IPV- Instituto Politécnico de Viseu

ESTGV - Escola Superior de Tecnologia e Gestão de Viseu

Departamento de Informática

Licenciatura de Engenharia Informática

Base de Dados I

Relatório do Trabalho Prático

Trabalho realizado por: André Filipe Almeida Pinto – 19018

João Gomes Francisco Carmo – 18703

Joaquim Pedro da Costa Rodrigues – 19034

Miguel Ângelo Gonçalves Campos – 19029

Ricardo Miguel Dos Santos Monteiro – 19022

IPV- Instituto Politécnico de Viseu
ESTGV - Escola Superior de Tecnologia e Gestão de Viseu
Departamento de Informática

Relatório do Trabalho Prático

Realizado por: André Filipe Almeida Pinto – 19018
João Gomes Francisco Carmo – 18703
Joaquim Pedro da Costa Rodrigues – 19034
Miguel Ângelo Gonçalves Campos – 19029
Ricardo Miguel Dos Santos Monteiro – 19022

Índice

Índice de Figuras

1 – Introdução

O presente relatório descreve requisitos propostos para a realização deste trabalho prático, a maneira como foram abordados e resolvidos, implementando as técnicas e conhecimentos de SQL aprendidos nas aulas de Base de Dados I, e a explicação da criação e de uma aplicação web para gerir e visualizar os requisitos criados

Com este trabalho esperamos conseguir implementar com sucesso o conhecimento adquirido nas aulas teóricas e práticas, de maneira a criar um programa que se assemelhe a software utilizado em situações reais.

2 – Código

Neste segmento vamos explicar os requisitos que foram pedidos, como os concretizamos e explicando o código que achámos que se melhor adequava para cada situação.

2.1 – Consultas

2.1.1 – Reservar uma certa sala num certo dia

```
SELECT
    Reservations.ID,
    CAST(StartDateTime AS DATE) AS [Date],
    CAST(StartDateTime AS TIME(0)) AS [Start Time],
    CAST(EndDateTime AS TIME(0)) AS [End Time],
    Room,
    CAST(EndDateTime - StartDateTime AS TIME(0)) AS [Reservation Time],
    CAST(dateadd(MINUTE, CleaningTime, '00:00:00') AS TIME(0)) AS [Cleaning Time],
    dateadd(MINUTE, CleaningTime, CAST(EndDateTime - StartDateTime AS TIME(0))) AS [Total Time],
    dateadd(MINUTE, CleaningTime, CAST(EndDateTime AS TIME(0))) AS [Available After]
FROM
Reservations
INNER JOIN Rooms ON Rooms.ID = Reservations.Room
WHERE
    CAST(StartDateTime AS DATE) = '2022-04-29' AND
    Room = 3;
```

Figura 1 - Código para reservar salas num certo dia

Primeiro usamos a função CAST para converter as variáveis para o desejado tipo e com o nome que pretendemos, depois usamos CAST mais uma vez para definir o tempo da reserva e para calcular o tempo que vai ser necessário para limpar a sala. Depois de termos o tempo da reserva mais o tempo de limpeza já temos o tempo total em que a sala vai estar indisponível e daí podemos extrair quando é que a sala vai estar disponível, de maneira a podermos bloquear reservas para essa sala.

2.1.2 – Obter todas as salas livres numa certa hora num certo GeoCenter

```sql
SELECT * FROM
Rooms
WHERE ID NOT IN (
    SELECT Reservations.Room FROM
    Reservations
    INNER JOIN Rooms ON Rooms.ID = Reservations.Room
    WHERE
        '2022-04-29 19:00:00' <= dateadd(MINUTE, CleaningTime, EndDateTime) AND
        '2022-04-29 20:00:00' >= StartDateTime
) AND
GeoCenter = 3;
```

Figura 2 - Código para obter as salas livres num certo horário de um GeoCenter

Nesta consulta vamos ver todas as salas que não estão na tabela de reservados, numa hora específica, e ainda determinamos que esta consulta só se aplica ao GeoCenter nº 3.

2.1.3 – Ver o status de uma sala consoante o horário atual

```sql
SELECT
    ID,
    Name,
    CASE
        WHEN ID IN (
            SELECT Room FROM
            Reservations
            WHERE GETDATE() BETWEEN StartDateTime AND EndDateTime
        ) THEN 'Reserva em curso'
        WHEN ID IN (
            SELECT Room FROM
            Reservations
            INNER JOIN Rooms ON Rooms.ID = Reservations.Room
            WHERE GETDATE() BETWEEN EndDateTime AND dateadd(MINUTE, CleaningTime, EndDateTime)
        ) THEN 'Limpeza em curso'
        WHEN ID NOT IN (
            SELECT Room FROM
            Reservations
            WHERE GETDATE() BETWEEN StartDateTime AND EndDateTime
        ) THEN 'Sala Livre'
        ELSE 'Error'
    END AS [Status]
FROM
Rooms
WHERE ID = 3;
```

Figura 3 - Código para obter o status de uma sala consoante o horário atual

Primeiro vamos pedir o Nome e o ID de uma sala, e vamos ter um CASE para determinar o estado da sala, se estiver reservada, a ser limpa, se está livre ou se nenhum dos anteriores, o que vai dar erro, e passamos o estado como STATUS para o utilizador saber o estado da sala.

2.1.4 – Obter o número total de reservas feitas entre duas datas

```
SELECT COUNT(Reservations.ID) AS 'Number of Reservations' FROM
Reservations
INNER JOIN Rooms ON Rooms.ID = Reservations.Room
    WHERE
        '2022-04-29' <= CAST(dateadd(MINUTE, CleaningTime, EndDateTime) AS DATE) AND
        '2022-09-06' >= CAST(StartDateTime AS DATE)
```

Figura 4 - Obter o número de reservas entre duas datas

Nesta consulta vamos usar a função COUNT para obter o número de reservas na tabela de reservas, mas vamos limitar a procura entre as duas datas que pretendemos.

2.1.5 – Número de utilizadores registados

```
SELECT COUNT(ID) AS [Number of Users] FROM
Users;
```

Figura 5- Número de utilizadores registados

Nesta consulta bastante simples simplesmente vamos contar o número de Ids que temos na tabela USERS.

2.2 – DDL (Data Definiton Language)

2.2.1 – Create tables

```
CREATE TABLE LogTypes(
    ID  int IDENTITY(1,1)   CONSTRAINT PK_LogTypes PRIMARY KEY,
    Name varchar(50) NOT NULL
);

CREATE TABLE Logs(
    ID  int IDENTITY(1,1)   CONSTRAINT PK_Logs PRIMARY KEY,
    LogDescription varchar(2000),
    LogDate datetime NOT NULL DEFAULT GETDATE(),
    LogType int NOT NULL,
    CONSTRAINT FK_Logs_LogTypes FOREIGN KEY (LogType) REFERENCES LogTypes(ID)
);

CREATE TABLE GeoCenters (
    ID  int  IDENTITY(1,1)  CONSTRAINT PK_GeoCenters PRIMARY KEY,
    Name varchar(50)  NOT NULL,
    Location varchar(50)  NOT NULL,
    IsActive bit  NOT NULL  DEFAULT 1,
    CreatedAt datetime  NOT NULL    DEFAULT GETDATE()
);

CREATE TABLE Roles (
    ID int  IDENTITY(1,1)   CONSTRAINT PK_Roles PRIMARY KEY,
    Code int NOT NULL,
    Name varchar(50),
    ReserveRooms bit  NOT NULL   DEFAULT 1,
    CreateRooms bit  NOT NULL  DEFAULT 0,
    EditRooms bit  NOT NULL  DEFAULT 0,
    CreateGeoCenters bit  NOT NULL  DEFAULT 0,
    EditGeoCenters bit  NOT NULL  DEFAULT 0,
    CreateUsers bit  NOT NULL  DEFAULT 0,
    EditUsers bit  NOT NULL  DEFAULT 0,
    IsActive bit  NOT NULL  DEFAULT 1,
    CreatedAt datetime  NOT NULL    DEFAULT GETDATE()
);
```

Figura 6 - Código para criar as tabelas

Nesta imagem podemos ver alguns dos CREATE tables usados para fazer as tabelas na base de dados. Essencialmente definimos um ID para cada tabela, usamos CONSTRAINT “nome” PRIMARY KEY, para definir o campo da tabela que vai ser a chave primária, e de seguida começamos a adicionar os outros campos, primeiro definimos o nome, depois o tipo de variável, e de seguida definimos regras, como por exemplo NOT NULL que não permite que o campo fique vazio. Também utilizamos outras regras como por exemplo DEFAULT na eventualidade de o campo ser deixado vazio, assemelha este valor por defeito. Por fim em algumas tabelas definimos as chaves estrangeiras que vão ser uteis para quando precisarmos de invocar valores de outras tabelas.

2.2.2 – Drop tables

```
IF OBJECT_ID('Reservation_Groups','U') IS NOT NULL
DROP TABLE Reservation_Groups;

IF OBJECT_ID('Reservations','U') IS NOT NULL
DROP TABLE Reservations;

IF OBJECT_ID('Users_GeoCenters','U') IS NOT NULL
DROP TABLE Users_GeoCenters;

IF OBJECT_ID('Rooms','U') IS NOT NULL
DROP TABLE Rooms;

IF OBJECT_ID('Users','U') IS NOT NULL
DROP TABLE Users;

IF OBJECT_ID('Roles','U') IS NOT NULL
DROP TABLE Roles;

IF OBJECT_ID('GeoCenters','U') IS NOT NULL
DROP TABLE GeoCenters;

IF OBJECT_ID('Logs','U') IS NOT NULL
DROP TABLE Logs;

IF OBJECT_ID('LogTypes','U') IS NOT NULL
DROP TABLE LogTypes;
```

Figura 7- Código para fazer drop de tabelas

Com este código vamos poder eliminar uma certa tabela da base de dados.

2.3 – DML (Data Managing Language)

2.3.1 – Inserir dados

```
INSERT INTO GeoCenters (Name, Location)
VALUES
('Viseu', 'Viseu'),
('Lisboa Norte', 'Lisboa'),
('Lisboa Sul', 'Lisboa'),
('Porto Norte', 'Porto'),
('Porto Sul', 'Porto'),
('Coimbra', 'Coimbra');

SELECT * FROM GeoCenters;

/*-------- Tabela Roles --------*/
INSERT INTO Roles (
    Code,
    Name,
    ReserveRooms,
    CreateRooms,
    EditRooms,
    CreateGeoCenters,
    EditGeoCenters,
    CreateUsers,
    EditUsers
) VALUES
(0, 'Administrador', 1, 1, 1, 1, 1, 1, 1),
(1, 'Utilizador B◆sico', 1, 0, 0, 0, 0, 0, 0),
(2, 'Gestor de Utilizadores', 1, 0, 0, 0, 0, 1, 1),
(3, 'Gestor de Salas e Centros', 1, 1, 1, 1, 1, 0, 0);

SELECT * FROM Roles;
```

Figura 8 - Código para inserir dados nas tabelas

Neste código podemos ver a abordagem tomada para inserir dados nas tabelas da base de dados, utilizamos INSERT INTO “nome da tabela” (), dentro dos parenteses é onde vamos definir quais campos vamos inserir os dados, depois com a função VALUES () vamos colocar pela ordem dos campos na tabelam, os valores correspondentes.

2.4 – Objetos Lógicos

Os objetos lógicos são funções que criámos para fazer certos pedidos específicos, nomeadamente aqueles referidos no enunciado.

```
IF OBJECT_ID('proDeactivatesAllRoomsFromGeoCenter') IS NOT NULL
    DROP PROCEDURE proDeactivatesAllRoomsFromGeoCenter
GO

CREATE PROC proDeactivatesAllRoomsFromGeoCenter
    @GeoCenterID INT = NULL
AS
BEGIN
    IF  @GeoCenterID = NULL
        BEGIN
            PRINT '? necess?rio mencionar um @GeoCenterID'
        END
    ELSE
        BEGIN
            DECLARE @RoomID INT

            DECLARE curGeoCenterDeactivation CURSOR
            FOR
                SELECT Rooms.ID
                FROM Rooms
                WHERE Rooms.GeoCenter = @GeoCenterID

            OPEN curGeoCenterDeactivation
                FETCH NEXT FROM curGeoCenterDeactivation
                INTO @RoomID

                WHILE @@FETCH_STATUS = 0
                BEGIN
                    UPDATE Rooms
                    SET IsActive = 0
                    WHERE Rooms.ID = @RoomID

                    FETCH NEXT FROM curGeoCenterDeactivation
                    INTO @RoomID
                END
            CLOSE curGeoCenterDeactivation
            DEALLOCATE curGeoCenterDeactivation
        END
END
GO
```

Figura 9 - Código de um processo armazenado de um cursor par desativar salas

Primeiro definimos o nome que vamos dar ao procedimento, de seguida criamos uma variável RoomId que vai guardas os IDs de todas as salas que vamos desativar, de seguida vamos informar o utilizador que precisa de referir o nome do GeoCenter do qual pretende desativar as salas, depois vamos iniciar um cursor que vai buscar as salas todas do GeoCenter para consequentemente poder começar o processo de passar por todas as salas, mudar o seu estado de IsActive para 0 e fazer este *loop* até passar as salas todas. Por fim vamos fechar o processo e para concluir a execução de comandos.

3 – Aplicação Web

Neste segmento iremos explicar como abordamos o processo de criação de uma aplicação web para utilização, visualização e gestão do programa em SQL.

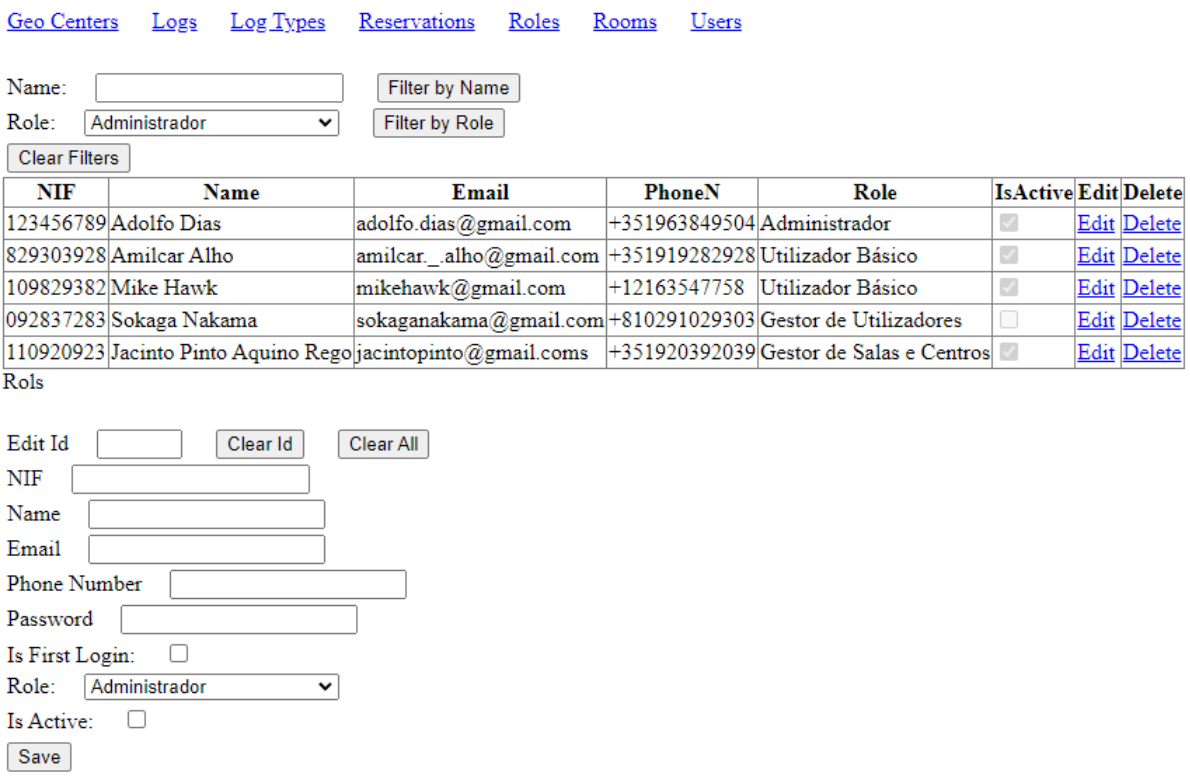

Figura 10 - Página para gerir a tabela Users

Nesta página é onde vamos gerir a tabela USERS, o site tem uma página por tabela e funcionam da mesma maneira essencialmente, claro que quanto mais complexa a tabela mais elementos o site vai ter para gerir. Como podemos ver na imagem podemos filtrar os utilizadores por nome ou por cargo, de seguida temos representada visualmente a tabela e no fim as opções de editar ou apagar campos, sendo que alguns campos se forem eliminados simplesmente passam para um IsActive false em vez de apagar por completo, nomeadamente para evitar conflitos de dados e não eliminar elementos que afetem outras tabelas. Por fim temos os campos onde vamos poder editar um campo procurando o mesmo pelo seu ID, ou até criar um campo novo deixando a caixa do ID vazia.

4 – Conclusão

Concluindo a realização deste trabalho podemos com confiança dizer que aplicamos com sucesso muito do que aprendemos durante as aulas, e que somos capazes de desenvolver uma um site de gestão de base de dados básico.

Fomos capazes de analisar os requisitos pedidos pelo cliente, elaborar uma estratégia para abordar os mesmos de maneira a alcançar os objetivos pretendidos da forma mais otimizada possível e trabalhar em equipa para dividir as tarefas de maneira a concluir o trabalho da maneira mais rápida.

Gostaríamos também de agradecer a todos os professores e colegas que se disponibilizaram para ajudar a resolver dúvidas que surgiram ao longo do desenvolvimento deste trabalho.