INSTITUTO FEDERAL
Brasília
Campus Brasília

Instituto Federal de Brasília

Campus Brasília

Técnico em Desenvolvimento de Sistemas

DOCUMENTAÇÃO DO SISTEMA GAME MACHINE

Disciplina: Banco de Dados II

Estudantes

1. Eduarda Estefany Sousa Ribeiro Mendes
2. Eduardo Figueiredo
3. Letícia Moreira Guimarães
4. Victor Hugo Batista Tavares

Professora

Josane Borges das Neves Guimarães

2025/2

INSTITUTO FEDERAL
Brasília
Campus Brasília

1. RESUMO

O presente resumo detalha o contexto do projeto, que se baseia na modelagem de um sistema de banco de dados para a empresa **GameCoreDW**. Trata-se de uma organização de médio porte especializada no **desenvolvimento e distribuição de jogos digitais**, além de oferecer uma **engine própria de criação**, utilizada internamente e por parceiros ou desenvolvedores independentes.

Com o crescimento da empresa e a diversificação de seus produtos, surgiu a necessidade de implantar um **sistema de gerenciamento centralizado** capaz de organizar informações relacionadas a todo o ciclo de desenvolvimento de jogos. O sistema, denominado **Sistema Game Machine**, foi concebido para integrar e otimizar o controle das principais atividades da empresa, abrangendo o **gerenciamento de projetos e equipes multidisciplinares** (programadores, designers e roteiristas), a **administração das engines e suas versões**, o **acompanhamento de parcerias externas** e a **gestão da publicação e distribuição dos jogos** em diversas plataformas, como **Steam, consoles e dispositivos móveis**.

A iniciativa responde à crescente demanda por **organização e rastreabilidade nos fluxos de trabalho** em estúdios de desenvolvimento, que operam com múltiplas linguagens, ferramentas e etapas simultâneas de produção. As informações essenciais sobre jogos, projetos, desenvolvedores e assets (imagens, áudios e scripts) encontravam-se dispersas em planilhas e documentos manuais, o que dificultava o acompanhamento das tarefas e comprometia a colaboração entre equipes.

O sistema foi implementado por meio de uma **interface web com diferentes níveis de acesso**, permitindo o **cadastro, atualização, consulta e geração de relatórios** sobre as entidades e seus relacionamentos. Além disso, conta com **mecanismos de rastreamento do histórico de versões** dos jogos e das engines, possibilitando maior controle e confiabilidade no ciclo de dados registrados.

Até o momento, o projeto passou pelas **três primeiras fases de desenvolvimento de banco de dados**: **análise de requisitos**, **modelagem conceitual** (por meio do diagrama Entidade-Relacionamento) e **modelagem relacional com normalização até a 3ª Forma Normal (3FN)**. Durante esse processo, foram definidas as entidades principais — *Usuário, Projeto, Jogo, Engine, Versão, Plataforma, Asset/Biblioteca* e *Publicação* —, bem como os relacionamentos entre elas e as regras de integridade referencial. Essa estrutura foi refinada para

eliminar redundâncias e garantir a consistência dos dados, resultando em um modelo eficiente e alinhado aos objetivos da GameCoreDW.

Os **requisitos funcionais** do sistema envolvem o controle de usuários e equipes, o cadastro e gerenciamento de jogos, o acompanhamento do desenvolvimento de projetos, o registro das engines e de suas versões, além da gestão de publicações e do controle de qualidade. Já os **requisitos não funcionais** abrangem desempenho, segurança, usabilidade, confiabilidade, manutenibilidade e portabilidade, assegurando que o sistema seja ágil, seguro e compatível com diferentes plataformas. Os **critérios de aceitação** definem as métricas que permitirão avaliar o sucesso do sistema em relação aos requisitos estabelecidos. Considerando **requisitos adicionais** relevantes que não se encaixam nas categorias anteriores, por exemplo, restrições com orçamento, prazo e tecnologias.

Na próxima etapa, o foco será a **implementação prática do modelo no PostgreSQL**, com a criação das tabelas, inserção de dados de teste e execução de consultas SQL para validação dos relacionamentos. É importante ressaltar que o **PostgreSQL é uma ferramenta de código aberto** e foi **adotado como Sistema Gerenciador de Banco de Dados (SGBD)** da aplicação, por oferecer estabilidade, segurança e aderência aos padrões SQL. A **concepção da interface do sistema** refletirá a estrutura do banco de dados e permitirá uma **interação intuitiva entre os usuários e as informações**, consolidando um ambiente de gestão eficiente e integrado.

Palavras-chave: Banco de Dados. Modelagem Relacional. PostgreSQL. Desenvolvimento de Jogos. Engine de Jogos.

INSTITUTO FEDERAL
Brasília
Campus Brasília

1.1 REQUISITOS FUNCIONAIS:

- ☐ **Cadastro e Gerenciamento dos Jogos:** Deve permitir o cadastro, edição e exclusão de jogos.

- ☐ **Login dos Usuários:** Deve-se fazer o controle de acessos dos usuários permitindo funções para cada tipo de usuário.

- ☐ **Cadastro de Usuários:** Deve permitir o cadastro de novos usuários desenvolvedores e jogadores.

- ☐ **Controle de Equipe:** Deve permitir vincular um ou mais desenvolvedores a diferentes projetos.

- ☐ **Cadastro do Projeto** (Andamento do desenvolvimento): Deve registrar os projetos em andamento e associá-los aos respectivos jogos.
  Deve armazenar e gerenciar os assets utilizados em cada projeto (imagens, áudios, modelos 3D, scripts etc.).
  Deve registrar o progresso individual dos desenvolvedores em cada projeto.

- ☐ **Cadastro de Engines e Versões:** Deve permitir registrar e consultar as versões das engines utilizadas em cada jogo.

- ☐ **Controle de Fornecimento:** Deve manter o histórico de publicação dos jogos, indicando as plataformas e datas de lançamento.

- ☐ **Controle de Qualidade:** Deve permitir o gerenciamento do relacionamento com desenvolvedores externos que utilizam a engine proprietária.

1.2 REQUISITOS NÃO FUNCIONAIS:

☐ **Desempenho**
Deve carregar imagens no máximo em 1 segundo.
Deve carregar áudios no máximo em 2 segundos.
Deve carregar vídeos no máximo em 3 segundos.
Deve estar disponível 24 horas por dia, 7 dias por semana.

☐ **Segurança**
Deve permitir acesso multiusuário com controle de permissões por perfil.
Deve exigir que cada usuário faça seu próprio login para utilizar o sistema.
Deve permitir que somente usuários administradores possam cadastrar novos itens.

☐ **Usabilidade**
Deve ser compatível como banco de imagens e vídeo.
Deve possuir interface intuitiva.

☐ **Confiabilidade**
Deve fazer o salvamento automático de jogos em desenvolvimento.

☐ **Manutenibilidade**
Deve possuir um campo para feedback de usuários/ testers e jogadores.
Deve-se guardar o histórico de versões do jogo em desenvolvimento.
Deve registrar logs de acesso e alterações feitas por usuários no banco de dados.

☐ **Portabilidade**
Deve funcionar em todos os navegadores.
Deve ser adaptado para dispositivos móveis.

1.3 CRITÉRIOS DE ACEITAÇÃO:

Definição: Avaliação do atendimento aos requisitos e indicadores de sucesso do sistema.

☐ **Funcionalidades Essenciais Implementadas:**

Sucesso: Todas as funcionalidades listadas nos Requisitos Funcionais (Cadastro de Jogos, Login, Gestão de Equipes, Projetos, Engines, Publicação, Controle de Qualidade) foram implementadas e funcionam conforme o esperado.

Indicador: 100% dos casos de uso críticos executados com sucesso em testes de aceitação do usuário (UAT).

☐ **Melhora na Rastreabilidade e Colaboração:**

Sucesso: Redução significativa do tempo gasto na busca de informações e aumento da comunicação entre as equipes.

Indicador: Pesquisas de satisfação da equipe mostram um aumento de 80% na percepção de organização e colaboração. Tempo médio para encontrar informações sobre um projeto/asset reduzido em 50%.

☐ **Desempenho da Aplicação:**

Sucesso: O sistema atende aos tempos de carregamento especificados para imagens, áudios e vídeos.

Indicador: Testes de desempenho automatizados confirmam que o carregamento de imagens está abaixo de 1s, áudios abaixo de 2s e vídeos abaixo de 3s em 95% das requisições.

☐ **Segurança e Acesso Controlado:**

Sucesso: O sistema implementa efetivamente o controle de acesso por perfil e exige autenticação para todos os usuários.

Indicador: Auditoria de segurança não encontra vulnerabilidades críticas relacionadas ao controle de acesso. Tentativas de acesso não autorizadas são bloqueadas.

☐ **Usabilidade e Experiência do Usuário (UX):**

Sucesso: A interface é intuitiva e fácil de usar para todos os perfis de usuário, resultando em menor necessidade de treinamento.

Indicador: Usuários conseguem completar tarefas essenciais sem assistência em 90% das vezes em testes de usabilidade. Taxa de erro dos usuários menor que 5%.

☐ **Confiabilidade e Disponibilidade:**

Sucesso: O sistema demonstra alta disponibilidade e capacidade de recuperação, com o salvamento automático funcionando como esperado.

Indicador: Uptime do sistema de 99,9% durante o período de teste. Nenhum dado é perdido devido a falhas inesperadas em 100% dos testes de interrupção simulada.

☐ **Manutenibilidade e Evolução:**

Sucesso: O sistema é facilmente atualizável e corrigível, com um bom registro de feedback e histórico de versões.

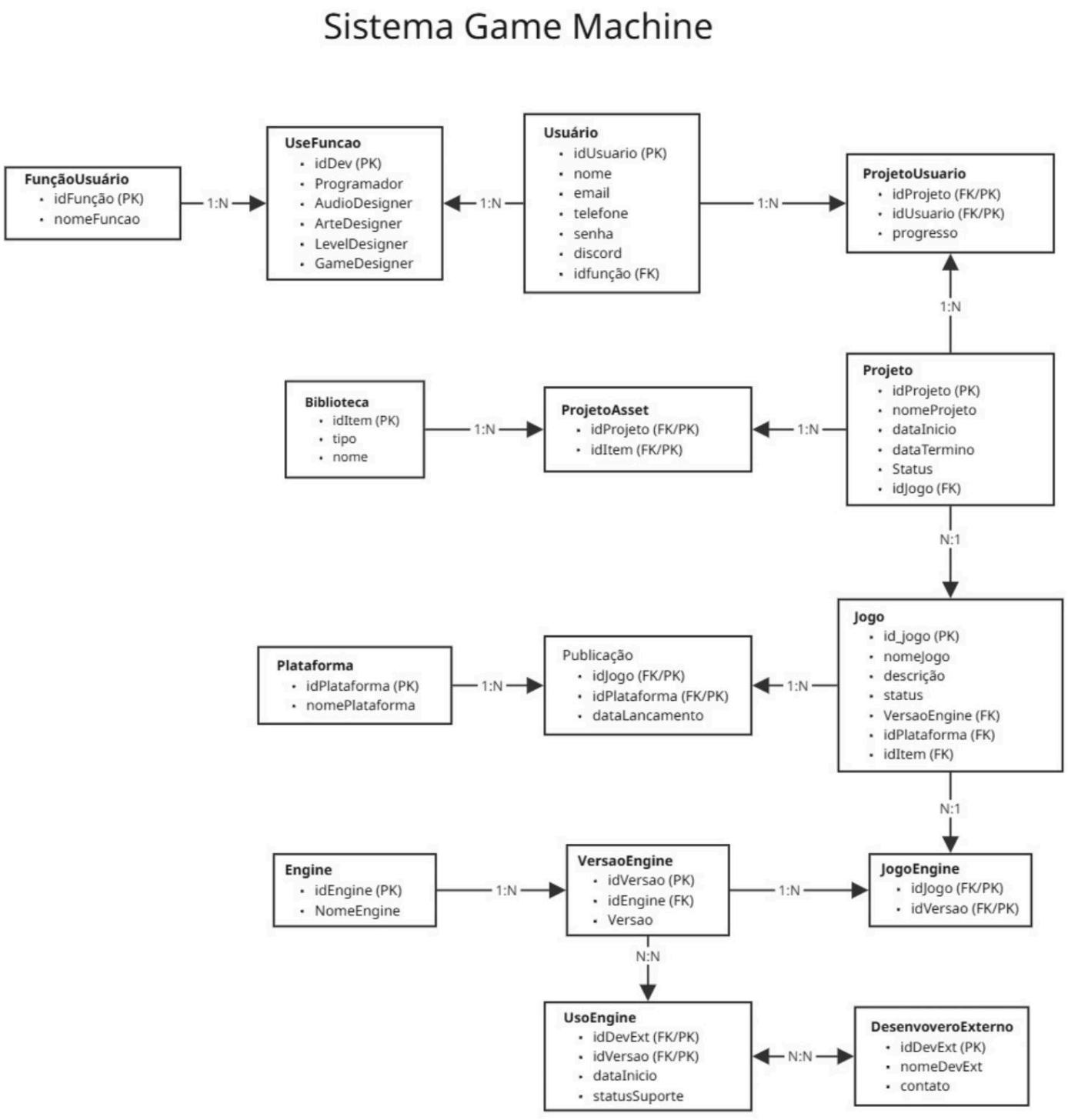
Indicador: O tempo médio para resolução de bugs críticos é de no máximo 24 horas. O sistema de feedback registra e processa 95% das sugestões e erros reportados.

☐ **Portabilidade:**

Sucesso: O sistema funciona perfeitamente em diferentes navegadores e dispositivos móveis.

Indicador: Testes de compatibilidade com navegadores e dispositivos móveis não identificam problemas críticos de layout ou funcionalidade.

1.4 MODELAGEM CONCEITUAL E LÓGICA:



1.4.1 DESCRIÇÃO DO MODELO ENTIDADE-RELACIONAMENTO

O **Modelo Entidade-Relacionamento** do projeto Game Machine é estruturado para gerenciar o ciclo de vida completo de desenvolvimento de jogos, desde o planejamento até a distribuição e o suporte a parceiros.

- **Entidades Principais e Atributos:**

As entidades centrais representam os pilares do negócio e possuem atributos essenciais para identificação e detalhamento:

Entidade	Descrição no Contexto do Projeto	Atributos Chave
Jogo	O produto final da empresa. É o elemento central que agrega diversos projetos.	idJogo, nome, descricao
Projeto	Representa as iniciativas de desenvolvimento (e.g., o jogo base, uma DLC, ou um port para um console). Um projeto é sempre associado a um Jogo específico.	idProjeto, nomeProjeto, dataInicio, status
Usuario	Os desenvolvedores internos da empresa (programadores, designers, artistas, etc.).	idUsuario, nome, email, dataContratacao
FuncaoUsuario	Classifica os perfis dos Usuários para controle de permissões e hierarquia.	idFuncao, nomeFuncao
Engine	A tecnologia base utilizada para criar os jogos (e.g., GameCore, Unity).	idEngine, nomeEngine

VersaoEngine	Versões específicas de uma Engine (e.g., v1.0.1, v2.0.0). É a versão que é, de fato, utilizada por um Projeto ou por um Desenvolvedor Externo.	idVersao, numeroVersao, dataRelease
Plataforma	Locais onde o Jogo pode ser distribuído e vendido (e.g., Steam, Google Play, PS5).	idPlataforma, nomePlataforma
Biblioteca	Repositório central de Assets (recursos como imagens, áudios, modelos 3D) que são usados nos Projetos.	idAsset, tipoAsset, descricaoAsset
DesenvolvedorExterno	Parceiros ou estúdios licenciados para utilizar a Engine proprietária da Game Machine.	idDevExt, nomeEmpresa, contato

- **Relacionamentos 1:N**

Jogo e Projeto (1:N): Um Jogo pode ser desmembrado em vários Projetos (Projeto Base, Projeto de Expansão).

Engine e VersaoEngine (1:N): Uma Engine pode ter várias VersaoEngine ao longo do tempo.

FuncaoUsuario e Usuario (1:N): Uma Função (Programador Sênior) pode ser atribuída a vários Usuários.

- **Relacionamentos N:N (Quebrados por Entidades Associativas)**

Os relacionamentos Muitos para Muitos (N:N) são cruciais para o gerenciamento centralizado e são mapeados para entidades associativas para registrar informações detalhadas sobre a participação ou uso.

Entidade Associativa	Relacionamento Quebrado	Atributo Adicional	Contexto no Game Machine
DesenvolvedorProjeto	Usuario N:N Projeto	progresso, dataAlocacao	Rastreia a participação de vários Usuários em vários Projetos, registrando o progresso individual (progresso %).
Publicacao	Jogo N:N Plataforma	dataLancamento	Registra em quais Plataformas o Jogo foi lançado. O atributo dataLancamento é vital, pois a data é diferente para cada plataforma.
ProjetoAsset	Projeto N:N Biblioteca	qtdUsada	Indica quais Assets foram usados em quais Projetos, permitindo registrar a quantidadeUsada do asset.

UsoEngine	VersaoEngine N:N DesenvolvedorExterno	dataInicio, statusSuporte	Controla qual Versão da Engine está sendo utilizada por cada Desenvolvedor Externo parceiro, permitindo gerenciar o statusSuporte de cada licença.

- **Relacionamentos e Cardinalidade**

Relacionamento é a conexão lógica entre duas ou mais entidades (tabelas), que reflete uma regra de negócio que precisa ser mantida no banco de dados.

Por exemplo, o relacionamento entre a entidade Jogo e a entidade.

Plataforma estabelece que um Jogo pode ser distribuído em diferentes plataformas (como Steam ou consoles).

Cardinalidade define o grau de participação ou a quantidade de ocorrências de uma entidade que se relacionam com as ocorrências de outra. É o que transforma uma regra de negócio em uma regra de estrutura do banco.

Por exemplo, *vários* usuários (desenvolvedores) trabalham em *vários* Projetos. (Isso requer uma tabela intermediária).

Entidade A	Relacionamento	Entidade B	Cardinalidade	Observação (Tabela Associativa/FK)
Usuário	usa	FunçãoUsuario	N:N	Tabela UsaFuncao
Usuário	participa	Projeto	N:N	Tabela DesenvolvedorProjeto
Projeto	utiliza	Biblioteca (Assets)	N:N	Tabela ProjetoAsset
Jogo	tem	Projeto	1:N	FK idJogo em Projeto
Engine	possui	VersaoEngine	1:N	FK idEngine em VersaoEngine
Jogo	usa	VersaoEng	N:N	Tabela JogoEngine

		ine		
Jogo	é publicado em	Plataforma	N:N	Tabela Publicacao
VersaoEngine	é usada por	DesenvolvedorExterno	N:N	Tabela UsoEngine

2. MODELO RELACIONAL DESCRITIVO

O **Modelo Relacional Descritivo** é a representação final e estruturada do seu banco de dados, sendo o resultado do mapeamento do Modelo Entidade-Relacionamento (MER).

Ele serve como a **especificação técnica** exata para a criação das tabelas no sistema de gerenciamento de banco de dados (SGBD).

Tabela	Colunas (Atributos)	Chaves (PK / FK)
Jogo	idJogo, nome, descricao	PK: idJogo
FuncaoUsuario	idFuncao, nomeFuncao	PK: idFuncao
Usuario	idUsuario, idFuncaoUsuario, nome, email, dataContratacao	PK: idUsuario / FK: idFuncaoUsuario ref. FuncaoUsuario
Engine	idEngine, nomeEngine	PK: idEngine
VersaoEngine	idVersao, idEngine, numeroVersao, dataRelease	PK: idVersao / FK: idEngine ref. Engine
Projeto	idProjeto, idJogo, idVersaoEngine, nomeProjeto, dataInicio, dataPrevTermino, status	PK: idProjeto / FK: idJogo ref. Jogo, FK: idVersaoEngine ref. VersaoEngine
Plataforma	idPlataforma, nomePlataforma	PK: idPlataforma
Biblioteca	idAsset, tipoAsset, descricaoAsset,	PK: idAsset

	caminhoArquivo	
Desenvolvedor Externo	idDevExt, nomeEmpresa, contato, cnpj	PK: idDevExt
Desenvolvedor Projeto	idUsuario, idProjeto, progresso, dataAlocacao	PK: (idUsuario, idProjeto) / FK: idUsuario ref. Usuario, FK: idProjeto ref. Projeto
Publicacao	idJogo, idPlataforma, dataLancamento, urlStore	PK: (idJogo, idPlataforma) / FK: idJogo ref. Jogo, FK: idPlataforma ref. Plataforma
ProjetoAsset	idProjeto, idAsset, qtdUsada	PK: (idProjeto, idAsset) / FK: idProjeto ref. Projeto, FK: idAsset ref. Biblioteca
UsoEngine	idVersao, idDevExt, dataInicio, statusSuporte, licencaValidaAte	PK: (idVersao, idDevExt) / FK: idVersao ref. VersaoEngine, FK: idDevExt ref. DesenvolvedorExterno

ANEXO I - COMANDOS SQL UTILIZADOS

```
-- 1. CRIAÇÃO DAS TABELAS SEM CHAVES ESTRANGEIRAS (FK)

-- Tabela 1: JOGO
-- Armazena o produto final da empresa.
CREATE TABLE Jogo (
    idJogo      SERIAL PRIMARY KEY,      -- Chave Primária, auto-incrementável
    nome        VARCHAR(100) NOT NULL,   -- Nome do jogo (Obrigatório)
    descricao   TEXT,                    -- Descrição detalhada do jogo
    dataCriacao DATE DEFAULT CURRENT_DATE
);

-- Tabela 2: ENGINE
-- Lista as tecnologias bases de desenvolvimento (ex: Unity, Unreal, GameCore).
CREATE TABLE Engine (
    idEngine    SERIAL PRIMARY KEY,
    nomeEngine  VARCHAR(50) NOT NULL UNIQUE
);

-- Tabela 3: PLATAFORMA
-- Lista os canais de distribuição dos jogos (ex: Steam, PS5, Android).
CREATE TABLE Plataforma (
    idPlataforma    SERIAL PRIMARY KEY,
    nomePlataforma  VARCHAR(50) NOT NULL UNIQUE
);

-- Tabela 4: FUNCAO_USUARIO
-- Classifica os perfis dos desenvolvedores (ex: Programador, Designer, Artista).
CREATE TABLE FuncaoUsuario (
```

```
    idFuncao    SERIAL PRIMARY KEY,
    nomeFuncao  VARCHAR(50) NOT NULL UNIQUE
);

-- Tabela 5: BIBLIOTECA (Assets)
-- Repositório de assets reutilizáveis (imagens, áudios, scripts, modelos 3D).
CREATE TABLE Biblioteca (
    idAsset         SERIAL PRIMARY KEY,
    tipoAsset       VARCHAR(50) NOT NULL, -- Ex: 'Imagem', 'Áudio', 'Modelo 3D'
    descricaoAsset  VARCHAR(255),
    caminhoArquivo  VARCHAR(255) NOT NULL UNIQUE
);

-- Tabela 6: DESENVOLVEDOR_EXTERNO
-- Parceiros ou estúdios que licenciam a Engine proprietária.
CREATE TABLE DesenvolvedorExterno (
    idDevExt    SERIAL PRIMARY KEY,
    nomeEmpresa VARCHAR(100) NOT NULL,
    contato     VARCHAR(100),
    cnpj        VARCHAR(18) UNIQUE
);

-- 2. CRIAÇÃO DAS TABELAS COM DEPENDÊNCIAS DE FK SIMPLES

-- Tabela 7: VERSAO_ENGINE
-- Lista as versões específicas de uma Engine, dependendo da Tabela Engine.
CREATE TABLE VersaoEngine (
    idVersao      SERIAL PRIMARY KEY,
    idEngine      INTEGER NOT NULL,
    numeroVersao  VARCHAR(20) NOT NULL,
```

```
    dataRelease   DATE,

    FOREIGN KEY (idEngine) REFERENCES Engine(idEngine)
        ON DELETE RESTRICT ON UPDATE CASCADE
);

-- Tabela 8: USUARIO (Desenvolvedores Internos)
-- Lista os desenvolvedores internos, dependendo da Tabela FuncaoUsuario.
CREATE TABLE Usuario (
    idUsuario         SERIAL PRIMARY KEY,
    idFuncaoUsuario   INTEGER NOT NULL,
    nome              VARCHAR(100) NOT NULL,
    email             VARCHAR(100) NOT NULL UNIQUE,
    dataContratacao   DATE DEFAULT CURRENT_DATE,

    FOREIGN KEY (idFuncaoUsuario) REFERENCES FuncaoUsuario(idFuncao)
        ON DELETE RESTRICT ON UPDATE CASCADE
);

-- Tabela 9: PROJETO
-- Representa a unidade de trabalho, dependendo de Jogo e VersaoEngine.
CREATE TABLE Projeto (
    idProjeto         SERIAL PRIMARY KEY,
    idJogo            INTEGER NOT NULL,
    idVersaoEngine    INTEGER NOT NULL,
    nomeProjeto       VARCHAR(100) NOT NULL,
    dataInicio        DATE NOT NULL,
    dataPrevTermino   DATE,
     status                VARCHAR(50) NOT NULL, -- Ex: 'Em Andamento', 'Concluído',
'Arquivado'
```

```
    FOREIGN KEY (idJogo) REFERENCES Jogo(idJogo)
        ON DELETE RESTRICT ON UPDATE CASCADE,
    FOREIGN KEY (idVersaoEngine) REFERENCES VersaoEngine(idVersao)
        ON DELETE RESTRICT ON UPDATE CASCADE
);

-- 3. CRIAÇÃO DAS TABELAS ASSOCIATIVAS (Relacionamentos N:N)

-- Tabela 10: DESENVOLVEDOR_PROJETO
-- Associa N Usuários a N Projetos, registrando a participação e o progresso.
CREATE TABLE DesenvolvedorProjeto (
    idUsuario       INTEGER NOT NULL,
    idProjeto       INTEGER NOT NULL,
        progresso           NUMERIC(5,2) DEFAULT 0 CHECK (progresso >= 0 AND
progresso <= 100), -- Percentual
    dataAlocacao    DATE DEFAULT CURRENT_DATE,

    PRIMARY KEY (idUsuario, idProjeto),
    FOREIGN KEY (idUsuario) REFERENCES Usuario(idUsuario)
        ON DELETE RESTRICT ON UPDATE CASCADE,
    FOREIGN KEY (idProjeto) REFERENCES Projeto(idProjeto)
        ON DELETE RESTRICT ON UPDATE CASCADE
);

-- Tabela 11: PUBLICACAO
-- Associa N Jogos a N Plataformas, registrando a data de lançamento específica.
CREATE TABLE Publicacao (
    idJogo          INTEGER NOT NULL,
    idPlataforma    INTEGER NOT NULL,
```

```
    dataLancamento  DATE NOT NULL,
    urlStore        VARCHAR(255),

    PRIMARY KEY (idJogo, idPlataforma),
    FOREIGN KEY (idJogo) REFERENCES Jogo(idJogo)
        ON DELETE RESTRICT ON UPDATE CASCADE,
    FOREIGN KEY (idPlataforma) REFERENCES Plataforma(idPlataforma)
        ON DELETE RESTRICT ON UPDATE CASCADE
);

-- Tabela 12: PROJETO_ASSET
-- Associa N Projetos a N Assets da Biblioteca, registrando a quantidade usada.
CREATE TABLE ProjetoAsset (
    idProjeto   INTEGER NOT NULL,
    idAsset     INTEGER NOT NULL,
    qtdUsada    INTEGER DEFAULT 1,

    PRIMARY KEY (idProjeto, idAsset),
    FOREIGN KEY (idProjeto) REFERENCES Projeto(idProjeto)
        ON DELETE RESTRICT ON UPDATE CASCADE,
    FOREIGN KEY (idAsset) REFERENCES Biblioteca(idAsset)
        ON DELETE RESTRICT ON UPDATE CASCADE
);

-- Tabela 13: USO_ENGINE
-- Associa N Versões da Engine a N Desenvolvedores Externos, para controle de licença.
CREATE TABLE UsoEngine (
    idVersao            INTEGER NOT NULL,
    idDevExt            INTEGER NOT NULL,
```

```
    dataInicio          DATE NOT NULL,
    statusSuporte       VARCHAR(50) NOT NULL, -- Ex: 'Ativo', 'Expirado', 'Pendente'
    licencaValidaAte    DATE,

    PRIMARY KEY (idVersao, idDevExt),
    FOREIGN KEY (idVersao) REFERENCES VersaoEngine(idVersao)
        ON DELETE RESTRICT ON UPDATE CASCADE,
    FOREIGN KEY (idDevExt) REFERENCES DesenvolvedorExterno(idDevExt)
        ON DELETE RESTRICT ON UPDATE CASCADE
);

-- Foi utilizada uma IA para geração de dados e organização dos comandos "INSERT INTO", segue abaixo...

-- [i. INSERT] 1. Insere funções de usuário
INSERT INTO FuncaoUsuario (nomeFuncao)
VALUES ('Programador Sênior'),
       ('Artista 2D/3D'),
       ('Game Designer'),
       ('Gerente de Projeto');

-- [i. INSERT] 2. Insere Engines
INSERT INTO Engine (nomeEngine)
VALUES ('GameCore v1.0'),
       ('Unity 2024 LTS');

-- [i. INSERT] 3. Insere Plataformas
INSERT INTO Plataforma (nomePlataforma)
VALUES ('Steam PC'),
       ('PlayStation 5'),
```

```
    ('Mobile Android');

-- [i. INSERT] 4. Insere Jogos
INSERT INTO Jogo (nome, descricao)
VALUES ('Chronos Rift', 'RPG de ação e ficção científica.'),
    ('Dino Dash Racing', 'Jogo de corrida casual para mobile.');

-- [i. INSERT] 5. Insere Assets na Biblioteca
INSERT INTO Biblioteca (tipoAsset, descricaoAsset, caminhoArquivo)
VALUES ('Modelo 3D', 'Personagem principal, herói.', '/assets/models/hero.fbx'),
    ('Áudio', 'Música tema da fase 1.', '/assets/audio/theme1.mp3');

-- [i. INSERT] 6. Insere Desenvolvedor Externo (Parceiro)
INSERT INTO DesenvolvedorExterno (nomeEmpresa, contato)
VALUES ('Aero Studios', 'contato@aero.com.br');

-- [i. INSERT] 7. Insere Versões de Engine (FK para Engine)
-- Usa idEngine = 1 (GameCore v1.0) e idEngine = 2 (Unity 2024 LTS)
INSERT INTO VersaoEngine (idEngine, numeroVersao, dataRelease)
VALUES (1, '1.0.12', '2025-05-01'),
    (2, '2024.1.5f1', '2025-07-20');

-- [i. INSERT] 8. Insere Usuários (FK para FuncaoUsuario)
-- Usa idFuncaoUsuario = 1 (Programador) e idFuncaoUsuario = 4 (Gerente)
INSERT INTO Usuario (idFuncaoUsuario, nome, email, dataContratacao)
VALUES (1, 'Alan Smith', 'alan@game.com', '2024-01-15'),
    (4, 'Danielly Reis', 'dani@game.com', '2023-05-20');

-- [i. INSERT] 9. Insere Projetos (FK para Jogo e VersaoEngine)
-- Usa idJogo = 1 ('Chronos Rift') e idVersaoEngine = 1 (GameCore 1.0.12)
```

```
INSERT INTO Projeto (idJogo, idVersaoEngine, nomeProjeto, dataInicio, dataPrevTermino, status)
VALUES (1, 1, 'CR - Módulo de Combate', '2025-08-01', '2026-03-30', 'Em Andamento'),
    (2, 2, 'DD - UI/UX Mobile', '2025-09-10', '2025-12-15', 'Em Andamento');

-- [i. INSERT] 10. Associa Desenvolvedores a Projetos (FK para Usuario e Projeto)
-- Associa Alan Smith (idUsuario=1) ao Projeto CR - Módulo de Combate (idProjeto=1)
INSERT INTO DesenvolvedorProjeto (idUsuario, idProjeto, progresso)
VALUES (1, 1, 65.50);

-- [i. INSERT] 11. Registra a Publicação do Jogo (FK para Jogo e Plataforma)
-- Publica 'Chronos Rift' (idJogo=1) na 'Steam PC' (idPlataforma=1)
INSERT INTO Publicacao (idJogo, idPlataforma, dataLancamento)
VALUES (1, 1, '2026-04-10');

-- [i. INSERT] 12. Associa Assets a Projetos (FK para Projeto e Biblioteca)
-- Projeto 'CR - Módulo de Combate' (idProjeto=1) usa Asset 'Modelo 3D' (idAsset=1)
INSERT INTO ProjetoAsset (idProjeto, idAsset, qtdUsada)
VALUES (1, 1, 4);

-- [i. INSERT] 13. Registra o Uso da Engine por Parceiro (FK para VersaoEngine e DevExt)
-- Aero Studios (idDevExt=1) usa Versão 1.0.12 (idVersao=1)
INSERT INTO UsoEngine (idVersao, idDevExt, dataInicio, statusSuporte, licencaValidaAte)
VALUES (1, 1, '2025-10-01', 'Ativo', '2026-10-01');
```

```
-- Para facilitar a inserção de dados, solicitei à IA para gerar os dados a partir de um arquivo csv.
-- Vamos precisar apagar os dados para inserir os dados a partir do CSV devido aos cadastros de itens SERIAL.

-- Antes para atender o topico da tarefa ii. Remover dados das tabelas; vou remover um dos dados.
-- Remove um projeto específico que foi cancelado antes do início.

DELETE FROM Projeto
-- Condição: Apaga o projeto com ID 2 (CR - Expansão I: O Despertar) e status 'Em Planejamento'
WHERE idProjeto = 2 AND status = 'Em Planejamento';

-- Agora vamos seguir com a limpeza de dados de todas as tabelas...
-- TRUNCATE TABLE: Comando mais rápido para remover todos os dados e resetar contadores.
-- CASCADE: Remove os dados das tabelas que fazem referência a estas.
-- RESTART IDENTITY: Reseta o contador SERIAL das chaves primárias para 1.
TRUNCATE TABLE
    Jogo, Engine, Plataforma, FuncaoUsuario, Biblioteca, DesenvolvedorExterno,
    VersaoEngine, Usuario, Projeto,
    DesenvolvedorProjeto, Publicacao, ProjetoAsset, UsoEngine
RESTART IDENTITY CASCADE;

-- Apaga os dados das tabelas associativas e de relacionamento (devem ser apagadas primeiro devido às FKs)
TRUNCATE TABLE DesenvolvedorProjeto RESTART IDENTITY;
TRUNCATE TABLE Publicacao RESTART IDENTITY;
TRUNCATE TABLE ProjetoAsset RESTART IDENTITY;
```

```
TRUNCATE TABLE UsoEngine RESTART IDENTITY;

-- Apaga os dados das tabelas com dependências de FK simples
TRUNCATE TABLE Projeto RESTART IDENTITY;
TRUNCATE TABLE Usuario RESTART IDENTITY;
TRUNCATE TABLE VersaoEngine RESTART IDENTITY;

-- Apaga os dados das tabelas mestras (sem FKs de entrada)
TRUNCATE TABLE Jogo RESTART IDENTITY;
TRUNCATE TABLE Engine RESTART IDENTITY;
TRUNCATE TABLE Plataforma RESTART IDENTITY;
TRUNCATE TABLE FuncaoUsuario RESTART IDENTITY;
TRUNCATE TABLE Biblioteca RESTART IDENTITY;
TRUNCATE TABLE DesenvolvedorExterno RESTART IDENTITY;

SELECT COUNT(*) FROM jogo;
SELECT COUNT(*) FROM Engine;
SELECT COUNT(*) FROM Plataforma;
SELECT COUNT(*) FROM FuncaoUsuario;
SELECT COUNT(*) FROM Biblioteca;
SELECT COUNT(*) FROM DesenvolvedorExterno;
-- Tabelas com FKs Simples
SELECT COUNT(*) FROM VersaoEngine;
SELECT COUNT(*) FROM Usuario;
SELECT COUNT(*) FROM Projeto;
-- Tabelas Associativas (N:N)
SELECT COUNT(*) FROM DesenvolvedorProjeto;
SELECT COUNT(*) FROM Publicacao;
SELECT COUNT(*) FROM ProjetoAsset;
SELECT COUNT(*) FROM UsoEngine;
```

```
SELECT * FROM Jogo;


-- #### APRESENTAR 10 CONSULTAS DIFERENTES ####
-- 01. Consulta (LIKE): Lista todos os jogos cujo nome começa com 'C'.
SELECT nome, descricao FROM Jogo WHERE nome LIKE 'C%';

-- 02. Consulta (AND / ORDER BY): Lista programadores (idFuncaoUsuario = 1, 2 ou 3) contratados após 2024, ordenados do mais novo ao mais antigo na empresa.
SELECT nome, dataContratacao, idFuncaoUsuario
FROM Usuario
WHERE idFuncaoUsuario IN (1, 2, 3) AND dataContratacao > '2024-01-01'
ORDER BY dataContratacao DESC;

-- 03. Consulta (OR): Lista os projetos que estão 'Em Andamento' OU 'Em Planejamento'.
SELECT idProjeto, nomeProjeto, status
FROM Projeto
WHERE status = 'Em Andamento' OR status = 'Em Planejamento';

-- 04. Consulta (Operador Aritmético): Calcula a duração (em dias) de projetos CONCLUÍDOS.
SELECT nomeProjeto, (dataPrevTermino - dataInicio) AS duracao_em_dias
FROM Projeto
WHERE status = 'Concluído';

-- 05. Consulta (NOT IN): Lista os Desenvolvedores Externos que NÃO estão nas empresas com ID 1, 2 ou 3.
SELECT nomeEmpresa, contato
```

```
FROM DesenvolvedorExterno
WHERE idDevExt NOT IN (1, 2, 3);

-- 06. Consulta (JOIN / GROUP BY): Conta quantos desenvolvedores estão alocados em cada projeto.
SELECT P.nomeProjeto, COUNT(DP.idUsuario) AS total_desenvolvedores
FROM Projeto P
JOIN DesenvolvedorProjeto DP ON P.idProjeto = DP.idProjeto
GROUP BY P.nomeProjeto
ORDER BY total_desenvolvedores DESC;

-- 07. Consulta (JOIN / GROUP BY / HAVING): Lista os Jogos que foram publicados em mais de uma plataforma.
SELECT J.nome AS NomeDoJogo, COUNT(P.idPlataforma) AS TotalDePlataformas
FROM Jogo J
JOIN Publicacao P ON J.idJogo = P.idJogo
GROUP BY J.nome
HAVING COUNT(P.idPlataforma) > 1;

-- 08. Consulta (JOIN / IN): Lista o nome do projeto e a Engine que usam versões da 'GameCore'.
SELECT P.nomeProjeto, E.nomeEngine || ' (' || VE.numeroVersao || ')' AS EngineUtilizada
FROM Projeto P
JOIN VersaoEngine VE ON P.idVersaoEngine = VE.idVersao
JOIN Engine E ON VE.idEngine = E.idEngine
WHERE E.nomeEngine IN ('GameCore v1.0', 'GameCore v1.1');

-- 09. Consulta (EXISTS): Lista os nomes dos Desenvolvedores Externos que possuem pelo menos uma licença de Engine com status de Suporte ATIVO.
```

```
SELECT nomeEmpresa
FROM DesenvolvedorExterno DE
WHERE EXISTS (
    SELECT 1
    FROM UsoEngine UE
    WHERE UE.idDevExt = DE.idDevExt AND UE.statusSuporte = 'Ativo'
);

-- 10. Consulta (Múltiplos JOIN): Lista o Nome do Jogo, Nome do Projeto e o Desenvolvedor alocado, para rastreamento completo.
SELECT J.nome AS Jogo, P.nomeProjeto AS Projeto, U.nome AS Desenvolvedor, DP.progresso
FROM Jogo J
JOIN Projeto P ON J.idJogo = P.idJogo
JOIN DesenvolvedorProjeto DP ON P.idProjeto = DP.idProjeto
JOIN Usuario U ON DP.idUsuario = U.idUsuario
ORDER BY J.nome, P.nomeProjeto;
```