Plataforma Multifacetada de Jogos Online - STEAM

BASE DE DADOS

Turma 6, Grupo 603

Filipe de Azevedo Cardoso

(up202006409@up.pt)

José Miguel Moreira Isidro

(up202006485@up.pt)

Miguel Ângelo Aguiar e Nogueira

(up202005488@up.pt)

Indíce

1 Contexto da Base de Dados

1.1 INTRODUÇÃO

No âmbito da unidade curricular Base de Dados lecionada no primeiro semestre do segundo ano da Licenciatura em Engenharia Informática e Computação, decidimos recriar de forma sintética a base de dados da plataforma multifacetada de jogos online Steam. Como tal, decidimos focar o nosso esquema UML nas principais funcionalidades da mesma. De um ponto de vista geral, o trabalho gira em torno da sua loja de jogos e ainda na plataforma comunitária desta. Implementando assim as funcionalidades de Perfil, Amigos, Grupos Temáticos, Biblioteca e Loja de jogos.

1.2 UTILIZADOR

Primeiramente podemos observar a classe Utilizador que representa cada um dos diferentes utilizadores da plataforma. Sobre um utilizador é necessário guardar o seu email, username, password e a data de inscrição na plataforma. Cada utilizador tem associado a si uma morada. Pode ter ainda um ou mais carrinhos de compra (um por cada transação) no qual pode adicionar e remover jogos para comprar no futuro. Em adição, tem uma associação própria que representa a possibilidade de dois utilizadores serem amigos na plataforma.

Quando um utilizador compra um jogo, ocorre uma transação. Esta informação será guardada numa classe de associação entre Utilizador e Carrinho chamada Transação. Cada transação deve guardar o seu valor, a data da transação e ainda o método de pagamento utilizado na compra.

A informação de cada utilizador está exibida num perfil. Este é público e pode ser visto por todos os utilizadores da plataforma. Pode ser comparado a um perfil de uma rede social, em que é mostrada uma informação breve sobre cada utilizador.

Relativamente ao perfil, cada utilizador guarda ainda um nickname, uma breve descrição e o nível do utilizador, que corresponde ao número de jogos comprados/jogados pelo utilizador. Para além disso, as horas jogadas pelo utilizador num determinado jogo podem ser visualizadas no perfil.

1.3 JOGO

A classe Jogo representa os diferentes jogos disponíveis para compra na plataforma. Cada jogo tem um título, uma classificação, um preço, a data de lançamento e um rating (PEGI). A classe jogo tem uma associação com a classe Categoria que indica a(s) categoria(s) em que esse jogo se encaixa. Tem uma associação com a classe Desenvolvedores que indica os

desenvolvedores do jogo em questão. Para além disso, tem ainda duas associações com a classe Carrinho que indicam quando um jogo é adicionado ou removido do carrinho.

Por cada jogo comprado, guarda-se numa classe de associação Biblioteca (relativa a cada utilizador) o número de horas jogadas pelo utilizador no jogo em concreto. Guarda-se também o número de horas jogadas pelo utilizador apenas nas 2 últimas semanas para fins estatísticos.

1.4 GRUPO

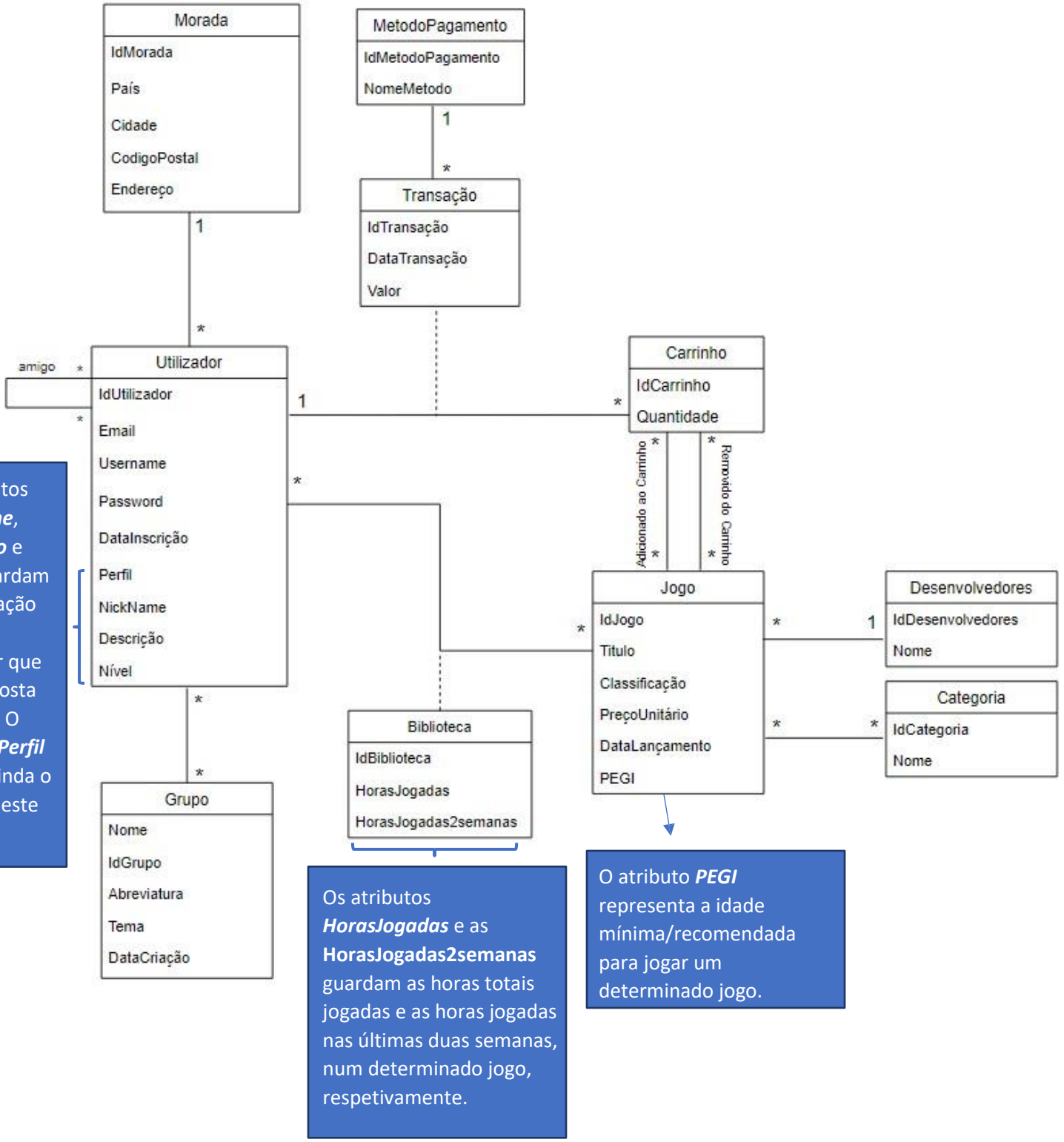
Utilizadores podem entrar em grupos para conviver entre si. Um grupo tem 4 atributos, um nome, uma abreviatura (ex. Futebol Clube do Porto – FCP) e um tema (Luis Díaz Fan Club - LDFC) e ainda a data de criação do grupo.

1.5 CARRINHO

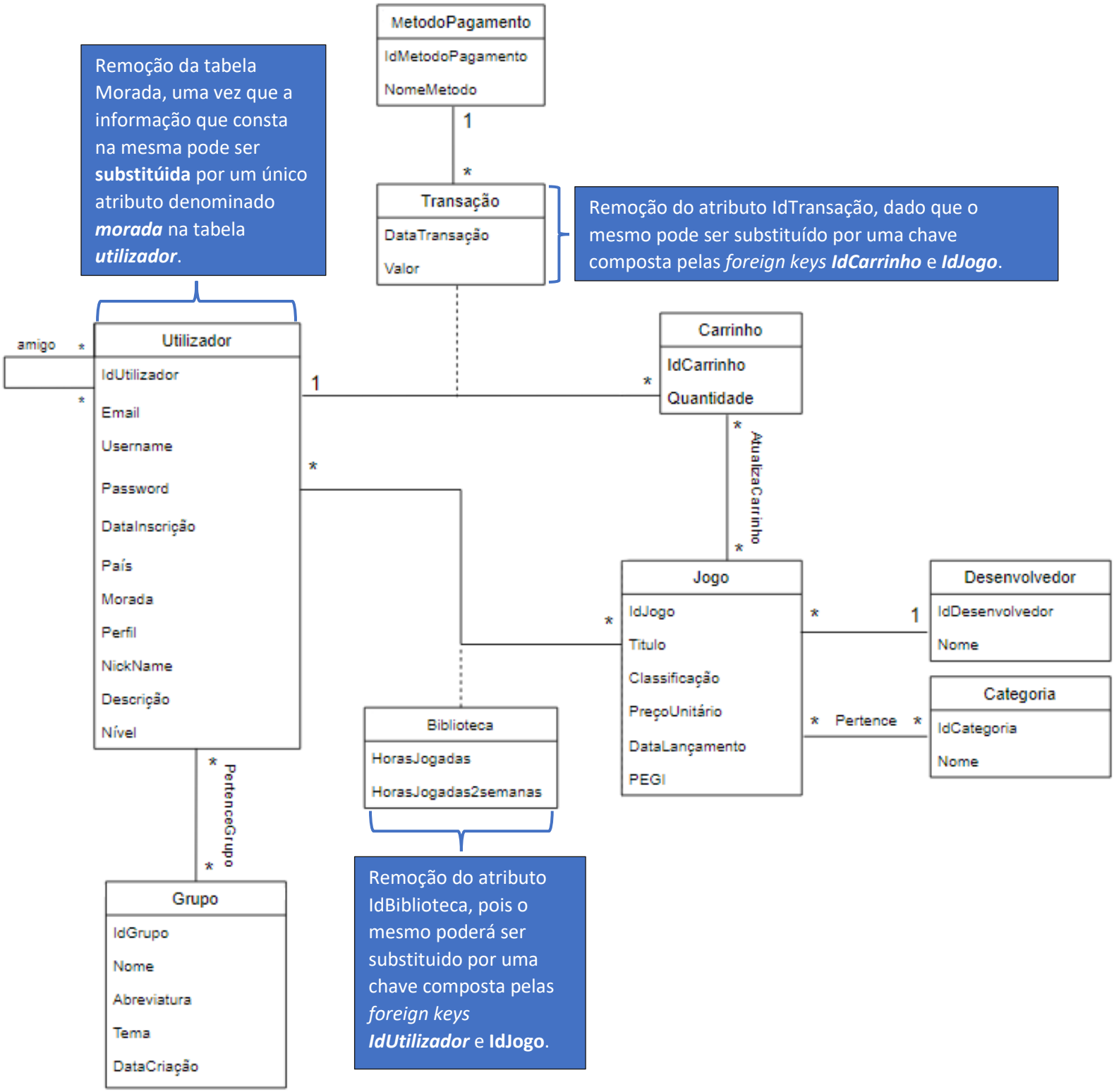
A classe Carrinho serve para a compra dos jogos. Esta guarda a quantidade de cada jogo que está inserido num carrinho. Um utilizador pode ter um e um só carrinho.

Como referido anteriormente em Utilizador e em Jogo, esta classe tem associações com a classe Utilizador (classe de associação entre Utilizador e Carrinho chamada Transação) e com a classe Jogo (as 2 associações indicam quando um jogo é adicionado ou removido do carrinho).

2 Diagrama UML



3 Diagrama UML Atualizado



4 Esquema Relacional

Utilizador(IdUtilizador, Email, Username, Password, DataInscirção, País, Morada, Perfil, NickName, Descrição, Nível)

Grupo (IdGrupo, Nome, Abreviatura, Tema, DataCriação)

Categoria(IdCategoria, Nome)

Desenvolvedor(IdDesenvolvedor, Nome)

Jogo(IdJogo, Título, Classificação, PreçoUnitário, DataLançamento, PEGI, IdDesenvolvedor->Desenvolvedor)

Carrinho(IdCarrinho, IdJogo->Jogo, Quantidade)

MetodoPagamento(IdMetodoPagamento, NomeMetodo)

Transação(IdCarrinho->Carrinho, IdJogo->Carrinho, DataTransação, Valor, IdMetodoPagamento->MetodoPagamento, IdUtilizador->Utilizador)

Biblioteca(IdJogo->Jogo, IdUtilizador->Utilizador, HorasJogadas, HorasJogadas2semanas)

CategoriaJogo(IdJogo->Jogo, IdCategoria->Categoria)

Amigo (IdUtilizador1->Utilizador, IdUtilizador2->Utilizador)

PertenceGrupo(IdUtilizador->Utilizador, IdGrupo->Grupo)

5 Análise Dependências Funcionais e Formas Normais

Utilizador (IdUtilizador, Email, Username, Password, DataInscirção, País, Morada, Perfil, NickName, Descrição, Nível)

1. IdUtilizador -> Email, Username, Password, DataInscrição, País, Morada, Perfil, NickName, Descrição, Nível
2. Email -> IdUtilizador, Username, Password, Data Inscrição, País, Morada, Perfil, NickName, Descrição, Nível
3. Username -> IdUtilizador, Email, Password, Data Inscrição, País, Morada, Perfil, NickName, Descrição, Nível
4. Perfil -> IdUtilizador, Email, Username, Password, Data Inscrição, País, Morada, NickName, Descrição, Nível

Satisfaz 3NF e BCNF

Grupo (IdGrupo, Nome, Abreviatura, Tema, DataCriação)

1. IdGrupo -> Nome, Abreviatura, Tema, DataCriação
2. Nome -> IdGrupo, Abreviatura, Tema, DataCriação
3. Abreviatura -> IdGrupo, Nome, Tema, DataCriação

Satisfaz 3NF e BCNF

Categoria (IdCategoria, Nome)

1. IdCategoria -> Nome
2. Nome -> IdCategoria

Satisfaz 3NF e BCNF

Desenvolvedor (IdDesenvolvedor, Nome)

1. IdDesenvolvedor -> Nome
2. Nome -> IdDesenvolvedor

Satisfaz 3NF e BCNF

Jogo (IdJogo, Título, Classificação, PreçoUnitário, DataLançamento, PEGI, IdDesenvolvedor->Desenvolvedor)

1. IdJogo -> Título, Classificação, PreçoUnitário, DataLançamento, PEGI, IdDesenvolvedor
2. Titulo -> IdJogo, Classificação, PreçoUnitário, DataLançamento, PEGI, IdDesenvolvedor

Satisfaz 3NF e BCNF

Carrinho (IdCarrinho, IdJogo->Jogo, Quantidade)

1. IdCarrinho , IdJogo -> Quantidade

Satisfaz 3NF e BCNF

MetodoPagamento (IdMetodoPagamento, NomeMetodo)

1. IdMetodoPagamento -> NomeMetodo
2. NomeMetodo -> IdMetodoPagamento

Satisfaz 3NF e BCNF

Transação (IdCarrinho->Carrinho, IdJogo->Carrinho, DataTransação, Valor, IdMetodoPagamento->MetodoPagamento, IdUtilizador->Utilizador)

1. IdCarrinho, IdJogo -> Data Transação, Valor, IdMetodoPagamento, IdUtilizador

Satisfaz 3NF e BCNF

Biblioteca (IdJogo->Jogo, IdUtilizador->Utilizador, HorasJogadas, HorasJogadas2semanas)

1. IdUtilizador, IdJogo -> HorasJogadas, HorasJogadas2Semanas

Satisfaz 3NF e BCNF

As tabelas **PertenceGrupo**, **CategoriaJogo** e **Amigo** não possuem dependências funcionais.

A forma normal de Boyce Codd carateriza-se pelo facto de para toda dependencia funcional A->B ou esta é trivial, ou então A é uma superkey. Para além disso para uma tabela se encontrar na forma normal de Boyce Codd é necessário que se encontre na terceira forma normal.

Para estar na terceira forma normal é necessário que a tabela esteja na segunda forma normal e não haja dependências funcionais transitivas.

A segunda forma normal carateriza-se por estar na primeira forma normal e todos os atributos que não são chave(s) da tabela serem dependentes funcionalmente na primary key da tabela.

Em adição, para estar na primeira forma normal é necessário que cada coluna da tabela tenha apenas um valor na mesma, os mesmo tipos de valores guardados e um nome unico.

Efetivamente, como em todas as tabelas apresentadas as colunas têm nomes unicos e as mesmas têm apenas um tipo de valores guardado, assim como apenas um valor por linha, podemos concluir que estão na primeira forma normal. Para além disso, devido ao facto de todos os atributos de todas as tabelas que não são chaves da respetiva tabela são funcionalmente dependentes na primary key da mesma, podemos concluir que todas as tabelas satisfazem a segunda forma normal. Finalmente, como nenhuma tabela possui dependências funcionas transitivas e as dependencias funcionais do tipo A->B possuem sempre um A que é superkey, então podemos concluir que todas as tabelas satisfazem a terceira forma normal e a forma normal de Boyce Codd, como pretendiamos demonstrar.

6 Lista E Forma De Implementação Das Restrições

Tabela: **Utilizador**

- Não podem existir utilizadores com o mesmo ID:
  - IdUtilizador PRIMARY KEY.
- Não podem existir utilizadores com o mesmo Username, Email, Perfil:
  - Username UNIQUE ; Email UNIQUE ; Perfil UNIQUE
- Todos os utilizadores devem possuir um Email, Username, Password, DataInscricao, Pais, Morada, Perfil, Nickname:
  - Username NOT NULL ; Email NOT NULL ; Password NOT NULL ; DataInscricao NOT NULL ; Pais NOT NULL ; Morada NOT NULL ; Perfil NOT NULL ; Nickname NOT NULL
- O nivel padrão de um utilizador novo que ainda não possui jogos é 1:
  - Nivel DEFAULT 1
- A descricao de um utilizador é opcional, assumindo um valor nulo, caso o utilizador não crie uma:
  - Descricao DEFAULT NULL

Tabela: **Grupo**

- Não podem existir grupos com o mesmo ID:
  - IdGrupo PRIMARY KEY
- Não podem existir grupos com o mesmo nome ou abreviatura:
  - Nome UNIQUE ; Abreviatura UNIQUE
- Todos os grupos têm um Nome, uma Abreviatura, um Tema e uma Data de Criação:
  - Nome NOT NULL ; Abreviatura NOT NULL ; Tema NOT NULL ; DataCriacao NOT NULL

Tabela: **Categoria**

- Não podem existir categorias com o mesmo ID:
  - IdCategoria PRIMARY KEY
- Não podem existir categorias com o mesmo Nome:
  - Nome UNIQUE
- Todas as categorias têm um Nome:
  - Nome NOT NULL

Tabela: **Desenvolvedor**

- Não podem existir desenvolvedores com o mesmo ID:

    - IdDesenvolvedor PRIMARY KEY
- Não podem existir desenvolvedores com o mesmo Nome:
    - Nome UNIQUE
- Todas os desenvolvedores têm um Nome:
    - Nome NOT NULL

Tabela: **Jogo**

- Não podem existir jogos com o mesmo ID:
    - IdJogo PRIMARY KEY
- Não podem existir Jogos com o mesmo Titulo:
    - Titulo UNIQUE
- Todos os jogos têm de ter um Titulo, Classificação, Preço Unitário, Data de Lançamento, PEGI, IdDesenvolvedor
    - Titulo NOT NULL ; Classificacao NOT NULL ; PrecoUnitario NOT NULL ; DataLancamento NOT NULL ; PEGI NOT NULL ; IdDesenvolvedor NOT NULL
- O Preço tem de ser maior ou igual a zero (igual a zero se o jogo for grátis/*Free to Play*)
    - CONSTRAINT PrecoValido CHECK (Preco >=0)
- Uma Classificação tem de ser maior ou igual a zero e menor ou igual a cem.
    - CONSTRAINT ClassificacaoValida CHECK (Classificacao>=0 and Classificacao<=100)
- O IdDesenvolvedor deverá corresponder a um Id de uma instância da tabela Desenvolvedor.
    - CONSTRAINT IdDesenvolvedorReferenceJogo FOREING KEY (IdDesenvolvedor) REFERENCES Desenvolvedor

Tabela: **Carrinho**

- Cada carrinho irá ter um Jogo *X,* como tal irá ter um atributo que corresponderá a um Id de uma instância da tabela Jogo.
    - CONSTRAINT IdJogoReferenceCarrinho FOREIGN KEY(IdJogo) REFERENCES Jogo
- Não devem existir duas instâncias com o mesmo par(IdCarrinho,IdJogo)
    - PRIMARY KEY(IdCarrinho, IdJogo)
- Para um Jogo num determinado Carrinho é necessário guardar a sua Quantidade.Como tal, este campo irá sempre ter um valor que não poderá ser nulo.
    - Quantidade NOT NULL

Tabela: **Metodo Pagamento**

- Não podem existir Metodos de Pagamento com o mesmo ID:
    - IdMetodoPagamento PRIMARY KEY
- Não podem existir Metodos de Pagamento com o mesmo Nome:
    - NomeMetodo: UNIQUE
- Todas os Metodos de Pagamento têm um Nome:
    - NomeMetodo: NOT NULL

Tabela: **Transacao**

- Não devem existir duas instâncias com o mesmo par(IdCarrinho, IdJogo)
    - PRIMARY KEY(IdCarrinho, IdJogo)
- Todas as transações têm uma DataTransacao, um Valor, um IdMetodoPagamento e um IdUtilizador.
    - DataTransacao NOT NULL ; Valor NOT NULL ; IdMetodoPagamento NOT NULL ; IdUtilizador NOT NULL
- O IdMetodoPagamento deverá corresponder a um Id de uma instância da tabela MetodoPagamento.
    - CONSTRAINT IdMetodoPagamentoReferenceTransacao FOREIGN KEY (IdMetodoPagamento) REFERENCES MetodoPagamento
- O IdUtilizador deverá corresponder a um Id de uma instância da tabela Utilizador
    - CONSTRAINT IdUtilizadorReferenceTransacao FOREIGN KEY (IdUtilizador) REFERENCES Utilizador
- O par IdCarrinho e IdJogo deverá corresponder a uma instância de um par da tabela carrinho
    - CONSTRAINT IdCarrinhoReferenceTransacao FOREIGN KEY (IdCarrinho, IdJogo) REFERENCES Carrinho
- O Valor da Transação tem der maior ou igual a zero:
    - CONSTRAINT ValidValor CHECK(Valor >= 0)

Tabela: **Biblioteca**

- Não deve haver duas instâncias com o mesmo par (IdUtilizador, IdJogo)
    - CONSTRAINT PrimaryKeyDefinitionBiblioteca PRIMARY KEY (IdUtilizador, IdJogo)
- Um utilizador pode ter comprado um determinado jogo e nunca o ter jogado, assim como não o ter jogado nas últimas duas semanas.
    - HorasJogadas DEFAULT 0 ; HorasJogadas2Semanas DEFAULT 0
- Um utilizador não pode ter mais hora jogadas nas duas últimas semanas que no total e não pode ter mais horas jogadas nas últimas duas semanas do que o tempo total nas duas semanas
    - CONSTRAINT ValidHorasJogadas2Semanas CHECK (HorasJogadas >= HorasJogadas2Semanas and HorasJogadas2Semanas <= (14*24))
- Um utilizador possui um determinado número de horas jogadas totais e horas jogadas nas últimas duas semanas para um certo jogo. Como tal, estes atributos irão formar um par do tipo: “Um certo utilizador tem X horas jogadas totais e Y horas jogadas nas últimas duas semanas de um jogo Z”.
    - CONSTRAINT IdUtilizadorReferenceBiblioteca FOREIGN KEY (IdUtilizador) REFERENCES Utilizador
    - CONSTRAINT IdJogoReferenceBiblioteca FOREIGN KEY (IdJogo) REFERENCES Jogo

Tabela: **CategoriaJogo**

- Não deve haver duas instâncias com o mesmo par (IdJogo,IdCategoria)
    - PRIMARY KEY(IdJogo,IdCategoria)
- O IdCategoria deverá corresponder a um Id de uma instância da tabela Categoria.
    - CONSTRAINT IdCategoriaReferenceCategoriaJogo FOREIGN KEY(IdCategoria) REFERENCES Categoria
- O IdJogo deverá corresponder a um Id de uma instância da tabela Jogo.
    - CONSTRAINT IDJogoReferenceCategoriaJogo FOREIGN KEY(IdJogo) REFERENCES Jogo

Tabela: **Amigo**

- Não deve haver duas instâncias com o mesmo par(IdUtilizador1,IdUtilizador2)
    - PRIMARY KEY(IdUtilizador1,IdUtilizador2)
- O IdUtilizador deverá corresponder a um Id de uma instância da tabela Utilizador.
    - CONSTRAINT IdUtilizadorReferenceUtilizador FOREIGN KEY(IdUtilizador1,IdUtilizador2) REFERENCES Utilizador
- Um utilizador não pode ser amigo de si próprio
    - CONSTRAINT UtilizadorDiferente CHECK(IdUtilizador1 <> IdUtilizador2)

Tabela: **PertenceGrupo**

- Não deve haver duas instâncias com o mesmo par(IdGrupo,IdUtilizador)
    - PRIMARY KEY(IdGrupo, IdUtilizador)
- O IdGrupo deverá corresponder a um Id de uma instância da tabela Grupo.
    - CONSTRAINT IdGrupoReferencePertenceGrupo FOREIGN KEY(IdGrupo) REFERENCES Grupo
- O IdUtilizador deverá corresponder a um Id de uma instância da tabela Utilizador.
    - CONSTRAINT IdUtilizadorReferencePertenceGrupo FOREIGN KEY(IdUtilizador) REFERENCES Utilizador