

Plataforma de jogos

DESIGN BANCO DE DADOS

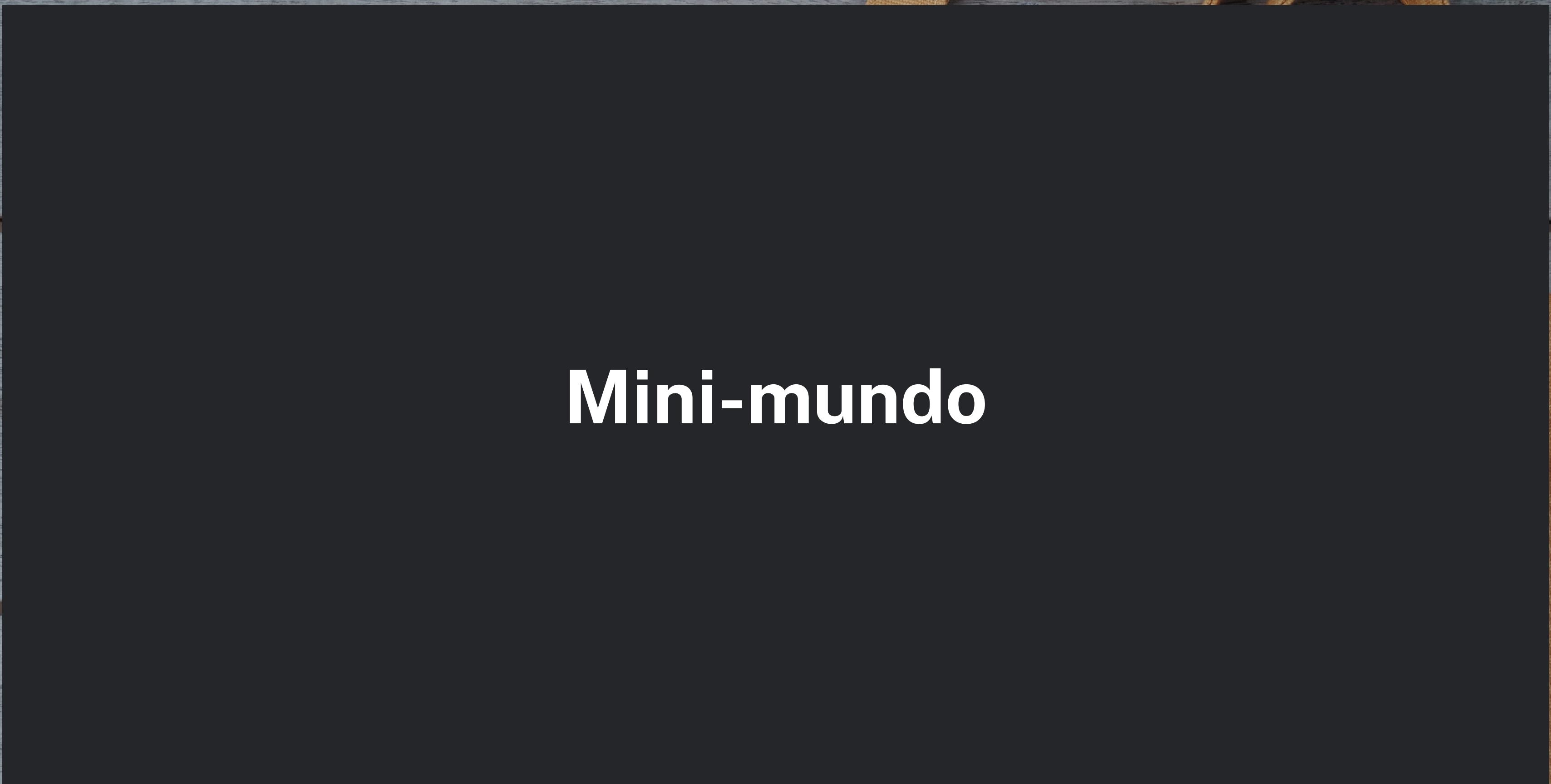
ALUNOS: DANIEL (DSFS); DAVI (DSB6); JOÃO PEDRO (JPBM);
MARCELO (MANBR); RAFAEL (RPF); VINÍCIUS (VLSM)



Sumário

- Mini-mundo
- Projeto Conceitual
- Projeto lógico
- Consultas
- PL-SQL





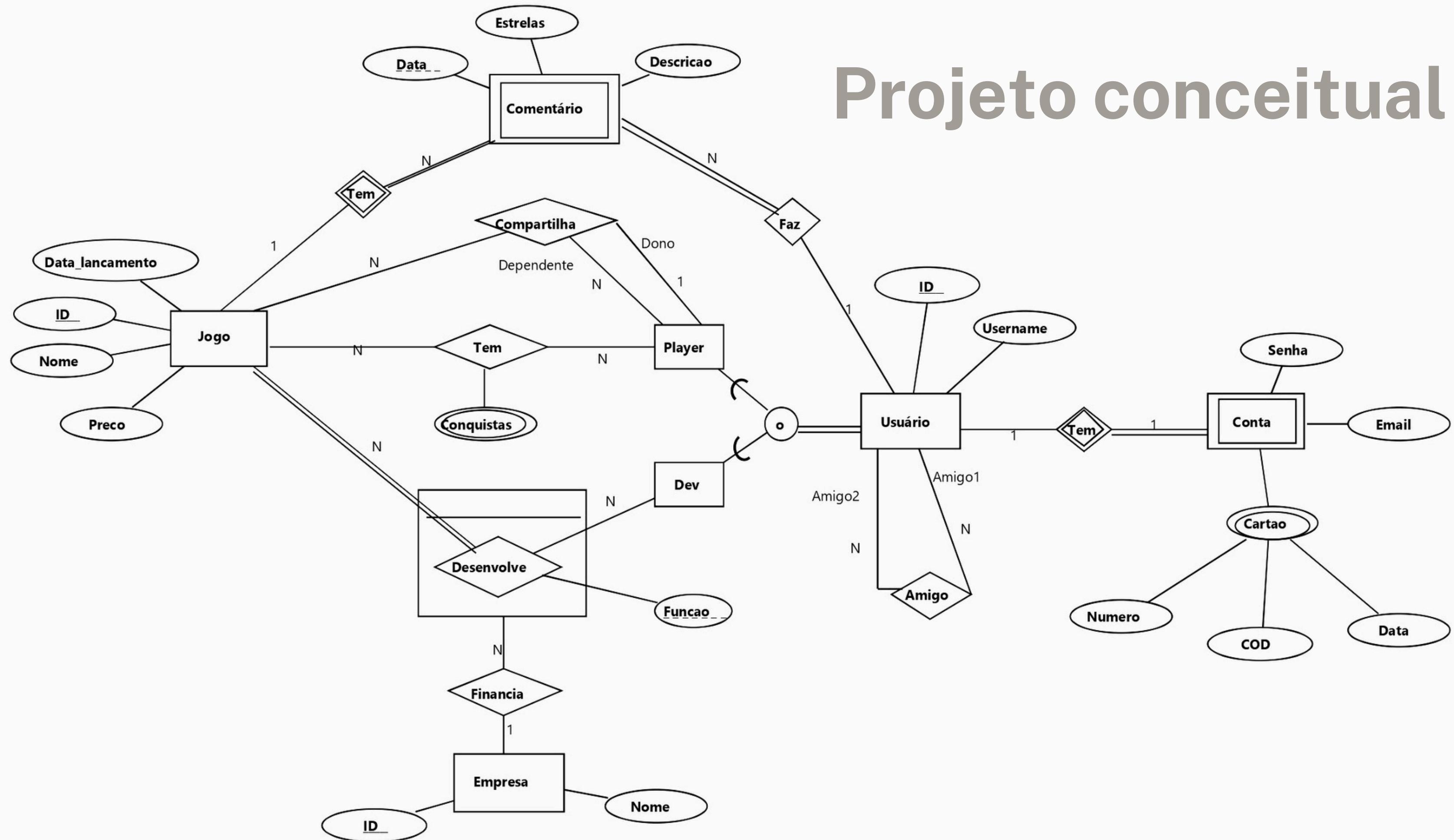
Mini-mundo

Mini-mundo

A ideia central é de uma plataforma que funciona como uma loja virtual de jogos, semelhante à Steam, onde pessoas se cadastram para comprar e jogar títulos diversos. Ao criar uma conta, cada pessoa passa a ser reconhecida como um usuário do sistema, podendo manter uma lista de amigos para interação social. Esses usuários adquirem jogos, que ficam associados à sua biblioteca pessoal, e podem conceder acesso a outros usuários por meio de um recurso de compartilhamento. Para cada jogo, o usuário pode escrever comentários, atribuir notas e registrar suas impressões. Em paralelo, há usuários que atuam como desenvolvedores (ou “devs”), os quais podem criar e lançar novos jogos na plataforma, muitas vezes em parceria com empresas especializadas e podendo ter diferentes atuações no desenvolvimento do mesmo jogo. Assim, o objetivo da plataforma é permitir que as pessoas descubram, comprem e joguem jogos, interajam com amigos, compartilhem experiências, e ainda forneçam espaço para que criadores de conteúdo (desenvolvedores e empresas) lancem seus produtos.

Projeto conceitual

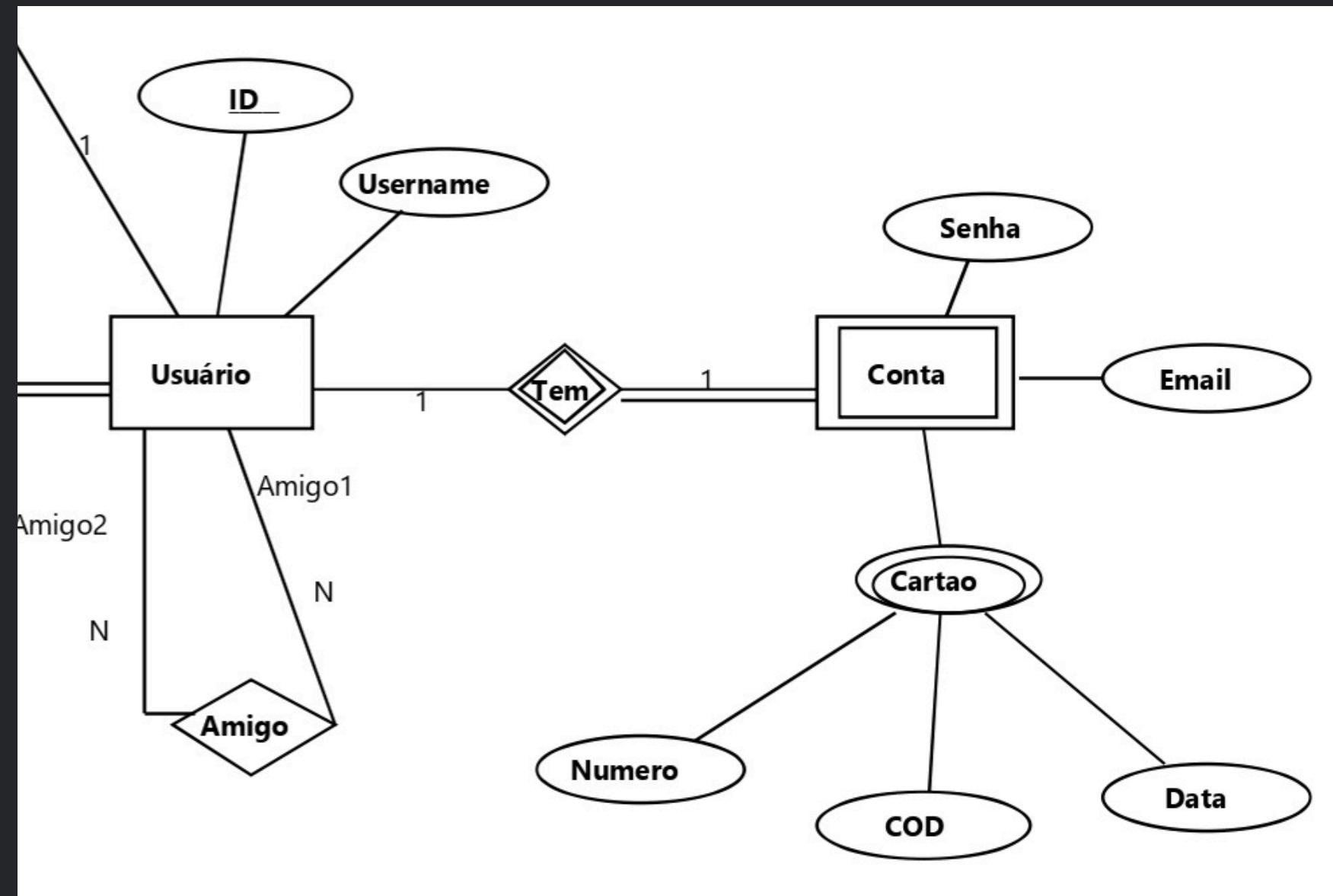
Projeto conceitual



Projeto lógico

1

Usuário | Amigo | Conta | Cartão



Usuário(ID, Username)

Amigo(Amigo1, Amigo2)

Amigo1 -> Usuário(ID)

Amigo2 -> Usuário(ID)

Conta(ID, Senha, Email)

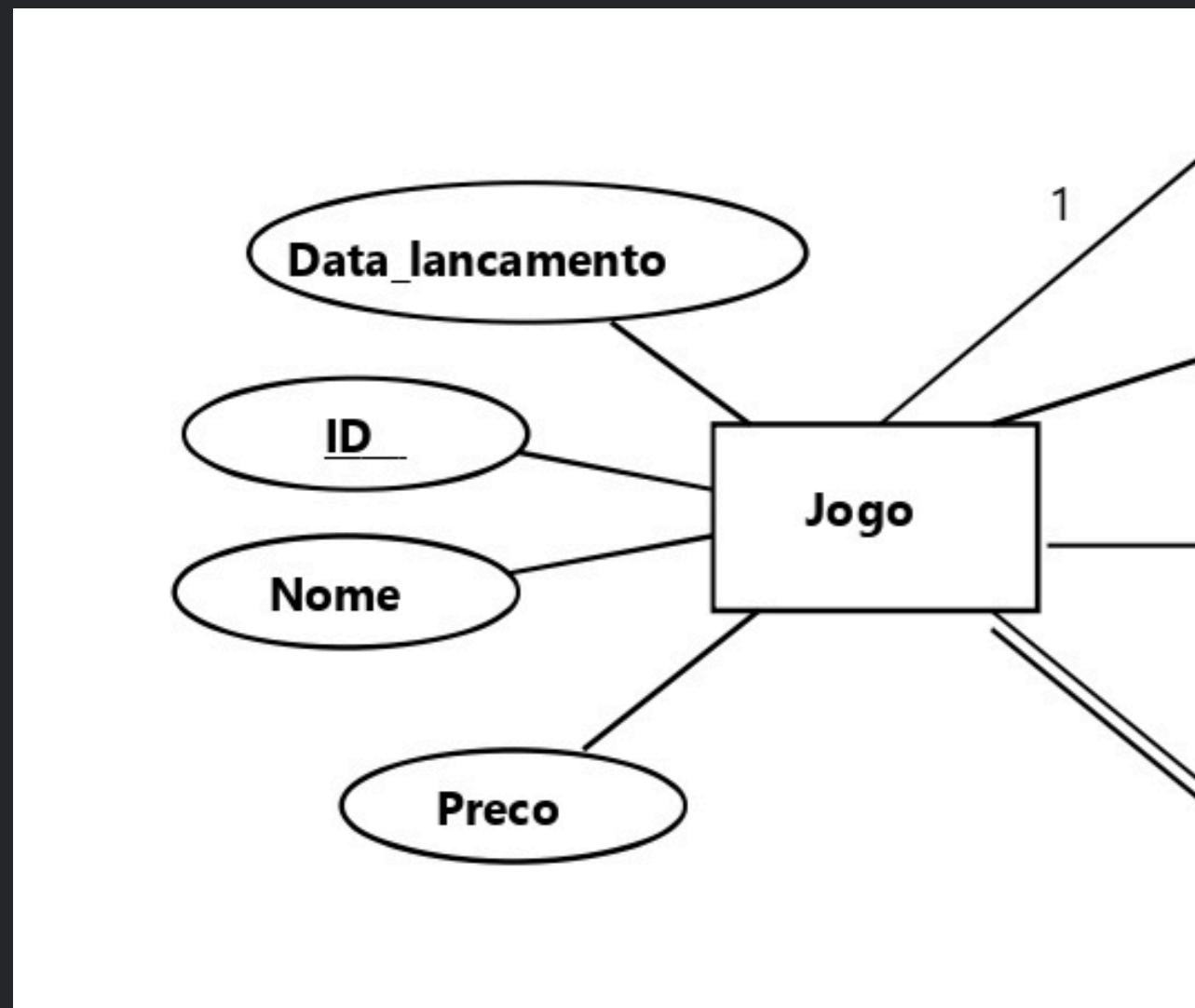
ID -> Usuário(ID)

Cartão(ID, Num, Cod, DT_Exp)

ID -> Conta(ID)

2

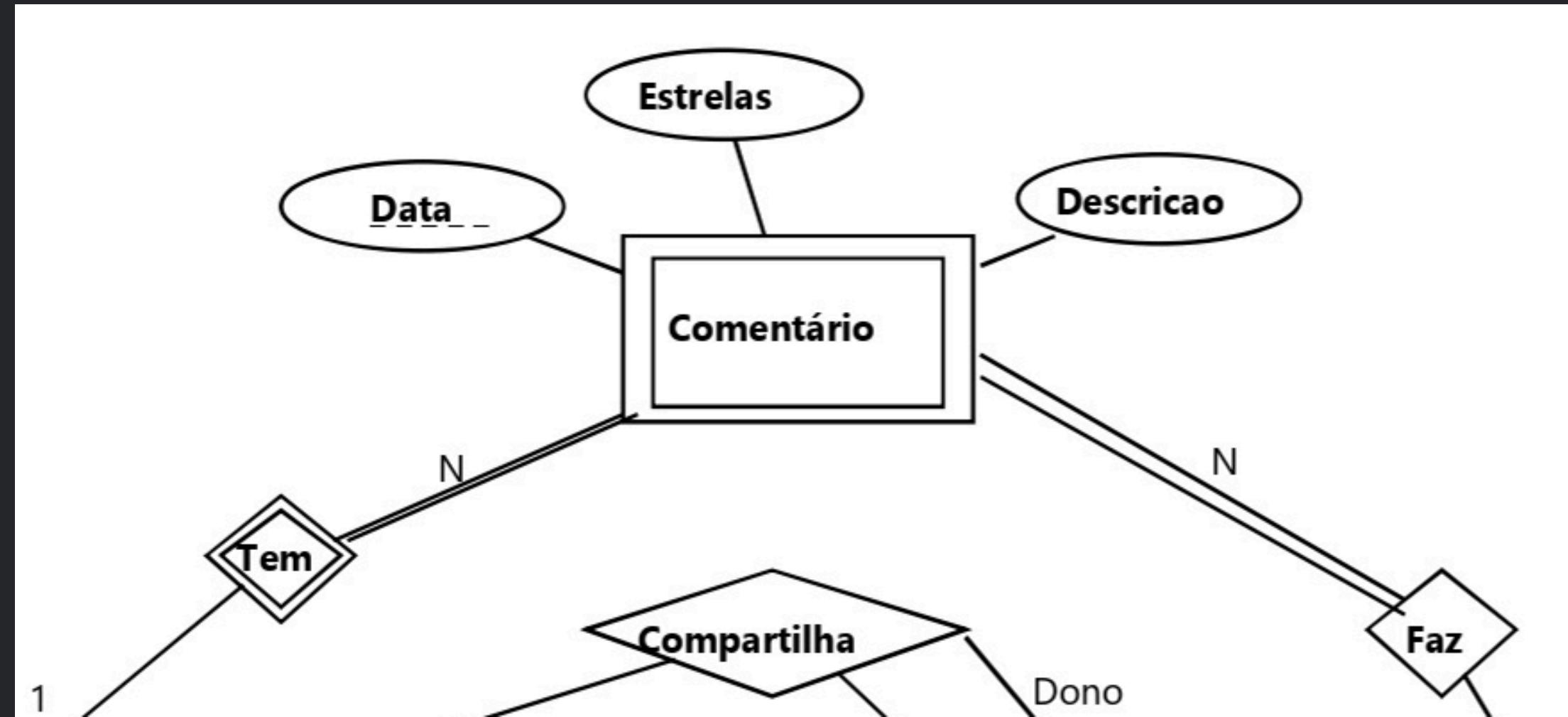
Jogo



Jogo(ID, Nome, DT_Lançamento, Preço)

3

Comentário



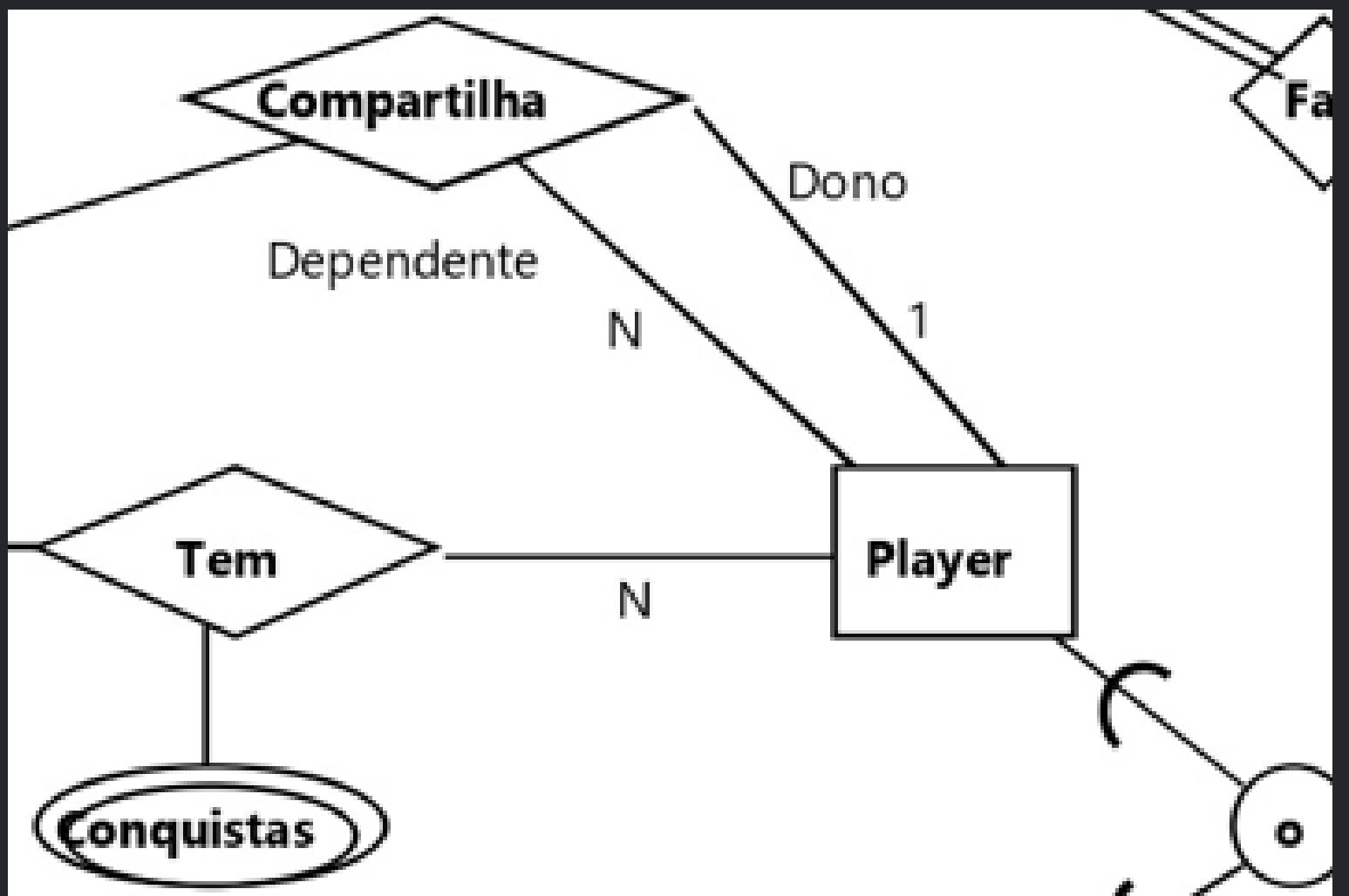
Comentário(Jogo_ID, Data, Estrelas, Descrição, Usuário_ID!)

Jogo_ID -> Jogo(ID)

Usuário_ID -> Usuário(ID)

4

Player | Compartilha | Tem | Conquistas



Player(Usuário_ID)

$\text{Usuário_ID} \rightarrow \text{Usuário(ID)}$

Compartilha(Jogo_ID, Dependente_ID, Dono_ID!)

$\text{Jogo_ID} \rightarrow \text{Jogo(ID)}$

$\text{Dependente_ID} \rightarrow \text{Player(ID)}$

$\text{Dono_ID} \rightarrow \text{Player(ID)}$

Tem(Usuário_ID, Jogo_ID)

$\text{Jogo_ID} \rightarrow \text{Jogo(ID)}$

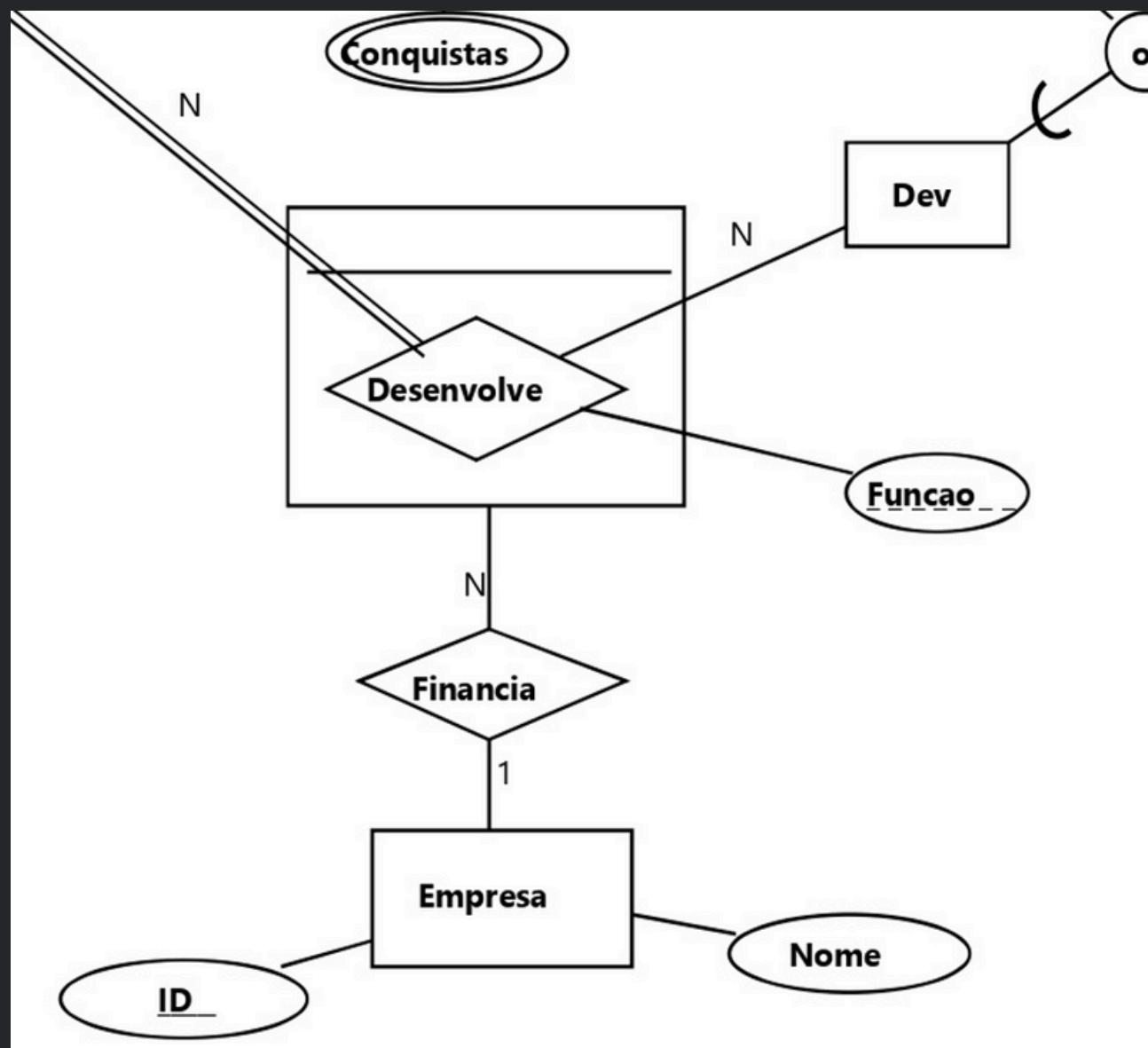
$\text{Usuário_ID} \rightarrow \text{Usuário(ID)}$

Conquistas(Usuário_ID, Jogo_ID, Conquista)

$(\text{Usuário_ID}, \text{Jogo_ID}) \rightarrow \text{Tem}(\text{Usuário_ID}, \text{Jogo_ID})$

5

Dev | Empresa | Desenvolve



Dev(Usuário_ID)

Usuário_ID -> Usuário(ID)

Empresa(ID, nome)

Desenvolve(Usuário_ID, Jogo_ID, Função, Empresa_ID)

Jogo_ID -> Jogo(ID)

Usuário_ID -> Usuário(ID)

Empresa_ID -> Empresa(ID)

Projeto físico

1

Usuário | Amigo | Conta | Cartão

Usuário(ID, Username)

Amigo(Amigo1, Amigo2)

Amigo1 -> Usuário(ID)

Amigo2 -> Usuário(ID)

Conta(ID, Senha, Email)

ID -> Usuário(ID)

Cartão(ID, Num, Cod, DT_Exp)

ID -> Conta(ID)

-- Tabela Usuario

```
CREATE TABLE Usuario (
    ID INT PRIMARY KEY,
    Username VARCHAR(50) NOT NULL
);
```

-- Tabela Amigo

```
CREATE TABLE Amigo (
    Amigo1 INT,
    Amigo2 INT,
    PRIMARY KEY (Amigo1, Amigo2),
    FOREIGN KEY (Amigo1) REFERENCES Usuario(ID),
    FOREIGN KEY (Amigo2) REFERENCES Usuario(ID)
);
```

DT_Exp DATE,

PRIMARY KEY (ID, Num, Cod, DT_Exp),

FOREIGN KEY (ID) REFERENCES Conta(ID)

);

1

Usuário | Amigo | Conta | Cartão

Usuário(ID, Username)

Amigo(Amigo1, Amigo2)

Amigo1 -> Usuário(ID)

Amigo2 -> Usuário(ID)

Conta(ID, Senha, Email)

ID -> Usuário(ID)

Cartão(ID, Num, Cod, DT_Exp)

ID -> Conta(ID)

```
-- Tabela Usuario
CREATE TABLE Usuario (
    ID INT PRIMARY KEY,
    Username VARCHAR(100) NOT NULL,
    Password VARCHAR(100) NOT NULL
);

-- Tabela Conta
CREATE TABLE Conta (
    ID INT PRIMARY KEY,
    Senha VARCHAR(100) NOT NULL,
    Email VARCHAR(100) NOT NULL,
    FOREIGN KEY (ID) REFERENCES Usuario(ID)
);

-- Tabela Cartao
CREATE TABLE Cartao (
    ID INT,
    Num VARCHAR(20),
    Cod INT,
    DT_Exp DATE,
    PRIMARY KEY (ID, Num, Cod, DT_Exp),
    FOREIGN KEY (ID) REFERENCES Conta(ID)
);
```

2

Jogo

Jogo(ID, Nome, DT_Lançamento, Preço)

```
-- Tabela Jogo
CREATE TABLE Jogo (
    ID INT PRIMARY KEY,
    Nome VARCHAR(100) NOT NULL,
    DT_Lancamento DATE NOT NULL,
    Preco DECIMAL(10,2) NOT NULL
);
```

3

Comentário

Comentário(Jogo_ID, Data, Estrelas, Descrição, Usuário_ID!)

Jogo_ID -> Jogo(ID)

Usuário_ID -> Usuário(ID)

```
-- Tabela Comentario
CREATE TABLE Comentario (
    Jogo_ID INT,
    Usuario_ID INT NOT NULL,
    Data DATE,
    Estrelas INT NOT NULL CHECK (Estrelas BETWEEN 0 AND 5),
    Descricao VARCHAR(100),
    PRIMARY KEY (Jogo_ID, Data),
    FOREIGN KEY (Jogo_ID) REFERENCES Jogo(ID),
    FOREIGN KEY (Usuario_ID) REFERENCES Usuario(ID)
);
```

4

Player | Compartilha

```
-- Tabela Player
CREATE TABLE Player (
    Usuário_ID INT PRIMARY KEY,
    FOREIGN KEY (Usuário_ID) REFERENCES Usuário(ID)
);

-- Tabela Compartilha
CREATE TABLE Compartilha (
    Jogo_ID INT,
    Dependente_ID INT,
    Dono_ID INT NOT NULL,
    PRIMARY KEY (Jogo_ID, Dependente_ID),
    FOREIGN KEY (Jogo_ID) REFERENCES Jogo(ID),
    FOREIGN KEY (Dependente_ID) REFERENCES Player(Usuário_ID),
    FOREIGN KEY (Dono_ID) REFERENCES Player(Usuário_ID)
);
```

Player(Usuário_ID)

Usuário_ID -> Usuário(ID)

Compartilha(Jogo_ID, Dependente_ID, Dono_ID!)

Jogo_ID -> Jogo(ID)

Dependente_ID -> Player(ID)

Dono_ID -> Player(ID)

4

Tem | Conquistas

Tem(Usuário_ID, Jogo_ID)

Jogo_ID -> Jogo(ID)

Usuário_ID -> Usuário(ID)

Conquistas(Usuário_ID, Jogo_ID, Conquista)

(Usuário_ID, Jogo_ID) -> Tem(Usuário_ID, Jogo_ID)

```
-- Tabela Tem
CREATE TABLE Tem (
    Usuario_ID INT,
    Jogo_ID INT,
    PRIMARY KEY (Usuario_ID, Jogo_ID),
    FOREIGN KEY (Jogo_ID) REFERENCES Jogo(ID),
    FOREIGN KEY (Usuario_ID) REFERENCES Usuario(ID)
);

-- Tabela Conquistas
CREATE TABLE Conquistas (
    Usuario_ID INT,
    Jogo_ID INT,
    Conquista VARCHAR(100),
    PRIMARY KEY (Usuario_ID, Jogo_ID, Conquista),
    FOREIGN KEY (Usuario_ID, Jogo_ID) REFERENCES Tem(Usuario_ID, Jogo_ID)
);
```

5

Dev | Empresa | Desenvolve

Dev(Usuário_ID)

Usuário_ID -> Usuário(ID)

Empresa(ID, nome)

Desenvolve(Usuário_ID, Jogo_ID, Função, Empresa_ID)

Jogo_ID -> Jogo(ID)

Usuário_ID -> Usuário(ID)

Empresa_ID -> Empresa(ID)

```
-- Tabela Dev
CREATE TABLE Dev (
    Usuario_ID INT PRIMARY KEY,
    FOREIGN KEY (Usuario_ID) REFERENCES Usuario(ID)
);

-- Tabela Empresa
CREATE TABLE Empresa (
    ID INT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL
);

-- Tabela Desenvolve
CREATE TABLE Desenvolve (
    Usuario_ID INT,
    Jogo_ID INT,
    Empresa_ID INT,
    Funcao VARCHAR(100),
    PRIMARY KEY (Usuario_ID, Jogo_ID, Funcao),
    FOREIGN KEY (Usuario_ID) REFERENCES Usuario(ID),
    FOREIGN KEY (Jogo_ID) REFERENCES Jogo(ID),
    FOREIGN KEY (Empresa_ID) REFERENCES Empresa(ID)
);
```

Consultas

1

Quantos jogos foram financiados por alguma empresa?

```
SELECT EMPRESA_ID, COUNT(*) AS QTD  
FROM DESENVOLVE D  
GROUP BY EMPRESA_ID  
HAVING EMPRESA_ID IS NOT NULL;
```

HAVING / GROUP BY

2

Qual o valor que o player de ID 'X' gastou em jogos?

```
--INNER JOIN (QUAL O VALOR TOTAL QUE O PLAYER DE ID 'X' GASTOU EM JOGOS)
SELECT P.USUARIO_ID, SUM(J.PRECO) AS TOTAL_GASTO
FROM PLAYER P INNER JOIN
      TEM T ON T.USUARIO_ID = P.USUARIO_ID INNER JOIN
      JOGO J ON J.ID = T.JOGO_ID
GROUP BY P.USUARIO_ID;
```

INNER JOIN

3

Nome das empresas que nunca financiaram jogos.

```
SELECT E.NOME  
FROM EMPRESA E LEFT OUTER JOIN  
DESENVOLVE D ON E.ID = D.EMPRESA_ID  
WHERE D.JOGO_ID IS NULL;
```

OUTER JOIN

4

Nome dos desenvolvedores que desenvolveram jogos.

SEMI JOIN

```
SELECT U.USERNAME  
FROM DEV D INNER JOIN  
    USUARIO U ON D.USUARIO_ID = U.ID  
WHERE EXISTS (SELECT * FROM DESENVOLVE DE WHERE DE.USUARIO_ID = U.ID);
```

5

Nome dos jogos que não foram comprados por algum player.

ANTI JOIN

```
SELECT J.NOME  
FROM JOGO J  
WHERE NOT EXISTS (SELECT * FROM TEM T WHERE T.JOGO_ID = J.ID);
```

6

Qual nome dos jogos que são mais caros que a média de preços?

SUBCONSULTA DO TIPO ESCALAR

```
SELECT J.NOME  
FROM JOGO J  
WHERE J.PRECO > (SELECT AVG(PRECO) FROM JOGO);
```

7

Qual nome dos jogos que foram lançados na mesma data que o jogo de id '101'?

SUBCONSULTA DO TIPO LINHA

```
--SUBCONSULTA DO TIPO LINHA (QUAIS OS NOMES DOS JOGOS QUE FORAM LANÇADOS NO MESMO MES E ANO QUE O JOGO DE ID '101'  
SELECT J.NOME  
FROM JOGO J  
WHERE (TO_CHAR(J.DT_LANCAMENTO, 'MM'), TO_CHAR(J.DT_LANCAMENTO, 'YY'))  
      = (SELECT TO_CHAR(DT_LANCAMENTO, 'MM'), TO_CHAR(DT_LANCAMENTO, 'YY') FROM JOGO WHERE ID = '101');
```

8

Qual o nome dos jogos compartilhados entre o mesmo par de dono (ID ‘1111’) e dependente (ID ‘1112’)?

SUBCONSULTA DO TIPO TABELA

```
SELECT J.NOME  
FROM JOGO J  
WHERE J.ID IN (SELECT C.JOGO_ID FROM COMPARTILHA C WHERE C.DONO_ID = '1111' AND CDEPENDENTE_ID = '1112');
```

9

Qual o ID dos players que também são desenvolvedores?

OPERAÇÃO DE CONJUNTO

```
--OPERAÇÃO DE CONJUNTO (QUAIS OS ID'S DOS PLAYERS QUE TAMBÉM SÃO DEV'S)
SELECT * FROM (SELECT USUARIO_ID FROM DEV) INTERSECT (SELECT USUARIO_ID FROM PLAYER);
```

PL-SQL

1

Procedimento para cadastrar novo usuário

```
CREATE OR REPLACE PROCEDURE CADASTRAR_USUARIO(idPlayer NUMBER, name VARCHAR, isDev NUMBER) IS
BEGIN
    INSERT INTO USUARIO (ID, USERNAME)
        VALUES (idPlayer, name);

    IF (isDev = 1 OR isDev = 2) THEN
        INSERT INTO DEV (USUARIO_ID)
            VALUES (idPlayer);
    END IF;
    IF (isDev = 0 OR isDev = 2) THEN
        INSERT INTO PLAYER (USUARIO_ID)
            VALUES (idPlayer);
    END IF;

    DBMS_OUTPUT.PUT_LINE('USUARIO ' || name || ' CADASTRADO COM SUCESSO');
END;
```

Procedimento que reduz o preço de jogos antigos

```
CREATE OR REPLACE PROCEDURE ABAIXAR_PRECO IS
    CURSOR cur_jogo IS
        SELECT ID, PRECO, EXTRACT(YEAR FROM (SELECT SYSDATE FROM DUAL)) - EXTRACT(YEAR FROM DT_LANCAMENTO) AS IDADE
        FROM JOGO;
    reg_jogo cur_jogo%ROWTYPE;
BEGIN
    OPEN cur_jogo;

    LOOP
        FETCH cur_jogo INTO reg_jogo;
        EXIT WHEN cur_jogo%NOTFOUND;
        IF (reg_jogo.PRECO > 250 AND reg_jogo.IDADE = 1) THEN
            UPDATE JOGO
            SET PRECO = 250
            WHERE ID = reg_jogo.id;
            DBMS_OUTPUT.PUT_LINE('PREÇO REAJUSTADO PARA 250 DO JOGO ' || reg_jogo.id);
        ELSIF (reg_jogo.PRECO > 200 AND reg_jogo.IDADE = 2) THEN
            UPDATE JOGO
            SET PRECO = 200
            WHERE ID = reg_jogo.id;
            DBMS_OUTPUT.PUT_LINE('PREÇO REAJUSTADO PARA 200 DO JOGO ' || reg_jogo.id);
        ELSIF (reg_jogo.PRECO > 150 AND reg_jogo.IDADE = 3) THEN
            UPDATE JOGO
            SET PRECO = 150
            WHERE ID = reg_jogo.id;
            DBMS_OUTPUT.PUT_LINE('PREÇO REAJUSTADO PARA 150 DO JOGO ' || reg_jogo.id);
        ELSIF (reg_jogo.PRECO > 100 AND reg_jogo.IDADE >= 4) THEN
            UPDATE JOGO
            SET PRECO = 100
            WHERE ID = reg_jogo.id;
            DBMS_OUTPUT.PUT_LINE('PREÇO REAJUSTADO PARA 100 DO JOGO ' || reg_jogo.id);
        END IF;
    END LOOP;

    CLOSE cur_jogo;
END;
```

2

Procedimento que reduz o preço de jogos antigos

```
CREATE OR REPLACE PROCEDURE ABAIXAR_PRECO IS
    CURSOR cur_jogo IS
        SELECT ID, PRECO, EXTRACT(YEAR FROM (SELECT SYSDATE FROM DUAL)) - EXTRACT(YEAR FROM DT_LANCAMENTO) AS IDADE
        FROM JOGO;
    reg_jogo cur_jogo%ROWTYPE;
BEGIN
    OPEN cur_jogo;

LOOP
    FETCH cur_jogo INTO reg_jogo;
    EXIT WHEN cur_jogo%NOTFOUND;
    IF (reg_jogo.PRECO > 250 AND reg_jogo.IDADE = 1) THEN
        UPDATE JOGO
        SET PRECO = 250
        WHERE ID = reg_jogo.id;
        DBMS_OUTPUT.PUT_LINE('PREÇO REAJUSTADO PARA 250 DO JOGO ' || reg_jogo.id);
    ELSIF (reg_jogo.PRECO > 200 AND reg_jogo.IDADE = 2) THEN
        UPDATE JOGO
        SET PRECO = 200
        WHERE ID = reg_jogo.id;
    END IF;
END;
```

```
RETRN cur_jogo INTO reg_jogo;
EXIT WHEN cur_jogo%NOTFOUND;
IF (reg_jogo.PRECO > 250 AND reg_jogo.IDADE = 1) THEN
    UPDATE JOGO
    SET PRECO = 250
    WHERE ID = reg_jogo.id;
    DBMS_OUTPUT.PUT_LINE('PREÇO REAJUSTADO PARA 250 DO JOGO ' || reg_jogo.id);
ELSIF (reg_jogo.PRECO > 200 AND reg_jogo.IDADE = 2) THEN
    UPDATE JOGO
    SET PRECO = 200
    WHERE ID = reg_jogo.id;
    DBMS_OUTPUT.PUT_LINE('PREÇO REAJUSTADO PARA 200 DO JOGO ' || reg_jogo.id);
ELSIF (reg_jogo.PRECO > 150 AND reg_jogo.IDADE = 3) THEN
    UPDATE JOGO
    SET PRECO = 150
    WHERE ID = reg_jogo.id;
    DBMS_OUTPUT.PUT_LINE('PREÇO REAJUSTADO PARA 150 DO JOGO ' || reg_jogo.id);
ELSIF (reg_jogo.PRECO > 100 AND reg_jogo.IDADE >= 4) THEN
    UPDATE JOGO
    SET PRECO = 100
    WHERE ID = reg_jogo.id;
    DBMS_OUTPUT.PUT_LINE('PREÇO REAJUSTADO PARA 100 DO JOGO ' || reg_jogo.id);
END IF;
END LOOP;

CLOSE cur_jogo;
END;
```

3

Função para contratar dev e retorna a nova quantidade de devs

```
CREATE OR REPLACE FUNCTION CONTRATAR_DEV (codDEV NUMBER, idJOGO NUMBER, idEMPRESA NUMBER, func VARCHAR2) RETURN NUMBER IS
    new_dev NUMBER;
    cod_empresa NUMBER;
    cod_jogo NUMBER;
    qtd_dev NUMBER;
BEGIN
    SELECT USUARIO_ID INTO new_dev
    FROM DEV
    WHERE USUARIO_ID = codDEV;

    SELECT ID INTO cod_empresa
    FROM EMPRESA
    WHERE ID = idEMPRESA;

    SELECT ID INTO cod_jogo
    FROM JOGO
    WHERE ID = idJOGO;

    INSERT INTO DESENVOLVE VALUES (new_dev,idJogo,idEmpresa,func);
    DBMS_OUTPUT.PUT_LINE('DESENVOLVEDOR CONTRATADO');

    SELECT COUNT(QTD) INTO qtd_dev
    FROM (SELECT COUNT(*) AS QTD
          FROM DESENVOLVE
          WHERE EMPRESA_ID = cod_empresa
          GROUP BY USUARIO_ID);

    RETURN qtd_dev;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('UM DOS CAMPOS INFORMADOS NÃO EXISTE');
        RETURN NULL;
END;
```

3

Função para contratar dev e retorna a nova quantidade de devs

```
CREATE OR REPLACE FUNCTION CONTRATAR_DEV (codDEV NUMBER, idJOGO NUMBER, idEMPRESA NUMBER, func VARCHAR2) RETURN NUMBER IS
    new_dev NUMBER;
    cod_empresa NUMBER;
    cod_jogo NUMBER;
    qtd_dev NUMBER;
BEGIN
    SELECT USUARIO_ID INTO new_dev
    FROM DEV
    WHERE USUARIO_ID = codDEV;

    SELECT ID INTO cod_empresa
    FROM EMPRESA
    WHERE ID = idEMPRESA;

    SELECT ID INTO cod_jogo
    FROM JOGO
    WHERE ID = idJOGO;

    INSERT INTO DESENVOLVE VALUES (new_dev,idJogo,idEmpresa,func);
    DBMS_OUTPUT.PUT_LINE('DESENVOLVEDOR CONTRATADO');

    SELECT COUNT(OTD) INTO qtd_dev
    FROM DEV
    WHERE USUARIO_ID = new_dev;
```

3

```
SELECT USUARIO_ID INTO new_dev
FROM DEV
WHERE USUARIO_ID = codDEV;

SELECT ID INTO cod_empresa
FROM EMPRESA
WHERE ID = idEMPRESA;

SELECT ID INTO cod_jogo
FROM JOGO
WHERE ID = idJOGO;

INSERT INTO DESENVOLVE VALUES (new_dev,idJogo,idEmpresa,func);
DBMS_OUTPUT.PUT_LINE('DESENVOLVEDOR CONTRATADO');

SELECT COUNT(QTD) INTO qtd_dev
FROM (SELECT COUNT(*) AS QTD
      FROM DESENVOLVE
      WHERE EMPRESA_ID = cod_empresa
      GROUP BY USUARIO_ID);

RETURN qtd_dev;

EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('UM DOS CAMPOS INFORMADOS NÃO EXISTE');
    RETURN NULL;
END;
```

4

Função que analisa comentário e retorna qualidade do jogo

```
CREATE OR REPLACE FUNCTION MEDIA_ESTRELAS(ID_JOGO INT) RETURN VARCHAR2 IS
    MEDIA NUMBER;

BEGIN
    SELECT AVG(ESTRELAS) INTO MEDIA
    FROM COMENTARIO
    GROUP BY JOGO_ID
    HAVING ID_JOGO = JOGO_ID;

    IF (MEDIA > 4.5) THEN RETURN 'MUITO BOM';
    ELSIF (MEDIA > 4) THEN RETURN 'BOM';
    ELSIF (MEDIA > 3) THEN RETURN 'MÉDIO';
    ELSE RETURN 'RUIM';
    END IF;
END;
```

5

Trigger para quando o cartão for cadastrado com número repetido

```
CREATE OR REPLACE TRIGGER CARTAO_REPEATIDO
BEFORE INSERT ON CARTAO
FOR EACH ROW
DECLARE
    cartoes_repetidos NUMBER;
BEGIN
    SELECT COUNT(*) INTO cartoes_repetidos
    FROM CARTAO
    WHERE NUM = :NEW.NUM AND ID = :NEW.ID;

    IF cartoes_repetidos > 0 THEN
        RAISE_APPLICATION_ERROR(-20003, 'ESSE CARTAO JÁ FOI CADASTRADO');
    END IF;
END;
```

Trigger que limita o compartilhamento de jogos

```
CREATE OR REPLACE TRIGGER LIMITE_COMPARTILHAMENTO
BEFORE INSERT ON COMPARTILHA
FOR EACH ROW
DECLARE
    qtdComp NUMBER;
BEGIN
    SELECT COUNT(*) INTO qtdComp
    FROM COMPARTILHA
    WHERE DONO_ID = :NEW.DONO_ID AND JOGO_ID = :NEW.JOGO_ID;

    IF (qtdComp >= 5) THEN
        RAISE_APPLICATION_ERROR(-20001, 'LIMITE MÁXIMO DE COMPARTILHAMENTOS ATINGIDO');
    END IF;
END;
```

Obrigado!