
P8-G10 / PROFESSOR JOAQUIM SOUSA PINTO



BASE DE DADOS FESTIVAL DE VERÃO

Gonçalo Ferreira (107853) (50.00%) / Sara Almeida (108796) (50.00%)

ÍNDICE

- INDÍCE
- INTRODUÇÃO
- ENTIDADES
- DIAGRAMA ENTIDADE-RELAÇÃO (DER)
- DIAGRAMA RELACIONAL
- ANÁLISE DE REQUISITOS
- DDL
- DML
- SP, VIEW, UDF, TRIGGERS, INDEXES, CURSOR
- INTERFACE GRÁFICA
- CONCLUSÃO



INTRODUÇÃO



Este projeto foi realizado no âmbito da disciplina de Base de Dados no ano de 2022/2023 e tem como objetivo recriar um sistema de gestão de dados de festivais de verão, que contém toda a informação recorrente a cada um dos diferentes festivais.

Apresentaremos ao longo do relatório a maneira como implementámos a base de dados recorrendo ao SQL Server Management Studio e guardando as Queries executadas no servidor do IEETA.

A implementação da interface foi realizada na linguagem C# recorrendo ao Windows Forms que está presente na ferramenta Visual Studio 2022 da Microsoft.

O planeamento do projeto foi baseado em diagramas típicos para o desenvolvimento de bases de dados, como o diagrama relacional e o diagrama entidade-relação (DER)



ENTIDADES

ARTISTA Esta entidade representa todos os artistas que vão atuar/já atuaram em concertos dos nossos festivais de verão	FESTIVAL Esta entidade representa todos os festivais de verão que vão ocorrer/já ocorreram	CONCERTO Esta entidade representa todos os concertos que estão ligados/estiveram ligados ou que fazem/fizeram parte dos festivais de verão da nossa base de dados
LOCALIZAÇÃO Esta entidade demarca geograficamente a localização de cada festival e de cada acampamento adjacente	PASSE Esta entidade é referente a cada um dos passes/bilhetes que dão/deram acesso a cada um dos festivais e/ou acampamentos da nossa base de dados	ACAMPAMENTO Esta entidade serve para identificar cada um dos acampamentos aderentes ou com parcerias com os festivais de verão
PATROCINADOR Esta entidade é para todas as empresas que participem diretamente ou indiretamente no patrocínio de barracas e palcos pelos diferentes festivais de verão	PALCO Esta entidade é para todos englobar todos os palcos que estão a alojar/já alojaram concertos para os diferentes festivais de verão	BARRACA Esta entidade identifica as barracas que estão/estiveram a prestar serviços a cada barraca, seja vendendo bebida ou comida





ARTISTA ID Artista Nome Artístico Nome Verdadeiro Idade Prémios Nacionalidade Estilo Musical	FESTIVAL ID Festival Nome Data de Início Dias de Duracão Localização Lotação Máxima ID Localizaçãp	CONCERTO Número do Concerto Data do Concerto Duração em Dias ID Palco ID Festival
LOCALIZAÇÃO ID Localização Coordenadas País Local Cidade	PASSE Número de Série Dias de Duração Preço ID Festival	ACAMPAMENTO ID Acampamento Espaço Disponível Nome do Acampamento Duração em Dias Localização Acomodidades ID Localização
PATROCINADOR ID Patrocinador Nome	PALCO ID Palco Nome ID Patrocinador	BARRACA ID Barraca Tipo de Alimentação ID Patrocinador



DER

BASE DE DADOS 22/23

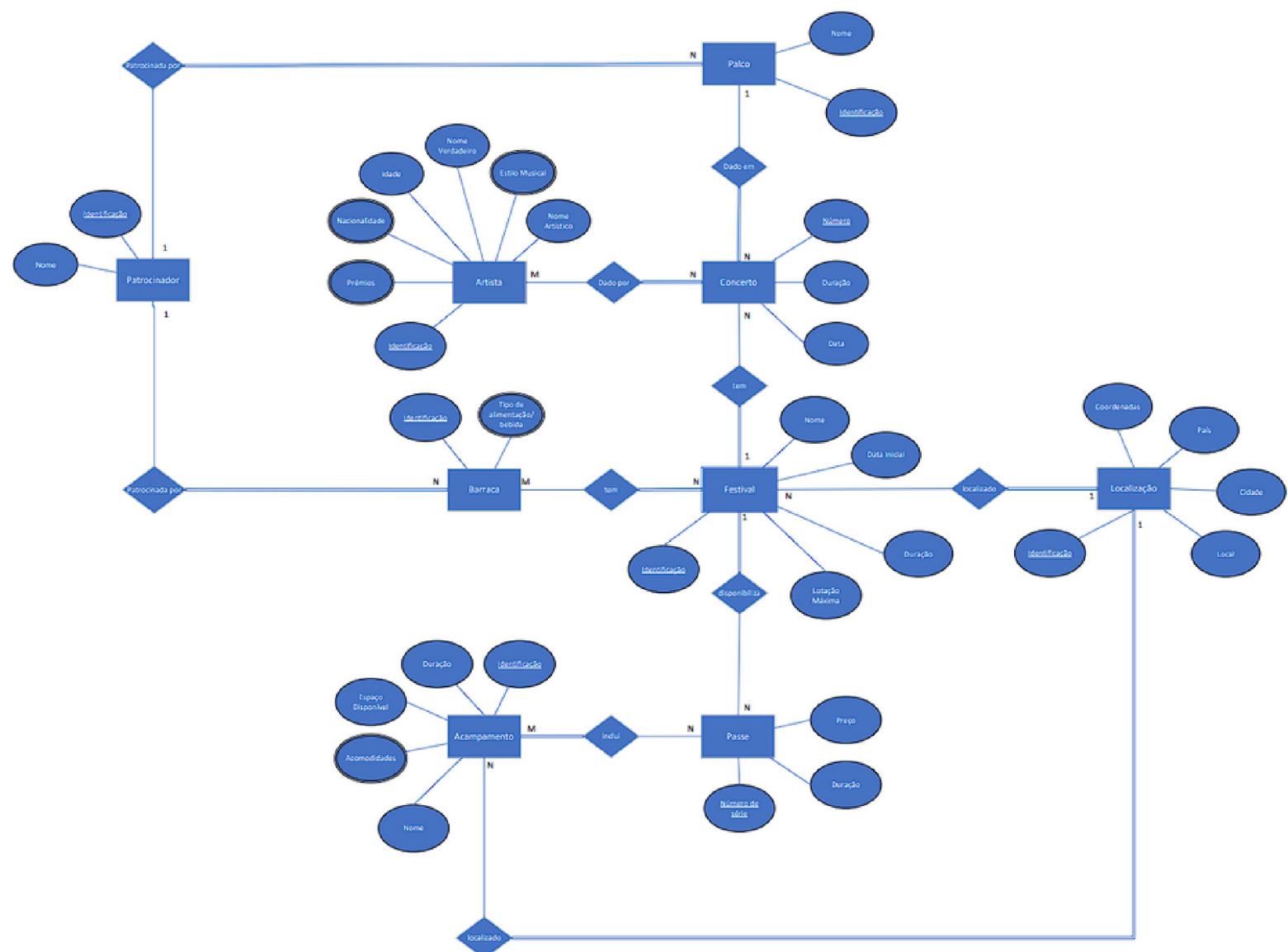


Figura 1. Diagrama Entidade/Relacionamento

RELACIONAL

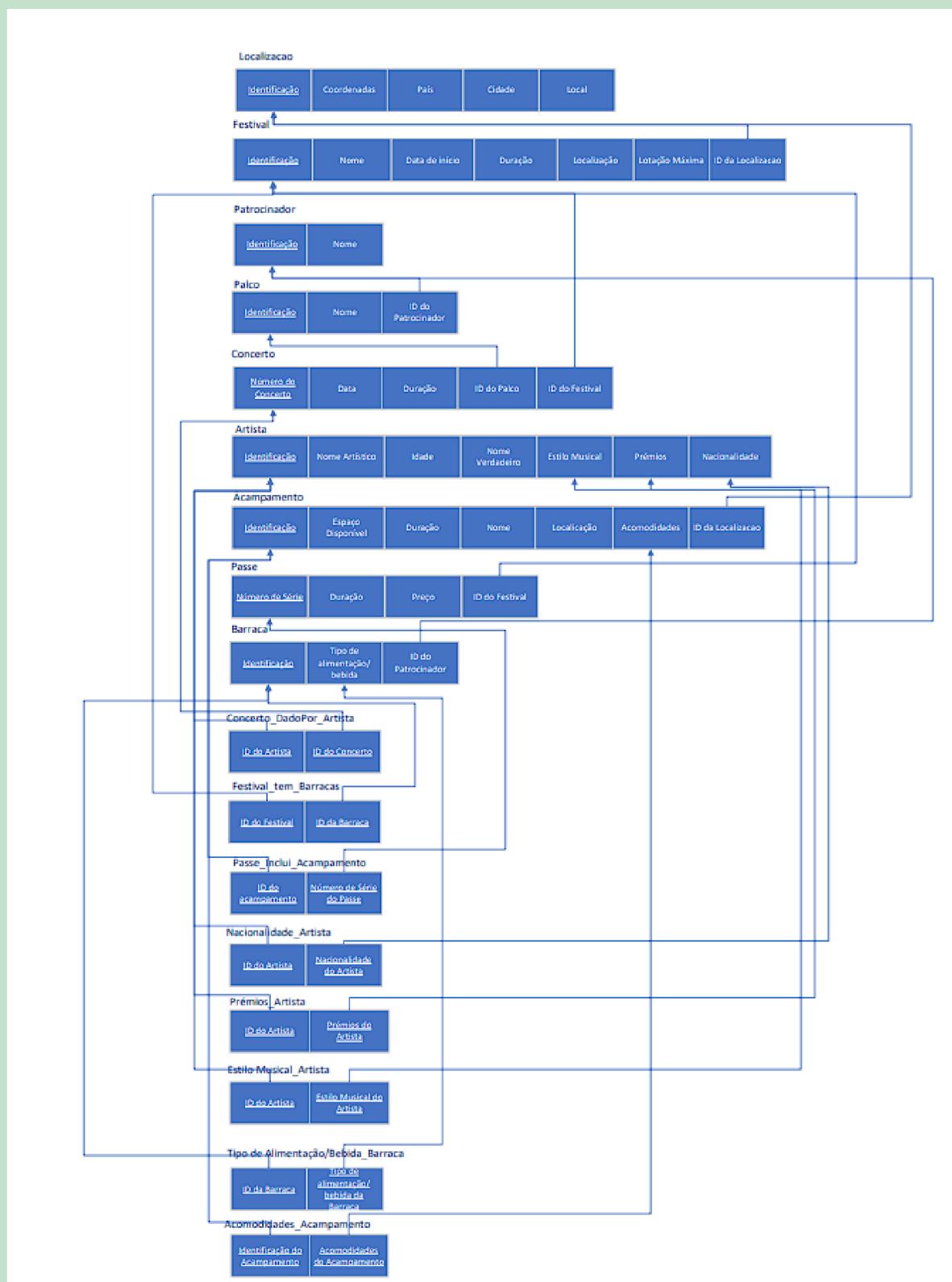


Figura 2. Modelo Relacional





ANÁLISE DE REQUISITOS

Considerando então um sistema de gestão de festivais de verão e tudo o que os envolve, o nosso objetivo é que o cliente interessado possa visualizar todas as informações acerca de um festival em específico, como as características específicas do seu cartaz e acomodidades disponibilizadas. Assim, definimos os seguintes requisitos funcionais:

O cliente deve conseguir:

- ver informações acerca do festival, como a sua localização, data, etc.;
- ver os concertos disponibilizados em cada dia;
- identificar o palco em que se vai realizar cada concerto;
- ver informações sobre os artistas que atuam nos concertos;
- ver informações acerca dos passes como a quantidade ainda disponível, o seu preço, etc.;
- ver a disponibilidade e características do acampamento existente no recinto;
- ver a oferta de alimentação e bebida disponibilizada em cada barraca



DDL

Para inserir as tabelas na base de dados, tivemos que usar código DDL que significa Data Definition Language, e serve para estabelecer então a definição das tabelas que escolhemos para o nosso projeto

```
CREATE TABLE SummerFest.Localizacao(
    ID_Localizacao INT PRIMARY KEY,
    Coordenadas VARCHAR(64),
    Pais VARCHAR(32),
    Cidade VARCHAR(64),
    Local VARCHAR(64),
)

CREATE TABLE SummerFest.Festival(
    ID_Festival INT PRIMARY KEY,
    Nome VARCHAR(64),
    Data_De_Inicio DATE,
    Duracao_Dias INT,
    Localizacao VARCHAR(64),
    Lotacao_Maxima INT,
    ID_Localizacao INT REFERENCES SummerFest.Localizacao(ID_Localizacao)
)

CREATE TABLE SummerFest.Patrocinador(
    ID_Patrocinador INT PRIMARY KEY,
    Nome VARCHAR(128),
)

CREATE TABLE SummerFest.Palco(
    ID_Palco INT PRIMARY KEY,
    Nome VARCHAR(64),
    Patrocinador INT REFERENCES SummerFest.Patrocinador(ID_Patrocinador)
)
```

Figura 3. DDL PARA DEFINIR PATROCINADOR, FESTIVAL , PALCO E LOCALIZAÇÃO



DML

DML ou Data Management Language é a ferramenta utilizada para inserir dados dentro da base de dados por nós criada, utilizando como é claro uma o tipo de dados corretos de dados para cada coluna. Muitos dos dados da nossa BD não são verdadeiros ,tal como o nome dos patrocinadores e dos artistas, os lugares dos acampamentos e dos festivais.

```
INSERT INTO SummerFest.Acampamento (ID_Acampamento, Nome, Duracao_Dias, Acomodidades, ID_Localizacao)
VALUES
(4989237, 'Acampamento da Herdade da Quinta Branca', 7, 'Bengaloos', 1312424),
(4389473, 'Penhasco de Idanha a Velha', 4, 'Tendas', 5231793),
(3137891, 'Parque de Campismo da Herdade da Quinta Branca', 8, 'Car-Camping', 1312424),
(4781323, 'Cabeço de Serra da Ponte Maior', 2, 'Tendas', 109067),
(2139832, 'Planície do lado velho do Alentejo', 3, 'Barracas', 9128310),
(1382903, 'Parque de Campismo de São Jacinto', 7, 'Tendas', 2839023),
(1238033, 'Car Camping de Portimão', 8, 'Car-Camping', 37445932);

INSERT INTO SummerFest.Artista (ID_Artista, Nome_Artistico, Nome_Verdadeiro, Estilo_Musical, Idade, Premios, Nacionalidade)
VALUES
(89321830, 'Don Toliver', 'Caleb Toliver', 'Rap', 28, 'Billboard Music Awards (United States)', 'United States'),
(48423893, 'Travis Scott', 'Jacques Webster', 'Trap', 27, 'Rock and Roll Hall of Fame (United States)', 'United States'),
(90543210, 'Lil Uzi Vert', 'Symere Woods', 'Trap', 27, 'American Music Awards (United States)', 'United States'),
(57940383, 'L7NNON', 'Lennon dos Santos Barbosa Frassetti', 'Reggae', 29, 'BET Awards (United States)', 'Brasil'),
(34980023, 'X-Tense', 'Nuno Barreiros', 'Rap-Tuga', 32, 'Ivor Novello Awards (United Kingdom)', 'Portugal'),
(32743284, 'Dillaz', 'André Saramago', 'Rap-Tuga', 34, 'Juno Awards (Canada)', 'Portugal'),
(74783973, '6lack', 'Ricardo Valentine', 'Rap Consciente', 60, 'BET Awards (United States)', 'United States'),
(89748318, 'The Weeknd', 'Abel Tesfaye', 'R&B', 33, 'Juno Awards (Canada)', 'Canada'),
(92813987, 'Matuê', 'Matheus Aguiar', 'Trap', 29, 'Mercury Prize (United Kingdom)', 'Brasil'),
(10312321, 'Zé Cabra', 'Casimiro Afonso', 'Pimba', 57, 'Rock and Roll Hall of Fame (United States)', 'Portugal'),
(11281301, 'Slow J', 'Joao Coelho', 'RapTuga', 30, 'Academy Awards (Oscars) for Best Original Song (United States)', 'Portugal');
```

Figura 4. DML PARA DEFINIR ACAMPAMENTO OU ARTISTA



STORED PROCEDURES

Uma Stored Procedure baseia-se numa batch armazenada com um dado nome e cujo código não tem de ser recompilado cada vez que a mesma é invocada, pois os procedimentos ficam guardados em memória cache na primeira vez que são executados, o que representa para nós uma vantagem na sua utilização para todas as operações de adicionar, eliminar e editar, como exemplificado a seguir:

```
--Festival
CREATE PROCEDURE SummerFest.AdicionarFestival (@ID_Festival INT,
                                                @Nome VARCHAR(64),
                                                @Data_De_Inicio DATE,
                                                @Duracao_Dias INT,
                                                @Lotacao_Maxima INT,
                                                @Lugar VARCHAR(64))
AS
BEGIN
    DECLARE @verification INT;
    DECLARE @erro VARCHAR(100);
    SET @verification = (SELECT dbo.checkID_Festival(@ID_Festival))
    IF(@verification>=1)
        RAISERROR ('O ID introduzido já existe, modifique-o para adicionar o festival!', 16,1);
    ELSE
        BEGIN
            DECLARE @ID_Localizacao INT;
            SET @ID_Localizacao = (SELECT dbo.BuscarID_Localizacao(@Lugar));

            BEGIN TRY
                INSERT INTO SummerFest.Festival (ID_Festival, Nome, Data_De_Inicio, Duracao_Dias, Lotacao_Maxima, ID_Localizacao)
                VALUES (@ID_Festival, @Nome, @Data_De_Inicio, @Duracao_Dias, @Lotacao_Maxima, @ID_Localizacao);
            END TRY
            BEGIN CATCH
                SELECT @erro = ERROR_MESSAGE();
                SET @erro = 'Não foi possível adicionar o festival, algum valor introduzido incorretamente!';
                RAISERROR (@erro, 16,1);
            END CATCH
        END
    END
GO
```

Figura 5. SP para adicionar Festival



STORED PROCEDURES

```
CREATE PROCEDURE SummerFest.EliminarFestival (@ID_Festival INT)
AS
BEGIN
    DECLARE @verification INT;
    DECLARE @erro VARCHAR(100);

    SET @verification = (SELECT dbo.checkID_Festival(@ID_Festival));

    IF (@verification = 0)
    BEGIN
        SET @erro = 'O ID do festival não existe, verifique o ID e tente novamente!';
        RAISERROR (@erro, 16, 1);
    END
    ELSE
    BEGIN
        BEGIN TRY
            BEGIN TRAN
                DELETE FROM SummerFest.Concerto WHERE ID_Festival = @ID_Festival;
                DELETE FROM SummerFest.Passe WHERE ID_Festival = @ID_Festival;
                DELETE FROM SummerFest.Festival WHERE ID_Festival = @ID_Festival;
            COMMIT TRAN
        END TRY
        BEGIN CATCH
            ROLLBACK TRAN
            SELECT @erro = ERROR_MESSAGE();
            SET @erro = 'Não foi possível eliminar o festival, algum valor introduzido incorretamente!';
            RAISERROR (@erro, 16, 1);
        END CATCH
    END
END
GO
```

Figura 6. SP para eliminar Festival

```
CREATE PROCEDURE SummerFest.EditarFestival (@ID_Festival INT,
                                             @Nome VARCHAR(64),
                                             @Data_De_Inicio DATE,
                                             @Duracao_Dias INT,
                                             @Lotacao_Maxima INT,
                                             @ID_Localizacao INT)
AS
BEGIN
    DECLARE @verification INT;
    DECLARE @erro VARCHAR(100);

    SET @verification = (SELECT dbo.checkID_Festival(@ID_Festival));

    IF (@verification = 0)
    BEGIN
        SET @erro = 'O ID do festival não existe, verifique o ID e tente novamente!';
        RAISERROR (@erro, 16, 1);
    END
    ELSE
    BEGIN
        SET NOCOUNT ON;
        BEGIN
            BEGIN TRY
                BEGIN TRAN
                UPDATE SummerFest.Festival
                SET
                    Nome = @Nome,
                    Data_De_Inicio = @Data_De_Inicio,
                    Duracao_Dias = @Duracao_Dias,
                    Lotacao_Maxima = @Lotacao_Maxima,
                    ID_Localizacao = @ID_Localizacao
                WHERE
                    ID_Festival = @ID_Festival;
                COMMIT TRAN
            END TRY
            BEGIN CATCH
                Rollback TRAN
                SELECT @erro = ERROR_MESSAGE();
                SET @erro = 'Não foi possível adicionar o festival, algum valor introduzido incorretamente!';
                RAISERROR (@erro, 16, 1);
            END CATCH
        END
    END
END
GO
```

Figura 7. SP para editar Festival

VIEWS

Quanto às views, estas podem ser usadas como fonte de dados para um conjunto de instruções SQL, como tal, desenvolvemos-las não só como forma de visualizar as tabelas na interface gráfica como também para facilitar a consulta na mesma de forma a permitir visualizar tabelas de dados específicos exemplificados nas figuras seguintes:

```
CREATE VIEW Concertos AS  
    SELECT * FROM SummerFest.Concerto  
  
CREATE VIEW Artistas AS  
    SELECT * FROM SummerFest.Artista
```

Figura 8.Views para Concertos e Artistas

```
CREATE VIEW ViewBarracas_Festival AS  
    SELECT b1.ID_Barraca , b1.Tipo_de_Alimentacao  
    FROM SummerFest.Barraca b1  
    INNER JOIN SummerFest.Festival_tem_Barracas fb1 ON fb1.ID_Barraca = b1.ID_Barraca  
    INNER JOIN SummerFest.Festival f1 ON f1.ID_Festival = fb1.ID_festival
```

Figura 9.View de Barraca para Festival

```
CREATE VIEW ViewPasses_Acampamentos AS  
    SELECT pa1.Numero_De_Serie , pa1.Duracao_Dias , pa1.Preco , pa1.ID_Festival  
    FROM SummerFest.Passe pa1  
    INNER JOIN SummerFest.Passe_Inclui_Acampamento pia1 ON pa1.Numero_De_Serie = pia1.Numero_De_Serie_Passe  
    INNER JOIN SummerFest.Acampamento a1 ON a1.ID_Acampamento = pia1.ID_Acampamento
```

Figura 10.View de Passes para Acampamentos



UDF's

Podendo estas aceitar argumentos de entrada e retornar valores, utilizámos as UDFs para verificações tais como se algum atributo, principalmente chaves primárias, já se encontravam definidas numa tabela e para retornar valores específicos, como podemos ver nos exemplos seguintes:

```
CREATE FUNCTION dbo.checkNumero_De_Serie_Passe(@Numero_De_Serie INT) RETURNS INT
AS
BEGIN
    DECLARE @counter INT
    SELECT @counter=COUNT(1) FROM SummerFest.Passe WHERE Numero_De_Serie=@Numero_De_Serie
    RETURN @counter
END
GO
```

```
CREATE FUNCTION dbo.BuscarID_Localizacao (@Lugar VARCHAR(64)) RETURNS INT
AS
BEGIN
    DECLARE @ID_Localizacao INT
    SELECT @ID_Localizacao=ID_Localizacao FROM SummerFest.Localizacao WHERE Lugar=@Lugar
    RETURN @ID_Localizacao
END
GO
```

Figura 10 e 11. UDF's para ir buscar ID da localização e o número de série



TRIGGERS

Os triggers são apenas executados em determinadas circunstâncias associadas à manipulação de dados, ou seja, quando uma ação prevista é realizada eles são "disparados". Utilizámos apenas triggers do tipo AFTER INSERT nas situações de adição de um artista menor de idade ou de adição de um palco cujo patrocinador não existe:

```
ALTER TRIGGER SummerFest.verificar_idade_artista ON SummerFest.Artista
AFTER INSERT
AS
BEGIN
    DELETE FROM SummerFest.Artista
    WHERE Artista.Idade < 18;
END;
GO
```

```
ALTER TRIGGER SummerFest.verificar_patrocinador_palco ON SummerFest.Palco
AFTER INSERT
AS
BEGIN
    IF NOT EXISTS (SELECT 1 FROM SummerFest.Patrocinador WHERE ID_Patrocinador IN (SELECT Patrocinador FROM inserted))
    BEGIN
        DELETE FROM SummerFest.Palco
        WHERE ID_Palco = (SELECT Patrocinador FROM inserted);
        RAISERROR ('O patrocinador especificado não existe.', 16, 1);
        RETURN;
    END
END
```

Figura 12 e 13.Triggers para verificar idade mínima e patrocinador existente



INDEXES

A implementação de indexes permite aumentar a eficiência na pesquisa de dados, logo, ainda que esta seja uma base de dados pequena decidimos utilizá-los para a pesquisa de um festival pelo seu ID e localização bem como na tabela localização para a pesquisa da mesma pelo lugar e ID:

```
1  CREATE INDEX searchID_Festival
2    ON SummerFest.Festival(ID_Festival)
3    GO
4
```

```
CREATE INDEX searchLocalizacao_Festival
ON SummerFest.Festival(ID_Localizacao)
GO
```

```
CREATE INDEX searchID_Localizacao
ON SummerFest.Localizacao(ID_Localizacao)
GO
```

```
CREATE INDEX searchLugar
ON SummerFest.Localizacao(Lugar)
GO
```

Figura 14, 15 e 16. Indexes para pesquisa eficiente nas tabelas Festival e Localização



CURSOR

O cursor permite percorrer sequencialmente os tuplos retornados por determinada consulta e como tal decidimos utilizá-lo para permitir a visualização dos passes disponíveis para cada festival, sendo que sempre que é adicionado ou eliminado um novo passe, este valor é alterado e, a nível de interface, pode ser visto através de um botão que abre uma message box com o valor.

```
CREATE PROCEDURE SummerFest.PassesDisponiveis (@ID_Festival INT, @PassesDisponiveis INT OUTPUT)
AS
BEGIN
    DECLARE @Passe_Contador INT;

    DECLARE PasseCursor CURSOR FOR
        SELECT COUNT(*) AS passe_contador
        FROM SummerFest.Passe
        WHERE ID_Festival = @ID_Festival;

    OPEN PasseCursor;

    FETCH NEXT FROM PasseCursor INTO @Passe_Contador;

    IF @@FETCH_STATUS = 0
    BEGIN
        SET @PassesDisponiveis = @Passe_Contador;
    END

    CLOSE PasseCursor;

    DEALLOCATE PasseCursor;
END
```

ID_Festival	Nome	Data_De_Inicio	Duracao_Dias	Lotacao_Maxima	ID_Localizacao
1	Reverberação de Verão	29/05/2022	3	16000	22
2	Festival Esplendor Estival	05/03/2023	6	2100	68
3	Festival Horizonte Harmônico	26/10/2023	-	9000	66
4	Eco do Éden	16/01/2023	-	30000	17
5	Extravagância Estival	23/08/2022	-	20000	64
6	Exuberância de Verão	24/05/2022	-	18000	95
7	Festival Eclipse Estival	03/03/2023	-	3000	87
8	Festival Vibrante Verão	29/08/2023	-	32000	7
9	Flamingo Flamboyant	20/08/2023	-	25000	8
10	Festival Aventura de Verão	23/12/2023	-	26000	4
11	Festival Onda de Verão	26/04/2023	5	1200	12

Figura 17. Cursor para visualizar os passes disponíveis para um festival





INTERFACE GRÁFICA

Como já foi referido anteriormente, a interface foi desenvolvida no Visual Studio, através da ferramenta Windows Forms, tendo sido o código escrito em C#.

O efeito de diferentes páginas foi conseguido através de panels sobrepostos que são exibidos ao clicar no botão do tema específico.

Quanto às tabelas, os dados são exibidos através de DataGridViews, permitindo a organização dos valores da tabela pela respetiva coluna. Para além disso, todos os panels têm no canto superior direito um conjunto de uma ComboBox com os nomes das colunas da tabela, uma TextBox para os dados de pesquisa e um botão de limpar filtros que auxiliam à pesquisa de dados por categoria.

Para permitir a manipulação de dados, todos os panels têm um conjunto de botões de adicionar, eliminar e editar a respetiva tabela, bem como uma TextBox para cada atributo da tabela permitindo ao utilizador inserir os dados desejados para realizar uma das operações. Caso um dos botões referidos seja clicado aparecerão também os botões de voltar, limpar dados (das TextBoxs) e o botão com a operação desejada.

Caso contrário, nalguns panels existem botões que tornam possível visualizar outras tabelas, como por exemplo, no panel Festival onde existem os botões Ver Artista, Ver Barraca e Ver Passes, que abrem um pop-up com um novo formulário com toda a informação.

Todas estas interações são facilitadas pela facilidade de, ao clicar numa célula da tabela, as TextBoxs são preenchidas com os dados de cada coluna daquela linha.

Por fim, ao realizar as operações de adicionar, eliminar ou editar, caso ocorra algum erro este será exibido numa MessageBox, bem como caso a operação seja sucedida é exibida da mesma forma uma mensagem informando que a ação foi realizada com sucesso sendo a tabela exibida ao utilizador atualizada com os respetivos dados.



Conclusão

Em conclusão, este projeto apresentou uma solução que nós achamos minimamente interessante para a gestão de festivais de verão, utilizando a SQL e a C# como ferramentas de desenvolvimento. A SQL mostrou ser crucial na construção de uma base de dados eficiente e segura, facilitando o armazenamento, organização e recuperação de informações, assegurando, assim, a integridade e a precisão dos dados. Por outro lado, a linguagem C# permitiu a construção de uma interface de utilizador amigável e intuitiva, além de integrar de maneira eficaz as funcionalidades da base de dados, facilitando a interação do utilizador final com o sistema.

Por fim, consideramos ter cumprido os objetivos a que nos propusemos para este projeto e agradecemos aos professores envolvidos por nos guiarem na realização do mesmo, tanto ao professor Carlos Costa na componente teórica como ao professor Joaquim Sousa Pinto na componente prática.

