Apresentação do Trabalho Final de Bases de Dados

Sistema de Distribuição de VideoJogos

Trabalho realizado por :

- Daniel Gomes, nºmec 93015
- Bruno Bastos, nºmec 93302

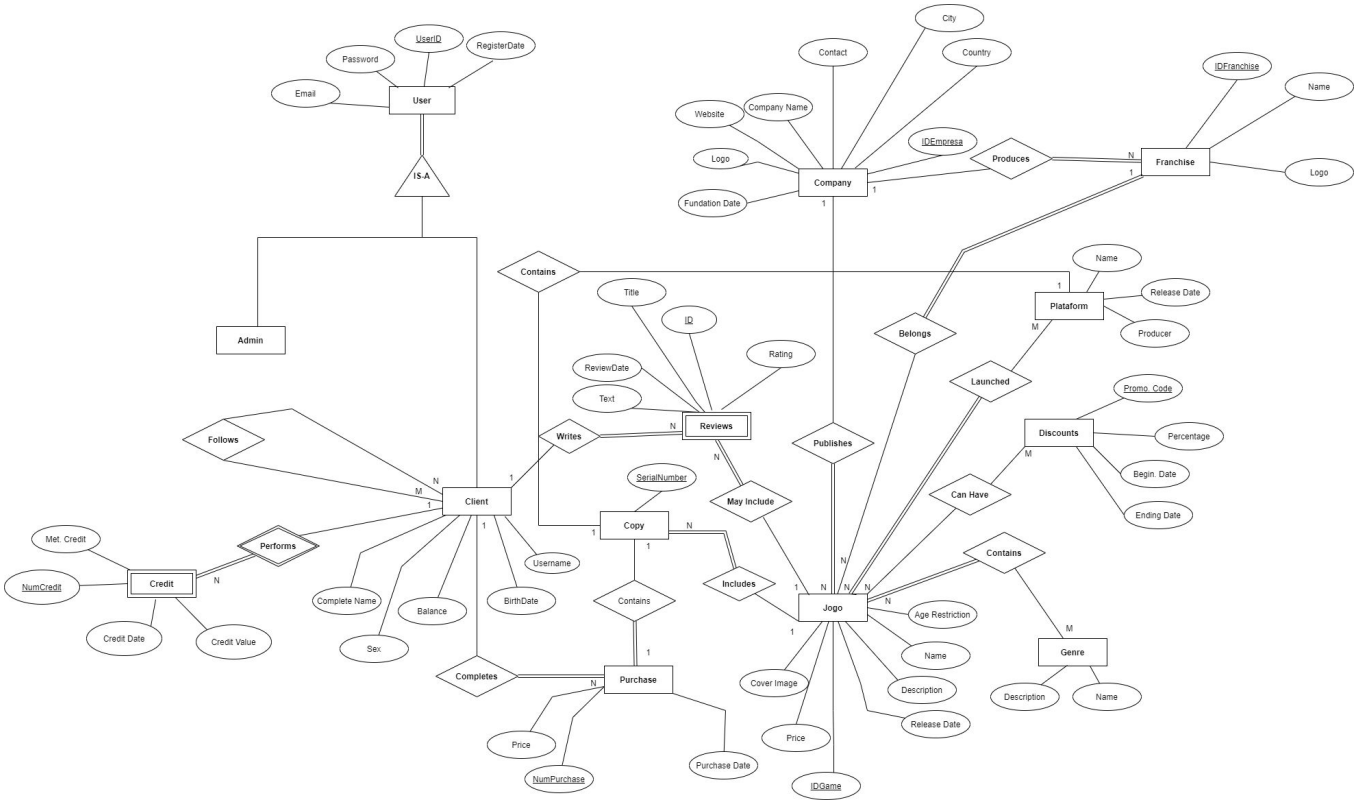
Tema

- Sistema de Gestão de distribuição de VideoJogos.
- Foco no desenvolvimento e desenho da camada de base de dados.
- Pretende-se que clientes, já registados, tenham a possibilidade de realizar compras de Jogos, visualizar todas as informações acerca destes, guardando assim os mais diversos dados e registos.
- Permitida a Escrita de Reviews sobre os VideoJogos.
- Possibilidade de clientes seguirem outros e serem seguidos

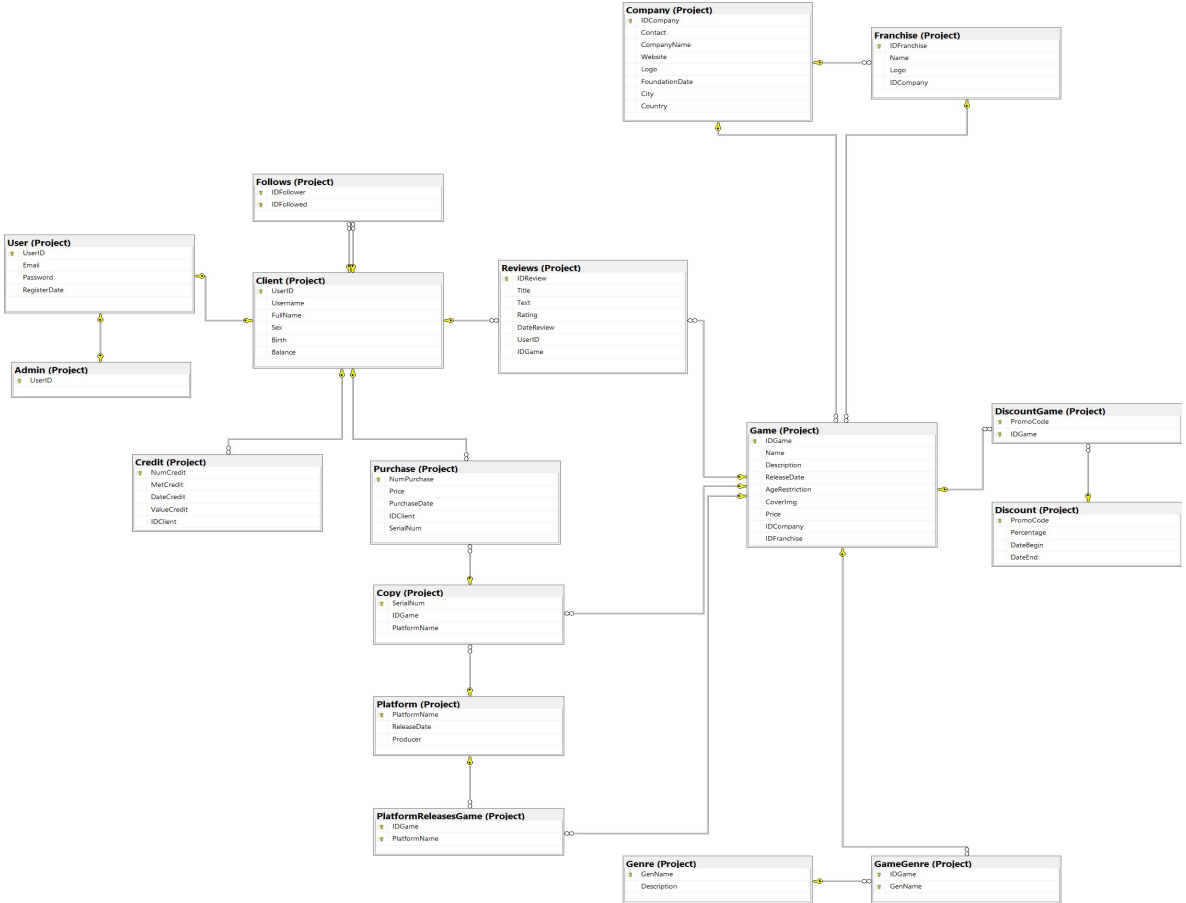
Desenho Conceptual

- Diagrama Entidade-Relação (DER)
- Esquema Relacional (ER)

Diagrama Entidade- Relação



Esquema Relacional (SQL-Server)



SQL Programming

- Stored Procedures
- User Defined Functions
- Triggers

Stored Procedures

- Utilizadas tendo em vista a utilização dos dados da Aplicação de uma forma mais segura
- Adicionar Jogos,Géneros, Plataformas,etc
- Remover descontos associados a um Jogo, Género associados a um Jogo, deixar de seguir um Cliente...
- Filtrar por Género,Preço,Franchise,data de lançamento,desconto, nome de Jogo,entre outros.
- Atualizar algumas informações da conta do Cliente, das Empresas, Franchises..
- Efetuar Login/Register

Stored Procedures

```
CREATE PROCEDURE Project.pd_filter_CreditHistory(
    @IDClient INT,
    @MinValue DECIMAL(5,2),
    @MaxValue DECIMAL(5,2),
    @MinDate DATE,
    @MaxDate DATE,
    @selectedMets VARCHAR(max) -- selected payment methods in the app checkbox
    )
    AS
        BEGIN
              DECLARE @temp TABLE (
               MetCredit VARCHAR(20),
               ValueCredit DECIMAL(5,2),
               DateCredit DATE)
               INSERT INTO @temp SELECT MetCredit,ValueCredit,DateCredit FROM Project.Credit where Project.Credit.IDClient =@IDClient
               IF @MinValue is not null
                   DELETE FROM @temp WHERE @MinValue>ValueCredit
               IF @MaxValue is not null
                   DELETE FROM @temp WHERE @MaxValue<ValueCredit
               IF @MinDate is not  null
                   DELETE FROM @temp WHERE DATEDIFF(DAY,@MinDate,DateCredit) < 0
               IF @MaxDate is not null
                   DELETE FROM @temp WHERE DATEDIFF(DAY,@MaxDate,DateCredit) > 0
               IF @selectedMets is not null
                   DELETE FROM @temp WHERE  MetCredit NOT  IN (SELECT value FROM STRING_SPLIT(@selectedMets, ','));

               SELECT * FROM @temp
        END
```

User Defined Functions (UDF)

- Utilizadas para processar os pedidos de leitura à base de Dados.
- Verificar Cópias disponíveis, Descontos válidos …
- Retornar Jogos de Um Cliente, detalhes de uma Compra,etc
- Estatísticas para o Administrador da Plataforma: jogos mais vendidos, total de Revenue com Jogos, Géneros mais Comprados.

UDFs

```
CREATE FUNCTION Project.[udf_checkGameDiscount] (@IDGame INT) RETURNS TABLE
AS
    RETURN (SELECT [Percentage] AS [Percentage] FROM Project.Game
    JOIN Project.DiscountGame ON Game.IDGame =DiscountGame.IDGame
    JOIN Project.Discount ON Discount.PromoCode =DiscountGame.PromoCode
    WHERE DATEDIFF(DAY,DateEnd,GETDATE()) <0  AND DATEDIFF(DAY,DateBegin,GETDATE()) >0 AND Game.IDGame=@IDGame)

GO
```

```
CREATE FUNCTION Project.[udf_mostSoldGames]() RETURNS TABLE
AS
    RETURN (SELECT  top 1000 COUNT(Game.IDGame) as CountPurchases,Game.IDGame, Game.[Name], SUM(Purchase.Price) as Revenue FROM Project.Purchase
    JOIN Project.[Copy] ON [Copy].SerialNum=Purchase.SerialNum
    JOIN Project.Game ON Game.IDGame = Copy.IDGame GROUP BY Game.IDGame,Game.Name,Purchase.Price ORDER BY CountPurchases DESC )

go
CREATE FUNCTION Project.udf_getTotalMoney() RETURNS TABLE
    RETURN((select SUM(Revenue) AS totMoney FROM Project.udf_mostSoldGames()))
go
CREATE FUNCTION Project.[udf_leastSoldGames]() RETURNS TABLE
AS
    RETURN (SELECT  top 1000 COUNT(Game.IDGame) as CountPurchases,Game.IDGame, Game.[Name] FROM Project.Purchase
    JOIN Project.[Copy] ON [Copy].SerialNum=Purchase.SerialNum
    JOIN Project.Game ON Game.IDGame = Copy.IDGame GROUP BY Game.IDGame,Game.Name ORDER BY CountPurchases ASC )
```

Triggers

- Utilizados com o objetivo de Garantir que as operações de escrita sobre a Base de Dados eram válidas.
- Triggers instead of
- Adicionar Jogos,Franchises,Descontos,Compras…

Triggers

```
CREATE TRIGGER Project.trigger_purchase ON Project.Purchase
INSTEAD OF INSERT
AS
    BEGIN
            DECLARE @Price  AS DECIMAL(5,2);
            DECLARE @PurchaseDate AS DATE;
            DECLARE @IDClient AS INT;
            DECLARE @SerialNum AS INT;
            DECLARE @IDGame AS INT;
            DECLARE @tempPer AS INT;
            SELECT * FROM inserted
            SELECT @Price = Price , @PurchaseDate=PurchaseDate,@IDClient=IDClient,@SerialNum=SerialNum FROM INSERTED;
            if @Price - (SELECT Balance FROM Project.Client WHERE Client.UserID =@IDClient) >0
            BEGIN
                raiserror('Not enough balance to buy this game',16,1)

            END
            ELSE
                BEGIN TRY
                    UPDATE Project.Client
                    SET Balance-=@Price WHERE Project.Client.UserID=@IDClient
                    INSERT INTO Project.Purchase(Price,PurchaseDate,IDClient,SerialNum) VALUES (@Price,@PurchaseDate,@IDClient,@SerialNum)

                END TRY
                BEGIN CATCH
                 raiserror ('Error while inserting purchase values', 16, 1);
                END CATCH
    END
```

Transactions

- Utilizadas quando se envolvia a inserção/atualização de dados em diversas tabelas, assim poderíamos dar *rollback* a operação inteira, caso ocorresse um erro.
- Adicionar Jogos, Crédito à conta, uma Compra, Admin

Transactions

```
CREATE PROCEDURE Project.pd_insertPurchase(
    @PurchaseDate DATE,
    @IDClient INT,
    @IDGame INT,
    @PlatformName VARCHAR(30),
    @res VARCHAR(35) output
    )
    AS
        BEGIN
            BEGIN TRAN
            DECLARE @Price DECIMAL(5,2)
            DECLARE @countDispCopies INT;
            DECLARE @SerialNum INT;
            DECLARE @tempPer DECIMAL(5,2);
            BEGIN TRY
            IF ( (SELECT Project.udf_checkUserPurchase(@IDClient,@IDGame)) = 1)
                raiserror ('User Already Contains that Game',16,1);

            SET @Price = (SELECT Price from Project.Game WHERE Game.IDGame=@IDGame)
            SET @SerialNum= (SELECT top 1 notBought FROM Project.[udf_checkGameCopies] (@IDGame,@PlatformName))
            IF @SerialNum IS NULL
                BEGIN
                    INSERT INTO Project.Copy VALUES (@IDGame,@PlatformName)
                    SET @SerialNum =( SELECT TOP 1 Copy.SerialNum FROM Project.[Copy] WHERE Copy.IDGame=@IDGame AND Copy.PlatformName=@PlatformName ORDER BY SerialNum DESC  )
                END

            SET @tempPer = (SELECT TOP 1 * FROM Project.[udf_checkGameDiscount](@IDGame))
                IF @tempPer is not null
                    BEGIN
                        SET @Price-=@Price*(@tempPer/100)
                    END
            INSERT INTO Project.Purchase(Price,PurchaseDate,IDClient,SerialNum) VALUES (@Price,@PurchaseDate,@IDClient,@SerialNum)
            SET @res ='Success!'
            END TRY
            BEGIN CATCH
                SET @res= ERROR_MESSAGE()
                rollback tran
            END CATCH
            if @@TRANCOUNT >0
            COMMIT TRAN
        END
```

Segurança

- Utilização de SQL Parametrizado, evitando por isso o uso de Queries ad-hoc
- Validação dos formulários da Aplicação
- Apresentação de Erros Customizados
- Encriptação de Palavras-Passe utilizando a função do SQL-Server , ENCRYPTBYPASSPHRASE
- Realização de alguns testes de SQL Injection

Demo