universidade de aveiro deti departamento de electrónica, telecomunicações e informática

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

Entrega do Trabalho Final - Instruções

A data limite de entrega do trabalho e relatório é **dia 6 de Junho às 23:59** e **todos têm de submeter o seu trabalho**.

Devem entregar um ficheiro zip contendo:

1. Um **relatório completo** sobre o trabalho desenvolvido, evidenciando e contextualizando os conhecimentos aplicados. (ficheiro PDF)

Exemplo de conteúdos a incluir no relatório: Análise de requisitos, DER, Esquema Relacional da BD, SQL DDL associados à definição da estrutura da BD em SQL Server, SQL DML associadas a cada formulário gráfico, Normalização, Índices, Triggers, Stored Procedures, UDF, etc.

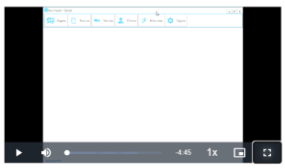
2. **Trabalho Final**

a. Base de Dados: Mesmo que tenham desenvolvido noutro SGBD, devem replicar todo o vosso trabalho no servidor das aulas. São obrigados a submeter todas as scripts SQL que permitiram criar (replicar) o vosso trabalho. Recomenda-se que criem um ficheiro por tópico (SQL DDL, Inserções, SP, UDF, Triggers, etc).

b. Formulário gráfico de interacção:

1. Projecto - submeter o projecto completo (source code), a compilar e com as DB connection strings a apontar para a base de dados da aula. Devem ainda identificar o local (form, ficheiro, etc) onde devemos alterar o vosso utilizador da base de dados (para o nosso) de forma a testar a vossa aplicação;
2. Relatório Complementar - caso optem por não utilizar Windows Forms em Visual Studio (C#).

c. Video demonstrador: Devem criar um video onde demonstrem as funcionalidades da vossa aplicação de base de dados. Ver o seguinte exemplo de um trabalho realizado numa edição anterior da disciplina:



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 planear, 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 planear 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

- Tabelas.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)
- UDF.sql (Todos as UDF criadas)
- Views.sql (Todas as views criadas)
- Pasta “Interface” com a interface criada para a gestão e manipulação da base de dados.
- Vídeo da apresentação (vídeo com uma pequena 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.

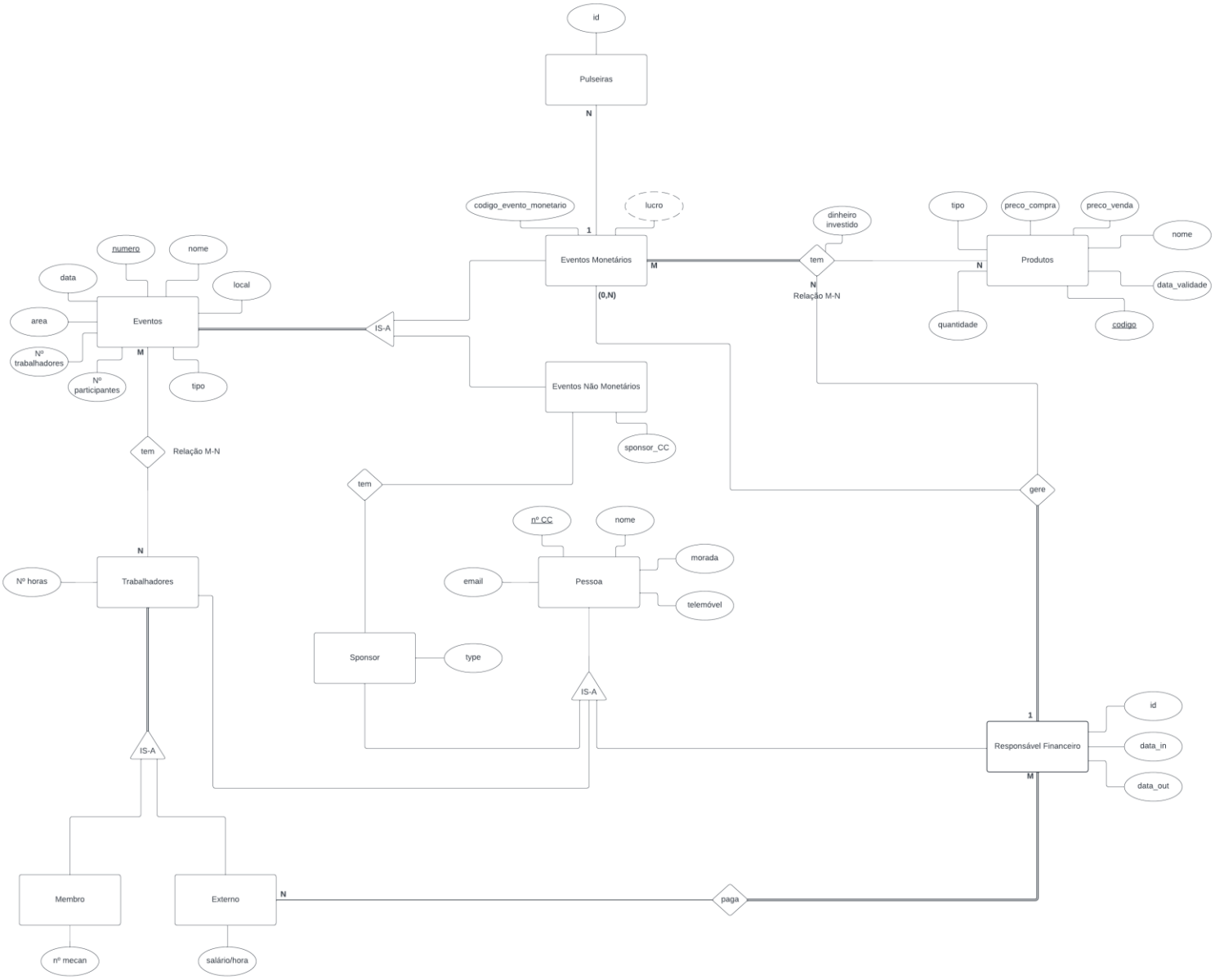


Fig.1 – Diagrama Entidade Relacionamento

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.

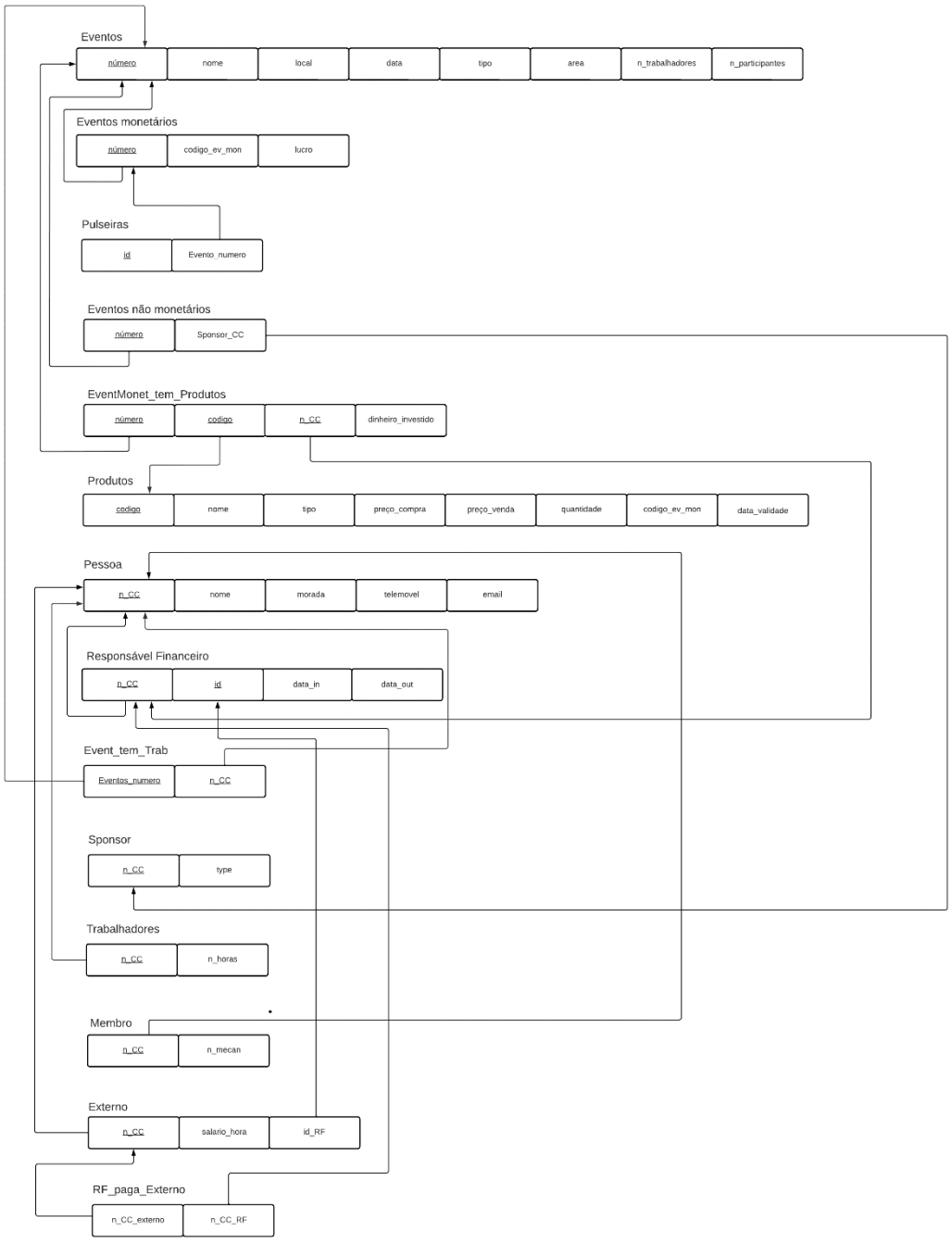


Fig. 2 – Esquema Relacional

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.

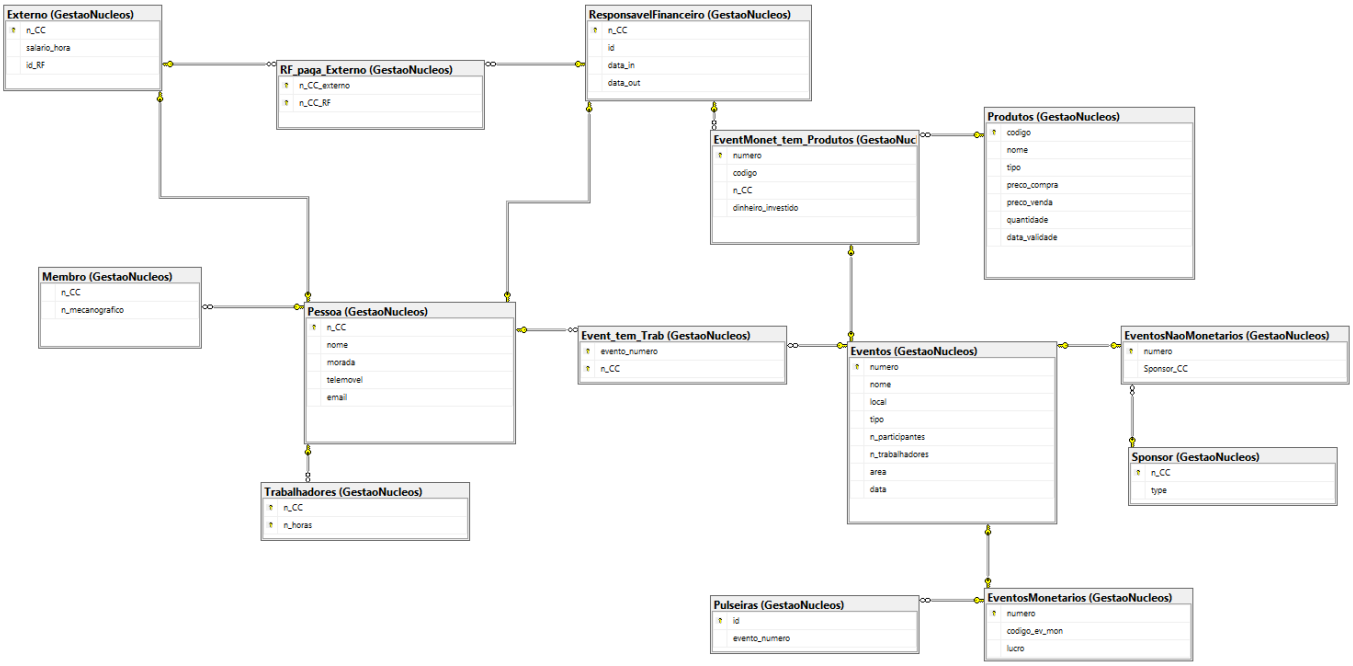


Fig. 3 – Modelo Relacional-SQL

Normalização

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:

```
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);
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');
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');
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
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');
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');
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');

-- Inserir dados relativos a Eventos Monetários
INSERT INTO GestaoNucleos.EventosMonetarios (numero, codigo_ev_mon, lucro) values (1, 200, 8.8);
INSERT INTO GestaoNucleos.EventosMonetarios (numero, codigo_ev_mon, lucro) values (3, 202, 1000.0);
INSERT INTO GestaoNucleos.EventosMonetarios (numero, codigo_ev_mon, lucro) values (6, 303, 200.0);
INSERT INTO GestaoNucleos.EventosMonetarios (numero, codigo_ev_mon, lucro) values (7, 457, 199.22);
INSERT INTO GestaoNucleos.EventosMonetarios (numero, codigo_ev_mon, lucro) values (8, 292, 2.03);
INSERT INTO GestaoNucleos.EventosMonetarios (numero, codigo_ev_mon, lucro) values (12, 301, -22.2);
INSERT INTO GestaoNucleos.EventosMonetarios (numero, codigo_ev_mon, lucro) values (13, 195, 111.01);
INSERT INTO GestaoNucleos.EventosMonetarios (numero, codigo_ev_mon, lucro) values (15, 920, 87.3);
INSERT INTO GestaoNucleos.EventosMonetarios (numero, codigo_ev_mon, lucro) values (16, 902, 89.9);
INSERT INTO GestaoNucleos.EventosMonetarios (numero, codigo_ev_mon, lucro) values (17, 650, 2.08);
INSERT INTO GestaoNucleos.EventosMonetarios (numero, codigo_ev_mon, lucro) values (18, 222, 14.0);
INSERT INTO GestaoNucleos.EventosMonetarios (numero, codigo_ev_mon, lucro) values (5, 455, 10897.41);
INSERT INTO GestaoNucleos.EventosMonetarios (numero, codigo_ev_mon, lucro) values (2, 234, 564.3)
INSERT INTO GestaoNucleos.EventosMonetarios (numero, codigo_ev_mon, lucro) values (20, 276, 2398.4);
INSERT INTO GestaoNucleos.EventosMonetarios (numero, codigo_ev_mon, lucro) values (21, 877, 232.0);

-- Inserir dados relativos a RF
INSERT INTO GestaoNucleos.ResponsavelFinanceiro(n_CC, id, data_in, data_out) values ('23456789', 5, '2022-09-30', null)

-- Inserir dados na relação EventMonet_tem_Produtos
INSERT INTO GestaoNucleos.EventMonet_tem_Produtos(numero, codigo, n_CC, dinheiro_investido) VALUES (1, 8, 23456789, 3455.6);
INSERT INTO GestaoNucleos.EventMonet_tem_Produtos(numero, codigo, n_CC, dinheiro_investido) VALUES (3, 8, 23456789, 657676.3);
INSERT INTO GestaoNucleos.EventMonet_tem_Produtos(numero, codigo, n_CC, dinheiro_investido) VALUES (6, 8, 23456789, 5456);
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.

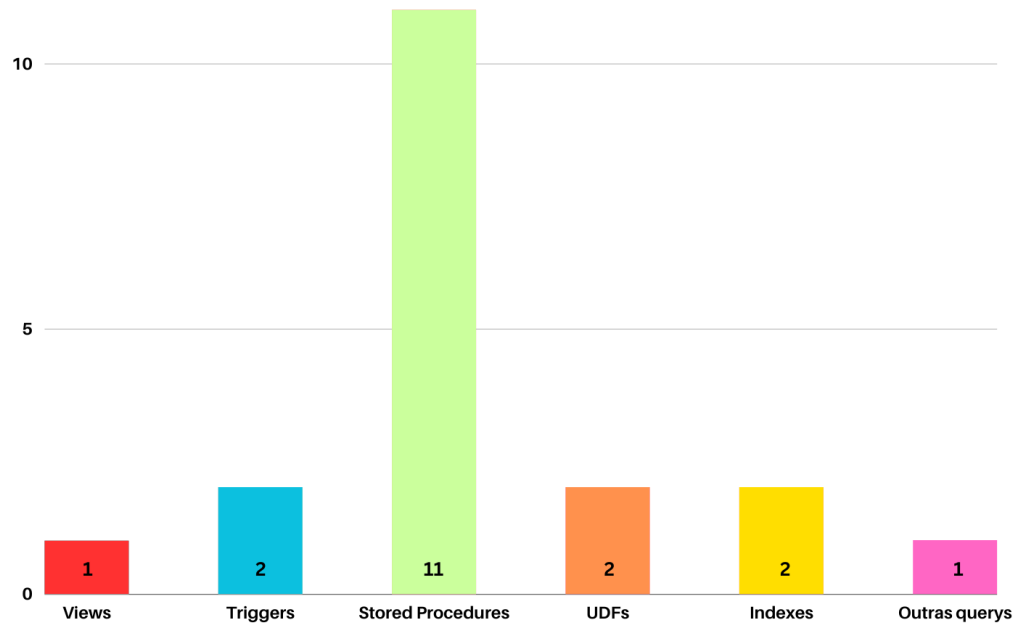
De modo a tornar a interface mais organizada, intuitiva e acessível ao utilizador, depois do login, dividimos a aplicação em 4 partes:

- Eventos (onde se pode pesquisar pelo nome dos eventos através da View *dbo.Relatorios*) e selecionar 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:

```
-- Nome, numero, local, data, codigo_event_monet, lucro, numero de trabalhadores e numero de participantes
CREATE VIEW INFO_EVENTO AS
    SELECT GestaoNucleos.Eventos.numero, nome, local, data, codigo_ev_mon, lucro
    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:

```
CREATE PROCEDURE [dbo].[AddTrabalhador]
  @numero int,
  @n_CC varchar(64),
  @nome varchar(128),
  @morada varchar(256),
  @telemovel int,
  @email varchar(128),
  @n_horas float
AS
BEGIN
  BEGIN TRY

    -- Verificar se o número de evento existe na tabela GestaoNucleos.Eventos
    IF EXISTS (SELECT 1 FROM GestaoNucleos.Eventos WHERE numero = @numero)
    BEGIN
      INSERT INTO GestaoNucleos.Pessoa(n_CC, nome, morada, telemovel, email)
      VALUES (@n_CC, @nome, @morada, @telemovel, @email);

      IF EXISTS (SELECT 1 FROM GestaoNucleos.Pessoa WHERE n_CC = @n_CC)
      BEGIN
        INSERT INTO GestaoNucleos.Event_tem_Trab(evento_numero, n_CC)
        VALUES (@numero, @n_CC);

        INSERT INTO GestaoNucleos.Trabalhadores(n_CC, n_horas)
        VALUES (@n_CC, @n_horas);
      END
    END
    ELSE
    BEGIN
      PRINT 'Evento nao existe, impossivel introduzir trabalhador';
    END;

  END TRY
  BEGIN CATCH
    DECLARE @ErrorMessage NVARCHAR(4000);
    SET @ErrorMessage = ERROR_MESSAGE();
    RAISERROR(@ErrorMessage, 16, 1);
    SELECT ERROR_MESSAGE() AS ErrorMessage;
  END CATCH;
END;
GO
```

- Uder Defined Functions:

```
CREATE FUNCTION dbo.ContarBebidas ()
RETURNS INT
AS
BEGIN
    DECLARE @countProdutosBebidas INT;

    SELECT @countProdutosBebidas = COUNT()
    FROM GestaoNucleos.Produtos
    WHERE tipo LIKE '%Bebida%';

    RETURN @countProdutosBebidas;
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.