




Market Chat

Base de Dados
P2G4
Luís Diogo, 108668
Nuno Carvalho, 97783



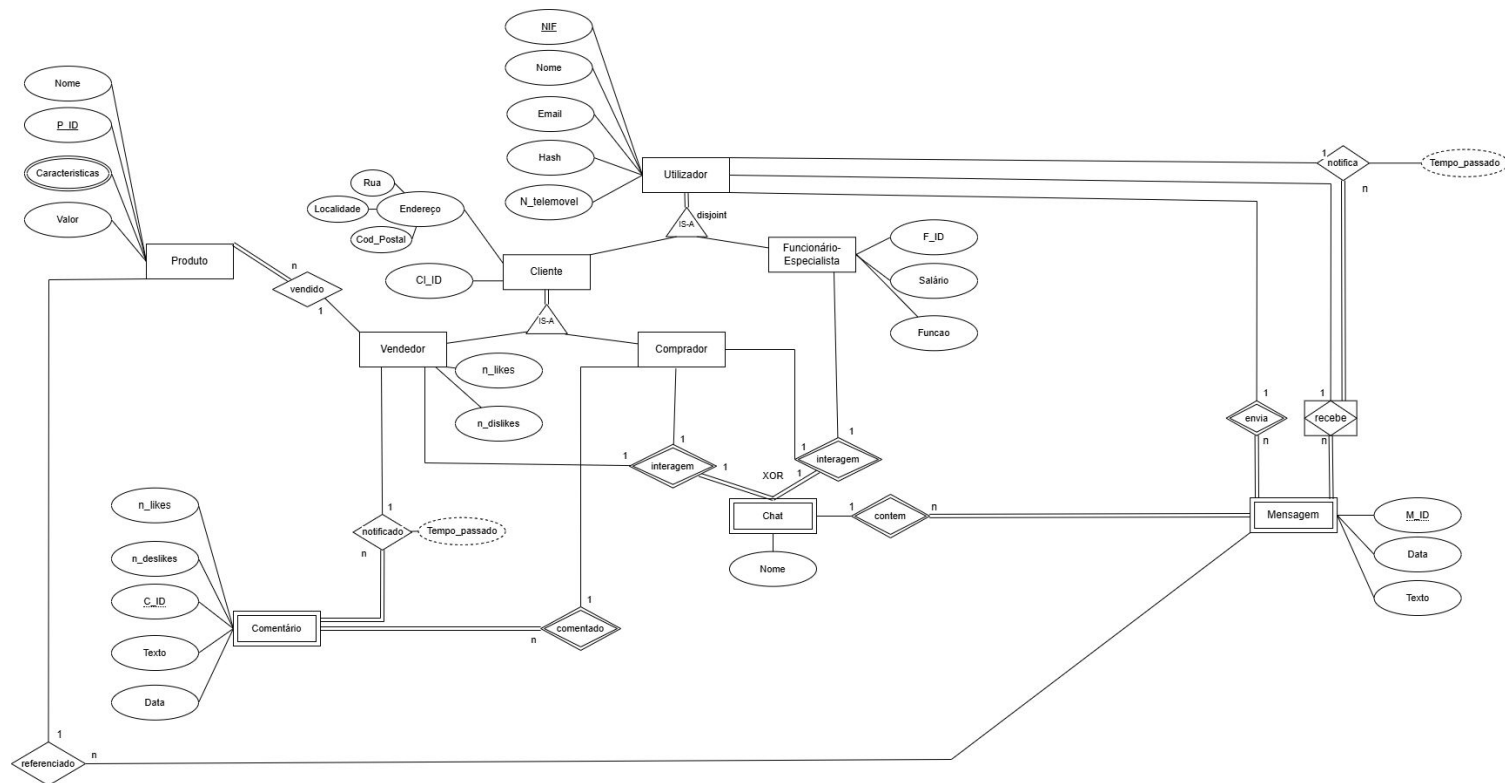
Introdução

Market Chat é uma aplicação que garante serviços de chat e comentários a websites de compras e vendas

Requisitos:

- Registo de Utilizadores identificados pelos NIFs, diferenciam-se com os papéis de funcionário e cliente sendo que o cliente pode ser comprador e/ou vendedor.
- Funcionários podem:
 - Registrar produtos para vender associados ao seu vendedor.
 - Responder a mensagens de clientes se forem especialistas no tema.
- Compradores podem:
 - Fazer pesquisa filtrada dos produtos disponíveis .
 - Iniciar chats com vendedores ou o especialista.
 - Comentar vendedores e avaliar vendedores e comentários.
- Vendedores podem:
 - Responder a mensagens de clientes.
 - Iniciar chat com o especialista.
- Uma mensagem ao ser enviada notificará o recetor.

Diagrama Entidade-Relação



SQL scripts

- Views
 - Pesquisas que não requerem parâmetros nem lógica procedural
- User Defined Functions
 - Pesquisas que requerem parâmetros e/ou lógica procedural
- Stored Procedures
 - Controlo de integridade
- Triggers
 - Controlo de integridade

UDF

```
CREATE FUNCTION dbo.ObterMensagensDoChat
(
    @UtilizadorNIF INT,
    @Chat_NIF INT
)
RETURNS @Mensagens TABLE
(
    MensagemID INT,
    Chat_A_NIF INT,
    Chat_B_NIF INT,
    EnviadoPor INT,
    HoraEnviada DATETIME,
    Conteudo NVARCHAR(MAX)
)
AS
BEGIN
    --para ver se o utilizador esta no chat
    IF EXISTS (
        SELECT 1
        FROM Chat
        WHERE (A_NIF = @UtilizadorNIF AND B_NIF = @Chat_NIF)
            OR (A_NIF = @Chat_NIF AND B_NIF = @UtilizadorNIF)
    )
    BEGIN
        -- preencher a tabela de return
        INSERT INTO @Mensagens
        SELECT MensagemID, Chat_A_NIF, Chat_B_NIF, EnviadoPor, HoraEnviada, Conteudo
        FROM Mensagem
        WHERE (Chat_A_NIF = @UtilizadorNIF AND Chat_B_NIF = @Chat_NIF)
            OR (Chat_A_NIF = @Chat_NIF AND Chat_B_NIF = @UtilizadorNIF)
        ORDER BY HoraEnviada;
    END

    RETURN;
END;
GO
```

Stored Procedures

```
CREATE PROCEDURE sp_InsertUtilizadorAsCliente
    @NIF INT,
    @Nome VARCHAR(255),
    @Email VARCHAR(255),
    @Hash VARCHAR(255),
    @N_telemovei INT,
    @Rua VARCHAR(255),
    @Localidade VARCHAR(255),
    @Cod_Postal VARCHAR(20),
    @IsComprador BIT,
    @IsVendedor BIT
AS
BEGIN
    IF @IsComprador = 0 AND @IsVendedor = 0 -- tem que ser alguma coisa IS-A obrigatorio
    BEGIN
        RAISERROR ('Um Cliente deve ser Comprador ou Vendedor ou ambos', 16, 1);
        RETURN;
    END

    BEGIN TRANSACTION;

    BEGIN TRY
        INSERT INTO Utilizador (NIF, Nome, Email, Hash, N_telemovei)
        VALUES (@NIF, @Nome, @Email, @Hash, @N_telemovei);
        INSERT INTO Cliente (NIF)
        VALUES (@NIF/*, @Rua, @Localidade, @Cod_Postal*/);
        IF @IsComprador = 1
        BEGIN
            INSERT INTO Comprador (NIF)
            VALUES (@NIF);
        END
        IF @IsVendedor = 1
        BEGIN
            INSERT INTO Vendedor (NIF)
            VALUES (@NIF);
        END

        COMMIT TRANSACTION;
    END TRY
    BEGIN CATCH
        ROLLBACK TRANSACTION;
        THROW;
    END CATCH;
END;
GO
```

Triggers

```
CREATE TRIGGER trg_VerificarChat
ON Chat
INSTEAD OF INSERT, UPDATE
AS
BEGIN
    DECLARE @ErrorMessage NVARCHAR(4000);
    DECLARE @A_NIF INT;
    DECLARE @B_NIF INT;

    DECLARE Cursor CURSOR FOR
    SELECT A_NIF, B_NIF FROM inserted;
    OPEN Cursor;
    FETCH NEXT FROM Cursor INTO @A_NIF, @B_NIF;

    WHILE @@FETCH_STATUS = 0
    BEGIN
        IF @A_NIF = @B_NIF
        BEGIN
            SET @ErrorMessage = 'A_NIF nao pode ser igual a B_NIF.';
            RAISERROR (@ErrorMessage, 16, 1);
            ROLLBACK TRANSACTION;
            RETURN;
        END;

        -- para garantir que é criado por c->v ou cl->f
        IF NOT (
            (EXISTS (SELECT 1 FROM Comprador WHERE Comprador.NIF = @A_NIF) AND
              EXISTS (SELECT 1 FROM Vendedor WHERE Vendedor.NIF = @B_NIF)) OR
            (EXISTS (SELECT 1 FROM Cliente WHERE Cliente.NIF = @A_NIF) AND
              EXISTS (SELECT 1 FROM Funcionario WHERE Funcionario.NIF = @B_NIF))
        )
        BEGIN
            SET @ErrorMessage = 'Um chat deve ser entre um Comprador e um Vendedor.';
            RAISERROR (@ErrorMessage, 16, 1);
            ROLLBACK TRANSACTION;
            RETURN;
        END;

        FETCH NEXT FROM Cursor INTO @A_NIF, @B_NIF;
    END;

    CLOSE Cursor;
    DEALLOCATE Cursor;
    INSERT INTO Chat (A_NIF, B_NIF)
    SELECT A_NIF, B_NIF
    FROM inserted;
END;
GO
```