09/2025*	*LEONARDO INACIO GUILHERME DANTAS*	*- Links padrões de projeto* *- Diagrama de Caso e Uso*

SUMÁRIO

1. APRESENTAÇÃO

Este documento procura descrever as especificações do sistema ODSlize, um jogo digital de *slide puzzle*. A documentação inclui a descrição geral do sistema, requisitos funcionais e não funcionais, arquitetura do sistema, modelagem de dados, diagramas (de classes e de caso de uso) e regras de negócio. Esses artefatos servem de base para o desenvolvimento, elaboração de testes e manutenções futuras.

O jogo sobre os 17 Objetivos do Desenvolvimento Sustentável (ODS) trata-se de um slide puzzle, desafiando o jogador a deslizar peças em um tabuleiro a fim de organizá-lo a partir de uma configuração específica. O objetivo é utilizar um espaço vazio para mover as peças adjacentes sem as retirar até que a imagem ou sequência seja completada.

2. VISÃO GERAL DO SISTEMA

A gamificação consiste em aplicar elementos do design de jogos em ambientes não relacionados a jogos (Deterding et. al., 2011). Para o ensino e aprendizagem, jogos digitais educativos são considerados abordagens eficientes (Borges; Neves, 2023).

Segundo Werbach e Hunter (2014), os elementos de gamificação podem ser divididos em três níveis: dinâmicas, mecânicas e componentes (DMC).

- **Dinâmicas**: são os aspectos "gerais" do sistema gamificado que devem ser considerados e gerenciados, mas que nunca podem entrar diretamente no jogo. Os exemplos de dinâmica dados por Werbach (Werbach; Hunter, 2014) são: restrições, emoções, narrativas, progressão e relacionamentos.
- **Mecânicas**: os processos básicos que impulsionam a ação e geram o engajamento dos jogadores; por exemplo, desafios, mudanças, competição, cooperação, feedback, recursos, aquisições, recompensas, transações, turnos, estados de vitória e perfis (Werbach; Hunter, 2014).
- **Componentes**: são instâncias específicas da mecânica e dinâmica, são os elementos com os quais os jogadores interagem diretamente. Por exemplo: conquistas, avatares distintivos, brigas de chefes, coleções, combate, desbloqueio de conteúdo, tabelas de classificação, níveis, pontos, missões, gráficos sociais, equipes e produtos virtuais (Werbach; Hunter, 2014).

Além disso, os detalhes na construção do design para a gamificação devem ser planejados e fundamentados para garantir o sucesso da gamificação (Rocha et al., 2021). Para isso, é preciso entender o contexto e o problema, bem como incluir a descrição dos usuários de um sistema gamificado. Podendo ser considerados dados demográficos, faixas etárias, tipo de comportamento, por exemplo (Werbach; Hunter, 2012).

2.1. Aplicação do Modelo DMC no ODSlize

Unidade 2: tratar do contexto e problema e DMC implementados.

2.2. Perfil de jogador

Unidade 2: tratar do perfil de jogador.

3. REQUISITOS DO SISTEMA

3.1. Requisitos Funcionais

- **[RF001] Iniciar partida** - O sistema deve permitir que o jogador inicie uma nova partida, com o tabuleiro embaralhado automaticamente de forma que possa ser resolvido.

  ☑ Essencial ☐ Importante ☐ Desejável

- **[RF002] Movimentar peças** - O sistema deve permitir que o jogador mova uma peça para a casa vazia adjacente por meio de um clique ou gesto (arrastando).

  ☑ Essencial ☐ Importante ☐ Desejável

- **[RF003] Reiniciar partida** - O sistema deve permitir ao jogador reiniciar a partida atual com um novo embaralhamento.

  ☑ Essencial ☐ Importante ☐ Desejável

- **[RF004] Contabilizar tempo e movimentos** - O sistema deve contabilizar o tempo decorrido desde o início da partida e a quantidade de movimentos realizados por rodada.

  ☐ Essencial ☑ Importante ☐ Desejável

- **[RF005] Possuir níveis de dificuldade** - O sistema deve permitir que o jogador avance no jogo por meio de níveis de dificuldade que devem variar conforme o tamanho do tabuleiro (ex.: 3x3, 4x4).

  ☐ Essencial ☑ Importante ☐ Desejável

- **[RF006] Armazenar temporariamente resultados do jogo** - O sistema deve armazenar temporariamente o melhor tempo de jogo e a menor quantidade de movimentos daquele jogador.

  ☐ Essencial ☐ Importante ☑ Desejável

- **[RF007] Informar a vitória do jogador** - O sistema deve verificar automaticamente se as peças estão na ordem correta, e em caso afirmativo, deve informar a vitória do jogador.

  ☑ Essencial ☐ Importante ☐ Desejável

- **[RF008] Explicar o ODS** - Após o estado de vitória, o sistema deve retornar uma explicação sobre o ODS associado àquele *slide puzzle*, promovendo o conhecimento ao jogador.

  ☐ Essencial ☑ Importante ☐ Desejável

- **[RF009] Apresentar resultado esperado** - O sistema deve apresentar o resultado esperado para que o jogador visualize a organização esperada das peças.

☑ Essencial ☐ Importante ☐ Desejável

3.2. Requisitos Não Funcionais

- **[RNF001] - Usabilidade (operabilidade):** a interface do sistema deve ser responsiva, adaptando-se automaticamente a diferentes tamanhos de tela.

  ☑ Essencial ☐ Importante ☐ Desejável

- **[RNF002] - Usabilidade (aprendizibilidade):** o sistema deve informar as instruções de como se joga, facilitando o aprendizado do usuário.

  ☐ Essencial ☑ Importante ☐ Desejável

- **[RNF003] - Desempenho (comportamento em relação ao tempo):** o tempo de resposta para as ações do usuário no jogo deve ser inferior a 1 segundo.

  ☑ Essencial ☐ Importante ☐ Desejável

3.3. Requisitos Funcionais Futuros

- **[RFF001] -** O sistema deve fornecer *feedback* visual (destacando a peça selecionada) e efeitos sonoros opcionais ao mover peças ou vencer uma partida.

  ☐ Essencial ☐ Importante ☑ Desejável

- **[RFF002]** O sistema deve permitir ao jogador salvar o estado atual do jogo e retomar mais tarde do ponto onde parou.

  ☐ Essencial ☐ Importante ☑ Desejável

- **[RFF003]** O sistema deve conceder XP (pontos de experiência) ao jogador sempre que ele concluir uma partida com sucesso, com base no desempenho.

  ☐ Essencial ☐ Importante ☑ Desejável

- **[RFF004]** O sistema deve contar o número de dias que o usuário jogou.

  ☐ Essencial ☐ Importante ☑ Desejável

- **[RFF005]** O sistema deve possuir modos de jogo (ex.: modo história, modo personalizado).

  ☐ Essencial ☐ Importante ☑ Desejável

- **[RFF006]** O sistema deve possuir a opção de pausa para suspender o tempo para a resolução de jogo.

  ☐ Essencial ☐ Importante ☑ Desejável

4. ARQUITETURA DO SISTEMA

Priorizando uma experiência de usuário fluida, responsiva e com baixa latência, a arquitetura do sistema ODSlize foi definida como uma *Single Page Application* (SPA). Esta abordagem centraliza a maioria da lógica e da renderização no cliente (navegador), minimizando a dependência de um servidor de backend complexo para execução das operações principais do jogo.

4.1. Visão Geral

O sistema é dividido em duas grandes áreas funcionais:

- Aplicação Frontend (SPA): core do jogo, responsável por toda a interação do usuário, lógica de negócio, persistência de dados do jogador (*localStorage*) e gerenciamento de estado.
- Serviços de Mídia em Nuvem: voltados para a entrega eficiente e global de todos os ativos visuais do jogo (imagens).

Esta arquitetura visa a máxima otimização da experiência de jogo no navegador, com carregamento rápido e transições instantâneas entre as telas do jogo.

4.2. Componentes da arquitetura

4.2.1. Aplicação Frontend (Single Page Application - SPA)

A camada de cliente proporciona uma experiência de usuário contínua e imersiva, eliminando carregamentos de página, executando a lógica completa do jogo diretamente no navegador e garantindo alta responsividade e desempenho. Além disso, é responsável por gerenciar todas as interações do usuário e apresentar *feedback* visual.

Em relação às tecnologias, a aplicação será construída nos pilares da web moderna: HTML5, CSS3 e JavaScript. O HTML5 é utilizado para estruturar o conteúdo do jogo, o CSS3 garante a estilização e a responsividade da interface, adaptando-se a diferentes tamanhos de tela, e por fim, o JavaScript é responsável por implementar toda a interatividade e a lógica principal do jogo.

Para a construção da interface, será utilizado o framework JavaScript React. Essa escolha ocorre devido à facilidade de construir componentes modulares e reativos, tornando a UI dinâmica e de fácil manutenção.

Quanto à persistência de dados do jogador, como melhores tempos e quantidade de movimentos, será feita com o localStorage do navegador. Ele oferece um mecanismo simples para armazenar dados-chave/valor no navegador do usuário, que permanecem acessíveis mesmo após o fechamento da aba ou do navegador, simplificando a arquitetura.

4.2.2. Serviços de Mídia em Nuvem

Os ativos visuais do jogo e informações estáticas, como as imagens dos ODS e alguns conteúdos, que compõem os *puzzles*, foram adotados uma abordagem de entrega de conteúdo otimizada. Os objetivos são garantir o

carregamento rápido e confiável dessas imagens, oferecer escalabilidade e alta disponibilidade para os *assets*, e reduzir a carga sobre a aplicação principal.

Para alcançar esses objetivos, o Amazon S3 (*Simple Storage Service*) é utilizado para o armazenamento durável, seguro e altamente escalável para todas as imagens, assets estáticos do jogo e demais recursos gráficos do ODSlize.

Complementando o S3, será utilizado o Amazon CloudFront (ou outro serviço de CDN). Integrado ao S3, o CloudFront atua como uma *Content Delivery Network* (CDN), que cacheia as imagens em servidores ("*edge locations*") distribuídos globalmente. Esta estratégia significa que, independentemente da localização geográfica do jogador, as imagens serão entregues a partir do servidor mais próximo, reduzindo a latência e acelerando significativamente o tempo de carregamento dos ativos visuais, o que contribui diretamente para uma experiência de usuário fluida e responsiva.

4.2.3. Controle de Versão (Git com GitHub)

O controle de versão é essencial para o trabalho em equipe, permitindo o rastreamento de mudanças, colaboração entre os desenvolvedores conforme o histórico de registros, facilitando a integração contínua.

O uso, possibilita automatizar o processo de *deploy* a cada nova versão do código, em diversas plataformas como Netlify, Vercel GitHub Pages ou AWS Amplify, além de geralmente configurarem HTTPS automaticamente, uma prática recomendada para qualquer aplicação web moderna para garantir a segurança dos dados em trânsito.

4.3. Decisões Técnicas

Visando melhorar a qualidade do código, manutenibilidade e escalabilidade da lógica do jogo dentro do Frontend, será utilizada uma série de padrões de projeto consagrados. Estes padrões pretendem estruturar o código do jogo de tal forma modular e organizada, facilitando a adição de novas funcionalidades e a manutenção futura, e promovendo a colaboração eficiente entre os membros da equipe de desenvolvimento. Dentre eles estão:

- **State Pattern[1] (Padrão de Estado)**: utilizado para gerenciar os diferentes estados do jogo, como Embaralhando, Em Jogo, Pausado e Concluído. Normalmente, esse padrão encapsula o comportamento específico de cada estado em objetos distintos, simplificando transições entre eles e tornando a lógica do jogo mais simples e clara, assim como menos propensa a erros, evitando extensos blocos condicionais e melhorando a organização do código.

- **Strategy Pattern[2] (Padrão de Estratégia)**: define algoritmos específicos para determinados comportamentos variáveis é muito útil em casos como a *ShufflerStrategy* para o embaralhamento do tabuleiro e a *ScoringStrategy* para a contagem de pontos e métricas. Este padrão permite trocar algoritmos em tempo de execução, facilitando a implementação de diferentes níveis de

[1] Disponível em: https://refactoring.guru/design-patterns/state
[2] Disponível em: https://refactoring.guru/design-patterns/strategy

dificuldade ou modos de jogo sem a necessidade de modificar o código central da aplicação, promovendo flexibilidade e extensibilidade.

- **Factory[3] (Padrões de Fábrica)**: são empregados para a criação de objetos complexos, como instâncias de Board (tabuleiros) baseadas em parâmetros como o ID do ODS, por exemplo, o tamanho do tabuleiro e o conjunto de imagens. O objetivo é centralizar a lógica de criação de objetos, desacoplando o código cliente dos detalhes de instanciação, o que facilita a adição de novos tipos de tabuleiros ou temas visuais e garante que os objetos sejam criados de forma consistente.

- **Observer Pattern[4] (Padrão de Observador)**: muito utilizado para notificar componentes da interface do usuário sobre mudanças de estado no jogo, por exemplo, quando uma peça é movida ou quando o jogo é ganho. Este padrão permite que diferentes partes do sistema reajam a eventos importantes de forma desacoplada, melhorando muito a reatividade da interface do usuário e facilitando a implementação de funcionalidades como telemetria interna ou atualizações visuais dinâmicas.

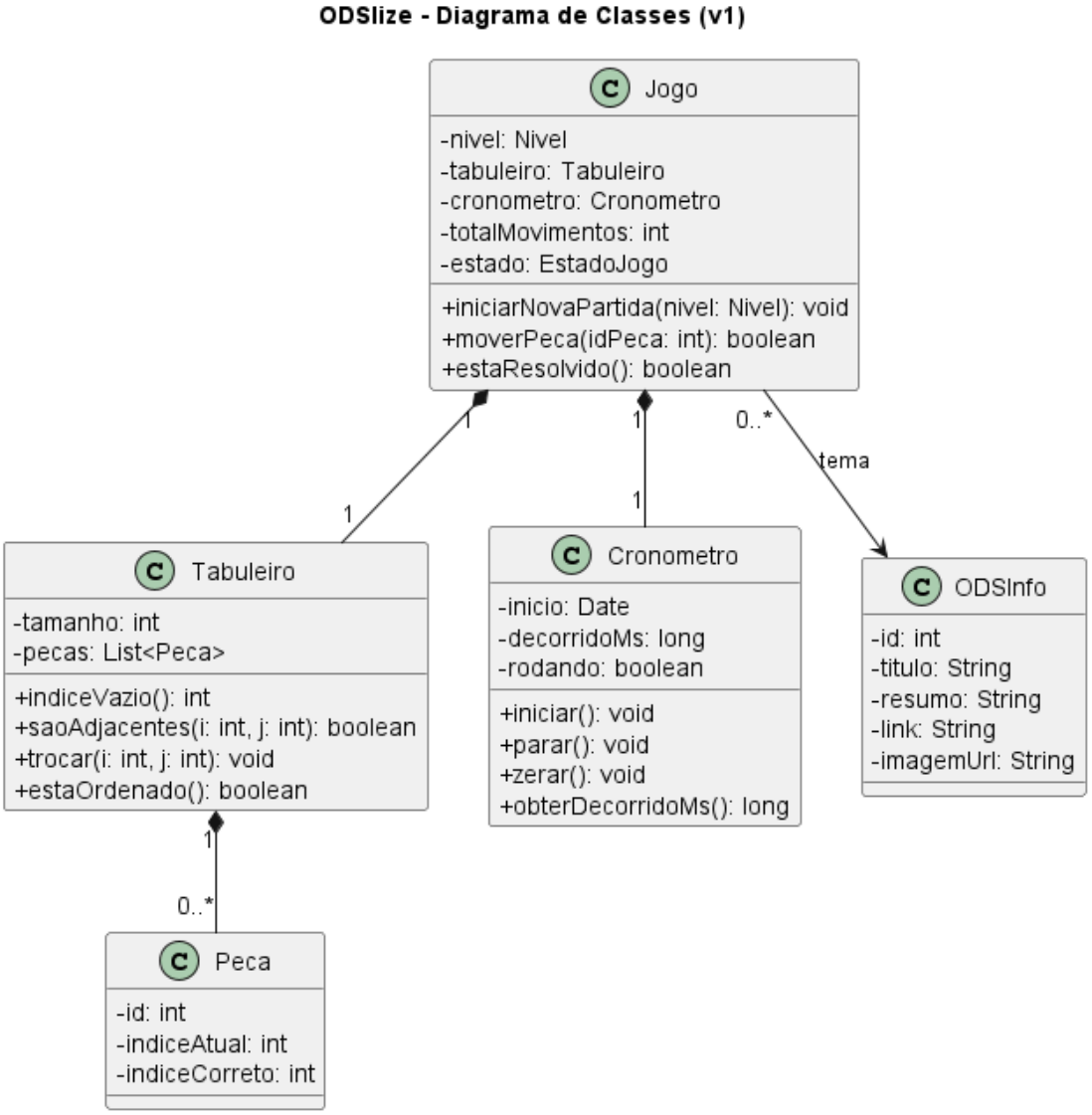
- **Command Pattern[5] (Padrão de Comando)**: aplicado para representar as ações do jogador como objetos, como um *MoveTileCommand*. Isso permite desacoplar a interface do usuário das operações que ela invoca, abrindo caminho para possíveis funcionalidades futuras como "desfazer/refazer" movimentos, podendo até mesmo facilitar a implementação de *replays* ou a gravação de sequências de ações para testes.

Essa arquitetura descrita, foi desenhada para otimizar o desenvolvimento, manter a simplicidade operacional e entregar uma aplicação performática e agradável ao usuário.

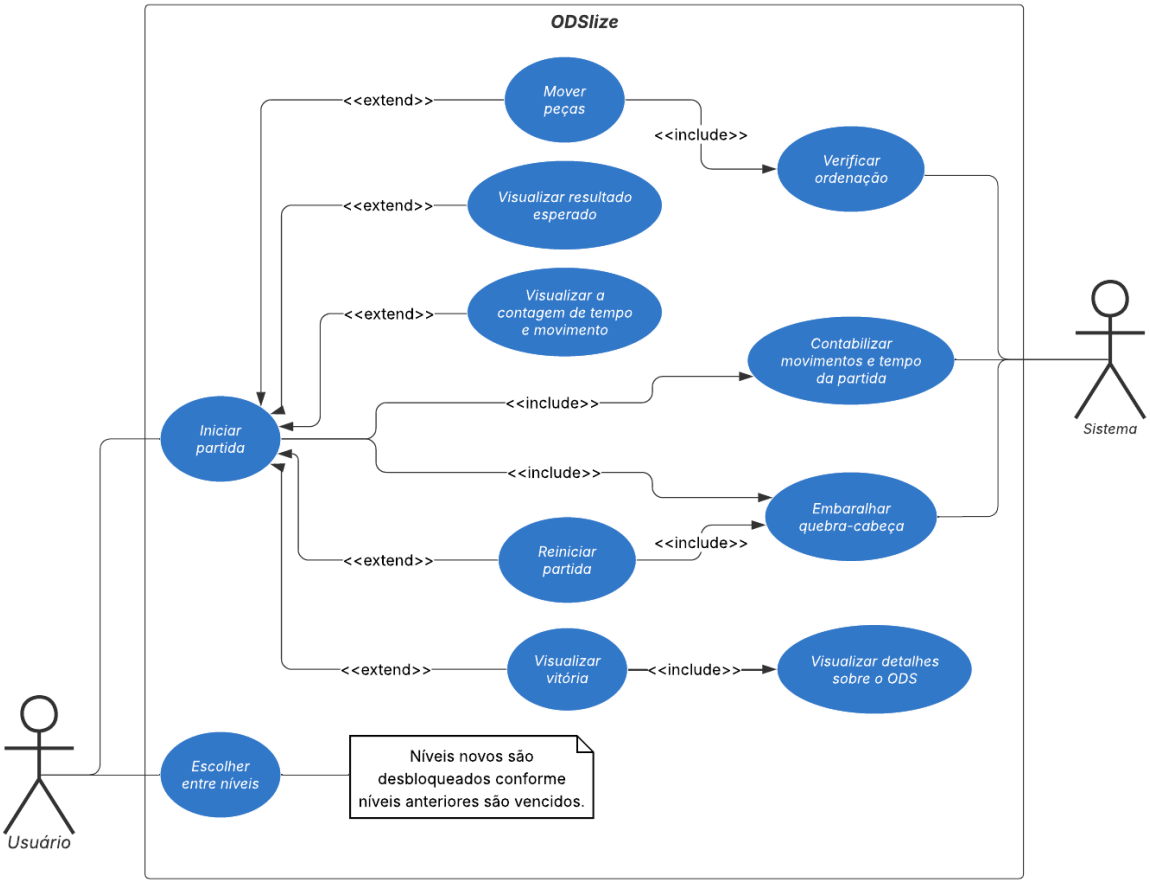
[3] Disponível em: https://refactoring.guru/design-patterns/factory-method
[4] Disponível em: https://refactoring.guru/design-patterns/observer
[5] Disponível em: https://refactoring.guru/design-patterns/command

5. DIAGRAMAS

5.1. Diagrama de classes



5.2. Diagrama de caso de uso



6. REGRAS DE NEGÓCIO

- **[RN001] -** Em qualquer momento do jogo, o tabuleiro deve conter exatamente uma casa vazia (sem uma peça).
- **[RN002] -** Uma peça pode ser movida apenas para uma casa que esteja adjacente (horizontal ou vertical) à casa vazia.
- **[RN003] -** Não é permitido mover mais de uma peça ao mesmo tempo.
- **[RN004] -** Ao iniciar uma nova partida, as peças devem estar em uma configuração desordenada, garantindo que o tabuleiro seja matematicamente solúvel.
- **[RN005]** - Uma partida é considerada vitoriosa quando todas as peças são organizadas na sua sequência numérica ou visual pré-definida, reconstituindo a imagem ou padrão do ODS associado ao nível.
- **[RN006]** - Cada *slide puzzle* deve estar associado a, no mínimo, um dos 17 Objetivos do Desenvolvimento Sustentável (ODS), e esta associação determina a imagem/tema a ser montado.
- **[RN007]** - Um "movimento" é definido como o deslizamento de uma peça para a casa vazia. O "tempo de jogo" é contabilizado desde o início da partida até o momento em que a condição de vitória é detectada.

7. REFERÊNCIAS

BORGES, R. K.; NEVES, C. A. de A. Explorando jogos educativos para o ensino dos objetivos de desenvolvimento sustentável (odss: Uma revisão sistemática. Revista Gestão & Sustentabilidade Ambiental, v. 12, p. e20167–e20167, 2023.

DETERDING, S. et al. Gamification. using game-design elements in non-gaming contexts. In: ACM. CHI'11 Extended Abstracts on Human Factors in Computing Systems. [S.l.], 2011.

ROCHA, F. D. F. d. et al. Gamificação e modelo aberto de aprendizagem: estudo experimental sobre os efeitos nas características de autorregulação da aprendizagem. Universidade Federal de Alagoas, 2021.

WERBACH, K. (re)defining gamification: A process approach. In: Persuasive Technology: 9th International Conference, PERSUASIVE 2014, Padua, Italy, May 21-23, 2014. Proceedings. Cham: Springer International Publishing, 2014.

WERBACH, K.; HUNTER, D. For the win: how game thinking can revolutionize your business. [S.l.]: Wharton, 2012.