universidade de aveiro

23/06/2022

VideoClube

Base de Dados

Grupo P2G7
Frederico Vieira, Nmec:98518
Tiago Coelho, Nmec:98385

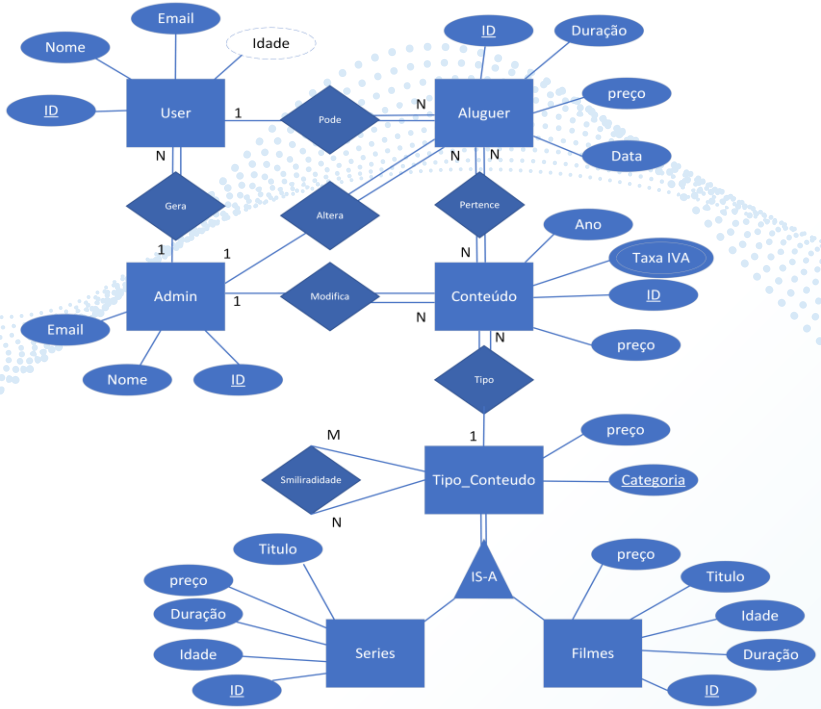
Introdução

Nos videoclubes físicos que existiam, havia sempre vários problemas:

- Continha apenas alguns exemplares dos filmes;
- Atrasos na entrega do filme;
- Regressar com algum dano de tal forma que não funcionará mais;

Existe assim uma transformação digital para uma plataforma streaming de aluguer de filmes e series, surge uma oportunidade de corrigir todos esses problemas indicados e adicionar mais funcionalidades que possam ser úteis ao utilizador.

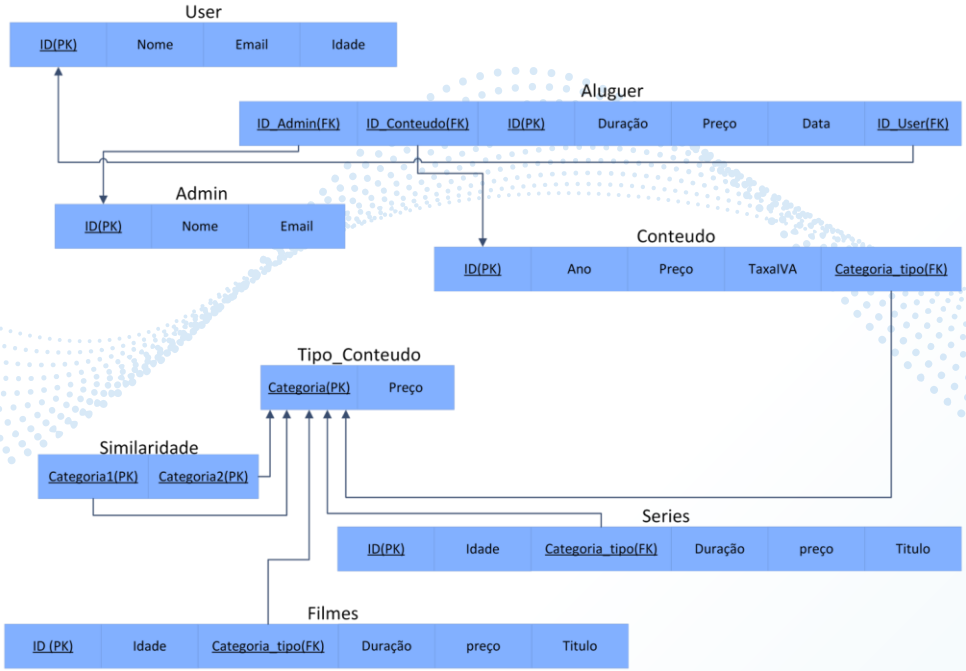
Diagrama Entidade/Relacionamento



Alterações:

1. Remoção de atributos duplicados;
2. Preço como derivado;
3. Adição de Entidades para cupões e promoções;
4. Adição do atributo pontos;

Esquema Relacional



Alterações:

1. Alteração do Conteudo_Taxa;

2. Remoção de FK's desnecessárias;

3. Adição das devidas tabelas em relação às promoções e cupões;

1.Funcionalidades

Casos de utilização

Funcionalidades

Utilizador pode:

- Criar Conta e ligar-se na mesma;
- Procurar um filme/série e alugar o mesmo;
- Pesquisar por um filme/série;
- Ordenar por preço, ano e título;
- Filtrar por categorias e/ou por filmes ou séries;
- Usar/ganhar pontos no aluguer;
- Usar cupão do mês atual e ver promoções;
- Ver dados de utilizador e histórico de aluguer e cupões;

Admin pode:

- Ligar-se na conta;
- Adicionar conteúdos;
- Editar conteúdos;
- Adicionar admins;
- Adicionar promoções/cupões;

Casos de utilização

Utilizador:

- O utilizador pesquisa uma série e aluga a mesma;
- O utilizador usa um cupão do mês atual;
- O utilizador vê o histórico de compras;

Admin:

- O admin edita um conteúdo;
- O admin adiciona um cupão;

Demonstração - O utilizador pesquisa uma série e aluga a mesma;

O Paulo pesquisa uma série e aluga a mesma

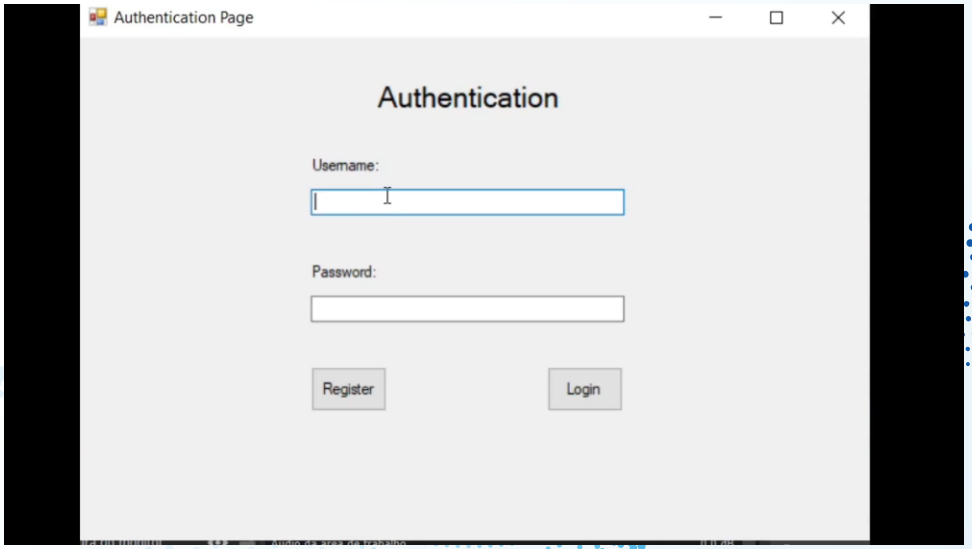
Accepted

DESCRIPTION

O Paulo sendo um cliente pretende pesquisar por uma série e de seguida alugar a mesma, gastando 100 pontos e pagando com Paypal. Pretendo que estes requerimentos sejam cumpridos e responsivos.

TASKS (4/4)

1. ~~Criar tabelas~~
2. ~~Criar SPs, UDFs e Triggers~~
3. ~~Adicionar às tabelas o que era alugado~~
4. ~~Criar página principal, forms e butões~~



Demonstração - O utilizador usa um cupão do mês atual;

O Paulo usa o cupão do mês

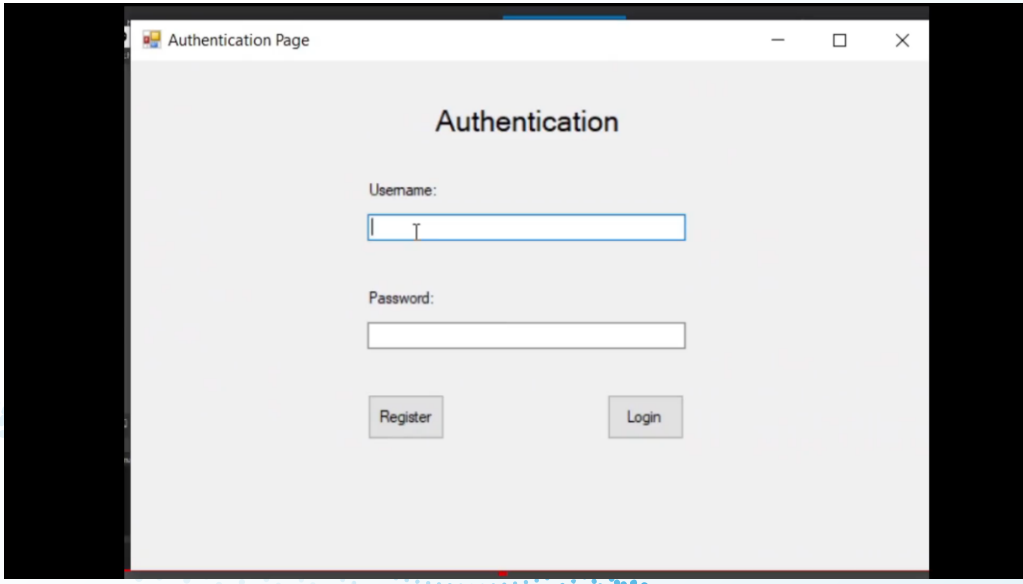
Accepted

DESCRIPTION

O Paulo sendo um cliente pretende ir às "Promotions e Cupons" e usar o cupão do mês em que se encontra.
Pretendo que estes requerimentos sejam cumpridos e responsivos.

TASKS (4/4)

1. ~~Criar tabelas~~
2. ~~Criar SPs, UDFs e Triggers~~
3. ~~Adicionar às tabelas os pontos~~
4. ~~Criar página "Promotions e Cupons"~~



Demonstração - O utilizador vê o histórico de compras;

O Paulo vai ver o seu histórico de compras

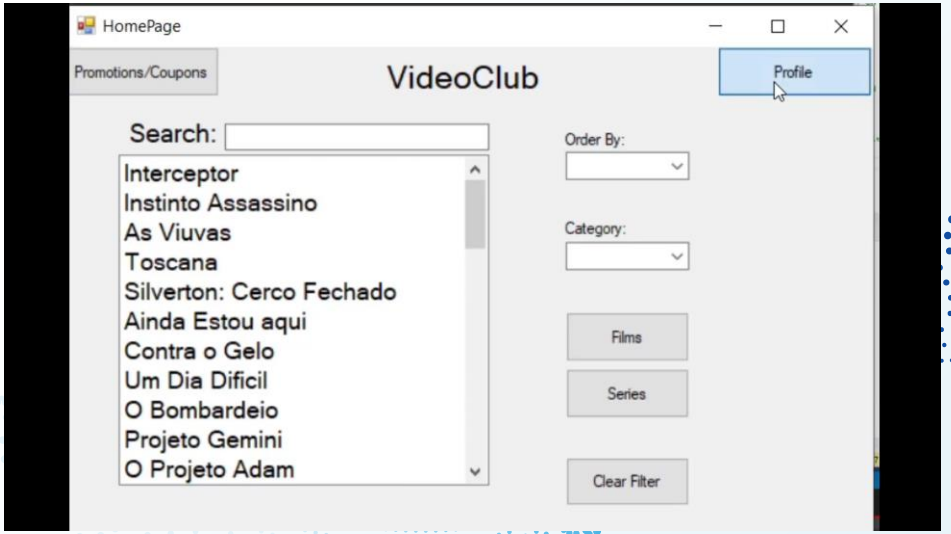
Accepted

DESCRIPTION

O Paulo sendo um cliente pretende ir ao seu perfil e poder ver o seu histórico de compras e devidas informações.
Pretendo que estes requerimentos sejam cumpridos e responsivos.

TASKS (4/4)

1. ~~Criar tabelas~~
2. ~~Criar SPs, UDFs e Triggers~~
3. ~~Criar página de perfil~~
4. ~~Colocar dados na página~~



Demonstração - O admin edita um conteúdo;

O Marco edita um conteúdo presente no sistema

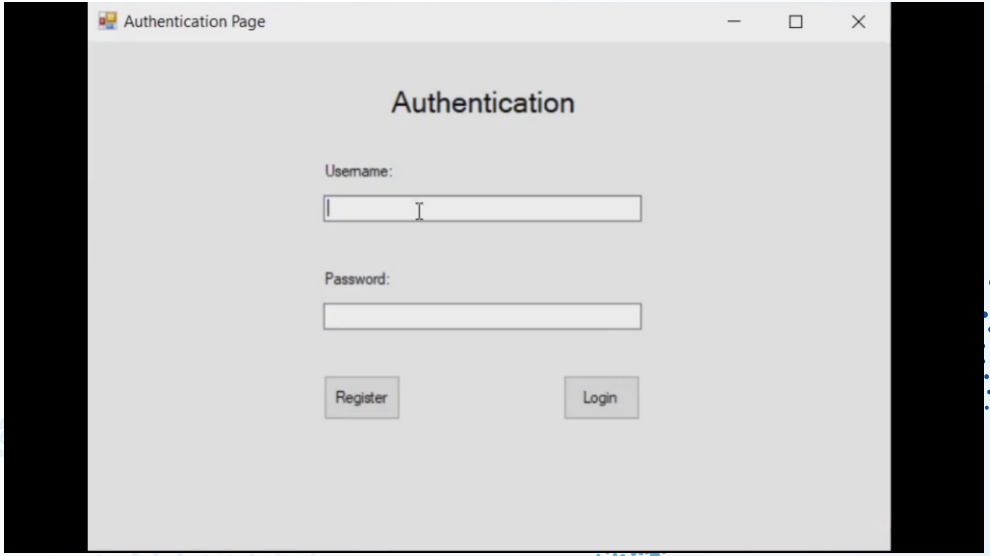
Accepted

DESCRIPTION

O Marco sendo um admin da aplicação pretende poder editar o conteúdo de um filme/série.
Pretendo que estes requerimentos sejam cumpridos e responsivos.

TASKS (4/4)

1. ~~Criar tabelas~~
2. ~~Criar SPs, UDFs e Triggers~~
3. ~~Criar página para as operações do admin~~
4. ~~Editar tabelas referentes~~



Demonstração - O admin adiciona um cupão;

O Marco adiciona um cupão ao sistem

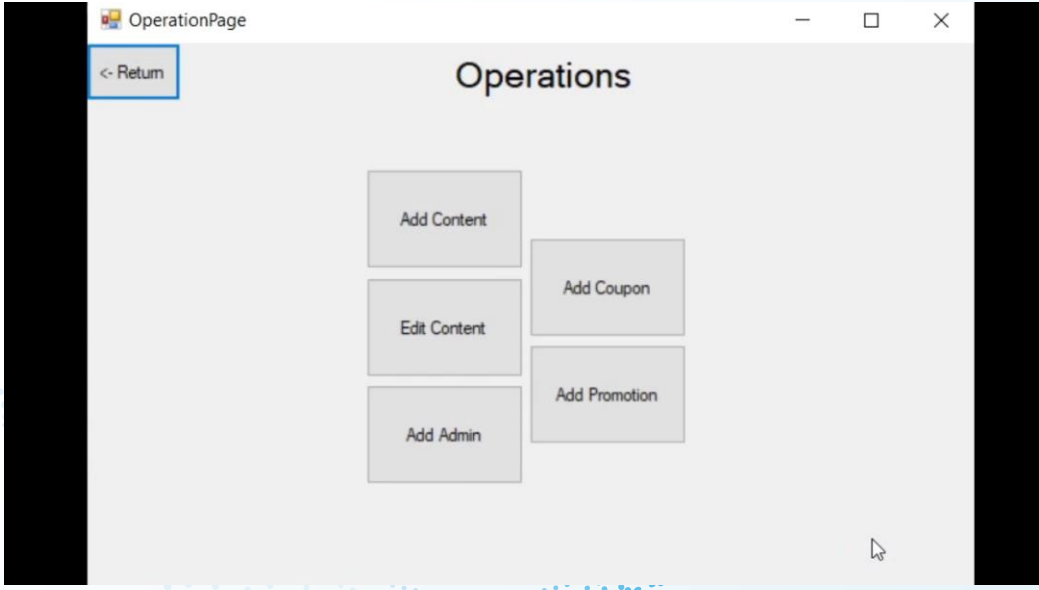
Accepted

DESCRIPTION

O Marco sendo um admin da aplicação pretende poder adicionar um cupão novo ao sistema.
Pretendo que estes requerimentos sejam cumpridos e responsivos.

TASKS (3/3)

1. ~~Criar tabelas~~
2. ~~Criar SPs, UDFs e Triggers~~
3. ~~Adicionar cupões~~



2. SQL Programming

Index, Stored Procedures, Triggers, UDFs, ...

Index

```
DROP INDEX idx_conteudo on Conteudo;
GO
CREATE INDEX idx_conteudo ON Conteudo (Preco, Categoria,TaxaIva, Titulo, Idade);
GO
```

Stored Procedures

```
DROP PROCEDURE Conteudo_filter;
GO
CREATE PROCEDURE Conteudo_filter
    @Categoria VARCHAR(50) = NULL,
    @Titulo VARCHAR(100) = NULL,
    @orderby VARCHAR(50) = NULL
AS
BEGIN
        SET NOCOUNT ON
        BEGIN
            SELECT ID, Ano, Preco, Categoria, TaxaIva, Titulo, Idade, Duracao
            FROM Conteudo WHERE Titulo LIKE ('%'+ISNULL(@Titulo, Titulo)+'%')
            AND Categoria = ISNULL(@Categoria,Categoria)
            ORDER BY CASE WHEN @orderby='Titulo' THEN Titulo END,
                     CASE WHEN @orderby='Categoria' THEN Categoria END,
                     CASE WHEN @orderby='Preco' THEN Preco END,
                     CASE WHEN @orderby='Ano' THEN Ano END
        END
END
GO
```

```
CREATE PROCEDURE ChangeConteudo
    @id INT,
    @ano INT,
    @preco FLOAT,
    @categoria VARCHAR(50),
    @taxaiva INT,
    @titulo VARCHAR(100),
    @idade INT,
    @duracao INT,
    @emailadmin VARCHAR(255)
AS
BEGIN
    BEGIN TRANSACTION
    SET NOCOUNT ON
    BEGIN TRY
        IF EXISTS(SELECT email FROM admin WHERE email=@emailadmin)
            BEGIN
                IF @ano <> 0
                BEGIN
                    UPDATE Conteudo SET Ano = @ano WHERE id = @id
                END

                IF @preco <> 0
                BEGIN
                    UPDATE Conteudo SET Preco = @preco WHERE id = @id
                END

                IF @categoria IS NOT NULL
                BEGIN
                    UPDATE Conteudo SET Categoria = @categoria WHERE id = @id
                END

                IF @taxaiva <> 0
                BEGIN
                    UPDATE Conteudo SET TaxaIva = @taxaiva WHERE id = @id
                END

                IF @titulo IS NOT NULL
                BEGIN
                    UPDATE Conteudo SET Titulo = @titulo WHERE id = @id
                END

                IF @idade <> 0
                BEGIN
                    UPDATE Conteudo SET Idade = @idade WHERE id = @id
                END

                IF @duracao <> 0
                BEGIN
                    UPDATE Conteudo SET Duracao = @duracao WHERE id = @id
                END
            END
            COMMIT TRANSACTION
    END TRY
    BEGIN CATCH
    ROLLBACK
    END CATCH
```

Stored Procedures

```
CREATE PROCEDURE login
    @email VARCHAR(255),
    @pass VARCHAR(20),
    @responseMessage VARCHAR(250)='' OUTPUT,
    @type BIT=0 OUTPUT,
    @outemail VARCHAR(255) OUTPUT
AS
BEGIN
    SET NOCOUNT ON
    SET @type=0
    SET @responseMessage='Invalid login'
    IF EXISTS (SELECT TOP 1 email FROM usuario WHERE email = @email)
    BEGIN
        SET @email=(SELECT email FROM usuario WHERE email=@email AND pass=HASHBYTES('SHA2_512', @pass))

        IF(@email IS NULL)
        BEGIN
            SET @type=0
            SET @responseMessage='Incorrect password'
        END
        ELSE
        BEGIN
           SET @responseMessage='User successfully logged in'
           SET @type = 1
           SET @outemail = @email
        END
    END
    ELSE
        IF EXISTS (SELECT TOP 1 email FROM admin WHERE email = @email)
        BEGIN
            SET @email=(SELECT email FROM admin WHERE email=@email AND pass=HASHBYTES('SHA2_512', @pass))

            IF(@email IS NULL)
            BEGIN
                SET @type=0
                SET @responseMessage='Incorrect password'
            END
            ELSE
            BEGIN
               SET @responseMessage='Admin successfully logged in'
               SET @type = 1
               SET @outemail = @email
            END
        END

END
```

Triggers

```
DROP TRIGGER usaCupaoTrigger;
GO

CREATE TRIGGER usaCupaoTrigger ON [usuario_usa_cupao]
AFTER INSERT
AS
BEGIN
    SET NOCOUNT ON;
    DECLARE @cupao as CHAR(10);
    DECLARE @email as VARCHAR(255);
    DECLARE @pontos as INT;
    SELECT @cupao = cupao_id, @email = usuario_email FROM inserted;
    SELECT @pontos = cupao.pontos FROM cupao WHERE id = @cupao;
    BEGIN TRY
        UPDATE usuario SET usuario.pontos += @pontos WHERE email = @email
    END TRY
    BEGIN CATCH
        RAISERROR ('Nao foi possivel atribuir os pontos', 16, 1);
        ROLLBACK TRAN
    END CATCH

END
GO
```

UDFs

```
DROP FUNCTION historico_aluguer;
GO

CREATE FUNCTION historico_aluguer (@email VARCHAR(255)) RETURNS TABLE
AS
    RETURN (SELECT Titulo, Preco, pagamento, pontos_acumulados FROM
    (aluguer JOIN Pertence ON aluguer.id= Pertence.ID_Aluguer)
    JOIN Conteudo ON Pertence.ID_Conteudo=Conteudo.ID
    WHERE usuario_email= @email
    ORDER BY aluguer.datain DESC OFFSET 0 ROWS)
GO
```

```
DROP FUNCTION getUsuarioCoupao;
GO

CREATE FUNCTION getUsuarioCoupao (@email VARCHAR(255)) RETURNS TABLE
AS
    RETURN (SELECT id, datain, dataen, pontos
    FROM (usuario_usa_cupao JOIN cupao ON cupao_id=id)
    WHERE usuario_email=@email ORDER BY datain DESC OFFSET 0 ROWS)

GO
```

Obrigado

Alguma questão? 🔍