



Universidade do Minho

Mestrado Integrado em Engenharia Informática
Licenciatura em Ciências da Computação

Unidade Curricular de Bases de Dados

Ano Lectivo de 2017/2018

Sétima Arte: gestão de sessões de cinema

Lúcia Abreu

Novembro, 2017

BD

Data de Recepção	
Responsável	
Avaliação	
Observações	

Sétima Arte: gestão de sessões de cinema

Lúcia Abreu

Novembro, 2017

Resumo

Este projeto consiste no desenvolvimento e implementação da base de dados para a reserva de bilhetes de sessões de cinema, pedida pela empresa **Braga Cinema Center** (BCC). Para tal, seguiu-se a metodologia apresentada e discutida no livro *Connolly, T., Begg, C., Database Systems, A Practical Approach to Design, Implementation, and Management, Addison-Wesley, 4ª Edição, 2004*. Abordando-se metodicamente todas as etapas da metodologia: Modelo Conceptual, Modelo Lógico e Modelo Físico.

A etapa que necessitou de uma fase de maturação, bem como de validação, mais extensa foi o desenvolvimento do modelo conceptual, dado que foi necessário assegurar que o modelo permite responder a todos os requisitos requeridos. Como o modelo conceptual é a base das restantes etapas foi fundamental garantir que este representa o problema em causa. As etapas seguintes, visto que derivam de uma boa conceção do modelo conceptual, decorreram de forma mais rápida e direta.

A base de dados foi implementada em MySQL uma vez que é um SGBD *open-source* com bastante reconhecimento e suporte da comunidade. A base de dados implementada permite a validação dos dados e das restrições gerais identificadas nos requisitos. Desta forma garante-se consistência de dados e regras de negócio.

Área de Aplicação: Desenvolvimento e implementação de Sistemas de Bases de Dados.

Palavras-Chave: Modelo Conceptual, Modelo Lógico, Modelo Físico.



Índice

1. Introdução	1
1.1. Contextualização	1
1.2. Apresentação do Caso de Estudo	2
1.3. Motivação e Objetivos	3
1.3.1 Motivação	3
1.3.2 Objetivos	4
1.4. Estrutura do Relatório	4
2. Viabilidade do projeto	5
2.1. Análise e Justificação da viabilidade do projeto	5
2.1.1 Facilitar a vida ao cliente	5
2.1.2 A que preço (para cumprir os objetivos)?	5
2.2. Vantagens e ganhos operacionais	5
3. Levantamento de requisitos	7
3.1. Análise do domínio	7
3.2. Identificação das fontes dos requisitos	8
3.3. Análise dos stakeholders	8
3.4. Seleção das técnicas, abordagens e ferramentas a utilizar	9
3.5. Identificação dos perfis de utilização e requisitos	9
4. Modelo Conceptual	11
4.1. Identificação das Entidades	11
4.1.1 Cliente	11
4.1.2 Bilhete	11
4.1.3 Sessão	11
4.1.4 Filme	11
4.1.5 Sala	12
4.1.6 Horário	12
4.2. Identificação dos Relacionamentos	12
4.2.1 Relacionamento Cliente – Bilhete	12
4.2.2 Relacionamento Bilhete – Sessão	13
4.2.3 Relacionamento Sessão – Filme	13
4.2.4 Relacionamento Sala - Filme	13

4.2.5 Relacionamento Horário - Sessão	14
4.3. Identificação dos Atributos	14
4.3.1 Atributos da Entidade Cliente	14
4.3.2 Atributos da Entidade Bilhete	15
4.3.3 Atributos da Entidade Sessão	16
4.3.4 Atributos da Entidade Filme	16
4.3.5 Atributos da Entidade Sala	16
4.3.6 Atributos da Entidade Horário	17
4.4. Determinar o domínio dos atributos	17
4.5. Determinação chaves candidatas, primárias e alternativas	18
4.5.1 Cliente	18
4.5.2 Bilhete	18
4.5.3 Sessão	18
4.5.4 Filme	18
4.5.5 Sala	19
4.5.6 Horário	19
4.6. Conceitos de melhoria	19
4.7. Verificar redundâncias	19
4.8. Validar modelo conceptual segundo as transações	20
4.8.1 Adicionar um novo Cliente	20
4.8.2 Adicionar uma nova Sessão	21
4.8.3 Adicionar um novo Filme	21
4.8.4 Adicionar uma nova Sala	21
4.8.5 Adicionar um novo Horário	22
4.8.6 Fazer uma reserva/compra de bilhete	22
4.9. Validação do modelo conceptual com o utilizador	23
4.10. Modelo conceptual final	31
5. Modelo Lógico	32
5.1. Derivação do Modelo Lógico	32
5.1.1 Derivação do modelo de dados lógico para obtenção de tabelas	32
5.1.2 Seleção e justificação das chaves estrangeiras	34
5.2. Validação das relações segundo regras de Normalização	35
5.2.1 Primeira Forma Normal (1FN)	35
5.2.2 Segunda Forma Normal (2FN)	35
5.2.3 Terceira Forma Normal (3FN)	37
5.3. Validar modelo segundo as transações do utilizador	38
5.3.1 Adicionar um novo cliente	38
5.3.2 Adicionar um novo filme	38
5.3.3 Adicionar uma nova sala	39
5.3.4 Fazer uma reserva/compra de bilhete	39

5.4. Verificação das restrições de integridade	40
5.4.1 Dados requeridos	40
5.4.2 Integridade de domínio	40
5.4.3 Restrições de multiplicidade	41
5.4.4 Integridade de entidade	41
5.4.5 Integridade referencial	41
5.4.6 Restrições Gerais Regras de Negócio	42
5.5. Revisão do modelo lógico com o utilizador	42
5.6. Fusão dos modelos lógicos num modelo global	42
5.7. Analisar Crescimento Futuro	42
5.8. Modelo Lógico Final	43
6. Modelo Físico	44
6.1. Representação das Relações Base	44
6.1.1 Tabela Cliente	45
6.1.2 Tabela Bilhete	46
6.1.3 Tabela Sessão	47
6.1.4 Tabela Filme	48
6.1.5 Tabela Sala	48
6.1.6 Tabela Lugar	49
6.2. Representação dos atributos derivados	50
6.2.1 Atributo derivado desconto	50
6.2.2 Atributo derivado preçoTotal	51
6.2.3 Atributo derivado tipoBilhete	51
6.2.4 Atributo derivado capacidade	51
6.3. Restrições Gerais	52
6.4. Análise de Transações	53
6.4.1 Adicionar um novo cliente	53
6.4.2 Adicionar um novo filme	53
6.4.3 Fazer uma reserva/compra de bilhete	53
6.4.4 Adicionar uma nova sala	54
6.5. Definição das vistas dos utilizadores e regras de acesso	55
6.5.1 Vistas dos utilizadores	55
6.5.2 Utilizadores e regras de acesso	56
6.6. Crescimento futuro e estimativa de espaço em disco.	58
6.6.1 Tamanho inicial	58
6.6.2 Crescimento futuro	59
6.7. Script de povoamento inicial	60
6.8. Exemplos de Queries	60
7. Conclusões e Trabalho Futuro	63

Anexos

I. Anexo 1 - Dicionário de Dados das Entidades	68
II. Anexo 2 - Dicionário de Dados dos Relacionamentos	69
III. Anexo 3 - Dicionário de Dados dos Atributos	70
IV. Anexo 4 – Script de Inicialização da Base de Dados bccBD	74
V. Anexo 5 – Script de Validação de Dados	78
VI. Anexo 6 – Script de Povoamento Inicial	83
VII. Anexo 7 – Restantes Queries	87

Índice de Figuras

Figura 1 - Diagrama de Gantt do processo de levantamento de requisitos.	7
Figura 2 - Entidade Cliente e seus atributos	20
Figura 3 - Entidade Sessão e seus atributos	21
Figura 4 - Entidade Filme e seus atributos	21
Figura 5 - Entidade Sala e seus atributos	21
Figura 6 - Entidade Horário e seus atributos	22
Figura 7 – Entidades Cliente, Bilhete, Sessão, Horário, Sala e Filme e relacionamentos	22
Figura 8 - Entidade Cliente e os seus atributos	23
Figura 9 - Relacionamento entre Cliente e Bilhete e os atributos de Bilhete	23
Figura 10 - Relacionamentos entre Cliente, Bilhete, Sessão, Horário e Filme	24
Figura 11 - Relacionamento entre Bilhete, Sessão, Horário, Filme e Sala	24
Figura 12 – Relacionamento entre entidade Bilhete e entidade Cliente	24
Figura 13 - Entidade Filme e seus atributos	25
Figura 14 – Relacionamentos entre as entidades Filme, Sessão e Horário	25
Figura 15 - Relacionamento entre a entidade Sala e a entidade Filme	25
Figura 16 - Relacionamento entre as entidades Bilhete, Sessão e Filme	26
Figura 17 – Relacionamento entre entidade Sessão e entidade Filme	26
Figura 18 - Entidade Sala e os seus atributos	26
Figura 19 - Atributo tipoSala (associado à entidade Sala)	27
Figura 20 – Relacionamento entre as entidades Sala e Filme	27
Figura 21 - Relacionamento entre as entidades Bilhete, Sessão, Filme e Sala	28
Figura 22 – Entidade Filme e alguns dos seus atributos	28
Figura 23 - Relacionamento entre Bilhete, Sessão e Filme	29
Figura 24 - Relacionamento entre as entidades Bilhete e Sessão	29
Figura 25 - Entidade Sala e seus atributos	30
Figura 26 – Entidades Horário, Sessão e Filme	30
Figura 27 – Modelo Conceptual Final	31
Figura 28 – Mapa da transação “Adicionar um novo cliente”	38
Figura 29 – Mapa da transação “Adicionar um novo filme”	38
Figura 30 – Mapa da transação “Adicionar uma nova sala”	39
Figura 31 – Mapa da transação “Fazer uma reserva/compra de bilhete”	39
Figura 32 – Modelo Lógico Final	43
Figura 33 – Tabela Cliente: representa a entidade Cliente	45
Figura 34 – Tabela Bilhete: representa a entidade Bilhete	46
Figura 35 – Tabela Sessão: representa a entidade Sessão	47
Figura 36 – Tabela Filme: representa a entidade Filme	48
Figura 37 – Tabela Sala: representa a entidade Sala	48
Figura 38 – Tabela Lugar: representa o atributo multivalor lugar	49
Figura 39 – Tabela que representa uma vista do Cliente	55
Figura 40 – Tabela que representa uma vista do Funcionário	56

Índice de Tabelas

Tabela 1 - Permissões para os utilizadores da base de dados: cliente e funcionário	57
Tabela 2 - Tamanho médio ocupado por cada registo de cada tabela da base de dados	58
Tabela 3 – Descrição das Entidades	68
Tabela 4 – Descrição dos relacionamentos entre as entidades	69
Tabela 5 – Descrição dos atributos da entidade Cliente	70
Tabela 6 – Descrição dos atributos da entidade Bilhete	71
Tabela 7 – Descrição dos atributos da entidade Horário	72
Tabela 8 – Descrição dos atributos da entidade Sessão	72
Tabela 9 – Descrição dos atributos da entidade Filme	72
Tabela 10 – Descrição dos atributos da entidade Sala	73
Tabela 11 – Descrição dos atributos do atributo Multivalor Lugar	73

1. Introdução

O projeto realizado no âmbito da unidade curricular de Base de Dados, consiste na implementação de uma base de dados para a gestão de sessões de cinema. Neste capítulo contextualizou-se o caso de estudo, mencionando-se empresas que já prestam o serviço de sistema de reservas online, bem como a **Braga Cinema Center**. De seguida é apresentado o caso de estudo, onde é descrito o serviço de reservas prestado pela empresa. Por fim, são apresentadas as motivações para o desenvolvimento desta base de dados e quais os objetivos que se pretende alcançar com a mesma.

1.1. Contextualização

Cada vez mais o cinema clássico tem menos público, principalmente público jovem. De ano para ano, são cada vez menos as pessoas que se deslocam ao cinema para assistir a filmes mais clássicos, curtas-metragens e filmes locais.

Tradicionalmente as salas de cinema que oferecem este tipo de filmes não possuem um sistema de reservas *online*. E são prejudicadas por este fator, pois o público mais jovem, com um ritmo de vida cada vez mais frenético, e que passam cada vez mais tempo no local de trabalho, preferem fazer a compra/reserva dos seus bilhetes de cinema *online*. Obrigando o cliente a fazer a reserva ou compra presencialmente, após todos estes fatores, acaba por restar pouco tempo para tratar dos afazeres mais banais do dia-a-dia e da família.

Com o avanço tecnológico, hoje em dia é cada vez mais usual efetuar-se todo ou quase todo o tipo de compras online. O tempo livre durante um dia de trabalho é cada vez menor, para que uma pessoa possa deslocar-se a uma bilheteira de cinema para comprar um bilhete. Tal não é necessário nas principais salas de cinema que projetam os típicos *blockbusters* de *Hollywood*, porém as salas de cinemas mais clássicas exigem-no. Porque não facilitar a vida das pessoas ao disponibilizar-lhes um sistema de reservas de bilhetes de cinema *online*? Assim sendo, basta ter acesso à internet e aceder ao site da empresa ou à sua aplicação, registar-se numa conta e depois efetuar a compra, de forma simples, prática e rápida. Desta forma não é necessário ter de se lidar com dinheiro.

As empresas locais da área do cinema, para além de um disponibilizarem um serviço cultural, têm também os seus interesses económicos e sociais. De modo a cumprir a sua missão, estas devem almejar alcançar níveis de eficiência e eficácia na execução dos serviços, de forma a serem consistentes e a permitirem a sua sustentação e expansão. Como tal, têm

que estar recetivas a analisar as mudanças de comportamento dos seus clientes, bem como abertas a ouvir as suas reclamações e atentas aos avanços tecnológicos.

Presentemente muitas empresas desta área detetaram e analisaram esta realidade, e como tal, decidiram investir num sistema de reservas de sessões de cinema *online*. A **Braga Cinema Center (BCC)** foi uma delas. A BCC é uma empresa que projeta filmes clássicos, curtas-metragens, filmes locais e também internacionais, presentes nos maiores festivais de cinema europeus, como o Festival de Cinema de Veneza. A empresa constatou que os seus concorrentes no mercado, que projetam os afamados filmes de *Hollywood*, estavam a conseguir uma maior parcela do negócio, e que a aderência a filmes mais clássicos e culturais estava a baixar drasticamente, principalmente o público mais jovem.

Como tal, tinham que determinar qual o motivo do decréscimo das suas vendas. Os concorrentes, situados em grandes centros comerciais e com as suas infraestruturas tecnológicas avançadas, tinham apostado num sistema de compra e reserva de bilhetes de sessões de cinema *online*, concluindo assim a BCC que a ausência deste sistema na sua empresa era uma grande lacuna.

A falta deste serviço impedia a empresa de alcançar um maior crescimento no mercado, através de um aumento substancial no número de clientes alvo. Principalmente do público jovem que não é tão adepto deste tipo de cinema mais clássico e pouco usual. Como tal a empresa pretende conceber uma aplicação de forma a prestar tal serviço, pelo que necessita de implementar uma Base de Dados, de forma a guardar os dados dos seus clientes, das suas sessões, dos seus filmes, horários e salas.

1.2. Apresentação do Caso de Estudo

A Braga Cinema Center (BCC)

A BCC é uma empresa local, da cidade de Braga, que projeta filmes clássicos, curtas-metragens, filmes locais e também internacionais, presentes nos maiores festivais de cinema europeus, como o Festival de Cinema de Veneza.

A fim de responder às exigências do mercado, como referido na contextualização anterior, a BCC decidiu disponibilizar aos seus clientes um sistema de reservas de bilhetes de sessões de cinema *online*.

Um cliente poderá efetuar uma reserva de um bilhete numa sessão de um filme, sendo logo paga no ato da reserva. Para tal este deverá fornecer o seu nome, *username*, data de nascimento, telefone, email e password, de forma a se poder registar numa conta que lhe irá permitir reservar e consultar as suas reservas.

As salas de cinema da empresa são de três tipos: *Basic* (capacidade máxima para 15 pessoas), *Silver* (capacidade máxima para 20 pessoas) e *Platinum* (capacidade máxima para 25 pessoas).

No processo de reserva o cliente deverá qual sessão de um filme deseja assistir, e conforme a escolha tomada consultar o(s) lugar(es) disponíveis de forma a escolher qual o(s) lugar(es) que deseja. Como o pagamento é realizado no ato da reserva, a empresa decidiu não permitir o cancelamento de reservas.

Com vista a atrair e alcançar uma vasta população alvo, a empresa dispõe de um desconto para jovens e outro para idosos. Os jovens com idade inferior a 25 anos terão um desconto de 15% nas reservas efetuadas e as pessoas com idade superior a 65 anos terão um desconto de 25%. E para atrair ainda mais público, principalmente os verdadeiros amantes de filmes clássicos, institui um desconto para membros *Premium*, que atribui um desconto de 30% em todas as sessões de cinema.

A BCC pretende analisar o comportamento dos nossos clientes e com isso realocar melhor os seus recursos, logo deverá ser possível fazer uma listagem de todas as reservas de uma determinada sessão ou de um filme.

1.3. Motivação e Objetivos

1.3.1 Motivação

Nesta secção foram identificadas as principais lacunas que o anterior serviço da BCC oferecia aos seus clientes. Concluindo-se a partir daí o que se deve alterar de modo a satisfazer o cliente e o porquê da empresa necessitar de uma Base de Dados.

- Os clientes queixavam-se que não conseguiam fazer reservas com antecedência e quando iam comprar o bilhete já não havia lugares disponíveis (lugares esgotados)
- Os clientes acabavam por escolher as salas de cinema das superfícies comerciais com o seu serviço *online*, porque não tinham acesso à disponibilidade de lugares e não tinham tempo nem paciência para se deslocar aos postos de venda
- Os utilizadores manifestaram vontade de escolher o seu próprio lugar
- Disponibilizar serviços a pensar no utilizador para aumentar as vendas (descontos para membros, possibilidade de reserva)
- Possibilitar uma futura aplicação mobile onde cada utilizador possa gerir as suas reservas (aproximar o cliente da empresa)
- Desconhecimento do perfil do cliente e recolha de estatísticas/informação relevantes para otimizações futuras:
- Com quanta antecedência é feita a reserva (estudar possíveis promoções)
- Criação de sistema de pontos
- Gestão de recursos (salas, lugares) antecipada (em vez de previsão)

1.3.2 Objetivos

Nesta secção foram determinados os objetivos que a empresa BCC visa alcançar com a implementação de uma Base de dados.

- Possibilitar um serviço ao cliente, onde este possa efetuar as suas reservas e informatizar esse processo
- Permitir ao utilizador consultar, ter acesso e escolher o lugar que quer reserva
- Gestão de reserva de bilhetes eficiente e centralizada
- Obter uma forma simples de consultar todas as reservas de uma sessão de um filme
- Recolher dados que permitam traçar perfis de clientes
- Futura personalização da experiência

1.4. Estrutura do Relatório

Este relatório está dividido em sete capítulos.

O primeiro capítulo é um capítulo introdutório que faz uma contextualização e apresentação do caso de estudo, assim como aborda a motivação e objetivos para a implementação da base de dados.

O segundo capítulo trata-se da análise e viabilidade do projeto e o terceiro capítulo refere-se ao levantamento dos requisitos.

O quarto capítulo, sendo um dos mais importantes, aborda a construção do modelo conceptual, no qual se identificam as entidades, relacionamentos e atributos; é também validado o modelo segundo as transações.

No quinto capítulo, desenvolveu-se o modelo lógico a partir do modelo conceptual; após obtenção deste, seguindo a metodologia da bibliografia, validou-se as relações segundo as regras da normalização e as transações.

No sexto capítulo, que aborda a implementação do modelo físico, a base de dados construída permite a validação dos dados e das restrições gerais identificadas nos requisitos.

No sétimo capítulo realizou-se uma análise crítica do trabalho realizado.

2. Viabilidade do projeto

2.1. Análise e Justificação da viabilidade do projeto

2.1.1 Facilitar a vida ao cliente

Hoje em dia, a maioria dos clientes não tem tempo para, durante o horário de bilheteira, fazer a compra do bilhete que deseja. Assim, com o novo sistema, quando chega a casa, já durante a noite, tem a possibilidade de fazer a reserva do seu bilhete online, sem ser necessário fazer um telefonema ou deslocar-se à bilheteira durante *business hours*. Estatísticas revelam que cada vez mais reservas hoje em dia são feitas online, em casa durante a noite.

O cliente consegue obter uma visão automática da disponibilidade de lugares na sala.

2.1.2 A que preço (para cumprir os objetivos)?

A empresa já possui a equipa informática que vai desenvolver a aplicação e respetiva base de dados. A nível de *hardware* e equipamento, não será necessário um *upgrade*, os *desktops* que a empresa possui são satisfatórios e suficientes para a utilização do novo sistema.

No entanto, é necessária uma ligação ao servidor, cuja implementação e manutenção são de relativo baixo custo, e implementação rápida e de fácil acesso, muito graças à inovação das tecnologias e surgimento do *cloud computing*.

2.2. Vantagens e ganhos operacionais

A implementação de um sistema de bases de dados é uma mais valia para a empresa, por vários fatores. Primeiramente, com a implementação deste sistema, é possível no mesmo minuto termos por exemplo dez pessoas ao mesmo tempo a fazer a reserva de bilhetes.

O cliente passa a poder fazer uma reserva de uma forma muito mais simples, rápida e direta, o que vai fazer com que o queira fazer de modo mais frequente, aumentando o número de vendas de bilhetes.

O sistema vai automatizar o processo de reserva de bilhetes, não sendo necessário um funcionário para realizar essa tarefa, sendo o próprio cliente a fazê-la diretamente na aplicação. Assim sendo, os funcionários anteriormente responsáveis por essa tarefa serão agora alocados para outros serviços, otimizando o funcionamento da empresa no geral.

Com este sistema vai ser possível maximizar o número de reservas, pois como o ato de reserva é tão imediato, vai ser possível, a todo o momento ver quais os lugares disponíveis, no segundo após a reserva ser libertada. O cliente pode cancelar a sua reserva até meia hora antes do início do filme. E assim, outro cliente interessado pode fazer a reserva daquele lugar, mal ele esteja disponível. Sem burocracia, telefonemas, e com a vantagem da atualização em tempo real da disponibilidade dos bilhetes.

Será possível uma melhor gestão das sessões, no futuro, com a informação de quais os horários onde são reservados mais bilhetes, alocando mais sessões para esses filmes a essa hora, retirando sessões nos horários onde há menos clientes a assistir ao filme. Quais são os filmes mais procurados e o intervalo de idades que mais assistem aos filmes são informações que mais tarde serão úteis no planeamento do funcionamento da empresa.

O pagamento total do bilhete é realizado no momento da reserva. É uma mais valia para a empresa, que assim garante o rendimento instantâneo proveniente da reserva do bilhete. Caso o cliente deseje cancelar a reserva, tem até meia hora antes do início do filme para fazê-lo, e o dinheiro ser-lhe-á devolvido automaticamente.

3. Levantamento de requisitos

O processo de levantamento de requisitos é um processo longo e complexo. De forma a controlar este processo definiu-se um plano que consiste nas etapas que se encontram enumeradas a seguir:

1. Análise do domínio
2. Identificação das fontes dos requisitos
3. Análise dos *stakeholders*
4. Seleção de técnicas, abordagens e ferramentas a utilizar
5. Identificação dos requisitos

Para a gestão da execução destas etapas elaborou-se um plano (diagrama de *Gantt*) de 5 semanas, que se encontra definido na Figura 1, onde se indica o início e o fim de cada uma das etapas ao longo do tempo.

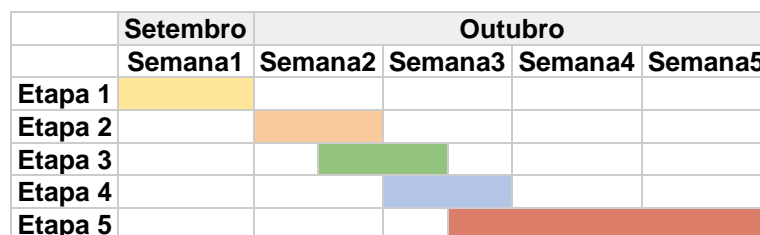


Figura 1 - Diagrama de *Gantt* do processo de levantamento de requisitos.

De seguida encontram-se definidas cada uma destas etapas e as conclusões retiradas da execução de cada uma delas.

3.1. Análise do domínio

Nesta etapa o intuito é perceber o contexto onde o sistema vai ser implementado, ou seja, o domínio do sistema.

Durante esta etapa, identificou-se o processo de reserva de bilhete já existente: cada cliente desloca-se fisicamente até à bilheteira do cinema para efetuar a compra do bilhete. O bilhete tem o preço e a informação da sessão a que está associado bem como a data e hora. De acordo com o contexto informático atual, foi feita uma análise ao significado atual de

reserva, que correntemente se encontra ligado ao *online* e permite comodidade na operação bem como nas operações consulta e histórico associadas.

Esta etapa é importante pois pretende-se implementar um sistema que complemente e adicione funcionalidades sem complicar nem impossibilitar funcionalidades que já existiam.

3.2. Identificação das fontes dos requisitos

A segunda etapa passa por identificar possíveis fontes que possam transmitir requisitos. Estas fontes podem não ser totalmente óbvias à primeira vista.

Ao longo desta etapa a pesquisa foi exaustiva na procura por fontes de requisitos. As partes interessadas no sistema (*stakeholders*) foram a fonte mais óbvia para obter os requisitos e fornecer informações relevantes sobre as necessidades do sistema e utilizadores. Contudo, sistemas existentes ou similares (de reservas) foram outras fontes identificadas para extrair requisitos.

Quanto mais completa for a lista das fontes, melhor será a documentação e identificação dos requisitos.

3.3. Análise dos *stakeholders*

A etapa de análise dos *stakeholders* (partes interessadas) é a fase que permite perceber quem são os *stakeholders* permitindo assim que as técnicas da próxima etapa sejam adaptadas a cada um deles.

Esta análise permitiu identificar a empresa que está a desenvolver o sistema como um *stakeholder* óbvio. Contudo, o cliente final a quem o sistema se destina, tem igual importância e foi também identificado como um *stakeholder*. Outra das partes interessadas são os gestores de conteúdo das sessões que ficarão responsáveis por adicionar e editar filmes e sessões, mantendo-os atualizados. Assim, os *stakeholders* identificados foram:

- Empresa
- Cliente
- Gestor de conteúdo das sessões

Esta identificação permitiu adaptar as técnicas escolhidas na etapa seguinte.

3.4. Seleção das técnicas, abordagens e ferramentas a utilizar

Esta etapa prende-se com a seleção da(s) técnica(s) para o levantamento de requisitos. Estas técnicas deverão ser escolhidas de acordo com o sistema e *stakeholders*.

De acordo com cada um dos *stakeholders* identificados, foram escolhidas as seguintes técnicas de levantamento de requisitos:

- **Entrevistas/Conversas** - Foram feitas algumas entrevistas a pessoas que têm como hábito assistir a filmes no cinema. Eram pessoas que já tinham efetuado diversas reservas de bilhete e tinham informações relevantes sobre vantagens do processo atual e entraves que encontravam na experiência existente.
- **Observação** - Sem alertar nenhum dos intervenientes foi feito um processo de observação do processo de reserva de bilhete. Foram anotados os diferentes passos e identificados alguns pontos de melhoria. Foi decisiva a observação anónima uma vez que permitiu analisar o processo sem nenhuma condicionante.
- **Introspeção e sessões de *brainstorming*** - A introspeção foi escolhida como uma das técnicas e onde foram identificadas algumas funcionalidades que se pensa que o utilizador possa tirar vantagens. As sessões de brainstorming com a equipa de desenvolvimento foram utilizadas para refinar e produzir novas funcionalidades baseadas no que se obteve da introspeção.
- **Personas** - A utilização das personas como técnica para extração de requisitos permitiu extrair e catalogar utilizadores do sistema e assim melhor delinear os perfis que se encontram identificados na etapa seguinte

3.5. Identificação dos perfis de utilização e requisitos

Perfis de utilização:

Com as técnicas de levantamento de requisitos selecionadas, foi possível identificar os seguintes perfis de utilização do sistema:

- **Cliente normal:** é o utilizador do sistema que quer efetuar reservas de bilhetes
- **Gestor de conteúdo de sessões:** é o utilizador que faz a gestão das sessões de filmes que a empresa disponibiliza bem como salas que a empresa possui
- **Analista de informação da empresa:** é o utilizador que extrai análises e estatísticas das reservas e das sessões.

Os perfis de utilização identificados permitiram perceber casos de uso normal do sistema e em conjunto com as técnicas de levantamentos de requisitos escolhidas, permitiram que fossem identificados os seguintes requisitos.

Requisitos:

1. Um cliente pode registar-se com o seu nome, *username*, e-mail, telefone, password e data de nascimento.
2. Um cliente deverá poder consultar todos os seus bilhetes comprados, bem como os seus detalhes.
3. O cliente deve conseguir reservar um bilhete para uma sessão de um determinado filme, a determinada hora.
4. Um bilhete corresponde a um lugar na sala de cinema da sessão de um filme a determinada hora.
5. Um bilhete pode ter um desconto associado. (jovens, +65, membro *premium*).
6. Um filme deve ter associado um título, duração, idade mínima, data de estreia e data de fim de projeção.
7. Consultar o horário de um filme.
8. Cada filme só é projetado numa sala específica.
9. Consultar todos os bilhetes comprados para uma sessão de um filme.
10. Cada sessão de um filme pode ter diferentes preços.
11. Uma sala é caracterizada pelo seu nome e tipo, e tem um número de lugares fixo consoante o tipo, que corresponde à sua capacidade.
12. Uma sala pode ser do tipo *Basic*, *Silver* ou *Platinum*.
13. Uma sala pode projetar vários filmes, mas um filme só pode ser projetado numa sala.
14. Consultar a lista de lugares disponíveis para uma determinada sessão.
15. Um filme tem uma data de estreia e uma data em que deixa de ser projetado.
16. Deverá ser possível consultar dado um mês, o número de bilhetes vendidos e o total (€) para uma dada sessão ou filme.
17. Dado um mês, deverá poder consultar-se o top 10 das sessões mais, e menos realizadas
18. Deverá ser possível adicionar mais salas e respetivos lugares ao sistema
19. Deverá ser possível adicionar mais filmes, sessões e horários ao sistema

4. Modelo Conceptual

4.1. Identificação das Entidades

O primeiro passo a realizar no modelo conceptual é definir as entidades, face aos requisitos considerados. As entidades e as suas descrições encontram-se registadas num dicionário de dados, que pode consultar no Anexo I.

4.1.1 Cliente

A entidade Cliente representa a pessoa que reserva os bilhetes de cinema. O Cliente possui um nome, email, telefone, data de nascimento (para verificação de elegibilidade para usufruto do desconto jovem ou idosos), um atributo *membroPremium* (que verifica se o Cliente é membro *Premium* da BCC e pode usufruir do respetivo desconto) e um *username* e password para se autenticar na aplicação.

4.1.2 Bilhete

A entidade Bilhete refere-se à informação de reserva de um lugar na sessão de um filme a determinada hora, possuindo informação sobre a data da sessão, a data de compra/reserva, o lugar na sala, o tipo de bilhete (normal ou promocional), o desconto aplicado, o IVA em vigor e o preço total do bilhete.

4.1.3 Sessão

A entidade Sessão refere-se à projeção de um determinado filme a determinada hora de um dia. Diferentes sessões de um filme podem ter preços diferentes.

4.1.4 Filme

A entidade Filme representa o filme que vai ser projetado numa sala, em diferentes sessões (que tomam lugar a diferentes horas do dia). É caracterizado pelo seu título, duração,

data de estreia, data de fim da projeção e idade mínima para assistir ao filme (não acompanhado de um adulto responsável devidamente identificado).

4.1.5 Sala

A entidade Sala representa o local onde vai ser projetado um filme. A Sala possui um nome, um tipo, um conjunto de lugares e uma capacidade, que depende do seu tipo.

4.1.6 Horário

A entidade Horário representa os diferentes horários que uma sessão pode tomar. O atributo hora refere-se à hora do dia que a sessão tomará início.

4.2. Identificação dos Relacionamentos

O passo 2 consiste em identificar os relacionamentos existentes entre as entidades.

No anexo II encontra-se o dicionário de dados com a documentação referente à identificação e descrição dos relacionamentos entre as entidades.

Foram identificados cinco relacionamentos entre as entidades identificadas no passo anterior

4.2.1 Relacionamento Cliente – Bilhete

Relacionamento: Cliente reserva um Bilhete

Descrição: Identifica a ação do cliente: realizar a reserva de um bilhete

Cardinalidade: Cliente(1); Bilhete(N)

- Um cliente pode reservar vários bilhetes logo, um cliente vai estar associado aos N bilhetes reservados
- Um bilhete está associado a apenas um cliente

Participação: Cliente(P); Bilhete(T)

- Um cliente pode ter zero ou mais bilhetes
- Um bilhete tem que ter sempre um cliente associado

Atributos: Não existem atributos no relacionamento entre as duas entidades

4.2.2 Relacionamento Bilhete – Sessão

Relacionamento: Bilhete refere Sessão

Descrição: Identifica que cada bilhete refere-se a uma sessão

Cardinalidade: Bilhete(N); Sessão(1)

- Um bilhete refere-se apenas a uma sessão
- Uma sessão encontra-se associada a vários bilhetes

Participação: Bilhete(T); Sessão(P)

- Um bilhete tem que ter sempre associada uma sessão
- Uma sessão pode não ter nenhum bilhete ou ter vários bilhetes associados

Atributos: Não existem atributos no relacionamento entre as duas entidades

4.2.3 Relacionamento Sessão – Filme

Relacionamento: Sessão refere Filme

Descrição: Identifica que cada sessão se refere a um filme

Cardinalidade: Sessão(N); Filme(1)

- Uma sessão refere-se apenas a um filme
- Um filme encontra-se associado a várias sessões

Participação: Sessão(T); Filme(P)

- Uma sessão tem que ter sempre associado um filme
- Um filme pode não ter nenhuma ou ter várias sessões associadas

Atributos: Não existem atributos no relacionamento entre as duas entidades

4.2.4 Relacionamento Sala - Filme

Relacionamento: Sala projeta Filme

Descrição: Identifica que a Sala projeta um Filme

Cardinalidade: Sala(1); Filme(N)

- Uma sala projeta vários filmes
- Um filme é apenas projetado numa sala

Participação: Sala(P); Filme(T)

- Uma sala pode projetar zero ou mais filmes
- Um filme tem que ter sempre uma sala associada

Atributos: Não existem atributos no relacionamento entre as duas entidades

4.2.5 Relacionamento Horário - Sessão

Relacionamento: Horário refere Sessão

Descrição: Identifica que Horário refere uma Sessão

Cardinalidade: Horário(1); Sessão(N)

- Um horário agrupa várias sessões
- Uma sessão tem apenas um horário

Participação: Horário(P); Sessão(T)

- Um horário pertence a zero ou várias sessões
- Uma sessão tem que ter sempre um horário associado

Atributos: Não existem atributos no relacionamento entre as duas entidades

4.3. Identificação dos Atributos

No anexo III pode-se consultar o dicionário de dados com a descrição dos atributos de cada entidade. Nesta secção, pretende-se complementar os dicionários de dados com algumas justificações face aos requisitos que levaram a considerar os atributos que se tem no modelo conceptual.

4.3.1 Atributos da Entidade Cliente

A entidade Cliente tem os seguintes atributos:

- idCliente
- nome
- dataNascimento
- email
- telefone
- dataRegisto
- password
- *username*
- *membroPremium*

Todos estes atributos são simples. A justificação para a existência destes atributos prende-se com a necessidade de cada cliente se registar numa conta de modo a poder reservar e consultar bilhetes para uma sessão de cinema. Para se registar numa conta, o cliente precisa de disponibilizar o seu nome, data de nascimento, telefone, email, *username* e password. Por fim, para cada cliente o sistema deve guardar esta informação. Se o cliente for membro *Premium* da BCC, o sistema guarda essa informação no atributo *membroPremium*.

4.3.2 Atributos da Entidade Bilhete

A entidade Bilhete tem os seguintes atributos:

- idBilhete
- tipoBilhete
- dataSessao
- dataCompra
- lugar
- preçoTotal
- IVA
- desconto

Todos os atributos são simples, exceto os atributos desconto, tipoBilhete, preçoTotal.

- O atributo desconto é derivado pois depende se o cliente tem direito a desconto, isto é, se a sua idade (verificada no atributo dataNascimento) no ato da compra for inferior a 25 anos ou superior a 65 anos, ou se for membro *Premium*. Se usufruir de desconto, o atributo desconto irá ter o valor de 0.15, 0.25 ou 0.30, respetivamente. É calculada a idade do cliente, e a verificação de que é membro *Premium*, que por sua vez indica o desconto que irá ser aplicado. Os descontos não são cumulativos.
- O atributo tipoBilhete é derivado pois depende se o cliente tem direito a desconto, se o valor do atributo desconto é 0.15, 0.25 ou 0.30, o bilhete é do tipo promocional (P). Se o valor for 0.0 o bilhete é do tipo normal (N).
- O atributo preçoTotal é derivado dos atributos preçoBase, IVA e desconto. É calculado da seguinte forma:

$\text{preçoTotal} = ((\text{preçoBase} * (1 + \text{IVA})) * (1 - \text{desconto}))$

Ao preço base, é adicionado o valor do IVA, e no final subtrai-se o desconto.

Um bilhete corresponde a um lugar na sala de cinema numa sessão de um filme, sendo obrigatório ter como atributo de Bilhete, o número do lugar na sala.

O atributo tipo de Bilhete é utilizado para caracterizar um bilhete, dado que a BCC tem como um dos seus requisitos, que o sistema deva permitir reservas com desconto (jovens com menos de 25, pessoas com mais de 65, e membros *Premium*). Logo há a necessidade de distinguir entre bilhetes normais e promocionais.

O atributo dataSessao é necessário, visto que é requerido que um cliente consiga reservar um bilhete para uma sessão de um filme, numa determinada data, assim como ser posteriormente possível consultar dado um determinado mês, o número de bilhetes vendidos e o total faturado para um dado filme.

4.3.3 Atributos da Entidade Sessão

A entidade Sessão tem os seguintes atributos:

- idSessao
- preçoBase

Todos estes atributos são simples. A justificação para a existência deste atributo preçoBase na entidade Sessão prende-se com a necessidade de estabelecer o valor monetário de assistir a uma sessão de um filme, sem acréscimo do IVA e sem a subtração do desconto. Este atributo encontra-se na entidade Sessão e não na entidade Filme para que se possam atribuir diferentes preços a diferentes sessões de um mesmo filme.

4.3.4 Atributos da Entidade Filme

A entidade Filme tem os seguintes atributos:

- idFilme
- título
- duração
- dataEstreia
- dataFim
- idadeMínima

Todos estes atributos são simples. A justificação para a existência destes atributos é garantida pelos requisitos: permite a reserva de um lugar numa sessão de um filme com um título, duração, idadeMínima e dataEstreia, dataFim, para verificar se um filme ainda se encontra em projeção no cinema.

4.3.5 Atributos da Entidade Sala

A entidade Sala tem os seguintes atributos:

- idSala
- nomeSala
- tipoSala
- capacidade
- lugar

Todos estes atributos são simples, exceto o atributo capacidade e lugar. O atributo capacidade é derivado pois necessita do atributo tipoSala para saber se a sala é *Basic*, *Silver* ou *Platinum*, pois cada uma delas tem uma capacidade máxima definida diferente (15, 20 e 25 respetivamente).

O requisito que permite a consulta dos lugares disponíveis na sala de determinada sessão de um filme, de modo ao cliente escolher um, assim como o requisito que não permite a reserva de uma sessão já esgotada, ambos garantem desde logo a necessidade de saber qual a capacidade da sala de cinema.

O atributo multivalor lugar corresponde à lista de todos os lugares de determinada sala.

A BCC só possui 3 tipos de salas de cinema: *Basic*, *Silver* ou *Platinum*, é necessário ter um atributo que caracterize o tipo de sala, tipoSala.

O atributo nomeSala é o nome da sala de cinema, único para cada sala.

4.3.6 Atributos da Entidade Horário

A entidade Horário tem os seguintes atributos:

- idHorario
- hora

Ambos são atributos simples. O atributo hora corresponde à hora de início que determinada sessão terá.

4.4. Determinar o domínio dos atributos

Anteriormente foram identificados todos os atributos de cada entidade e foram caracterizados os seus tipos. Uma vez que, todos que a maioria dos atributos identificados nas entidades são do tipo simples, não têm nenhuma especificidade, logo o seu domínio está ligado intrinsecamente apenas ao domínio do tipo de dados.

Contudo para alguns dos os atributos foi determinado o seu domínio mais restrito, consulta anexo III.

4.5. Determinação chaves candidatas, primárias e alternativas

4.5.1 Cliente

Chaves candidatas: idCliente, email, *username*

Chave primária: idCliente

Justificação: Como o cliente tem de fornecer um email e *username*, únicos e alusivos só a ele, ambos são atributos candidatos para chave primária, pois não vão possuir nenhum valor nulo ou repetido. Porém, escolhemos o atributo idCliente para chave primária para que o utilizador possa mudar o seu *username* e email sempre que pretender.

Chaves alternativas: *username*, email

4.5.2 Bilhete

Chaves candidatas: idBilhete

Chave primária: idBilhete

Justificação: O atributo idBilhete foi escolhido como chave primária pois é o único candidato a chave, e também porque é a identificação única do bilhete, o valor nunca vai repetir, e nunca haverá dois bilhetes com o mesmo número.

Chaves alternativas: Nenhuma

4.5.3 Sessão

Chaves candidatas: idSessão

Chave primária: idSessão

Justificação: O atributo idSessão foi escolhido como chave primária pois é o único candidato a chave, e também porque é a identificação única da sessão, o valor nunca vai repetir, e nunca haverá duas sessões com o mesmo número.

Chaves alternativas: Nenhuma

4.5.4 Filme

Chaves candidatas: idFilme, título

Chave primária: idFilme

Justificação: O atributo idFilme foi escolhido como chave primária pois identifica o filme de forma clara e não ambígua. Como o título de um filme está sujeito a erros ou duplicados, decidiu-se optar por um idFilme sem significado, disponibilizado pela base de dados.

Chaves alternativas: título

4.5.5 Sala

Chaves candidatas: idSala, nomeSala

Chave primária: idSala

Justificação: O atributo idSala foi escolhido como chave primária pois identifica a sala de forma clara e não ambígua. Como o nome de uma sala está sujeito a erros ou duplicados, decidiu-se optar por um idSala sem significado, disponibilizado pela base de dados.

Chaves alternativas: nomeSala

4.5.6 Horário

Chaves candidatas: idHorario, hora

Chave primária: idHorario

Justificação: O atributo idHorario foi escolhido como chave primária pois identifica a sala de forma clara e não ambígua, e também porque é a identificação única da hora, o valor nunca vai repetir, e nunca haverá duas horas com o mesmo número. Decidiu-se optar por um idHorario sem significado, disponibilizado pela base de dados.

Chaves alternativas: hora

4.6. Conceitos de melhoria

Neste ponto do projeto não foram considerados conceitos de melhoria. Poderão ser uma possibilidade futura a ter em consideração.

4.7. Verificar redundâncias

A metodologia seguida especifica o presente passo para analisar o modelo conceptual e verificar a existência de possíveis redundâncias.

Esta etapa foi verificada diversas vezes ao analisar o modelo conceptual que foi sendo obtido e refinado. Analisando o último modelo conceptual:

1. Reexaminar os relacionamentos um para um (1:1)

O modelo conceptual não apresenta relacionamentos um para um.

2. Relacionamentos redundantes

Analisando os relacionamentos identificados nenhum se verifica redundante. Todos os relacionamentos se mostram fundamentais para obter diferentes informações. Não existem “ciclos” de relacionamentos entre as diferentes entidades, uma vez que, para cada uma das entidades existe apenas um caminho:

Cliente => Bilhete => Sessão => Filme => Sala
Cliente => Bilhete => Sessão => Horário

3. Considerar a dimensão temporal

Analisando os relacionamentos entre as entidades verifica-se que, são suficientes, para assegurar as associações perante possíveis alterações futuras.

4.8. Validar modelo conceptual segundo as transações

O objetivo deste passo é verificar se o modelo conceptual suporta as transações requeridas. Usando o modelo, tentamos realizar as operações manualmente. Se as conseguirmos realizar, confirmamos que o modelo conceptual comporta as transações requeridas. Se não conseguirmos realizar a transação manualmente, então significa que existe um problema no modelo conceptual que precisa de ser resolvido, como por exemplo uma entidade, atributo ou relacionamento que foi omitido do modelo.

4.8.1 Adicionar um novo Cliente

Para se adicionar um novo utilizador, é necessário verificar se o nome de utilizador e email que escolheu já existem no sistema. Tal pode ser verificado nos atributos **username** e **email**. Caso este nome de utilizador e email ainda não estejam na base de dados, o atributo **username** guarda o nome de utilizador e o atributo **email** guarda o email sugerido pelo cliente e são adicionadas as restantes informações relativas à entidade cliente: password, nome, data de nascimento, telefone e data de registo. O atributo *membroPremium* é falso por *default*.



Figura 2 - Entidade Cliente e seus atributos

4.8.2 Adicionar uma nova Sessão

No modelo conceptual como temos a entidade Sessão, é possível adicionar novas sessões. Ao se adicionar uma nova sessão, esta fica identificada pelo atributo chave primária idSessão, e adicionamos informação ao atributo restante, o preço base.



Figura 3 - Entidade Sessão e seus atributos

4.8.3 Adicionar um novo Filme

No modelo conceptual como temos a entidade Filme, é possível adicionar novos filmes. Ao se adicionar um novo filme, este fica identificado pelo atributo chave primária idFilme, e adicionamos informação aos restantes atributos: título, duração, dataEstreia, dataFim e idadeMínima.

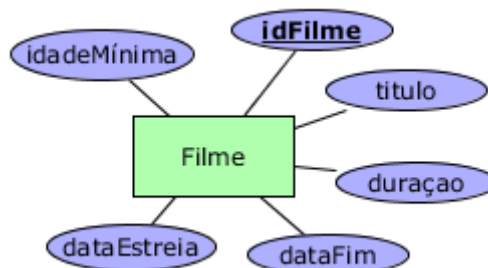


Figura 4 - Entidade Filme e seus atributos

4.8.4 Adicionar uma nova Sala

A entidade Sala permite que possamos adicionar uma nova sala. Uma sala é identificada pelo seu atributo chave primária idSala, e depois é adicionada informação sobre o seu nome, tipo, capacidade e lugar. Pelo modelo conceptual, é possível adicionar a informação sobre o lugar através do atributo multivalor lugar.

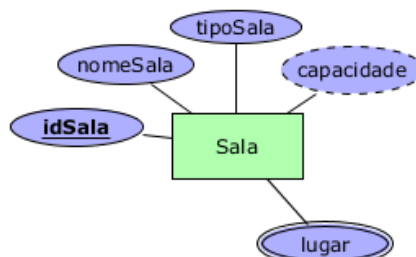


Figura 5 - Entidade Sala e seus atributos

4.8.5 Adicionar um novo Horário

No modelo conceptual como temos a entidade Horário, é possível adicionar novos horários. Ao se adicionar um novo horário, este fica identificado pelo atributo chave primária idHorário, e adicionamos informação ao atributo restante hora.

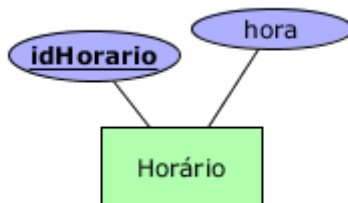


Figura 6 - Entidade Horário e seus atributos

4.8.6 Fazer uma reserva/compra de bilhete

Através do relacionamento entre a entidade Cliente e entidade Bilhete é possível um cliente reservar/comprar um bilhete.

O relacionamento entre a entidade Bilhete e a entidade Sessão permite associar o bilhete reservado/comprado a determinada sessão.

O relacionamento entre a entidade Sessão e a entidade Horário permite associar o bilhete a determinado horário. O relacionamento entre a entidade Sessão e a entidade Filme permite associar o bilhete a determinado filme.

O relacionamento entre a entidade Filme e a entidade Sala permite associar o bilhete um lugar. Toda a informação sobre o bilhete é adicionada à base de dados através dos atributos definidos para a entidade Bilhete: lugar, data da sessão, data da compra, tipo de bilhete, preço total, IVA e desconto. O bilhete é identificado pelo atributo chave primária idBilhete.

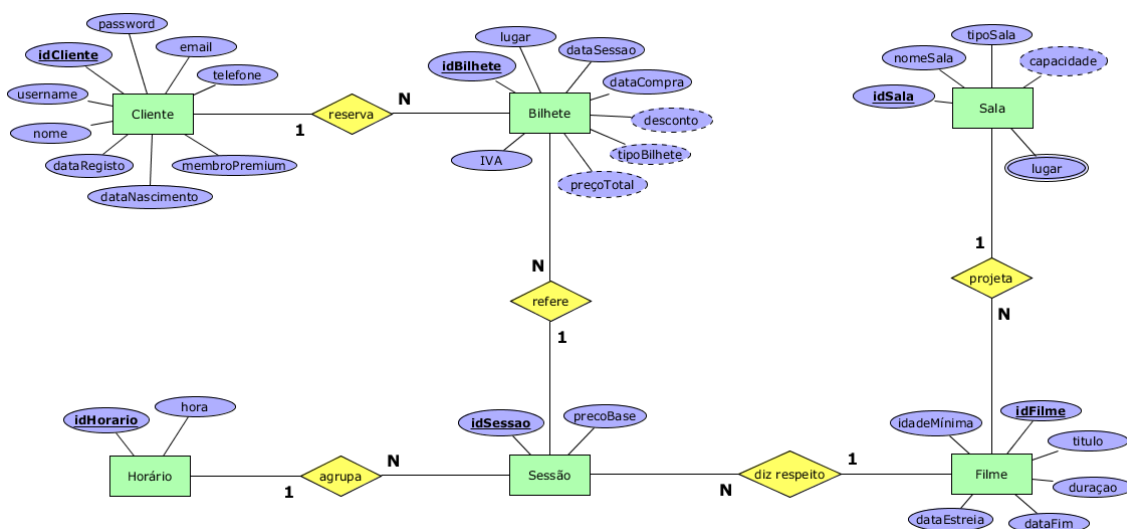


Figura 7 – Entidades Cliente, Bilhete, Sessão, Horário, Sala e Filme e respectivos relacionamentos

4.9. Validação do modelo conceptual com o utilizador

No sentido de validar o modelo conceptual com o utilizador verificou-se se o modelo conceptual consegue responder aos requisitos do utilizador. Para tal, analisou-se requisito a requisito, identificando-se o mais relevante do modelo conceptual para cada requisito, omitindo eventuais pontos menos importantes.

1. Um cliente pode registar-se com o seu nome, username, e-mail, telefone, password e data de nascimento



Figura 8 - Entidade Cliente e os seus atributos

A entidade Cliente tem associado todos os atributos referidos.

2. Um cliente deverá poder consultar todos os seus bilhetes comprados, bem como os seus detalhes

O relacionamento entre cliente e bilhete permite que um cliente possa ter associado N bilhetes e, portanto, ter acesso às várias reservas/compras efetuadas. Os atributos de bilhete permitem consultas aos detalhes dos bilhetes reservados/comprados.

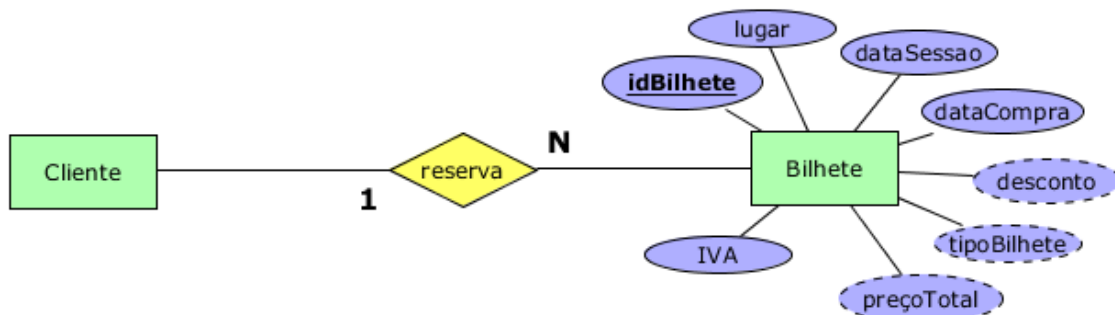


Figura 9 - Relacionamento entre Cliente e Bilhete e os atributos de Bilhete

3. O cliente deve conseguir reservar um bilhete para uma sessão de um determinado filme, a determinada hora.

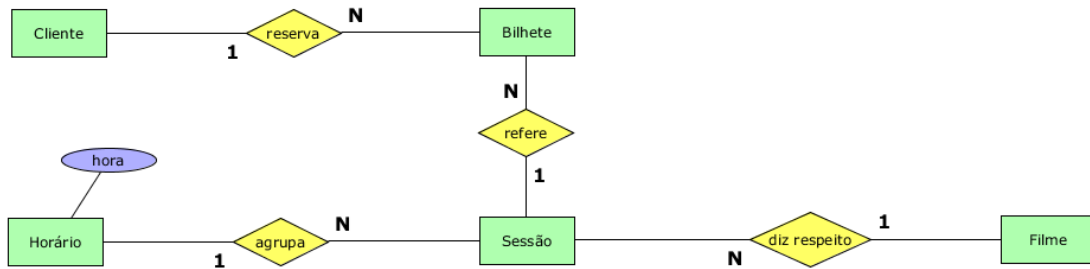


Figura 10 - Relacionamentos entre Cliente, Bilhete, Sessão, Horário e Filme

O relacionamento entre a entidade cliente e a entidade bilhete permite que exista a ação de o cliente reservar um bilhete. Uma vez que bilhete se refere a uma sessão então permite-se que um cliente reserve um bilhete para um filme a determinada hora.

4. Um bilhete corresponde a um lugar na sala de cinema da sessão de um filme a determinada hora.

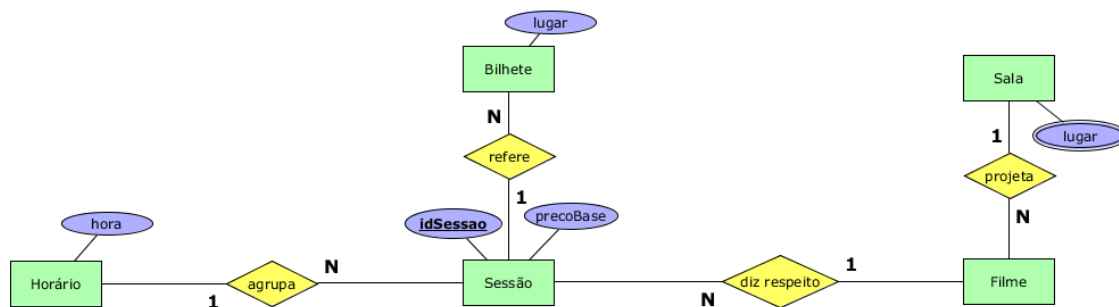


Figura 11 - Relacionamento entre Bilhete, Sessão, Horário, Filme e Sala

A entidade bilhete define como atributo número de lugar (*lugar*) que está reservado através do bilhete.

Uma vez que, o bilhete se encontra associado a uma sessão que, por sua vez, está associada a um filme e hora, e que utiliza uma determinada sala então, esse lugar corresponde a um lugar na sala de cinema da sessão de um filme a determinada hora.

5. Um bilhete pode ter um desconto associado. (jovens, +65, membro premium).

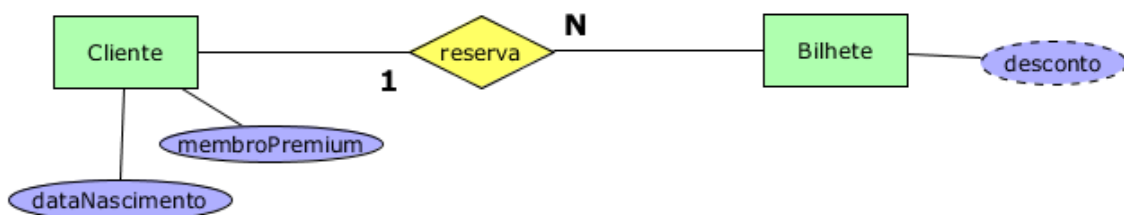


Figura 12 – Relacionamento entre entidade Bilhete e entidade Cliente

O atributo desconto da entidade bilhete permite possibilitar um desconto associado ao bilhete, e os atributos dataNascimento e *membroPremium* permitem que esse desconto seja derivado deles derivados.

6. Um filme deve ter associado um título, duração, idade mínima, data de estreia e data de fim de projeção.

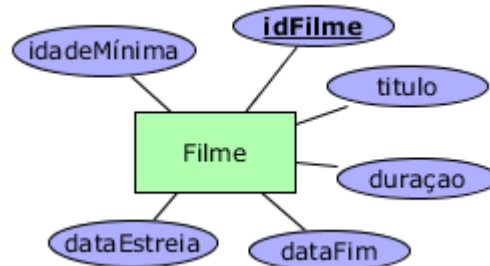


Figura 13 - Entidade Filme e seus atributos

Os atributos título, duração, idade mínima, data de estreia e data de fim da entidade Filme permitem definir esta entidade.

7. Consultar o horário de um filme

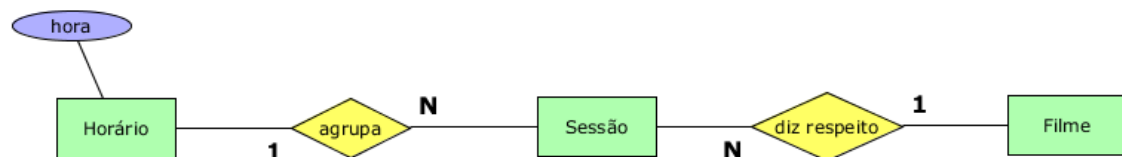


Figura 14 – Relacionamentos entre as entidades Filme, Sessão e Horário

A entidade Horário tem o atributo hora, que é a hora de início da sessão de um filme. Este atributo permite consultar o horário do Filme.

8. Cada filme só é projetado numa sala específica

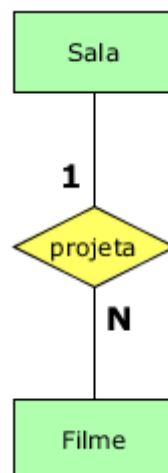


Figura 15 - Relacionamento entre a entidade Sala e a entidade Filme

O relacionamento entre a entidade Sala e a entidade Filme é de 1 para N, definindo-se que um filme apenas utiliza uma sala para ser projetado.

9. Consultar todos os bilhetes comprados para uma sessão de um filme.

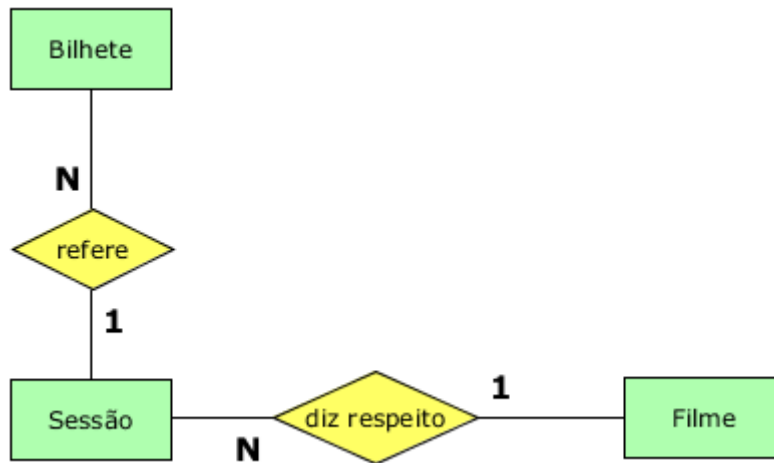


Figura 16 - Relacionamento entre as entidades Bilhete, Sessão e Filme

O relacionamento entre a entidade Bilhete e a entidade Sessão definem que uma sessão abrange vários bilhetes, portanto, possibilita consultar todos os bilhetes associados a essa sessão, que por sua vez está associada a um filme.

10. Cada sessão de um filme pode ter diferentes preços.



Figura 17 – Relacionamento entre entidade Sessão e entidade Filme

A entidade Sessão possui o atributo preçoBase e não o filme. Por este motivo, é possível ter diferentes preços para diferentes projeções de um filme.

11. Uma sala é caracterizada pelo seu nome e tipo, e tem um número de lugares fixo consoante o tipo, que corresponde à sua capacidade

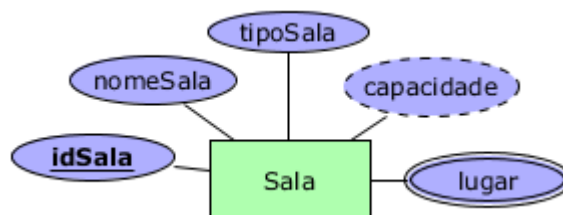


Figura 18 - Entidade Sala e os seus atributos

A entidade Sala tem como atributos nomeSala, tipoSala, capacidade e um atributo multivalor lugar, que corresponde aos lugares que a sala possui.

12. Uma sala pode ser do tipo Basic, Silver ou Platinum

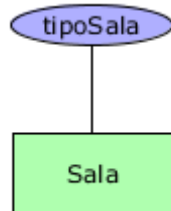


Figura 19 - Atributo tipoSala (associado à entidade Sala)

O atributo tipoSala, da entidade Sala permite que uma sala possa ser do tipo Basic, Silver ou Platinum.

13. Uma sala pode projetar vários filmes, mas um filme só pode ser projetado numa sala

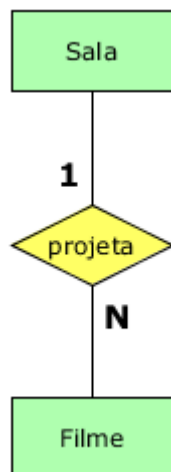


Figura 20 – Relacionamento entre as entidades Sala e Filme

O relacionamento entre a entidade Sala e a entidade Filme é de 1 para N, definindo-se que um filme apenas utiliza uma sala para ser projetado, mas uma sala pode projetar vários filmes.

14. Consultar a lista de lugares disponíveis para uma determinada sessão

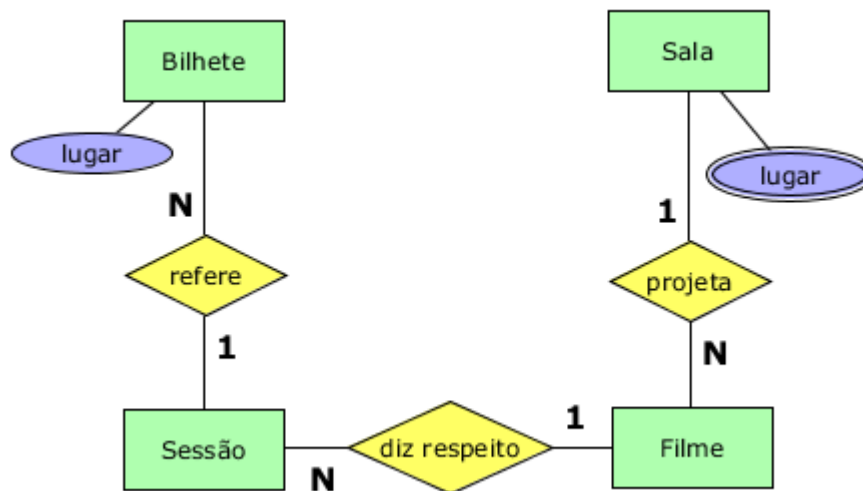


Figura 21 - Relacionamento entre as entidades Bilhete, Sessão, Filme e Sala

A informação de todos os lugares que vão sendo ocupados (para uma determinada sessão) através das reservas de bilhetes é guardada no atributo lugar da entidade Bilhete.

Uma vez que a entidade Sala tem como atributo a lista dos lugares que possui, é possível identificar os lugares que ainda se encontram disponíveis naquela sala que está associada à sessão para a qual se reservou o bilhete. Assim, os lugares disponíveis correspondem aos lugares que naquela sala não têm bilhete (lugar) associado.

15. Um filme tem uma data de estreia e uma data em que deixa de ser projetado

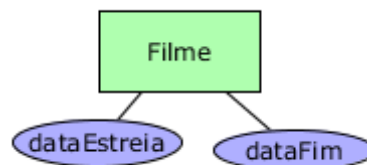


Figura 22 – Entidade Filme e alguns dos seus atributos

Os atributos dataEstreia e dataFim permitem que um filme tenha uma data de estreia e uma data em que deixa de ser projetado.

16. Deverá ser possível consultar dado um mês, o número de bilhetes vendidos e o total (€) para uma dada sessão ou filme

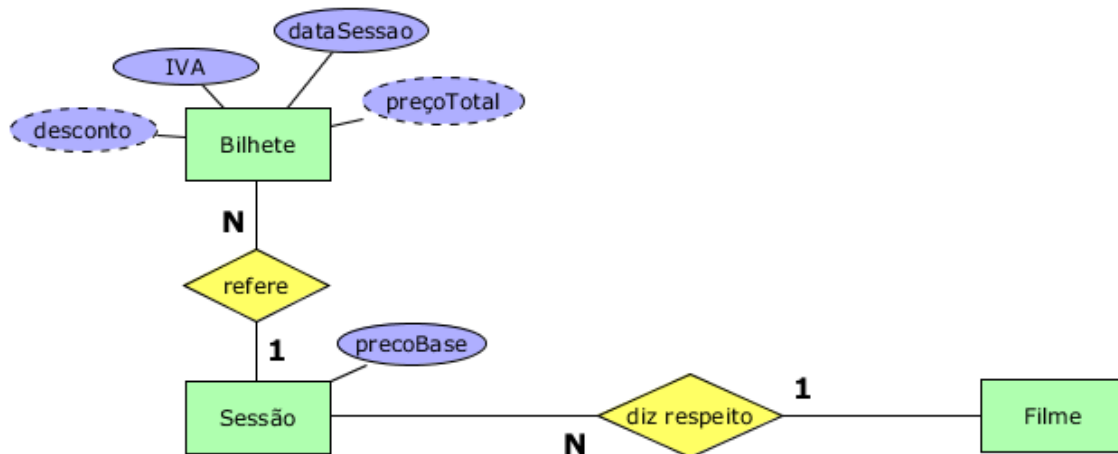


Figura 23 - Relacionamento entre Bilhete, Sessão e Filme

O relacionamento entre as entidades Bilhete e Sessão permite ter acesso à informação de todos os bilhetes associados à sessão. Uma vez que a entidade Bilhete possui os atributos dataSessão e preçoTotal é possível obter a informação de todos os bilhetes daquela sessão e o correspondente preço de cada uma. Assim, para um dado mês, os bilhetes podem ser quantificados e o valor total obtido através do somatório do preço total de todos os bilhetes daquele mês.

17. Dado um mês, deverá poder consultar-se o top 10 das sessões mais, e menos assistidas

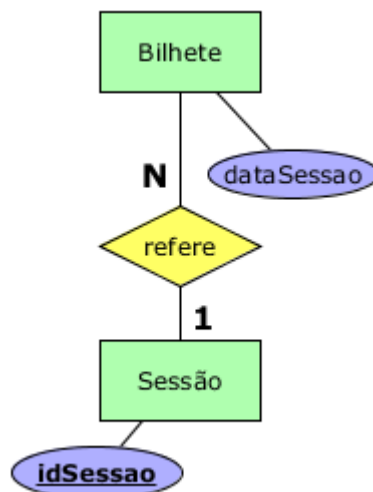


Figura 24 - Relacionamento entre as entidades Bilhete e Sessão e atributos relativos à data da sessão

O relacionamento entre bilhete e sessão permite obter informação das sessões que mais se assistem num determinado mês (ou noutro período de tempo). A sessão mais assistida corresponde àquela que terá mais bilhetes associados, todos estes bilhetes com a mesma data associada. O mesmo sucede para a sessão menos assistida, que tem associados menos bilhetes.

18. Deverá ser possível adicionar mais Salas e respetivos lugares ao Sistema

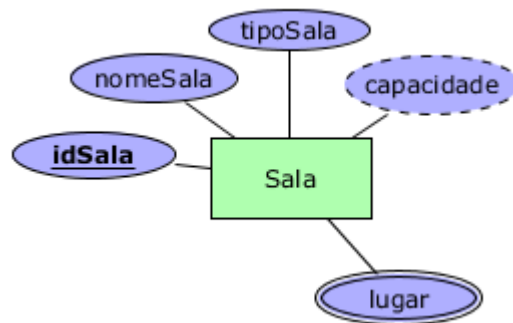


Figura 25 - Entidade Sala e seus atributos

A entidade Sala permite a adição de novas instâncias de Sala e seus lugares associados.

19. Deverá ser possível adicionar mais filmes, sessões e horários ao Sistema

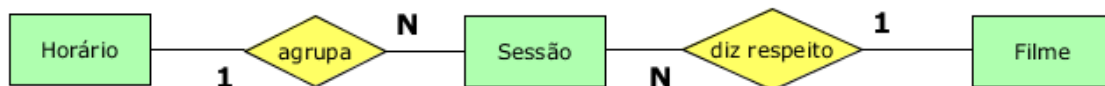


Figura 26 – Entidades Horário, Sessão e Filme

As entidades Horário, Sessão e Filme permitem a adição de novas instâncias de horários, sessões e filmes.

4.10. Modelo conceptual final

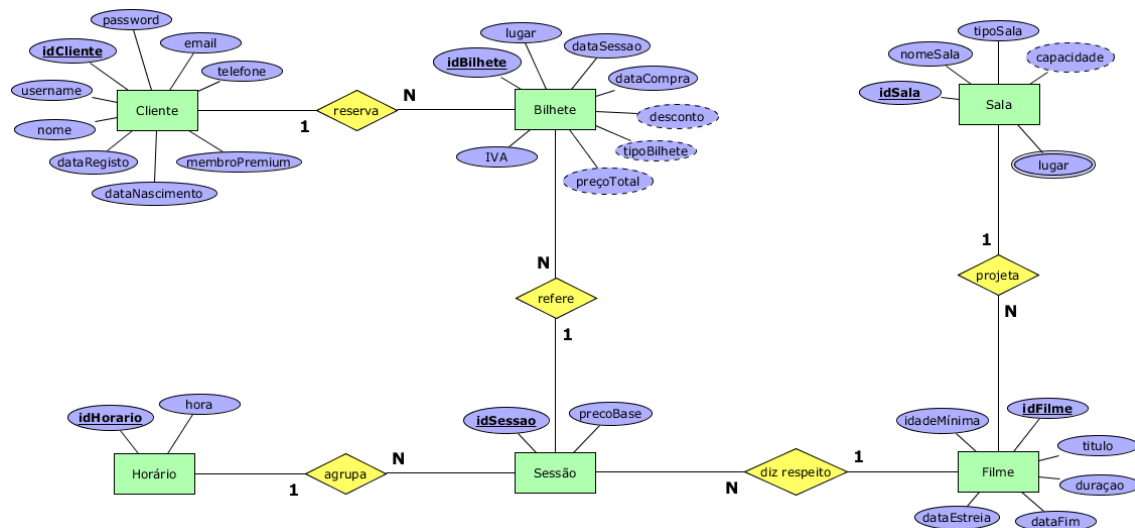


Figura 27 – Modelo Conceptual Final

5. Modelo Lógico

Neste capítulo apresenta-se a construção e validação do modelo lógico.

Na construção e validação do modelo lógico seguiram-se os vários passos especificados na metodologia seguida.

5.1. Derivação do Modelo Lógico

O modelo lógico é derivado do modelo conceptual aplicando as regras definidas para a construção das tabelas.

5.1.1 Derivação do modelo de dados lógico para obtenção de tabelas

O primeiro passo para a construção do modelo lógico consiste em derivar as tabelas a partir do modelo conceptual. De seguida, encontram-se as tabelas que se obteve, identificando as suas colunas, chave primária, chave estrangeira e observações.

- **Entidade Cliente => Tabela Cliente**

Cliente = {idCliente, username, email, password, nome, dataNascimento, telefone, dataRegisto, membroPremium}

Chave primária: idCliente

Chave estrangeira: Nenhuma

Observações: Relação de base.

Cliente é uma entidade no modelo conceptual.

- **Entidade Bilhete => Tabela Bilhete**

Bilhete = {idBilhete, tipoBilhete, dataCompra, dataSessao, lugar, preçoTotal, iva, desconto, idCliente, idSessao}

Chave primária: idBilhete

Chave estrangeira: 1) **idCliente** - referência à tabela Cliente (idCliente)

2) **idSessão** - referência à tabela Sessão (idSessão)

Observações: Relação de base. Bilhete é uma entidade no modelo conceptual

- **Entidade Sessão => Tabela Sessão**

Sessão = {idSessao, preçoBase, idHorário, idFilme}

Chave primária: idSessao

Chave estrangeira: 1) idHorário - referência à tabela Horário (idHorário)
2) idFilme - referência à tabela Filme (idFilme)

Observações: Relação de base.

Sessão é uma entidade no modelo conceptual

- **Entidade Horário => Tabela Horário**

Cliente = {idHorário, hora}

Chave primária: idHorário

Chave estrangeira: Nenhuma

Observações: Relação de base.

Horário é uma entidade no modelo conceptual.

- **Entidade Filme => Tabela Filme**

Filme = {idFilme, título, duração, dataEstreia, dataFim, idadeMínima, idSala}

Chave primária: idFilme

Chave estrangeira: idSala - referência à tabela Sala (idSala)

Observações: Relação de base.

Filme é uma entidade no modelo conceptual

- **Entidade Sala => Tabela Sala**

Sala = {idSala, nomeSala, tipoSala, capacidade}

Chave primária: idSala

Chave estrangeira: Nenhuma

Observações: Relação de base.

Sala é uma entidade no modelo conceptual

- **Atributo multivalor da entidade Sala => Tabela Lugar**

Lugar = {lugar, idSala}

Chave primária: lugar, idSala

Chave estrangeira: idSala - referência à tabela Sala (idSala)

Observações: Derivação do atributo multivalor “lugar” da entidade “Sala”

Obtém-se assim um esquema lógico composto por 5 tabelas/relações:

- 6 tabelas de base
- 1 tabela resultante da derivação do atributo multivalor lugar.

5.1.2 Seleção e justificação das chaves estrangeiras

As tabelas obtidas que necessitam de referenciar outras tabelas na base de dados necessitam de ter uma chave estrangeira. No ponto anterior identificou-se a chave estrangeira das tabelas que se obtiveram a partir do processo de derivação seguindo as regras estabelecidas. Segue-se a explicação de como se obtiveram e a justificação da seleção das chaves estrangeiras para cada uma das tabelas.

- **Tabela Cliente**

A tabela Cliente não tem chaves estrangeiras.

- **Tabela Bilhete:** Tem duas chaves estrangeiras:

- **idCliente:** Existe um relacionamento de 1 para N, entre as entidades Cliente e Bilhete. Pelo que se identifica a tabela “Bilhete” como tabela filho e a tabela “Cliente” como tabela pai. Assim, a tabela “Bilhete” funciona como filho e, portanto, recebe uma cópia da chave primária da tabela “Cliente” (*idCliente*). Desta forma, sempre que se precisar de mapear a que cliente pertence um bilhete utiliza-se a chave estrangeira *idCliente* que está associada ao bilhete para verificar a quem pertence na tabela Cliente.

- **idSessão:** Existe um relacionamento de 1 para N, entre as entidades Sessão e Bilhete. Identifica-se a tabela “Bilhete” como tabela filho e a tabela “Sessão” como tabela pai. Uma vez que, a tabela “Bilhete” funciona como filho então, recebe uma cópia da chave primária da tabela “Sessão” (*idSessão*). Portanto, a chave estrangeira permite referenciar a que sessão cada um dos bilhetes está associado.

- **Tabela Sessão:** Tem duas chaves estrangeiras:

- **idHorário:** Existe um relacionamento de 1 para N, entre as entidades Horário e Sessão. Pelo que se identifica a tabela “Sessão” como tabela filho e a tabela “Horário” como tabela pai. Assim, a tabela “Sessão” funciona como filho e, portanto, recebe uma cópia da chave primária da tabela “Horário” (*idHorário*). Desta forma, sempre que se precisar de mapear a que horário pertence uma sessão, utiliza-se a chave estrangeira *idHorário* que está associada à sessão.

- **idFilme:** Existe um relacionamento de 1 para N, entre as entidades Filme e Sessão. Identifica-se a tabela “Sessão” como tabela filho e a tabela “Filme” como tabela pai. Uma vez que, a tabela “Sessão” funciona como filho então, recebe uma cópia da chave primária da tabela “Filme” (*idFilme*). Portanto, a chave estrangeira permite referenciar a que filme cada uma das sessões está associada.

- **Tabela Filme:** tem uma chave estrangeira
 - **idSala:** Existe um relacionamento de 1 para N, entre as entidades Sala e Filme. Desta forma, a tabela “Filme” é a tabela filho e a tabela “Sala” é a tabela pai. Logo, a tabela “Filme” funciona como filho e recebe uma cópia da chave primária da tabela “Sala” (*idSala*). Assim, permite-se identificar, em cada filme, a sala que o projeta.
- **Tabela Sala**

A tabela Sala não tem chaves estrangeiras.
- **Tabela Horário**

A tabela Horário não tem chaves estrangeiras.
- **Tabela Lugar**

Esta tabela resulta da derivação do atributo multivalor “Lugar” da entidade “Sala”, contendo por isso uma cópia da chave primária da entidade “Sala”.

 - **idSala:** Esta tabela resulta da derivação do atributo multivalor “Lugar” da entidade “Sala”, contendo por isso uma cópia da chave primária da entidade “Sala” (*idSala*). Assim, para cada lugar é possível identificar a que Sala pertence.

5.2. Validação das relações segundo regras de Normalização

5.2.1 Primeira Forma Normal (1FN)

Para cada tabela do modelo lógico selecionou-se uma chave primária, garantindo-se assim que a interseção de cada linha e coluna contém uma forma (1FN) e apenas um valor. Concluindo-se, portanto, que o modelo lógico obtido se encontra na Primeira Forma Normal.

5.2.2 Segunda Forma Normal (2FN)

De forma a verificar que todas as relações se encontram na segunda forma normal, ter-se-á que mostrar que todos os atributos, que não sejam candidatos a chave primária, têm uma dependência funcional total em relação às chaves candidatas, eliminando-se assim dependências parciais. Para tal, realizou-se uma análise das dependências funcionais para cada relação:

1. Relação Cliente

Atributos da Relação: idCliente, username, email, password, nome, dataNascimento, telefone, dataRegisto, membroPremium

Dependências Funcionais:

- idCliente (chave primária) -> password, nome, dataNascimento, telefone, dataRegisto, membroPremium
- **username** (chave candidata) -> password, nome, dataNascimento, telefone, dataRegisto, membroPremium
- **email** (chave candidata) -> password, nome, dataNascimento, telefone, dataRegisto, membroPremium

2. Relação Bilhete

Atributos da Relação: idBilhete, lugar, tipoBilhete, dataCompra, dataSessao, IVA, preçoTotal, desconto, idCliente, idSessao

Dependências Funcionais:

- idBilhete (chave primária) -> lugar, tipoBilhete, dataCompra, dataSessao, IVA, preçoTotal, desconto, idCliente, idSessao

3. Relação Sessão

Atributos da Relação: idSessao, preçoBase, idHorário, idFilme

Dependências Funcionais:

- idSessao (chave primária) -> preçoBase, idHorário, idFilme

4. Relação Horário

Atributos da Relação: idHorario, hora

Dependências Funcionais:

- idHorario (chave primária) -> hora

5. Relação Filme

Atributos da Relação: idFilme, titulo, duração, dataEstreia, dataFim, idadeMinima, idSala

Dependências Funcionais:

- idFilme (chave primária) -> duração, dataEstreia, dataFim, idadeMinima, idSala
- titulo (chave candidata) -> duração, dataEstreia, dataFim, idadeMinima, idSala

6. Relação Sala

Atributos da Relação: idSala, nomeSala, tipoSala, capacidade

Dependências Funcionais:

- idSala (chave primária) -> tipoSala, capacidade
- nomeSala (chave candidata) -> tipoSala, capacidade

7. Relação Lugar

Atributos da Relação: lugar, idSala

A junção de todos os atributos da relação constitui a chave primária da tabela Lugar, logo é garantida automaticamente a segunda forma normal.

Conclui-se, portanto, que todas as relações respeitam a segunda forma normal.

5.2.3 Terceira Forma Normal (3FN)

No passo anterior da normalização mostrou-se que as relações estão na 2FN. Com base nas dependências funcionais apresentadas anteriormente, conclui-se que nenhum dos atributos não chave primária depende de outro também não chave primária. Pelo que, não existem dependências transitivas e por isso o modelo lógico encontra-se na terceira forma normal.

5.3. Validar modelo segundo as transações do utilizador

Uma transação é uma sequência de operações executadas como se fossem uma só, que alteram o conteúdo da base de dados.

5.3.1 Adicionar um novo cliente

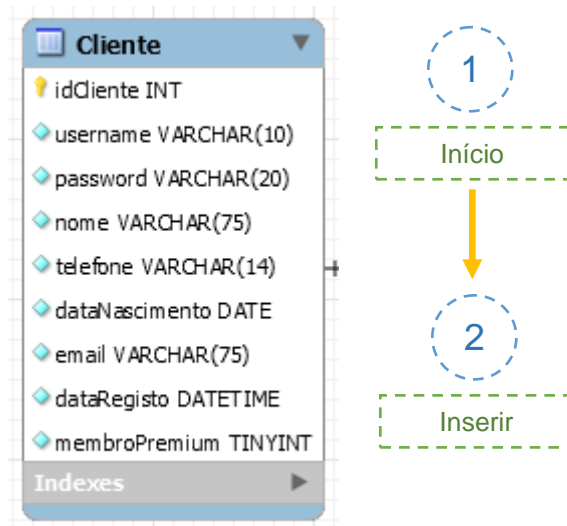


Figura 28 – Mapa da transação “Adicionar um novo cliente”

Ao se adicionar um novo cliente, se este ainda não existe no sistema, são adicionadas as informações relativas à entidade Cliente: username, password, nome, email, telefone, dataNascimento, dataRegisto e *membroPremium* (que é falso, por defeito). A tabela Cliente fica assim com um novo registo.

5.3.2 Adicionar um novo filme

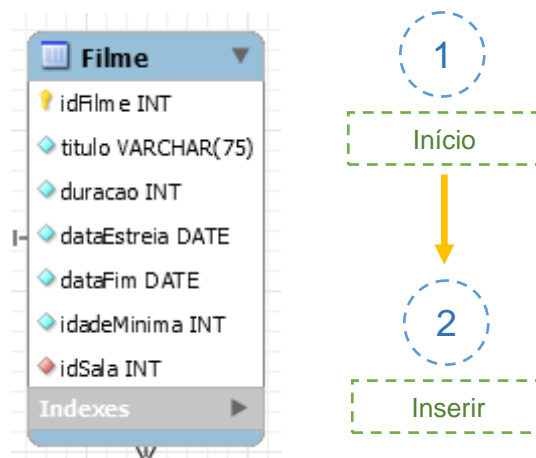


Figura 29 – Mapa da transação “Adicionar um novo filme”

Ao se adicionar um novo filme, se esta ainda não existe no sistema, é adicionada a sua identificação - o atributo chave primária **idFilme** - e os restantes atributos da entidade Filme: título, duração, dataEstreia, dataFim e idadeMínima.

5.3.3 Adicionar uma nova sala

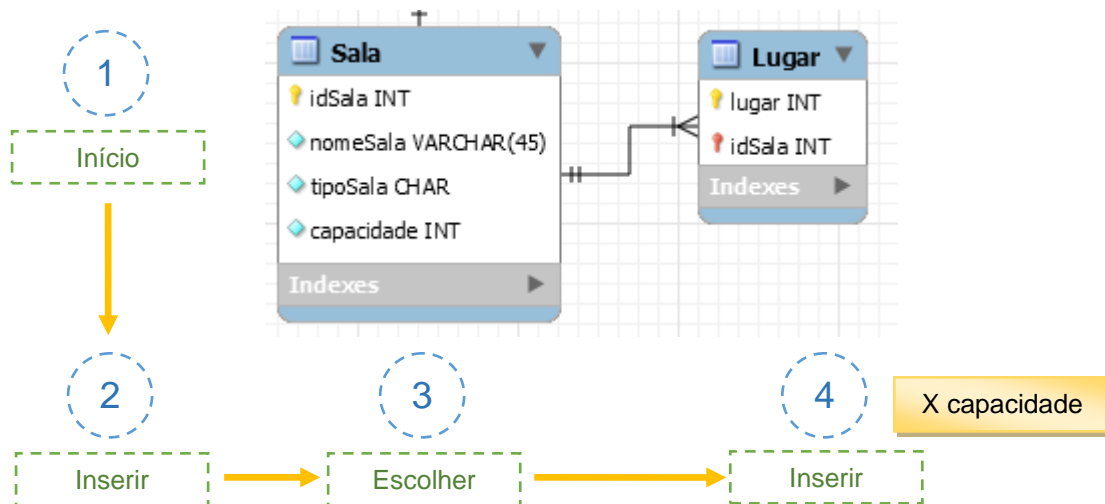


Figura 30 – Mapa da transação “Adicionar uma nova sala”

Ao se adicionar uma nova sala, este só é adicionado se ainda não estiver registado na base de dados. Assim, é adicionada a sua identificação atributo chave primária **idSala** e a sua restante informação: nomeSala, tipoSala, capacidade e o atributo multivalor lugar.

O atributo lugar permite adicionar a informação sobre os lugares da sala, adicionando-se então essa informação um determinado número de vezes: a capacidade da sala.

5.3.4 Fazer uma reserva/compra de bilhete

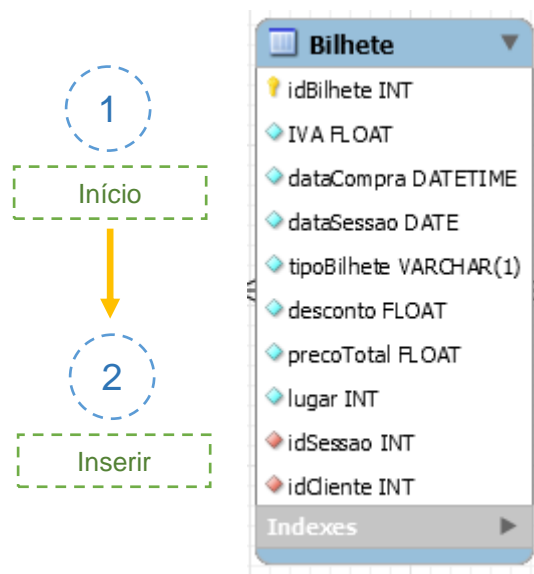


Figura 31 – Mapa da transação “Fazer uma reserva/compra de bilhete”

Para se efetuar a reserva/compra de um bilhete, é necessário verificar se esta pode ser efetuada, ou seja, observar se já existe um bilhete com a mesma **idSessao** e **lugar** na sala que aquele que o cliente pretende. Se ainda não existir um bilhete com essas duas características únicas (sessão e lugar específico na sala), aí sim será adicionado um bilhete, com os atributos **idBilhete**, tipoBilhete, data da compra, data da sessão, lugar, preço total, IVA e desconto.

5.4. Verificação das restrições de integridade

5.4.1 Dados requeridos

Alguns atributos devem ter um valor válido, isto é não devem permitir valores nulos. Estas restrições foram identificadas na documentação do dicionário de dados dos atributos (consultar anexo III).

5.4.2 Integridade de domínio

Os atributos têm um domínio, isto é um conjunto de valores permitido. Todos atributos respeitam o domínio do seu tipo de dados. Existem ainda atributos que têm o seu domínio de dados ainda mais restrito, sendo de seguida identificados.

1. O atributo telefone, da relação Cliente, é do tipo VARCHAR com tamanho 14, uma vez que os números telefónicos em Portugal têm este comprimento. Os cinco primeiros caracteres são +0351 e os restantes o número telefónico:
 - +0351 XXX XXX XXX
2. O atributo tipoBilhete, da relação Bilhete, foi definido como um VARCHAR(1) que pode ser:
 - P - bilhete do tipo promocional
 - N - bilhete do tipo normal
3. O atributo lugar, da relação Bilhete, foi definido como INT, consoante a sua capacidade do tipo de sala.
 - *Basic* (15): 1 - 15
 - *Silver* (20): 1 - 20
 - *Platinum* (25): 1 - 25

4. O atributo IVA, da relação Bilhete, é do tipo FLOAT e foi definido pelo valor 0.13, isto é, um IVA de 13%.
5. O atributo tipoSala, da relação Sala, foi definido como um CHAR que pode ser:
 - B - sala do tipo *Basic*
 - S - sala do tipo *Silver*
 - P - sala do tipo *Platinum*
6. O atributo capacidade, da relação Sala, é do tipo INT e suporta três valores:
 - 15 - para salas do tipo *Basic*
 - 20 - para salas do tipo *Silver*
 - 25 - para salas do tipo *Platinum*

5.4.3 Restrições de multiplicidade

As restrições de multiplicidade foram impostas quando se decidiu os relacionamentos entre as entidades, que podem ser consultadas no dicionário de dados, na tabela dos relacionamentos (consultar anexo II).

5.4.4 Integridade de entidade

O valor das chaves primárias não pode ser nulo, cumprindo assim a integridade de entidade. Esta restrição foi garantida pelo SGBD, que não permite que o atributo chave primária seja nulo, isto é, cada tuplo de uma relação deve ter um valor para o atributo da chave primária. Esta restrição foi considerada na identificação das chaves primárias de cada entidade.

5.4.5 Integridade referencial

A integridade referencial garante que uma chave estrangeira contem um valor, e este valor deve referir um tuplo existente na relação pai. Isto é, a chave estrangeira da tabela “filho” têm de coincidir com a chave primária da sua tabela “pai”. Por exemplo. a chave estrangeira da relação Filme é idSala, que por sua vez é a chave primária na relação Sala.

5.4.6 Restrições Gerais | Regras de Negócio

Os valores dos atributos das entidades da base de dados podem ser controlados por restrições que controlam as transações no “mundo real”, isto é, restrições gerais, muitas vezes arbitrárias, impostas pela definição do domínio do problema. Assim, foram identificadas as seguintes restrições gerais:

- **Um bilhete é promocional se o cliente tem menos de 25 anos ou se tem mais de 65 anos, ou se o utilizador é membro *premium*.**

Um cliente poderá usufruir de um desconto jovem se tiver menos de 25 anos, de um desconto idosos se tiver uma idade superior a 65 anos, ou de um desconto de membro se for membro *premium*.

- **O número de bilhetes vendidos/reservados não pode ultrapassar a capacidade da sala associada à sessão de um filme.**

As sessões de cinema que a empresa BCC oferece requerem a reserva de lugares sentados. Por isso, não devem ser vendidos mais bilhetes do que o número de lugares existentes na sala.

5.5. Revisão do modelo lógico com o utilizador

A metodologia seguida é iterativa e prevê o constante envolvimento e validação dos modelos obtidos com o utilizador. Neste passo, após se obter o modelo lógico é importante validá-lo com o utilizador para se compreender se o modelo garante os requisitos pretendidos.

5.6. Fusão dos modelos lógicos num modelo global

Uma vez que a base de dados que se está a desenvolver apenas é constituído por um modelo lógico então não existe a necessidade de fundir os modelos lógicos num modelo global.

O único modelo lógico local corresponde ao modelo global.

5.7. Analisar Crescimento Futuro

O modelo lógico foi desenvolvido de acordo com os requisitos recolhidos e foi elaborado para lhes dar resposta.

Houve uma preocupação ao longo do desenvolvimento do modelo lógico para suportar futuras alterações que pudessem surgir no sistema por necessidade da empresa ou do utilizador com o mínimo impacto no modelo.

Para além dos requisitos atuais, o modelo permite que sejam adicionadas mais sessões, filmes, salas e horários. Para cada sala, pode ser definido um novo tipo, para além das existentes, bem como uma capacidade de lugares diferente das existentes.

O IVA associado ao bilhete também pode ser facilmente alterado, permitindo assim uma adaptação a possíveis alterações nas condições económicas.

Apesar de se ter desenvolvido um modelo com alguns pontos de extensão para possíveis futuras alterações, a preocupação foi em responder às atuais necessidades. Pôs-se de parte fazer uma solução genérica que pudesse comportar outras alterações, uma vez que um modelo tão genérico traria impactos na implementação e consequente utilização.

5.8. Modelo Lógico Final

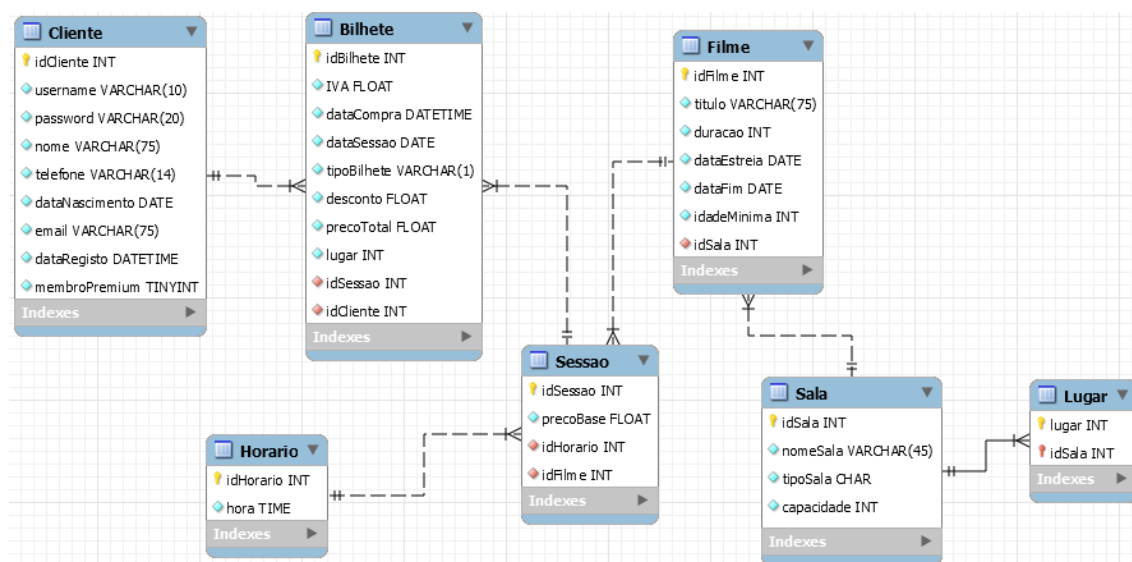


Figura 32 – Modelo Lógico Final

6. Modelo Físico

Neste capítulo apresenta-se a implementação do modelo de dados físico feita a partir do modelo de dados lógico (onde as relações foram descritas utilizando DBDL) e da documentação que foi gerada durante o desenvolvimento da base de dados.

O modelo de dados físico é dependente do SGBD utilizado. Portanto, o primeiro passo foi selecionar o SGBD. Desde o início da construção da base de dados relacional que se decidiu utilizar o MySQL uma vez que é um sistema seguro (apresenta alta proteção de dados) e que se obtém facilmente, sendo bastante fácil de compreender, manter e gerir. Para além das características enumeradas, é também um sistema que consegue dar resposta às necessidades requeridas, nomeadamente:

- suporta a criação de chaves primárias únicas,
- admite a definição de chave estrangeira e chaves alternativas
- permite definir atributos como o valor de “não nulo” (NOT NULL)
- proporcionar a definição de domínios
- possibilita restrições de integridade relacional
- suporta a definição de restrições de integridade
- oferece um sistema robusto para ambientes de transações
- permite definir vistas de utilizadores

6.1. Representação das Relações Base

O primeiro passo da conceção do modelo físico envolve a tradução das relações do modelo de dados lógico, de forma a que possa ser implementado no DBMS relacional, MySQL.

O modelo lógico foi esquematizado no Workbench do MySQL, uma ferramenta visual unificada para modelar, estruturar e desenvolver uma base de dados MySQL. Por isso, o processo de criação das tabelas foi automático e realizado simultaneamente aquando a elaboração do esquema lógico.

Nesta etapa, a partir da informação definida em etapas anteriores, foi especificada informação sobre as entidades, domínio dos atributos, definição de chaves primárias e estrangeiras, obtendo-se as respetivas tabelas que suportam o esquema físico.

Além disso, de forma a manter a integridade e consistência de dados, foram implementadas as respetivas restrições definidas anteriormente.

Para implementar restrições é frequente utilizar-se a instrução CHECK, especificando-se uma expressão que tem de ser avaliada como verdadeira para satisfazer as restrições definidas. Contudo, o MySQL apresenta limitações ao utilizar a instrução CHECK, mas possibilita a implementação destas restrições através de outros métodos.

De seguida apresenta-se a definição da criação de cada uma das tabelas existente no sistema. Em Anexo IV e V, podem ser consultados os *scripts* obtidos através da definição destas tabelas e de restrições adicionadas.

Legenda de apoio às Figuras:

- **PK** – Primary Key (chave primária)
- **NN** – Not Null (o atributo não pode ser nulo)
- **UQ** – Unique (o atributo deve ser único)
- **B** – Binary
- **UN** – Unsigned
- **ZF** – Zero-Filled
- **AI** – Auto Incremental
- **G** – Generated Column Default/Expression (valor por defeito)

6.1.1 Tabela Cliente

- **Tabela Cliente** = {idCliente, username, password, nome, telefone, dataNascimento, email, dataRegisto, membroPremium}










Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
 idCliente	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
 username	VARCHAR(10)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 password	VARCHAR(20)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 nome	VARCHAR(75)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 telefone	VARCHAR(14)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 dataNascimento	DATE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 email	VARCHAR(75)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 dataRegisto	DATETIME	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NOW()
 membroPremium	TINYINT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0

Figura 33 – Tabela Cliente: representa a entidade Cliente

O atributo telefone é do tipo VARCHAR(14) em que, como definido no dicionário de dados, os 5 primeiros caracteres são '+0351'.

Para garantir esta restrição foi implementado o TRIGGER TR_Cliente_BI que é executado antes da inserção de um cliente e garante que esta restrição é cumprida.

```

DELIMITER $$
CREATE TRIGGER TR_Cliente_BI
  BEFORE INSERT ON Cliente
  FOR EACH ROW
BEGIN

  IF (NEW.telefone NOT LIKE "+0351%") THEN
    SIGNAL SQLSTATE '45000';
  END IF;

  IF (CHAR_LENGTH(NEW.telefone) <> 14) THEN
    SIGNAL SQLSTATE '45000';
  END IF;
END $$

```

O valor *default* do atributo *dataRegistro*, que armazena a informação da data em que o cliente se registou no sistema, foi definido através da função *NOW()* que devolve a data atual no momento do registo.

O valor *default* do atributo *membroPremium*, que informa se um cliente é ou não membro premium da BCC, foi definido como 0 (falso).

6.1.2 Tabela Bilhete

- **Tabela Bilhete** = {*idBilhete*, IVA, *dataCompra*, *dataSessao*, *tipoBilhete*, *desconto*, *preçoTotal*, *lugar*, *idCliente*, *idSessao*}











Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
 <i>idBilhete</i>	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
 IVA	FLOAT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0.13
 <i>dataCompra</i>	DATETIME	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NOW()
 <i>dataSessao</i>	DATE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 <i>tipoBilhete</i>	VARCHAR(1)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 <i>desconto</i>	FLOAT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 <i>preçoTotal</i>	FLOAT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 <i>lugar</i>	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 <i>idSessao</i>	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 <i>idCliente</i>	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Figura 34 – Tabela Bilhete: representa a entidade Bilhete

O atributo *data* da compra do bilhete (*dataCompra*) representa o momento em que o cliente comprou o bilhete e, portanto, foi definido como valor *default* a função *NOW()* que vai registar o momento em que o bilhete foi inserido na base de dados, o que corresponde ao momento em que foi reservado/comprado.

O atributo IVA consiste num valor percentual a acrescentar a um preço base. Este valor é atualmente de 0.13 e não depende de cada bilhete e por isso, foi definido como valor *default*.

O atributo chave primária é do tipo INT sendo automaticamente incrementado após a inserção de um registo de um bilhete, permitindo identificar univocamente o bilhete. No entanto, para garantir que um bilhete é único, ou seja, o conjunto de atributos: idSessao, dataSessao e lugar, deve ser único para todos os bilhetes.

Index Name	Type	Index Columns			
PRIMARY	PRIMARY	Column	#	Order	Length
idBilhete_UNIQUE	UNIQUE	<input type="checkbox"/> idBilhete		ASC	
fk_Bilhete_Sessao1_idx	INDEX	<input type="checkbox"/> IVA		ASC	
fk_Bilhete_Cliente1_idx	INDEX	<input type="checkbox"/> dataCompra		ASC	
		<input checked="" type="checkbox"/> dataSessao	3	ASC	
		<input type="checkbox"/> tipoBilhete		ASC	
		<input type="checkbox"/> desconto		ASC	
		<input type="checkbox"/> precoTotal		ASC	
		<input checked="" type="checkbox"/> lugar	2	ASC	
		<input checked="" type="checkbox"/> idSessao	1	ASC	
		<input type="checkbox"/> idCliente		ASC	

6.1.3 Tabela Sessão

- **Tabela Sessão** = {idSessão, preçoBase, idHorário, idFilme}





Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
 idSessao	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
 precoBase	FLOAT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 idHorario	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 idFilme	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Figura 35 – Tabela Sessão: representa a entidade Sessão

O atributo chave primária é do tipo INT sendo automaticamente incrementado após a inserção de uma nova sessão.

Com o objetivo de garantir que não existem duas sessões iguais na base de dados, isto é, o mesmo filme projetado à mesma hora, definiu-se, através da restrição idSessão_UNIQUE, que o conjunto de atributos: idFilme e idHorário, deve ser único para todas as sessões.

Index Name	Type	Index Columns			
PRIMARY	PRIMARY	Column	#	Order	Length
idSessao_UNIQUE	UNIQUE	<input type="checkbox"/> idSessao		ASC	
fk_Sessao_Horario1_idx	INDEX	<input type="checkbox"/> precoBase		ASC	
fk_Sessao_Filme1_idx	INDEX	<input checked="" type="checkbox"/> idHorario	1	ASC	
		<input checked="" type="checkbox"/> idFilme	2	ASC	

6.1.4 Tabela Filme

- **Tabela Filme** = {idFilme, título, duração, dataEstreia, dataFim, idadeMínima, idSala}

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
idFilme	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
título	VARCHAR(75)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
duracao	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
dataEstreia	DATE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
dataFim	DATE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
idadeMinima	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
idSala	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Figura 36 – Tabela Filme: representa a entidade Filme

O atributo chave primária é do tipo INT sendo automaticamente incrementado após a inserção de um novo filme. Com o objetivo de garantir que não existem dois filmes iguais, o atributo título deve ser único.

6.1.5 Tabela Sala

- **Tabela Sala** = {idSala, nomeSala, tipoSala, capacidade}

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
idSala	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
nomeSala	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
tipoSala	CHAR	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
capacidade	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Figura 37 – Tabela Sala: representa a entidade Sala

O atributo chave primária é do tipo INT sendo automaticamente incrementado quando se adiciona uma nova sala.

O atributo nomeSala permite identificar uma sala pelo seu nome. Este atributo foi definido como único, garantindo-se que não existe a mesma sala repetida na base de dados.

O atributo que define o tipo de sala (tipoSala) é do tipo CHAR e deve ser apenas B (*Basic*), S (*Silver*) ou P (*Platinum*).

Para garantir estas restrições foi definido o TRIGGER TR_Sala_BI que é executado antes de se inserir uma sala, validando os respetivos dados.

```

DELIMITER $$
CREATE TRIGGER TR_Sala_BI
  BEFORE INSERT ON Sala
  FOR EACH ROW
BEGIN
  IF (NEW.tipoSala <> 'B' AND NEW.tipoSala <> 'S' AND NEW.tipoSala <> 'P') THEN
    SIGNAL SQLSTATE '45000';
  END IF;
END $$

```

6.1.6 Tabela Lugar

- **Tabela Lugar** = {lugar, idSala}

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
🔦 lugar	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
🔦 idSala	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Figura 38 – Tabela Lugar: representa o atributo multivalor lugar

Um determinado lugar é identificado pela chave primária composta: lugar e idSala.

Para garantir a consistência de dados foi necessário garantir que:

- O atributo lugar tem um valor inferior à capacidade da sala.
- Não são inseridos mais lugares do que a capacidade da sala.

Para tal, implementou-se o TRIGGER TR_lugar_BI que valida se as restrições se cumprem antes de um novo lugar ser inserido.

```
DELIMITER $$
CREATE TRIGGER TR_lugar_BI
  BEFORE INSERT ON lugar
  FOR EACH ROW
BEGIN
  DECLARE nrLugaresExistentes INT;
  DECLARE capacidade INT;

  SET capacidade = (SELECT capacidade
                    FROM Sala
                    WHERE idSala = NEW.idSala);

  -- VALIDACAO: número de lugar é inferior à capacidade da sala
  IF (NEW.lugar > capacidade) THEN
    SIGNAL SQLSTATE '45000';
  END IF;

  SET nrLugaresExistentes = (SELECT COUNT(*)
                             FROM lugar
                             WHERE idSala = NEW.idSala);

  -- VALIDACAO: número de lugares de um comboio não excede a sua capacidade máxima
  IF nrLugaresExistentes >= capacidade THEN
    SIGNAL SQLSTATE '45000';
  END IF;
END $$
```

6.2. Representação dos atributos derivados

Esta etapa do modelo físico visa decidir como se representar os dados derivados no modelo lógico de dados da DBMS. Ou seja, definir como implementar o cálculo dos atributos derivados a partir do valor de outros atributos no modelo.

No modelo conceptual determinou-se os atributos derivados da entidade Bilhete: desconto, preçoTotal e tipoBilhete. De seguida, apresenta-se como se implementou cada um destes atributos derivados.

6.2.1 Atributo derivado desconto

O atributo derivado desconto é calculado segundo a idade do cliente, ou se este é membro *premium*.

Definiu-se o TRIGGER TR_Bilhete_Desconto_BI que permite definir este atributo, sendo executado antes de se inserir um novo bilhete. No TRIGGER, calcula-se a idade através da função INT calculaIdade(dataNascimento DATE) e verifica-se se o cliente é membro *premium* aplica-se o desconto correspondente.

```
DELIMITER $$
CREATE TRIGGER TR_Bilhete_Desconto_BI
  BEFORE INSERT ON Bilhete
  FOR EACH ROW
  BEGIN
    DECLARE idade INT;
    DECLARE membro BOOLEAN;

    SET idade = (SELECT calculaIdade(dataNascimento)
                  FROM Cliente
                  WHERE idCliente = NEW.idCliente);

    SET membro = (SELECT membroPremium
                   FROM Cliente
                   WHERE idCliente = NEW.idCliente);

    IF (idade < 25 ) THEN
      SET NEW.desconto = 0.15;
    ELSEIF (idade > 65) THEN
      SET NEW.desconto = 0.25;
    ELSE
      SET NEW.desconto = 0;
    END IF;

    IF (membro=1) THEN
      SET NEW.desconto = 0.30;
    END IF;
  END
END $$
```

6.2.2 Atributo derivado preçoTotal

Para o cálculo do atributo derivado preçoTotal foi definido o TRIGGER TR_Bilhete_PrecoTotal_BI que é executado após o TRIGGER TR_Bilhete_Desconto_BI sempre que se insere um novo bilhete.

```
DELIMITER $$
CREATE TRIGGER TR_Bilhete_PrecoTotal_BI
  BEFORE INSERT ON Bilhete
  FOR EACH ROW FOLLOWS TR_Bilhete_Desconto_BI
BEGIN
  DECLARE precoB FLOAT;
  DECLARE precoComIVA FLOAT;
  DECLARE precoComDesconto FLOAT;

  SET precoB = (SELECT `precoBase`
                FROM idSessao
                WHERE idSessao = NEW.idSessao);

  SET precoComIVA = precoB * (1 + NEW.IVA);

  SET precoComDesconto = precoComIVA * (1 - NEW.desconto);

  SET NEW.`precoTotal` = ROUND(precoComDesconto,2);

END $$
```

6.2.3 Atributo derivado tipoBilhete

O atributo tipo de bilhete é definido através do atributo desconto. Uma vez que o atributo desconto pertence à mesma tabela que o atributo que se pretende derivar, utilizou-se uma *generated column* (definida aquando a criação da tabela Bilhete). Esta coluna define a expressão que é utilizada para derivar o atributo tipoBilhete (P ou N).

```
`tipoBilhete` VARCHAR(1) AS (CASE WHEN desconto > 0 THEN 'P' ELSE 'N' END) STORED NOT NULL,
```

6.2.4 Atributo derivado capacidade

A tabela Sala apresenta como atributo derivado a capacidade da Sala (*capacidade*). Este atributo é definido através do atributo tipoSala, uma vez que cada tipo de sala tem a correspondente capacidade de lugares. Para garantir a consistência de dados entre estes dois atributos, definiu-se uma *generated column* (definida aquando a criação da tabela Sala).

```
`capacidade` INT AS (CASE WHEN tipoSala = 'B' THEN 15 WHEN tipoSala = 'S' THEN 20 ELSE 25 END) STORED NOT NULL,
```

6.3. Restrições Gerais

A implementação das restrições gerais depende da escolha da SGBD.

As restrições do modelo foram implementadas por via de TRIGGERS, dado o suporte limitado do MySQL à instrução CHECK da linguagem SQL, que impede que algumas das verificações apresentadas pudessem estar associadas às tabelas propriamente ditas.

De seguida, apresentam-se as implementações das restrições gerais que foram definidas no modelo lógico:

- **Um bilhete é promocional se o cliente tem menos de 25 anos ou se tem mais de 65 anos, ou se o cliente é membro *premium*.**

Um cliente poderá usufruir de um desconto jovem se tiver menos de 25 anos, de um desconto idosos se tiver uma idade superior a 65 anos, ou de um desconto de membro se for membro *premium*.

- **O número de bilhetes vendidos/reservados não pode ultrapassar a capacidade da sala associada à sessão de um filme**

A restrição de negócio que especifica que o número de bilhetes vendidos para uma sessão não pode ser superior ao número de lugares da respetiva sala foi implementada através do TRIGGER TR_Bilhete_BI. Antes de inserir um bilhete, este TRIGGER contabiliza o número de bilhetes reservados para a sessão associada e verifica se ainda existem lugares disponíveis.

```
DELIMITER $$
CREATE TRIGGER TR_Bilhete_BI
  BEFORE INSERT ON Bilhete
  FOR EACH ROW
  BEGIN
    DECLARE nrBilhetes INT;
    DECLARE nrBilhetesMax INT;

    SET nrBilhetes = nrBilhetesParaSessaoData(NEW.idSessao, NEW.dataSessao);

    SET nrBilhetesMax = (SELECT Sala.capacidade
                        FROM Sessao
                        JOIN Filme ON Sessao.idFilme = Filme.idFilme
                        JOIN Sala ON Filme.idSala = Sala.idSala
                        WHERE Sessao.idSessao = NEW.idSessao);

    IF nrBilhetes >= nrBilhetesMax THEN
      SIGNAL SQLSTATE '45000';
    END IF;
  END
END $$
```

6.4. Análise de Transações

6.4.1 Adicionar um novo cliente

```
-- Início da transação.
START TRANSACTION;
-- 1ª Operação
INSERT INTO cliente
    (username, password, telefone, dataNascimento, email, nome)

VALUES
    ('luciaabreu17', 'lucia153', '+0351253193212', '1944-03-21', 'luciaabreu@gmail.com', 'Lúcia Abreu');
```

Para se adicionar um novo utilizador, faz-se um INSERT na tabela Cliente dos seguintes dados: *username*, *password*, *telefone*, *dataNascimento*, *email*, *nome*.

Como *idCliente* é chave primária, a própria base de dados incrementa esse atributo na base de dados.

O atributo *membroPremium* é atribuído por *default*.

6.4.2 Adicionar um novo filme

```
-- Início da transação.
START TRANSACTION;
-- 1ª Operação
INSERT INTO filme
    (titulo, duracao, dataEstreia, dataFim, idadeMinima, idSala)

VALUES
    ('The House By The Sea', 127, '2016-07-15', '2016-09-17', 16, 1);
```

Para se adicionar um novo filme, faz-se um INSERT na tabela Filme dos seguintes dados: *titulo*, *duracao*, *dataEstreia*, *dataFim*, *idadeMínima* e *idSala*. O atributo *idFilme* é incrementado automaticamente e o atributo *idSala* pode ser nulo, ou seja, pode optar-se por não o introduzir.

6.4.3 Fazer uma reserva/compra de bilhete

```
-- Início da transação.
START TRANSACTION;
-- 1ª Operação
INSERT INTO bilhete
    (lugar, dataSessao, dataCompra, idCliente, idSessao)

VALUES('12', '2016-07-15', '2016-05-15', 3, 1);
```

Para se efetuar a reserva/compra de um bilhete, faz-se um INSERT na tabela Bilhete dos seguintes dados: *lugar*, *dataSessao*, *dataCompra*, *idCliente*, *idSessao*. Não é necessário verificar se esta pode ser efetuada, as restrições adicionadas à base de dados assim o garantem.

6.4.4 Adicionar uma nova sala

```
DROP PROCEDURE IF EXISTS spAdicionarSala;
DELIMITER $$
CREATE PROCEDURE spAdicionarSala
    (IN tipoSala CHAR, nomeSala VARCHAR(45))
BEGIN
    DECLARE a INT;
    DECLARE idS INT;
    DECLARE cap INT;

    -- Declaração de um handler para tratamento de erros.
    DECLARE ErroTransacao BOOL DEFAULT 0;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET ErroTransacao = 1;

    -- Início da transação.
    START TRANSACTION;

    -- 1ª Operação
    INSERT INTO Sala
        (tipoSala, nomeSala)
        VALUES(tipoSala, nomeSala);

    SELECT idSala, capacidade INTO idS, cap
    FROM Sala
        WHERE nomeSala = nSala;

    -- Inserir lugares
    SET a = 1;
    WHILE a <= cap DO
        INSERT INTO lugar
            (lugar, idSala)
            VALUES(a, idS);
        SET a = a+1;
    END WHILE;

    -- Verificação da ocorrência de um erro.
    IF ErroTransacao THEN
        -- Desfazer as operações realizadas.
        ROLLBACK;
    ELSE
        -- Confirmar as operações realizadas.
        COMMIT;
    END IF;
END $$
```

Para se adicionar uma nova sala, faz-se um INSERT na tabela Sala dos seguintes dados: nomeSala e tipoSala. O atributo idSala é incrementado automaticamente e o atributo capacidade é derivado do tipoSala.

Depois faz-se um determinado número de INSERT (igual à capacidade da sala) na tabela Lugar para adicionar os lugares da sala à base de dados.

6.5. Definição das vistas dos utilizadores e regras de acesso

6.5.1 Vistas dos utilizadores

Durante a aplicação da metodologia para a futura implementação da base de dados para reserva de bilhetes de cinema, não foram identificadas vistas de utilizador. Contudo, neste passo do desenvolvimento do modelo físico, pode considerar-se dois tipos de utilizadores da futura base de dados: os clientes e os funcionários. De seguida ir-se-á apresentar o código SQL, que criará uma vista destes utilizadores sobre a base de dados.

1. Vista do Cliente

A seguinte vista permite ao cliente consultar os filmes disponíveis, a sua duração, a idade mínima, a hora de início, o número de lugares e o preço total.

```
DROP VIEW IF EXISTS vwClientes;
CREATE VIEW vwClientes AS
    SELECT titulo      AS 'Titulo',
           duracao     AS 'Duração',
           idadeMinima AS 'Idade Mínima',
           hora        AS 'Hora de Início',
           count(*)    AS 'NrLugares',
           precoTotal  AS 'Preço Total'
    FROM Bilhete
    JOIN Sessao ON Bilhete.idSessao = Sessao.idSessao
    JOIN Horario ON Sessao.idHorario = Horario.idHorario
    JOIN Filme ON Sessao.idFilme = Filme.idFilme
    INNER JOIN Sala ON Filme.idSala = Sala.idSala
    INNER JOIN lugar ON Sala.idSala = lugar.idSala

    GROUP BY titulo
```

De seguida mostra-se um exemplo do acesso a esta vista:

```
SELECT * FROM vwClientes;
```

	Titulo	Duração	Idade Mínima	Hora de Início	NrLugares	Preço Total
	The House By The Sea	127	16	09:00:00	30	0
	The Order Of Things	117	12	11:30:00	40	0

Figura 39 – Tabela que representa uma vista do Cliente

2. Vista do Funcionário

A seguinte vista permite ao funcionário consultar :

```
DROP VIEW IF EXISTS vwFuncionarios;  
CREATE VIEW vwFuncionarios AS  
    SELECT DISTINCT nomeSala AS 'Nome da Sala',  
        tipoSala AS 'Tipo da Sala',  
        capacidade AS 'Capacidade',  
        titulo AS 'Filme',  
        idadeMinima AS 'Idade Mínima',  
        duracao AS 'Duração'  
FROM Filme  
LEFT JOIN Sala  
    ON Filme.idSala = Sala.idSala;
```

De seguida mostra-se um exemplo do acesso a esta vista:

```
SELECT * FROM vwFuncionario;
```

	Nome da Sala	Tipo da Sala	Capacidade	Filme	Idade Mínima	Duração
	Martin Scorsese	B	15	The House Bv The Sea	16	127
	Steven Spielbera	S	20	The Order Of Things	12	117
	Woodv Allen	P	25	The Phantom Thread	12	117

Figura 40 – Tabela que representa uma vista do Funcionário

6.5.2 Utilizadores e regras de acesso

A nossa base de dados suporta dois utilizadores diferentes: clientes e funcionários.

Para estes utilizadores, analisou-se e determinou-se quais as tabelas que estes podem ou não aceder e as operações que podem realizar.

Os clientes podem efetuar reservas online de sessões e consultar os seus dados de utilizador, assim como rever o seu histórico de compra de bilhetes. Os funcionários podem consultar/inserir/modificar/remover sessões, horários, filmes, salas e lugares na base de dados.

A Tabela 1 descreve resumidamente as permissões dos dois tipos de utilizadores. Durante a aplicação da metodologia para a futura implementação da base de dados para reserva de bilhetes de sessões de cinema, não foram identificadas.

Tabela	Cliente				Funcionário			
	SELECT	INSERT	UPDATE	DELETE	SELECT	INSERT	UPDATE	DELETE
Cliente	x	x	x					
Bilhete	x	x			x			
Sessão	x				x	x	x	x
Horário	x				x	x	x	x
Filme	x				x	x	x	x
Sala	x				x	x	x	x
Lugar	x				x	x	x	x

Tabela 1 - Permissões para os utilizadores da base de dados: cliente e funcionário

De seguida, mostra-se o SQL que cria estes utilizadores e insere as permissões corretas para cada utilizador e respetiva tabela.

Código SQL que cria o utilizador cliente e lhe atribui permissões.

```
DROP USER IF EXISTS 'cliente'@'localhost:3306';
CREATE USER 'cliente'@'localhost:3306';
SET PASSWORD FOR 'cliente'@'localhost:3306' = PASSWORD('clien1234');

GRANT SELECT, INSERT, UPDATE ON bccDB.Cliente TO 'cliente'@'localhost:3306';
GRANT SELECT, INSERT ON bccDB.Bilhete TO 'cliente'@'localhost:3306';
GRANT SELECT ON bccDB.Sessao TO 'cliente'@'localhost:3306';
GRANT SELECT ON bccDB.Filme TO 'cliente'@'localhost:3306';
GRANT SELECT ON bccDB.Sala TO 'cliente'@'localhost:3306';
GRANT SELECT ON bccDB.Lugar TO 'cliente'@'localhost:3306';
GRANT SELECT ON bccDB.Horario TO 'cliente'@'localhost:3306';
GRANT SELECT ON bccDB.vwClientes TO 'cliente'@'localhost:3306';

SHOW GRANTS FOR 'cliente'@'localhost';
```

Código SQL que cria o utilizador funcionário e lhe atribui permissões.

```
DROP USER IF EXISTS 'funcionario'@'localhost:3306';
CREATE USER 'funcionario'@'localhost:3306';
SET PASSWORD FOR 'funcionario'@'localhost:3306' = PASSWORD('func1234');

GRANT SELECT ON bccDB.Bilhete TO 'funcionario'@'localhost:3306';
GRANT SELECT, INSERT, UPDATE, DELETE ON bccDB.Sessao TO 'funcionario'@'localhost:3306';
GRANT SELECT, INSERT, UPDATE, DELETE ON bccDB.Filme TO 'funcionario'@'localhost:3306';
GRANT SELECT, INSERT, UPDATE, DELETE ON bccDB.Sala TO 'funcionario'@'localhost:3306';
GRANT SELECT, INSERT, UPDATE, DELETE ON bccDB.Lugar TO 'funcionario'@'localhost:3306';
GRANT SELECT, INSERT, UPDATE, DELETE ON bccDB.Horario TO 'funcionario'@'localhost:3306';
GRANT SELECT ON bccDB.vwFuncionarios TO 'funcionario'@'localhost:3306';

SHOW GRANTS FOR 'funcionario'@'localhost:3306';
```

6.6. Crescimento futuro e estimativa de espaço em disco.

O tamanho da base de dados após a sua definição, apenas tem em consideração a estrutura da base de dados e todos os TRIGGERS e funções necessários para manter a integridade e consistência de dados. Portanto, sem qualquer tipo de dados inseridos nas tabelas o valor inicial obtido para o base de dados bccDB foi de 316 KB.

Com o objetivo de prever o espaço em disco ocupado pela base de dados segundo uma estimativa do crescimento previsto calculou-se o tamanho singular de cada registo de cada uma das tabelas. Para tal, considerou-se o tipo de dados definido no dicionário de dados e o modelo de dados lógico. O tamanho médio de cada regista em cada uma das tabelas que constituem a base de dados encontra-se representado na Tabela 2.

Tabela/Relação	Tamanho médio ocupado por registo (Bytes)
Cliente	100
Bilhete	50
Sessão	10
Horário	10
Filme	100
Sala	50
lugar	10

Tabela 2 - Tamanho médio ocupado por cada registo de cada tabela da base de dados

6.6.1 Tamanho inicial

Salas: 3

Considerando que, atualmente, a empresa BCC dispõe das seguintes salas:

- 1 sala *Basic* (que tem capacidade para 15 lugares)
- 1 sala *Silver* (que tem capacidade para 15 lugares)
- 1 sala *Platinum* (que tem capacidade para 15 lugares)

Sessões: 11

E possibilita os seguintes **filmes**:

- The House By The Sea
- The Order Of Things
- The Phantom Thread

Tomou-se como tamanho inicial o número médio de clientes que visualizam filmes na BCC, antes da disponibilização do sistema de reservas.

Espera-se que, no momento, em que o sistema for disponibilizado cerca de 80% dos clientes habituais se registem no sistema, durante os primeiros dias. Prevê-se também que

alguns curiosos também se registem. Assim, estima-se que no início existam cerca de 700 clientes. Supondo que, inicialmente, 60% destes clientes fazem uma reserva inicialmente, haverá 420 bilhetes.

Resumindo os dados recolhidos para o tamanho inicial:

Tamanho inicial

=> Clientes: 900

=> Salas: 3

=> Filmes: 3

=> Sessões: 11

=> Horários: 7

=> Lugares: 60

=> Bilhetes/Reservas: 420

Com base na tabela anteriormente apresentada, o espaço ocupado em disco por estes registos é de $(700 * 100) + (3 * 50) + (11 * 10) + (420 * 50) + (60 * 10) + (7 * 10) = 91930$ Bytes ~ 91KB.

6.6.2 Crescimento futuro

Fazendo uma estimativa do crescimento de utilizadores através de manobras de marketing e crescimento natural, esperam-se 2500 clientes / ano.

Estima-se que cada cliente existente reserve 8 bilhetes por ano. Portanto, somando o número de bilhetes reservados pelos clientes existentes e pelos novos obtém-se cerca de 28 000 bilhetes/ano.

Espera-se também que o número de filmes aumente em cerca de 40 filmes/ano.

Para suportar este crescimento ao nível de clientes serão adquiridas cerca de 4 salas por ano.

Resumindo estes dados para o crescimento futuro vem:

Crescimento futuro

=> Clientes: 3500/ano

=> Sessões: 280/ano

=> Salas: 7/ano

=> Filmes: 40/ano

=> Bilhetes/reservas: 28 000/ano

Com base na tabela anteriormente apresentada, o espaço ocupado em disco pelo crescimento anual dos registos é de

$(3500 * 100) + (7 * 50) + (280 * 10) + (28\ 000 * 50) + (40 * 100) + (7 * 140) = 1758150$ Bytes ~ 1.75 MB.

6.7. Script de povoamento inicial

Desenvolveu-se um script de povoamento inicial com alguns dados de teste para cada uma das tabelas que constitui a base de dados. O script encontra-se no anexo (Anexo VI).

6.8. Exemplos de *Queries*

A base de dados implementada visa responder a *queries* que foram identificadas como requisitos na fase inicial do projeto. Apresentam-se alguns exemplos, de *queries* importantes que a base de dados consegue responder. Para cada *query* apresenta-se o resultado obtido com o povoamento inicial de teste (AnexoVII)

QUERY 1: Obter o número de lugares reservados para uma dada sessão

```
-----  
-- QUERY 1: Obter o número de lugares reservados para uma dada sessão  
-----
```

```
DELIMITER $$  
CREATE PROCEDURE nrLugaresReservadosSessao(IN idSessao INT)  
BEGIN  
    SELECT COUNT(*) AS NrLugaresReservados  
    FROM Bilhete  
    WHERE Bilhete.idSessao = idSessao;  
END $$  
  
CALL nrLugaresReservadosSessao(1);
```

O resultado obtido é:

NrLugaresReservados
1

QUERY 2: Obter o número total de bilhetes vendidos

```
-----  
-- QUERY 2: Obter o número total de bilhetes vendidos  
-----
```

```
DELIMITER $$  
CREATE PROCEDURE nrBilhetesVendidos()  
BEGIN  
    SELECT COUNT(*) AS NrBilhetesVendidos  
    FROM Bilhete;  
END $$  
  
CALL nrBilhetesVendidos();
```

O resultado obtido é:

NrBilhetesVendidos
4

QUERY 3: Número de lugares reservados para um determinado dia de sessão

```
-- QUERY 3: Número de lugares reservados para um determinado dia de uma sessão
```

```
DELIMITER $$
CREATE PROCEDURE nrLugaresReservadosEmData(IN dataSessao DATE)
BEGIN
    SELECT COUNT(*) AS NrBilhetesReservados
    FROM Bilhete
    WHERE Bilhete.dataSessao = dataSessao;
END $$

CALL nrLugaresReservadosEmData('2016-07-15');
```

O resultado obtido é:

NrBilhetesReservados
1

QUERY 4: Obter as reservas/compras de um cliente

```
DELIMITER $$
CREATE PROCEDURE reservasCliente(IN usern VARCHAR(10))
BEGIN
    SELECT Bilhete.dataSessao, Horario.hora, Filme.titulo, Lugar.lugar, Bilhete.precoTotal
    FROM Cliente
    JOIN Bilhete ON Bilhete.idCliente = Cliente.idCliente
    JOIN Sessao ON Bilhete.idSessao = Sessao.idSessao
    JOIN Horario ON Sessao.idHorario = Horario.idHorario
    JOIN Filme ON Sessao.idFilme = Filme.idFilme
    JOIN Sala ON Filme.idSala = Sala.idSala
    JOIN lugar ON Sala.idSala = lugar.idSala
    WHERE Lugar.lugar = Bilhete.lugar AND Cliente.username = usern;
END $$

CALL reservasCliente('joaoS');
```

O resultado obtido é:

	dataSessao	hora	titulo	lugar	precoTotal
	2014-10-27	21:00:00	The Order Of Things	7	0

QUERY 5: Obter os lugares disponíveis de uma sessão

```
DELIMITER $$
CREATE PROCEDURE lugaresDisponiveisSessao(IN idSess INT)
BEGIN
    SELECT Bilhete.dataSessao, Horario.hora, Filme.titulo, Sala.nomeSala, Lugar.lugar
    FROM Bilhete
    JOIN Sessao ON Bilhete.idSessao = Sessao.idSessao
    JOIN Horario ON Sessao.idHorario = Horario.idHorario
    JOIN Filme ON Sessao.idFilme = Filme.idFilme
    JOIN Sala ON Filme.idSala = Sala.idSala
    JOIN lugar ON Sala.idSala = lugar.idSala
    WHERE Bilhete.idSessao = idSess AND Bilhete.lugar <> Lugar.lugar;
END $$

CALL lugaresDisponiveisSessao(1);
```

O resultado obtido é:

	dataSessao	hora	titulo	nomeSala	lugar
	2016-07-15	09:00:00	The House Bv The Sea	Martin Scorsese	4
	2016-07-15	09:00:00	The House Bv The Sea	Martin Scorsese	5
	2016-07-15	09:00:00	The House Bv The Sea	Martin Scorsese	6
	2016-07-15	09:00:00	The House Bv The Sea	Martin Scorsese	7
	2016-07-15	09:00:00	The House Bv The Sea	Martin Scorsese	8
	2016-07-15	09:00:00	The House Bv The Sea	Martin Scorsese	9
	2016-07-15	09:00:00	The House Bv The Sea	Martin Scorsese	10
	2016-07-15	09:00:00	The House Bv The Sea	Martin Scorsese	11
	2016-07-15	09:00:00	The House Bv The Sea	Martin Scorsese	13
	2016-07-15	09:00:00	The House Bv The Sea	Martin Scorsese	14
	2016-07-15	09:00:00	The House Bv The Sea	Martin Scorsese	15

QUERY 6: Obter Top 10 das sessões com mais clientes

```
DELIMITER $$
CREATE PROCEDURE top10SessoesComMaisClientesEntreDatas(IN data1 DATE, IN data2 DATE)
BEGIN
    SELECT Filme.titulo, Sessao.idSessao, COUNT(*) AS NrClientes
    FROM Bilhete
    JOIN Sessao ON Bilhete.idSessao = Sessao.idSessao
    JOIN Filme ON Sessao.idFilme = Filme.idFilme
    WHERE Bilhete.dataSessao BETWEEN data1 AND data2
    GROUP BY Bilhete.idSessao
    ORDER BY COUNT(*) DESC
    LIMIT 10;
END $$

CALL top10ViagensComMaisClientesEntreDatas('2000-04-01', '2017-10-31');
```

O resultado obtido é:

	titulo	idSessao	NrClientes
	The House Bv The Sea	3	1
	The Order Of Things	5	1
	The Order Of Things	7	1
	The House Bv The Sea	1	1

7. Conclusões e Trabalho Futuro

No capítulo 1 deste relatório contextualizou-se e analisou-se o caso em estudo, isto é, qual a realidade em que a Braga Cinema Center está inserida e como deseja implementar uma base de dados para a reserva de sessões de cinema. Para além disso, identificou-se os motivos que levaram a empresa a solicitar a implementação de uma base de dados, bem como os objetivos que visa alcançar com esta. Concluindo-se assim que a maioria dos objetivos são alcançados, o que implica que está pronta a cumprir com os restantes objetivos como: recolha de dados que permitam traçar perfis de clientes e futura personalização da experiência. Com vista à implementação da base de dados pedida pela BCC, seguiu-se a metodologia apresentada e discutida no livro *Connolly, T., Begg, C., Database Systems, A Practical Approach to Design, Implementation, and Management, Addison-Wesley, 4ª Edição, 2004*.

Neste relatório foram apresentadas as primeiras etapas do desenvolvimento da base de dados da empresa BCC. A primeira etapa consistiu em realizar um plano para o levantamento de requisitos. Este plano mostrou-se essencial pois permitiu definir e refletir sobre o domínio do problema antes de proceder de forma imatura para as etapas seguintes.

Foram identificadas as potenciais fontes de requisitos e os perfis de utilização. Sistemas de reservas já existentes e utilizadores que reservam frequentemente bilhetes mostraram ser a principal fonte de requisitos. Identificaram-se dois perfis de utilizadores principais: cliente que reserva/compra os bilhetes e o funcionário da empresa que faz a gestão das sessões/filmes/salas/horários que a BCC disponibiliza.

Analisaram-se as fontes de requisitos e os perfis e extraíram-se os requisitos que a base de dados deve implementar para ser útil e responder às necessidades de cada um dos perfis de utilizadores. Foram identificados 19 requisitos fundamentais.

O levantamento de requisitos mostrou ser uma tarefa árdua e complicada, uma vez que é difícil extrair as funcionalidades exatas que os utilizadores mais precisam. É fundamental perceber o que os utilizadores pretendem e necessitam. Sem isso, o sistema torna-se inútil. Nesta etapa foi onde se encontrou mais dificuldades, a identificação dos requisitos não é imediata e estes necessitam de constante validação.

Como ponto de melhoria futura sugere-se a validação dos requisitos com um maior número de utilizadores reais da base de dados e, se necessário, o ajuste desses requisitos.

Na segunda etapa, com base nos requisitos identificados, foi construído o diagrama conceptual. Esta construção foi feita de forma iterativa e o diagrama conceptual foi sendo refinado constantemente de forma a representar o mais fidedigno possível as entidades do sistema e os seus relacionamentos.

Esta etapa tomou grande parte do tempo disponível para a construção da base de dados, uma vez que foi necessário certificar e validar segundo todos os requisitos definidos de forma a garantir que o sistema permite responder a todas as operações requeridas. Além disso, o diagrama conceptual é a base para todas as etapas que se seguem e portanto é fundamental que seja o mais preciso possível. Esta etapa foi totalmente documentada através de descrições e da elaboração do dicionário de dados contendo informação sobre as entidades, os atributos de cada entidade e os relacionamentos entre as entidades.

A terceira etapa da construção da base de dados consistiu em derivar as tabelas/relações do modelo conceptual para o modelo lógico através das regras definidas na metodologia.

Posteriormente, validou-se as relações segundo a normalização de forma a garantir um armazenamento consistente e um eficiente acesso aos dados na base de dados. Neste processo verificou-se que o modelo desenvolvido se encontra normalizado, cumprindo a primeira, segunda e terceira forma normal. Dado que o modelo conceptual foi definido antecipadamente e foram aplicadas as respetivas regras de derivação, já se esperava que o modelo lógico obtido se encontrasse normalizado.

A última etapa consiste na construção do modelo físico e depende do SGBD a utilizar. O SGBD escolhido foi o MySQL pois é um sistema *open-source*, fácil de utilizar, com bastante reconhecimento e suporte da comunidade, apresentando ainda outras vantagens. A implementação da base de dados foi concebida de forma a garantir a integridade de dados e as restrições impostas pelas regras de negócio.

De modo a validar o modelo físico analisaram-se as transações mais relevantes e implementaram-se os mecanismos que asseguram que estas transações são executadas adequadamente.

Confirmou-se e assegurou-se que os requisitos definidos são respondidos pelo sistema de base de dados, através das implementações das respetivas *queries*. Adicionalmente criaram-se duas vistas para os dois utilizadores do sistema da base de dados: o cliente e o funcionário, e definiram-se as respetivas regras de acesso relativamente a cada uma das tabelas.

Tudo isto foi verificado, pelo que podemos concluir que todos os passos da metodologia foram bem executados.

Após as várias etapas e validação efetuada em cada um dos modelos obteve-se a confirmação de que o sistema foi implementado com sucesso, isto é, que é capaz de responder adequadamente aos requisitos identificados e é capaz de garantir a integridade de dados e das regras de negócio. Além disso, a base de dados foi pensada e desenvolvida de forma a suportar futuras alterações que possam surgir.

Referências

Connolly, T., Begg, C., 2004, Database Systems, A Practical Approach to Design, Implementation, and Management, 4ª Edição, Addison-Wesley.

Lista de Siglas e Acrónimos

Lista com todas as siglas e acrónimos utilizados durante a realização do trabalho.

BD	Base de Dados
SGBD	Sistema de Gestão de Base de Dados
BCC	<i>Braga Cinema Center</i>
DBDL	<i>Database Design Language</i>
DBMS	<i>Database Management System</i>

Anexos

Anexos utilizados para a inclusão de informação adicional.

I. Anexo 1 - Dicionário de Dados das Entidades

Na Tabela 3 encontram-se documentadas as Entidades identificadas no modelo

Nome da Entidade	Descrição	Aliases	Ocorrência
Cliente	Pessoa que deseja reservar um ou vários bilhetes de sessão de cinema, mediante pagamento	utilizador, utente, cinéfilo	Cada cliente faz a reserva de um ou mais bilhetes de cinema. Para isso, necessita de estar autenticado na aplicação, com um <i>username</i> e <i>password</i> .
Bilhete	Nota que contém a informação sobre o tipo de bilhete, o lugar a que se refere, a data da sessão, o preço total, se usufrui de desconto e qual o seu IVA.	Vale, ingresso, mensagem, informação, compra	A reserva de um bilhete é efetuada por um cliente. O bilhete representa a reserva de um lugar numa sessão de um filme, tem um tipo (promocional ou normal) e contém informação sobre a data e o preço da sessão e o lugar na sala de cinema.
Sessão	Entidade que faz a ligação entre o filme e a hora do dia em que ele é projetado.	Projeção	
Horário	Agrupar as várias horas do dia a que uma sessão tem início	Hora, início	Cada sessão de um filme tem um início a uma hora do dia específica.
Filme	Representa a película que vai ser várias vezes projetada no cinema. Tem um título, uma duração, uma data de estreia, data de fim e idade mínima.	Película, curta-metragem	O filme tem lugar uma só sala, várias vezes e em diversas horas do dia, e cada uma dessas vezes é considerada uma sessão do filme.
Sala	É o local onde os clientes assistem à sessão de um filme. É caracterizada pelo seu nome, tipo e capacidade. Tem um número exato de lugares.	Local	Cada filme é projetado num local, neste caso a sala de cinema, que é composta por diversos lugares, que correspondem ao bilhete.

Tabela 3 – Descrição das Entidades

II. Anexo 2 - Dicionário de Dados dos Relacionamentos

Na Tabela 4 encontram-se documentados os relacionamentos identificados entre as várias entidades existentes no modelo conceptual.

Entidade	Multiplicidade	Relação	Multiplicidade	Entidade (Relacionada)
Cliente	1	reserva compra	N	Bilhete
Bilhete	N	refere está associada a	1	Sessão
Horário	1	agrupa	N	Sessão
Sessão	N	diz respeito	1	Filme
Sala	1	projeta	N	Filme

Tabela 4 – Descrição dos relacionamentos entre as entidades

III. Anexo 3 - Dicionário de Dados dos Atributos

- Dicionário de Dados dos Atributos da entidade Cliente**

Na Tabela 5 encontram-se documentados os atributos da entidade Cliente

Nome da Entidade	Atributo	Descrição	Tipo de Dados e Comprimento /Domínio	Nulo	Tipo de atributo	Valor Por Defeito
Cliente	idCliente	Identificador único de Cliente	INT	Não	Simples	Auto Incrementado
	nome	Nome do cliente	VARCHAR(75)	Não	Simples	-
	dataNascimento	Data de nascimento do cliente	DATE	Não	Simples	-
	email	Email do cliente	VARCHAR(75)	Não	Simples	-
	telefone	Telefone do cliente	VARCHAR(14) Telefone deve começar por: +0351 Portugal	Não	Simples	-
	dataRegisto	Data em que o cliente se registou no sistema	DATETIME	Não	Simples	NOW()
	password	Password que permite ao cliente aceder ao sistema	VARCHAR(20)	Não	Simples	-
	username	Identificador único do cliente no sistema	VARCHAR(10)	Não	Simples	-
	<i>membroPremium</i>	Identifica se o utilizador é membroPremium	TINYINT	Não	Simples	0

Tabela 5 – Descrição dos atributos da entidade Cliente

- **Dicionário de Dados dos Atributos da entidade Bilhete**

Na Tabela 6 encontram-se documentados os atributos da entidade Bilhete

Nome da Entidade	Atributo	Descrição	Tipo de Dados e Comprimento /Domínio	Nulo	Tipo de atributo	Valor Por Defeito
Bilhete	idBilhete	Identificador único de Bilhete	INT	Não	Simple	Auto Incrementado
	tipoBilhete	Identifica o tipo de Bilhete que foi comprado pelo cliente, sendo Normal ou Promocional	VARCHAR(1) Tipo do Bilhete: P ou N	Não	Derivado	-
	dataCompra	Data de compra do Bilhete	DATETIME	Não	Simple	NOW()
	dataSessao	Data da Sessão	DATE	Não	Simple	-
	lugar	Número do lugar da sala que projeta o filme	INT Pertence ao intervalo: [1, capacidade]	Não	Simple	-
	precoTotal	Preço pago pelo bilhete	FLOAT	Não	Derivado	-
	IVA	Imposto IVA adicionado ao preço do bilhete	FLOAT	Não	Simple	0.13
	desconto	Valor subtraído ao preço do bilhete consoante a idade do cliente, ou se é membroPremium	FLOAT Sem desconto: 0.0 Jovens(<25): 0.15, Idosos(> 65): 0.20 Membro: 0.30	Não	Derivado	-

Tabela 6 – Descrição dos atributos da entidade Bilhete

- **Dicionário de Dados dos Atributos da entidade Horário**

Na Tabela 7 encontram-se documentados os atributos da entidade Horário

Nome da Entidade	Atributo	Descrição	Tipo de Dados e Comprimento /Domínio	Nulo	Tipo de atributo	Valor Por Defeito
Horário	idHorário	Identificador único do Horário	INT	Não	Simple	Auto Incrementado
	hora	Hora de início da sessão	TIME	Não	Simple	-

Tabela 7 – Descrição dos atributos da entidade Horário

- **Dicionário de Dados dos Atributos da entidade Sessão**

Na Tabela 8 encontram-se documentados os atributos da entidade Sessão

Nome da Entidade	Atributo	Descrição	Tipo de Dados e Comprimento /Domínio	Nulo	Tipo de atributo	Valor Por Defeito
Sessão	idSessão	Identificador único da Sessão	INT	Não	Simple	Auto Incrementado
	preçoBase	Preço base da sessão	FLOAT	Não	Simple	-

Tabela 8 – Descrição dos atributos da entidade Sessão

- **Dicionário de Dados dos Atributos da entidade Filme**

Na Tabela 9 encontram-se documentados os atributos da entidade Filme

Nome da Entidade	Atributo	Descrição	Tipo de Dados e Comprimento /Domínio	Nulo	Tipo de atributo	Valor Por Defeito
Filme	idFilme	Identificador único do Filme	INT	Não	Simple	Auto Incrementado
	título	Título do filme	VARCHAR(75)	Não	Simple	-
	duracao	Duração do filme	INT	Não	Simple	-
	idadeMinima	Idade mínima para se assistir ao filme		Não	Simple	-
	dataEstreia	Data de Estreia do filme	DATE	Não	Simple	-
	dataFim	Data de saída das salas	DATE	Não	Simple	-

Tabela 9 – Descrição dos atributos da entidade Filme

- **Dicionário de Dados dos Atributos da entidade Sala**

Na Tabela 10 encontram-se documentados os atributos da entidade Sala

Nome da Entidade	Atributo	Descrição	Tipo de Dados e Comprimento /Domínio	Nulo	Tipo de atributo	Valor Por Defeito
Sala	idSala	Identificador único da sala	INT	Não	Simples	Auto Incrementado
	capacidade	Número de lugares da sala	INT <i>Basic: 15</i> <i>Silver: 20</i> <i>Platinum: 25</i>	Não	Derivado	-
	tipoSala	Tipo de Sala	CHAR <i>Basic: B</i> <i>Silver: S</i> <i>Platinum: P</i>	Não	Simples	-
	nomeSala	Nome da sala	VARCHAR(45)	Não	Simples	-

Tabela 10 – Descrição dos atributos da entidade Sala

- **Dicionário de Dados dos Atributos do Atributo Multivalor Lugar**

Na Tabela 11 encontram-se documentados os atributos do atributo Multivalor Lugar

Nome atributo MultiValor	Atributo	Descrição	Tipo de Dados e Comprimento /Domínio	Nulo	Tipo de atributo	Valor Por Defeito
Lugar	lugar	Identifica o lugar da sala reservado para a assistir ao filme	INT [1, capacidade]	Não	Simples	-

Tabela 11 – Descrição dos atributos do atributo Multivalor Lugar

IV. Anexo 4 – Script de Inicialização da Base de Dados bccBD

```
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

-- -----
-- Schema bccDB
-- -----

-- -----
-- Schema bccDB
-- -----

CREATE SCHEMA IF NOT EXISTS `bccDB` DEFAULT CHARACTER SET utf8 ;
USE `bccDB` ;

-- -----
-- Table `bccDB`.`Cliente`
-- -----

CREATE TABLE IF NOT EXISTS `bccDB`.`Cliente` (
  `idCliente` INT NOT NULL AUTO_INCREMENT,
  `username` VARCHAR(10) NOT NULL,
  `password` VARCHAR(20) NOT NULL,
  `nome` VARCHAR(75) NOT NULL,
  `telefone` VARCHAR(14) NOT NULL,
  `dataNascimento` DATE NOT NULL,
  `email` VARCHAR(75) NOT NULL,
  `dataRegisto` DATETIME NOT NULL DEFAULT NOW(),
  `membroPremium` TINYINT NOT NULL DEFAULT 0,
  PRIMARY KEY (`idCliente`),
  UNIQUE INDEX `username_UNIQUE` (`username` ASC),
  UNIQUE INDEX `idCliente_UNIQUE` (`idCliente` ASC),
  UNIQUE INDEX `email_UNIQUE` (`email` ASC))
ENGINE = InnoDB;
```

```
-- -----  
-- Table `bccDB`.`Horario`  
-- -----
```

```
CREATE TABLE IF NOT EXISTS `bccDB`.`Horario` (  
  `idHorario` INT NOT NULL AUTO_INCREMENT,  
  `hora` TIME NOT NULL,  
  PRIMARY KEY (`idHorario`),  
  UNIQUE INDEX `idHorario_UNIQUE` (`idHorario` ASC),  
  UNIQUE INDEX `horaInicio_UNIQUE` (`hora` ASC))  
ENGINE = InnoDB;
```

```
-- -----  
-- Table `bccDB`.`Sala`  
-- -----
```

```
CREATE TABLE IF NOT EXISTS `bccDB`.`Sala` (  
  `idSala` INT NOT NULL AUTO_INCREMENT,  
  `nomeSala` VARCHAR(45) NOT NULL,  
  `tipoSala` CHAR NOT NULL,  
  `capacidade` INT AS (CASE WHEN tipoSala = 'B' THEN 15 WHEN tipoSala = 'S' THEN 20  
ELSE 25 END) STORED NOT NULL,  
  PRIMARY KEY (`idSala`),  
  UNIQUE INDEX `idFilme_UNIQUE` (`idSala` ASC),  
  UNIQUE INDEX `nomeSala_UNIQUE` (`nomeSala` ASC))  
ENGINE = InnoDB;
```

```
-- -----  
-- Table `bccDB`.`Filme`  
-- -----
```

```
CREATE TABLE IF NOT EXISTS `bccDB`.`Filme` (  
  `idFilme` INT NOT NULL AUTO_INCREMENT,  
  `titulo` VARCHAR(75) NOT NULL,  
  `duracao` INT NOT NULL,  
  `dataEstreia` DATE NOT NULL,  
  `dataFim` DATE NOT NULL,  
  `idadeMinima` INT NOT NULL,  
  `idSala` INT NOT NULL,  
  UNIQUE INDEX `idFilme_UNIQUE` (`idFilme` ASC),  
  PRIMARY KEY (`idFilme`),  
  INDEX `fk_Filme_Sala1_idx` (`idSala` ASC),  
  UNIQUE INDEX `titulo_UNIQUE` (`titulo` ASC),  
  CONSTRAINT `fk_Filme_Sala1`  
    FOREIGN KEY (`idSala`)  
    REFERENCES `bccDB`.`Sala` (`idSala`)  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION)  
ENGINE = InnoDB;
```

```
-- Table `bccDB`.`Sessao`
```

```
CREATE TABLE IF NOT EXISTS `bccDB`.`Sessao` (
  `idSessao` INT NOT NULL AUTO_INCREMENT,
  `precoBase` FLOAT NOT NULL,
  `idHorario` INT NOT NULL,
  `idFilme` INT NOT NULL,
  PRIMARY KEY (`idSessao`),
  UNIQUE INDEX `idSessao_UNIQUE` (`idHorario` ASC, `idFilme` ASC),
  INDEX `fk_Sessao_Horario1_idx` (`idHorario` ASC),
  INDEX `fk_Sessao_Filme1_idx` (`idFilme` ASC),
  CONSTRAINT `fk_Sessao_Horario1`
    FOREIGN KEY (`idHorario`)
    REFERENCES `bccDB`.`Horario` (`idHorario`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Sessao_Filme1`
    FOREIGN KEY (`idFilme`)
    REFERENCES `bccDB`.`Filme` (`idFilme`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- Table `bccDB`.`Bilhete`
```

```
CREATE TABLE IF NOT EXISTS `bccDB`.`Bilhete` (
  `idBilhete` INT NOT NULL AUTO_INCREMENT,
  `IVA` FLOAT NOT NULL DEFAULT 0.13,
  `dataCompra` DATETIME NOT NULL DEFAULT NOW(),
  `dataSessao` DATE NOT NULL,
  `tipoBilhete` VARCHAR(1) AS (CASE WHEN desconto > 0 THEN 'P' ELSE 'N' END) NOT
  NULL,
  `desconto` FLOAT NOT NULL,
  `precoTotal` FLOAT NOT NULL,
  `lugar` INT NOT NULL,
  `idSessao` INT NOT NULL,
  `idCliente` INT NOT NULL,
  PRIMARY KEY (`idBilhete`),
  UNIQUE INDEX `idBilhete_UNIQUE` (`idSessao` ASC, `lugar` ASC, `dataSessao` ASC),
  INDEX `fk_Bilhete_Sessao1_idx` (`idSessao` ASC),
  INDEX `fk_Bilhete_Cliente1_idx` (`idCliente` ASC),
  CONSTRAINT `fk_Bilhete_Sessao1`
    FOREIGN KEY (`idSessao`)
    REFERENCES `bccDB`.`Sessao` (`idSessao`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
```

```

REFERENCES `bccDB`.`Sessao` (`idSessao`)
ON DELETE NO ACTION
ON UPDATE NO ACTION,
CONSTRAINT `fk_Bilhete_Cliente1`
FOREIGN KEY (`idCliente`)
REFERENCES `bccDB`.`Cliente` (`idCliente`)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

-----
-- Table `bccDB`.`Lugar`
-----

CREATE TABLE IF NOT EXISTS `bccDB`.`Lugar` (
  `lugar` INT NOT NULL,
  `idSala` INT NOT NULL,
  PRIMARY KEY (`lugar`, `idSala`),
  INDEX `fk_lugar_Sala1_idx` (`idSala` ASC),
  CONSTRAINT `fk_lugar_Sala1`
    FOREIGN KEY (`idSala`)
    REFERENCES `bccDB`.`Sala` (`idSala`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
ENGINE = InnoDB;

```

V. Anexo 5 – Script de Validação de Dados

```
-----  
-- TRIGGER que garante que o número de telefone é válido  
-- TRIGGER que garante que o número de telefone tem 14 dígitos  
-----
```

```
DELIMITER $$  
CREATE TRIGGER TR_Cliente_BI  
    BEFORE INSERT ON Cliente  
    FOR EACH ROW  
BEGIN  
    IF (NEW.telefone NOT LIKE "+0351%") THEN  
        SIGNAL SQLSTATE '45000';  
    END IF;  
  
    IF (CHAR_LENGTH(NEW.telefone) <> 14) THEN  
        SIGNAL SQLSTATE '45000';  
    END IF;  
END $$
```

```
-----  
-- TRIGGER que garante que o tipo de sala é B (Basic), S (Silver) ou P (Platinum)  
-----
```

```
DELIMITER $$  
CREATE TRIGGER TR_Sala_BI  
    BEFORE INSERT ON Sala  
    FOR EACH ROW  
BEGIN  
    IF (NEW.tipoSala <> 'B' AND NEW.tipoSala <> 'S' AND NEW.tipoSala <> 'P') THEN  
        SIGNAL SQLSTATE '45000';  
    END IF;  
END $$
```

```
-----
-- TRIGEEER que garante que o número de lugar é inferior à capacidade da sala
-- TRIGGER que garante que o número de lugares de uma sala não excede sua capacidade
-----
```

```
DELIMITER $$
```

```
CREATE TRIGGER TR_lugar_BI
```

```
    BEFORE INSERT ON lugar
```

```
    FOR EACH ROW
```

```
BEGIN
```

```
    DECLARE nrLugaresExistentes INT;
```

```
    DECLARE capacidade INT;
```

```
    SET capacidade = (    SELECT capacidade
                        FROM Sala
                        WHERE idSala = NEW.idSala);
```

```
-- VALIDACAO: número de lugar é inferior à capacidade da sala
```

```
    IF (NEW.lugar > capacidade) THEN
```

```
        SIGNAL SQLSTATE '45000';
```

```
    END IF;
```

```
    SET nrLugaresExistentes = (    SELECT COUNT(*)
                                FROM lugar
                                WHERE idSala = NEW.idSala);
```

```
-- VALIDACAO: número de lugares de uma sala não excede a sua capacidade
```

```
    IF nrLugaresExistentes >= capacidade THEN
```

```
        SIGNAL SQLSTATE '45000';
```

```
    END IF;
```

```
END $$
```

```
-----
-- TRIGGER que permite calcular o atributo derivado: desconto de um bilhete
-----
```

```
DELIMITER $$
```

```
CREATE TRIGGER TR_Bilhete_Desconto_BI
```

```
    BEFORE INSERT ON Bilhete
```

```
    FOR EACH ROW
```

```
BEGIN
```

```
    DECLARE idade INT;
```

```
    DECLARE membro BOOLEAN;
```

```
    SET idade = (    SELECT calculaIdade(dataNascimento)
                    FROM Cliente
                    WHERE idCliente = NEW.idCliente);
```



```

SET membro =( SELECT membroPremium
                FROM Cliente
                WHERE idCliente = NEW.idCliente);

IF (idade < 25 ) THEN
    SET NEW.desconto = 0.15;
ELSEIF (idade > 65) THEN
    SET NEW.desconto = 0.25;
ELSE
    SET NEW.desconto = 0;
END IF;

IF (membro=1) THEN
    SET NEW.desconto = 0.30;
END IF;

END $$

-----
-- FUNCTION que dada uma data de nascimento calcula a idade
-----

DELIMITER $$

CREATE FUNCTION calculaIdade(dataNascimento DATE) RETURNS INT
BEGIN
    DECLARE idade INT;

    SET idade = TIMESTAMPDIFF(YEAR, dataNascimento, CURDATE());

    RETURN idade;
END $$

-----
-- TRIGGER que permite calcular o atributo derivado: precoTotal de um Bilhete
-----

DELIMITER $$

CREATE TRIGGER TR_Bilhete_PrecosTotal_BI
    BEFORE INSERT ON Bilhete
    FOR EACH ROW FOLLOWS TR_Bilhete_Desconto_BI
BEGIN
    DECLARE precoB FLOAT;
    DECLARE precoComIVA FLOAT;
    DECLARE precoComDesconto FLOAT;

```

```

SET precoB = (SELECT `precoBase`
                FROM idSessao
                WHERE idSessao = NEW.idSessao);

SET precoComIVA = precoB * (1 + NEW.IVA);

SET precoComDesconto = precoComIVA * (1 - NEW.desconto);

SET NEW.`precoTotal` = ROUND(precoComDesconto,2);

END $$

-----
-- TRIGGER que garante que o número de bilhetes comprados/reservados não excede a
capacidade máxima da sala
-----

DELIMITER $$
CREATE TRIGGER TR_Bilhete_BI
    BEFORE INSERT ON Bilhete
    FOR EACH ROW
BEGIN
    DECLARE nrBilhetes INT;
    DECLARE nrBilhetesMax INT;

    SET nrBilhetes = nrBilhetesParaSessaoData(NEW.idSessao, NEW.dataSessao);

    SET nrBilhetesMax = ( SELECT Sala.capacidade
                        FROM Sessao
                        JOIN Filme ON Sessao.idFilme = Filme.idFilme
                        JOIN Sala ON Filme.idSala = Sala.idSala
                        WHERE Sessao.idSessao = NEW.idSessao);

    IF nrBilhetes >= nrBilhetesMax THEN
        SIGNAL SQLSTATE '45000';
    END IF;
END $$

```

```

-----
-- FUNCTION que dado uma data de uma sessao, e uma sessao (idSessao) dá quantos
bilhetes estão vendidos/reservados
-----

DELIMITER $$

CREATE FUNCTION nrBilhetesSessaoVendidos(idSessao INT, dataSessao DATE) RETURNS INT
BEGIN
    DECLARE nrBilhetes INT;

    SET nrBilhetes = (SELECT COUNT(*)
                      FROM Bilhete
                      WHERE Bilhete.idSessao = idSessao AND Bilhete.dataSessao = dataSessao);

    RETURN nrBilhetes;
END $$

SET

preçoTotal` = ROUND(precoComDesconto,2);
END $$

```

VI. Anexo 6 – *Script* de Povoamento Inicial

```
-- Universidade do Minho
-- Mestrado Integrado em Engenharia Informática
-- Unidade Curricular de Bases de Dados
-- 2017/2018
--
-- Caso de Estudo: "Sétima Arte: gestão de sessões de cinema"

-- Base de dados de trabalho
USE `bccDB`;

-- Povoamento da tabela "Cliente"
INSERT INTO Cliente
    (username, password, nome, telefone, dataNascimento, email)
VALUES
    ('joaoS', 'joao1234', 'João Martins Silva', '+0351258843219', '1997-03-11', 'joaomsilva@gmail.com'),
    ('mariaS', 'maria1234', 'Maria Santos', '+0351253193219', '1983-07-21', 'mariasantos@gmail.com'),
    ('pauloR', 'pablo1234', 'Paulo Rodrigues', '+0351258741596', '1989-05-05', 'pablorodriguez@hotmail.com'),
    ('penelopeL', 'penelope1234', 'Penelope Silva', '+0351914785965', '1945-02-17', 'penelopelopez@gmail.com'),
    ('andreC', 'andre1234', 'Andre Carvalho', '+0351243222111', '1990-07-23', 'andre@gmail.com');
-- SELECT * FROM Cliente;

-- Povoamento da tabela "Horario"
INSERT INTO Horario
    (hora)
VALUES
    ('09:00:00'),
    ('11:30:00'),
    ('13:00:00'),
    ('16:00:00'),
    ('18:30:00'),
    ('21:00:00'),
    ('23:30:00');
```

```

-- SELECT * FROM Sessao;

-- Povoamento da tabela "Sala"
INSERT INTO Sala
    (nomeSala, tipoSala, capacidade)
VALUES
    ('Martin Scorsese', 'B', 15),
    ('Steven Spielberg', 'S', 20),
    ('Woody Allen', 'P', 25);
-- SELECT * FROM Sala;

-- Povoamento da tabela "Filme"
INSERT INTO Filme
    (titulo, duracao, dataEstreia, dataFim, idadeMinima, idSala)
VALUES
    ('The House By The Sea', 127, '2016-07-15', '2016-09-17', 16, 1),
    ('The Order Of Things', 117, '2017-03-17', '2017-08-11', 12, 2),
    ('The Phantom Thread', 117, '2017-08-07', '2017-09-21', 12, 3);
-- SELECT * FROM Filme;

-- Povoamento da tabela "Sessao"
INSERT INTO Sessao
    (precoBase, idHorario, idFilme)
VALUES
    (3.50, 1, 1),
    (5.50, 3, 1),
    (6.50, 5, 1),
    (7.00, 7, 1),
    (5.50, 2, 2),
    (5.50, 4, 2),
    (6.50, 6, 2),
    (3.50, 1, 3),
    (5.50, 3, 3),
    (6.50, 5, 3),
    (7.00, 7, 3);

-- SELECT * FROM Sessao;

```

```
-- Povoamento da tabela "Bilhete"
INSERT INTO Bilhete
    (lugar, dataSessao, dataCompra, idCliente, idSessao)
VALUES
    ('12', '2016-07-15', '2016-05-15', 3, 1),
    ('1', '2015-04-30', '2015-03-12', 2, 5),
    ('7', '2014-10-27', '2014-09-13', 1, 7),
    ('9', '2015-09-10', '2015-07-11', 5, 3);
-- SELECT * FROM Bilhete;
```

```
-- Povoamento da tabela "Lugar"
```

```
INSERT INTO Lugar
    (lugar, idSala)
VALUES
    (1, 1),
    (2, 1),
    (3, 1),
    (4, 1),
    (5, 1),
    (6, 1),
    (7, 1),
    (8, 1),
    (9, 1),
    (10, 1),
    (11, 1),
    (12, 1),
    (13, 1),
    (14, 1),
    (15, 1),
    (1, 2),
    (2, 2),
    (3, 2),
    (4, 2),
    (5, 2),
    (6, 2),
    (7, 2),
    (8, 2),
    (9, 2),
    (10, 2),
    (11, 2),
    (12, 2),
    (13, 2),
    (14, 2),
    (15, 2),
```

```

(16, 2),
(17, 2),
(18, 2),
(19, 2),
(20, 2),
(1, 3),
(2, 3),
(3, 3),
(4, 3),
(5, 3),
(6, 3),
(7, 3),
(8, 3),
(9, 3),
(10, 3),
(11, 3),
(12, 3),
(13, 3),
(14, 3),
(15, 3),
(16, 3),
(17, 3),
(18, 3),
(19, 3),
(20, 3),
(21, 3),
(22, 3),
(23, 3),
(24, 3),
(25, 3);
-- SELECT * FROM Lugar;

-- <fim>
--

```

VII. Anexo 7 – Restantes Queries

-- Dada uma Sessao, quais os lugares livres na Sala?

```
SELECT Bilhete.dataSessao, Horario.hora, Filme.titulo, Sala.nomeSala, Lugar.lugar
FROM Bilhete
JOIN Sessao ON Bilhete.idSessao = Sessao.idSessao
JOIN Horario on Sessao.idHorario = Horario.idHorario
JOIN Filme ON Sessao.idFilme = Filme.idFilme
JOIN Sala ON Filme.idSala = Sala.idSala
JOIN lugar ON Sala.idSala = lugar.idSala
WHERE Bilhete.idSessao = 1 AND Bilhete.lugar <> Lugar.lugar;
```

-- Bilhetes de um utilizador

```
SELECT *
FROM Bilhete
JOIN Cliente ON Cliente.idCliente = Bilhete.idCliente
WHERE Cliente.username = 'joaoS';
```

-- Número de bilhetes de um utilizador

```
SELECT COUNT(*)
FROM Bilhete
JOIN Cliente ON Cliente.idCliente = Bilhete.idCliente
WHERE Cliente.username = 'joaoS';
```

-- Saber o número de filmes

```
SELECT COUNT(*)
FROM Filme;
```

-- Saber o número de sessões

```
SELECT COUNT(*)  
FROM Sessao;
```

-- Saber quantos filmes projeta uma sala

```
SELECT COUNT(*)  
FROM Filme  
WHERE idSala = 1;
```

-- Quantos bilhetes vendidos existem para uma sessão

```
SELECT COUNT(*)  
FROM Bilhete  
WHERE idSessao = 1;
```

-- Qual o total (em €) de bilhetes vendidos para um determinado Filme

```
SELECT SUM(Bilhete.precoTotal)  
FROM Bilhete  
JOIN Sessao ON Sessao.idSessao = Bilhete.idSessao  
JOIN Filme ON Filme.idFilme = Sessao.idFilme  
WHERE Filme.titulo = 'The House By The Sea';
```

-- Tipo de sala que projeta um determinado filme

```
SELECT Sala.tipoSala  
FROM Bilhete  
JOIN Sessao ON Bilhete.idSessao = Sessao.idSessao  
JOIN Filme ON Sessao.idFilme = Filme.idFilme  
JOIN Sala ON Filme.idSala = Sala.idSala  
WHERE Filme.idFilme = 1;
```

-- Durante um intervalo, qual o top 10 de filmes com o menor número de bilhetes
vendidos -----

```
SELECT Filme.titulo, Filme.idFilme, COUNT(*)
FROM Bilhete
JOIN Sessao ON Bilhete.idSessao = Sessao.idSessao
JOIN Filme ON Sessao.idFilme = Filme.idFilme
WHERE Bilhete.dataSessao BETWEEN '2010-04-01' AND '2017-10-31'
GROUP BY Bilhete.idSessao
ORDER BY COUNT(*)
LIMIT 10;
```

-- Filmes de uma sala

```
SELECT Filme.titulo, Filme.duracao, Filme.idadeMinima, Filme.dataEstreia
FROM Sala
JOIN Filme ON Filme.idSala = Sala.idSala
WHERE Sala.idSala = 1;
```

-- Filmes de um cliente

```
SELECT DISTINCT Filme.titulo
FROM Cliente
JOIN Bilhete ON Cliente.idCliente = Bilhete.idCliente
JOIN Sessao ON Bilhete.idSessao = Sessao.idSessao
JOIN Filme ON Filme.idFilme = Sessao.idFilme
WHERE Cliente.username = 'joaoS';
```

-- Nomes dos clientes de um determinado filme

```
SELECT Cliente.nome
FROM Cliente
JOIN Bilhete ON Cliente.idCliente = Bilhete.idCliente
JOIN Sessao ON Sessao.idSessao = Bilhete.idSessao
JOIN Filme ON Filme.idFilme = Sessao.idFilme
WHERE Filme.idFilme = 1;
```