# Table of Contents

# Introdução

Bem-vindo à documentação do projeto. Esta documentação serve como um recurso central para desenvolvedores, designers e qualquer pessoa interessada em entender os detalhes técnicos e de arquitetura do projeto.

Nosso objetivo é fornecer informações claras, concisas e práticas para que você possa começar a trabalhar. Seja você um novo membro da equipe aprendendo o básico ou um contribuidor procurando por informações específicas, você deve encontrá-las aqui.

Esta documentação abrange:

- **Arquitetura:** Visão geral de alto nível do design e dos componentes do sistema.
- **Requisitos de Engenharia:** As especificações e restrições técnicas.
- **Recursos de Desenvolvimento:** Guias e referências para as ferramentas e bibliotecas utilizadas.
- **Recursos do Jogo:** Informações sobre sprites, sons e outros recursos criativos.

Vamos começar!

# Visão Geral do Projeto

Este documento oferece uma visão abrangente do projeto de recriação do clássico jogo **Space Invaders**. Desenvolvido como parte da disciplina de Programação 3, o projeto foca na aplicação prática de conceitos de Programação Orientada a Objetos (OOP), Estruturas de Dados e manipulação de eventos em C#.

## 1. Objetivo e Escopo

O principal objetivo é desenvolver uma aplicação de desktop funcional que recrie a experiência do Space Invaders original. O projeto é um esforço individual, construindo a interface e a lógica do jogo do zero, sem o auxílio de um motor de jogo externo (game engine). As entregas são divididas em fases parciais e uma entrega final, abrangendo todos os requisitos funcionais e não-funcionais.

## 2. Tecnologias Utilizadas

O projeto é construído sobre uma pilha de tecnologias modernas e robustas, garantindo portabilidade e desempenho:

- **Linguagem de Programação**: C#
- **Plataforma**: Uno Platform (para desenvolvimento de aplicações desktop multiplataforma)
- **Interface de Usuário (UI)**: XAML
- **Persistência de Dados**: Entity Framework Core com PostgreSQL (para gerenciamento de placares)
- **Padrão de Arquitetura**: MVVM (Model-View-ViewModel)
- **Áudio**: NAudio (para manipulação de efeitos sonoros.
- **SCM**: GitLab

> **Nota**: Em ambientes Linux, a reprodução de áudio é gerenciada através de um processo externo que invoca o `mpg123`, garantindo a compatibilidade com o

sistema operacional.)

## 3. Principais Funcionalidades

O jogo implementa as seguintes funcionalidades chave:

- **Controle da Nave**: Movimento horizontal do jogador e disparo de lasers.
- **Inimigos (Alienígenas)**: Múltiplas ondas de alienígenas com movimento em bloco, descida progressiva e aumento de velocidade/frequência de disparo. Inclui alienígenas especiais com pontuação variável.
- **Barreiras de Proteção**: Escudos destrutíveis que bloqueiam tiros.
- **Sistema de Pontuação e Vidas**: Gerenciamento de pontuações por tipo de alienígena e controle de vidas do jogador.
- **Fim de Jogo**: Condições claras para o término do jogo (perda de todas as vidas ou alienígenas atingindo a base).
- **Persistência de Dados**: Salvamento de placares (pontuação e apelido) em um arquivo de texto.
- **Interface do Usuário**: Telas iniciais, de jogo, de fim de jogo e de placares.
- **Efeitos Sonoros**: Sons para ações significativas do jogo.

## 4. Estrutura do Projeto

O projeto segue uma estrutura organizada para facilitar o desenvolvimento e a manutenção:

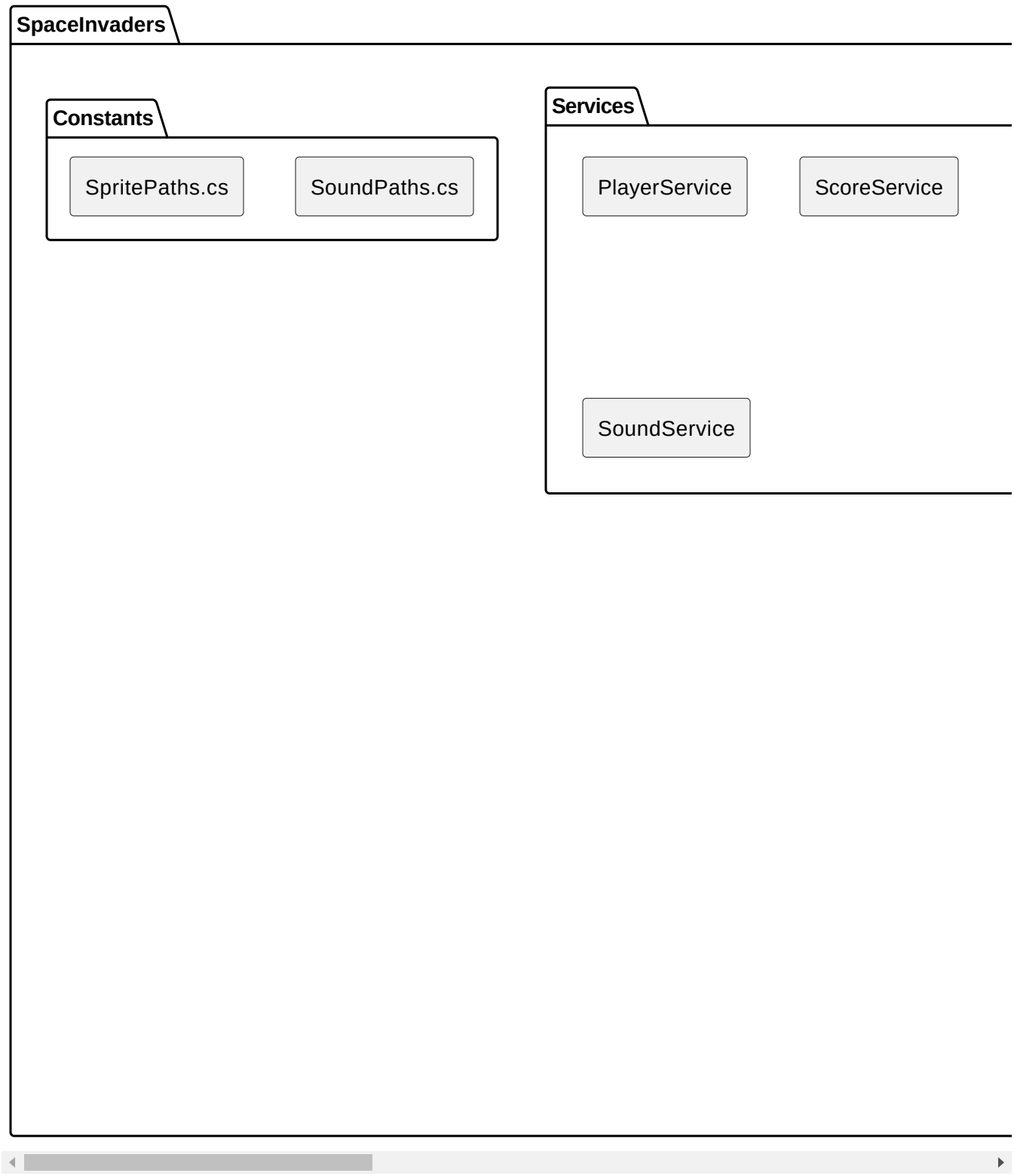
- `SpaceInvaders/Models`: Define as entidades de dados do jogo (ex: `Player`, `Alien`, `Score`).
- `SpaceInvaders/ViewModels`: Contém a lógica de apresentação e a interação entre a UI e os modelos.
- `SpaceInvaders/Services`: Implementa a lógica de negócio e a interação com o banco

de dados e outros serviços (ex: `PlayerService`, `ScoreService`, `SoundService`).

- **`SpaceInvaders/Data`**: Gerencia o contexto do banco de dados e as migrações do Entity Framework Core.
- **`SpaceInvaders/Presentation`**: Contém os arquivos XAML que definem a interface do usuário.
- **`SpaceInvaders/Constants`**: Armazena constantes globais, como caminhos para assets.
- **`SpaceInvaders/Assets`**: Inclui todos os recursos visuais (sprites, backgrounds) e sonoros (músicas, efeitos) do jogo.

## Diagrama de Estrutura do Projeto

Abaixo, um diagrama que ilustra a organização das principais pastas e a relação entre elas:




## 5. Fases do Projeto

O desenvolvimento foi dividido em fases:

- **Entrega Intermediária (Midterm)**: Foco em um subconjunto de requisitos funcionais, incluindo movimento e disparo do jogador, alienígenas estáticos, escudos destrutíveis, atualização de pontuação em memória, tela inicial e sons básicos.
- **Entrega Final**: Implementação de todos os requisitos funcionais e não-funcionais, incluindo movimento complexo dos alienígenas, persistência de placares e todas as telas e mecânicas detalhadas.

## 6. Ambiente de Desenvolvimento com Docker

Para facilitar a configuração do ambiente de desenvolvimento, especialmente para o banco de dados PostgreSQL, o projeto utiliza Docker. O arquivo `docker-compose.yml` define um serviço de banco de dados que pode ser facilmente inicializado:

```yaml
version: '3.8'

services:
  db:
    image: postgres:16-alpine
    restart: always
    environment:
      POSTGRES_DB: spaceinvaders_db
      POSTGRES_USER: user
      POSTGRES_PASSWORD: password
    ports:
      - "5432:5432"
    volumes:
      - db_data:/var/lib/postgresql/data

volumes:
  db_data:
```

Para iniciar o banco de dados, basta executar o comando `docker-compose up -d db` ou `docker-compose up -d db` no diretório raiz do projeto. Isso garantirá que o banco de dados PostgreSQL esteja disponível e configurado para o desenvolvimento e testes.

# Guia de Instalação e Execução

Este guia detalha os passos necessários para configurar o ambiente de desenvolvimento, instalar as dependências e executar o projeto Space Invaders.

## 1. Pré-requisitos

Certifique-se de que os seguintes softwares estejam instalados em sua máquina:

- **SDK do .NET 9.0 (ou superior)**: Essencial para compilar e executar aplicações .NET. Você pode baixá-lo em dotnet.microsoft.com/download (https://dotnet.microsoft.com/download).
- **Docker Desktop (ou Docker Engine)**: Necessário para rodar o banco de dados PostgreSQL. Disponível em docker.com/products/docker-desktop (https://www.docker.com/products/docker-desktop).
- **Visual Studio 2022 (ou Visual Studio Code ou Rider)**: IDEs recomendadas para desenvolvimento C# e Uno Platform. Embora não sejam estritamente obrigatórias, facilitam muito o processo.
  - **Visual Studio**: Certifique-se de ter as cargas de trabalho "Desenvolvimento para desktop com .NET" e "Desenvolvimento multiplataforma de interface do usuário .NET" instaladas.
- **Templates do Uno Platform**: Para criar e gerenciar projetos Uno Platform, você precisará instalar os templates. Execute o seguinte comando no terminal:

  ```
  dotnet new install Uno.Templates
  ```

  Para mais detalhes, consulte a documentação oficial: platform.uno/docs/articles/get-started.html (https://platform.uno/docs/articles/get-started.html)

## 2. Configuração do Banco de Dados (PostgreSQL com Docker)

O projeto utiliza PostgreSQL para persistência de dados (placares), gerenciado via

Docker para simplificar a configuração.

1. **Navegue até o diretório raiz do projeto** no seu terminal (onde o arquivo `docker-compose.yml` está localizado):
2. **Inicie o serviço do banco de dados** usando Docker Compose:

   ```
   docker-compose up -d db
   ```

   Este comando irá baixar a imagem do PostgreSQL (se ainda não tiver), criar e iniciar um contêiner de banco de dados configurado para o projeto. O banco de dados estará acessível na porta `5432`.
3. **Verifique se o contêiner está rodando** (opcional):

   ```
   docker ps
   ```

   Você deverá ver um contêiner chamado `capstone-programming-3-db-1` (ou similar) na lista.

## 3. Configuração e Compilação do Projeto

1. **Navegue até o diretório do projeto SpaceInvaders**:

   ```
   cd SpaceInvaders/SpaceInvaders
   ```
2. **Restaure as dependências do projeto**:

   ```
   dotnet restore
   ```
3. **Compile o projeto**:

   ```
   dotnet build
   ```

   Este comando irá compilar o código-fonte e verificar se há erros.

# 4. Execução do Projeto

Após a compilação bem-sucedida e com o banco de dados rodando, você pode executar a aplicação.

1. **Execute a aplicação desktop**:

```
dotnet run --project SpaceInvaders/SpaceInvaders.csproj -f
net9.0-desktop
```

   A aplicação Space Invaders deverá ser iniciada em uma nova janela.

### Nota sobre Áudio no Linux

Se você estiver executando o projeto em um ambiente Linux, certifique-se de ter o `mpg123` instalado para a reprodução de áudio. O `SoundService` do projeto utiliza este utilitário externo para garantir a compatibilidade de áudio no Linux.

Para instalar o `mpg123` (exemplo para sistemas baseados em Debian/Ubuntu):

```
sudo apt-get update
sudo apt-get install mpg123
```

Com esses passos, você estará pronto para desenvolver e testar o projeto Space Invaders em sua máquina.

# BRD

## 1. Introdução

### 1.1. Objetivo do Projeto

O objetivo deste projeto é desenvolver uma recriação do clássico jogo "Space Invaders" como uma aplicação de desktop. O projeto servirá como avaliação final para a disciplina de Programação 3, focando na aplicação prática de conceitos de Programação Orientada a Objetos (OOP), Estruturas de Dados e manipulação de eventos em C#.

### 1.2. Escopo

O projeto será desenvolvido individualmente, utilizando C# e a Uno Platform, sem o auxílio de qualquer motor de jogo (game engine). A interface e a lógica do jogo serão construídas do zero, com entregas parciais e uma entrega final.

## 2. Requisitos Funcionais

### 2.1. Tela Inicial

- O jogo deve apresentar uma tela inicial com as seguintes opções:
  - **Iniciar um novo jogo:** Começa uma nova partida.
  - **Ver a tabela com os placares:** Exibe a lista de pontuações salvas.
  - **Ver os controles do jogo:** Mostra as instruções de como jogar.

### 2.2. Jogabilidade

#### 2.2.1. Nave do Jogador

- O jogador controla uma nave localizada na parte inferior da tela.
- A nave pode se mover horizontalmente (esquerda e direita).
- A nave pode disparar um projétil (laser) para cima.
- O jogador só pode ter um projétil ativo na tela por vez.

#### 2.2.2. Inimigos (Alienígenas)

- Múltiplas ondas de alienígenas são posicionadas na parte superior da tela.
- Os alienígenas se movem em um bloco coeso, da esquerda para a direita.
- Ao atingir a borda da tela, o bloco de alienígenas desce uma linha e inverte sua direção de movimento.
- A velocidade de movimento dos alienígenas e a frequência de seus disparos aumentam a cada vez que o bloco desce.
- Apenas o alienígena de 40 pontos pode atirar.
- Um alienígena especial (vermelho) aparece periodicamente no topo da tela, movendo-se horizontalmente e oferecendo uma pontuação variável.

#### 2.2.3. Barreiras de Proteção

- Haverá 4 blocos de proteção (escudos) localizados acima da nave do jogador.
- Os escudos servem para bloquear tiros, tanto do jogador quanto dos alienígenas.
- Os escudos se degradam visualmente (mudando de cor de branco para chumbo) ao serem atingidos e desaparecem após sofrerem uma certa quantidade de dano.

#### 2.2.4. Sistema de Pontuação e Vidas

- O jogador começa com um número fixo de vidas.
- A pontuação é incrementada ao destruir um alienígena, com valores diferentes para cada tipo:
    - Alienígena tipo 1: 10 pontos
    - Alienígena tipo 2: 20 pontos
    - Alienígena tipo 3: 40 pontos
    - Alienígena especial: Pontuação misteri00 pontos, o jogador ganha uma vida extra, até um máximo de 6 vidas.

- A pontuação atual é sempre visível no canto superior esquerdo da tela.

### 2.3. Fim de Jogo

- O jogo termina se osa (???)
- A cada 10uma das seguintes condições for atendida:
  - O jogador perde todas as suas vidas.
  - O bloco de alienígenas alcança a parte inferior da tela (posição do jogador).
- Após o fim do jogo, o jogador tem a opção de:
  - Salvar sua pontuação, associando-a a um apelido.
  - Voltar a jogar.
  - Retornar à tela inicial.

### 2.4. Persistência de Dados

- As informações do placar (pontuação e apelido) devem ser salvas em um arquivo de texto.

## 3. Requisitos Não-Funcionais

### 3.1. Plataforma

- O jogo deve ser uma aplicação de desktop, construída com a Uno Platform para garantir a portabilidade.

### 3.2. Tecnologia

- A linguagem de programação será C#.
- A interface do usuário (UI) será definida usando XAML.
- Nenhuma game engine externa (ex: Unity, Godot) será utilizada.

### 3.3. Som

- Cada ação significativa no jogo (disparo, destruição de inimigo, morte do jogador, etc.) deve ser acompanhada por um efeito sonoro correspondente.

## 4. Fases do Projeto

### 4.1. Entrega Intermediária (Midterm - Semana 4)

- **Escopo Reduzido:**
  - Alienígenas não precisam se mover.
  - O jogador deve ser capaz de se mover e atirar.
  - Os blocos de proteção devem ser destrutíveis pelo jogador.
  - A pontuação deve ser atualizada ao destruir um alienígena.
  - O jogo termina quando o jogador atinge 500 pontos.
  - A pontuação é mantida apenas em memória (perdida ao fechar o jogo).
  - Deve haver uma tela inicial e sons para as ações.

### 4.2. Entrega Final

- Implementação de todos os requisitos funcionais e não-funcionais descritos neste documento.

# Space Invaders

## O que é o jogo?

Space Invaders é um jogo eletrônico arcade que se tornou um marco na história dos videogames, revolucionando a indústria do entretenimento digital no final dos anos 1970. Criado pela empresa japonesa Taito em 1978 e projetado por Tomohiro Nishikado, o jogo rapidamente se transformou em um fenômeno global que definiria muitos elementos fundamentais dos jogos de nave e shoot 'em up.

## Conceito do Jogo

No jogo, o jogador controla uma nave de defesa terrestre posicionada na parte inferior da tela, movendo-se horizontalmente e disparando contra ondas de invasores alienígenas que descem gradualmente em direção à Terra. O objetivo é simples, porém desafiador: destruir todos os invasores antes que eles alcancem a superfície do planeta.

### Características Principais:

- **Jogabilidade Simples**: Movimento lateral e tiros verticais
- **Progressão de Dificuldade**: Invasores ficam mais rápidos conforme são eliminados
- **Elementos Gráficos Básicos**: Sprites pixelados que se tornaram um ícone da era dos videogames retrô
- **Mecânica de Defesa**: Uso de "barricadas" que protegem temporariamente a nave do jogador

## Explicação Detalhada do Jogo

**Objetivo Principal:** O jogador deve destruir todas as naves alienígenas que formam uma falange na parte superior da tela antes que elas consigam descer e alcançar a parte inferior, onde a nave do jogador está posicionada.

**Mecânicas de Jogo:**

1. **Controle do Jogador:**

- O jogador controla um canhão laser (nave) que só pode se mover para a esquerda e para a direita na parte inferior da tela.
- O disparo é sempre vertical, para cima. Só pode haver um tiro do jogador na tela por vez.

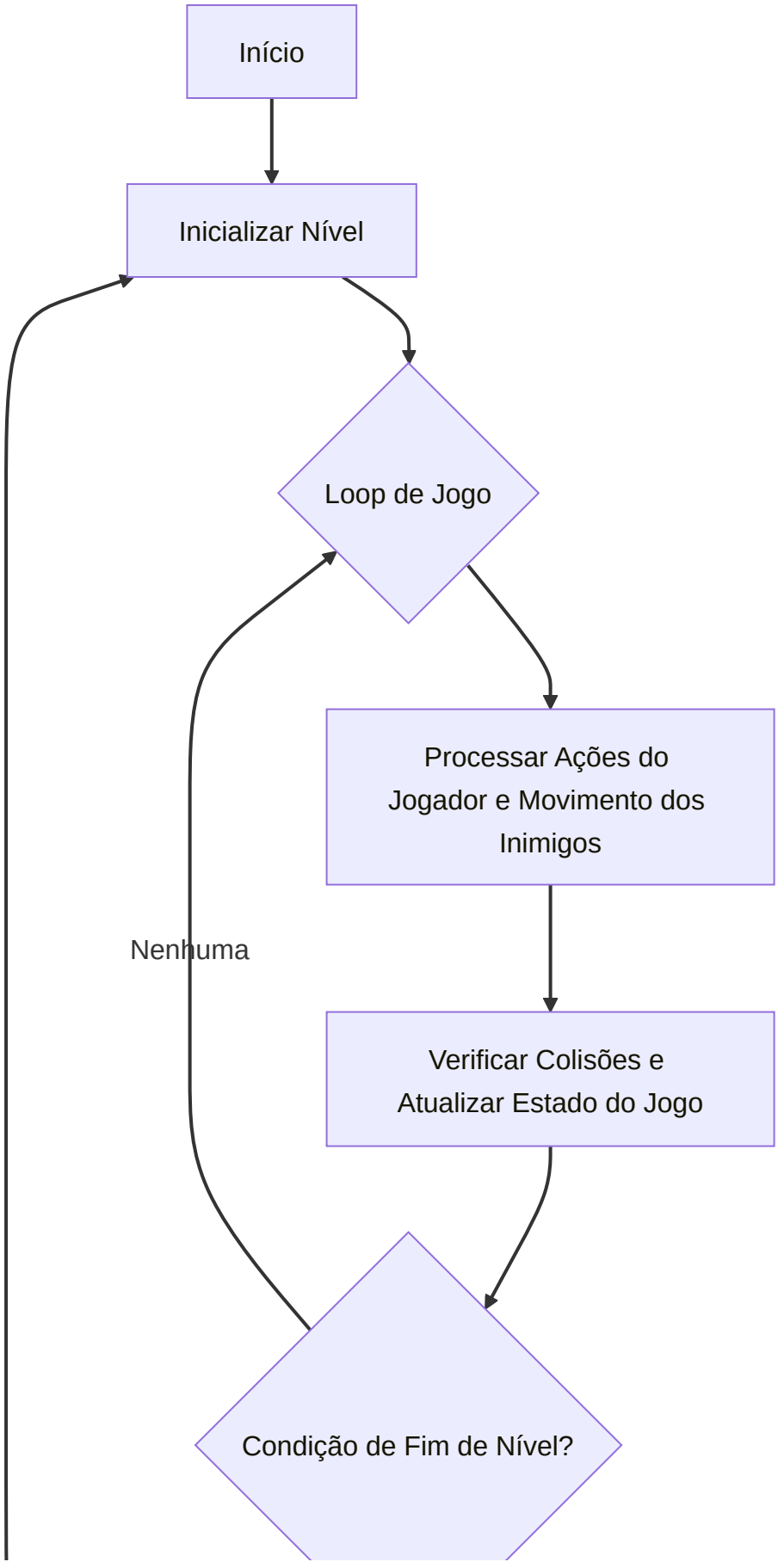
2. **Comportamento dos Invasores:**
   - Os alienígenas se movem em bloco, horizontalmente. Quando um alienígena na borda da formação atinge o limite da tela, todo o bloco desce um nível e inverte a direção do movimento.
   - À medida que o jogador destrói os invasores, a velocidade de movimento e a frequência dos disparos dos alienígenas restantes aumentam. Isso acontece porque, originalmente, o hardware tinha dificuldade para processar muitos sprites, e com menos inimigos na tela, o jogo rodava mais rápido. Este "bug" se tornou uma característica fundamental da jogabilidade.
   - Ocasionalmente, uma nave especial (um "disco voador") cruza o topo da tela, oferecendo mais pontos se for destruída.
3. **Sistema de Defesa (Barricadas):**
   - Existem quatro barricadas posicionadas acima da nave do jogador.
   - Elas servem como escudos, absorvendo tiros tanto dos inimigos quanto do próprio jogador.
   - Cada tiro destrói uma pequena parte da barricada, que vai se deteriorando até desaparecer completamente.

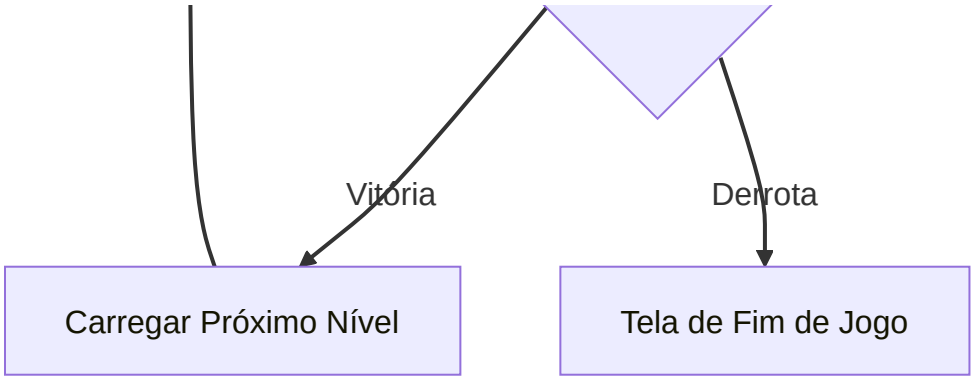
**Condições de Fim de Jogo:**

- **Vitória (Nível):** O jogador vence o nível ao destruir todos os invasores da onda. O jogo então avança para o próximo nível, geralmente com os inimigos começando em uma posição mais baixa.
- **Derrota:** O jogo termina se:
  1. Os invasores conseguirem alcançar a parte inferior da tela.

2. A nave do jogador for destruída por um tiro inimigo e não houver mais vidas restantes.

## Fluxograma do Jogo


Início
Inicializar Nível
Loop de Jogo
Processar Ações do Jogador e Movimento dos Inimigos
Verificar Colisões e Atualizar Estado do Jogo
Condição de Fim de Nível?
Nenhuma




## Impacto Cultural

Space Invaders não foi apenas um jogo, mas um fenômeno cultural que:

- Popularizou os jogos arcade
- Inspirou gerações de desenvolvedores de jogos
- Criou um novo gênero de jogos de nave e tiro
- Ajudou a estabelecer o Japão como potência na indústria de jogos eletrônicos

### Curiosidades:

1. O jogo foi tão popular no Japão que causou uma "escassez de moedas" nas máquinas arcade
2. Inspirou diversos jogos e franquias posteriores
3. Seus gráficos pixelados se tornaram um símbolo da cultura pop dos anos 80

Space Invaders representa mais do que um simples jogo: é um marco tecnológico que ajudou a definir a cultura dos videogames como a conhecemos hoje.

**Links sobre o jogo em ação:**

- Space Invaders 1978 - Arcade Gameplay (https://www.youtube.com/watch?v=MU4psw3ccUI)
- Space Invaders Creator Thinks Video Games Are Made The Wrong Way (https://www.youtube.com/watch?v=M7cQoRXjPEY)

- How Space Invaders Birthed Japanese Games (https://www.youtube.com/watch?v=Jbn8IRmSq8M)

> Jogue online: Versão online (https://freeinvaders.org/)

# Metodologia de Desenvolvimento

Para o desenvolvimento do projeto Space Invaders, foi adotada uma metodologia ágil, com forte inspiração no **Kanban**. Embora elementos do Scrum, como a divisão em "releases" e a utilização de User Stories, estivessem presentes, a natureza individual do projeto naturalmente direcionou para um fluxo de trabalho mais contínuo e visual, característico do Kanban.

## Abordagem Kanban para Projeto Individual

A escolha do Kanban se deu pela sua flexibilidade e foco na visualização do trabalho, o que é ideal para um desenvolvedor único. O fluxo de trabalho foi organizado da seguinte forma:

1. **Visualização do Trabalho**: Todas as tarefas e funcionalidades foram representadas visualmente, permitindo uma clara compreensão do que precisava ser feito, o que estava em andamento e o que já havia sido concluído.
2. **Limitação do Trabalho em Andamento (WIP)**: Embora não houvesse um limite formal de WIP como em equipes, a natureza individual impôs um limite natural, garantindo que o foco estivesse em poucas tarefas por vez para maximizar a eficiência.
3. **Gestão do Fluxo**: O progresso das tarefas foi monitorado continuamente, identificando e resolvendo gargalos rapidamente.
4. **Melhoria Contínua**: A cada ciclo de desenvolvimento, o processo foi ajustado para otimizar a entrega.

## Ferramentas e Artefatos de Suporte

Diversos artefatos e ferramentas foram utilizados para apoiar essa metodologia:

- **Product Backlog (`Product-Backlog.md` e `Product-Backlog.csv`)**: O coração do planejamento, contendo todas as funcionalidades e requisitos do projeto, priorizados e detalhados em formato de User Stories. O arquivo CSV serviu como uma fonte de dados estruturada para o backlog.

- **User Stories (`User-Stories.md`)**: Cada funcionalidade foi descrita sob a perspectiva do usuário, garantindo que o valor entregue fosse sempre o foco.
- **Epics (`Epics.md`)**: Agrupamentos maiores de User Stories, representando funcionalidades mais abrangentes do jogo.
- **Roadmap (`Roadmap.md`)**: Uma visão de alto nível do plano de desenvolvimento ao longo do tempo, indicando as principais entregas e fases do projeto.
- **Labels do GitLab (`GitLab-Labels.md`)**: Utilizadas para categorizar e organizar as tarefas (issues) no GitLab, facilitando a filtragem e o acompanhamento do progresso. As labels de `Type`, `Workflow`, `Priority` e `Component` foram cruciais para manter a organização, mesmo em um contexto individual.

Essa abordagem permitiu um desenvolvimento ágil e organizado, adaptando-se às necessidades de um projeto individual, mas mantendo a disciplina e a clareza necessárias para um projeto de Capstone.

# Labels do GitLab

Este documento descreve as labels utilizadas no GitLab para organizar e categorizar as issues e merge requests do projeto Capstone Space Invaders. Mesmo sendo um projeto individual, a utilização dessas labels ajuda na organização e no acompanhamento do progresso.

## Categorias de Labels

### 1. Tipo de Trabalho (Type)

Para classificar a natureza da tarefa.

| Title (Nome da Label) | Description (Descrição) | Color (Hex Code) |
|---|---|---|
| `Type::Feature` | Nova funcionalidade a ser implementada. | `#0066CC` (Azul) |
| `Type::Bug` | Defeito ou comportamento inesperado que precisa ser corrigido. | `#CC0000` (Vermelho Escuro) |
| `Type::Enhancement` | Melhoria em uma funcionalidade existente. | `#FF9900` (Laranja) |
| `Type::Refactor` | Reestruturação de código sem alteração de funcionalidade. | `#9933CC` (Roxo) |
| `Type::Documentation` | Tarefa relacionada à criação ou atualização de documentação. | `#009933` (Verde Escuro) |
| `Type::Technical Debt` | Dívida técnica a ser resolvida. | `#666666` (Cinza Escuro) |

### 2. Status do Fluxo (Workflow)

Para indicar o progresso da tarefa.

| Title (Nome da Label) | Description (Descrição) | Color (Hex Code) |
|---|---|---|
| `Workflow::To Do` | Tarefa pronta para ser iniciada. | `#CCCCCC` (Cinza Claro) |
| `Workflow::In Progress` | Tarefa em desenvolvimento ativo. | `#FFFF00` (Amarelo) |
| `Workflow::Done` | Tarefa concluída e verificada. | `#00CC00` (Verde Claro) |

### 3. Prioridade (Priority)

Para indicar a urgência ou importância da tarefa.

| Title (Nome da Label) | Description (Descrição) | Color (Hex Code) |
|---|---|---|
| `Priority::High` | Tarefa de alta prioridade, deve ser feita o mais rápido possível. | `#FF0000` (Vermelho) |
| `Priority::Medium` | Tarefa de prioridade média, importante mas não urgente. | `#FFCC00` (Amarelo Alaranjado) |
| `Priority::Low` | Tarefa de baixa prioridade, pode ser feita quando houver tempo. | `#3399FF` (Azul Claro) |

### 4. Componente/Área (Component)

Para identificar a parte do sistema afetada.

| Title (Nome da Label) | Description (Descrição) | Color (Hex Code) |
| --- | --- | --- |
| `Component::Gameplay` | Relacionado à lógica central do jogo e mecânicas. | `#663399` (Roxo Escuro) |
| `Component::UI` | Relacionado à interface do usuário (menus, HUD, elementos visuais). | `#FF66B2` (Rosa) |
| `Component::Audio` | Relacionado a sons, música e efeitos sonoros. | `#00CC99` (Verde Água) |
| `Component::Persistence` | Relacionado a salvamento, carregamento de dados e placares. | `#996633` (Marrom) |
| `Component::Build/Deploy` | Relacionado a configurações de build, scripts de automação, etc. | `#333333` (Preto) |

# Epics

Epics são grandes blocos de trabalho que representam funcionalidades significativas ou áreas de desenvolvimento no projeto. Eles são decompostos em User Stories menores, que são detalhadas no Product Backlog. Os épicos fornecem uma visão de alto nível do que será construído e ajudam a organizar o trabalho em temas maiores.

Para o projeto Space Invaders, os seguintes épicos foram identificados:

## 1. Jogabilidade Central (Core Gameplay)

Este épico abrange as mecânicas fundamentais do jogo, permitindo que o jogador interaja com o ambiente e os inimigos.

- **Funcionalidades Principais**: Movimento da nave do jogador, disparo de lasers, aparecimento e destruição de alienígenas, movimento dos alienígenas em bloco, aumento de dificuldade, alienígenas especiais e barreiras de proteção.

## 2. Fluxo do Jogo e Interface do Usuário (Game Flow & UI)

Este épico foca na experiência do usuário através das diferentes telas e interações visuais do jogo.

- **Funcionalidades Principais**: Tela inicial, tela de jogo (HUD com pontuação e vidas), tela de fim de jogo (com opção de salvar pontuação e reiniciar), tela de placares e tela de controles.

## 3. Persistência e Sistema de Pontuação

Este épico lida com o armazenamento de dados e a lógica de pontuação do jogo.

- **Funcionalidades Principais**: Salvamento e carregamento de pontuações, exibição de placares, cálculo de pontuação por tipo de inimigo e sistema de vidas extras.

## 4. Áudio e Recursos Visuais (Audio & Visuals)

Este épico concentra-se na imersão do jogador através de elementos sonoros e visuais.

- **Funcionalidades Principais**: Efeitos sonoros para ações do jogo (disparos, explosões, etc.), música de fundo, gerenciamento de sprites e backgrounds.

# User Stories

As User Stories detalhadas para o projeto Space Invaders podem ser encontradas no documento de Product Backlog. Este documento serve como a fonte primária para todas as funcionalidades e requisitos do projeto, apresentados em formato de User Stories com seus respectivos critérios de aceitação, labels e releases.

Para acessar o Product Backlog completo, por favor, consulte: Product Backlog (Product Backlog)

# Roadmap do Projeto

O roadmap do projeto Space Invaders descreve as principais fases de desenvolvimento e as funcionalidades esperadas em cada entrega. Ele fornece uma visão de alto nível do progresso do projeto e dos marcos importantes.

## 1. Entrega Intermediária (Midterm)

Esta fase focou na implementação de um subconjunto essencial de requisitos funcionais, estabelecendo a base para a jogabilidade principal.

### Funcionalidades Entregues:

- **Controle da Nave**: Movimento horizontal do jogador.
- **Disparo de Lasers**: Capacidade de disparar projéteis.
- **Geração de Inimigos**: Alienígenas exibidos na tela (estáticos).
- **Barreiras de Proteção**: Barreiras destrutíveis.
- **Dano em Inimigos**: Alienígenas removidos com um único tiro.
- **Exibição de Pontuação**: Pontuação visível na tela.
- **Incremento de Pontuação**: Pontuação atualizada ao destruir inimigos.
- **Tela Inicial**: Uma tela inicial simples.
- **Efeitos Sonoros**: Sons para ações principais (tiro, destruição).
- **Lógica de Jogo Simplificada**: Jogo termina ao atingir 500 pontos. Pontuação mantida em memória.

## 2. Entrega Final

Esta fase visa a implementação de todos os requisitos funcionais e não-funcionais, completando o jogo com mecânicas avançadas e persistência de dados.

### Funcionalidades a Serem Completadas/Adicionadas:

- **Movimento dos Inimigos**: Movimento em bloco, descida e inversão de direção.
- **Inimigo Especial**: Aparecimento periódico de nave vermelha com pontos extras.
- **Sistema de Vidas**: Gerenciamento de vidas do jogador.
- **Condições de Fim de Jogo**: Jogo termina ao perder todas as vidas ou alienígenas alcançarem a base.
- **Aumento de Dificuldade**: Velocidade e frequência de tiro dos inimigos aumentam.
- **Ataque Inimigo**: Alienígena de 40 pontos capaz de atirar.
- **Novas Ondas**: Geração de novas ondas de inimigos.
- **Vida Extra**: Ganho de vida extra a cada 1000 pontos.
- **Tela de Fim de Jogo**: Implementação completa com opções de salvar placar e jogar novamente.
- **Menu Completo**: Menu inicial funcional com opções de "Novo Jogo", "Placares" e "Controles".
- **Salvamento de Placar**: Pontuação salva em arquivo de texto (`scores.txt`).

# Product Backlog

Este documento detalha o Product Backlog do projeto de recriação do Space Invaders, apresentado em formato de User Stories. Cada User Story inclui sua descrição detalhada, critérios de aceitação, labels de categorização e a release à qual pertence.

## User Stories

### 1. Tela Inicial e Fluxo do Jogo

- **User Story:** Como um **jogador**, eu quero **ver uma tela inicial clara** para que eu possa **escolher entre iniciar um novo jogo, ver placares ou consultar os controles**.
  - **Descrição:** A tela inicial deve ser a primeira interface apresentada ao usuário ao iniciar o jogo. Ela deve conter botões ou opções claras para iniciar uma nova partida, acessar a tela de placares e visualizar as instruções de controle do jogo. O design deve ser intuitivo e fácil de navegar.
  - **Critérios de Aceitação:**
    - [ ] Ao iniciar o jogo, a tela inicial é exibida automaticamente.
    - [ ] A tela inicial contém um botão ou opção claramente identificada para "Iniciar Novo Jogo".
    - [ ] A tela inicial contém um botão ou opção claramente identificada para "Ver Placares".
    - [ ] A tela inicial contém um botão ou opção claramente identificada para "Ver Controles".
    - [ ] Clicar em "Iniciar Novo Jogo" inicia uma nova partida (mesmo que o jogo ainda não tenha funcionalidade completa).
    - [ ] Clicar em "Ver Placares" leva à tela de placares (mesmo que vazia).
    - [ ] Clicar em "Ver Controles" leva à tela de instruções de controle.
    - [ ] O layout da tela inicial é limpo e fácil de entender.

    - **Labels:** `Type::Feature`, `Priority::High`, `Component::UI`, `Workflow::To Do`
    - **(Release 1)**
- **User Story:** Como um **jogador**, eu quero **ver uma tela de fim de jogo** para que eu possa **salvar minha pontuação, jogar novamente ou retornar ao menu principal**.
    - **Descrição:** Após o jogo terminar (seja por perda de vidas ou alienígenas alcançando a base), uma tela de fim de jogo deve ser exibida. Esta tela deve permitir ao jogador registrar sua pontuação, iniciar uma nova partida ou voltar para a tela inicial.
    - **Critérios de Aceitação:**
        - [ ] Ao final do jogo, a tela de fim de jogo é exibida.
        - [ ] A tela de fim de jogo permite ao jogador inserir um apelido para salvar a pontuação.
        - [ ] A tela de fim de jogo contém uma opção para "Jogar Novamente".
        - [ ] A tela de fim de jogo contém uma opção para "Retornar ao Menu Principal".
    - **Labels:** `Type::Feature`, `Priority::High`, `Component::UI`, `Workflow::To Do`
    - **(Future Release)**
- **User Story:** Como um **jogador**, eu quero **que o jogo termine automaticamente** para que eu saiba **quando perdi todas as vidas ou quando os alienígenas alcançaram a base**.
    - **Descrição:** O jogo deve monitorar continuamente as condições de derrota. Assim que uma das condições for satisfeita, o jogo deve parar e transicionar para a tela de fim de jogo.
    - **Critérios de Aceitação:**
        - [ ] O jogo termina quando o jogador perde todas as vidas.
        - [ ] O jogo termina quando o bloco de alienígenas atinge a linha inferior da tela.
        - [ ] Ao término do jogo, a tela de fim de jogo é acionada.
    - **Labels:** `Type::Feature`, `Priority::High`, `Component::Gameplay`, `Workflow::To Do`

- **(Future Release)**

## 2. Jogabilidade Principal (Nave do Jogador)

- **User Story:** Como um **jogador**, eu quero **mover minha nave horizontalmente** para que eu possa **desviar de ataques e posicionar-me para atirar nos inimigos**.
  - **Descrição:** A nave do jogador deve ser controlável horizontalmente (esquerda e direita) através de inputs do teclado (ex: setas ou A/D). A movimentação deve ser fluida e responsiva, permitindo ao jogador desviar de projéteis inimigos e alinhar-se com os alvos.
  - **Critérios de Aceitação:**
    - [ ] A nave do jogador se move para a esquerda ao pressionar a tecla designada (ex: seta esquerda ou 'A').
    - [ ] A nave do jogador se move para a direita ao pressionar a tecla designada (ex: seta direita ou 'D').
    - [ ] A nave do jogador não pode sair dos limites horizontais da tela (paredes invisíveis).
    - [ ] A movimentação da nave é suave e sem travamentos.
  - **Labels:** `Type::Feature`, `Priority::High`, `Component::Gameplay`, `Workflow::To Do`
  - **(Release 1)**
- **User Story:** Como um **jogador**, eu quero **disparar lasers da minha nave** para que eu possa **destruir os alienígenas**.
  - **Descrição:** O jogador deve ser capaz de disparar um projétil (laser) verticalmente para cima a partir da nave, utilizando um input do teclado (ex: barra de espaço).
  - **Critérios de Aceitação:**
    - [ ] Ao pressionar a tecla de disparo, um laser é gerado na posição da nave e se move para cima.
    - [ ] O laser desaparece ao atingir o topo da tela ou um inimigo/barreira.

- **Labels:** `Type::Feature`, `Priority::High`, `Component::Gameplay`, `Workflow::To Do`
- **(Release 1)**

- **User Story:** Como um **jogador**, eu quero **que meu laser recarregue após cada disparo** para que eu possa **gerenciar meus ataques de forma estratégica**.
  - **Descrição:** O jogador só deve ser capaz de disparar um novo laser após o laser anterior ter saído da tela ou atingido algo.
  - **Critérios de Aceitação:**
    - [ ] O jogador não pode disparar um novo laser enquanto um laser anterior estiver ativo na tela.
    - [ ] Um novo laser pode ser disparado assim que o laser anterior desaparecer.
  - **Labels:** `Type::Feature`, `Priority::High`, `Component::Gameplay`, `Workflow::To Do`
  - **(Future Release)**

## 3. Inimigos e Interação

- **User Story:** Como um **jogador**, eu quero **ver os alienígenas aparecerem na tela** para que eu saiba **quais inimigos preciso destruir**.
  - **Descrição:** Múltiplas ondas de alienígenas devem ser posicionadas na parte superior da tela. Para a Release 1, eles podem permanecer estáticos.
  - **Critérios de Aceitação:**
    - [ ] Um conjunto de alienígenas (diferentes tipos) é exibido na parte superior da tela ao iniciar o jogo.
    - [ ] Os alienígenas são visíveis e distinguíveis.
  - **Labels:** `Type::Feature`, `Priority::High`, `Component::Gameplay`, `Workflow::To Do`
  - **(Release 1)**
- **User Story:** Como um **jogador**, eu quero **destruir os alienígenas com um único tiro** para que eu possa **limpar a tela e avançar no jogo**.

    - **Descrição:** Quando um laser do jogador atinge um alienígena, o alienígena deve ser removido da tela.
    - **Critérios de Aceitação:**
        - [ ] Um alienígena desaparece da tela ao ser atingido por um laser do jogador.
        - [ ] O laser do jogador desaparece ao atingir um alienígena.
    - **Labels:** `Type::Feature`, `Priority::High`, `Component::Gameplay`, `Workflow::To Do`
    - **(Release 1)**
- **User Story:** Como um **jogador**, eu quero **que os alienígenas se movam em bloco** para que o jogo **simule o comportamento clássico do Space Invaders**.
    - **Descrição:** Os alienígenas devem se mover em um bloco coeso, da esquerda para a direita, descendo uma linha e invertendo a direção ao atingir a borda da tela.
    - **Critérios de Aceitação:**
        - [ ] Os alienígenas se movem horizontalmente em conjunto.
        - [ ] Ao atingir a borda da tela, o bloco de alienígenas desce uma linha.
        - [ ] A direção do movimento horizontal do bloco é invertida após descer.
    - **Labels:** `Type::Feature`, `Priority::Medium`, `Component::Gameplay`, `Workflow::To Do`
    - **(Future Release)**
- **User Story:** Como um **jogador**, eu quero **que a dificuldade do jogo aumente gradualmente** para que eu **seja desafiado à medida que avanço**.
    - **Descrição:** A velocidade de movimento dos alienígenas e a frequência de seus disparos devem aumentar a cada vez que o bloco desce.
    - **Critérios de Aceitação:**
        - [ ] A velocidade de movimento dos alienígenas aumenta após cada descida do bloco.
        - [ ] A frequência de disparo dos alienígenas aumenta após cada descida do bloco.

- **Labels:** `Type::Feature`, `Priority::Medium`, `Component::Gameplay`, `Workflow::To Do`
- **(Future Release)**

- **User Story:** Como um **jogador**, eu quero **que apenas certos alienígenas atirem** para que eu possa **identificar as ameaças principais**.
  - **Descrição:** Apenas o alienígena de 40 pontos (Tipo 3) deve ser capaz de disparar projéteis.
  - **Critérios de Aceitação:**
    - [ ] Somente alienígenas do Tipo 3 disparam projéteis.
    - [ ] Projéteis inimigos se movem para baixo.
  - **Labels:** `Type::Feature`, `Priority::Medium`, `Component::Gameplay`, `Workflow::To Do`
  - **(Future Release)**
- **User Story:** Como um **jogador**, eu quero **ver um alienígena especial aparecer periodicamente** para que eu possa **ter a chance de ganhar pontos extras**.
  - **Descrição:** Um alienígena vermelho especial deve aparecer periodicamente no topo da tela, movendo-se horizontalmente e oferecendo uma pontuação variável ao ser destruído.
  - **Critérios de Aceitação:**
    - [ ] Um alienígena especial aparece aleatoriamente no topo da tela.
    - [ ] O alienígena especial se move horizontalmente.
    - [ ] Destruir o alienígena especial concede uma pontuação variável.
  - **Labels:** `Type::Feature`, `Priority::Medium`, `Component::Gameplay`, `Workflow::To Do`
  - **(Future Release)**
- **User Story:** Como um **jogador**, eu quero **que os alienígenas não colidam com as barreiras** para que a **mecânica de proteção funcione corretamente**.
  - **Descrição:** Os alienígenas devem passar por cima ou desviar das barreiras de proteção, sem interagir com elas diretamente.

- **Critérios de Aceitação:**
  - [ ] Alienígenas não colidem com as barreiras de proteção.
  - [ ] Alienígenas não são impedidos de se mover pelas barreiras.
- **Labels:** `Type::Feature`, `Priority::Low`, `Component::Gameplay`, `Workflow::To Do`
- **(Future Release)**

- **User Story:** Como um **jogador**, eu quero **que novas ondas de inimigos apareçam** para que o jogo **continue após eu destruir uma onda completa**.
  - **Descrição:** Após todos os alienígenas de uma onda serem destruídos, uma nova onda deve ser gerada, potencialmente com dificuldade aumentada.
  - **Critérios de Aceitação:**
    - [ ] Uma nova onda de alienígenas é gerada após a destruição da onda anterior.
    - [ ] A nova onda pode apresentar maior dificuldade (ex: mais alienígenas, maior velocidade inicial).
  - **Labels:** `Type::Feature`, `Priority::High`, `Component::Gameplay`, `Workflow::To Do`
  - **(Future Release)**

## 4. Barreiras de Proteção

- **User Story:** Como um **jogador**, eu quero **ter barreiras de proteção na tela** para que eu possa **me proteger dos tiros inimigos**.
  - **Descrição:** Haverá 4 blocos de proteção (escudos) localizados acima da nave do jogador. Eles devem se degradar visualmente ao serem atingidos e desaparecer após sofrerem uma certa quantidade de dano.
  - **Critérios de Aceitação:**
    - [ ] 4 barreiras de proteção são exibidas na tela.
    - [ ] As barreiras bloqueiam projéteis (do jogador e inimigos).
    - [ ] As barreiras mudam de aparência (degradam) ao serem atingidas.

    - [ ] As barreiras desaparecem após sofrerem dano suficiente.
- **Labels:** `Type::Feature`, `Priority::High`, `Component::Gameplay`, `Workflow::To Do`
- **(Release 1)**

## 5. Pontuação e Vidas

- **User Story:** Como um **jogador**, eu quero **ver minha pontuação atualizada na tela** para que eu possa **acompanhar meu desempenho**.
    - **Descrição:** A pontuação atual do jogador deve ser exibida de forma clara e visível no canto superior esquerdo da tela durante o jogo.
    - **Critérios de Aceitação:**
        - [ ] A pontuação é exibida no canto superior esquerdo da tela.
        - [ ] A pontuação é atualizada em tempo real.
    - **Labels:** `Type::Feature`, `Priority::High`, `Component::UI`, `Workflow::To Do`
    - **(Release 1)**
- **User Story:** Como um **jogador**, eu quero **ganhar pontos ao destruir inimigos** para que eu possa **aumentar minha pontuação total**.
    - **Descrição:** A pontuação do jogador deve ser incrementada ao destruir um alienígena, com valores diferentes para cada tipo de inimigo.
    - **Critérios de Aceitação:**
        - [ ] Destruir um alienígena Tipo 1 adiciona 10 pontos.
        - [ ] Destruir um alienígena Tipo 2 adiciona 20 pontos.
        - [ ] Destruir um alienígena Tipo 3 adiciona 40 pontos.
        - [ ] Destruir o alienígena especial adiciona uma pontuação variável (50-250).
    - **Labels:** `Type::Feature`, `Priority::High`, `Component::Gameplay`, `Workflow::To Do`
    - **(Release 1)**

- **User Story:** Como um **jogador**, eu quero **ter um sistema de vidas** para que eu possa **saber quantas vezes posso ser atingido antes do fim do jogo**.
  - **Descrição:** O jogador começa com um número fixo de vidas (ex: 3) e perde uma vida ao ser atingido por um projétil inimigo ou colidir com um alienígena.
  - **Critérios de Aceitação:**
    - [ ] O jogador começa com 3 vidas.
    - [ ] Uma vida é perdida quando o jogador é atingido por um projétil inimigo.
    - [ ] Uma vida é perdida quando o jogador colide com um alienígena.
    - [ ] O número de vidas restantes é exibido na tela.
  - **Labels:** `Type::Feature`, `Priority::High`, `Component::Gameplay`, `Workflow::To Do`
  - **(Future Release)**
- **User Story:** Como um **jogador**, eu quero **ganhar uma vida extra ao atingir uma certa pontuação** para que eu **seja recompensado por um bom desempenho**.
  - **Descrição:** O jogador deve ganhar uma vida extra a cada 1000 pontos, até um máximo de 6 vidas.
  - **Critérios de Aceitação:**
    - [ ] Uma vida extra é concedida a cada 1000 pontos.
    - [ ] O número máximo de vidas é 6.
  - **Labels:** `Type::Feature`, `Priority::Medium`, `Component::Gameplay`, `Workflow::To Do`
  - **(Future Release)**

## 6. Áudio e Persistência

- **User Story:** Como um **jogador**, eu quero **ouvir efeitos sonoros para as ações do jogo** para que a **experiência de jogo seja mais imersiva e responsiva**.
  - **Descrição:** Cada ação significativa no jogo (disparo do jogador, destruição de inimigo, morte do jogador, etc.) deve ser acompanhada por um efeito sonoro

correspondente.

- **Critérios de Aceitação:**
  - [ ] Um som é reproduzido ao disparar um laser.
  - [ ] Um som é reproduzido ao destruir um inimigo.
  - [ ] Um som é reproduzido ao ser atingido.
  - [ ] Outros sons relevantes (ex: fim de jogo) são reproduzidos.
- **Labels:** `Type::Feature`, `Priority::Medium`, `Component::Audio`, `Workflow::To Do`
- **(Release 1)**

- **User Story:** Como um **jogador**, eu quero **que minhas pontuações mais altas sejam salvas** para que eu possa **competir comigo mesmo e com outros (hipoteticamente)**.
  - **Descrição:** As informações do placar (pontuação e apelido) devem ser salvas em um arquivo de texto para persistência entre as sessões de jogo.
  - **Critérios de Aceitação:**
    - [ ] As pontuações são salvas em um arquivo de texto.
    - [ ] As pontuações salvas podem ser carregadas e exibidas na tela de placares.
    - [ ] O arquivo de placares é atualizado corretamente com novas pontuações.
  - **Labels:** `Type::Feature`, `Priority::High`, `Component::Persistence`, `Workflow::To Do`
  - **(Future Release)**

# Engenharia de Requisitos

## Requisitos Funcionais

| ID | Requisito | Descrição | Prioridade |
|---|---|---|---|
| RF 01 | Controle da Nave | O sistema deve permitir que o usuário controle uma nave, movimentando-a horizontalmente com as setas direcionais. | Alta |
| RF 02 | Disparo de Lasers | O usuário deve conseguir disparar lasers verticalmente com a barra de espaço. | Alta |
| RF 03 | Pontuação por Inimigo | O sistema deve atribuir pontos ao usuário quando um laser acerta um inimigo, com base na tabela de inimigos ("Tabela de Inimigos" in "Engenharia de Requisitos"). | Alta |
| RF 04 | Geração de Inimigos | Inimigos (tipos 1, 2, 3) devem aparecer na tela e se mover sem sair do quadro de jogo. | Alta |
| RF 05 | Inimigo Especial | Um alienígena vermelho especial deve cruzar a tela periodicamente, oferecendo pontos variáveis. | Média |
| RF 06 | Sistema de Vidas | O usuário começa com 3 vidas e perde uma ao ser atingido ou tocar em um alienígena. | Alta |
| RF 07 | Condição de Fim de Jogo | O jogo termina se o jogador perder todas as vidas ou se os alienígenas alcançarem a base. | Alta |
| RF 08 | Barreiras de Proteção | O sistema deve ter 4 blocos de proteção que se degradam com o dano. | Alta |
| RF 09 | Aumento de Dificulda | A velocidade de movimento e de tiro dos alienígenas deve aumentar conforme eles descem. | Média |

| | | | |
|---|---|---|---|
| | de | | |
| RF 10 | Ataque Inimigo | Apenas o tipo de alienígena que vale 40 pontos pode atirar. | Média |
| RF 11 | Recarga de Tiro | O usuário só pode disparar novamente após o tiro anterior atingir um alvo ou sair da tela. | Alta |
| RF 12 | Comportamento dos Inimigos | Os alienígenas não devem colidir com as barreiras de proteção. | Baixa |
| RF 13 | Dano em Inimigos | O usuário deve conseguir destruir naves alienígenas com um único tiro. | Alta |
| RF 14 | Novas Ondas | Uma nova onda de inimigos deve aparecer após a anterior ser destruída, com dificuldade aumentada. | Alta |
| RF 15 | Vida Extra | O jogador ganha uma vida extra a cada 1000 pontos. | Média |
| RF 16 | Exibição de Pontuação | A pontuação deve ser exibida no canto superior esquerdo da tela. | Alta |
| RF 17 | Incremento de Pontuação | A pontuação deve ser incrementada de acordo com a tabela de inimigos ("Tabela de Inimigos" in "Engenharia de Requisitos"). | Alta |
| RF 18 | Tela de Fim de Jogo | Ao final do jogo, deve ser possível salvar a pontuação com um apelido ou jogar novamente. | Alta |
| RF 19 | Tela Inicial | O jogo deve ter uma tela inicial com opções para "Novo Jogo", "Placares" e "Controles". | Alta |

| | | | |
|---|---|---|---|
| RF 20 | Efeitos Sonoros | O sistema deve ter um som representativo para cada ação principal do jogo. | Média |
| RF 21 | Salvamento de Placar | As informações do painel de pontuação devem ser salvas em um arquivo de texto. | Alta |

## Tabela de Inimigos

| Tipo de Inimigo | Pontuação | Observações |
|---|---|---|
| Inimigo Tipo 1 | 10 | Inimigo padrão, não atira. |
| Inimigo Tipo 2 | 20 | Inimigo padrão, não atira. |
| Inimigo Tipo 3 | 40 | Inimigo que pode atirar. |
| Alienígena Vermelho Especial | 50-250 (variável) | Cruza a tela periodicamente. |

## Requisitos Não Funcionais

### 1. Plataforma

- **RNF01**: O jogo deve ser uma aplicação de desktop, construída com a Uno Platform para garantir a portabilidade.

### 2. Tecnologia

- **RNF02**: A linguagem de programação será C#.
- **RNF03**: A interface do usuário (UI) será definida usando XAML.
- **RNF04**: Nenhuma game engine externa (ex: Unity, Godot) será utilizada.

### 3. Som

- **RNF05**: Cada ação significativa no jogo (disparo, destruição de inimigo, morte do jogador, etc.) deve ser acompanhada por um efeito sonoro correspondente.

# Apresentação Parcial (MidTerm)

**Projeto:** Versão Inicial do Jogo "Space Invaders"

**Data da Apresentação:** Semana 4 do Módulo

## Escopo da Entrega

Para a avaliação parcial, o projeto deve implementar um subconjunto dos requisitos funcionais, focando na mecânica central do jogo. A tabela abaixo detalha o que precisa ser entregue.

| ID do Requisito | Funcionalidade | Detalhes da Entrega Parcial |
|---|---|---|
| **RF01** | Controle da Nave | A nave do jogador deve se mover horizontalmente. |
| **RF02** | Disparo de Lasers | O jogador deve ser capaz de disparar projéteis para cima. |
| **RF04** | Geração de Inimigos | Os alienígenas devem ser exibidos na tela, mas podem permanecer estáticos (sem movimento). |
| **RF08** | Barreiras de Proteção | As barreiras devem ser exibidas e podem ser destruídas pelos tiros do jogador. |
| **RF13** | Dano em Inimigos | Alienígenas devem ser removidos ao serem atingidos por um tiro. |
| **RF16** | Exibição de Pontuação | A pontuação deve ser visível na tela. |
| **RF17** | Incremento de Pontuação | A pontuação deve ser atualizada sempre que um inimigo for destruído. |
| **RF19** | Tela Inicial | O jogo deve apresentar uma tela inicial simples. |
| **RF20** | Efeitos Sonoros | Deve haver sons para as ações principais (tiro, destruição). |

## Lógica de Jogo Simplificada

- **Condição de Vitória:** Para esta entrega, o jogo termina quando o jogador atingir **500 pontos**.

- **Armazenamento:** A pontuação é mantida apenas em memória durante a sessão de jogo.
- **Inimigos:** O movimento complexo dos inimigos (descida, aumento de velocidade) não é necessário nesta fase.

# Entrega Final

**Projeto:** Versão Completa do Jogo "Space Invaders"

**Data da Apresentação:** Semana 7 do Módulo

## Escopo da Entrega

Para a avaliação final, o projeto deve implementar **todos os requisitos funcionais** definidos na documentação. Isso inclui completar as funcionalidades da entrega parcial e adicionar as mecânicas avançadas listadas abaixo.

### Funcionalidades a Serem Completadas ou Adicionadas

| ID do Requisito | Funcionalidade | Detalhes da Entrega Final |
|---|---|---|
| **RF04** | Movimento dos Inimigos | Implementar o movimento em bloco, com descida e inversão de direção nas bordas. |
| **RF05** | Inimigo Especial | A nave vermelha deve aparecer periodicamente no topo da tela. |
| **RF06** | Sistema de Vidas | O jogador deve começar com 3 vidas e perdê-las ao ser atingido. |
| **RF07** | Condições de Fim de Jogo | O jogo deve terminar se as vidas acabarem ou se os inimigos alcançarem a base. |
| **RF09** | Aumento de Dificuldade | A velocidade dos inimigos deve aumentar conforme o jogo avança. |
| **RF10** | Ataque Inimigo | O inimigo de 40 pontos deve ser capaz de atirar no jogador. |
| **RF14** | Novas Ondas | O jogo deve gerar uma nova onda de inimigos após a anterior ser destruída. |
| **RF15** | Vida Extra | O jogador deve ganhar uma vida extra a cada 1000 pontos. |
| **RF18** | Tela de Fim de Jogo | Implementar a tela de "Game Over" com opções para salvar o placar e jogar novamente. |
| **RF19** | Menu Completo | O menu inicial deve ser funcional, com opções para "Novo Jogo", "Placares" e "Controles". |

| | | |
|---|---|---|
| **RF21** | Salvamento de Placar | A pontuação deve ser salva em um arquivo de texto (`scores.txt`). |

### Requisitos Já Entregues (MidTerm)

As seguintes funcionalidades, já implementadas na entrega parcial, devem estar completamente integradas e funcionais na versão final:

- **RF01**: Controle da Nave
- **RF02**: Disparo de Lasers
- **RF08**: Barreiras de Proteção
- **RF13**: Dano em Inimigos
- **RF16** & **RF17**: Sistema de Pontuação
- **RF20**: Efeitos Sonoros

# Arquitetura do Sistema

A arquitetura do sistema do projeto Space Invaders foi projetada para ser modular, escalável e de fácil manutenção, seguindo princípios de design que promovem a separação de responsabilidades e a clareza do código. Esta seção oferece uma visão abrangente da estrutura técnica do jogo, das tecnologias empregadas e dos padrões de design que guiaram seu desenvolvimento.

## 1. Visão Geral

O sistema é construído sobre a **Uno Platform** e **C#**, utilizando **XAML** para a interface do usuário e **Entity Framework Core** com **PostgreSQL** para persistência de dados. A aplicação segue o padrão **Model-View-ViewModel (MVVM)**, que é fundamental para a organização e testabilidade do código.

## 2. Componentes da Arquitetura

Para uma compreensão mais aprofundada dos diferentes aspectos da arquitetura, consulte os seguintes documentos:

- **Padrões de Design**: Detalha os padrões de design aplicados no projeto, como MVVM, Factory Method (implícito) e Singleton (implícito), explicando como eles contribuem para a estrutura do código.
- **Diagramas UML**: Apresenta diagramas que visualizam a estrutura do projeto, incluindo o diagrama de pacotes (PlantUML) e discute outros tipos de diagramas UML relevantes para a modelagem do sistema.
- **Modelo de Dados**: Descreve a estrutura das informações persistidas pelo jogo, com foco nas entidades principais (como `Score`) e no mecanismo de persistência (Entity Framework Core com PostgreSQL).
- **Design de Interface**: Aborda os princípios de design da interface do usuário, as tecnologias de UI (XAML) e as telas principais do jogo, garantindo uma experiência intuitiva e fiel ao original.
- **Pilha de Tecnologia**: Lista e descreve todas as tecnologias e ferramentas essenciais utilizadas no desenvolvimento do projeto, incluindo C#, Uno Platform, XAML, Entity

Framework Core, PostgreSQL, NAudio e Git.

Esta estrutura arquitetural visa proporcionar um ambiente de desenvolvimento robusto e um produto final de alta qualidade.

# Padrões de Design

Este documento descreve os principais padrões de design utilizados no desenvolvimento do projeto Space Invaders. A aplicação de padrões de design visa promover a modularidade, a manutenibilidade, a escalabilidade e a testabilidade do código.

## 1. Model-View-ViewModel (MVVM)

O padrão MVVM é amplamente utilizado em aplicações com interfaces de usuário ricas, como as desenvolvidas com XAML e Uno Platform. Ele separa a lógica de negócio e a interface do usuário em três componentes distintos:

- **Model**: Representa os dados e a lógica de negócio. No projeto Space Invaders, as classes dentro da pasta `SpaceInvaders/Models` (como `Player`, `Alien`, `Score`) atuam como Models.
- **View**: É a interface do usuário, responsável por exibir os dados e capturar as interações do usuário. No projeto, os arquivos XAML na pasta `SpaceInvaders/Presentation` (como `MainPage.xaml`, `GameOver.xaml`) são as Views.
- **ViewModel**: Atua como uma camada intermediária entre o Model e a View. Ele expõe os dados do Model de forma que a View possa facilmente se ligar a eles (data binding) e contém a lógica de apresentação e comandos que respondem às interações do usuário. As classes na pasta `SpaceInvaders/ViewModels` (como `MainViewModel`, `GameOverViewModel`) são os ViewModels.

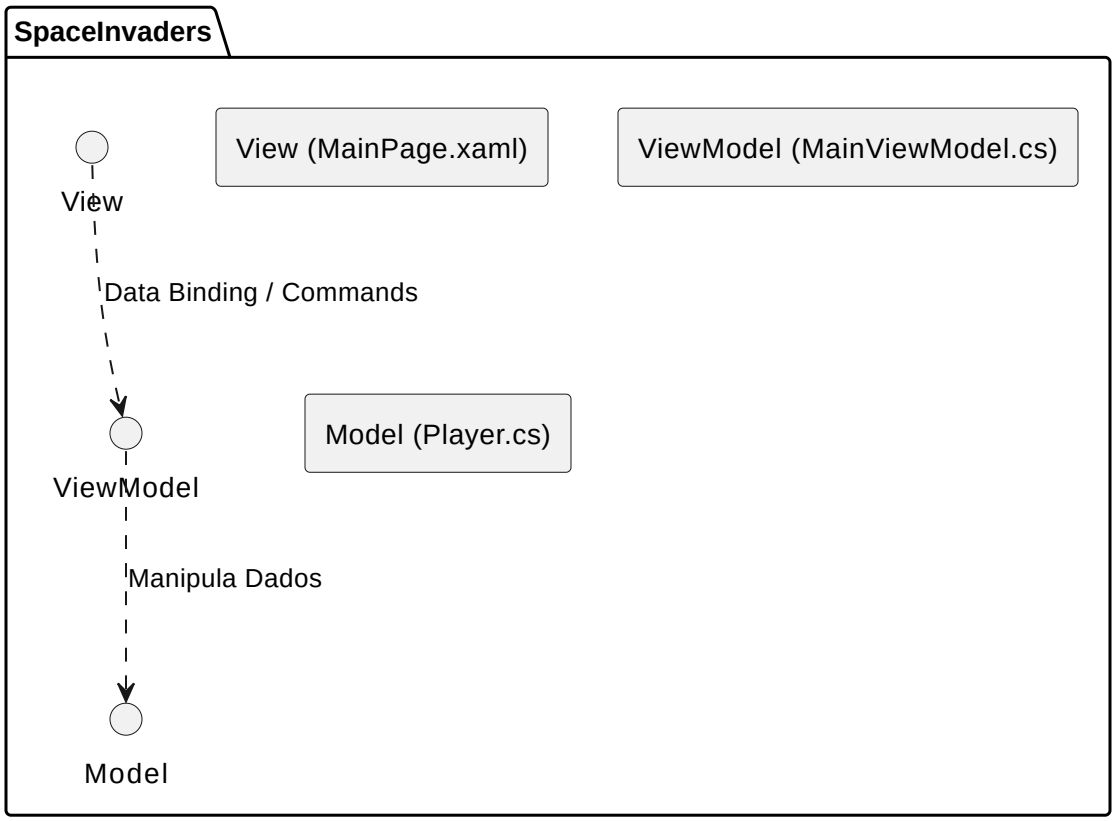
### Aplicação no Projeto Space Invaders:

O MVVM é fundamental para a organização do projeto, permitindo:

- **Separação de Responsabilidades**: A lógica da UI é separada da lógica de negócio, facilitando a manutenção e o desenvolvimento paralelo.
- **Testabilidade**: Os ViewModels podem ser testados independentemente da UI, o que melhora a qualidade do código.

- **Reutilização de Código**: A lógica de negócio nos Models e ViewModels pode ser reutilizada em diferentes Views ou plataformas.

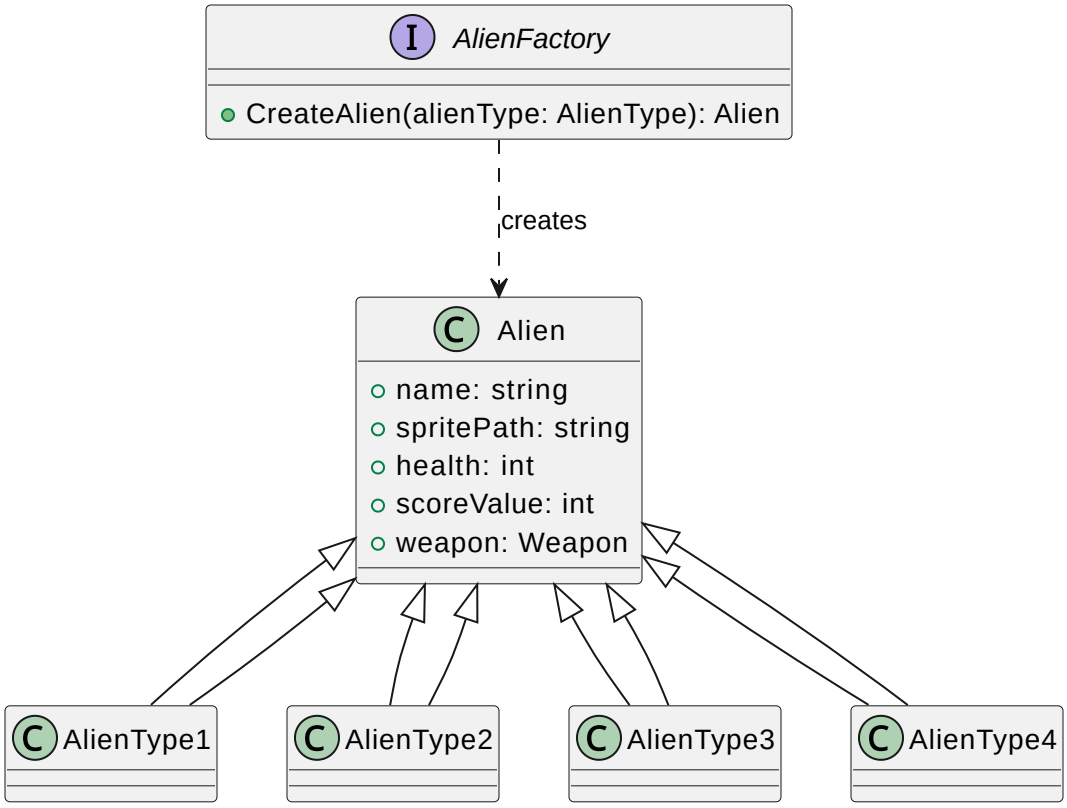
### Diagrama MVVM




## 2. Factory Method (Implícito)

Embora não explicitamente nomeado como um padrão Factory Method em uma classe dedicada, a criação de diferentes tipos de alienígenas (AlienType1, AlienType2, etc.) pode ser vista como uma aplicação implícita deste padrão. A `AlienFactory.cs` na pasta `SpaceInvaders/Factories` é um exemplo claro de como diferentes objetos (aliens) são criados com base em um tipo, abstraindo o processo de instanciação.

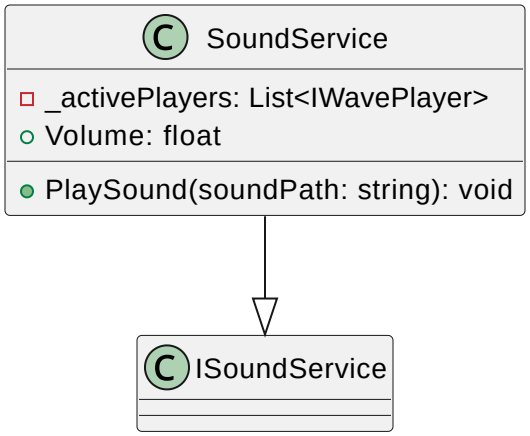
### Diagrama Factory Method



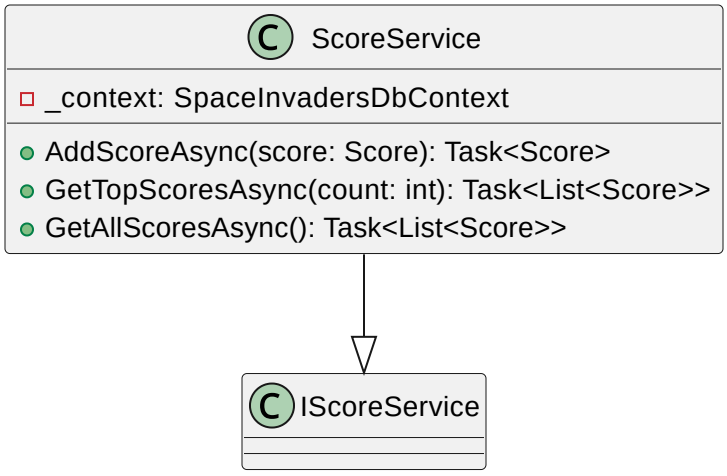

## 3. Singleton (Implícito)

O padrão Singleton pode ser observado implicitamente em serviços que precisam ter uma única instância global acessível em toda a aplicação, como o `SoundService` ou `ScoreService`. Embora não haja uma implementação estrita de Singleton com construtores privados e propriedades estáticas, a injeção de dependência (se utilizada) ou a forma como esses serviços são instanciados e acessados pode resultar em um comportamento de instância única.

### Diagrama Singleton (SoundService)




## Diagrama Singleton (ScoreService)



# Diagramas UML

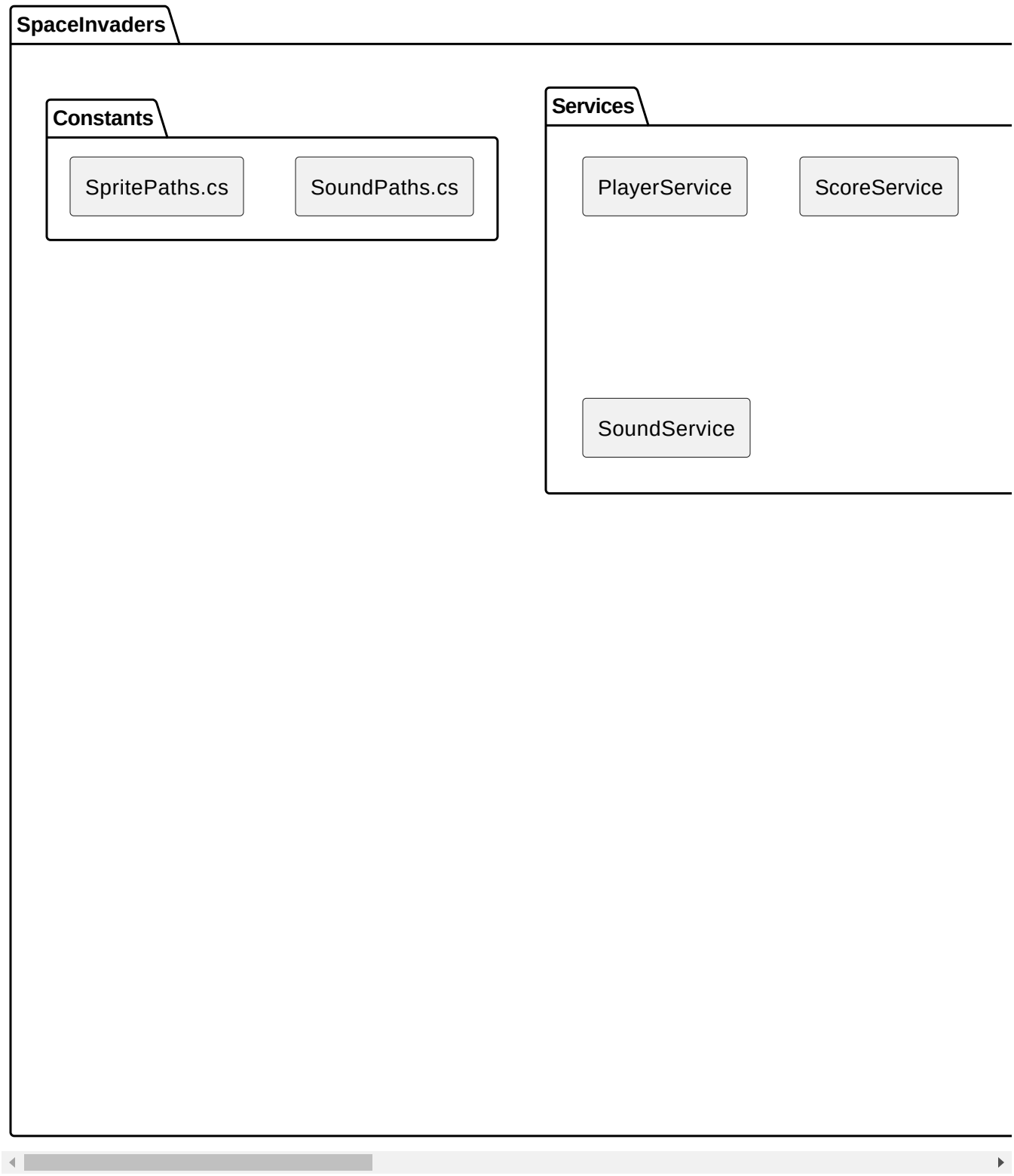
Os Diagramas de Linguagem de Modelagem Unificada (UML) são ferramentas essenciais para visualizar, especificar, construir e documentar os artefatos de um sistema de software. Eles fornecem uma representação gráfica da estrutura e do comportamento do sistema, facilitando a compreensão e a comunicação entre os membros da equipe.

## 1. Diagramas Estruturais

Os diagramas estruturais focam na visualização da estrutura estática do sistema, seus componentes e como eles se relacionam.

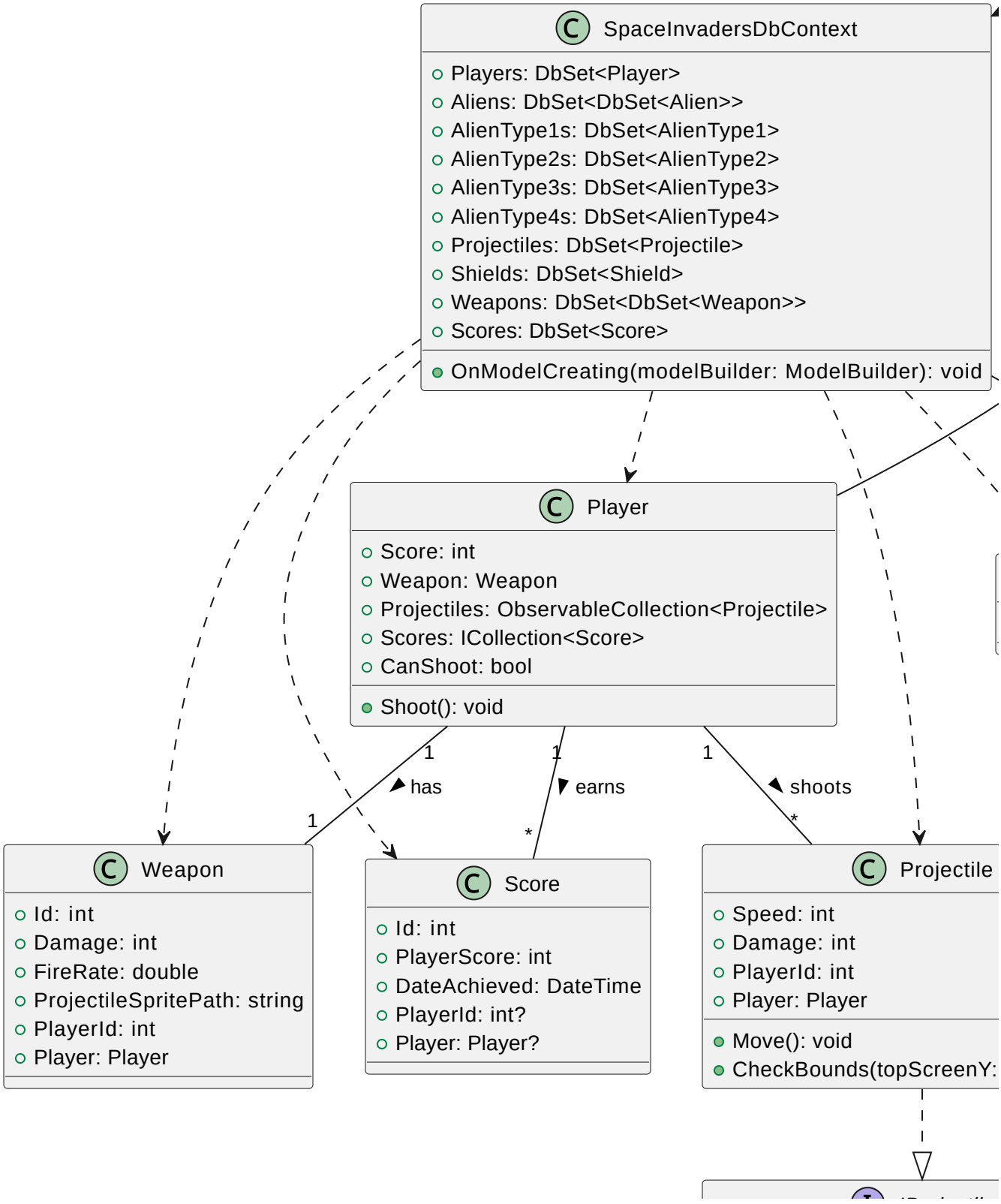
### 1.1. Diagrama de Estrutura do Projeto

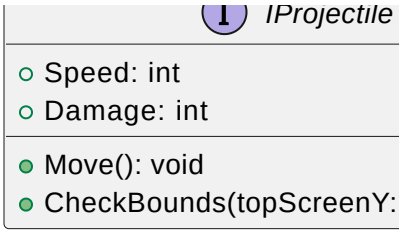
Este diagrama ilustra a organização das principais pastas do projeto Space Invaders e a relação entre elas, oferecendo uma visão de alto nível da arquitetura do código.




## 1.2. Diagrama de Classes

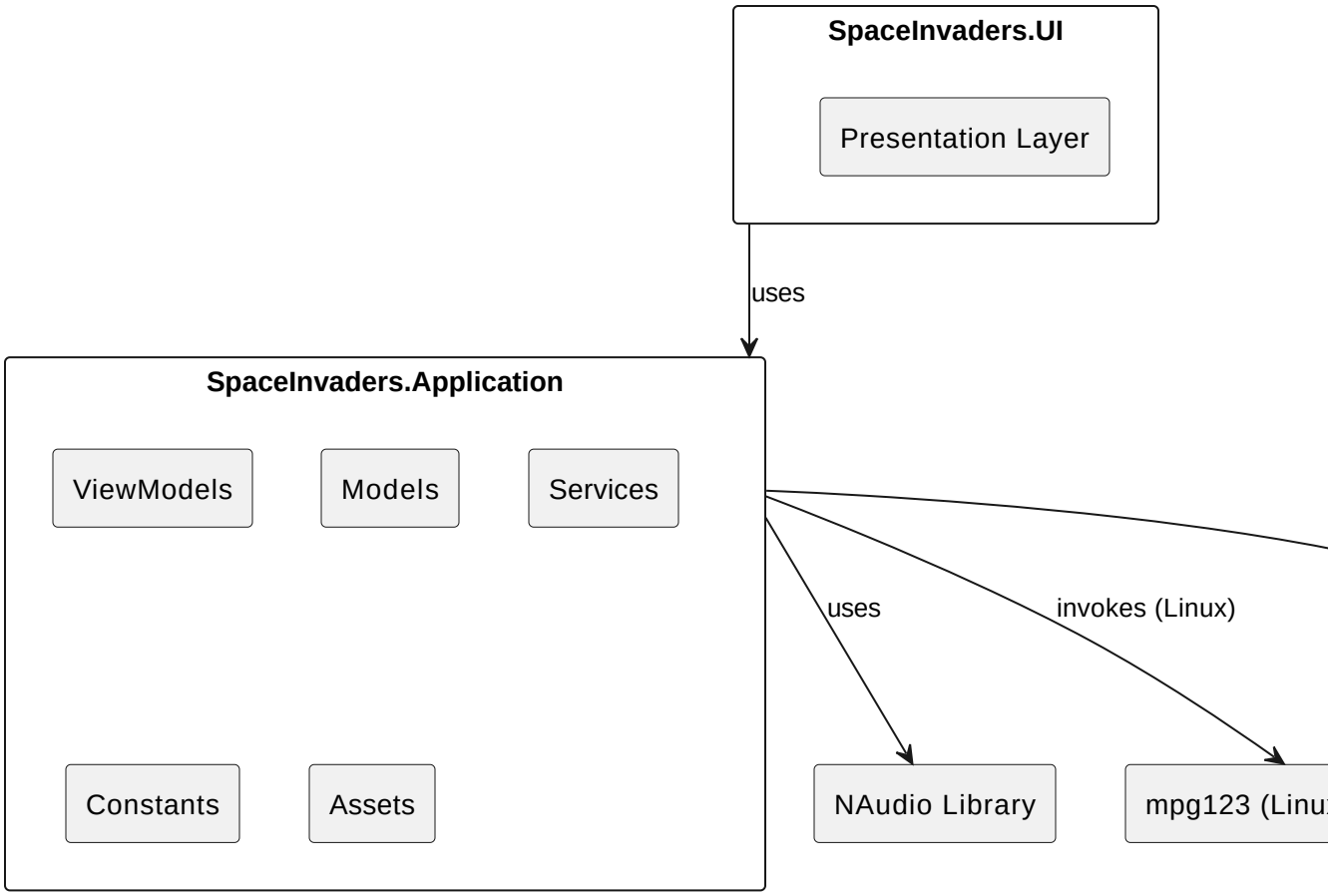
Este diagrama detalha as classes do sistema, seus atributos, métodos e relacionamentos, fornecendo uma visão estrutural do modelo de domínio do jogo.


SpaceInvadersDbContext
Players: DbSet<Player>
Aliens: DbSet<DbSet<Alien>>
AlienType1s: DbSet<AlienType1>
AlienType2s: DbSet<AlienType2>
AlienType3s: DbSet<AlienType3>
AlienType4s: DbSet<AlienType4>
Projectiles: DbSet<Projectile>
Shields: DbSet<Shield>
Weapons: DbSet<DbSet<Weapon>>
Scores: DbSet<Score>
OnModelCreating(modelBuilder: ModelBuilder): void
Player
Score: int
Weapon: Weapon
Projectiles: ObservableCollection<Projectile>
Scores: ICollection<Score>
CanShoot: bool
Shoot(): void
1
has
1
1
earns
*
1
shoots
*
Weapon
Id: int
Damage: int
FireRate: double
ProjectileSpritePath: string
PlayerId: int
Player: Player
Score
Id: int
PlayerScore: int
DateAchieved: DateTime
PlayerId: int?
Player: Player?
Projectile
Speed: int
Damage: int
PlayerId: int
Player: Player
Move(): void
CheckBounds(topScreenY:




### 1.3. Diagrama de Componentes

Este diagrama ilustra a estrutura dos componentes de software do projeto e suas dependências, fornecendo uma visão de alto nível de como as diferentes partes do sistema se interligam.

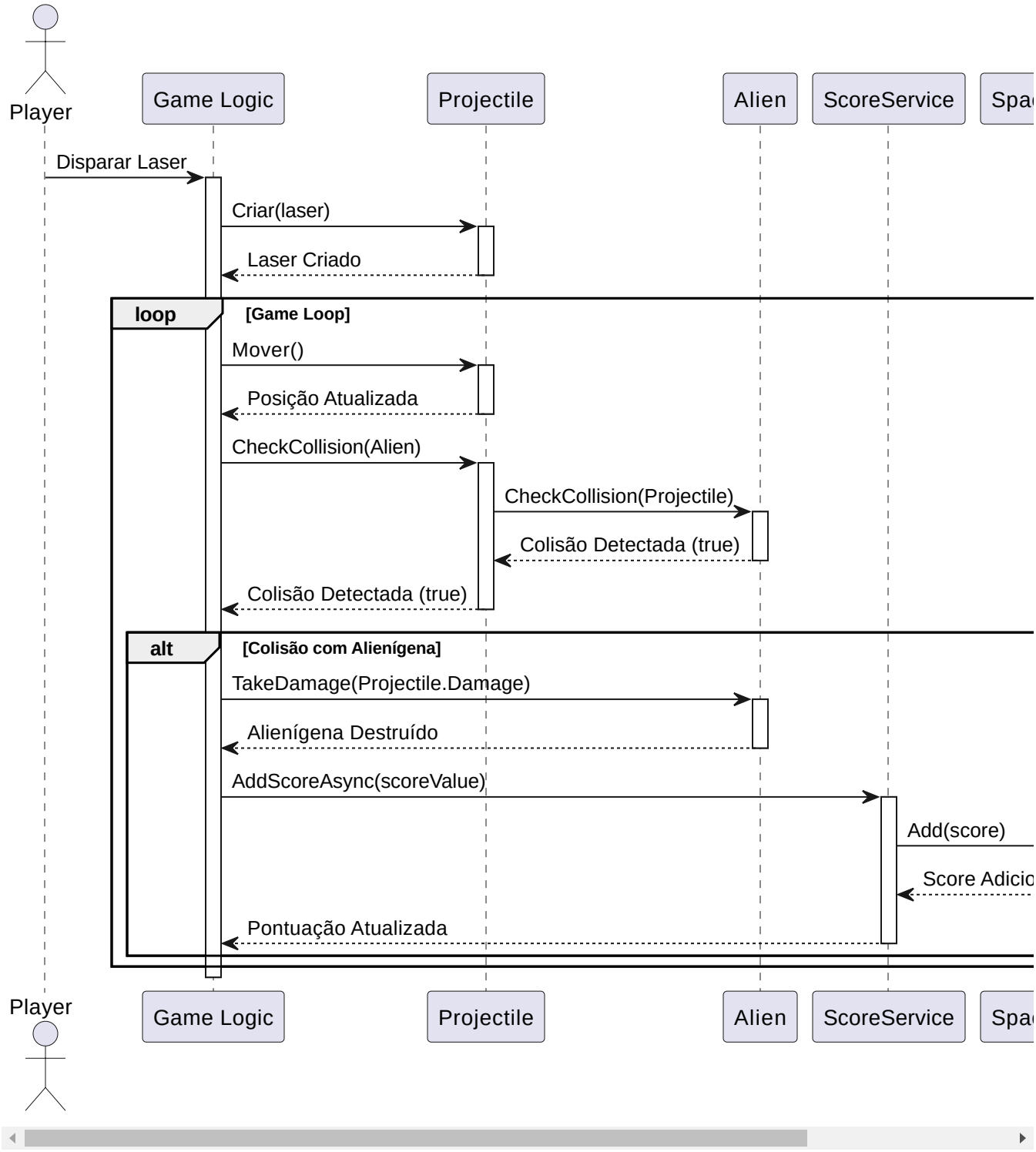



## 2. Diagramas Comportamentais

Os diagramas comportamentais focam na visualização da dinâmica do sistema, mostrando como os objetos interagem e como o sistema responde a eventos.

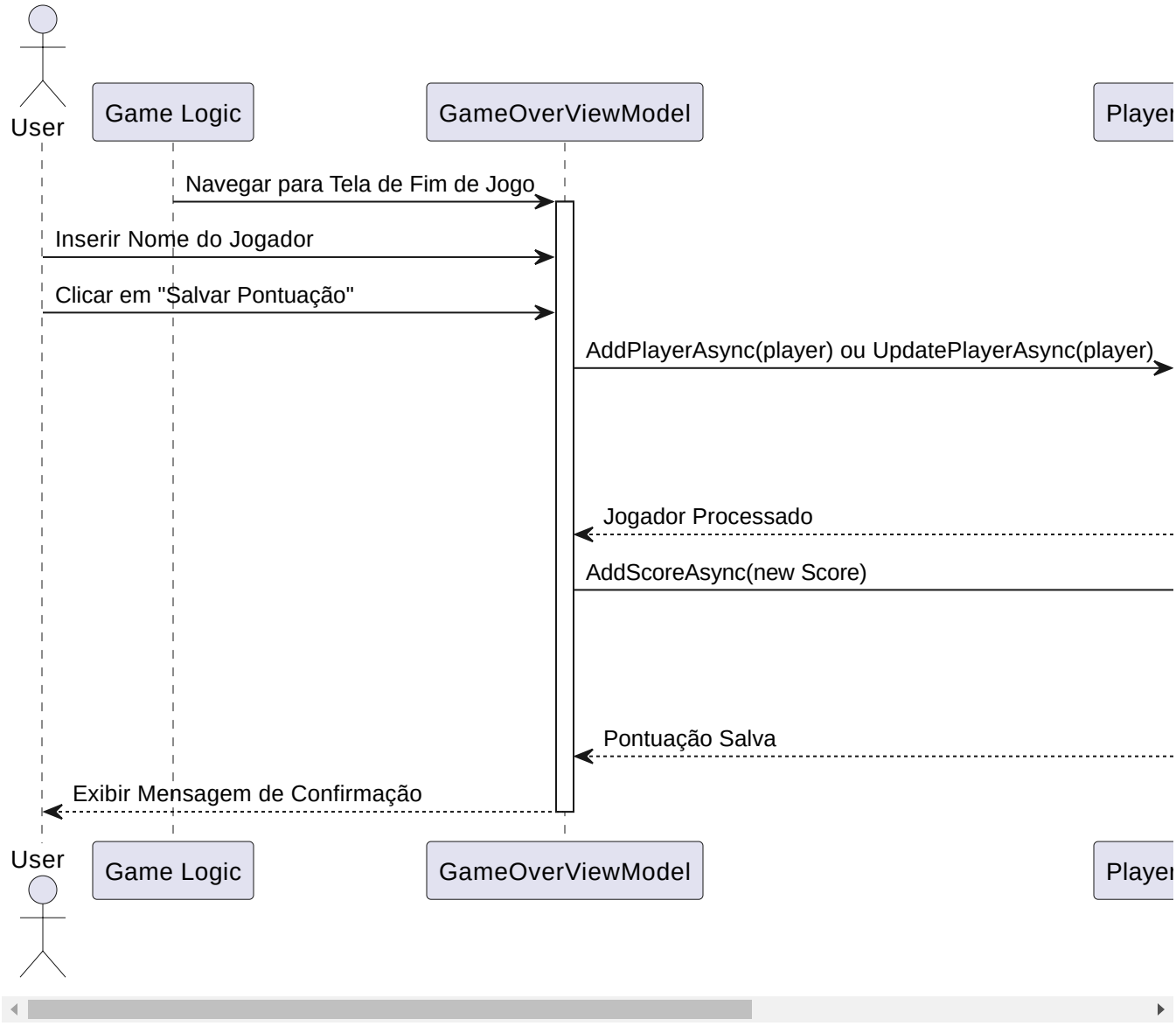
## 2.1. Diagrama de Sequência: Jogador Atira em Alienígena e Pontuação é Atualizada

Este diagrama ilustra a sequência de interações quando o jogador dispara um projétil que atinge um alienígena, resultando na atualização da pontuação.



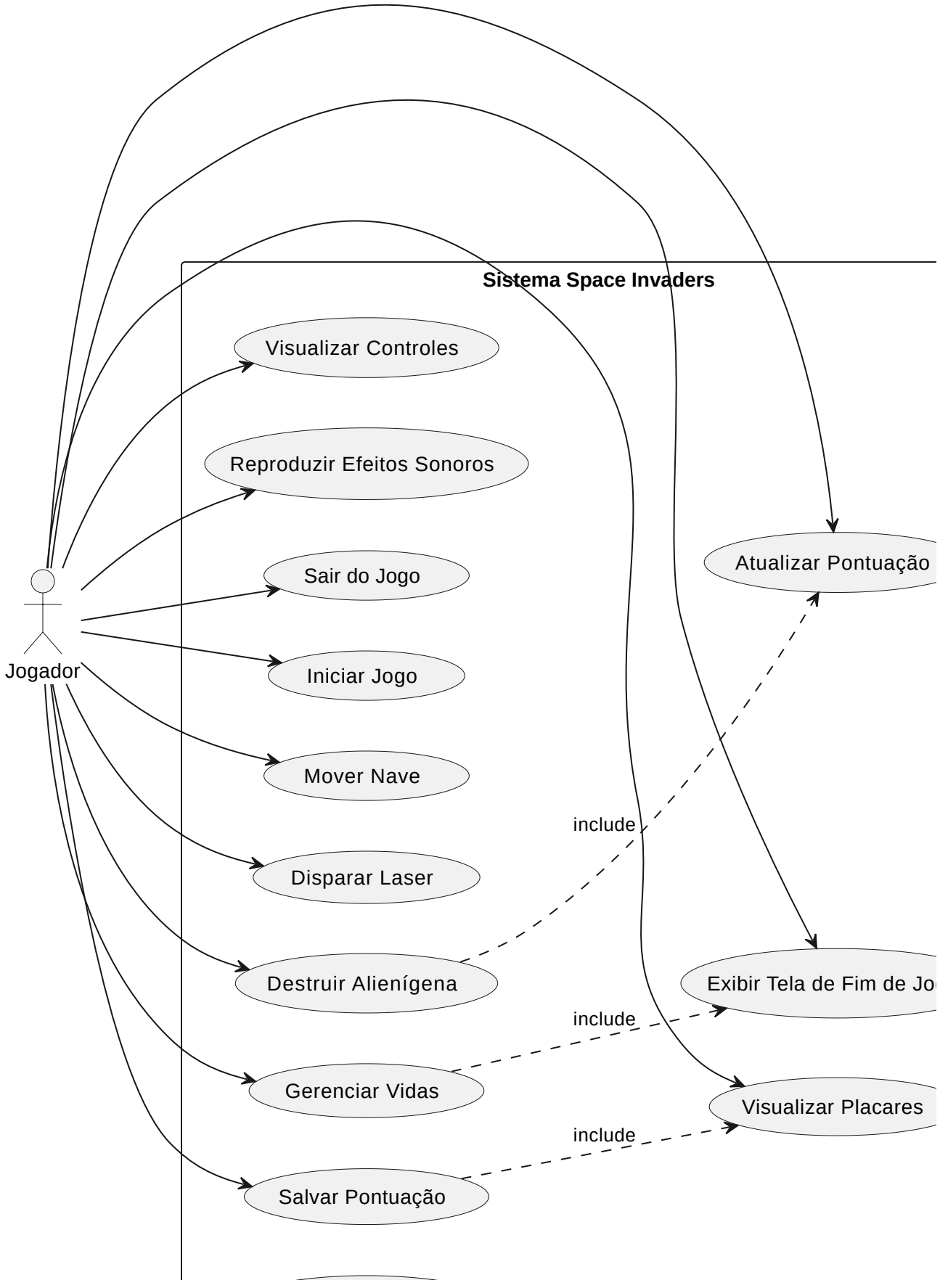

## 2.2. Diagrama de Sequência: Fim de Jogo e Salvamento de Pontuação

Este diagrama ilustra a sequência de interações que ocorrem quando o jogo termina e o jogador opta por salvar sua pontuação.



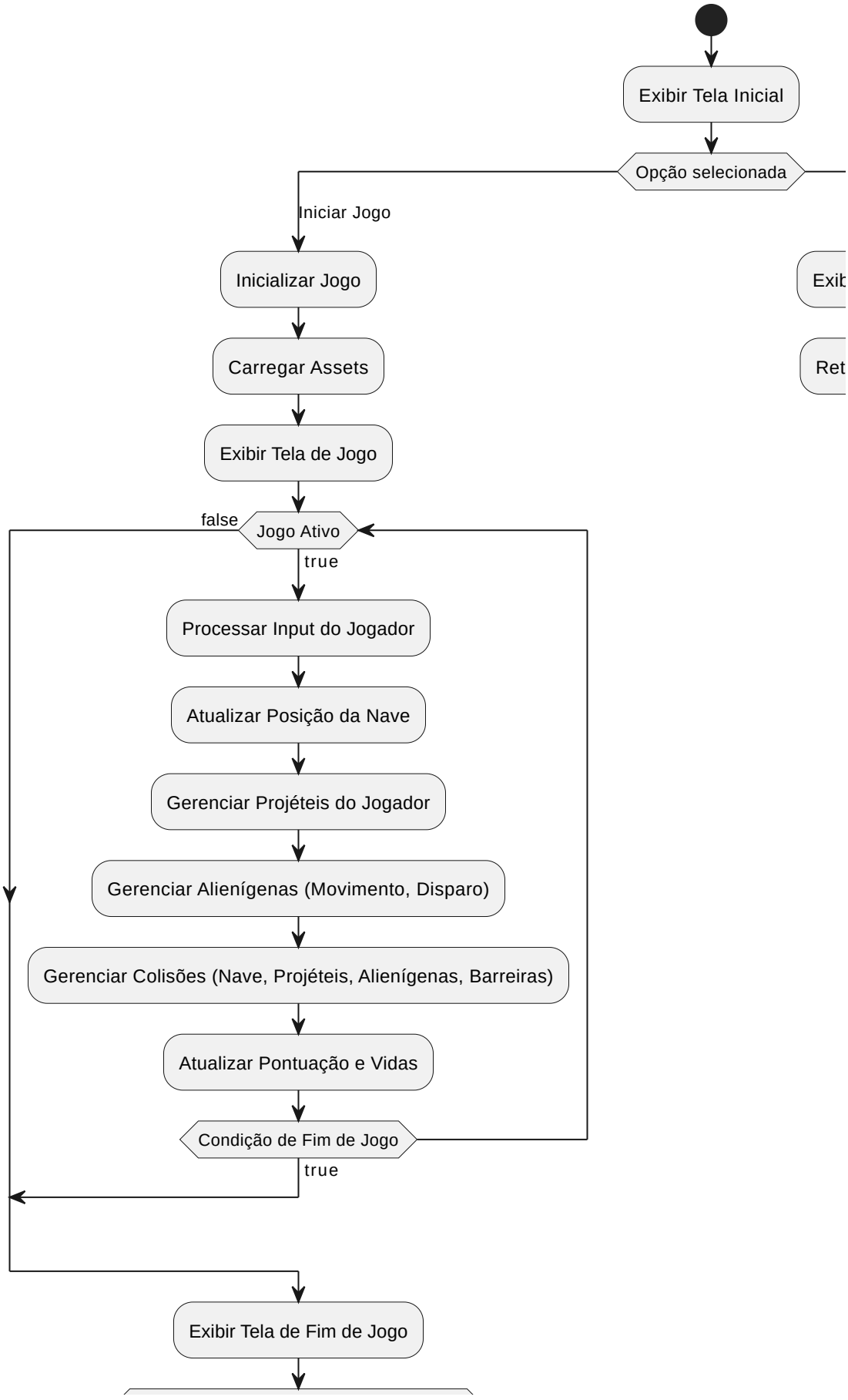

## 2.3. Diagrama de Casos de Uso

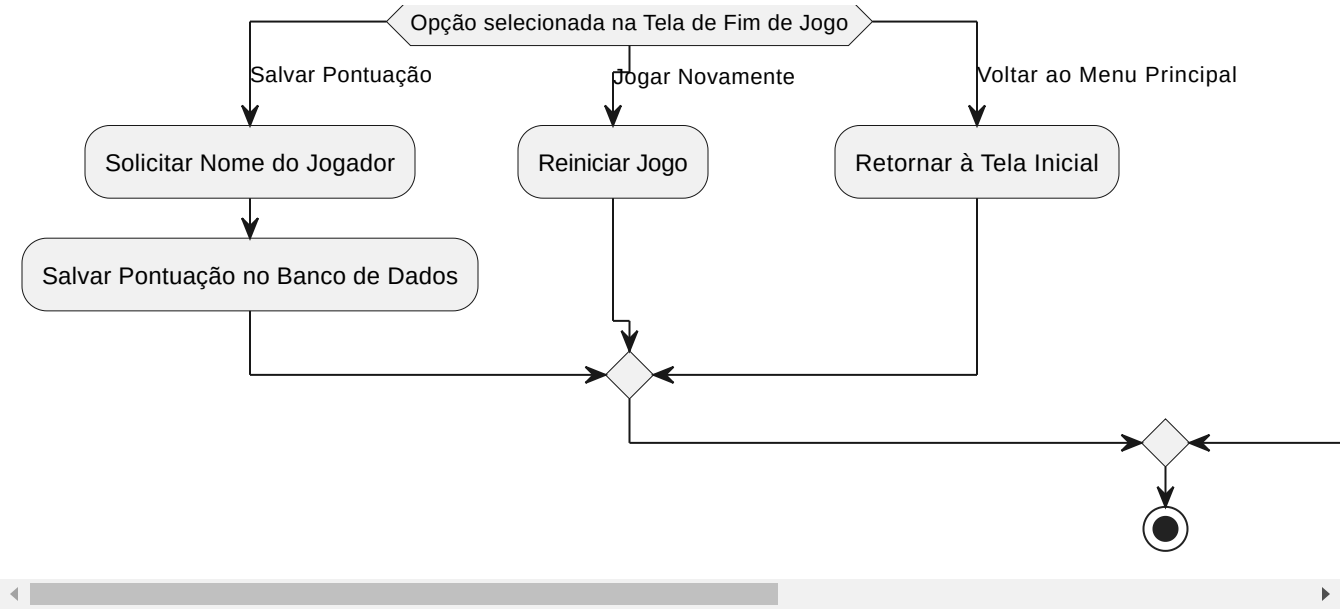
Este diagrama descreve as funcionalidades do sistema do ponto de vista do usuário, mostrando como os atores interagem com o sistema.


Sistema Space Invaders
Visualizar Controles
Reproduzir Efeitos Sonoros
Sair do Jogo
Iniciar Jogo
Mover Nave
Disparar Laser
Destruir Alienígena
Gerenciar Vidas
Salvar Pontuação
Atualizar Pontuação
Exibir Tela de Fim de Jo
Visualizar Placares
include
include
include
Jogador


## 2.4. Diagrama de Atividades: Fluxo Principal do Jogo

Este diagrama representa o fluxo principal de atividades do jogo, desde o início da partida até o seu término.


Exibir Tela Inicial
Opção selecionada
Iniciar Jogo
Inicializar Jogo
Carregar Assets
Exibir Tela de Jogo
false
Jogo Ativo
true
Processar Input do Jogador
Atualizar Posição da Nave
Gerenciar Projéteis do Jogador
Gerenciar Alienígenas (Movimento, Disparo)
Gerenciar Colisões (Nave, Projéteis, Alienígenas, Barreiras)
Atualizar Pontuação e Vidas
Condição de Fim de Jogo
true
Exibir Tela de Fim de Jogo


Opção selecionada na Tela de Fim de Jogo
Salvar Pontuação
Jogar Novamente
Voltar ao Menu Principal
Solicitar Nome do Jogador
Reiniciar Jogo
Retornar à Tela Inicial
Salvar Pontuação no Banco de Dados

# Modelo de Dados

O modelo de dados define a estrutura das informações que são gerenciadas e persistidas pelo sistema. No projeto Space Invaders, o foco principal da persistência de dados está nos placares dos jogadores.

## 1. Entidades Principais

As principais entidades que compõem o modelo de dados são representadas pelas classes `DbSet` no `SpaceInvadersDbContext`:

- **`Player`**: Representa o jogador principal, suas características e relacionamentos com outros dados.
- **`Alien`**: Classe base para os diferentes tipos de alienígenas no jogo.
- **`AlienType1`**, **`AlienType2`**, **`AlienType3`**, **`AlienType4`**: Tipos específicos de alienígenas que herdam de `Alien`.
- **`Projectile`**: Representa os projéteis disparados por jogadores ou alienígenas.
- **`Shield`**: Representa as barreiras de proteção no jogo.
- **`Weapon`**: Define as características das armas, como dano e taxa de disparo.
- **`Score`**: Representa a pontuação de um jogador em uma partida. Esta entidade armazena informações como:
  - `Id` (chave primária)
  - `PlayerScore` (pontuação alcançada)
  - `DateAchieved` (data em que a pontuação foi alcançada)
  - `PlayerId` (chave estrangeira para o jogador associado, pode ser nula)

As classes C# que representam essas entidades estão localizadas na pasta `SpaceInvaders/Models`.

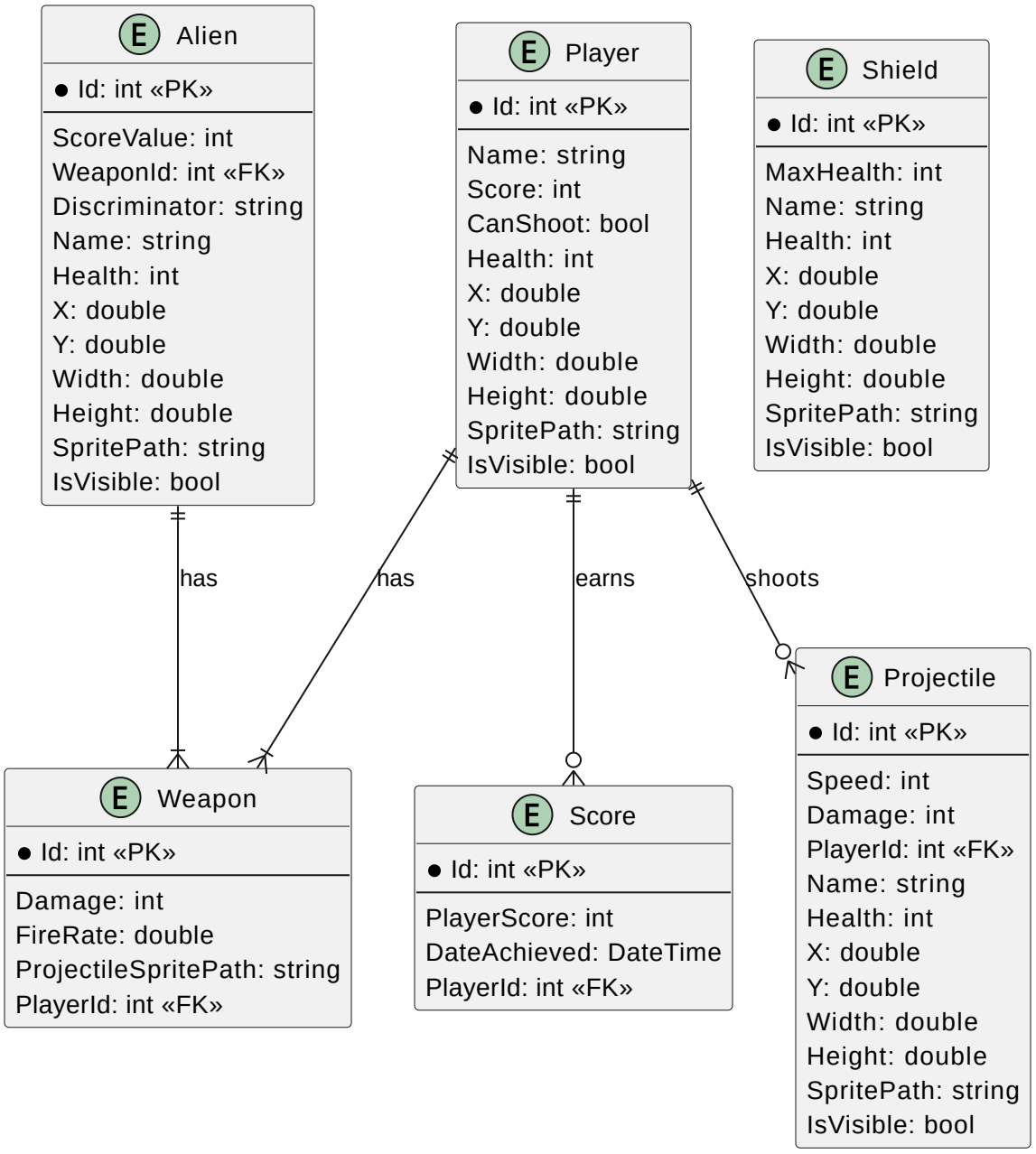
## 2. Mecanismo de Persistência

O projeto utiliza **Entity Framework Core** como ORM (Object-Relational Mapper) para interagir com o banco de dados **PostgreSQL**. Essa combinação oferece uma solução robusta e flexível para o gerenciamento de dados.

- **`SpaceInvadersDbContext`**: Esta classe, localizada na pasta `SpaceInvaders/Data`, é o contexto do banco de dados. Ela atua como uma ponte entre as entidades do modelo e o banco de dados, permitindo operações de consulta, inserção, atualização e exclusão. A configuração da conexão com o banco de dados é feita através do `appsettings.json`, utilizando o provedor `Npgsql` para PostgreSQL.
- **Migrações (Migrations)**: O Entity Framework Core utiliza migrações para gerenciar as alterações no esquema do banco de dados. As migrações são arquivos de código que descrevem como o esquema do banco de dados deve ser atualizado para refletir as mudanças no modelo de dados da aplicação. Elas estão localizadas na pasta `SpaceInvaders/Migrations`.

## 3. Diagrama de Entidade-Relacionamento (ERD)

Este diagrama ilustra as entidades principais do modelo de dados e seus relacionamentos:




## 4. Estrutura do Banco de Dados (Simplificada)

O banco de dados PostgreSQL conterá uma tabela principal para armazenar as pontuações, refletindo a estrutura da entidade `Score`.

```
CREATE TABLE "Scores" (
    "Id" INTEGER GENERATED BY DEFAULT AS IDENTITY,
    "PlayerScore" INTEGER NOT NULL,
    "DateAchieved" TIMESTAMP WITH TIME ZONE NOT NULL,
    "PlayerId" INTEGER,
```

```
    CONSTRAINT "PK_Scores" PRIMARY KEY ("Id"),
    CONSTRAINT "FK_Scores_Players_PlayerId" FOREIGN KEY
("PlayerId") REFERENCES "Players" ("Id")
);
```

Este modelo de dados simplificado é suficiente para atender aos requisitos de persistência de placares do projeto Space Invaders.

# Design de Interface

O design da interface do usuário (UI) do projeto Space Invaders foi concebido para ser intuitivo, funcional e fiel à experiência clássica do jogo, ao mesmo tempo em que incorpora as capacidades da Uno Platform e XAML.

## 1. Princípios de Design

Os seguintes princípios guiaram o desenvolvimento da UI:

- **Clareza**: As informações e opções devem ser apresentadas de forma clara e compreensível para o usuário.
- **Simplicidade**: Evitar elementos desnecessários para manter o foco na jogabilidade e nas funcionalidades essenciais.
- **Responsividade**: Embora seja uma aplicação desktop, a estrutura XAML permite uma adaptação flexível a diferentes resoluções de tela.
- **Fidelidade ao Original**: Manter a estética e a sensação do jogo Space Invaders clássico, especialmente nos elementos visuais da jogabilidade.

## 2. Tecnologia de UI: XAML

A interface do usuário é definida declarativamente utilizando **XAML (Extensible Application Markup Language)**. O XAML permite uma separação limpa entre o design visual da UI e a lógica de negócio (implementada nos ViewModels), seguindo o padrão MVVM.

Todos os arquivos XAML que definem as telas e componentes da UI estão localizados na pasta `SpaceInvaders/Presentation`.

## 3. Telas Principais da Interface

O jogo é composto por diversas telas que guiam o usuário através da experiência:

### 3.1. Tela Inicial (GameStartPage.xaml)

- **Propósito**: Ponto de entrada do jogo, oferecendo opções para o usuário.
- **Elementos**: Título do jogo, botões para "Iniciar Jogo", "Placares" e "Controles".

### 3.2. Tela de Jogo (MainPage.xaml)

- **Propósito**: Onde a ação principal do jogo acontece.
- **Elementos**: Área de jogo com a nave do jogador, alienígenas, projéteis e barreiras. HUD (Head-Up Display) exibindo a pontuação atual e o número de vidas restantes.

### 3.3. Tela de Fim de Jogo (GameOver.xaml)

- **Propósito**: Exibida após o término de uma partida.
- **Elementos**: Mensagem de "Fim de Jogo", pontuação final do jogador, campo para inserir apelido e salvar a pontuação, botões para "Jogar Novamente" e "Voltar ao Menu Principal".

### 3.4. Tela de Placares (ScorePage.xaml)

- **Propósito**: Exibir as pontuações mais altas salvas.
- **Elementos**: Lista das pontuações, com apelido do jogador e pontuação. Botão para retornar ao menu principal.

### 3.5. Tela de Controles (ControllersPage.xaml)

- **Propósito**: Informar o usuário sobre os comandos do jogo.
- **Elementos**: Descrição dos controles de movimento e disparo da nave.

## 4. Estilos e Recursos Visuais

O projeto utiliza estilos e recursos visuais definidos em arquivos XAML separados (como `AppStyles.xaml` e `GameOverStyles.xaml` na pasta `SpaceInvaders/Styles`) para garantir

consistência visual e facilitar a manutenção do design. Os assets visuais (sprites, backgrounds) estão organizados na pasta `SpaceInvaders/Assets`.

# Tecnologias

Abaixo estão os principais componentes da tecnologia stack:

## 1. Linguagem de Programação: C#

- **Descrição**: C# é uma linguagem de programação moderna, orientada a objetos e desenvolvida pela Microsoft. É amplamente utilizada para construir uma vasta gama de aplicações, incluindo desktop, web, mobile e jogos.
- **Motivo da Escolha**: Sua robustez, ecossistema maduro (.NET), forte tipagem e recursos avançados tornam-na ideal para o desenvolvimento de lógicas de jogo complexas e aplicações de alto desempenho.

## 2. Plataforma: Uno Platform

- **Descrição**: Uno Platform é uma plataforma de código aberto que permite construir aplicações nativas para WebAssembly, iOS, Android, macOS, Linux e Windows a partir de uma única base de código C# e XAML.
- **Motivo da Escolha**: Garante a portabilidade da aplicação de desktop para diferentes sistemas operacionais, além de oferecer a flexibilidade para futuras expansões para outras plataformas, se necessário.

## 3. Interface de Usuário (UI): XAML

- **Descrição**: XAML (Extensible Application Markup Language) é uma linguagem de marcação declarativa utilizada para definir interfaces de usuário em aplicações baseadas em .NET, especialmente com tecnologias como WPF, UWP e Uno Platform.
- **Motivo da Escolha**: Permite uma separação clara entre o design da interface e a lógica de negócio (seguindo o padrão MVVM), facilitando o desenvolvimento e a manutenção da UI.

## 4. Persistência de Dados: Entity Framework Core com PostgreSQL

- **Entity Framework Core (EF Core)**:
  - **Descrição**: É um ORM (Object-Relational Mapper) leve, extensível e multiplataforma da Microsoft. Ele permite que desenvolvedores .NET trabalhem com um banco de dados usando objetos .NET, eliminando a necessidade da maior parte do código de acesso a dados que eles geralmente precisariam escrever.
  - **Motivo da Escolha**: Simplifica a interação com o banco de dados, permitindo o uso de objetos C# para gerenciar os placares, além de oferecer recursos como migrações para controle de versão do esquema do banco de dados.
- **PostgreSQL**:
  - **Descrição**: É um sistema de gerenciamento de banco de dados relacional de código aberto, robusto e altamente extensível, conhecido por sua confiabilidade e conformidade com padrões.
  - **Motivo da Escolha**: Oferece um banco de dados confiável e performático para armazenar os placares do jogo, sendo uma escolha popular em ambientes de desenvolvimento e produção.

## 5. Áudio: NAudio

- **Descrição**: NAudio é uma biblioteca de áudio de código aberto para .NET, que fornece funcionalidades para reprodução, gravação e processamento de áudio.
- **Motivo da Escolha**: Permite a manipulação e reprodução de efeitos sonoros e músicas no jogo, contribuindo para uma experiência imersiva. A implementação considera a compatibilidade com diferentes sistemas operacionais, incluindo Linux (via `mpg123`).

## 6. Controle de Versão: Git

- **Descrição**: Git é um sistema de controle de versão distribuído, amplamente utilizado para rastrear mudanças no código-fonte durante o desenvolvimento de software.
- **Motivo da Escolha**: Facilita o gerenciamento do código, o histórico de alterações e a colaboração (mesmo em projetos individuais, ajuda na organização e no versionamento de entregas).

# Plano de Testes

Este documento descreve o plano de testes para o projeto Space Invaders, delineando as estratégias, tipos de testes e abordagens para garantir a qualidade e o funcionamento correto da aplicação.

## 1. Estratégia de Testes

A estratégia de testes visa cobrir as principais funcionalidades do jogo, desde a lógica de negócio até a interação com a interface do usuário. Dada a natureza do projeto, uma combinação de testes manuais e, quando aplicável, testes automatizados será utilizada.

## 2. Tipos de Testes

### 2.1. Testes Funcionais

- **Objetivo**: Verificar se as funcionalidades do jogo operam de acordo com os requisitos definidos no Product Backlog e nos Requisitos de Engenharia.
- **Cobertura**: Inclui o movimento do jogador, disparo de lasers, comportamento dos alienígenas, sistema de pontuação, gerenciamento de vidas, telas de menu, fim de jogo e placares.
- **Abordagem**: Principalmente testes manuais, simulando a interação do usuário com o jogo.

### 2.2. Testes de Unidade

- **Objetivo**: Isolar e testar componentes individuais do código (métodos, classes) para garantir que funcionem conforme o esperado.
- **Cobertura**: Lógica de negócio em `Services` (ex: `ScoreService`, `PlayerService`), modelos (`Models`), e lógica de apresentação em `ViewModels`.
- **Ferramentas Potenciais**: Frameworks de teste como NUnit ou xUnit, e bibliotecas de mocking como Moq (se a complexidade justificar).
- **Abordagem**: Testes automatizados para garantir a correção da lógica interna.

### 2.3. Testes de Integração

- **Objetivo**: Verificar a interação entre diferentes módulos ou componentes do sistema.
- **Cobertura**: Integração entre `Services` e o banco de dados (Entity Framework Core com PostgreSQL), interação entre `ViewModels` e `Models`, e o fluxo de dados entre as camadas.
- **Abordagem**: Testes automatizados para validar a comunicação entre os componentes.

### 2.4. Testes de Interface do Usuário (UI)

- **Objetivo**: Assegurar que a interface do usuário se comporta corretamente, exibe os elementos visuais como esperado e responde às interações do usuário.
- **Cobertura**: Navegação entre telas, exibição de sprites, atualização de HUD (pontuação, vidas), e responsividade dos controles.
- **Abordagem**: Principalmente testes manuais, com validação visual e de interação.

### 2.5. Testes de Áudio

- **Objetivo**: Verificar se os efeitos sonoros e músicas são reproduzidos corretamente em momentos apropriados do jogo.
- **Cobertura**: Disparos, explosões, sons de colisão, música de fundo, etc.
- **Abordagem**: Testes manuais durante a jogabilidade.

## 3. Casos de Teste (Exemplos)

Os casos de teste detalhados seriam desenvolvidos com base nos requisitos funcionais e não funcionais. Alguns exemplos incluem:

- **RF01 - Controle da Nave**: Testar o movimento da nave para a esquerda e direita, e verificar se ela não ultrapassa os limites da tela.
- **RF02 - Disparo de Lasers**: Testar o disparo do laser, sua trajetória e o desaparecimento ao atingir um alvo ou o topo da tela.

- **RF03 - Pontuação por Inimigo**: Destruir diferentes tipos de alienígenas e verificar se a pontuação é incrementada corretamente.
- **RF07 - Condição de Fim de Jogo**: Simular a perda de todas as vidas e o alcance dos alienígenas à base para verificar a transição para a tela de fim de jogo.

## 4. Ambiente de Testes

Os testes serão realizados no ambiente de desenvolvimento local, garantindo que o banco de dados PostgreSQL (via Docker) esteja em execução e que todas as dependências do projeto estejam instaladas e configuradas corretamente.

# Funcionalidades Básicas

Este documento detalha as funcionalidades básicas do projeto Space Invaders, que foram implementadas e testadas nas fases iniciais do desenvolvimento, especialmente durante a Entrega Intermediária (Midterm). Estas funcionalidades formam o núcleo da jogabilidade e são essenciais para o funcionamento do jogo.

## 1. Lista de Funcionalidades Básicas

As seguintes funcionalidades são consideradas básicas e foram verificadas:

- **Controle da Nave (RF01)**: O jogador pode mover a nave horizontalmente (esquerda e direita) dentro dos limites da tela.
- **Disparo de Lasers (RF02)**: A nave do jogador pode disparar projéteis verticalmente para cima.
- **Geração de Inimigos (RF04)**: Alienígenas são exibidos na parte superior da tela.
- **Barreiras de Proteção (RF08)**: Barreiras são exibidas na tela e podem ser destruídas.
- **Dano em Inimigos (RF13)**: Alienígenas desaparecem ao serem atingidos por um laser do jogador.
- **Exibição de Pontuação (RF16)**: A pontuação atual do jogador é visível na tela.
- **Incremento de Pontuação (RF17)**: A pontuação do jogador aumenta ao destruir alienígenas.
- **Tela Inicial (RF19)**: Uma tela inicial simples é apresentada ao iniciar o jogo.
- **Efeitos Sonoros (RF20)**: Sons são reproduzidos para ações chave do jogo (disparo, destruição de inimigo).

## 2. Verificação e Testes

Para cada uma dessas funcionalidades básicas, a verificação foi realizada principalmente através de testes manuais e observação direta durante a execução do jogo. Os critérios

de aceitação definidos nas User Stories e nos Requisitos Funcionais serviram como base para validar o comportamento esperado.

### Exemplos de Verificação:

- **Controle da Nave**: Iniciar o jogo e tentar mover a nave para a esquerda e direita, observando se ela se move suavemente e não sai da tela.
- **Disparo de Lasers**: Pressionar a tecla de disparo e verificar se um laser é gerado e se move para cima.
- **Dano em Inimigos**: Disparar em um alienígena e observar se ele desaparece e se a pontuação é atualizada.
- **Tela Inicial**: Iniciar a aplicação e verificar se a tela inicial é a primeira a ser exibida e se as opções básicas estão presentes.

Essas funcionalidades básicas garantem que o jogo seja jogável em sua essência e servem como um ponto de partida sólido para a implementação das funcionalidades mais avançadas.

# Manual do Usuário

Este manual irá guiá-lo através dos conceitos básicos do jogo, controles e objetivos para que você possa começar a jogar e se divertir.

## 1. Como Jogar

### 1.1. Iniciando o Jogo

Ao iniciar a aplicação, você será levado à **Tela Inicial**. Nela, você encontrará as seguintes opções:

- **Iniciar Jogo**: Começa uma nova partida.
- **Placares**: Exibe as pontuações mais altas salvas.
- **Controles**: Mostra este manual de controles.

Selecione "Iniciar Jogo" para começar sua aventura e defender a Terra dos invasores!

### 1.2. Objetivo do Jogo

Seu objetivo é destruir todas as ondas de alienígenas que descem em direção à sua nave. Não deixe que eles alcancem a parte inferior da tela, e proteja sua nave a todo custo!

## 2. Controles

Os controles são simples e intuitivos:

- **Mover para a Esquerda**: Use a tecla `Seta Esquerda` ou `A`.
- **Mover para a Direita**: Use a tecla `Seta Direita` ou `D`.
- **Disparar Laser**: Use a tecla `Barra de Espaço`.

## 3. Elementos da Interface (HUD)

Durante o jogo, você verá informações importantes na parte superior da tela:

- **Pontuação**: Exibida no canto superior esquerdo. Aumenta à medida que você destrói alienígenas.
- **Vidas**: Exibidas no canto superior direito. Representam quantas vezes sua nave pode ser atingida antes do jogo terminar.

## 4. Inimigos e Barreiras

- **Alienígenas**: Eles descem em ondas e disparam contra sua nave. Destrua-os para ganhar pontos.
- **Barreiras de Proteção**: Existem barreiras na parte inferior da tela que podem protegê-lo dos tiros inimigos. Elas se degradam e desaparecem ao serem atingidas.

## 5. Fim de Jogo

O jogo termina quando:

- Você perde todas as suas vidas.
- Os alienígenas conseguem alcançar a parte inferior da tela.

Ao final do jogo, a **Tela de Fim de Jogo** será exibida, onde você poderá:

- **Salvar sua Pontuação**: Insira seu apelido para registrar sua pontuação nos placares.
- **Jogar Novamente**: Inicia uma nova partida.
- **Voltar ao Menu Principal**: Retorna à tela inicial.

# Solução de Problemas

Este documento fornece soluções para problemas comuns que podem surgir durante a configuração, compilação ou execução do projeto.

## 1. Problemas de Conexão com o Banco de Dados (PostgreSQL)

### Problema: A aplicação não consegue se conectar ao banco de dados.

**Possíveis Causas e Soluções:**

- **Contêiner Docker não está rodando**: O banco de dados PostgreSQL é executado em um contêiner Docker. Certifique-se de que ele está ativo.
  - **Solução**: Abra um terminal no diretório raiz do projeto (`capstone-programming-3`) e execute:

    ```
    docker-compose up -d db
    ```

    Verifique o status com `docker ps`.
- **Porta 5432 já em uso**: Outro serviço pode estar utilizando a porta padrão do PostgreSQL.
  - **Solução**: Verifique qual processo está usando a porta 5432 (ex: `sudo lsof -i :5432` no Linux/macOS, ou `netstat -ano | findstr :5432` no Windows) e encerre-o, ou altere a porta no `docker-compose.yml` e na string de conexão da aplicação.
- **Credenciais incorretas**: O nome de usuário, senha ou nome do banco de dados na string de conexão da aplicação podem estar incorretos.
  - **Solução**: Verifique as variáveis de ambiente no `docker-compose.yml` (`POSTGRES_USER`, `POSTGRES_PASSWORD`, `POSTGRES_DB`) e compare com a string de conexão utilizada na aplicação.
- **Volume de dados corrompido**: Em casos raros, o volume de dados do Docker pode estar corrompido.

- **Solução**: Pare o contêiner (`docker-compose down`), remova o volume (`docker volume rm capstone-programming-3_db_data`) e inicie o contêiner novamente (`docker-compose up -d db`). **Atenção**: Isso apagará todos os dados do banco de dados.

# 2. Problemas de Áudio no Linux

## Problema: O jogo não reproduz sons no Linux.

**Possíveis Causas e Soluções:**

- **`mpg123` não está instalado**: O `SoundService` do projeto utiliza o utilitário `mpg123` para reprodução de áudio em ambientes Linux.
  - **Solução**: Instale o `mpg123` no seu sistema. Para sistemas baseados em Debian/Ubuntu:

    ```
    sudo apt-get update
    sudo apt-get install mpg123
    ```

    Para outras distribuições, consulte a documentação do seu gerenciador de pacotes.
- **Caminho do arquivo de áudio incorreto**: O jogo não consegue encontrar os arquivos de som.
  - **Solução**: Verifique se os arquivos `.mp3` estão na pasta `SpaceInvaders/Assets/sounds/` e se os caminhos em `Constants/SoundPaths.cs` estão corretos.

# 3. Problemas de Compilação e Execução do Projeto

## Problema: O projeto não compila ou não executa.

**Possíveis Causas e Soluções:**

- **SDK do .NET ausente ou versão incorreta**: O projeto requer o .NET SDK 9.0 (ou superior).

    - **Solução**: Baixe e instale a versão correta do .NET SDK em dotnet.microsoft.com/download (https://dotnet.microsoft.com/download).
- **Dependências não restauradas**: As bibliotecas e pacotes NuGet podem não ter sido baixados.
    - **Solução**: No diretório `SpaceInvaders/SpaceInvaders`, execute `dotnet restore`.
- **Templates do Uno Platform não instalados**: Se você estiver criando um novo projeto ou tiver problemas relacionados ao Uno Platform.
    - **Solução**: Execute `dotnet new install Uno.Templates`.
- **Erros de código**: Erros de sintaxe ou lógica no seu código.
    - **Solução**: Verifique as mensagens de erro no terminal ou na IDE e corrija o código. Utilize o comando `dotnet build` para verificar erros de compilação.
- **Caminho do projeto incorreto ao executar**: Ao usar `dotnet run`, o caminho para o `.csproj` deve estar correto.
    - **Solução**: Certifique-se de que você está no diretório `SpaceInvaders/SpaceInvaders` ou especifique o caminho completo: `dotnet run --project SpaceInvaders/SpaceInvaders.csproj -f net9.0-desktop`.