Universidade do Minho
Mestrado Integrado em Engenharia Informática
Licenciatura em Ciências da Computação

Unidade Curricular de Bases de Dados

Ano Lectivo de 2020/2021

Gestão de um cinema

Angélica Cunha (84398), Rui Chaves (83693), José Gomes (82418)

Dezembro, 2020

BD

Data de Recepção	
Responsável	
Avaliação	
Observações	

Gestão de um cinema

Angélica Cunha (84398), Rui Chaves (83693), José Gomes (82418)

Dezembro, 2020

Resumo

O presente trabalho foi desenvolvido no âmbito da UC de Bases de Dados, onde nos foi dada a liberdade de escolher um caso de estudo e desenvolver para este um SBD relacional.

Este relatório descreve uma proposta de implementação de um SGBD a ser implementado pelo *CineUM*. Começamos por fazer o levantamento de todos os requisitos e, após discussão e termo do trabalho de campo, passamos para a modelação conceptual e lógica do sistema a implementar. Tendo por base esta modelação, e, após a implementação em SQL, desenvolvemos um conjunto de *queries*, para tirar partido máximo do uso quotidiano do sistema.

Área de Aplicação: Planeamento e elaboração de um Sistema de Base de Dados
Palavras-Chave: MySQL, Bases de Dados Relacionais, Requisitos, Modelação, *Querie*.

Índice

Índice de Figuras

Índice de Tabelas

1. Definição do Sistema

O nosso sistema de Base de Dados faz a gestão dos dados de um cinema, o *CineUM.* Este sistema propõe-se a armazenar informação sobre os clientes que o frequentam, os seus funcionários e, está claro, os filmes que lá são rodados.

1.1. Contexto de Aplicação do sistema

Dada a situação que o país atravessa, com todas as restrições que nos são impostas de dia para dia, o que mais queremos é provar um pouco de normalidade. O cinema pode trazer-nos esse sentimento, portanto nada melhor que desfrutar de um bom filme e ajudar uma universidade pública com a compra do bilhete.

O *CineUM* preza por prestar um bom serviço aos clientes, sendo por essa razão que fará de tudo para recolher os dados dos seus espectadores, para posteriormente analisar cautelosamente e conseguir adequar este cinema às preferências do seu público. Caso veja que existe maior aderência a filmes de comédia, *CineUM* traz mais filmes de comédia, caso haja mais aderência a sessões noturnas tentaremos realizar mais sessões nesse horário, etc.

1.2. Fundamentação da implementação da base de dados

Concebemos desenvolver um SGBD por acharmos ser uma ferramenta necessária e essencial de apoio às necessidades do *CineUM*, tendo também em vista a resolução de eventuais problemas causados por uma incorreta e desatualizada forma de organização e preservação de dados. A Base de Dados a ser implementada irá servir de suporte diário a todas as atividades realizadas pelo *CineUM*, integrando e compilando informação num repositório único e centralizado. A utilização da Base de Dados irá prevenir redundância e ajudar na manutenção da consistência dos dados introduzidos por um possível funcionário do cinema. Pretende-se, por exemplo, que torne mais simples a gestão da lotação das salas de cinema e, consequentemente, com essa informação, como mencionado em cima, prever a aderência a uma determinada sessão e planear horários com mais adesão. O facto de a implementação ser desenvolvida de acordo com a sua utilização pelo *CineUM*, permite que seja adaptado às suas necessidades. Esta facilidade na gestão completa do cinema pelo sistema e partilha de dados

intrínseca à utilização de uma BD, releva-se uma mais valia e diferencia o *CineUM* dos outros cinemas.

2. Levantamento e Análise de Requisitos

Nesta etapa será abordado o método utilizado para o levantamento e análise dos vários requisitos levantados, e posteriormente será feita uma análise aos requisitos apresentados.

2.1 Método de levantamento e de análise de requisitos adotado

Depois de algum debate, chegamos à conclusão que o nosso sistema terá que, primeiro de tudo, ter informação sobre os seus clientes, tal como o seu nome, a sua data de nascimento de forma a saber a sua idade e o seu contacto telefónico para lhe poder mandar por SMS as novidades do *CineUM.* Achamos também imperativo saber mais informação acerca dos filmes que lá são exibidos, tais como o seu nome, a sua duração, qual o tipo de filme (Comédia, Ação, Aventura, ...) e a sua classificação no IMDB (escala de 0 a 10). Também achamos relevante guardar a capacidade de cada sala do cinema. Por fim, mas não menos importante, teremos também os funcionários, e guardaremos o seu nome, contacto telefónico, morada, data de nascimento.

Visto que sempre que um cliente vai assistir a um filme, precisa de se dirigir a um funcionário para efetuar a compra de um bilhete, surge então a necessidade de armazenar a informação relativa a este como a sua data de compra, a data do filme, o preço, o número da sala.

2.2 Requisitos levantados

De seguida serão apresentados os requisitos que foram no nosso debate e que deverão ser suportados pelo sistema de gestão de base de dados, tanto a nível da sua administração como da sua exploração.

2.2.1. Requisitos de descrição

D1- um cliente tem um nome, data de nascimento, um contacto SMS e um código postal;
D2- um funcionário tem um nome, data de nascimento, um contacto, uma morada e um código postal;
D3- um filme tem um nome, uma duração, uma classificação e um tipo;
D4- uma sala tem uma capacidade e um número;
D5- um bilhete tem data de compra, data do filme, preço e número da sala;

2.2.2. Requisitos de exploração

E1- Permitir a consulta dos dados de um cliente dado o seu id;
E2- Permitir a consulta dos números de bilhetes comprados por cada cliente;
E3- Permitir a consulta dos dados de um funcionário dado o seu id;
E4- Permitir a consulta de quantos bilhetes foram vendidos por cada funcionário;
E5- Permitir a consulta as informações de um filme dado o seu id;
E6- Permitir a consulta da capacidade de cada sala;
E7- Listar todos os funcionários que trabalham no cinema
E8- Listar quais os clientes que compraram o bilhete antes da hora do filme, qual o filme
E9- Permitir a consulta da média das idades dos clientes que assistiram a cada filme
E10- Permitir a consulta do número de clientes que assistiram a um filme por localidade
E11- Listar todos os clientes que assistiram a um determinado filme
E12- Permitir a consulta da faturação de um determinado ano

2.2.3. Requisitos de controlo

C1- Não podem ser vendidos mais bilhetes do que a capacidade da sala

2.3 Análise e validação geral dos requisitos

Após o levantamento de requisitos, foram necessárias algumas reuniões até que o conjunto fina de requisitos ficasse estabelecido e de acordo com todos. Ainda há margem para enriquecer a base de dados no futuro, porém nesta fase estamos satisfeitos com os requisitos já apresentados.

3 Modelação Conceptual

Neste capítulo vamos expor a abordagem de modelação conceptual, assim como identificar, caracterizar, relacionar entidades e relacionamentos. Por fim, vamos mostrar as associações dos relacionamentos com as entidades.

3.1 Apresentação da abordagem de modelação realizada

A abordagem de modelação feita envolve duas vistas de análise: o funcionário que pode consultar, inserir e manipular as informações dos nossos clientes e bilhetes; a UM que poderá consultar a faturação do cinema.

3.2 Identificação e caracterização das entidades

Com base na análise de requisitos apresentados anteriormente, identificou-se a necessidade de existência de 7 entidades:

Cliente : representa os espectadores que vão ao cinema;

Funcionário: representa os funcionários que trabalham no cinema;

Filme: representa os filmes que são exibidos no cinema;

Sala: representa as salas do cinema;

Código Postal: representa o código postal relativo à morada de um Cliente ou funcionário;

Bilhete: representa os bilhetes que são vendidos aos clientes pelos funcionários.

Apresentamos de seguida o dicionário de dados que representa as entidades com algumas informações das mesmas:

Nome da Entidade	Descrição	Ocorrências
Cliente	Termo geral que descreve o nosso cliente.	Cada cliente vai ao cinema e compra um bilhete com um funcionário. Cada cliente vê um filme numa sala.
Funcionário	Termo geral que descreve todos os funcionários que trabalham para o cinema.	Um funcionário atende um cliente de cada vez. Um funcionário dá um bilhete para um filme a um cliente.

Filme	Termo geral que descreve todos os filmes exibidos no cinema.	Cada cliente compra um bilhete para um filme. Um filme é exibido numa sala.
Sala	Termo geral que descreve todas as salas existentes no cinema.	Cada filme é exibido numa sala.
Código Postal	Termo geral que descreve o código postal da morada de um cliente ou funcionário.	Cada cliente ou funcionário possui um código postal referente à sua morada.
Bilhete	Termo geral que descreve todos os bilhetes que são vendidos.	Cada cliente compra um bilhete a um funcionário. Um bilhete tem um filme e uma sala associado.

Tabela 1 - Dicionário de Dados das Entidades

3.3 Identificação e caracterização dos relacionamentos

Tendo em conta todas as entidades e requisitos apresentados acima, apresentaremos agora os relacionamentos entre as mesmas. A nossa entidade central é o Bilhete que relaciona o funcionário com o cliente. Um cliente compra bilhetes a um funcionário e os funcionários cedem os bilhetes a um cliente.

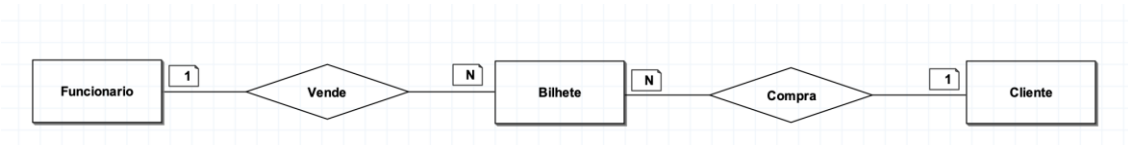

Figura 1 - Relacionamento Funcionário – Bilhete – Cliente

- **Relacionamento Funcionário – Bilhete – Cliente**

  Neste relacionamento, o Bilhete é gerado através da relação de N:M do Funcionário com o Cliente. Deste modo, temos que 1 Funcionário vende N bilhetes e um Cliente compra N bilhetes.

- **Relacionamento Bilhete – Filme**

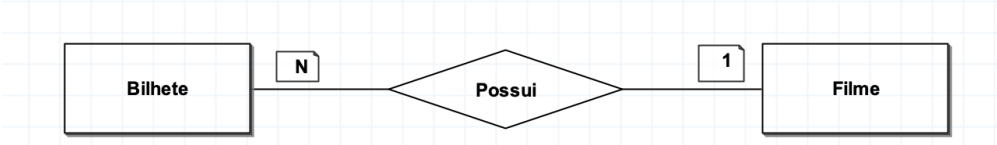

Figura 2 - Relacionamento Bilhete – Filme

Este relacionamento identifica o filme para o qual o bilhete foi vendido para. A multiplicidade apresentada deve-se ao facto de vários bilhetes poderem ser vendidos para o mesmo filme.

- **Relacionamento Bilhete – Sala**

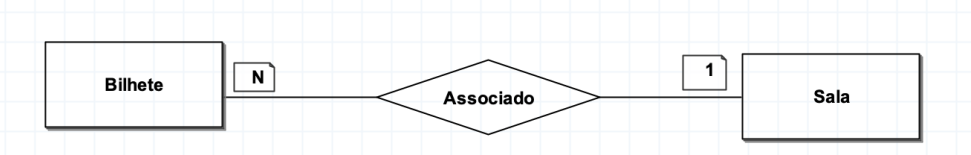

Figura 3 - Relacionamento Bilhete – Sala

Este relacionamento identifica a sala associada a cada bilhete. A multiplicidade apresentada corresponde ao facto de vários bilhetes poderem estar associados à mesma sala.

Entidade	Multiplicidade	Relação	Multiplicidade	Entidade
Funcionário	1	Vende	N	Bilhete
Bilhete	N	Possui	1	Filme
	N	Associado	1	Sala
Cliente	1	Compra	N	Bilhete

Tabela 2 - Dicionário de dados dos relacionamentos

3.4 Identificação e caracterização da associação dos atributos com as entidades e relacionamentos

Tendo em consideração os requisitos, as entidades e os relacionamentos estabelecidos anteriormente, determinamos agora os atributos referentes a cada uma das entidades.
Cada uma das entidades tem um atributo definido como a chave primária. Como esta tem de ser única criamos um ID para cada uma delas, sendo então o identificador único da entidade.

Entidade	Atributo	Domínio	Nulo	Composto	Multivalor	Derivado
Cliente	ID Cliente	INT	Não	Não	Não	Não
	Nome	VARCHAR(20)	Não	Não	Não	Não
	Data de Nascimento	DATE	Não	Não	Não	Não
	Contacto	INT(9)	Não	Não	Não	Não
	Código Postal	VARCHAR(8)	Não	Sim	Não	Não
Funcionário	ID Funcionário	INT	Não	Não	Não	Não
	Nome	VARCHAR(20)	Não	Não	Não	Não
	Contacto	INT(9)	Não	Não	Não	Não

	Morada	VARCHAR(45)	Não	Não	Não	Não
	Código Postal	VARCHAR(8)	Não	Sim	Sim	Não
	Data de Nascimento	DATE	Não	Não	Não	Não
Filme	ID Filme	INT	Não	Não	Não	Não
	Duração	TIME	Não	Não	Não	Não
	Classificação	INT(2)	Não	Não	Não	Não
	Tipo	VARCHAR(10)	Não	Não	Não	Não
	Nome	VARCHAR(45)	Não	Não	Não	Não
Código-Postal	Código Postal	VARCHAR(8)	Não	Não	Não	Não
	Localidade	VARCHAR(45)	Não	Não	Não	Não
Sala	Número	INT	Não	Não	Não	Não
	Capacidade	INT	Não	Não	Não	Não
Bilhete	ID Filme	INT	Não	Sim	Não	Não
	ID Cliente	INT	Não	Sim	Não	Não
	Data Compra	DATETME	Não	Não	Não	Não
	Data Filme	DATETIME	Não	Não	Não	Não
	Preço	FLOAT	Não	Não	Não	Não
	Número Sala	INT	Não	Sim	Não	Não
	ID Funcionário	INT	Não	Sim	Não	Não

Tabela 3 - Dicionário de dados dos atributos das entidades

3.5 Detalhe ou generalização de entidades

Análise detalhada das entidades:

- Cliente– A pessoa que compra os bilhetes para assistir a um filme. É composto pelos seguintes atributos: Identificador de Cliente; Nome; Data de Nascimento; Contacto Telefónico; Código Postal.
- Funcionário– A pessoa que vende os bilhetes aos clientes. É composto pelos seguintes atributos: Identificador de Funcionário; Nome; Contacto Telefónico; Morada; Código Postal; Data de Nascimento.
- Filme– Produto audiovisual exibido na sala de cinema. É composto pelos seguintes atributos: Identificador do Filme; Duração; Classificação (0 a 10); Tipo (Género); Nome;
- Código– Postal. Conjunto de números indicadores de um local. É composto pelos seguintes atributos: Código Postal; Localidade.
- Sala– Sala de Cinema. É composto pelos seguintes atributos: Número (Identificador da Sala); Capacidade.
- Bilhete – Papel com a informação necessária para a visualização do filme. É composto pelos seguintes atributos: Identificador do Filme; Identificador do Cliente; Data da compra; Data do filme; Preço; Número da Sala; Identificador do Funcionário.

3.6 Apresentação e explicação do diagrama ER

Tendo em conta as entidades, relacionamentos e atributos acima apresentados, e como forma de apresentação, criou-se o diagrama conceptual apresentado de seguida.

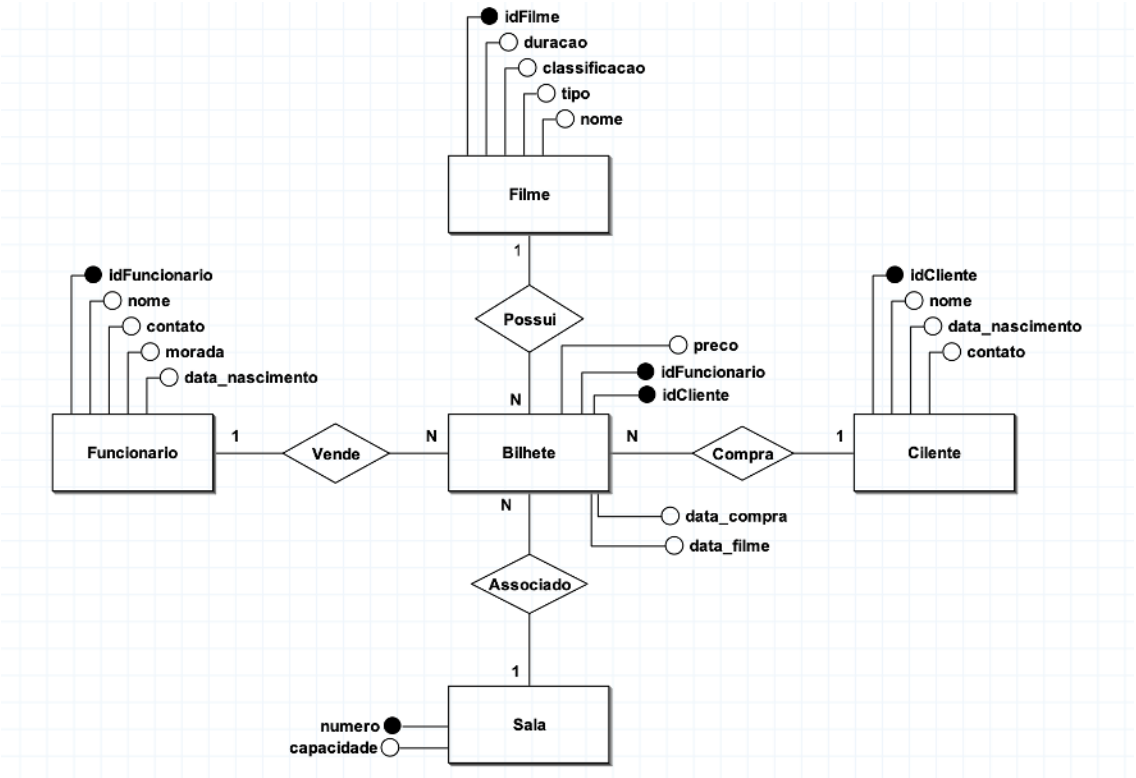

Figura 4 - Modelo Conceptual

3.7 Validação do modelo de dados produzido

Após verificar se existiam redundâncias nas relações entre entidades e nos atributos. Concluímos que tal não existe, e o modelo de dados produzido encontra-se válido.

4 Modelação Lógica

Iremos agora descrever a construção do modelo lógico, apresentando o desenho do mesmo com recurso ao programa *MySQL Workbench.*

4.1 Construção e validação do modelo de dados lógico

Através da aplicação de variados passos, elaboramos o modelo lógico a partir do conceptual. De acordo com esses passos, criamos uma tabela para cada entidade, e representamos os relacionamentos entre elas.

De seguida iremos enumerar as nossas relações, representando as chaves primárias a **negrito** e as chaves estrangeiras sublinhadas. Para as chaves que são simultaneamente primárias e estrangeiras vamos usar **ambos**.

- Cliente: **idCliente** , nome, data_nascimento, contacto, cod_postal.
- Funcionario: **idFuncionario** , nome, data_nascimento, contacto, cod_postal, morada.
- Bilhete: **idCliente** , **idFuncionario ,** idFilme, numSala, data_compra, data_filme, preço.
- Filme: **idFilme,** duracao, classificacao, tipo, nome.
- Sala: **numero**, capacidade.
- Codigo_Postal: **codigo_postal**, localidade.

4.2 Desenho do modelo lógico

Com recurso ao *MySQL Workbench* obtivemos o seguinte modelo lógico.

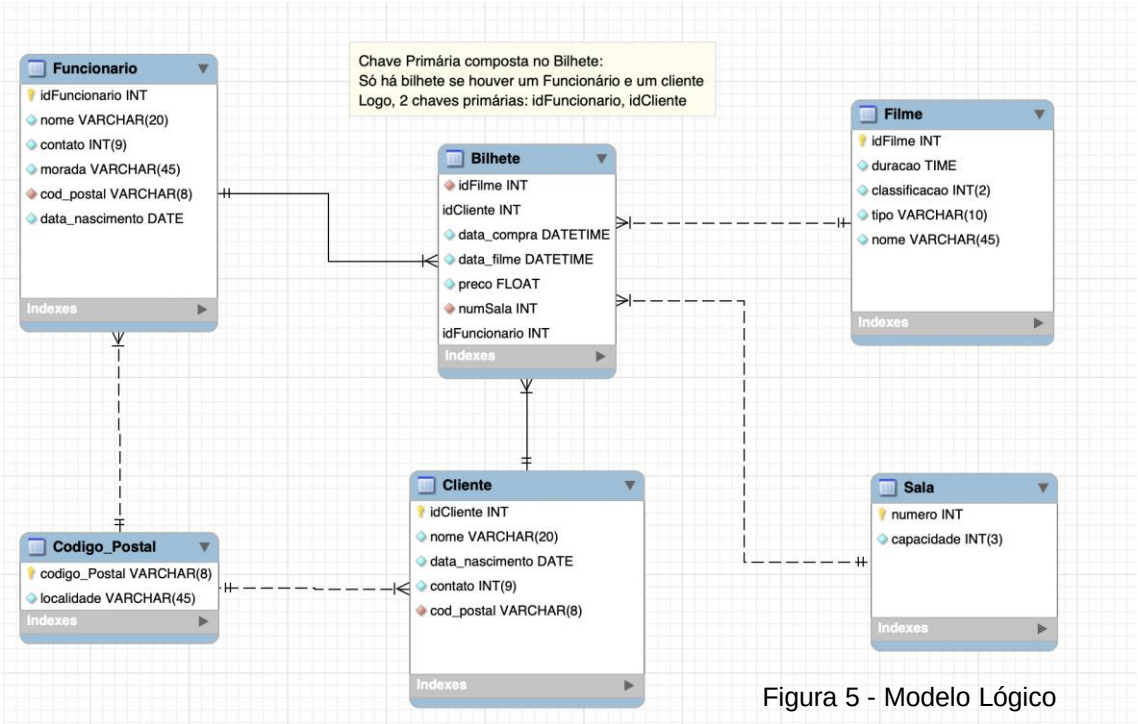

Figura 5 - Modelo Lógico

4.3 Validação do modelo através da normalização

- **Primeira Forma Normal**

Segundo a definição presente em *Connoly* (*Connoly*, 2015), se uma relação na qual a inserção de cada linha e coluna contém um e um só elemento, está na primeira forma normal.
O nosso modelo lógico obedece a esta regra logo nada temos a alterar.

- **Segunda Forma Normal**

Todos os atributos de uma tabela que não são chaves primárias têm de ser totalmente e funcionalmente dependentes da chave primária da mesma (*Connoly*, 2015).
O nosso modelo lógico cumpre esta regra, pois não existem dependências funcionais parciais.

- **Terceira Forma Normal**

Uma relação que está na primeira e segunda forma normal e em que todos os atributos, que não são chaves primárias, são transitivamente dependentes da chave primária está na terceira forma normal (Connoly, 2015).

O nosso modelo lógico cumpre a terceira forma normal e para se fazer esta verificação analisamos as dependências funcionais de cada relação do modelo lógico com o objetivo de se encontrar dependências transitivas. As dependências funcionais dessas relações são:

Cliente: ID → Nome, Data nascimento, Contacto, Código postal.
Funcionario: ID → Nome, Data nascimento , Contacto, Código postal, Morada.
Bilhete: (ID Funcionario, ID Cliente) → ID Filme, Número Sala, Data de compra, Data de filme, Preço.
Filme: ID → Duração, Classificação, Tipo, Nome.
Sala: Número → Capacidade.
Código postal: Código postal → Localidade

Concluímos assim que todos os atributos, à exceção daqueles que são chave primária, dependem apenas da respetiva chave primária da relação e não de outros atributos.

4.4 Validação do modelo com interrogações do utilizador

Neste tópico iremos apresentar as soluções dos requisitos de exploração apresentados anteriormente.

1 – Mostrar os Tipos de Filme

Π_{Tipo} (Filme)

2 – Mostra todos os dados dos funcionários

$\Pi_{IDFuncionario, Nome, Contacto, Morada, Cod\_Postal, Data\_Nascimento}$ (Funcionario)

3 – Mostrar todos os clientes que tem código postal 4711-909

Π_{Nome} ($\sigma_{(Cod\_Postal=\text{"4711-909"})}$(Cliente))

4 –Mostrar quais os clientes que compraram o bilhete na hora do filme

Π_{Nome} ($\sigma_{(Data\_compra=Data\_filme)}$ (Bilhete $\bowtie$ Cliente))

5-Mostrar quais os clientes que assistiram a um filme de Aventura

Π_{Nome} ($\sigma_{(Tipo=\text{"Aventura"})}$ (Filme $\bowtie$ Cliente))

4.5 Revisão do modelo lógico produzido

Ao terminarmos a elaboração do modelo lógico, reunimos novamente para tirar os pareceres finais sobre a precisão do mesmo.

Verificamos assim que todos os requisitos tinham sido cumpridos e que tínhamos atributos suficientes para cada entidade armazenar a informação pretendida.

5 Implementação Física

Nesta secção iremos abordar a tradução das questões do utilizador para SQL.

5.1 Tradução das interrogações do utilizador para SQL

Primeiramente, traduzimos as interrogações para SQL.

```sql
-- Query 1
-- Consultar os dados de um cliente dado o seu id
SELECT * FROM CLIENTE as C
WHERE c.idCliente = 1;
```

Figura 6 - *Query* 1: Consultar os dados de um cliente dado o seu id

```sql
-- Query 2
-- Consultar os número de bilhetes comprados por cada cliente
SELECT c.nome, count(b.idFilme) as BilhetesComprados  FROM CLIENTE as C
JOIN BILHETE AS B where c.idCliente=b.idCliente
group by c.nome;
```

Figura 7 - *Query* 2: Consultar os números de bilhetes comprados por cada cliente

```sql
-- Query 3
-- Consultar os dados de um funcionário dado o seu id (Neste caso id=1)
SELECT * FROM FUNCIONARIO as F
WHERE f.idFuncionario = 1;
```

Figura 8 - *Query* 3: Consultar os dados de um funcionário dado o seu id

```sql
-- Query 4
-- Consultar quantos bilhetes foram vendidos por cada funcionário
SELECT F.NOME, count(B.idFilme) as BilhetesVendidos FROM FUNCIONARIO AS F
JOIN BILHETE AS B where f.idFuncionario=b.idFuncionario
group by f.nome;
```

Figura 9 - *Query* 4: Consultar quantos bilhetes foram vendidos por cada funcionário

```sql
-- Query 5
-- Consultar as informações de um filme dado o seu id (Neste caso id=1)
SELECT * FROM FILME as F
WHERE f.idFilme = 1;
```

Figura 10 - *Query* 5: Consultar as informações de um filme dado o seu id

```sql
-- query 6
-- Permitir a ver a capacidade de cada sala
SELECT s.numero, s.capacidade FROM SALA AS S;
```

Figura 11 - *Query* 6: Permitir ver a capacidade de cada sala

```sql
-- query 7
-- Listar todos os funcionários que trabalham no cinema
SELECT * FROM Funcionario;
```

Figura 12 - *Query* 7: Listar todos os funcionários que trabalham no cinema

```sql
-- query 8
-- Listar quais os clientes que compraram o bilhete antes da hora do filme, qual o filme
-- e a sua hora de transmissão e de compra
select c.nome, f.nome, time(b.data_compra) as hora_compra, time(b.data_filme) as hora_filme from Cliente as c
join Bilhete as b on c.idCliente = b.idCliente
join Filme as f on b.idFilme = f.idFilme
where timediff(b.data_compra, b.data_filme) < 0;
```

Figura 13 - *Query* 8: Listar quais os clientes que compraram o bilhete antes da hora do filme, qual o filme e a sua hora de transmissão e de compra

```sql
-- query 9
-- Consultar a média das idades dos clientes que assistiram a cada filme
/*Function que calcula a idade dada a data de nascimento*/
DELIMITER //
CREATE FUNCTION `idade`(dta date) RETURNS int
BEGIN
RETURN timestampdiff(YEAR, dta, CURDATE());
END //
DELIMITER ;

SELECT f.nome, AVG(idade(c.data_nascimento)) as media_idades FROM Filme f
JOIN Bilhete as b on f.idFilme = b.idFilme
JOIN Cliente as c on c.idCliente = b.idCliente
GROUP BY f.nome;
```

Figura 14 - *Query* 9: Consultar a média de idades dos clientes que assistiram a cada filme

```sql
-- query 10
-- Consultar o número de clientes que assistiram a um filme(id) por localidade
SELECT cp.localidade, count(distinct c.idCliente) as nrClientes, f.nome FROM Cliente c
JOIN Codigo_Postal as cp on cp.cod_postal = c.cod_postal
JOIN Bilhete as b on c.idCliente = b.idCliente
JOIN Filme as f on f.idFilme = b.idFilme
where f.idFilme = 2
GROUP BY cp.localidade;
```

Figura 15 - *Query* 10: Consultar o número de clientes que assistiram a um filme(id) por localidade

```sql
-- query 11
-- Listar todos os clientes que assistiram a um determinado filme
DELIMITER //
CREATE PROCEDURE listarClientesFilme(IN filme varchar(45), INOUT result_list varchar(2000))
BEGIN

    DECLARE c_finished integer default 0;
    DECLARE c_nome_cliente varchar(45);

    DECLARE clientes_cursor CURSOR FOR
    SELECT c.nome
        FROM Cliente c, Bilhete b, Filme f WHERE c.idCliente= b.idCliente and b.idFilme = f.idFilme and f.nome = filme;


    DECLARE CONTINUE HANDLER FOR NOT FOUND SET c_finished = 1;

    OPEN clientes_cursor;
    FETCH NEXT FROM clientes_cursor into c_nome_cliente;

    get_cliente : LOOP
        FETCH clientes_cursor INTO c_nome_cliente;
        IF c_finished = 1 THEN
            LEAVE get_cliente;
        END IF;
        SET result_list = CONCAT(c_nome_cliente,"   \n", result_list);
    END LOOP;
    CLOSE clientes_cursor;
END//
DELIMITER ;

SET @res = "";
CALL listarClientesFilme('Maléfica', @res);
SELECT @res ;
```

Figura 16 - *Query* 11: Listar todos os clientes que assistiram a um determinado filme

```sql
-- Query 12
-- Consultar faturação de um determinado ano
DELIMITER //
CREATE PROCEDURE `faturacaoDeUmAno` (IN ano INT)
BEGIN
SELECT SUM(B.PRECO) FROM BILHETE AS B
WHERE YEAR(B.DATA_COMPRA) = ano AND DATEDIFF(CURDATE(), B.DATA_COMPRA) > 0;
END //
DELIMITER ;

CALL faturacaoDeUmAno(2010);
```

Figura 17 - *Query* 12: Consultar faturação de um determinado ano

5.2 Definição e caracterização das vistas de utilização em SQL

```sql
-- Criação da vista "vwlistaFuncionarios"
CREATE VIEW vwlistaFuncionarios AS
SELECT IDFuncionario AS Nr, Nome  , Data_Nascimento AS 'Data de Nascimento',
 Contacto, Morada, Cod_Postal AS 'Código Postal'
FROM Funcionario AS F;

SELECT * FROM vwlistaFuncionarios;
```

Figura 18 - *View*: Lista de Funcionários

```sql
-- Criação da vista "vwClientesbyLocalidade"
CREATE VIEW vwlistaClientesbyLocalidade AS
SELECT c.idcliente AS Nr, c.nome, cp.Localidade FROM Cliente c
JOIN Codigo_Postal AS cp ON  cp.cod_postal = c.cod_postal
order by cp.localidade desc;
```

Figura 19 - *View*: Clientes por localidade

```sql
-- Criação da vista "vwlistaClientes"
CREATE VIEW vwlistaClientes AS
SELECT IDCliente AS Nr, Nome , Data_Nascimento AS 'Data de Nascimento',
 Contacto, Cod_Postal AS 'Código Postal'
 FROM Cliente AS C;
```

Figura 20 - *View*: Lista de Clientes

```sql
-- Criação da vista "vwFilmesbyTipo"
CREATE VIEW vwFilmesbyTipo AS
SELECT f.nome AS Nome, f.tipo AS Tipo FROM Filme f
order by f.tipo ;
```

Figura 21 - *View*: Lista de Filmes por tipo

```sql
-- Criação da vista "vwlistaFilmes"
CREATE VIEW vwlistaFilmes AS
SELECT IDFilme AS Nr, Nome, tipo, duracao, classificacao
FROM Filme AS F;
```

Figura 22 - *View*: Lista de Filmes

```sql
-- Criação da vista "vwFuncionariosbyLocalidade"
CREATE VIEW vwlistaFuncionariosbyLocalidade AS
SELECT f.idfuncionario AS Nr, f.nome, morada,cp.Localidade, Data_Nascimento AS 'Data de Nascimento',Contacto FROM Funcionario f
JOIN Codigo_Postal AS cp ON  cp.cod_postal = f.cod_postal
order by cp.localidade desc;
```

Figura 23 - *View*: Lista de Funcionários por Localidade

```sql
-- Criação da vista "vwFilmesbyClassificação"
CREATE VIEW vwFilmesbyClassificação AS
SELECT f.nome AS Nome, f.classificacao AS Classificação FROM Filme f
order by f.classificacao desc;
```

Figura 24 - *View*: Lista de Filmes por Classificação

```sql
-- Criação da vista "vwFilmesbyDuracao"
CREATE VIEW vwFilmesbyDuracao AS
SELECT f.nome AS Nome, f.duracao AS Duração FROM Filme f
order by f.duracao asc ;
```

Figura 25 - *View*: Lista de FIlmes por Duração

```sql
-- Criação da vista "vwFilmesbySala"
CREATE VIEW vwFilmesbySala AS
SELECT distinct(f.nome) AS 'Nome do filme', s.numero AS 'Nº Sala' FROM Filme f
JOIN Bilhete AS b ON b.idfilme=f.idfilme
JOIN Sala AS s ON s.numero = b.num_sala
order by s.numero ;
```

Figura 26 - *View*: Lista de Filmes por Sala

5.3 Revisão do sistema implementado

Após o término da implementação do modelo físico, reunimos novamente para retificar possíveis incoerências e assegurar que todos os objetivos que nos propusemos a cumprir foram, de facto, cumpridos.

Concordamos todos que o sistema estava pronto para ser lançado.

6 Conclusões e Trabalho Futuro

Iniciamos o projeto fazendo um levantamento dos requisitos e a sua aplicação na base de dados prosseguido pela validação dos mesmos através de testes. A execução de testes que validam a implementação dos requisitos permite-nos ser consistentes, assegurando que não existem redundâncias que colocariam um problema significativo caso fossem deixados até às fases finais do projeto.

Em conclusão, consideramos ter correspondido com as expetativas do *CineUM*, implementando um sistema que ajuda o cinema nas diversas áreas que exigem gerir dados. Na nossa opinião, o cinema tem agora uma ferramenta que lhe irá permitir crescer e angariar clientes, com filmes, horários e salas de acordo com as preferências e gostos dos mesmos.

Quanto ao trabalho propriamente dito, pensamos que, apesar de não estar perfeito, atingimos aquilo a que nos propusemos inicialmente. No entanto, há certamente margem para melhoramentos. O que será exposto não representa assim algo que falhou na nossa base de dados, mas sim um ponto passível de melhorias a fazer.

Ao longo do desenvolvimento deste sistema para o *CineUM*, poderíamos, por exemplo, ter adicionado um atributo IBAN à Entidade Funcionário ou até um ID representativo de um determinado turno de trabalho. Ao centralizar esta última informação no sistema, seria possível e até relativamente simples fazer um horário e saber quantos funcionários estariam ao serviço num determinado turno, se por acaso algum estava a faltar, e se sim, tentar arranjar um substituto de outro turno. Estes são apenas exemplos, que, por não terem sido requisitados, não foram implementados e que representam melhorias a fazer, não só no nosso projeto, mas também no *CineUM* em si.

Referências

- Connolly, T. M. and Begg, C. E., Database Systems: A Practical Approach to Design, Implementation, and Management, 6th Edition, England: Pearson Education Limited, 2005

- Conteúdo disponibilizado pela equipa docente da Unidade curricular de Base de Dados 2020/2021

- SlideShare, Base de Dados, https://pt.slideshare.net/arturafonsosousa/bases-de-dados

Lista de Siglas e Acrónimos

Siglas e Acrónimos utilizados durante a realização do trabalho:

BD	Base de Dados
SGBD	Sistema de Gestão de Base de Dados
SQL	*Structure Query Language*
ID / Id / id	Identidade

Anexos

I. Povoamento das Tabelas SQL

```sql
-- Criação da Tabela Codigo_Postal
INSERT INTO `projeto`.`Codigo_Postal` VALUES
('4700-481','Palmeira, Braga'),
('4715-045','São Victor, Braga'),
('4711-909','Maximinos, Braga'),
('4705-474','Esporões, Braga'),
('4715-445','São Pedro(Este), Braga'),
('4700-912','São José de São Lázaro, Braga');
```

```sql
-- Criação da Tabela Funcionario
INSERT INTO `projeto`.`Funcionario` VALUES
(1,'Ana César',912348944,'Rua dos Quintais 34','4715-045','1999-10-14'),
(2,'Angélica Cunha',914445678,'Rua do Real 163','4705-474','1967-03-02'),
(3,'André Morais',935467556,'Rua dos Duques 80','4705-474','1980-12-20');
```

```sql
-- Criação da Tabela Filme
INSERT INTO `projeto`.`Filme` VALUES
(1, '01:28:00' ,'9', 'Animacao','Rei Leão' ),
(2, '01:37:00' ,'7', 'Ação'    ,'Maléfica'),
(3, '01:59:00' ,'8', 'Drama'   ,'1917'),
(4, '01:52:00' ,'6', 'Aventura','Crónicas de Natal: Parte Dois'),
(5, '01:36:00' ,'5', 'Comédia'    ,'A Princesa Volta a Ser Plebeia');
```

```sql
-- Criação da Tabela Sala
INSERT INTO `projeto`.`Sala` VALUES
(1, 50),
(2, 30),
(3, 40);
```

```sql
-- Criação da Tabela Bilhete
-- Filme 1  sala 1
INSERT INTO `projeto`.`Bilhete` VALUES
(1,1,'2010-02-01 22:00:00','2010-02-01 22:00:00',3.95 , 1 ,1 ),
(1,2,'2010-02-01 15:00:00','2010-02-01 22:00:00',5 , 1 ,1 ),
(1,6,'2010-01-28 13:00:00','2010-02-01 22:00:00',5 , 1 ,2 ),
(1,7,'2010-02-01 22:00:00','2010-02-01 22:00:00',5 , 1 ,3 ),
(1,9,'2010-02-01 22:00:00','2010-02-01 22:00:00',3.95 , 1 ,1 ),
(1,8,'2010-02-01 22:00:00','2010-02-01 22:00:00',5 , 1 ,2 ),
(1,10,'2010-01-29 09:00:00','2010-02-01 22:00:00',3.95 , 1 ,1 ),
(1,21,'2010-02-01 22:00:00','2010-02-01 22:00:00',5 , 1 ,2 ),
-- Filme 2 sala 2
(2,4,'2010-02-01 22:00:00','2010-02-01 22:00:00',3.95 , 2 ,1 ),
(2,6,'2010-02-01 15:00:00','2010-02-01 22:00:00',5, 2 ,1 ),
(2,7,'2010-02-01 15:00:00','2010-02-01 22:00:00',3.95 , 2 ,3 ),
(2,13,'2010-02-01 22:00:00','2010-02-01 22:00:00',3.95 , 2 ,2 ),
(2,14,'2010-01-28 13:00:00','2010-02-01 22:00:00',3.95 , 2 ,2 ),
(2,17,'2010-01-28 13:00:00','2010-02-01 22:00:00',5 , 2 ,2 ),
(2,18,'2010-01-28 13:00:00','2010-02-01 22:00:00',5 , 2 ,3 ),
(2,19,'2010-01-28 13:00:00','2010-02-01 22:00:00',3.95 , 2 ,2 ),
(2,20,'2010-01-29 09:00:00','2010-02-01 22:00:00',5 , 2 ,1 ),
(2,21,'2010-02-01 22:00:00','2010-02-01 22:00:00',3.95 , 2 ,1 ),
-- Filme 3 sala 1
(3,4,'2010-02-01 22:00:00','2010-02-01 22:00:00',3.95 , 1 ,1 ),
(3,6,'2010-02-01 15:00:00','2010-02-01 22:00:00',3.95 , 1 ,2 ),
(3,7,'2010-02-01 15:00:00','2010-02-01 22:00:00',5 , 1 ,3  ),
(3,10,'2010-02-01 22:00:00','2010-02-01 22:00:00',3.95 , 1 ,3 ),
(3,11,'2010-01-28 13:00:00','2010-02-01 22:00:00',5 , 1 ,2 ),
(3,12,'2010-01-28 13:00:00','2010-02-01 22:00:00',5 , 1 ,2 ),
(3,13,'2010-01-28 13:00:00','2010-02-01 22:00:00',5 , 1 ,3 ),
(3,14,'2010-01-28 13:00:00','2010-02-01 22:00:00',3.95 , 1 ,2 ),
(3,15,'2010-02-01 22:00:00','2010-02-01 22:00:00',3.95 , 1 ,3 ),
(3,16,'2010-02-01 22:00:00','2010-02-01 22:00:00',3.95 , 1 ,3 ),
(3,17,'2010-02-01 22:00:00','2010-02-01 22:00:00',5 , 1 ,3 ),
(3,18,'2010-02-01 22:00:00','2010-02-01 22:00:00',5 , 1 ,3 ),
(3,20,'2010-01-29 09:00:00','2010-02-01 22:00:00',3.95 , 1 ,1 ),
(3,21,'2010-02-01 22:00:00','2010-02-01 22:00:00',3.95 , 1 ,1 ),
```

```
-- Filme 4 sala 3
(4,1,'2010-02-01 22:00:00','2010-02-01 22:00:00',3.95 , 3 ,1 ),
(4,2,'2010-02-01 15:00:00','2010-02-01 22:00:00',5    , 3 ,1 ),
(4,3,'2010-01-28 13:00:00','2010-02-01 22:00:00',5    , 3 ,2 ),
(4,4,'2010-02-01 22:00:00','2010-02-01 22:00:00',5    , 3 ,2 ),
(4,5,'2010-02-01 22:00:00','2010-02-01 22:00:00',3.95 , 3 ,1 ),
(4,6,'2010-02-01 22:00:00','2010-02-01 22:00:00',5    , 3 ,2 ),
(4,7,'2010-01-29 09:00:00','2010-02-01 22:00:00',3.95 , 3 ,2 ),
(4,8,'2010-01-29 09:00:00','2010-02-01 22:00:00',3.95 , 3 ,2 ),
(4,9,'2010-01-29 09:00:00','2010-02-01 22:00:00',3.95 , 3 ,1 ),
(4,10,'2010-01-29 09:00:00','2010-02-01 22:00:00',3.95 , 3 ,1 ),
(4,11,'2010-01-29 09:00:00','2010-02-01 22:00:00',3.95 , 3 ,1 ),
(4,19,'2010-01-29 09:00:00','2010-02-01 22:00:00',3.95 , 3 ,1 ),
(4,20,'2010-01-29 09:00:00','2010-02-01 22:00:00',3.95 , 3 ,1 ),
(4,21,'2010-02-01 22:00:00','2010-02-01 22:00:00',5    , 3 ,2 ),
-- Filme 5 sala 3
(5,20,'2010-02-01 22:00:00','2010-02-01 22:00:00',3.95 , 3 ,1 ),
(5,18,'2010-02-01 15:00:00','2010-02-01 22:00:00',3.95 , 3 ,2 ),
(5,16,'2010-02-01 15:00:00','2010-02-01 22:00:00',5 , 3 ,3  ),
(5,14,'2010-02-01 22:00:00','2010-02-01 22:00:00',3.95 , 3 ,3 ),
(5,12,'2010-01-28 13:00:00','2010-02-01 22:00:00',5 , 3 ,2 ),
(5,10,'2010-01-28 13:00:00','2010-02-01 22:00:00',5 , 3 ,2 ),
(5,8,'2010-01-28 13:00:00','2010-02-01 22:00:00',5 , 3 ,3 ),
(5,6,'2010-01-28 13:00:00','2010-02-01 22:00:00',3.95 , 3 ,2 ),
(5,4,'2010-02-01 22:00:00','2010-02-01 22:00:00',3.95 , 3 ,3 ),
(5,2,'2010-02-01 22:00:00','2010-02-01 22:00:00',3.95 , 3 ,3 ),
(5,1,'2010-02-01 22:00:00','2010-02-01 22:00:00',5 , 3 ,3 );
```

```sql
-- Criação da Tabela Cliente
INSERT INTO `projeto`.`Cliente` VALUES
(1, 'José Gonçalves','1999-03-17',935441726,'4700-912'),
(2, 'João Carvalho', '2001-11-13' , 919867832 , '4715-045'),
(3, 'Manuel Oliveira', '1998-08-05' , 915690567 , '4715-045'),
(4, 'Manuel Oliveira', '2003-10-14' , 916539201,'4715-045'),
(5, 'Marta Esteves', '1990-02-10' , 937820182,'4700-481'),
(6, 'Carolina Maia', '2000-05-06' , 919028374 ,'4700-481'),
(7, 'Manuel Ferreira', '1995-11-02' , 939026789,'4715-045'),
(8, 'Luís Pereira', '1998-06-04' , 924389001,'4715-045'),
(9, 'Miguel Vieira', '2000-05-02' , 910054882,'4700-481'),
(10, 'Ana Rocha', '1995-06-01' , 928847703,'4715-045'),
(11, 'Manuel Figueiredo', '1999-09-05' , 916684002,'4715-045'),
(12, 'Alexandre Miranda', '1999-12-03' , 936689012, '4711-909'),
(13, 'Bruno Lobo', '1992-03-14' , 911234567,'4711-909'),
(14, 'Mafalda Oliveira', '2001-10-05' , 967745308,'4715-045'),
(15, 'Nuno Ribeiro', '1996-03-07' , 967891234,'4711-909'),
(16, 'João Macedo', '1993-08-05' , 915792730,'4705-474'),
(17, 'Filipe Silva', '2000-05-02' ,914567838,'4715-045'),
(18, 'Otávio Andrade', '1995-06-02' , 930009897,'4705-474'),
(19, 'Tiago Rodrigues', '1998-04-04' , 925577339,'4715-445'),
(20, 'André Vieira', '2000-01-02' , 910054882,'4715-445'),
(21, 'Vítor Abreu', '1995-01-01' , 928847709,'4715-445');
```