

Documento de Modelagem UML: Jogo "Dekora"

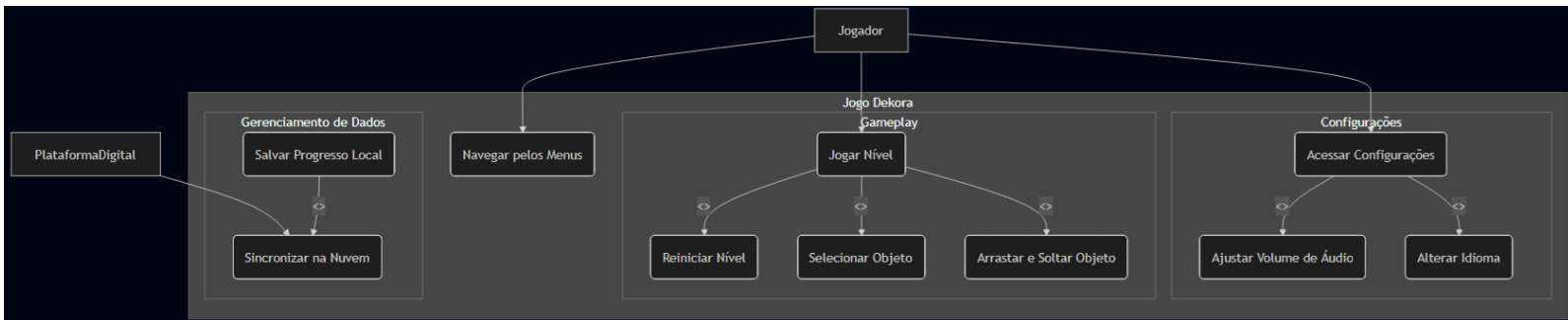
Versão: 1.0 Data: 13 de outubro de 2025

1. Introdução

Este documento apresenta a modelagem UML do sistema do jogo "Dekora". Seu objetivo é definir a arquitetura de software e as interações do usuário, servindo como um guia técnico claro para a equipe de desenvolvimento. A modelagem foi dividida em dois artefatos principais: o Diagrama de Casos de Uso e o Diagrama de Classes.

2. Diagrama de Casos de Uso

O Diagrama de Casos de Uso ilustra as funcionalidades do sistema do ponto de vista do usuário (o "Jogador") e as interações com sistemas externos (a "Plataforma Digital"). Ele foca em responder "O quê?" o sistema deve fazer.



2.1. Atores

- Jogador:** O ator principal do sistema. Ele é o usuário que busca a experiência de decoração e interage diretamente com a jogabilidade, menus e configurações.
- Plataforma Digital:** Um ator secundário que representa sistemas externos, como a Steam. Ele interage com o sistema para fornecer funcionalidades estendidas, como o salvamento em nuvem.

2.2. Principais Casos de Uso

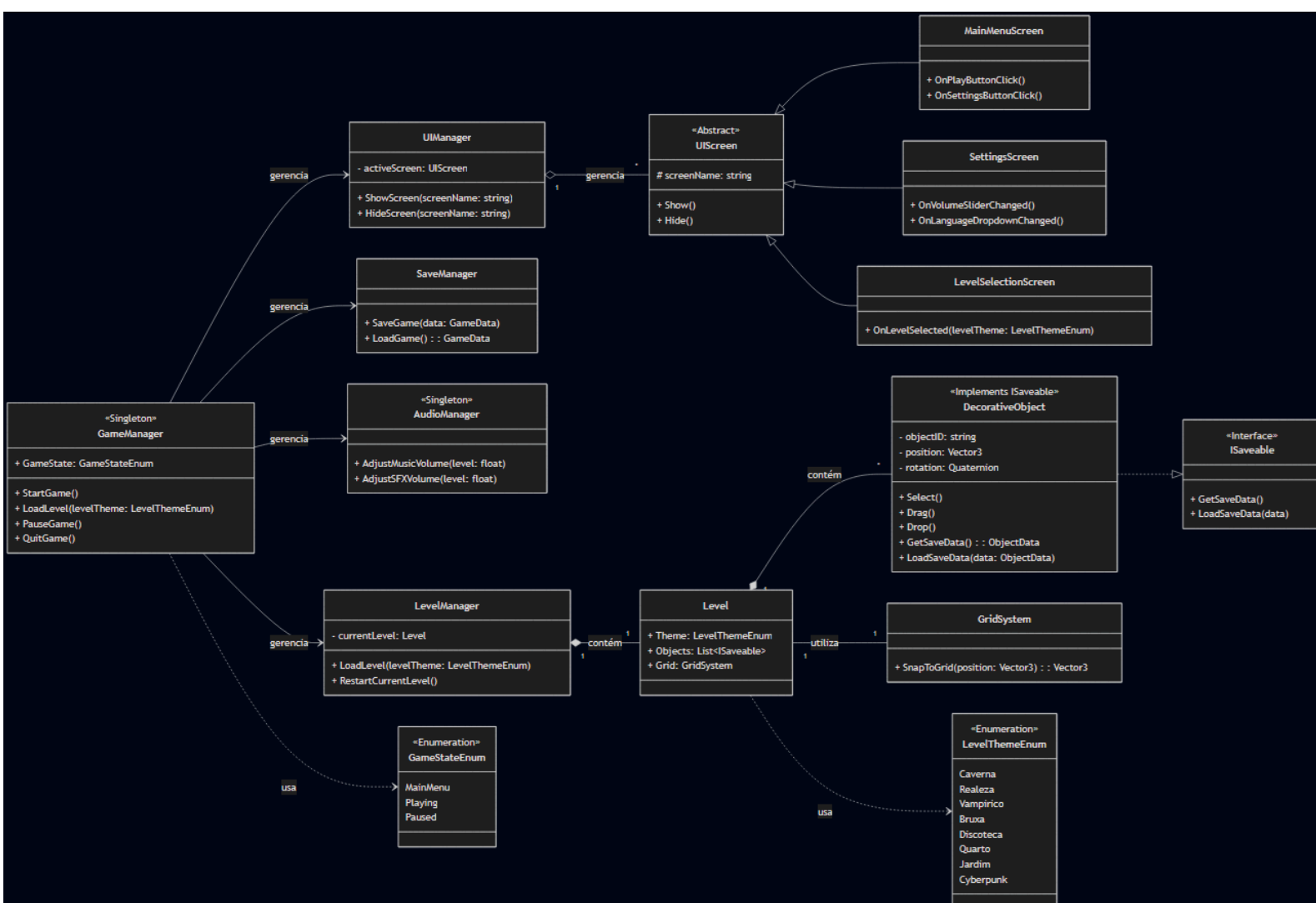
O diagrama é organizado em pacotes que agrupam funcionalidades relacionadas:

- Gameplay:** É o coração do jogo.
 - Jogar Nível:** O caso de uso central. Para que ele funcione, ele obrigatoriamente inclui as funcionalidades de **Selecionar Objeto** e **Arrastar e Soltar Objeto**.
 - Reiniciar Nível:** Esta é uma funcionalidade opcional, tratada como um extensão do caso de uso "Jogar Nível".
- Configurações:**

- **Acessar Configurações:** É o caso de uso base para o jogador gerenciar suas preferências. Ele inclui as ações de **Ajustar Volume de Áudio** e **Alterar Idioma**.
- **Gerenciamento de Dados:**
 - **Salvar Progresso Local:** O sistema salva o progresso do jogador.
 - **Sincronizar na Nuvem:** Esta funcionalidade estende o salvamento local, sendo uma interação opcional que depende da Plataforma Digital.

3. Diagrama de Classes (Arquitetura do Sistema)

O Diagrama de Classes detalha a estrutura estática do software, suas classes, atributos, métodos e os relacionamentos entre eles. Ele foca em responder "**Como?**" o sistema será construído.



3.1. Visão Geral da Arquitetura

A arquitetura foi projetada para ser modular, flexível e de fácil manutenção, aplicando os seguintes padrões e princípios:

- **Divisão de Responsabilidades (Managers):** A lógica central é dividida em Gerenciadores (**Managers**), onde cada um tem uma responsabilidade única.
 - **GameManager (Singleton):** Controla o estado geral do jogo (MainMenu, Playing, Paused) e coordena os outros gerentes.
 - **LevelManager:** Responsável exclusivamente por carregar, reiniciar e gerenciar o estado da fase (**Level**) atual.
 - **SaveManager:** Dedicado a serializar (salvar) e desserializar (carregar) os dados do jogo.
 - **UIManager:** Gerencia o ciclo de vida das telas de interface (ex: mostrar/esconder menus).
 - **AudioManager (Singleton):** Gerencia a execução de músicas e efeitos sonoros de forma global.
- **Herança no Sistema de UI:**
 - Para evitar código duplicado, foi criada uma classe `<<Abstract>> UIScreen`.
 - Todas as telas concretas (**MainMenuScreen**, **SettingsScreen**, **LevelSelectionScreen**) herdam desta classe base. Isso permite ao **UIManager** tratá-las de forma genérica, facilitando a adição de novas telas no futuro.
- **Interface para Salvamento Flexível:**
 - A `<<Interface>> ISaveable` define um "contrato" que obriga as classes a implementar métodos para obter e carregar dados de salvamento.
 - A classe **DecorativeObject** implementa esta interface.
 - **Vantagem:** O **SaveManager** não precisa saber o que é um **DecorativeObject**. Ele apenas pergunta: "Você é **ISaveable**? Me dê seus dados". Isso torna o sistema de save extensível para qualquer objeto (ex: **ConfiguraçõesDoJogador**) sem alterar o **SaveManager**.
- **Enumerações (Enums) para Segurança:**
 - Para evitar erros de digitação com "textos mágicos", foram criados **Enums**.
 - **GameStateEnum:** Define os estados de jogo permitidos (MainMenu, Playing, Paused).
 - **LevelThemeEnum:** Define os temas de fases disponíveis (Caverna, Realeza, etc.).

3.2. Classes do Core Gameplay

- **Level:** Contém os dados de uma fase específica, incluindo sua lista de objetos (**DecorativeObject**) e a instância do **GridSystem** que ela utiliza.
- **DecorativeObject:** Representa o item que o jogador pode selecionar, arrastar e soltar. É a classe principal de interação.
- **GridSystem:** Uma classe de lógica auxiliar que fornece a funcionalidade de "snap" (magnetismo) ao grid, usada pelo **Level**.

Declaração de Uso de Ferramentas

A elaboração deste documento contou com o suporte de uma ferramenta de Inteligência Artificial (IA). A IA foi utilizada como uma assistente para a estruturação de seções, brainstorming de ideias e refinamento da redação. Todo o conteúdo foi subsequentemente revisado, editado e validado pelos autores humanos, que assumem total responsabilidade pela sua precisão e versão final.