

Gestão de Núcleos (APF)

Docentes Joaquim Sousa Pinto e Carlos Costa

Sistema de Gestão de Núcleos

Turma P10

Realizado por:

Diogo Falcão Nº108712 - Licenciatura em Engenharia Informática – 50%

Matilde Teixeira Nº108193 - Licenciatura em Engenharia Informática – 50%

Índice

Introdução	3
Ficheiros Entregues em Anexo.....	4
Análise de Requisitos.....	5
Diagrama Entidade-Relacionamento.....	6
Esquema Relacional	7
Modelo Relacional – SQL.....	8
Normalização	9
DDL e DML	11
Interface Gráfica.....	13
Ferramentas utilizadas.....	15
Conclusão	18

Introdução

O projeto final consiste na criação de um sistema de gestão de eventos para núcleos de estudantes da Universidade de Aveiro. Este sistema abrange a gestão de eventos, trabalhadores, e bens alimentares, entre outras funcionalidades necessárias para atender às necessidades dos núcleos. Com este sistema, os membros do núcleo poderão planejar, organizar e acompanhar eventos, atribuir tarefas aos trabalhadores, controlar a quantidade de alimentos e bebidas, e ter uma gestão mais eficiente e transparente das atividades. O objetivo final é desenvolver um sistema eficiente e prático, que atenda às necessidades dos núcleos e facilite a gestão dos mesmos.

Para planejar o sistema, foram utilizados o Diagrama Entidade-Relacionamento e o Esquema Relacional (disponíveis nos próximos tópicos), que permitiram uma melhor visualização e organização dos dados, bem como a implementação de consultas e relatórios necessários para a gestão dos núcleos.

Ficheiros Entregues em Anexo

- Tables.sql (Criação das tabelas da base de dados, com aplicação de DDL)
- Insertions.sql (Preencher as tabelas com dados)
- Indexes.sql (Todos os índices utilizados)
- Procedures.sql (Todos os Stored Procedures criados)
- UDFs.sql (Todos as UDF criadas)
- Views.sql (Todas as views criadas)
- Pasta “GestaoNucleos” com a interface criada para a gestão e manipulação da base de dados.
- Pasta “vídeo” com o vídeo da apresentação (demonstração da interface)

Análise de Requisitos

O intuito do sistema de gestão de núcleos é a capacidade de incorporar algumas funcionalidades de acordo com as seguintes condições:

- O Núcleo faz vários eventos e estes podem ser monetários ou não monetários. Os eventos são caracterizados por número, nome, local, data, tipo (convívios, palestras, sessões de dúvidas), área (pedagógica, cultural), número de trabalhadores e participantes;
- Os eventos monetários, para além das caracterizações já referidas, são ainda descritos com o código do evento monetário, dinheiro investido e lucro;
- É determinado que nos eventos monetários existam alimentos - comidas e bebidas. Estes incluem informações como código do produto, nome, preço de compra, preço de venda, quantidade, data de validade e tipo.
- Relacionado também a cada evento monetário, encontra-se a entidade Pulseira, onde cada uma tem um ID. Vai ser desta forma que o dinheiro irá entrar nas contas dos lucros do evento monetário.
- Os eventos não monetários são eventos onde o Núcleo não investe. O investimento ocorre por parte de um Sponsor. Assim, os eventos não monetários estão caracterizados pelo CC do Sponsor.
- Todos os eventos (monetários e não monetários) têm obrigatoriamente 1 ou mais trabalhadores. Cada trabalhador provém de uma relação IS-A da entidade Pessoa. Esta é caracterizada pelo CC, nome, morada, telemóvel e email. Por conseguinte, O trabalhador acrescenta o número de horas a estes atributos.
- Um trabalhador pode ser externo (identificado pelo salário/hora) ou pode ser membro do Núcleo (identificado pelo nº mecanográfico);
- O Responsável Financeiro é a entidade mais importante, uma vez que se encarrega de gerir toda a parte financeira do Núcleo. Este gere as pulseiras, os eventos monetários, os produtos e o pagamento a trabalhadores externos. O Responsável Financeiro provém da entidade Pessoa através de uma relação IS-A, o que faz com que tenha os atributos da mesma mais o seu ID, data de início e data de fim das suas funções.

Diagrama Entidade-Relacionamento

A seguir, encontra-se apresentado o Diagrama Entidade-Relacionamento do nosso projeto de Gestão de Núcleos. Esse diagrama tem como objetivo representar as entidades e as relações que existem no sistema de forma visual e organizada.

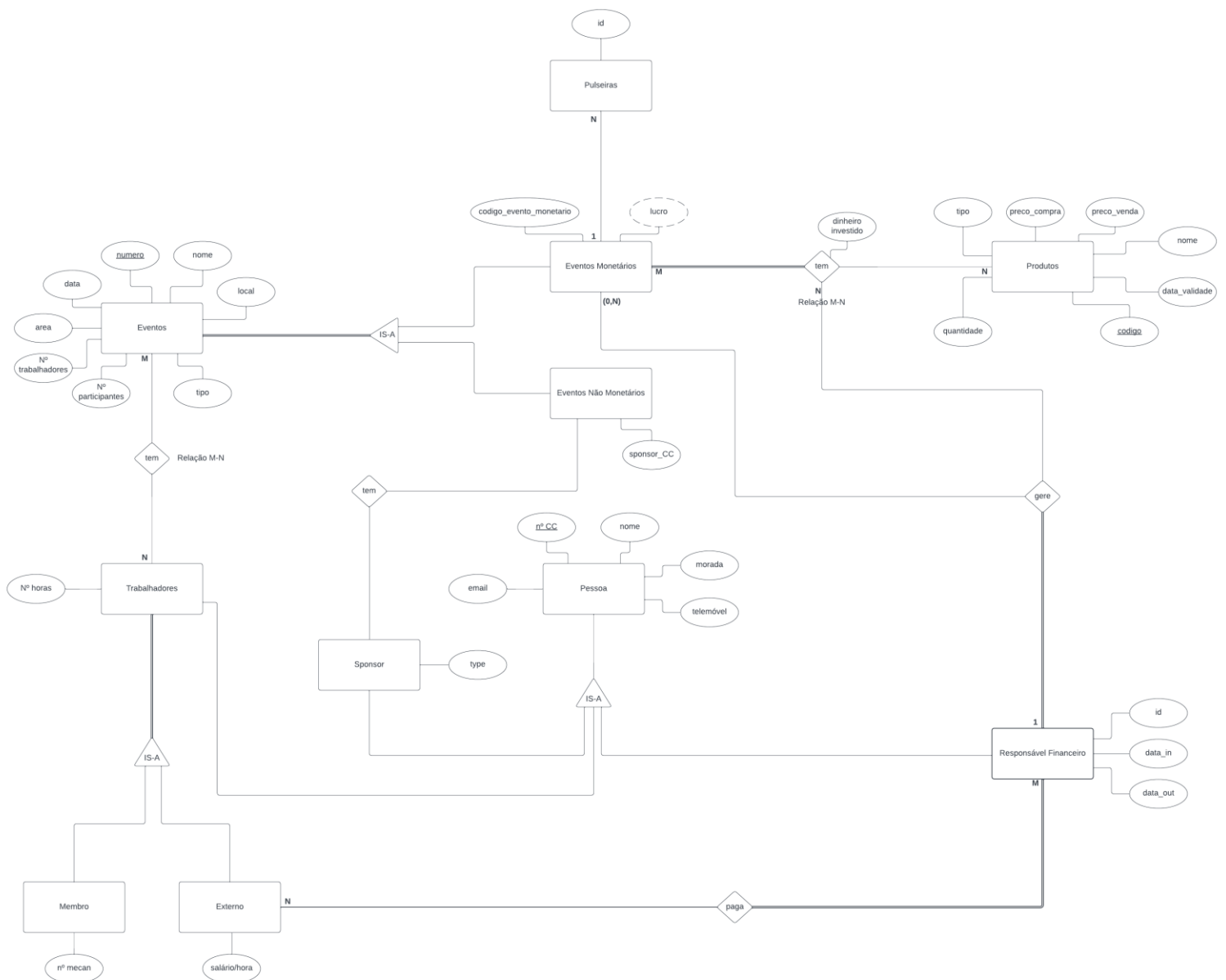


Fig.1 – Diagrama Entidade Relacionamento

Esquema Relacional

O esquema relacional é uma representação visual da estrutura de uma base de dados, permitindo identificar as tabelas, colunas e relações existentes entre elas. É fundamental para a organização dos dados e para garantir a consistência das informações. O esquema relacional do projeto de Gestão de Núcleos pode ser encontrado abaixo deste texto.

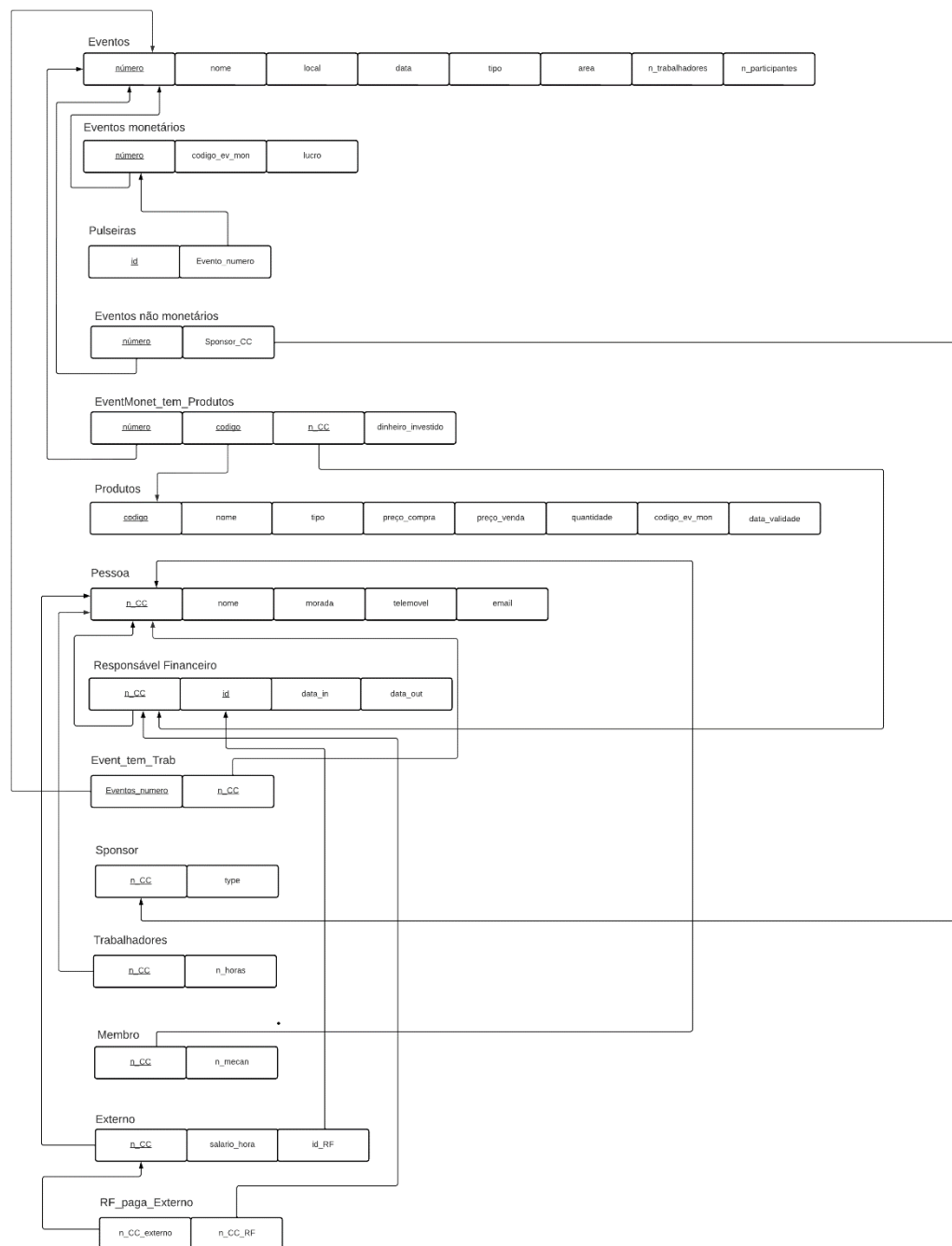


Fig. 2 – Esquema Relacional

Modelo Relacional – SQL

O Modelo Relacional - SQL é uma representação visual da estrutura de uma base de dados, permitindo identificar as tabelas, colunas e relações existentes. É fundamental para a organização dos dados e para garantir a consistência das informações. Em seguida, apresentamos o diagrama do Modelo Relacional gerado em SQL, após a construção das tabelas. Esse diagrama proporcionará uma visão clara e compreensível da estrutura da base de dados desenvolvida.

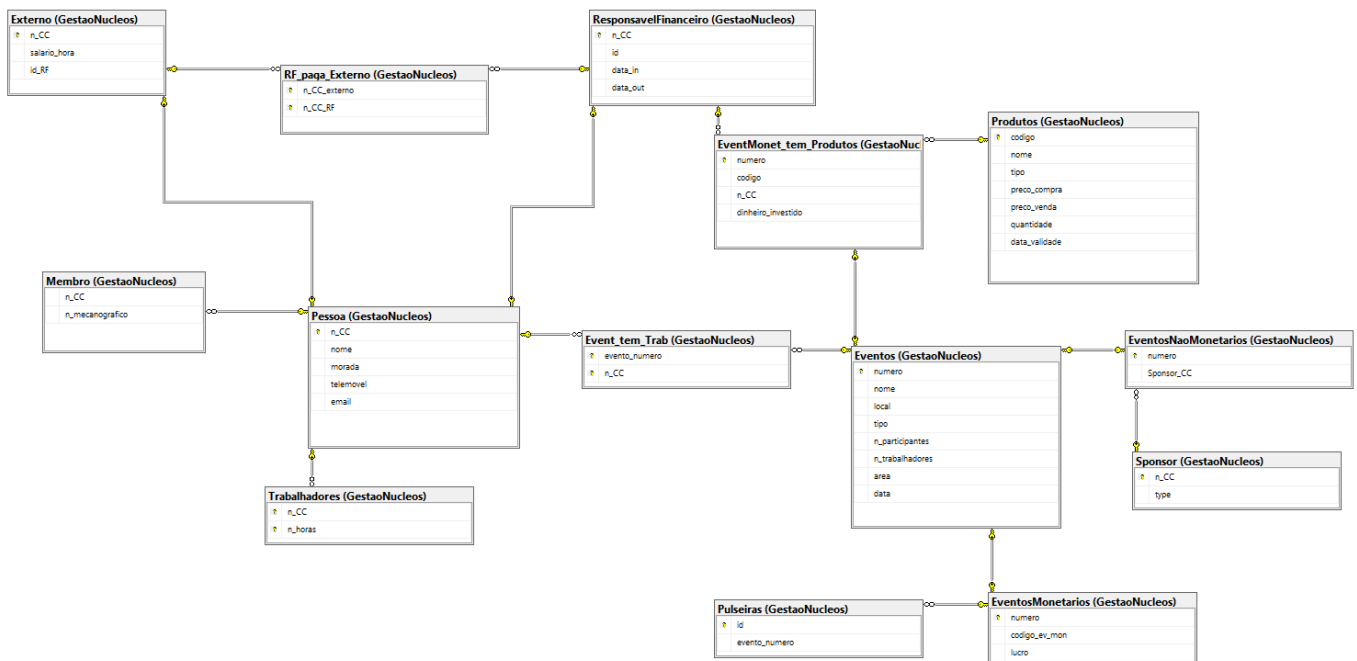


Fig. 3 – Modelo Relacional-SQL

Normalização

Nesta secção, iremos discutir os níveis de normalização alcançados no projeto. Com base nas tabelas fornecidas, foi aplicado um processo de normalização para garantir a organização eficiente e a integridade dos dados. A seguir, descreveremos os níveis de normalização atingidos para as tabelas relacionadas com os Eventos.

As tabelas *Eventos*, *EventosMonetarios*, *EventosNaoMonetarios*, *Produtos* e *EventMonet_tem_Produtos* foram projetadas com base nos conceitos de normalização. Estas foram normalizadas até a Terceira Forma Normal (3NF). A tabela *Eventos* representa as informações principais de cada evento, como o número do evento, o nome, o local, o tipo, o número de participantes, o número de trabalhadores, a área relacionada e a data do evento.

As tabelas *EventosMonetarios* e *EventosNaoMonetarios* estão relacionadas com a tabela *Eventos* através da chave primária número e representam, respetivamente, os eventos que possuem uma componente monetária e não monetária. A tabela *Produtos* contém informações sobre os produtos disponíveis, como código, nome, tipo, preço de compra, preço de venda, quantidade e data de validade. A tabela *EventMonet_tem_Produtos* relaciona os produtos a eventos monetários, através das chaves primárias número e código. Estas tabelas atendem aos requisitos da Primeira Forma Normal (1NF), uma vez que cada coluna contém apenas um valor e não há repetição de grupos de valores. Além disso, as chaves primárias garantem a unicidade de cada registo nas tabelas.

Em termos de Segunda Forma Normal (2NF), as tabelas também estão em conformidade, uma vez que não há dependências parciais nas colunas não chave. Todas as informações são diretamente dependentes das chaves primárias. Por fim, a Terceira Forma Normal (3NF) também é cumprida nas tabelas. As colunas não chave são dependentes exclusivamente das chaves primárias e não apresentam dependências transitivas.

Em conclusão, as tabelas relacionadas com os *Eventos* estão normalizadas até a Terceira Forma Normal (3NF), garantindo a minimização da redundância de dados e a integridade das informações relacionadas aos eventos.

DDL e DML

Para criar e gerir a base de dados, utilizamos a linguagem SQL, mais especificamente a linguagem de definição de dados (DDL) e a linguagem de manipulação de dados (DML). Desenvolvemos um esquema denominado "GestaoNucleos" para estruturar e organizar os dados. A linguagem de definição de dados (DDL) é utilizada para criar e modificar a estrutura da base de dados, incluindo a criação de tabelas, definição de colunas, restrições e índices. Por outro lado, a linguagem de manipulação de dados (DML) serve para realizar operações de consulta, inserção, atualização e exclusão de dados na base de dados. O esquema "GestaoNucleos" é uma representação organizada e estruturada da base de dados, onde são definidas as tabelas, relações e restrições necessárias para a gestão eficiente dos núcleos estudantis.

- Criação das tabelas com DDL:

```
43 CREATE TABLE GestaoNucleos.Pulseiras
44 (
45     [id] int NOT NULL,
46     [evento_numero] int NOT NULL,
47     PRIMARY KEY (id),
48     FOREIGN KEY (evento_numero) REFERENCES GestaoNucleos.EventosMonetarios (numero)
49 );
50
51 CREATE TABLE GestaoNucleos.EventosNaoMonetarios
52 (
53     [numero] int NOT NULL,
54     [Sponsor_CC] varchar(64) NOT NULL
55     PRIMARY KEY (numero),
56     FOREIGN KEY (numero) REFERENCES GestaoNucleos.Eventos (numero)
57 );
58
59 CREATE TABLE GestaoNucleos.Produtos
60 (
61     [codigo] int NOT NULL,
62     [nome] varchar(128),
63     [tipo] varchar(32) NOT NULL,
64     [preco_compra] float NOT NULL,
65     [preco_venda] float NOT NULL,
66     [quantidade] int NOT NULL,
67     [data_validade] datetime,
68     PRIMARY KEY (codigo),
69 );
```

- Fig. 4 – Exemplo DDL

- Inserção de dados com DML:

```

100 INSERT INTO GestaoNucleos.Produtos (codigo, nome, tipo, preco_compra, preco_venda, quantidade, data_validade) VALUES (11, N'Água Mineral', N'Bebida Não Alcoólica', 0.5, 1.0, 200, NULL);
101 INSERT INTO GestaoNucleos.Produtos (codigo, nome, tipo, preco_compra, preco_venda, quantidade, data_validade) VALUES (12, 'Salgadinhos de Festa', N'Alimentação', 2.0, 3.5, 150, '2024-04-30');
102 INSERT INTO GestaoNucleos.Produtos (codigo, nome, tipo, preco_compra, preco_venda, quantidade, data_validade) VALUES (13, 'Vinho Tinto', N'Bebida Alcoólica', 6.0, 10.0, 70, '2025-01-01');
103 INSERT INTO GestaoNucleos.Produtos (codigo, nome, tipo, preco_compra, preco_venda, quantidade, data_validade) VALUES (14, N'Champagne Moët & Chandon', N'Bebida Alcoólica', 40.0, 60.0, 30, '2024-04-30');
104 INSERT INTO GestaoNucleos.Produtos (codigo, nome, tipo, preco_compra, preco_venda, quantidade, data_validade) VALUES (15, 'Salsicha', N'Alimentação', 1.5, 3.0, 100, '2024-03-31');
105 INSERT INTO GestaoNucleos.Produtos (codigo, nome, tipo, preco_compra, preco_venda, quantidade, data_validade) VALUES (16, 'Sardinhas em Lata', N'Alimentação', 1.2, 2.5, 80, '2025-06-30');
106 INSERT INTO GestaoNucleos.Produtos (codigo, nome, tipo, preco_compra, preco_venda, quantidade, data_validade) VALUES (17, N'Café Solúvel', N'Bebida Não Alcoólica', 2.0, 4.0, 120, '2023-10-31');
107
108 -- Inserir dados relativos a Eventos Monetários
109 INSERT INTO GestaoNucleos.EventosMonetarios (numero, codigo_ev_mon, lucro) values (1, 200, 8.8);
110 INSERT INTO GestaoNucleos.EventosMonetarios (numero, codigo_ev_mon, lucro) values (3, 202, 1000.0);
111 INSERT INTO GestaoNucleos.EventosMonetarios (numero, codigo_ev_mon, lucro) values (6, 303, 200.0);
112 INSERT INTO GestaoNucleos.EventosMonetarios (numero, codigo_ev_mon, lucro) values (7, 457, 199.22);
113 INSERT INTO GestaoNucleos.EventosMonetarios (numero, codigo_ev_mon, lucro) values (8, 292, 2.03);
114 INSERT INTO GestaoNucleos.EventosMonetarios (numero, codigo_ev_mon, lucro) values (12, 301, -22.2);
115 INSERT INTO GestaoNucleos.EventosMonetarios (numero, codigo_ev_mon, lucro) values (13, 195, 111.01);
116 INSERT INTO GestaoNucleos.EventosMonetarios (numero, codigo_ev_mon, lucro) values (15, 920, 87.3);
117 INSERT INTO GestaoNucleos.EventosMonetarios (numero, codigo_ev_mon, lucro) values (16, 902, 89.9);
118 INSERT INTO GestaoNucleos.EventosMonetarios (numero, codigo_ev_mon, lucro) values (17, 650, 2.00);
119 INSERT INTO GestaoNucleos.EventosMonetarios (numero, codigo_ev_mon, lucro) values (18, 222, 14.0);
120 INSERT INTO GestaoNucleos.EventosMonetarios (numero, codigo_ev_mon, lucro) values (5, 455, 10897.41);
121 INSERT INTO GestaoNucleos.EventosMonetarios (numero, codigo_ev_mon, lucro) values (2, 234, 564.3);
122 INSERT INTO GestaoNucleos.EventosMonetarios (numero, codigo_ev_mon, lucro) values (20, 276, 2398.4);
123 INSERT INTO GestaoNucleos.EventosMonetarios (numero, codigo_ev_mon, lucro) values (21, 877, 232.0);
124
125 -- Inserir dados relativos a RF
126 INSERT INTO GestaoNucleos.ResponsavelFinanceiro(n_CC, id, data_in, data_out) values ('23456789', 5, '2022-09-30', null)
127
128 -- Inserir dados na relação EventMonet_tem_Produtos
129 INSERT INTO GestaoNucleos.EventMonet_tem_Produtos(numero, codigo, n_CC, dinheiro_investido) VALUES (1, 8, 23456789, 3455.6);
130 INSERT INTO GestaoNucleos.EventMonet_tem_Produtos(numero, codigo, n_CC, dinheiro_investido) VALUES (3, 8, 23456789, 657676.3);
131 INSERT INTO GestaoNucleos.EventMonet_tem_Produtos(numero, codigo, n_CC, dinheiro_investido) VALUES (6, 8, 23456789, 5456);
132 INSERT INTO GestaoNucleos.EventMonet_tem_Produtos(numero, codigo, n_CC, dinheiro_investido) VALUES (7, 9, 23456789, 656);

```

- Fig. 5 – Exemplo DML

Interface Gráfica

A plataforma escolhida para desenvolver a interface gráfica foi o C# onde através do Visual Studio, implementamos Windows Form da plataforma .NET.

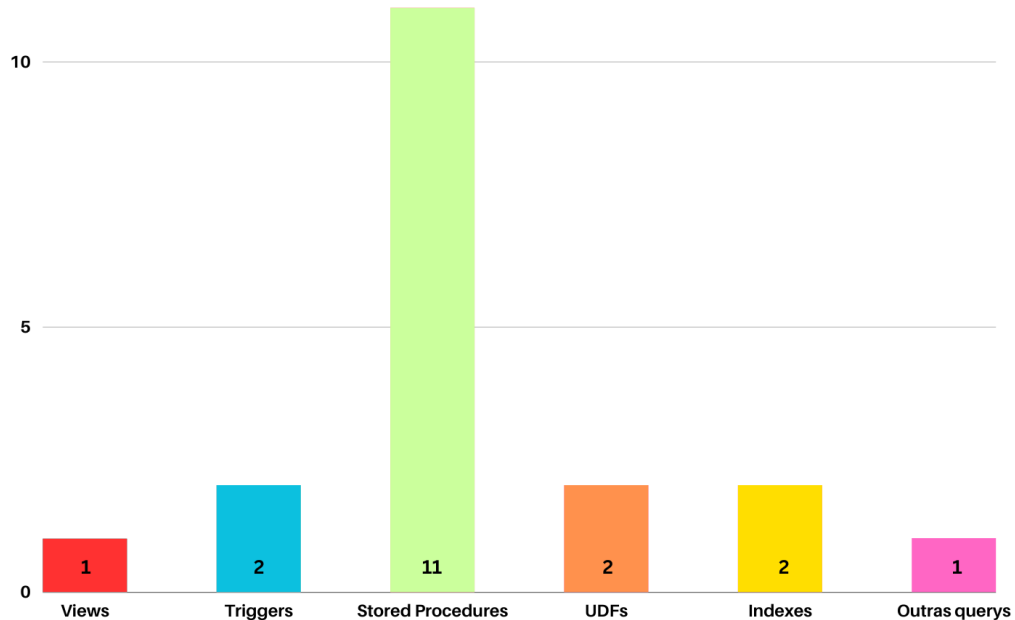
De modo a tornar a interface mais organizada, intuitiva e acessível ao utilizador, depois do login, dividimos a aplicação em 4 partes:

- Eventos (onde se pode pesquisar pelo nome dos eventos através da View *dbo.Relatorios*) e seleccionar ou Eventos Monetários ou Eventos Não Monetários.
 - Na secção do Eventos Monetários, podemos introduzir um número correspondente a determinado Evento e obter o lucro final relativo a este (SP *dbo.VerLucroTotal*). Podemos também adicionar um novo Evento Monetário onde o ID deste é automaticamente atribuído (SP *GestaoNucleos.AddEventoNaoMonetario*).
 - Na secção dos Eventos Não Monetários, podemos (para além de observar uma overview incluindo o número do evento, e o CC do sponsor do evento – SP *dbo.GetEventosNaoMonetarios*) introduzir o CC de determinado Sponsor a fim de obter as informações deste (*SELECT query*).
- Trabalhadores: aqui podemos procurar pelo nome de determinado trabalhador e ver o número de horas que trabalhou no total (SPs *dbo.GetTrabalhadorInfo* e *dbo.SearchTrabalhadorInfo*). Podemos editar um trabalhador ou eliminá-lo de determinado evento (SPs *dbo.AddTrabalhador* e SP *dbo.DelTrabalhador*). Na criação de um trabalhador, a incorreta inserção de dados pode resultar num disparar de um trigger (quer seja por introduzir um trabalhador com mais de 40 horas de trabalho ou introduzir um CC de um trabalhador que não contenha 8 caracteres).
- Pulseiras: nesta secção temos todas as pulseiras já criadas e uma opção de ver o número de pulseiras vendidas por número de Evento (SPs *dbo.GetPulseiraDetails* e *dbo.VerPulseiraPorEvento*).
- Armazém: finalmente, temos uma overview quer das comidas, quer das bebidas e a quantidade de cada comida/bebida (SPs *dbo.VerBebida* e *dbo.VerComida* e UDFs *dbo.ContarBebidas* e *dbo.ContarAlimentacao*).

Aqui estão apresentadas as funcionalidades principais da aplicação. É preciso notar que de modo a tornar estas funcionalidades possíveis, utilizamos comandos SQL-DML (INSERT, DELETE e UPDATE), Views, Index, UDF, Stored Procedure e Triggers. Pelo menos um exemplo de cada tópico irá ser apresentado na secção “Ferramentas Utilizadas” abaixo.

Ferramentas utilizadas

De seguida, apresentamos um gráfico com a quantidade de ferramentas utilizadas.



• Fig. 5 – Ferramentas Utilizadas

Exemplos das ferramentas:

- Views:

```
25 -- Nome, numero, local, data, codigo_event_monet, lucro, numero de trabalhadores e numero de participantes
26 CREATE VIEW INFO_EVENTO AS
27     SELECT GestaoNucleos.Eventos.numero, nome, local, data, codigo_ev_mon, lucro
28     FROM GestaoNucleos.Eventos JOIN GestaoNucleos.EventosMonetarios ON GestaoNucleos.Eventos.numero=GestaoNucleos.EventosMonetarios.numero;
```

- Triggers:

```
CREATE TRIGGER Trg_ControlarHorarioTrabalhadores
ON GestaoNucleos.Trabalhadores
FOR INSERT
AS
BEGIN
    DECLARE @n_CC varchar(64), @n_horas float;

    SELECT @n_CC = n_CC, @n_horas = n_horas
    FROM inserted;

    IF EXISTS (
        SELECT 1
        FROM GestaoNucleos.Trabalhadores
        WHERE n_CC = @n_CC AND n_horas + @n_horas > 40
    )
    BEGIN
        RAISERROR ('O trabalhador atingiu o limite de 40 horas de trabalho.', 16, 1);
        ROLLBACK TRANSACTION;
        RETURN;
    END;
```

- Stored Procedures:

```

35 CREATE PROCEDURE [dbo].[AddTrabalhador]
36     @numero int,
37     @n_CC varchar(64),
38     @nome varchar(128),
39     @morada varchar(256),
40     @telemovel int,
41     @email varchar(128),
42     @n_horas float
43 AS
44 BEGIN
45     BEGIN TRY
46
47         -- Verificar se o número de evento existe na tabela GestaoNucleos.Eventos
48         IF EXISTS (SELECT 1 FROM GestaoNucleos.Eventos WHERE numero = @numero)
49             BEGIN
50                 INSERT INTO GestaoNucleos.Pessoa(n_CC, nome, morada, telemovel, email)
51                 VALUES (@n_CC, @nome, @morada, @telemovel, @email);
52
53                 IF EXISTS (SELECT 1 FROM GestaoNucleos.Pessoa WHERE n_CC = @n_CC)
54                     BEGIN
55                         INSERT INTO GestaoNucleos.Event_tem_Trab(evento_numero, n_CC)
56                         VALUES (@numero, @n_CC);
57
58                         INSERT INTO GestaoNucleos.Trabalhadores(n_CC, n_horas)
59                         VALUES (@n_CC, @n_horas);
60                     END
61                 END
62             ELSE
63                 BEGIN
64                     PRINT 'Evento nao existe, impossivel introduzir trabalhador';
65                 END;
66
67             END TRY
68             BEGIN CATCH
69                 DECLARE @ErrorMessage NVARCHAR(4000);
70                 SET @ErrorMessage = ERROR_MESSAGE();
71                 RAISERROR(@ErrorMessage, 16, 1);
72                 SELECT ERROR_MESSAGE() AS ErrorMessage;
73             END CATCH;
74     END;
75 GO

```

- Uder Defined Functions:

```

25 CREATE FUNCTION dbo.ContarBebidas ()
26 RETURNS INT
27 AS
28 BEGIN
29     DECLARE @countProdutosBebidas INT;
30
31     SELECT @countProdutosBebidas = COUNT()
32     FROM GestaoNucleos.Produtos
33     WHERE tipo LIKE '%Bebida%';
34
35     RETURN @countProdutosBebidas;
36 END;

```

- Indexes:


```
25 | -- Eventos com determinado nome
26 | DROP INDEX IF EXISTS nome_event ON GestaoNucleos.Eventos;
27 | CREATE INDEX nome_event ON GestaoNucleos.Eventos(nome);
```

Conclusão

Em suma, o objetivo deste trabalho foi alcançado ao aplicarmos os conceitos aprendidos durante as aulas. Foi uma experiência extremamente enriquecedora e elucidativa, permitindo-nos obter uma compreensão mais aprofundada sobre o funcionamento de uma base de dados e todo o processo envolvido em sua criação.

Visando aprimorar ainda mais este trabalho, temos o interesse em implementar medidas de segurança adicionais, com o intuito de garantir uma proteção robusta dos dados.

Gostaríamos apenas de mencionar que não foi possível implementar todas as funcionalidades desejadas na interface. No entanto, conseguimos aplicar todas as funções necessárias, com exceção dos cursores e das medidas de segurança. Apesar dessas limitações, o projeto foi capaz de abranger completamente o conteúdo aprendido e demonstrar a aplicação dos conceitos em toda a interface.