Escola Profissional

BENTO DE JESUS CARAÇA

DELEGAÇÃO DO BARREIRO

PROJETO - BASE DE DADOS

CURSO PROFISSIONAL DE TÉCNICO DE GESTÃO E PROGRAMAÇÃO DE SISTEMAS INFORMÁTICOS

TURMA 2023/2026

Índice

Índice de Figuras

1. Introdução

Este trabalho foi desenvolvido no âmbito da disciplina de Programação de Sistemas Informáticos (PSI) e tem como objetivo a criação de uma base de dados relacional, desde a parte da realização dos modelos até à implementação do código final. A realização deste projeto foi proposta pelo professor André Rolo.

O objetivo deste projeto é aplicar conhecimentos sobre criação de base de dados, desde a estruturação de tabelas, definição de relações e implementação em SQL, através do desenvolvimento de um sistema funcional.

Entre as várias propostas apresentadas pelo professor, foi escolhida a base de dados de um parque de diversões, onde o objetivo é organizar e gerir as informações relacionadas com as atrações, visitantes, funcionários e manutenções.

Este tema permite explorar diferentes tipos de entidades e relações, representando de forma prática e realista o funcionamento de um sistema de gestão de um ambiente recreativo.

2. Objetivos

O principal objetivo deste projeto é adquirir e desenvolver uma base de dados relacional funcional que permita gerir as informações de um parque de diversões, assegurando a organização e integridade dos dados. De forma mais específica, este trabalho tem como objetivos:

- Aplicar os conhecimentos teóricos sobre bases de dados adquiridos nas aulas de PSI;

- Compreender o processo de modelação de dados, desde o modelo conceitual (Entidade-Relacionamento) até ao modelo lógico (Relacional);

- Desenvolver a capacidade de transformar um modelo conceptual em código SQL, criando tabelas, chaves primárias e estrangeiras, e relações entre entidades;

- Simular um ambiente real de gestão, onde é possível armazenar e consultar informações relacionadas com atrações, visitantes, funcionários, bilhetes e manutenções;

- Melhorar o raciocínio lógico e a capacidade de resolução de problemas, através da análise e implementação de soluções práticas;

- Consolidar o uso do MySQL como ferramenta de gestão e manipulação de bases de dados.

Este projeto, sendo o primeiro desenvolvido integralmente pelos alunos, visa ainda reforçar a autonomia, o trabalho em grupo e a compreensão prática dos conceitos fundamentais da programação de sistemas informáticos.

3. Metodologia

3.1 Diagramas

Modelo ER

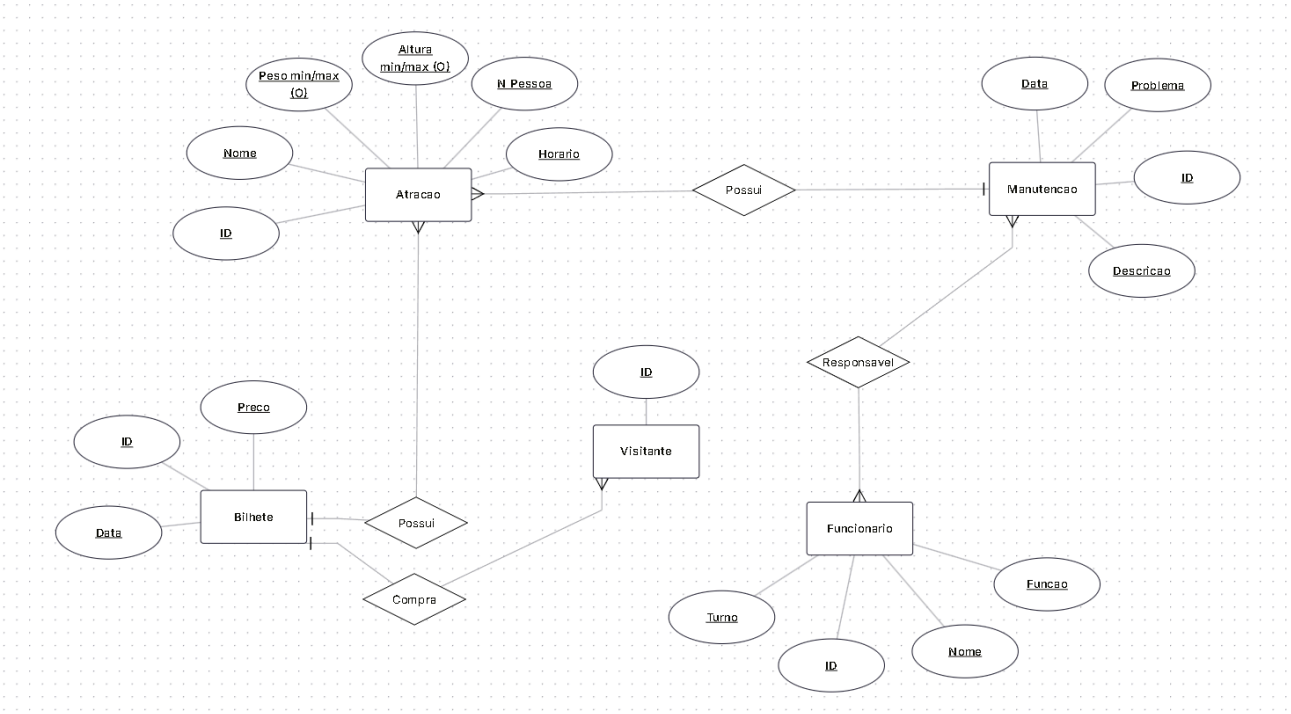

Figura 1 – Diagrama ER do Parque de Diversões

O diagrama ER apresenta as entidades principais do sistema, os seus atributos e as relações entre elas. As entidades identificadas são: Atração, Manutenção, Funcionário, Bilhete e Visitante.

- Atração: Representa cada brinquedo ou equipamento do parque. Inclui informações como nome, preço, horário, número máximo de pessoas e limites físicos (peso e altura mínimos/máximos).

- Manutenção: Guarda os registos de todas as manutenções realizadas nas atrações. Contém dados como data, problema identificado e descrição da intervenção.

- Funcionário: Representa os trabalhadores do parque, armazenando o nome, função, turno e a data de registo.

- Bilhete: Regista as vendas de bilhetes, incluindo o preço, data da compra e a atração associada.

- Visitante: Identifica os clientes que frequentam o parque, sendo utilizado para associar bilhetes às pessoas que os compram.

Relações (Losangos) :

1. Atração → Manutenção (1:N)
   Cada atração pode ter várias manutenções ao longo do tempo, mas cada manutenção pertence apenas a uma atração.
   Isto representa uma relação um-para-muitos, onde o lado “muitos” (N) está em *Manutenção*.

2. Manutenção ↔ Funcionário (N:M)
   Uma manutenção pode envolver vários funcionários, e um funcionário pode participar em várias manutenções.
   Esta relação muitos-para-muitos é resolvida por uma tabela intermédia chamada *Manutenção_Funcionário*, que armazena as chaves de ambas as tabelas (ID do Funcionário e ID da Manutenção).

3. Atração → Bilhete (1:N)
   Uma atração pode ter vários bilhetes emitidos, mas cada bilhete pertence apenas a uma atração.
   Esta relação liga as vendas ao tipo de atração.

4. Visitante → Bilhete (1:N)
   Um visitante pode comprar vários bilhetes, mas cada bilhete pertence a apenas um visitante.
   Esta relação permite associar as vendas aos clientes que as realizaram.

Cada relação no diagrama ER foi feita de forma a que haja coerência nos dados, evitando duplicações e assegurando que cada entidade tem um papel claro no sistema.

Modelo Relacional

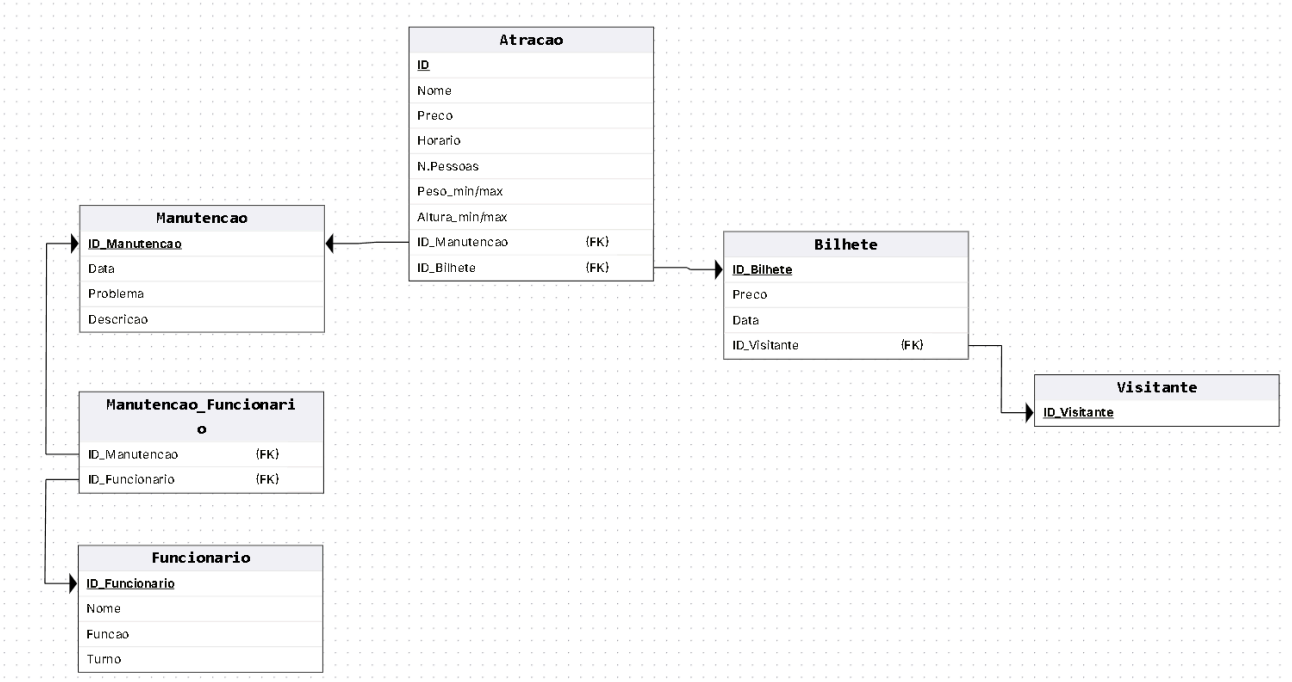

Figura 2 – Diagrama Relacional do Parque de Diversões

O diagrama relacional mostra a transformação do diagrama ER em tabelas com chaves primárias (PK) e chaves estrangeiras (FK), representando como os dados são realmente guardados na base de dados.

- Atração: A tabela principal que guarda as informações sobre cada atração. É referenciada por outras tabelas através da sua chave primária ID.

- Manutenção: Inclui a coluna AtracaoID como chave estrangeira (FK) para garantir a ligação entre a manutenção e a atração.

- Funcionário: Contém informações dos trabalhadores do parque.

- Manutenção_Funcionário: Tabela associativa que representa a relação N:M entre Manutenção e Funcionário. As suas chaves primárias são compostas por ID_Manutencao e ID_Funcionario.

- Bilhete: Tabela que guarda as vendas. Possui duas chaves estrangeiras: AtracaoID (ligação 1:N com Atração) e VisitanteID (ligação 1:N com Visitante).

- Visitante: Tabela que identifica os visitantes e está ligada aos bilhetes que cada um comprou.

As relações representadas no diagrama ER foram convertidas em ligações entre tabelas através das chaves estrangeiras.

Estas chaves garantem a integridade referencial, impedindo, por exemplo, que uma manutenção exista sem estar ligada a uma atração, ou que um bilhete seja criado sem um visitante.

3.3 Lógica

A lógica da base de dados define a forma como as entidades se relacionam entre si e como os dados são organizados, garantindo a melhor eficiência do sistema. Esta lógica foi aplicada nos modelos Entidade-Relacionamento (ER) e Relacional (RS).

Cada atração pode ter vários bilhetes e várias manutenções, uma vez que diferentes visitantes podem utilizá-la e diferentes intervenções de manutenção podem ocorrer ao longo do tempo.

A entidade visitante foi ligada ao bilhete através de uma relação de 1:N, pois um visitante pode adquirir vários bilhetes, mas cada bilhete pertence apenas a um visitante.

A entidade funcionário está relacionada com a manutenção numa relação N:M, pois uma manutenção pode envolver vários funcionários e um funcionário pode participar em várias manutenções. Esta relação é resolvida através de uma tabela intermédia.

Alguns atributos como o peso mínimo/máximo e altura mínima/máxima são opcionais, pois nem todas as atrações possuem um limite de peso ou altura.

O objetivo desta estrutura é garantir a integridade dos dados e permitir uma gestão eficaz de bilhetes, visitantes, atrações e manutenções.

3.4 Código SQL

Após a definição dos modelos e da lógica do sistema, foi criada a base de dados que representa as entidades e as relações identificadas no modelo Entidade-Relacionamento.

O código SQL abaixo define todas as tabelas, as suas chaves primárias e estrangeiras, as restrições de integridade e os índices necessários para garantir um bom desempenho.

Cada tabela foi criada com o objetivo de representar fielmente os elementos do parque de diversões, permitindo registar atrações, manutenções, funcionários, visitantes e bilhetes.

```sql
CREATE TABLE Atracao (
    ID INT PRIMARY KEY AUTO_INCREMENT,
    Nome VARCHAR(100) NOT NULL,
    Preco DECIMAL(6,2) NOT NULL DEFAULT 0.00,
    Horario_inicio TIME,
    Horario_fim TIME,
    Num_Pessoas INT NOT NULL DEFAULT 1,
    Peso_Min DECIMAL(5,2),
    Peso_Max DECIMAL(5,2),
    Altura_Min DECIMAL(5,2),
    Altura_Max DECIMAL(5,2),
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    CHECK (Num_Pessoas >= 1)
);
```

Figura 3 – Tabela Atracao em SQL

A tabela Atracao foi criada com o objetivo de armazenar todas as informações relacionadas com as atrações do parque, como o nome, o preço, o horário de funcionamento, a capacidade máxima e limites de peso e altura associados a cada uma. O campo ID funciona como chave primária e é gerado automaticamente, garantindo que cada atração seja identificada de forma única. O atributo Nome é obrigatório, assegurando que nenhuma atração seja registada sem identificação. Já o campo Preco define o valor do bilhete correspondente a cada atração, sendo armazenado com duas casas decimais para maior precisão.

Os campos Horario_inicio e Horario_fim indicam o período de funcionamento diário de cada atração, enquanto Num_Pessoas representa a capacidade máxima permitida, com uma verificação que impede a inserção de valores inferiores a 1. Além disso, os atributos Peso_Min, Peso_Max, Altura_Min e Altura_Max registam os limites físicos de segurança exigidos para utilização da atração, podendo ficar em branco (NULL) quando não se aplicam. Por fim, o campo created_at regista automaticamente a data e a hora em que o registo foi criado, facilitando o controlo e a gestão temporal dos dados.

```sql
CREATE TABLE Funcionario (
    ID INT PRIMARY KEY AUTO_INCREMENT,
    Nome VARCHAR(100) NOT NULL,
    Funcao VARCHAR(50),
    Turno VARCHAR(50),
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP
);
```

Figura 4 – Tabela Funcionario em SQL

A tabela Funcionario foi criada para registar todos os trabalhadores responsáveis por todas as funções do parque. Esta tabela permite identificar cada funcionário através do campo ID, definido como chave primária que é gerada automaticamente. Os atributos Nome, Funcao e Turno guardam respetivamente o nome completo do funcionário, a sua função no parque (por exemplo, técnico, operador, eletricista, etc.) e o turno em que trabalha.

O campo created_at tem como objetivo registar automaticamente a data e hora em que o registo foi criado na base de dados. Este atributo é útil para fins de organização e histórico, permitindo acompanhar quando cada funcionário foi adicionado ao sistema e manter um controle temporal sobre os dados inseridos.

```sql
CREATE TABLE Manutencao_Funcionario (
    ID_Funcionario INT NOT NULL,
    ID_Manutencao INT NOT NULL,
    PRIMARY KEY (ID_Funcionario, ID_Manutencao),
    FOREIGN KEY (ID_Funcionario) REFERENCES Funcionario(ID)
      ON UPDATE CASCADE
      ON DELETE CASCADE,
    FOREIGN KEY (ID_Manutencao) REFERENCES Manutencao(ID)
      ON UPDATE CASCADE
      ON DELETE CASCADE
);
```

Figura 5 – Tabela Manutencao_Funcionario em SQL

A tabela Manutencao_Funcionario foi criada para representar a relação muitos-para-muitos (N:M) entre funcionários e manutenções, refletindo a realidade do parque, onde uma única manutenção pode envolver vários funcionários e, por outro lado, um funcionário pode participar em várias manutenções ao longo do tempo.

Cada registo nesta tabela indica uma associação específica entre um funcionário e uma manutenção, funcionando como uma ligação entre as duas tabelas principais. As restrições definidas, nomeadamente o ON DELETE CASCADE, asseguram que, caso um funcionário ou uma manutenção seja removido da base de dados, todas as associações correspondentes são automaticamente eliminadas, mantendo a integridade referencial e evitando registos órfãos.

```sql
CREATE TABLE Visitante (
    ID INT PRIMARY KEY AUTO_INCREMENT,
    Nome VARCHAR(100),
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP
);
```

Figura 6 – Tabela Visitante em SQL

A tabela **Visitante** foi criada para armazenar informações sobre os clientes do parque, permitindo identificar cada visitante de forma única através do campo **ID**, que funciona como chave primária e é gerado automaticamente. Inicialmente, esta tabela contém apenas o nome do visitante, mas pode ser expandida no futuro para incluir outros atributos relevantes, como idade, contacto ou e-mail, de forma a permitir um melhor acompanhamento e personalização do serviço.

O campo **created_at** registra automaticamente a data e hora em que cada visitante é adicionado à base de dados, proporcionando um histórico temporal útil para análises e gestão de dados. Esta tabela serve como ponto de ligação para os bilhetes adquiridos, permitindo relacionar cada compra com o visitante correspondente, mantendo a integridade e rastreabilidade das transações dentro do sistema.

```sql
CREATE TABLE Bilhete (
    ID INT PRIMARY KEY AUTO_INCREMENT,
    Preco DECIMAL(6,2) NOT NULL,
    DataCompra DATE NOT NULL,
    AtracaoID INT NOT NULL,
    VisitanteID INT NOT NULL,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (AtracaoID) REFERENCES Atracao(ID)
      ON UPDATE CASCADE
      ON DELETE RESTRICT,
    FOREIGN KEY (VisitanteID) REFERENCES Visitante(ID)
      ON UPDATE CASCADE
      ON DELETE SET NULL
);
```

Figura 7 – Tabela Bilhete em SQL

A tabela Bilhete foi feita para registar todas as ações associadas à compra de bilhetes no parque de diversões. Cada bilhete é identificado de forma única através do campo ID, definido como chave primária e é gerado automaticamente. Esta tabela armazena informações essenciais como o preço do bilhete e a data da compra, permitindo o controle financeiro e temporal de cada transação.

Os campos AtracaoID e VisitanteID estabelecem ligações diretas com as tabelas Atração e Visitante, que se tornam chaves estrangeiras, cada bilhete está sempre associado a uma atração específica e a um visitante que o adquiriu.

As restrições definidas com ON UPDATE CASCADE fazem com que, caso os identificadores das atrações ou visitantes sejam atualizados, essas alterações sejam mudadas automaticamente na tabela de bilhetes. As opções ON DELETE RESTRICT (para atrações) e ON DELETE SET NULL (para visitantes) foram feitas para proteger os dados: impedindo a eliminação de uma atração que tenha bilhetes associados, e, caso um visitante seja removido, o bilhete mantém-se registado, mas o campo VisitanteID é definido como nulo, preservando o histórico da transação.

O campo created_at é usado para registar automaticamente a data e hora em que o bilhete foi inserido no sistema, contribuindo para o controle temporal e auditoria dos registos.

```sql
CREATE INDEX idx_bilhete_atracao ON Bilhete(AtracaoID);
CREATE INDEX idx_bilhete_visitante ON Bilhete(VisitanteID);
CREATE INDEX idx_manutencao_atracao ON Manutencao(AtracaoID);
```

Figura 8 – Índices em SQL

Foram criados índices nas tabelas para **aumentar a rapidez das pesquisas e consultas** dentro da base de dados. Estes índices ajudam o sistema a encontrar informações mais rapidamente, especialmente quando existem muitas ligações entre tabelas.

Foram definidos três índices principais:

- **idx_bilhete_atracao**: melhora a pesquisa de bilhetes por atração.
- **idx_bilhete_visitante**: acelera as consultas de bilhetes por visitante.
- **idx_manutencao_atracao**: otimiza a procura de manutenções associadas a uma atração.

Desta forma, a base de dados torna-se **mais rápida**, garantindo um melhor desempenho.

4. Ferramentas Utilizadas

Para a criação deste trabalho foram utilizadas algumas ferramentas indispensáveis como por exemplo o ERDPlus serviu para a criação dos diagramas que foram fundamentais para a formação da lógica, também foi usado a ferramenta MySQL para a implementação da lógica dos diagramas criados criando um código em SQL para a criação de uma base de dados.

Para a finalização do trabalho foram utilizadas essencialmente 2 ferramentas o canva para a criação do powerpoint e o google doc para a criação do relatório.

5. Conclusão

A realização deste projeto permitiu aplicar, de forma prática, os conhecimentos adquiridos na disciplina de Programação de Sistemas Informáticos (PSI), através do desenvolvimento de uma base de dados. Desde do pensamento lógico até à implementação em MySQL, foi possível compreender todo o processo envolvido na criação de um sistema de gestão estruturado e funcional.

A base de dados do parque de diversões foi desenhada para garantir a organização e integridade das informações, permitindo gerir de forma eficiente as atrações, visitantes, bilhetes, funcionários e manutenções. O processo de criação envolveu a definição das entidades, relações e restrições necessárias para assegurar a coerência e consistência dos dados.

Durante o desenvolvimento, foi possível reforçar conhecimentos essenciais como o raciocínio lógico, o planeamento de estruturas de dados, e a implementação de código SQL de forma estruturada e otimizada.

Em conclusão, este trabalho proporcionou uma experiência prática relevante e consolidou o entendimento sobre a importância de uma base de dados bem estruturada na gestão de sistemas informáticos, demonstrando como a programação e o planeamento caminham lado a lado na construção de soluções tecnológicas eficazes.